Université Paris 7 - Denis Diderot - UFR d'informatique - Année 2002

Généralisation des Jeux Combinatoires et Applications aux Langages Logiques

Thèse
pour l'obtention du diplôme de
docteur de l'université Paris 7, spécialité informatique
présentée et soutenue publiquement par

Jean-Vincent Loddo

le 16 décembre 2002

Directeur de thèse
Roberto Di Cosmo

Jury

M. Guy Cousineau	*président*
M. François Fages	*rapporteur*
M. Furio Honsell	*rapporteur*
M. Roberto Di Cosmo	*directeur de thèse*
M. Pierpaolo Degano	
M. Giorgio Levi	
Mlle Francesca Scozzari	

Remerciements

J'adresse mes remerciements, en premier lieu, à Roberto Di Cosmo pour avoir accepté d'encadrer ma thèse et pour tout le soutien qu'il m'a offert durant cette longue mais très enrichissante entreprise. J'ai beaucoup appris de cette expérience et je lui en suis reconnaissant.

Malgré leur charge académique, François Fages et Furio Honsell ont accepté d'être les rapporteurs de ma thèse. Je les remercie pour l'intérêt qu'ils ont ainsi manifesté envers mon travail de recherche et pour les suggestions, fort utiles, qui ont contribué au perfectionnement de ce mémoire. Je tiens à remercier Guy Cousineau non seulement pour avoir accepté d'être président de mon jury de thèse, mais aussi pour sa gentillesse rassurante et constante au fil des années, que j'observe avec admiration depuis mon inscription en DEA. Pour la même gentillesse et pour les conseils exprimés en toute simplicité et générosité, je remercie Jean-Louis Krivine et Pierre-Louis Curien. Je tiens aussi à remercier Pierre-Louis de m'avoir accueilli en septembre 1999 dans le tout jeune laboratoire *Preuves Programmes et Systèmes* (PPS) de Paris 7, après mon passage au laboratoire d'informatique de l'École normale supérieure.

Parmi les personnes qui sont venues à mon secours à plusieurs reprises durant la rédaction de la thèse, je voudrais citer Antonio Bucciarelli, Vincent Danos et Francesca Scozzari. Dans les périodes de difficulté, leur soutien moral et scientifique à été formidable, ma dette envers eux est donc immense. Merci aussi à Olivier Laurent, à Antonino Salibra et à Pierre-Louis Curien, encore une fois, pour l'attention avec laquelle ils ont bien voulu répondre à mes questions.

Ces années de thèse m'ont fait apprécier encore plus ma formation à la faculté d'informatique de l'université de Pise. Je suis reconnaissant aux personnes qui ont contribué à me donner cette formation, en particulier Pierpaolo Degano, pour ses qualités humaines et didactiques, et Giorgio Levi, dont le cours de *Linguaggi Speciali di Programmazione*, traitant de programmation logique, a de toute évidence laissé plus qu'une trace en moi.

Je remercie collectivement tous les auteurs des logiciels libres, qui mettent à disposition des chercheurs une panoplie d'outils indispensables à l'activité de recherche en général et de rédaction en particulier. Sans les incontournables Gnu/Linux, TEX, LATEX et LYX, je suis certain que la rédaction de ce mémoire aurait été bien plus pénible.

Je tiens à remercier Dominique Raharijésy, Joëlle Isnard, Noëlle Delgado et Odile Ainardi pour leur appui dans les tâches administratives de la vie de recherche, et pour la patience à l'égard de quelqu'un aussi maladroit que moi dans ce type de démarche. Merci à mes amis Stefano Chessa, pour l'aide lors de mon dernier séjour à Pisa, à Pierangelo Veltri pour l'attention et les conseils constamment prodigués, et à Viviane Gourmelon pour avoir accepté de réviser le mémoire. Merci à Francisco Alberti, Vincent Balat, Sylvain Baro, François Maurel, Emmanuel Polonovsky, et les "petits" Michel et Raphaël, pour leur sympathie et leur présence agréable au sein du laboratoire. Aux nouveaux thésards de PPS je souhaite une aussi bonne entente et fructueuse collaboration.

Enfin, merci à mes parents de m'avoir encouragé et d'avoir tout fait pour que je commence cette aventure. À Lelia, mon épouse, merci de m'avoir soutenu chaque jour de cette longue traversée. Je n'ai jamais été seul dans l'effort et dans les jours difficiles. C'est bien *notre* thèse, parce-que nous l'avons menée à bien ensemble. Merci aussi à ses parents, Rodica et Iulian, pour leur participation et leur touchant soutien moral.

Résumé

La théorie des jeux a développé, à ses débuts, une vocation pour les sciences sociales et économiques, avec des applications disparates, comme par exemple le traitement de données médicales. Elle apparaît aujourd'hui comme un paradigme de concepts et de techniques très général, dont le potentiel reste encore à exploiter en informatique. Dans cette thèse nous étudions une branche particulière, la théorie des jeux combinatoires (à deux joueurs), pour en tirer bénéfice dans le domaine, très actif, des sémantiques formelles des langages de programmation.

D'un jeu, nous pouvons séparer l'aspect syntaxique, inhérent aux dénouements possibles des matchs, de l'aspect sémantique, inhérent aux prévisions sur le gagnant et la quantification de son gain (en termes d'un enjeu quelconque, tel que l'argent ou le préstige). Pour modeliser la notion de gain, la structure d'évaluation choisie ne doit pas forcément être celle des booléens (*gagné* ou *perdu*), ou celle des entiers naturels ou relatifs. Il suffit qu'elle vérifie des propriétés, assez faibles, garantissant l'existence d'une sémantique même lorsque le jeu donne lieu à des matchs infinis, comme dans le cas du jeu de la *bisimulation* entre processus concurrents, et du jeu de la *programmation logique*.

Dans ce travail, nous étudions la caractérisation sémantique d'un langage logique (avec ou sans contrainte) en termes de jeu à deux joueurs. Au-delà du modèle intuitif des jeux, dont la valeur pédagogique mériterait d'être approfondie, une telle interprétation permet de réutiliser un des algorithmes les plus utilisés dans la théorie des jeux combinatoire, *Alpha-Bêta*, comme moteur de résolution pour les langages logiques. Les résultats récents et spectaculaires obtenus par les programmes d'échecs (souvenons-nous de la défaite du champion du monde Kasparov contre le programme Deep Blue d'IBM) témoignent d'une forme d'intelligence artificielle développée dans ces programmes qui peut être transposée et exploitée dans la résolution des langages logiques. La résolution d'interrogations existentielles conjonctives dans une théorie de clauses de Horn du premier ordre est en particulier concernée. En effet, la capacité d'Alpha-Bêta à simplifier le calcul ou, en d'autres termes, sa capacité à éliminer les coups inintéressants, n'est pas intimement liée à un type de jeu ou à un type de gain particuliers, mais demande juste des propriétés algébriques qui sont satisfaites dans le jeu de la programmation logique. La correction d'Alpha-Bêta est prouvée de façon formelle pour un éventail de structures très large. Les valeurs calculées pourront être aussi bien des entiers naturels, comme dans le cas du jeu d'échecs, que des substitutions ou des contraintes, comme dans le cas des langages logiques.

Mots-clés

Algèbres typées, arborescences, arbres, termes, théorie des jeux combinatoires, jeux, arènes, positions, joueurs, gain, arbres de jeu, préfixes de jeu, stratégies, structures d'évaluation, sémantique monosémique, structures de co-évaluation, sémantique bisémique, sémantique pessimiste, sémantique optimiste, algorithme Alpha-Bêta, théorème Alpha-Bêta polymorphe, algorithme Alpha-Bêta-Gamma-Delta, Programmation Logique, LP, Programmation Logique avec contraintes, CLP, résolution SLD, résolution CSLD.

Abstract

Game theory had, in its origins, a vocation for social and economic sciences, with disparate applications, for example in the processing of medical data. It is perceived today as a very general paradigm of concepts and techniques, whose potential still remains to be discovered in computer science. In this thesis we study a particular branch, combinatorial game theory (with two players), in order to profit from it in the very active field of formal semantics of programming languages.

Within a game, we can separate the syntactic aspect, inherent in the possible outcomes of matches, from the semantic aspect, inherent in the forecasts over the winning player and, possibly, inherent in a quantification of profit, in terms of a stake, like wealth or prestige. To model the concept of profit, the selected structure of evaluation should not necessarily be the structure of booleans (*win* or *loose*), or that of natural or relative numbers. It is enough for this structure to verify some properties, rather weak, guaranteeing the existence of a semantics, even when the play gives rise to infinite matches, as in the game of *bisimulation* between concurrent processes, or as in the game of *logic programming*.

In this work, we study the semantic characterization of a logical language (with or without constraints) in terms of a game with two players. The effect of such an interpretation, beyond the intuitive model whose didactic value would deserve to be examined, makes it possible to reuse *Alpha-Beta,* one of the most celebrated algorithms in combinatorial game theory, as a resolution engine for logical languages. The recent and spectacular results obtained by chess programs (like the defeat of the world champion Kasparov against the IBM program Deep Blue) testify to a kind of artificial intelligence acquired by these programs that might be transposed and exploited in the resolution of logical languages. The resolution of existential conjunctive goals in a first-order theory of Horn clauses provides an interesting case. Indeed, the ability of Alpha-Beta to simplify calculation or, in other words, to prune the uninteresting paths, is not closely related to a particular type of game or profit, but to a well-chosen set of algebraic properties, satisfied by the game of logical programming. The correction of Alpha-Beta is formally proven for a very wide variety of structures. In this way, the computed values could be natural numbers, as in the case of chess, or substitutions or constraints, as in the case of logical languages.

Keywords

Typed algebras, tree structures, trees, terms, combinatorial game theory, game, arena, position, players, value, game tree, game prefix, strategies, evaluation structure, monosemic value, co-evaluation structure, bisemic value, pessimistic semantics, optimistic semantics, Alpha-Beta algorithm, polymorphic Alpha-Beta theorem, Alpha-Beta-Gamma-Delta algorithm, Logical Programming, LP, Logical Programming with constraints, CLP, SLD resolution, CSLD resolution.

Table des matières

Introduction

Les interprétations en termes de jeux ont trouvé, depuis quelques années, des applications de plus en plus nombreuses en informatique, en particulier dans l'étude des sémantiques des langages de programmation. Les applications les plus remarquables de la théorie des jeux concernent l'introduction systématique du hasard en algorithmique (cf. [**Motwani & Raghavan, 1995**] pour une introduction éloquente) et, bien entendu, les programmes qui "jouent" à des jeux de stratégies.

Le hasard et l'adaptation en informatique

L'idée des algorithmes aléatoires est celle de considérer le programmeur comme un joueur ayant à sa disposition plusieurs algorithmes déterministes corrects (ses stratégies *pures*), et s'opposant à un adversaire, l'environnement du programme, dont les stratégies sont les valeurs possibles (ou les classes de valeurs) pour les entrées du programme. L'enjeu sera le temps, la qualité de la solution, l'espace mémoire occupé ou tout autre mesure d'efficacité du programme. Alors, une stratégie optimale (appelée *mixte*) correspondra à définir un algorithme choisissant dynamiquement, de façon aléatoire et selon une loi de probabilité établie, parmi les méthodes déterministes à sa disposition. Par exemple, toutes les variantes du fameux algorithme de tri *quicksort* qui choisissent de façon déterministe la position du pivot sont mises à mal (complexité quadratique) par certaines permutations du vecteur d'entrée. En revanche, le quicksort aléatoire "inventé" à la fin des années '70 (cf. [**Sedgewick, 1978**]), correspond exactement à la stratégie mixte naïve (et loin d'être optimale), qui donne la même probabilité d'utilisation à tous les quicksort déterministes, correspondant aux choix fixes de la position du pivot. On a évidemment l'impression que cette stratégie naïve est optimale si toutes les permutations du vecteur à trier sont équiprobables. Mais la théorie des jeux permet de définir *a priori* des algorithmes optimaux adaptés à toute distribution connue de probabilité sur les données, voire, grâce à des méthodes d'apprentissage, de concevoir des programmes qui s'adaptent (pendant l'exécution) aux caprices de l'environnement (cf. par exemple [**Owen, 1968**]).

L'idée des méthodes qui se spécialisent avec l'expérience n'est pas récente. Elle évoque au contraire un des premiers propos, au début du vingtième siècle, de la théorie des jeux : la modélisation des rapports économiques, de l'*offre* et la *demande*, c'est-à-dire la modélisation des deux acteurs ayant des intérêts opposés mais convergent vers une adaptation réciproque. En informatique, ce genre d'approche, où l'on fait "combattre" une spécification contre les données qui seront soumises à son implantation, est bien entendu envisageable pour n'importe quelle collection de programmes équivalents, mais plus ou moins performants suivant l'environnement (cf. entre autres [**Welsh, 1983**], [**Karp & Rabin, 1987**], [**Karp, 1991**], [**Motwani & Raghavan, 1995**]).

Les robots joueurs

Nous pouvons rappeler ici les succès récents et spectaculaires obtenus, dans le domaine de l'intelligence artificielle, par les programmes qui "jouent" aux Dames, aux Échecs, à Othello ou à d'autres jeux de stratégie classiques. La victoire du programme Deep Blue de IBM contre le champion du monde des Échecs, Garry Kasparov, en mai 1997, a marqué l'aboutissement de quarante années de recherche, tant sur le plan algorithmique que de la mise au point d'architectures spécialisées. Cette victoire a, en quelque sorte, revivifié les espoirs immenses soulevés par l'intelligence artificielle dans les dernières décennies, même si, à notre avis, elle a seulement démontré que les jeux sont relativement faciles à modéliser (puisque les machines y atteignent l'excellence), et non pas que les machines soient devenues intelligentes.

La sémantique des langages de programmation

L'étude des sémantiques des langages de programmation a, elle aussi, profité des concepts à la fois puissants et intuitifs déployés par la théorie des jeux, en particulier de la théorie des jeux *combinatoires*. En effet, après les travaux liminaires de Lamarche (cf. [**Lamarche, 1995**]), Blass (cf. [**Blass, 1992**]) et Joyal (cf. [**Joyal, 1995**]) qui, au début des années 90, démontrent qu'une certaine catégorie de stratégies était cartésienne fermée, et donc représentait un modèle du λ-calcul, les recherches d'Abramsky, Malacaria et Jagadeesan (cf. [**Abramsky & Jagad., 1994**] entre autres), ont permis de donner les premières sémantiques complètement abstraites pour des langages fonctionnels (PCF) ou impératifs (Idealized Algol). Enfin, plus récemment, des spécialistes de la logique linéaire comme Baillot, Danos, Ehrhard, Harmer et Regnier (cf. [**Danos & Harmer, 2000**] et [**Baillot et al., 1997**]), entre autres, ont travaillé sur les liens des jeux avec la géométrie de l'interaction, tandis que Curien et Herbelin (cf. [**Curien & Herbelin, 1998**]) ont montré que certaines machines abstraites classiques pouvaient être interprétées comme des jeux.

Si les travaux évoqués ci-dessus utilisent plus le vocabulaire des jeux (*joueur*, *coup*, *partie*, *arène*, *stratégie* etc.) que l'intuition ou les résultats propres à la théorie des jeux traditionnelle, en revanche, d'autres travaux, où l'on retrouve des concepts clefs comme ceux de *joueurs rationnels* et de *gain*, adoptent l'approche des jeux d'une façon plus importante et accomplie. Par exemple, en théorie des automates, où l'accent est mis sur la spécification et la vérification de programmes, certains problèmes de modélisation de langages infinis sont abordés avec les jeux (cf. par exemple [**Perrin & Pin, 2001**]). Aussi, dans le cadre de la programmation concurrente, les différentes relations de *bisimulation* entre processus, introduites par une technique de définition particulière, nécessitant la notion de plus grand point fixe, peuvent être réinterprétées, d'une façon bien plus intuitive, comme des jeux à deux joueurs : si un joueur est gagnant, les processus sont bisimilaires, s'il est perdant, les processus ne sont pas bisimilaires (cf. [**Stirling, 1997**] et [**Loddo & Nicolet, 1998**]).

La programmation logique peut, elle aussi, être abordée par les jeux combinatoires (cf. [**Loddo & Nicolet, 1998**] et [**Di Cosmo et al.,1998**]). On peut, dans un certain sens, affirmer qu'elle peut être complètement réexpliquée en termes de jeu. En effet, deux joueurs s'opposent : chaque but à satisfaire est la position initiale d'une partie dans laquelle l'un des joueurs veut démontrer que le but est satisfaisable dans le programme logique (faisant office de damier), tandis que l'autre cherche à l'en empêcher. Nous verrons que les réponses correspondantes à un but dans un programme logique coïncident précisément avec le gain du joueur dans la position définie par le couple but et programme.

L'objectif principal de cette thèse est, en premier lieu, de présenter, de façon rigoureuse, tous les aspects *syntaxiques* et *sémantiques* des jeux combinatoires, pour ensuite appliquer au paradigme de la programmation logique les outils formels développés. Nous adapterons des notions traditionnelles, telles que l'*heuristique* ou la *valeur* d'un jeu, au cadre des jeux *infinis*, dans l'intention de construire une théorie capable de décrire à la fois les jeux habituels, tels que les Échecs ou les Dames, et à la fois les jeux, inattendus, de la programmation logique. La souplesse du paradigme des jeux et son aptitude à la modélisation, nous ont encouragé à construire une théorie générale, nouvelle à notre connaissance, avec l'espoir de la réutiliser un jour pour d'autres applications.

Organisation des chapitres

Chapitre 1. Le premier chapitre est un travail de synthèse de notions générales d'algèbre universelle, telles que la *signature*, l'*homomorphisme* ou l'*isomorphisme*, avec les notions propres de la théorie du λ-calcul simplement typé, telles que *substitution*, *polymorphisme* ou *relation logique*. Le but de ce chapitre est de fournir un métalangage formel pour aborder les sujets des chapitres suivants, notamment la théorie des domaines (chapitre 2), l'algèbre des arbres, des arborescences et des termes (chapitres 3), l'algèbre syntaxique d'un jeu (chapitre 4), et l'homomorphisme sémantique des jeux (chapitre 5).

Chapitre 2. Le deuxième chapitre parcourt la théorie des *ordres partiels*, avec leurs propriétés de complétude et leurs homomorphismes et isomorphismes, fournissant le support formel utile, d'une part, à la définition des relations d'ordre, *élagage* et *préfixe*, sur les arbres (chapitre 3), et, d'autre part, à la définition des domaines d'évaluation d'un jeu (chapitre 5).

Chapitre 3. Le troisième chapitre introduit l'algèbre des arborescences et celle des arbres, avec la panoplie d'opérations nécessaires à leur manipulation, et l'ensemble des relations qui en font des ordres partiels. Les propriétés de complétude de ces ordres partiels seront abordées et étudiées en détail. Ces algèbres fourniront les outils nécessaires à la définition des notions du chapitre suivant, consacré aux jeux combinatoires, telles que la notion *d'arbre de jeu* ou celle de *préfixe de jeu*.

Chapitre 4. Le but du quatrième chapitre est de développer l'aspect syntaxique des jeux combinatoires à deux joueurs. On y trouve notamment la définition formelle d'*arène de jeu*, d'*arbres de jeu*, de *stratégie* et *contre-stratégie*, et d'*alternance* des joueurs.

Chapitre 5. Dans le cinquième chapitre, nous découvrons le sens des jeux définis par une arène, c'est-à-dire par une syntaxe de jeu. La notion cruciale de *structure d'évaluation* est introduite, et un homomorphisme est tracé de l'algèbre syntaxique (définie dans le chapitre précédent) vers l'algèbre sémantique d'un jeu. Cela conduit à deux types de sémantique pour les positions d'un jeu. La première sémantique, appelée *monosémique*, affecte une seule valeur à chaque position du jeu, grâce à une fonction *heuristique* ayant des propriétés adéquates vis-à-vis du jeu. La deuxième, appelée *bisémique*, affecte deux valeurs, c'est-à-dire un intervalle, à la même position du jeu, grâce, cette fois ci, à deux fonctions heuristiques duales, l'une représentant une approche d'évaluation *pessimiste* et l'autre représentant une approche, au contraire, *optimiste*. Nous verrons aussi que, dans certains cas, les deux approches convergent vers une seule valeur sémantique qui ne tient plus à l'attitude, pessimiste ou optimiste, adoptée, mais qui représente une sorte de sémantique absolue.

Pour terminer ce chapitre, nous développerons la technique d'évaluation des *stratégies* (du joueur) et des *contre-stratégies* (de l'opposant), qui deviennent, dans notre cadre théorique, des objets *infinis*, tout comme les arbres des positions du jeu.

Chapitre 6. Le sixième chapitre explore, d'une part, les capacités du fameux algorithme Alpha-Bêta, conçu pour les structures d'évaluation traditionnelles (les entiers naturels), de s'adapter à d'autres structures d'évaluation comme celle des jeux de la programmation logique. Et, d'autre part, il explore le caractère sémantique, pessimiste ou optimiste, des réponses fournies par cet algorithme.

L'algorithme Alpha-Bêta est avant tout généralisé, faisant abstraction de la structure d'évaluation utilisée, et ensuite reformulé dans le style plus formel des sémantiques opérationnelles *à grands pas* (*big steps*). Un théorème de correction précise, par la suite, la qualité des réponses fournies. Par rapport aux travaux présentés dans [**Di Cosmo & Loddo, 2000**], ce résultat élargit le spectre d'application aux jeux où les joueurs n'alternent pas forcément, autrement dit, les jeux où les joueurs ne laissent pas automatiquement, après chaque coup, la main à leur adversaire. Cela reflète l'intention d'appliquer l'algorithme non seulement aux jeux de la programmation logique (pure ou avec contraintes), mais aussi au jeu de la programmation logique avec *négation*, qui, cependant, ne sera pas étudié dans cette thèse.

Aussi, nous avons introduit un nouvel algorithme, que nous avons nommé Alpha-Bêta-Gamma-Delta, et un second théorème de correction afin d'optimiser le calcul des jeux évalués dans une structure *non distributive*, où Alpha-Bêta n'est plus correct. Le nom choisi évoque une certaine ressemblance des principes de simplifications utilisés par Alpha-Bêta, mais il est important de spécifier qu'il s'agit d'un algorithme différent, plus prudent dans l'élagage des arbres de jeu, qui évite ainsi les pièges d'une structure d'évaluation non distributive. Il pourrait se rendre utile, mais nous ne l'étudierons pas dans cette thèse, pour l'*interprétation abstraite* des programmes logiques, où l'abstraction accélère et force la fin du calcul, au prix d'une certaine imprécision des réponses. Les opérations sémantiques “abstraites” ont alors, par effet de l'imprécision tolérée, de moins bonnes propriétés algébriques, et peuvent perdre, notamment, la propriété distributive nécessaire à l'application de la méthode Alpha-Bêta.

Chapitre 8. Dans ce chapitre se trouve un résumé de la théorie des langages du premier ordre à la base du paradigme de la programmation logique avec ou sans contraintes. La présentation de cette théorie tire parti des notions introduites dans le premier chapitre, comme la notion d'algèbre typée et d'homomorphisme, et dans le troisième chapitre, comme la notion de terme, de substitution et d'environnement.

Chapitre 9. Le dernier chapitre est consacré à la programmation logique. Les définitions, les résultats démontrés et les méthodes énoncées au cours des chapitres précédents sont mis au service d'une définition formelle de la programmation logique en termes de jeux combinatoires à deux joueurs. Nous prouvons l'équivalence entre la sémantique traditionnelle (des *réponses calculées*) et la sémantique des jeux, puis nous prouvons également que la structure d'évaluation utilisée vérifie les hypothèses d'application du théorème de correction Alpha-Bêta du chapitre précédent. Nous pouvons résumer ces deux résultats et leur conséquence directe par un syllogisme :

> *la sémantique d'un but dans un programme logique coïncide avec la valeur sémantique d'un jeu à deux joueurs*

et

> *la méthode Alpha-Bêta permet, par des optimisations correctes, de calculer la valeur d'un jeu à deux joueurs*

donc

> *la méthode Alpha-Bêta permet de résoudre un but dans un programme logique, en calculant, par des simplifications correctes, l'ensemble des réponses qui font du but une conséquence logique du programme*

Nous verrons quelques exemples d'exécution de l'algorithme et nous prospecterons son utilisation comme moteur de résolution en programmation logique. À l'instar de la recherche d'une stratégie gagnante aux échecs, le moteur de résolution Alpha-Bêta apportera son "intelligence" au secours des performances. À la manière des fameux algorithmes *branch&bound* de la recherche opérationnelle, il sera lui-même capable d'élaguer les parcours inintéressants dans l'arbre des dénouements possibles du jeu.

Première partie

Préliminaires

CHAPITRE 1

Algèbres

Dans ce chapitre, aucune nouvelle notion est introduite. Il s'agit cependant d'un travail de synthèse de théories différentes, telles que la théorie de l'algèbre universelle et celle du λ-calcul simplement typé. Nous verrons comme la notion fondamentale d'homomorphisme, présente à plusieurs reprises et sous plusieurs formes dans la première théorie, peut être généralisée aux opérations d'un type appartenant à un langage des types évolué. Nous prouverons la spécificité des définitions habituelles par rapport à la nouvelle et plus générale notion d'homomorphisme. Ce métalangage algébrique nous permettra d'expliquer de façon avantageuse les algèbres que nous rencontrerons au cours des chapitres suivants.

D'une façon informelle, les algèbres sont des structures composées de deux genres d'objet mathématique : des ensembles de base, les *domaines*, et des constructions, les *opérations*, définies sur ces domaines, comme par exemple des relations ou des fonctions. Si en théorie des ensembles, où tout objet est un ensemble, la distinction entre domaines et opérations (sur les domaines) n'est pas cardinale, avec les algèbres la différence est explicite. D'une part, la notion de domaine est celle, naturelle, d'ensemble. D'autre part, à toute opération, qui est elle-même un ensemble, nous affecterons un *type* (ou une *signature*) représentant une construction mathématique : le domaine résultant de cette construction, contiendra l'opération. L'affectation d'un type à une opération traduit toujours la volonté d'encadrer l'opération dans un domaine qui la cernera d'une façon plus ou moins précise.

1.1. Relations, fonctions et propriétés

1.1.1. Ensembles de base. Dans cette thèse, nous utiliserons les ensembles de base suivants : un ensemble constitué par un seul élément $\{\bullet\}$, noté $\mathbb{1}$, un ensemble constitué par deux éléments $\{0, 1\}$ (les *booléens*) noté $\mathbb{2}$, les entiers naturels $\mathbb{N}$, les entiers naturels non nuls $\mathbb{N}_+$, et les entiers naturels avec le symbole dénotant l'infiniment grand $\mathbb{N}_\infty \triangleq \mathbb{N} \cup \{+\infty\}$.

À partir des ensembles de base, nous utiliserons, pour générer de nouveaux ensembles, les constructions mathématiques habituelles telles que le produit cartésien, l'union disjointe, les parties d'un ensemble, les relations, les fonctions, les fonctions partielles et les suites finies ou infinies.

1.1.2. Produits cartésiens, sommes. Étant donné deux ensembles A et B, nous noterons par $A \times B$ leur produit cartésien. Si leur intersection est vide $A \cap B = \emptyset$, nous noterons $A \oplus B$ leur union disjointe (ou *somme*) et nous dirons que A et B sont les *composantes* de la somme.

1.1.3. Relations unaires. Étant donné un ensemble D, une *partie* $\mathfrak{R}$ de D (ou *relation unaire* $\mathfrak{R}$ sur D), notée $\mathfrak{R} : \wp^{D}$ ou $\mathfrak{R} : \wp(D)$, est un sous-ensemble quelconque de D. La partie est *finie*, notée $\mathfrak{R} : \wp_{\mathrm{F}}(D)$, si elle est constituée d'un nombre fini d'éléments.

1.1.4. Relations binaires. Étant donné un ensemble D, le *carré de* D est le produit cartésien $D \times D$. Étant donné deux ensembles A et B, une *relation binaire* $\mathfrak{R}$ *sur* A *et* B est un sous-ensemble arbitraire du produit cartésien $A \times B$, donc notée $\mathfrak{R} : \wp^{A \times B}$; quand $A = B \triangleq D$, nous disons que $\mathfrak{R}$ est une *relation (binaire) sur* D. La notation *infixe* $x \mathfrak{R} y$ signifie $(x, y) \in \mathfrak{R}$. Le *domaine* de $\mathfrak{R}$ est le sous-ensemble de A : $\mathrm{dom}(\mathfrak{R}) = \{a \in A \mid \exists b.\, a\mathfrak{R}b\}$, le *codomaine* est le sous-ensemble de B : $\mathrm{cod}(\mathfrak{R}) = \{b \in B \mid \exists a :\ a\mathfrak{R}b\}$. La *restriction de* $\mathfrak{R}$ *à un sous-domaine* $A' \subseteq A$ est la relation $\mathfrak{R}_{|_{A' \times B}} \triangleq \{(a, b) \in \mathfrak{R} \mid a \in A'\} = \mathfrak{R} \cap (A' \times B)$. La *restriction de* $\mathfrak{R}$ *à un sous-codomaine* $B' \subseteq B$ est la relation $\mathfrak{R}_{|_{A \times B'}} \triangleq \{(a, b) \in \mathfrak{R} \mid b \in B'\} = \mathfrak{R} \cap (A \times B')$. L'*image* de $\mathfrak{R}$ sur un sous-domaine $A' \subseteq A$ est le codomaine de la restriction de $\mathfrak{R}$ au sous-domaine A' : $\mathrm{Im}_{\mathfrak{R}} A' \triangleq \{b \in B \mid (a, b) \in \mathfrak{R}_{|_{A' \times B}}\}$. Dans le cas où $A = B \triangleq D$, la *restriction de* $\mathfrak{R}$ *à un sous-domaine* $D' \subseteq D$ est la relation $\mathfrak{R}_{|_{D'}} \triangleq \{(a, b) \in \mathfrak{R} \mid a \in D',\ b \in D'\} = \mathfrak{R} \cap (D' \times D')$. Étant donné deux relations binaires $\mathfrak{R}_1 : \wp^{A \times B}$ et $\mathfrak{R}_2 : \wp^{B \times C}$, la *composition* $\mathfrak{R}_2 \circ \mathfrak{R}_1 \triangleq \{(a, c) \in A \times C \mid \exists b \in B.\ (a, b) \in \mathfrak{R}_1 \wedge (b, c) \in \mathfrak{R}_2\}$ est une relation binaire $\mathfrak{R}_2 \circ \mathfrak{R}_1 : \wp^{A \times C}$. La *puissance* n d'une relation $\mathfrak{R} : \wp^{D \times D}$, où $n \in \mathbb{N}$, $n \geq 1$, est la relation $\mathfrak{R}^1 \triangleq \mathfrak{R}$ si $n = 1$, ou la relation $\mathfrak{R}^n \triangleq \mathfrak{R} \circ \mathfrak{R}^{n-1}$ si $n > 1$. La *clôture transitive* de $\mathfrak{R}$ est l'union de toutes ses puissances : $\mathfrak{R}^* \triangleq \bigcup_{n \geq 1} \mathfrak{R}^n$.

1.1.5. Fonctions et fonctions partielles. Une *fonction partielle de* A *vers* B, est une relation binaire $\mathfrak{R}$ sur A et B, notée $\mathfrak{R} : A \mathrel{\circ\!\!\!\!\!\longrightarrow} B$, telle que $(a,b) \in \mathfrak{R} \wedge (a,c) \in \mathfrak{R} \;\Rightarrow\; b = c$. Pour les fonctions partielles nous utiliserons la notation $\exists\,\mathfrak{R}x$ pour dire $\exists y \in B.\,(x,y) \in \mathfrak{R}$. Puisque l'existence implique l'unicité, nous utiliserons la notation $\exists\,\mathfrak{R}x \triangleq y$ pour indiquer l'existence de l'élément en relation avec x et, en même temps, le baptiser y. En revanche, quand l'existence ne sera pas assurée, nous noterons $y \stackrel{\leftharpoonup}{=} \mathfrak{R}(x)$ pour signifier *égale s'il existe*, autrement dit, si $\exists\,\mathfrak{R}x$ alors $\exists\,\mathfrak{R}x \triangleq y$. Une fonction partielle $\mathfrak{R} : A \mathrel{\circ\!\!\!\!\!\longrightarrow} B$ devient une fonction à plein titre, on dit aussi *totale*, notée $\mathfrak{R} : A \to B$, quand $\mathrm{dom}(\mathfrak{R}) = A$. La composition de deux fonctions partielles $f : A \mathrel{\circ\!\!\!\!\!\longrightarrow} B$ et $g : B \mathrel{\circ\!\!\!\!\!\longrightarrow} C$ est, elle aussi, une fonction partielle $g \circ f : A \mathrel{\circ\!\!\!\!\!\longrightarrow} C$; la composition de deux fonctions totales est, elle aussi, totale $g \circ f : A \to C$. La puissance f^n d'une fonction partielle $f : A \mathrel{\circ\!\!\!\!\!\longrightarrow} A$ (resp. totale $f : A \to A$) est, elle aussi, une fonction partielle $f^n : A \mathrel{\circ\!\!\!\!\!\longrightarrow} A$ (resp. totale $f^n : A \to A$). Étant donné un ensemble D, toute fonction $\mathfrak{R} : 1 \to D$ est une *constante,* que nous noterons aussi $\mathfrak{R} :\to D$; l'écriture $\mathfrak{R}() = x$ remplace souvent l'expression $\mathfrak{R}(\bullet) = x$. De la même manière, toute fonction partielle $\mathfrak{R} : 1 \mathrel{\circ\!\!\!\!\!\longrightarrow} D$ est une *constante partielle,* que nous noterons aussi $\mathfrak{R} : \mathrel{\circ\!\!\!\!\!\longrightarrow} D$, et l'écriture $\mathfrak{R}() = x$ pourra remplacer l'expression $(\bullet, x) \in \mathfrak{R}$.

1.1.6. Prédicats, propriétés. L'*identité sur* X, définie par $\{(x,x) \mid x \in X\}$, est à la fois une fonction totale $\mathtt{id}_X : X \to X$ et partielle $\mathtt{id}_X : A \mathrel{\circ\!\!\!\!\!\longrightarrow} A$ pour tout sur-ensemble $A \supseteq X$. Toute fonction de type $p : D \to 2$ est un *prédicat* que nous noterons aussi $p : 2^D$. Toute fonction partielle de type $p : D \mathrel{\circ\!\!\!\!\!\longrightarrow} 1$ est un *semi-prédicat*[1] (ou *prédicat partiel*), noté aussi $p : 1^D$. Nous utiliserons le terme générique *propriété* (*sur* D) pour dénoter aussi bien une partie (relation unaire) $p : \wp^D$, qu'un prédicat $p : 2^D$ ou qu'un semi-prédicat $p : 1^D$. Les deux derniers représentant eux aussi, bien que implicitement, une partie (respectivement la partie $\{x \in D \mid p(x) = 1\}$ ou la partie $\mathrm{dom}(p)$), la notation $\hat{p}$ sera utilisée pour signifier la partie concernée. En d'autres termes, tout prédicat $p : 2^D$ ou semi-prédicat $p : 1^D$ pourra assumer, à l'occasion, le statut d'une partie $\hat{p} : \wp^D$. Dans l'esprit d'unifier la notation concernant les propriétés, l'écriture $p(x)$ sera utilisée pour exprimer la condition d'appartenance $x \in \hat{p}$, c'est-à-dire la condition $x \in p$, lorsque $p : \wp^D$, ou bien la condition $p(x) = 1$, lorsque $p : 2^D$, ou bien la condition $(x, \bullet) \in p$, lorsque $p : 1^D$. Étant donné une propriété $p : \wp^A \oplus 2^A \oplus 1^A$, où $A \neq 1$ et $A \neq 2$, et une fonction totale $f : A \to B$, ou partielle $f : A \mathrel{\circ\!\!\!\!\!\longrightarrow} B$, la *composition de* f *et* p, notée f_p, est la restriction de f au sous-domaine $\hat{p} \subseteq A$ dénoté par la propriété : $f_p \triangleq f_{|\hat{p}} = \{(a,b) \in f \mid p(a)\}$. La composition d'une fonction f avec une propriété p est toujours une fonction partielle $f_p : A \mathrel{\circ\!\!\!\!\!\longrightarrow} B$. Les conditions $A \neq 1$ et $A \neq 2$ évitent la possibilité que la fonction f soit compatible avec la propriété p pour une composition fonctionnelle classique (en effet, si $p : A \mathrel{\circ\!\!\!\!\!\longrightarrow} 1$ et le domaine de f était $A = 1$, ou bien si $p : A \to 2$ et le domaine de f était $A = 2$, alors la composition classique $f \circ p : A \mathrel{\circ\!\!\!\!\!\longrightarrow} B$ serait réalisable). Ainsi, à l'abri de toute ambiguïté sur l'opération dénotée, le nom *composition* se justifie en remarquant que la même construction peut être obtenue en composant

[1] Dans la programmation Lisp (et des dialectes comme Scheme), un *semi-prédicat* est une fonction qui renvoie *faux*, ou une autre valeur quelconque, différente de faux, qui sera interprétée comme vraie. Ici l'utilisation du terme n'est pas la même mais elle est similaire : le semi-prédicat indiquera avec exactitude une seule condition (que la propriété est vérifiée), non pas sa négation (comme dans le cas des *prédicats* à part entière).

(au sens de la composition fonctionnelle) la fonction f avec l'identité sur le signifié ensembliste de la propriété : $f_p = f \circ id_{\hat{p}}$.

1.1.7. Relations n-aires. Les notions particulières de relations unaires et binaires sont généralisées au cas n-aire : tout sous-ensemble d'un produit cartésien $\mathfrak{R} : \wp^{A_1 \times \cdots \times A_n}$, ou $n \geq 1$ est appelé *relation (n-aire)* sur $A_1 \times \cdots \times A_n$. Si $\forall i.\ A_i = D$ nous écrivons $\mathfrak{R} : \wp^{D^n}$. Étant donné une relation binaire $\mathfrak{R} : \wp^{D \times D}$ sur un domaine D, l'*extension produit à n coordonnées*, notée $\mathfrak{R}^n : \wp^{D^n \times D^n}$, est la relation définie par $(x_1, ..., x_n)\mathfrak{R}^n(y_1, ..., y_n) \stackrel{\triangle}{\Longleftrightarrow} \forall i = 1...n.\ x_i \mathfrak{R} y_i$.

1.1.8. Suites. Les suites finies ou infinies sont des exemples de relations binaires sur $\mathbb{N}_+$ et A. Une *suite (finie ou infinie)* dans A, est une fonction partielle $s : \mathbb{N}_+ \rightharpoonup A$, notée $s : A^{\leq \omega}$ ou $s : A^{\infty}$, telle que $\exists s(i+1) \Rightarrow \exists s(i)$ pour tout $i \in \mathbb{N}_+$. La suite *vide*, lorsque $\mathrm{dom}(s) = \emptyset$, est notée ε. La notation x_i est toujours synonyme de $x(i)$, et l'expression usuelle $s = x_1..x_n$, signifie $s = \{(1, x_1), .., (n, x_n)\}$. S'il existe un entier j tel que $\exists s(j)$ et $\nexists s(j+1)$ alors la suite est *finie*, notée $s : A^*$ou $s : A^{<\omega}$, et sa *longueur* est $|s| \triangleq j$. La longueur de la suite vide est nulle par définition : $|\varepsilon| = 0$. La *concaténation* de deux suites finies $x, y : A^*$, notée $x.y$, est la suite telle que $(x.y)(i) = x(i)$ si $i \leq |x|$, et sinon $(x.y)(i) \doteq y(i - |x|)$. Une *suite infinie* dans A, notée $s : A^{\omega}$, est une fonction (totale) $s : \mathbb{N}_+ \to A$ des entiers naturels non nuls $\mathbb{N}_+$ vers l'ensemble A. Ainsi, nous savons par définition que $A^{\leq \omega} = A^* \cup A^{\omega}$. Étant donné une relation binaire $\mathfrak{R} : \wp^{A \times A}$ sur A, l'*extension produit* (*aux suites finies*) de la relation $\mathfrak{R}$, notée $\mathfrak{R}^* : \wp^{A^* \times A^*}$, est la relation définie par $x \mathfrak{R}^* y \stackrel{\triangle}{\Longleftrightarrow} |x| = |y| = n \ \wedge\ \forall i = 1...n.\ x_i \mathfrak{R} y_i$. L'*inclusion ensembliste sur les suites*, notée $\subseteq^*$, est l'inclusion des suites considérées comme des simples ensembles non ordonnés : $s \subseteq^* s' \stackrel{\triangle}{\Longleftrightarrow} \{x \in A \mid x = s_i,\ i \in \mathbb{N}_+\} \subseteq \{y \in A \mid y = s'_j,\ j \in \mathbb{N}_+\}$.

1.1.9. Constructions polymorphes. Soit χ_1 et χ_2 des *classes*, i.e. des ensembles d'ensembles, et soit $t : \chi_1 \to \chi_2$ une fonction entre les deux classes. Si une fonction $p : \chi_1 \to \bigcup_{Y \in \chi_2} Y$ fait correspondre à tout ensemble $\alpha \in \chi_1$ une valeur $p(\alpha)$, notée p_α, appartenant à l'ensemble $t(\alpha)$, noté t_α :

$$\forall \alpha \in \chi_1.\ \ p_\alpha \in t_\alpha$$

alors nous disons que p est une *construction polymorphe*, notée :

$$p\ :\ \forall \alpha{:}\chi_1.\, t_\alpha$$

Le type $\forall \alpha{:}\chi_1.\, t_\alpha$ permettra de situer les couples (α, p_α) de la relation p d'une façon plus précise par rapport au type $\chi_1 \to \bigcup_{Y \in \chi_2} Y$: si ce dernier nous indique que $(\alpha, p_\alpha) \in \chi_1 \times \bigcup_{Y \in \chi_2} Y$, le premier nous indique, en revanche, que $(\alpha, p_\alpha) \in \chi_1 \times \bar{Y}$ où $\bar{Y} = t(\alpha)$.

1.1.9.1. *Test d'égalité.* Un exemple de construction polymorphe est le *test d'égalité* ou fonction *conditionnelle*. Étant donné une classe χ_t pour les tests et une classe χ_v pour les valeurs, la fonction $\mathrm{cond}_{X,Y}(x,y,z) : X \to Y$, où $x \in X$ et $y, z \in Y$, est définie pour tout ensemble $X \in \chi_t$ et tout ensemble $Y \in \chi_v$ de la façon suivante :

$$\mathrm{cond}_{X,Y}(x,y,z)(v) \triangleq \begin{cases} y & \text{si} \quad v = x \\ z & \text{si} \quad v \neq x \end{cases}$$

Nous pouvons schématiser la construction par l'écriture :

$$\mathrm{cond} : \forall\alpha{:}\chi_t.\forall\beta{:}\chi_v.\,(\alpha \times \beta \times \beta) \to \alpha \to \beta$$

Le *test d'inégalité*, représenté par le même schéma, est défini, pour tout $X \in \chi_t$ et tout $Y \in \chi_v$, par l'équation $\mathrm{cond}^{\perp}_{X,Y}(x,y,z)(v) \triangleq \mathrm{cond}_{X,Y}(x,z,y)(v)$. Dans le cas particulier où l'ensemble pour les tests et celui pour les valeurs coïncident $X = Y \in \chi$, les deux constructions peuvent être schématisées de la façon suivante ;

$$\mathrm{cond}, \mathrm{cond}^{\perp} : \forall\alpha{:}\chi.\,(\alpha \times \alpha \times \alpha) \to \alpha \to \alpha.$$

1.1.9.2. *Extension d'une relation aux parties.* Étant donné un ensemble D, la *puissance* $\wp^n(D)$ de l'opération de passage aux parties est définie par $\wp^0(D) \triangleq D$ et $\wp^{n+1}(D) \triangleq \wp(\wp^n(D))$. Étant donné une relation $\leq : \wp^{D\times D}$ sur un domaine D, l'*extension aux puissances* $\wp^n(D)$ est la construction polymorphe :

$$\leq_{(.)} : \forall\alpha{:}\{\wp^n(D) \mid n \in \mathbb{N}\}.\,\wp^{\alpha\times\alpha}$$

définie inductivement par :

$$\begin{cases} (\leq_0) = (\leq) & \\ X \leq_{n+1} Y & \stackrel{\triangle}{\Longleftrightarrow} \quad \forall x \in X.\ \forall y \in Y.\ \ x \leq_n y \end{cases}$$

Pour $n = 1$ nous avons l'extension de la propriété $\leq$ aux parties de D ; pour $n = 2$ nous avons l'extension de la propriété $\leq$ aux classes (de parties) de D.

1.1.10. Syntaxes. Une *syntaxe* ou *grammaire* (*formelle*) est un quadruplet $G = (N, T, s, ::=)$ où T et N sont des ensembles disjoints d'éléments appelés *symboles* respectivement *terminaux* et *non terminaux*, $s \in N$, et $(::=) : 2^{V^*\times V^*}$, où $V \triangleq N \oplus T$, est une relation (en notation infixe) telle que $(::=) \subseteq \{(x,y) \in V^* \times V^* \mid \exists i \in \mathbb{N}.\, x(i) \in N\}$. La *clôture par contexte* de $::=$ est la relation $\to_G \triangleq \{(x_1, x_2) \in V^* \times V^* \mid x_1 = y_1.u.z_1,\ x_2 = y_2.v.z_2,\ u ::= v\}$. Le *langage* généré par la syntaxe G est l'ensemble $L(G) \triangleq \{y \in T^* \mid s \to^*_G y\}$ où la relation $\to^*_G$ (en notation, elle aussi, infixe) est la clôture transitive de $\to_G$.

1.1.11. Relations bien fondées. Tous les principes d'induction mathématique se généralisent à un seul principe, celui de l'induction *bien fondée*[2]. Une relation binaire $<$ sur un domaine D est *bien fondée* s'il n'existe pas de chaînes descendantes infinies, i.e. :

$$\nexists (d_i)_{i\in\mathbb{N}} \subseteq D.\ (\forall i \in \mathbb{N}.\ d_{i+1} < d_i)$$

[2] *well founded induction*

Une relation binaire bien fondée doit être forcément *anti-réflexive*, i.e. $< \cap \{(x,x) \mid x \in D\} = \emptyset$. Étant donné une relation binaire $\leq$ sur un domaine D, la *composante stricte* (ou *anti-réflexive*) de la relation est le sous-ensemble $< \triangleq \{(x,y) \in \leq \mid x \neq y\}$. Le produit à n composantes $(x_1, ..., x_n) < (y_1, ..., y_n) \overset{\triangle}{\iff} \forall i = 1...n.\, x_i < y_i$ d'une relation bien fondée $<$ est, lui aussi, une relation bien fondée (sur le domaine D^n).

Le principe d'induction bien fondé s'appuie sur une relation bien fondée $<$ définie sur un domaine D : pour démontrer que les éléments d'un sous-ensemble $A \subseteq D$ vérifient tous une propriété P (i.e. $A \subseteq P$) il suffit de prouver que supposer la propriété vraie pour tout élément $b \in A$ tel que $b < a$, où $a \in A$, oblige la propriété à être vraie pour a :

$$\frac{\forall a \in A.\ (\forall b \in A.\ (b < a \Rightarrow P(b)) \Rightarrow P(a))}{\forall a \in A.\ P(a)}$$

1.2. Algèbres typées

Le mot *algèbre*, qui entraîne les notions corrélées d'homomorphisme et d'isomorphisme, se retrouve à plusieurs reprises en mathématique. L'inflation du mot, pour des signifiés qui sont souvent très proches mais différents, est telle que certains auteurs préfèrent d'autre noms, comme par exemple *magma* ou *structure*, pour éviter tout malentendu ou pour marquer la différence. De multiples définitions proviennent, d'une part, de la théorie mathématique des algèbres universelles[3], d'autre part, de la théorie plus orientée vers l'informatique, des algèbres de termes[4] ou Σ-algèbres. Dans les deux cadres théoriques, la notion de *type* apparaît d'une utilité fondamentale, à savoir celle de préciser, d'une façon plus ou moins fine, le périmètre d'une opération, c'est-à-dire son domaine d'appartenance. L'information véhiculée par le type indique, par exemple, qu'il s'agit d'un entier naturel, ou bien d'une propriété des entiers, ou d'une fonction etc.

Cependant, l'étendue du langage des types est toujours limitée aux fonctions $A_1 \times ... \times A_n \to A_{n+1}$, aux constantes $\to A$, et aux prédicats $A_1 \times ... \times A_n \to 2$, où les domaines A_i impliqués sont des ensembles de base (les *sortes*) et non pas des ensembles issus, eux-mêmes, d'une autre construction mathématique (par exemple des fonctions). La définition des types d'opération est toujours accompagnée d'une autre notion fondamentale, celle d'homomorphisme et d'isomorphisme entre algèbres. Intuitivement, un homomorphisme est une interprétation des valeurs d'une algèbre dans une autre qui conserve les opérations de départ : tout ce qui est dans une opération au départ le demeure à l'arrivée, dans l'opération correspondante et après la traduction des valeurs. Ces arguments, très généraux, visant la conservation des propriétés, constituent un métalangage mathématique qui permet d'aborder plusieurs sujets de nature apparemment différente. En effet, la *théorie des domaines*

[3] Voir, par exemple, la définition de *algebra* dans [**Burris & Sankap., 1981**], celle de *heterogeneous algebra* dans [**Lipson, 1981**], celle de *algebraic systems* dans [**Mal'cev, 1973**], celle de *formation* dans [**Gericke, 1966**], et celle de *many-sorted algebra* dans [**Plotkin, 1993**].

[4] Voir, par exemple, les définitions de *Σ-algebra* dans [**Loeckx et al., 2000**] et dans [**Tucker & Zucker, 2000**].

et la *théorie de l'interprétation abstraite* de programmes peuvent être approchées d'une telle façon.

La limitation principale des algèbres universelles, l'impossibilité d'imbriquer les types les uns dans les autres, est un inconvénient qui peut être dépassé en s'inspirant des *relations logiques*, une notion qui revient à la théorie du λ-calcul simplement typé[5]. Elle nous permettra d'étendre les notions d'homomorphisme et d'isomorphisme aux opérations d'un type appartenant à un langage étendu, inductif et défini formellement par une syntaxe.

Pour enrichir ultérieurement le métalangage des types, nous ajouterons l'expression dénotant les constructions polymorphes. Dans la problématique concernant les variables (*abstraction*, *substitution,* etc), nous serons guidés, une fois de plus, par la théorie du λ-calcul simplement typé.

1.2.1. Types. Les types sont les correspondants syntaxiques des domaines et des propriétés mathématiques. Étant donné un ensemble $\mathcal{B} \supseteq \{1, 2, \mathbb{N}, \mathbb{N}_+, \mathbb{N}_\infty\}$ de noms appelés *types de base*, un ensemble $S = \{D_i\}_{i \in I}$ disjoint de $\mathcal{B}$, de noms appelés *sortes*, et un ensemble de *variables* V disjoint de $\mathcal{B}$ et de S, l'ensemble des *types ouverts* sur $\mathcal{B}$ et S, noté $\widetilde{\mathcal{T}}(\mathcal{B}, S)$, est l'ensemble défini par la syntaxe $(\{\tau, \theta, \vartheta, \pi, \sigma, \chi, \ell\}, \mathcal{B} \cup S \cup V \cup C \cup \{\times, \oplus, \wp, \wp_{\mathsf{F}}, \rightarrow, \twoheadrightarrow, *, \forall, :, \mathsf{base}, \mathsf{sort}, \{, ,, \}, ., \mathsf{if}, =, \mathsf{then}, \mathsf{else}\}, \tau, ::=)$, où $::=$ est la relation suivante :

$$
\begin{array}{rcl}
\widetilde{\mathcal{T}}(\mathcal{B}, S) \ni \tau & ::= & \forall\alpha{:}\chi.\,\tau \mid \forall\alpha{:}\chi.\,\mathsf{if}\ \alpha = \vartheta\ \mathsf{then}\ \tau_1\ \mathsf{else}\ \tau_2 \mid \theta \\
\theta & ::= & \vartheta \mid \theta \times \theta \mid \theta \oplus \theta \mid \wp^{\theta} \mid \wp_{\mathsf{F}}^{\theta} \mid \theta \rightarrow \theta \mid \theta \twoheadrightarrow \theta \mid \theta^* \mid \theta^\infty \mid \alpha \\
\vartheta & ::= & \pi \mid \sigma \\
\chi & ::= & \mathsf{base} \mid \mathsf{sort} \mid \{\ell\} \\
\ell & ::= & \vartheta \mid \ell, \ell
\end{array}
$$

Les types *non polymorphes* (ou *monomorphes*) θ représentent, dans l'ordre, les éléments de type prédéfini $\pi \in \mathcal{B}$, les éléments des ensembles baptisés $\sigma \in S$, les éléments d'un produit cartésien ou d'une somme de types admissibles, les parties (finies ou infinies) d'un type admissible, les fonctions (totales ou partielles) d'un type admissible vers un type admissible, les suites (finies ou arbitraires) de types admissibles, et les variables de type $\alpha \in V$. Les types (*généraux* ou *polymorphes*) peuvent être soit des abstractions simples $\tau = \forall\alpha{:}\chi.\,\tau'$, soit des abstractions suivies d'un test d'égalité $\tau = \forall\alpha{:}\chi.\,\mathsf{if}\ \alpha = \vartheta\ \mathsf{then}\ \tau_1\ \mathsf{else}\ \tau_2$, soit des types non polymorphes $\tau = \theta$. Le symbole sort sera le représentant syntaxique de la classe S toute entière, et base le représentant de la classe des types de base $\mathcal{B}$. Nous pourrons écrire, par exemple, des types polymorphes du genre $\forall\alpha{:}\mathsf{sort}.\,\alpha^*$ ou $\forall\alpha{:}\mathsf{base}.\,\alpha \rightarrow \alpha$.

Nous supposerons toujours que les types utilisés soient des termes *clos*, autrement dit, que toute variable α apparaissant dans une expression de type soit liée par une abstraction correspondante $\forall\alpha$. D'une façon plus formelle, l'ensemble des *types* (*clos*), noté $\mathcal{T}(\mathcal{B}, S)$, est le sous-ensemble de types ouverts $\tau \in \widetilde{\mathcal{T}}(\mathcal{B}, S)$ dont l'ensemble des *variables libres*, défini dans la table 1, est vide :

$$\mathcal{T}(\mathcal{B}, S) \triangleq \{\tau \in \widetilde{\mathcal{T}}(\mathcal{B}, S) \mid FV(\tau) = \emptyset\}$$

[5] Voir, par exemple, la définition de *logical relation* dans [**Mitchell, 1996**] ou celle de *pre-logical relation* dans [**Honsell & Sanella, 1999**].

TAB. 1. Variables libres

$$\mathsf{FV}(\tau) \triangleq \begin{cases} \emptyset & \text{si} \quad \tau = \pi \in \mathcal{B} \ \text{ ou } \ \tau = \sigma \in S \\ \mathsf{FV}(\tau_1) \cup \mathsf{FV}(\tau_2) & \text{si} \quad \tau = \tau_1 \times \tau_2 \ \text{ ou } \ \tau = \tau_1 \oplus \tau_2 \\ \mathsf{FV}(\tau_1) \cup \mathsf{FV}(\tau_2) & \text{si} \quad \tau = \tau_1 \rightarrow \tau_2 \ \text{ ou } \ \tau = \tau_1 \twoheadrightarrow \tau_2 \\ \mathsf{FV}(\tau') & \text{si} \quad \tau = \wp^{\tau'} \ \text{ ou } \ \tau = \wp_{\mathsf{F}}^{\tau'} \\ \mathsf{FV}(\tau') & \text{si} \quad \tau = (\tau')^* \ \text{ ou } \ \tau = (\tau')^\infty \\ \{\alpha\} & \text{si} \quad \tau = \alpha \in V \\ \mathsf{FV}(\tau) \setminus \{\alpha\} & \text{si} \quad \tau = \forall\alpha{:}\chi.\,\tau \\ \mathsf{FV}(\tau_1) \cup \mathsf{FV}(\tau_2) \setminus \{\alpha\} & \text{si} \quad \tau = \forall\alpha{:}\chi.\ \mathsf{if}\ \alpha = \vartheta\ \mathsf{then}\ \tau_1\ \mathsf{else}\ \tau_2 \end{cases}$$

Les types de base étant souvent implicites, nous pourrons noter l'ensemble des types par l'écriture $\mathcal{T}(S)$. Nous supposerons aussi que l'ensemble des types de base contienne, tout au moins, les représentants des domaines de base 1, 2, $\mathbb{N}$, $\mathbb{N}_+$ et $\mathbb{N}_\infty$ (cf. section 1.1.1).

La contrepartie sémantique d'un *type* τ est toujours, bien entendu, un *domaine*, et l'écriture $x : \tau$ prend toujours le sens d'appartenance de x au domaine représenté. Si on permet de confondre, sans pour autant provoquer d'ambiguïté, par l'écriture τ, à la fois le type et le domaine représenté, nous pouvons dire que $x : \tau$ est toujours synonyme de $x \in \tau$. Dans le même esprit, nous pourrons noter par la lettre π à la fois un type prédéfini et le domaine correspondant implicitement associé, et noter que $\tau_1 \subseteq \tau_2$ pour signifier l'inclusion des domaines représentés. Nous pouvons représenter les bijections existantes entre constructions mathématiques en décrétant, sur le langage $\mathcal{T}(S)$, les lois associatives : $(\tau_1 \times \tau_2) \times \tau_3 = \tau_1 \times (\tau_2 \times \tau_3) \triangleq \tau_1 \times \tau_2 \times \tau_3$, $(\tau_1 \oplus \tau_2) \oplus \tau_3 = \tau_1 \oplus (\tau_2 \oplus \tau_3) \triangleq \tau_1 \oplus \tau_2 \oplus \tau_3$. Nous pourrons parfois utiliser les abréviations usuelles chez les domaines : $(\tau^\omega) \triangleq (\mathbb{N}_+ \rightarrow \tau)$, $(\rightarrow \tau) \triangleq (1 \rightarrow \tau)$ et $(2^\tau) \triangleq (\tau \rightarrow 2)$. Parmi les abréviations, nous nous autoriserons à factoriser les types dans les branches des expressions de type polymorphe munie d'un test d'égalité. Par exemple, nous pourrons écrire $\forall\alpha{:}\chi.\,\tau \rightarrow \mathsf{if}\ \alpha = \vartheta\ \mathsf{then}\ \tau_1\ \mathsf{else}\ \tau_2$ au lieu de $\forall\alpha{:}\chi.\ \mathsf{if}\ \alpha = \vartheta\ \mathsf{then}\ \tau \rightarrow \tau_1\ \mathsf{else}\ \tau \rightarrow \tau_2$.

1.2.1.1. *Types élémentaires et du premier ordre.* Parmi les types, nous appellerons *élémentaire* tout type appartenant au sous-langage $\mathcal{T}_{\mathcal{E}}(S) \subset \mathcal{T}(S)$ défini par la syntaxe $\mathcal{T}_{\mathcal{E}}(S) \ni \tau ::= \pi \,|\, \sigma \,|\, \wp^\tau \,|\, \wp_{\mathsf{F}}^\tau \,|\, \tau \times \tau \,|\, \tau \oplus \tau$ où $\pi \in \mathcal{B}$ et $\sigma \in S$. D'autre part, le langage des types *du premier ordre* $\mathcal{T}_{\mathcal{F}}(S) \subset \mathcal{T}(S)$ est défini par la syntaxe $\mathcal{T}_{\mathcal{F}}(S) \ni \tau ::= \tau_\varepsilon \,|\, \tau_\varepsilon \rightarrow \tau \,|\, \tau_\varepsilon \twoheadrightarrow \tau$ où $\tau_\varepsilon \in \mathcal{T}_{\mathcal{E}}(S)$. Les types du premier ordre représentent donc les types élémentaires et les fonctions, partielles ou totales, n'acceptant que des arguments de type élémentaire. En revanche, les fonctions, totales ou partielles, d'un type n'étant pas du premier ordre seront des fonctions *d'ordre supérieur*.

1.2.1.2. *Substitution des variables.* Nous allons adapter, au langage des types, l'opération de substitution des variables qui est le mécanisme fondamental du λ-calcul. Il s'agit d'une méta-opération de support aux traitements des constructions polymorphes[6]. La *substitution* d'une variable α avec une valeur a dans un type τ, noté $\tau[{}^{a}/_{\alpha}]$, est définie dans la table 2.

[6] Comme dans le cadre du lambda-calcul, la formalisation de l'opération de substitution se fait en dehors de la théorie elle-même, dans ce que Haskell Curry appelait l'épi-théorie

TAB. 2. Substitution des variables

$$
\tau[{}^{a}/_{\alpha}] \triangleq \begin{cases}
\pi & \text{si} \quad \tau = \pi \in \mathcal{B} \\
\sigma & \text{si} \quad \tau = \sigma \in S \\
\tau_1[{}^{a}/_{\alpha}] \times \tau_2[{}^{a}/_{\alpha}] & \text{si} \quad \tau = \tau_1 \times \tau_2 \\
\tau_1[{}^{a}/_{\alpha}] \oplus \tau_2[{}^{a}/_{\alpha}] & \text{si} \quad \tau = \tau_1 \oplus \tau_2 \\
\wp^{\tau'[{}^{a}/_{\alpha}]} & \text{si} \quad \tau = \wp^{\tau'} \\
\wp_{\mathsf{F}}^{\tau'[{}^{a}/_{\alpha}]} & \text{si} \quad \tau = \wp_{\mathsf{F}}^{\tau'} \\
\tau_1[{}^{a}/_{\alpha}] \rightarrow \tau_2[{}^{a}/_{\alpha}] & \text{si} \quad \tau = \tau_1 \rightarrow \tau_2 \\
\tau_1[{}^{a}/_{\alpha}] \nrightarrow \tau_2[{}^{a}/_{\alpha}] & \text{si} \quad \tau = \tau_1 \nrightarrow \tau_2 \\
(\tau'[{}^{a}/_{\alpha}])^{*} & \text{si} \quad \tau = (\tau')^{*} \\
(\tau'[{}^{a}/_{\alpha}])^{\infty} & \text{si} \quad \tau = (\tau')^{\infty} \\
a & \text{si} \quad \tau = \alpha \in V \\
\beta & \text{si} \quad \tau = \beta \in V,\ \beta \neq \alpha \\
\tau & \text{si} \quad \tau = \forall \alpha{:}\chi.\, \tau' \\
\tau & \text{si} \quad \tau = \forall \alpha{:}\chi.\, \mathsf{if}\ \alpha = \vartheta\ \mathsf{then}\ \tau_1\ \mathsf{else}\ \tau_2 \\
\forall \beta{:}\chi.\, \tau'[{}^{a}/_{\alpha}] & \text{si} \quad \tau = \forall \beta{:}\chi.\, \tau',\ \beta \neq \alpha \\
\forall \alpha{:}\chi.\, \mathsf{if}\ \alpha = \vartheta\ \mathsf{then}\ \tau_1[{}^{a}/_{\alpha}]\ \mathsf{else}\ \tau_2[{}^{a}/_{\alpha}] & \text{si} \quad \tau = \forall \alpha{:}\chi.\, \mathsf{if}\ \alpha = \vartheta\ \mathsf{then}\ \tau_1\ \mathsf{else}\ \tau_2,\ \beta \neq \alpha
\end{cases}
$$

1.2.2. Signatures. Une *signature algébrique* (ou *signature* tout court) est un quadruplet $\Sigma = (\mathcal{B}, S, F, T)$ où la première composante $\mathcal{B}$ est un ensemble de types de base, la deuxième $S = \{D_i\}_{i \in I}$ est un ensemble de *sortes,* la troisième F est un ensemble non vide de noms d'opérations $F = \{f_j\}_{j \in J}$, disjoint de S, et la quatrième est une application $T : F \rightarrow \mathcal{T}(\mathcal{B}, S)$ qui associe à tout *nom* d'opération $f \in F$, le *type* de l'opération.

NOTATION 1.2.1. Par la suite, nous pourrons sous-entendre l'ensemble des types de bases $\mathcal{B}$ (par exemple lorsque ce dernier est vide) et dénoter la signature par un triplé $\Sigma = (S, F, T)$. En outre, lorsque S ou F seront des ensembles finis, on s'autorisera, comme en littérature, des définitions de signatures du genre $\Sigma = (S, \{o_1 : \phi_1, ..., o_n : \phi_n\})$ pour signifier, bien entendu, $\Sigma = (S, \{o_1, .., o_n\}, \{(o_1, \phi_1), .., (o_n, \phi_n)\})$. Nous pourrons aussi préciser le type de plusieurs opérations à la fois, séparées par des virgules : $\Sigma = (S, \{o_{i_1}, .., o_{i_k} : \phi_1, .., o_{j_1}, .., o_{j_h} : \phi_n\})$ et éventuellement séparer les sortes des opérations par un point-virgule en évitant les accolades : $\Sigma = (D_1, .., D_n\,; o_{i_1}, .., o_{i_k} : \phi_1, .., o_{j_1}, .., o_{j_h} : \phi_m)$. Lorsque la différence entre sortes et opérations sera évidente, nous pourrons aussi éviter le point-virgule et/ou le type, comme pour les ordres partiels (cf. section 2.2) dont la signature est le plus souvent notée $(D, \leq)$ au lieu de $(\{D\}, \{\leq : \wp^{D \times D}\})$.

1.2.3. Algèbres. Une *algèbre* est un couple $A = ((.)^{A}, T)$ où :

(1) l*e signifié des noms* $(.)^{A}$ est une somme de fonctions *bijectives*

$$(.)^{A} : \mathcal{B} \rightarrow \mathcal{B}^{A} \ \oplus \ S \rightarrow S^{A} \ \oplus \ F \rightarrow F^{A}$$

où $\mathcal{B} = \{\mathcal{B}_k\}_{k \in K} \supseteq \{1, 2, \mathbb{N}, \mathbb{N}_{+}, \mathbb{N}_{\infty}\}$ est l'ensemble des types de base, $\mathcal{B}^{A} = \{\mathcal{B}_k^{A}\}_{k \in K}$ est la collection des domaines de base, $S = \{D_i\}_{i \in I}$ est un ensemble de sortes, $S^{A} = \{D_i^{A}\}_{i \in I}$ est une collection de domaines (les *signifiés des sortes*) ; $F = \{f_j\}_{j \in J}$ est un ensemble de noms d'opérations, disjoint de S, et $F^{A} = \{f_j^{A}\}_{j \in J}$ est une collection d'opérations (les *signifiés*

TAB. 3. Signifié ensembliste des types τ et des classes χ

$$\tau^A \triangleq \begin{cases} \pi^A & \text{si} \quad \tau = \pi \in \mathcal{B} \\ \sigma^A & \text{si} \quad \tau = \sigma \in S \\ \tau_1^A \times \tau_2^A & \text{si} \quad \tau = \tau_1 \times \tau_2 \\ \tau_1^A \oplus \tau_2^A & \text{si} \quad \tau = \tau_1 \oplus \tau_2 \\ \wp^{\tau'^A} & \text{si} \quad \tau = \wp^{\tau'} \\ \wp_F^{\tau'^A} & \text{si} \quad \tau = \wp_F^{\tau'} \\ \tau_1^A \to \tau_2^A & \text{si} \quad \tau = \tau_1 \to \tau_2 \\ \tau_1^A \mathrel{-\!\!\ominus\!\!\to} \tau_2^A & \text{si} \quad \tau = \tau_1 \mathrel{-\!\!\ominus\!\!\to} \tau_2 \\ (\tau^A)^* & \text{si} \quad \tau = (\tau')^* \\ (\tau^A)^\infty & \text{si} \quad \tau = (\tau')^\infty \\ \alpha & \text{si} \quad \tau = \alpha \in V \\ \forall \alpha{:}\chi^A.\, \tau'^A & \text{si} \quad \tau = \forall \alpha{:}\chi.\, \tau' \\ \forall \alpha{:}\chi^A.\, \mathrm{cond}(\vartheta^A, \tau_1^A, \tau_2^A)(\alpha) & \text{si} \quad \tau = \forall \alpha{:}\chi.\ \mathsf{if}\ \alpha = \vartheta\ \mathsf{then}\ \tau_1\ \mathsf{else}\ \tau_2 \end{cases}$$

$$\chi^A \triangleq \begin{cases} \{x^A \mid x \in \mathcal{B}\} & \text{si} \quad \chi = \mathsf{base} \\ \{s^A \mid s \in S\} & \text{si} \quad \chi = \mathsf{sort} \\ \{x_1^A, .., x_n^A\} & \text{si} \quad \chi = \{x_1, .., x_n\} \end{cases}$$

des noms d'opération). La fonction $(.)^A$ est étendue automatiquement à $\mathcal{T}(\mathcal{B}, S)$, l'ensemble de tous les types sur les noms $\mathcal{B}$ et S, comme indiqué dans la Table 3.

(2) le *type des opérations* T est une fonction $T : F \to \mathcal{T}(\mathcal{B}, S)$ qui affecte un type *correct* à tous les noms d'opération de l'algèbre, c'est-à-dire un type tel que :

$$\forall f \in F. \quad T(f) = \tau \quad \Rightarrow \quad f^A \in \tau^A.$$

Le quadruplet $\Sigma_A = (\mathcal{B}, S, F, T)$ est la signature de l'algèbre A. La famille des algèbres sur une signature Σ sera notée $\mathsf{Alg}(\Sigma)$.

NOTATION 1.2.2. Comme suggéré par la syntaxe des types, le signifié de toute *classe syntaxique* χ est une *classe de domaines* (de base ou sortes) χ^A. On s'autorisera ainsi l'écriture $x \in \chi$ pour signifier $x^A \in \chi^A$.

Nous remarquerons que le signifié ensembliste d'un type de la forme $\forall \alpha : \chi.\, \tau$ est le domaine des constructions polymorphes (cf. section 1.1.9), c'est-à-dire le domaine des applications $\{f : \forall \alpha : \chi^A.\, \tau^A\}$ qui, étant donné un ensemble x dans la classe de domaines (de base ou sortes) χ^A, renvoient un élément appartenant au domaine $\tau[{}^{x}/_{\alpha}]^A$. L'ordre *extensionnel* entre fonctions, qui sera défini et détaillé dans le chapitre suivant (cf. section 2.1.4), est un exemple typique de construction polymorphe.

La *classe* des domaines *impliqués* par l'algèbre est l'image de la fonction d'interprétation des types sur tous les types admissibles de la signature Σ :

$$\mathcal{T}(\mathcal{B}, S)^A \triangleq \{\tau^A \mid \tau \in \mathcal{T}(\mathcal{B}, S)\} \quad \text{où} \quad \Sigma_A = (\mathcal{B}, S, F, T)$$

Les types de base et leur signifiés, i.e. les domaines de base, sont souvent implicites dans la définition d'une algèbre. Ceci est le cas, en particulier, lorsque les types de base coïncident exactement avec l'ensemble $\{1, 2, \mathbb{N}, \mathbb{N}_+, \mathbb{N}_\infty\}$.

1.2.3.1. *Structures.* Nous appellerons *structure* toute algèbre constituée d'un seul domaine D et d'un nombre arbitraire d'opérations de type, lui aussi, arbitraire. Lorsque il n'y aura qu'un nombre fini d'opérations, pour la signature d'une structure, nous pourrons utiliser la notation $(D, O_1, ..., O_n)$ qui oublie les accolades. Il s'agit d'un cas exemplaire où la différence entre sortes et opérations est évidente : puisqu'il n'y a qu'une sorte, elle occupera la première place, et les autres symboles seront forcément les opérations.

1.3. Morphismes

Tout en étant une notion plus générale que l'homologue de l'algèbre universelle, les homomorphismes maintiendront leur prérogative habituelle liée à la conservation des propriétés. Nous verrons aussi comment retrouver leur interprétation catégorique usuelle, celle de *morphisme* entre objets, où les objets seront, à ce moment-là, les algèbres typées.

1.3.1. Algèbres, types et opérations compatibles. Si deux algèbres $A = ((.)^A, T)$ et $B = ((.)^B, T')$ sont définies sur les mêmes types de base $\mathcal{B}$, et les signifiés de ces derniers coïncident :

$$\forall \tau \in \mathcal{B}. \quad \tau^A = \tau^B$$

nous disons que les algèbres sont *compatibles*. Dans cette hypothèse, soient $\Sigma_A = (\mathcal{B}, S, F, T)$ et $\Sigma_B = (\mathcal{B}, S', F', T')$ les signatures des deux algèbres, et soit $\psi : S \to S'$ une application traduisant les sortes de A en sortes de B. La Table 4 illustre l'extension *naturelle* de ψ à tous les types admissibles. Ainsi, la traduction des noms des sortes $\psi : S \to S'$ définira implicitement une traduction $\psi : \mathcal{T}(\mathcal{B}, S) \to \mathcal{T}(\mathcal{B}, S')$ de tout type de l'algèbre A dans un type de l'algèbre B. Nous utiliserons souvent la notation τ_ψ pour indiquer le résultat de l'application de ψ à l'argument τ : $\tau_\psi = \psi(\tau)$.

La correspondance tracée par ψ entre les types des deux algèbres est une correspondance stricte, une forme d'*égalité syntaxique* (modulo la traduction des sortes). Une relation moins stricte, la *compatibilité* entre types, notée $\sim_\psi$: $\wp^{\mathcal{T}(\mathcal{B},S)\times\mathcal{T}(\mathcal{B},S')}$, est définie dans la table 5. Intuitivement, la relation $\sim_\psi$ étend la fonction ψ en permettant d'associer des types $\forall\alpha:\chi.\tau$ et $\forall\beta:\chi'.\tau'$ si le second type exprime, par rapport au premier, des constructions polymorphes plus générales. En particulier, la règle (const) permet aux types polymorphes d'être associés à des types non nécessairement polymorphes, ces derniers pouvant toujours être considérés, à l'occasion, comme des constructions polymorphes *constantes*, dont le résultat est le même indépendamment de l'argument.

Si $\tau \sim_\psi \tau'$, nous disons que le type $\tau \in \mathcal{T}(\mathcal{B}, S)$ de la première algèbre est *compatible* (*via la traduction* ψ) avec le type $\tau' \in \mathcal{T}(\mathcal{B}, S')$ de la seconde algèbre. Étant donné une opération $f \in F$ de la première algèbre et une opération $f' \in F'$ de la seconde, nous disons que les deux opérations sont *compatibles* si leurs types sont compatibles :

$$T(f) \sim_\psi T'(f')$$

TAB. 4. Interprétation ψ des types τ et des classes χ d'une algèbre à une autre

$$\psi(\tau) \triangleq \begin{cases} \pi & \text{si} \quad \tau = \pi \in \mathcal{B} \\ \psi(\sigma) & \text{si} \quad \tau = \sigma \in S \\ \psi(\tau_1) \times \psi(\tau_2) & \text{si} \quad \tau = \tau_1 \times \tau_2 \\ \psi(\tau_1) \oplus \psi(\tau_2) & \text{si} \quad \tau = \tau_1 \oplus \tau_2 \\ \wp^{\psi(\tau')} & \text{si} \quad \tau = \wp^{\tau'} \\ \wp_{\mathsf{F}}^{\psi(\tau')} & \text{si} \quad \tau = \wp_{\mathsf{F}}^{\tau'} \\ \psi(\tau_1) \rightarrow \psi(\tau_2) & \text{si} \quad \tau = \tau_1 \rightarrow \tau_2 \\ \psi(\tau_1) \nrightarrow \psi(\tau_2) & \text{si} \quad \tau = \tau_1 \nrightarrow \tau_2 \\ \psi(\tau')^* & \text{si} \quad \tau = (\tau')^* \\ \psi(\tau')^\infty & \text{si} \quad \tau = (\tau')^\infty \\ \alpha & \text{si} \quad \tau = \alpha \in V \\ \forall \alpha{:}\psi(\chi).\, \psi(\tau') & \text{si} \quad \tau = \forall \alpha{:}\chi.\, \tau' \\ \forall \alpha{:}\psi(\chi).\, \mathsf{if}\ \alpha = \psi(\vartheta)\ \mathsf{then}\ \psi(\tau_1)\ \mathsf{else}\ \psi(\tau_2) & \text{si} \quad \tau = \forall \alpha{:}\chi.\, \mathsf{if}\ \alpha = \vartheta\ \mathsf{then}\ \tau_1\ \mathsf{else}\ \tau_2 \end{cases}$$

$$\psi(\chi) \triangleq \begin{cases} \mathsf{base} & \text{si} \quad \chi = \mathsf{base} \\ \mathsf{sort} & \text{si} \quad \chi = \mathsf{sort} \\ \{\psi(x_1),..,\psi(x_n)\} & \text{si} \quad \chi = \{x_1,..,x_n\} \end{cases}$$

TAB. 5. Types compatibles via l'interprétation ψ

$$\frac{\psi(\tau) = \tau'}{\tau \sim_\psi \tau'} \text{ (ext)}$$

$$\frac{\forall x \in \chi. \quad \psi(x) \in \chi' \quad \wedge \quad \tau[{}^x/_\alpha] \sim_\psi \tau'[{}^x/_\beta]}{\forall \alpha{:}\chi.\, \tau \sim_\psi \forall \beta{:}\chi'.\, \tau'} \text{ (poly)}$$

$$\frac{FV(\tau') = \emptyset \qquad \forall \alpha{:}\chi.\tau \sim_\psi \forall \beta{:}\chi_\psi.\tau'}{\forall \alpha{:}\chi.\, \tau \sim_\psi \tau'} \text{ (const)}$$

EXAMPLE 1.3.1. Soit A une algèbre constituée de deux sortes $S = \{\mathbb{Z}, \mathbb{B}\}$, où $\mathbb{Z}$ représente le domaine des nombres entiers et $\mathbb{B}$ celui des booléens $\{tt, ff\}$, et d'une opération polymorphe $F = \{+_{(.)} : \forall \alpha : \{\mathbb{Z},\mathbb{B}\}.\, \alpha \rightarrow \alpha \rightarrow \alpha\}$, où $+_\mathbb{Z}$ est l'*addition* sur les entiers et $+_\mathbb{B}$ est le *ou logique* sur les booléens. Par ailleurs, soit B l'algèbre des nombres réels, où $S' = \{\mathbb{R}\}$ et $F' = \{+ : \mathbb{R} \rightarrow \mathbb{R} \rightarrow \mathbb{R}\}$. La seule traduction possible pour les noms sera $\psi(\mathbb{Z}) = \psi(\mathbb{B}) = \mathbb{R}$ et $\psi(+_{(.)}) = +$. Les deux opérations allogènes seront alors compatibles. En effet, la règle (const) nous permettra de considérer l'addition sur les réels comme une construction polymorphe constante, c'est-à-dire comme une opération de type $\forall \beta{:}\{\mathbb{R}\}.\, \beta \rightarrow \beta \rightarrow \beta$. Ensuite, en utilisant la règle (poly), nous pourrons déduire la compatibilité de l'addition polymorphe de l'algèbre A avec celle (monomorphe) des réels. □

1.3.2. Interprétations. Soient $A = ((.)^A, T)$ et $B = ((.)^B, T')$ deux algèbres compatibles, de signature $\Sigma_A = (\mathcal{B}, S, F, T)$ et $\Sigma_B = (\mathcal{B}, S', F', T')$. Une *interprétation* de l'algèbre A dans l'algèbre B est un couple (ψ, φ) où :

(1) l'*interprétation des noms* $\psi : S \to S' \oplus F \to F'$ est une somme de fonctions qui traduit toute sorte de A dans une sorte de B et toute opération de A dans une opération compatible de B :

$$\forall f \in F. \;\; T(f)_\psi \sim_\psi T'(f_\psi)$$

(2) l'*interprétation des valeurs* est une construction polymorphe :

$$\varphi \;:\; \forall \alpha{:}S.\, \alpha \to \psi(\alpha)$$

qui associe à toute sorte $D \in S$ une traduction des valeurs $\varphi_{D} : D^A \to D^B_\psi$.

L'application $\psi : S \to S'$ sera appelée *germe* de l'interprétation des types. Nous remarquerons qu'aucune des composantes ψ et φ de l'interprétation n'est forcée d'être injective ou surjective. Leur raison d'être est celle de traduire (d'où leurs noms) les objets du discours d'une algèbre à l'autre. Nous remarquerons aussi que les domaines correspondants aux sortes peuvent avoir une intersection non vide. L'interprétation d'un élément $x \in D_1^A \cap D_2^A$, où $D_1, D_2 \in S$, pourra donc se faire par l'intermédiaire de la fonction $\varphi_{D_1} : D_1^A \to D^B_{1_\psi}$ ou de la fonction $\varphi_{D_2} : D_2^A \to D^B_{2_\psi}$. Lorsque l'interprétation des valeurs ne sera pas ambiguë, ou lorsque les domaines seront disjoints, c'est-à-dire lorsque nous aurons la condition :

$$\forall D_1, D_2 \in S. \;\; \forall x \in D_1^A \cap D_2^A. \qquad \varphi_{D_1}(x) = \varphi_{D_2}(x) \in D^B_{1_\psi} \cap D^B_{2_\psi}$$

nous pourrons éviter de spécifier le domaine d'application de φ, considérant cette dernière comme une fonction de l'union des domaines de l'algèbre A vers l'union des domaines de B :

$$\varphi \;:\; (\bigcup_{D \in S} D^A) \to (\bigcup_{D \in S} D^B_\psi)$$

Une condition plus faible, que nous appellerons *cohérence* de l'interprétation (ψ, φ), demande que si deux sortes distinctes $D \neq D'$ de la première algèbre sont associées à la même sorte $D_\psi = D'_\psi$ de la seconde algèbre, alors l'interprétation d'une valeur $x \in D^A \cap D'^A$ coïncide, que l'on considère $x \in D^A$ ou que l'on considère $x \in D'^A$:

$$\forall D, D' \in S. \;\; \psi(D) = \psi(D') \;\;\; \Rightarrow \;\;\; \forall x \in D^A \cap D'^A. \;\; \varphi_D(x) = \varphi_{D'}(x)$$

La possibilité, dans la définition d'une algèbre, de séparer les *types de base* des *types sorte*, permet de distinguer les domaines qui ne doivent par être interprétés, des domaines qui doivent, en revanche, passer le processus de traduction. En effet, les types de base ne concernent ni l'interprétation des noms ni celle des valeurs. Dans les deux cas, la fonction *identité* véhicule implicitement les noms et les valeurs de base d'une algèbre vers l'autre.

1.3.3. Relation de conservation. Étant donné une interprétation (ψ, φ) d'une algèbre A de signature $\Sigma_A = (\mathcal{B}, S, F, T)$ dans une algèbre B de signature $\Sigma_B = (\mathcal{B}, S', F', T')$, nous dirons qu'un objet $x \in \tau^A$ de la première algèbre se *conserve* dans un objet de type compatible $x' \in \tau'^B$ de la seconde, si $x \xmapsto[\tau]{\varphi} x'$, où la famille de relations :

$$\{\xmapsto[\tau]{\varphi} \;\subseteq\; \tau^A \times \tau'^B \mid \tau \in \mathcal{T}(S), \;\tau \sim_\psi \tau'\}$$

TAB. 6. La relation logique $\{\xmapsto[\tau]{\varphi}\}$

$$\frac{\pi \in \mathcal{B}}{x \xmapsto[\pi]{\varphi} x}\ \text{(base)} \qquad\qquad \frac{\sigma \in S}{x \xmapsto[\sigma]{\varphi} \varphi_\sigma(x)}\ \text{(sort)}$$

$$\frac{x_1 \xmapsto[\tau_1]{\varphi} x_1 \quad x_2 \xmapsto[\tau_2]{\varphi} x_2}{(x_1, x_2) \xmapsto[\tau_1 \times \tau_2]{\varphi} (y_1, y_2)}\ \text{(prod)} \qquad\qquad \frac{x \xmapsto[\tau_1]{\varphi} y \ \vee\ x \xmapsto[\tau_2]{\varphi} y}{x \xmapsto[\tau_1 \oplus \tau_2]{\varphi} y}\ \text{(sum)}$$

$$\frac{\forall x \in X.\, \exists y \in Y.\, x \xmapsto[\tau]{\varphi} y \quad \forall y \in Y.\, \exists x \in X.\, x \xmapsto[\tau]{\varphi} y}{X \xmapsto[\wp^\tau]{\varphi} Y}\ \text{(set)} \qquad\qquad \frac{X \xmapsto[\wp^\tau]{\varphi} Y}{X \xmapsto[\wp^\tau_F]{\varphi} Y}\ \text{(fset)}$$

$$\frac{u \xmapsto[\mathbb{N}_+ \mathrel{-\!\!\circ\!\!\!\to} \tau]{\varphi} v}{u \xmapsto[\tau^\infty]{\varphi} v}\ \text{(seq)} \qquad\qquad \frac{u \xmapsto[\mathbb{N}_+ \mathrel{-\!\!\circ\!\!\!\to} \tau]{\varphi} v}{u \xmapsto[\tau^*]{\varphi} v}\ \text{(fseq)}$$

$$\frac{\forall x \in \tau_1^A,\ x' \in \tau_1^B.\quad x \xmapsto[\tau_1]{\varphi} x' \ \Rightarrow\ f(x) \xmapsto[\tau_2]{\varphi} g(x')}{f \xmapsto[\tau_1 \to \tau_2]{\varphi} g}\ \text{(fun)}$$

$$\frac{\forall x \in \mathrm{dom}(f).\quad x \xmapsto[\tau_1]{\varphi} x' \ \Rightarrow\ x' \in \mathrm{dom}(g) \wedge f(x) \xmapsto[\tau_2]{\varphi} g(x')}{f \xmapsto[\tau_1 \mathrel{-\!\!\circ\!\!\!\to} \tau_2]{\varphi} g}\ \text{(pfun)}$$

$$\frac{\forall x \in \chi.\quad p_{(x^A)} \xmapsto[\tau[{}^x\!/\!_\alpha]]{\varphi} p'_{(x^B)}}{p \xmapsto[\forall \alpha : \chi .\, \tau]{\varphi} p'}\ \text{(poly)}$$

$$\frac{\forall x \in \chi.\quad p_{(x^A)} \xmapsto[\tau[{}^x\!/\!_\alpha]]{\varphi} p'_{(x^B)} \quad \text{où}\ \ \tau = \mathrm{cond}(\vartheta, \tau_1, \tau_2)(x)}{p \xmapsto[\forall \alpha : \chi .\, \mathsf{if}\ \alpha = \vartheta\ \mathsf{then}\ \tau_1\ \mathsf{else}\ \tau_2]{\varphi} p'}\ \text{(cond)}$$

$$\frac{p \xmapsto[\forall \alpha : \chi .\, \tau]{\varphi} p'' \qquad p'' = \chi^B_\psi \times \{p'\}}{p \xmapsto[\forall \alpha : \chi .\, \tau]{\varphi} p'}\ \text{(const)}$$

est la plus petite famille engendrée par le système d'inférence de la Table 6.

Le système logique est *déterministe*, autrement dit, il existe tout au plus une règle applicable pour toute assertion de la forme $x \xmapsto[\tau]{\varphi} x'$. La règle (base) indique que tout objet d'un type de base, par exemple un naturel $n \in \mathbb{N}$, se conserve en lui-même, indépendamment de l'interprétation (ψ, φ) impliquée. La règle (sort) affirme que tout objet d'une sorte est conservé dans son image via φ. Les règles (prod) et (sum) ont un trait commun : elles renvoient la conservation d'un objet à celle des composantes. Pour la règle (set), qui est proposée dans [**Honsell & Sanella, 1999**], deux ensembles sont en relation si tout élément de l'un trouve un correspondant dans l'autre. La règle (fset), pour les ensembles finis, renvoie à celle, générale, des ensembles arbitraires. La règle (fun), calquée sur la définition de *relation logique* (dans, par exemple, [**Mitchell, 1996**] et [**Honsell & Sanella, 1999**]), exige que les fonctions f et g transforment des arguments correspondants en résultats correspondants pour qu'elles soient, elles-mêmes, correspondantes. La règle (pfun) étend la règle (fun) au cas des fonctions partielles : la quantification universelle se fait sur le domaine de la première fonction, et il est requis que tout élément en correspondance soit un argument possible pour la seconde fonction partielle. Puis, comme pour les fonctions totales, il faudra que les résultats soient, eux aussi, en correspondance. La règle (pfun) est donc équivalente à la règle (fun) lorsque les fonctions sont totales. Les règles (seq) et (fseq) pour les séquences (arbitraires ou finies) renvoient à la règle des fonctions partielles. Par ce choix, une séquence s se conservant en s', se conservera aussi dans toutes les séquences commençant par s'. Autrement dit, la conservation des séquences étend le sens de la relation *préfixe* entre mots sur un alphabet. En l'affirmant on n'oubliera pas, bien entendu, que les séquences appartiennent à deux mondes différents, le passage d'un monde à l'autre se faisant par l'intermédiaire d'une interprétation des valeurs. En ce qui concerne les constructions polymorphes, la règle (poly) indique que deux constructions polymorphes p et p' sont correspondantes lorsque pour tout argument x (domaine de base ou sorte) appartenant à la classe χ, les deux résultats $p_{(x^A)}$ et $p'_{(x^B)}$ sont, eux-mêmes, en correspondance sur le type τ. Ce dernier est le type polymorphe spécialisé par x. La règle (cond) discipline les constructions polymorphes munies d'un test d'égalité. Comme pour la règle (poly), pour tout $x \in \chi$, il faudra que $p_{(x^A)}$ et $p'_{(x^B)}$ soient correspondantes. Mais la correspondance devra avoir lieu sur un type τ dépendant de x, résultat du test d'égalité $\mathrm{cond}(\vartheta, \tau_1, \tau_2)(x)$. Et il sera ensuite spécialisé par la substitution $\tau[{}^{x}/_{\alpha}]$. Enfin, la règle (const) joue le même rôle que la règle homonyme pour les types compatibles. Elle nous permet de considérer toute opération p' de la seconde algèbre comme une construction polymorphe constante :

$$p'' \;=\; \lambda x {\in} \chi^B_\psi . p' \;=\; \{(x, p') \mid x {\in} \chi^B_\psi\} \;=\; \chi^B_\psi \times \{p'\}$$

Il est important de souligner que, par définition de la relation de conservation, deux objets x et x' peuvent être correspondants pour une interprétation (ψ, φ) donnée, seulement si dans leur algèbre respective, les deux objets appartiennent à des domaines compatibles :

$$x \xmapsto[\tau]{\varphi} x' \quad \Rightarrow \quad x \in \tau^A \;\wedge\; x' \in \tau'^B \;\wedge\; \tau \sim_\psi \tau'$$

1.3.4. Homomorphismes. Une traduction ψ des sortes et des opérations, couplée d'une traduction φ des valeurs, a le statut d'*interprétation* lorsque les opérations correspondantes $f' = \psi(f)$ sont *compatibles*. Pour que le couple de traductions (ψ, φ) ait le statut supérieur d'*homomorphisme*, la condition de *conservation* des opérations doit s'ajouter à la condition, toujours nécessaire, de compatibilité.

Un *homomorphisme* de A dans B, noté $(\psi, \varphi) : A \xmapsto{\mathcal{H}} B$, est une interprétation (ψ, φ) qui conserve toute opération $f^A \in F^A$ de l'algèbre de départ dans l'opération correspondante $f^B_\psi \in F'^B$:

$$\forall f \in F. \quad f^A \underset{T(f)}{\overset{\varphi}{\longmapsto}} f^B_\psi.$$

EXAMPLE 1.3.2. Considérons à nouveau l'exemple 1.3.1 où, d'une part, l'algèbre A, constituée des sortes $S = \{\mathbb{Z}, \mathbb{B}\}$, où $\mathbb{B} = \{tt, ff\}$, se trouve équipée de l'addition polymorphe $F = \{+_{(.)} : \forall\alpha : \{\mathbb{Z},\mathbb{B}\}.\, \alpha \to \alpha \to \alpha\}$; et, d'autre part, l'algèbre B a comme unique sorte celle des nombres réels $S' = \{\mathbb{R}\}$, et comme unique opération (monomorphe) l'addition entre réels $F' = \{+ : \mathbb{R} \to \mathbb{R} \to \mathbb{R}\}$. Nous avons vu que l'interprétation des noms est forcément $\psi(\mathbb{Z}) = \psi(\mathbb{B}) = \mathbb{R}$ et $\psi(+_{(.)}) = +$. Si nous considérons d'interpréter les valeurs entières $z \in \mathbb{Z}$ avec la fonction identité $\varphi(z) = z$, et les valeurs booléennes avec la fonction $\varphi(ff) = 0$, $\varphi(tt) = 1$, nous obtenons une *interprétation* de l'algèbre A dans l'algèbre B. En effet, la seule condition requise pour le statut d'interprétation, la compatibilité des opérations, est garantie et ne concerne pas la traduction φ des valeurs. Mais si les deux opérations allogènes sont compatibles, elle ne sont pas pour autant correspondantes au sens de la relation de conservation. L'instance $+_\mathbb{B}$ de l'addition polymorphe ne peut être conservée dans l'addition des réels. Alors que tt correspond à 1, le résultat de l'addition $tt +_\mathbb{B} tt = tt$ ne correspond pas à $2 = 1 + 1$. Autrement dit, l'interprétation définie n'est pas un homomorphisme. En revanche, considérant comme opération de l'algèbre A la multiplication polymorphe $F = \{*_{(.)} : \forall\alpha : \{\mathbb{Z},\mathbb{B}\}.\, \alpha \to \alpha \to \alpha\}$, où $*_\mathbb{N}$ serait le produit de nombres entiers et $*_\mathbb{B}$ serait le *et logique* ; et considérant l'algèbre B constituée de la seule multiplication entre réels $F' = \{\cdot : \mathbb{R} \to \mathbb{R} \to \mathbb{R}\}$, l'interprétation des valeurs définies ci-dessus constituerait, cette fois-ci, un homomorphisme de A vers B. □

1.3.4.1. *Extension de l'interprétation des valeurs aux types élémentaires.* La notion d'homomorphisme que nous avons définie généralise les définitions traditionnelles de l'algèbre universelle. Pour le montrer d'une façon formelle, nous allons étendre l'interprétation des valeurs à tous les domaines correspondant à un type élémentaire. Nous définissons la famille de fonctions $\{\vec{\varphi}_\tau : \tau^A \to \tau^B_\psi \mid \tau \in \mathcal{T}_\mathcal{E}(S)\}$ par induction sur les types élémentaires (dont la syntaxe, nous le rappelons, est $\tau ::= \pi \mid \sigma \mid \wp^\tau \mid \wp^\tau_F \mid \tau \times \tau \mid \tau \oplus \tau$). Pour les ensembles de base, nous utiliserons l'identité : $\vec{\varphi}_\pi(x) \triangleq x$. Pour les sortes, nous utiliserons l'interprétation des valeurs : $\vec{\varphi}_\sigma(x) \triangleq \varphi_\sigma(x)$. Pour les parties (finies ou infinies) nous construirons l'image : $\vec{\varphi}_{\wp^\tau_F}(X) = \vec{\varphi}_{\wp^\tau}(X) \triangleq \{\vec{\varphi}_\tau(x) \mid x \in X\}$. Enfin, pour les produits cartésiens et les sommes, nous utiliserons la décomposition habituelle qui applique les composantes à des fonctions de la même famille mais de type plus simple : $\vec{\varphi}_{\tau_1 \times \tau_2}(x_1, x_2) \triangleq (\vec{\varphi}_{\tau_1}(x_1), \vec{\varphi}_{\tau_2}(x_2))$, et : $\vec{\varphi}_{\tau_1 \oplus \tau_2}(x) \triangleq \vec{\varphi}_{\tau_1}(x)$ si $x \in \tau_1^A$, sinon $\vec{\varphi}_{\tau_1 \oplus \tau_2}(x) \triangleq \vec{\varphi}_{\tau_2}(x)$. Nous noterons $\vec{\varphi}$ la construction polymorphe définie ci-dessus :

$$\vec{\varphi} : \forall\alpha : \mathcal{T}_\mathcal{E}(S).\, \alpha^A \to \alpha^B_\psi$$

Nous omettrons le type en indice, en écrivant $\vec{\varphi}(x)$ au lieu de $\vec{\varphi}_\tau(x)$, lorsque le type τ sera implicite.

LEMMA 1.3.3. *Pour tout type élémentaire $\tau \in \mathcal{T}_{\mathcal{E}}(S)$, la condition de conservation de $x \in \tau^A$ dans $y \in \tau^B_\psi$ via une interprétation des valeurs φ, équivaut à l'équation $\vec{\varphi}_\tau(x) = y$:*

$$x \overset{\varphi}{\underset{\tau}{\longmapsto}} y \quad \Leftrightarrow \quad \vec{\varphi}_\tau(x) = y$$

DÉMONSTRATION. *Les types élémentaires sont engendrés par la syntaxe $\tau ::= \pi \,|\, \sigma \,|\, \wp^\tau \,|\, \wp^\tau_F \,|\, \tau \times \tau \,|\, \tau \oplus \tau$. L'énoncé est trivial pour les types de base $\tau = \pi$ et pour les sortes $\tau = \sigma$. Pour les parties $\tau = \wp^{\tau'}$ ou $\tau = \wp^{\tau'}_F$, en utilisant l'hypothèse d'induction, les prémisses de la règle (set) deviennent $\forall a \in x. \exists b \in y.\ b = \vec{\varphi}_{\tau'}(a)$ et réciproquement $\forall b \in y. \exists a \in x.\ b = \vec{\varphi}_{\tau'}(a)$, autrement dit $\vec{\varphi}_\tau(x) \subseteq y$ et $\vec{\varphi}_\tau(x) \supseteq y$, donc l'égalité. Pour le produit et la somme nous utiliserons le même type d'argument : l'induction sur les types et le déterminisme du système d'inférence (pas plus qu'une règle applicable).* □

La notion d'homomorphisme définie ici s'applique aux cas particuliers des fonctions à valeur dans un produit de sortes et qui renvoient une valeur incluse encore dans une sorte. Le corollaire suivant nous permet de retrouver l'équation habituellement énoncée pour ce type d'opération algébrique.

COROLLARY 1.3.4. *Soit $f : \tau_1 \to \tau_2$ et $g : \tau_{1_\psi} \to \tau_{2_\psi}$ où τ_1 et τ_2 sont des types élémentaires $\tau_1, \tau_2 \in \mathcal{T}_{\mathcal{E}}(S)$. Alors :*

$$f \overset{\varphi}{\underset{\tau_1 \to \tau_2}{\longmapsto}} g \quad \Leftrightarrow \quad \forall \vec{x} \in \tau_1^A.\ \ \vec{\varphi}_{\tau_2}(f(\vec{x})) = g(\vec{\varphi}_{\tau_1}(\vec{x}))$$

DÉMONSTRATION. *Le domaine de f correspondant à un type élémentaire $\tau_1 \in \mathcal{T}_{\mathcal{E}}(S)$, la condition $\vec{x} \overset{\varphi}{\underset{\tau_1}{\longmapsto}} \vec{x}'$ sera équivalente, pour tout argument $\vec{x}$ de f et tout argument $\vec{x}'$ de g, à l'équation $\vec{\varphi}_{\tau_1}(\vec{x}) = \vec{x}'$. De même, la conservation des résultats des applications $f(\vec{x}) \overset{\varphi}{\underset{\tau_2}{\longmapsto}} g(\vec{x}')$ sera équivalente à l'équation $\varphi_{\tau_2}(f(\vec{x})) = g(\vec{x}')$. En remplaçant, dans cette dernière, $\vec{x}'$ par $\vec{\varphi}_{\tau_1}(\vec{x})$, nous avons l'énoncé.* □

1.3.4.2. *Curryfication.* La *curryfication* est une opération qui permet de définir une fonction $\hat{f} : \tau_1 \to \tau_2 \to \tau_3$ pour toute fonction de type $f : \tau_1 \times \tau_2 \to \tau_3$: $\hat{f}\,x\,y \triangleq f(x, y)$. Dans le λ-calcul simple (non typé) de Alonzo Church (cf. par exemple [**Barendregt, 1984**]), où le type produit n'existe pas, il s'agit de la façon courante pour définir une fonction qui sera ensuite utilisée *comme si* elle prenait en argument un couple (ou plus généralement un n-uplet). La conservation d'une fonction et celle de sa version *curryfiée* sont des conditions équivalentes.

LEMMA 1.3.5. *Soient $f : \tau_1 \times \tau_2 \to \tau_3$ et $g : \tau_{1_\psi} \times \tau_{2_\psi} \to \tau_{3_\psi}$. Alors f se conserve en g si et seulement si la curryfication de f se conserve dans celle de g :*

$$f \overset{\varphi}{\underset{\tau_1 \times \tau_2 \to \tau_3}{\longmapsto}} g \quad \Leftrightarrow \quad \hat{f} \overset{\varphi}{\underset{\tau_1 \to \tau_2 \to \tau_3}{\longmapsto}} \hat{g}$$

DÉMONSTRATION. *(⇒) Supposons* $x \overset{\varphi}{\underset{\tau_1}{\longmapsto}} x'$. *Il nous faut démontrer* $\hat{f}x \overset{\varphi}{\underset{\tau_2\to\tau_3}{\longmapsto}} \hat{g}x'$. *Si l'on suppose aussi* $y \overset{\varphi}{\underset{\tau_2}{\longmapsto}} y'$, *il nous faudra prouver* $(\hat{f}x)y \overset{\varphi}{\underset{\tau_2\to\tau_3}{\longmapsto}} (\hat{g}x')y'$. *Par définition* $(\hat{f}x)y = f(x,y)$ *et* $(\hat{g}x')y' = g(x',y')$. *Puisque* $x \overset{\varphi}{\underset{\tau_1}{\longmapsto}} x'$ *et* $y \overset{\varphi}{\underset{\tau_2}{\longmapsto}} y'$ *nous avons* $(x,y) \overset{\varphi}{\underset{\tau_1\times\tau_2}{\longmapsto}} (x',y')$ *et donc, puisque* f *se conserve en* g, *nous avons* $f(x,y) \overset{\varphi}{\underset{\tau_3}{\longmapsto}} g(x',y')$, c.q.f.d. *(⇐) Supposons* $(x,y) \overset{\varphi}{\underset{\tau_1\times\tau_2}{\longmapsto}} (x',y')$. *Nous devons montrer que* $f(x,y) \overset{\varphi}{\underset{\tau_3}{\longmapsto}} g(x',y')$. *Si le couple* (x,y) *se conserve en* (x',y'), *alors les composantes correspondantes se conservent :* $x \overset{\varphi}{\underset{\tau_1}{\longmapsto}} x'$ *et* $y \overset{\varphi}{\underset{\tau_2}{\longmapsto}} y'$. *Par hypothèse nous avons* $\hat{f}x \overset{\varphi}{\underset{\tau_2\to\tau_3}{\longmapsto}} \hat{g}x'$, *donc* $(\hat{f}x)y \overset{\varphi}{\underset{\tau_3}{\longmapsto}} (\hat{g}x')y'$, c.q.f.d. □

1.3.4.3. *Homomorphismes de propriétés.* Les trois façons possibles de considérer une propriété, comme une *partie* $P : \wp^\tau$, comme un *prédicat* $P : \tau \to 2$ (nous noterons $P : 2^\tau$), ou bien comme un *semi-prédicat* $P : \tau \multimap 1$ (nous noterons $P : 1^\tau$), correspondent à trois notions différentes de conservation. En particulier, la conservation des semi-prédicats coïncide avec celle que l'on retrouve en littérature pour les prédicats (par exemple dans [**Gericke, 1966**] et [**Mal'cev, 1973**]). Autrement dit, tout en étant des fonctions à plein titre $P : \tau \to 2$, la condition prévue pour les fonctions, c'est-à-dire $\varphi(P(x)) = Q(\varphi(x))$, n'est pas celle demandée aux prédicats. Celle qui leur est demandée, en revanche, est une condition plus faible, $\varphi(P(x)) = 1 \Rightarrow Q(\varphi(x)) = 1$, qui est précisément, dans notre formalisme, celle des semi-prédicats.

LEMMA 1.3.6. *Soit* τ *un type élémentaire. Alors :*

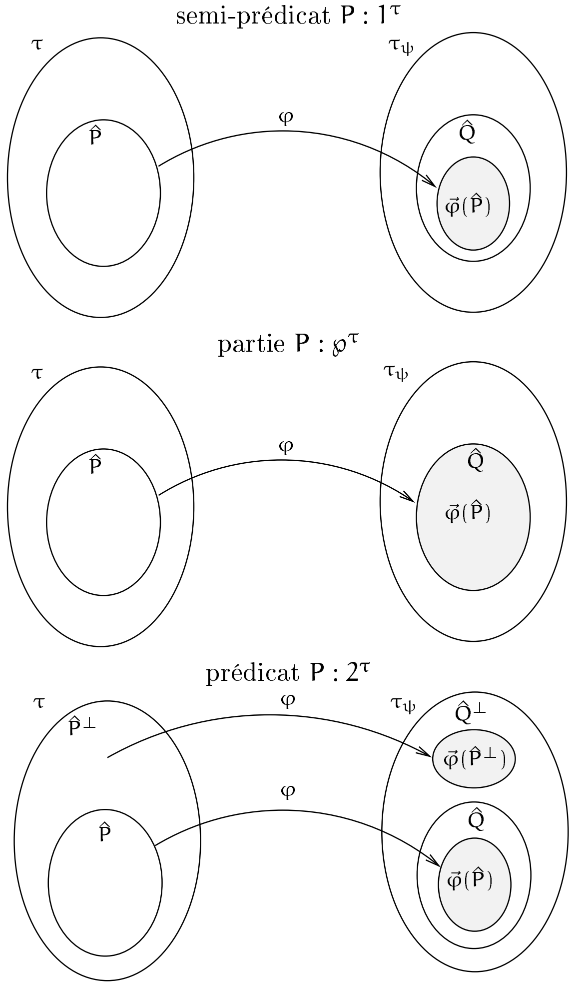
(1) $P \overset{\varphi}{\underset{1^\tau}{\longmapsto}} Q \quad \Leftrightarrow \quad \forall x \in \tau^A.\ P(x) \Rightarrow Q(\vec{\varphi}_\tau(x))$

(2) $P \overset{\varphi}{\underset{2^\tau}{\longmapsto}} Q \quad \Leftrightarrow \quad \forall x \in \tau^A.\ P(x) \Rightarrow Q(\vec{\varphi}_\tau(x)) \ \wedge\ \neg P(x) \Rightarrow \neg Q(\vec{\varphi}_\tau(x))$

(3) $P \overset{\varphi}{\underset{\wp^\tau}{\longmapsto}} Q \quad \Leftrightarrow \quad \vec{\varphi}_{\wp^\tau}(P) = Q$

DÉMONSTRATION. *Le lemme 1.3.3 assure que, pour les types élémentaires, la condition* $x \overset{\varphi}{\underset{\tau}{\longmapsto}} x'$ *est équivalente à* $x' = \vec{\varphi}_\tau(x)$. *(1) (⇒) Si* $P \overset{\varphi}{\underset{1^\tau}{\longmapsto}} Q$ *est prouvable, la seule règle utilisable du système d'inférence de la Table est la (pfun). Si l'on suppose* $P(x)$, *c'est-à-dire* $x \in \mathrm{dom}(P)$, *puisque* $x \overset{\varphi}{\underset{\tau}{\longmapsto}} \vec{\varphi}_\tau(x)$, *nous avons* $\vec{\varphi}_\tau(x) \in \mathrm{dom}(Q)$. *(⇐) Soit* $x \in \mathrm{dom}(P)$ *et* $x \overset{\varphi}{\underset{\tau}{\longmapsto}} x'$. *Cela implique* $x' = \vec{\varphi}_\tau(x)$. *Il faut démontrer que* $x' \in \mathrm{dom}(Q)$ *et que* $P(x) \overset{\varphi}{\underset{1}{\longmapsto}} Q(x')$. *Par hypothèse* $Q(\vec{\varphi}_\tau(x))$ *donc* $Q(x')$. *D'autre part* $P(x) = Q(x') = \bullet$ *et, en appliquant la règle (base),* $\bullet \overset{\varphi}{\underset{1}{\longmapsto}} \bullet$. *Les preuves pour (2) et (3) s'appuient sur des arguments similaires.* □

FIG. 1.3.1. Conservation d'une propriété selon le type



REMARK 1.3.7. Considérant la dénotation ensembliste $\hat{\mathsf{P}}$ d'une propriété P (cf. section 1.1.6), nous pouvons écrire que la conservation des semi-prédicats correspond à la condition $\vec{\varphi}(\hat{\mathsf{P}}) \subseteq \hat{\mathsf{Q}}$, la conservation des prédicats à la conjonction $\vec{\varphi}(\hat{\mathsf{P}}) \subseteq \hat{\mathsf{Q}} \ \wedge \ \vec{\varphi}(\hat{\mathsf{P}})^\perp \subseteq \hat{\mathsf{Q}}^\perp$ (où $\mathsf{X}^\perp$ est le complémentaire de X), et la conservation des parties à l'équation $\vec{\varphi}(\hat{\mathsf{P}}) = \hat{\mathsf{Q}}$. La Figure 1.3.1 illustre graphiquement les trois formes de conservation d'une propriété de type élémentaire.

L'équation du lemme 1.3.3 pour les types élémentaires, permet d'observer des propriétés de conservation intéressantes pour des objets construits par *union*, *intersection*, *composition fonctionnelle* ou *composition de fonction et propriété* (cf. section 1.1.6). Considérant l'interprétation des valeurs comme le pont entre deux mondes en vis-à-vis, nous allons vérifier que si tout matériau utilisé d'un côté se conserve dans le matériau utilisé de l'autre côté, alors le résultat de la construction sur le premier versant se conservera dans celui d'en face.

PROPOSITION 1.3.8. *Soient* $\tau,\tau_1,\tau_2,\tau_3$ *des types élémentaires. Alors :*

(1) $X \xmapsto[1^\tau]{\varphi} X' \quad \wedge \quad Y \xmapsto[1^\tau]{\varphi} Y' \quad \Rightarrow \quad X \cap Y \xmapsto[1^\tau]{\varphi} X' \cap Y'$

(2) $X \xmapsto[1^\tau]{\varphi} X' \quad \wedge \quad Y \xmapsto[1^\tau]{\varphi} Y' \quad \Rightarrow \quad X \cup Y \xmapsto[1^\tau]{\varphi} X' \cup Y'$

(3) $X \xmapsto[\wp^\tau]{\varphi} X' \quad \wedge \quad Y \xmapsto[\wp^\tau]{\varphi} Y' \quad \Rightarrow \quad X \cup Y \xmapsto[\wp^\tau]{\varphi} X' \cup Y'$

(4) $f \xmapsto[\tau_1 \nrightarrow \tau_2]{\varphi} f' \quad \wedge \quad g \xmapsto[\tau_2 \nrightarrow \tau_3]{\varphi} g' \quad \Rightarrow \quad g \circ f \xmapsto[\tau_1 \nrightarrow \tau_3]{\varphi} g' \circ f'$

(5) $f \xmapsto[\tau_1 \nrightarrow \tau_2]{\varphi} f' \quad \wedge \quad p \xmapsto[\wp^\tau \oplus 2^\tau \oplus 1^\tau]{\varphi} p' \quad \Rightarrow \quad f_p \xmapsto[\tau_1 \nrightarrow \tau_2]{\varphi} f'_{p'}$

DÉMONSTRATION. *Par cas.*

- (Intersection et union) *Par la remarque 1.3.7, l'hypothèse du point (1) se traduit en* $\vec{\varphi}(\hat{X}) \subseteq \widehat{X'}$ *et* $\vec{\varphi}(\hat{Y}) \subseteq \widehat{Y'}$ *et l'énoncé en* $\vec{\varphi}(\hat{X} \cap \hat{Y}) \subseteq \widehat{X'} \cap \widehat{Y'}$*, ce qui est trivialement vrai puisque, par définition d'image et d'intersection, nous savons que* $\vec{\varphi}(\hat{X} \cap \hat{Y}) \subseteq \vec{\varphi}(\hat{X}) \cap \vec{\varphi}(\hat{Y})$. *De façon similaire, l'union ensembliste étant telle que* $\vec{\varphi}(X \cup Y) = \vec{\varphi}(\hat{X}) \cup \vec{\varphi}(\hat{Y})$*, nous avons les points (2) et (3).*
- (Composition fonctionnelle) *Pour montrer que* $g \circ f \xmapsto[\tau_1 \nrightarrow \tau_3]{\varphi} g' \circ f'$ *supposons que* $x \in \mathrm{dom}(g \circ f)$ *et que* $x \xmapsto[\tau_1]{\varphi} x'$*, c'est-à-dire (cf. lemme 1.3.3) que* $\vec{\varphi}(x) = x'$. *Nous voulons prouver que* $x' \in \mathrm{dom}(g' \circ f')$ *et que* $g \circ f(x) \xmapsto[\tau_3]{\varphi} g' \circ f'(x')$*, c'est-à-dire* $\vec{\varphi}(g \circ f(x)) = g' \circ f'(x')$. *Par hypothèse* $x \in \mathrm{dom}(f)$ *et* f *se conserve en* f'*, donc* $x' \in \mathrm{dom}(f')$. *De plus,* $f(x) \in \mathrm{dom}(g)$ *et* g *se conserve en* g'*, donc* $f'(x') \in \mathrm{dom}(g')$. *Autrement dit, d'une part* $x' \in \mathrm{dom}(g' \circ f')$*, et d'autre part* $\vec{\varphi}(g \circ f(x)) = \vec{\varphi}(g[f(x)]) = g'[\vec{\varphi}(f(x))] = g'[f'(\vec{\varphi}(x))] = g' \circ f'(\vec{\varphi}(x)) = g' \circ f'(x')$.
- (Composition de fonction et propriété) *Nous allons revenir au cas de la composition fonctionnelle. Par la remarque 1.3.7 nous avons* $\varphi(\hat{p}) \subseteq \hat{p}'$. *Supposer* $x \xmapsto[\tau_1]{\varphi} x'$ *implique alors que* $x' = \varphi(x)$*, donc* $x' \in \hat{p}'$. *Autrement dit,* x' *appartient au domaine de la fonction identité* $\mathrm{id}_{\hat{p}'} : {\tau_1}_\psi \nrightarrow {\tau_1}_\psi$: $x' \in \mathrm{dom}(\mathrm{id}_{\hat{p}'})$. *De plus, puisque* $x = \mathrm{id}_{\hat{p}}(x)$ *et* $x' = \mathrm{id}_{\hat{p}'}(x')$ *nous avons* $\mathrm{id}_{\hat{p}}(x) \xmapsto[\tau_1]{\varphi} \mathrm{id}_{\hat{p}'}(x')$. *En d'autres termes, l'hypothèse* $p \xmapsto[\wp^\tau \oplus 2^\tau \oplus 1^\tau]{\varphi} p'$ *implique* $\mathrm{id}_{\hat{p}} \xmapsto[\tau_1 \nrightarrow \tau_1]{\varphi} \mathrm{id}_{\hat{p}'}$. *La composition* $f_p = f_{|\hat{p}}$ *étant égale à la composition fonctionnelle* $f \circ \mathrm{id}_{\hat{p}}$*, nous pouvons appliquer le point (4).*

□

1.3.4.4. *Cas particuliers.* Nous allons spécialiser la condition de conservation $x \xmapsto[\tau]{\varphi} y$ entre objets de deux algèbres, pour quelques cas spécifiques.

(1) Si $c :\to \sigma$ et $c' :\to \sigma_\psi$ sont deux constantes sur une sorte, alors :

$$c \xmapsto[\to \sigma]{\varphi} c' \quad \Leftrightarrow \quad \varphi_\sigma(c()) = c'()$$

(2) Si $\leq : 2^{\sigma \times \sigma}$ et $\leq' : 2^{\sigma_\psi \times \sigma_\psi}$ sont deux *prédicats* binaires sur une sorte, alors :

$$(\leq) \xmapsto[2^{\sigma\times\sigma}]{\varphi} (\leq') \quad \Leftrightarrow \quad \begin{cases} x \leq y \;\Rightarrow\; \varphi_\sigma(x) \leq' \varphi_\sigma(y) \\ x \not\leq y \;\Rightarrow\; \varphi_\sigma(x) \not\leq' \varphi_\sigma(y) \end{cases}$$

(3) Si $\leq : \wp^{\sigma\times\sigma}$ et $\leq' : \wp^{\sigma_\psi\times\sigma_\psi}$ sont deux *relations* binaires sur une sorte, alors :

$$(\leq) \xmapsto[\wp^{\sigma\times\sigma}]{\varphi} (\leq') \quad \Leftrightarrow \quad \forall x', y'.\, x' \leq' y' \Leftrightarrow \exists x, y. \begin{cases} x' = \varphi_\sigma(x) \\ y' = \varphi_\sigma(y) \\ x \leq y \end{cases}$$

(4) Si $\leq : 1^{\sigma\times\sigma}$ et $\leq' : 1^{\sigma_\psi\times\sigma_\psi}$ sont deux semi-prédicats binaires sur une sorte, alors :

$$(\leq) \xmapsto[1^{\sigma\times\sigma}]{\varphi} (\leq') \quad \Leftrightarrow \quad x \leq y \;\Rightarrow\; \varphi_\sigma(x) \leq' \varphi_\sigma(y)$$

(5) Si $u : \sigma^*$ et $v : \sigma_\psi^*$ sont des suites sur une sorte, alors :

$$u \xmapsto[\sigma^*]{\varphi} v \quad \Leftrightarrow \quad \forall i \in \mathrm{dom}(u).\; \varphi_\sigma(u_i) = v_i$$

1.3.5. Isomorphismes. On dit qu'un homomorphisme noté $(\psi, \varphi) : A \xmapsto{\mathcal{H}} B$, est un *isomorphisme*, noté $A \overset{\psi,\varphi}{\cong} B$, si ψ et φ sont bijectives (chacune de leurs fonctions composantes est bijective) et l'interprétation $(\psi^{-1}, \varphi^{-1})$ est, elle aussi, un homomorphisme : $(\psi^{-1}, \varphi^{-1}) : B \xmapsto{\mathcal{H}} A$.

NOTATION 1.3.9. Si les deux algèbres coïncident et qu'on considère la fonction identité comme bijection ψ, on dit qu'un homomorphisme est un *endomorphisme* et qu'un isomorphisme est un *automorphisme*. Quand les noms expriment, par eux-mêmes, l'association ψ, ou bien quand le type des opérations des deux algèbres oblige à une seule association possible, ou bien encore, quand les deux algèbres coïncident et qu'on prend la bijection identique, nous sous-entendrons l'application ψ. Ainsi, nous nous autorisons à noter $\varphi : A \xmapsto{\mathcal{H}} B$ et à dire que la construction polymorphe φ est, elle-même, un homomorphisme, un isomorphisme, un endomorphisme ou un automorphisme. Nous noterons l'existence d'un isomorphisme entre deux algèbres A et B par l'écriture $A \cong B$.

1.4. Restrictions et extensions

La notion d'isomorphisme peut être reconduite à la coïncidence de deux phénomènes orthogonaux : chaque algèbre d'un isomorphisme est une *restriction* et, à la fois, une *extension* de l'autre algèbre. En mathématiques, il est souvent utile, après avoir défini une structure, de définir également ce qu'est une *sous-structure* de la première. Ainsi, par exemple, un *monoïde* est une structure composée d'un ensemble M, d'une opération de *composition* $* : M \times M \to M$ (qui est associative, i.e. $x * (y * z) = (x * y) * z$ pour tout x, y et z) et d'un *élément neutre* 0 (tel que $0 * x = x * 0 = x$ pour tout x). Un *sous-monoïde* du monoïde $(M, *, 0)$ est un sous-ensemble de M qui est encore un monoïde (pour les mêmes opérations). De la même façon, un *sous-ordre partiel* est un sous-ensemble d'un ordre partiel qui est encore un ordre partiel. Et en théorie des catégories, une *sous-catégorie* est un *sous-graphe* qui est encore une catégorie (c'est-à-dire que l'on peut encore composer les flèches et que toutes les flèches identité sont présentes). Dans ces trois exemples, on retrouve la conjonction de deux exigences, celle qui prétend un rapport d'inclusion (sous-ensemble, sous-graphe), et celle qui prétend ne pas perdre les propriétés caractérisant la structure de départ.

1.4.1. Sous-algèbres. Étant donné une algèbre $A = ((.)^A, T)$, une autre algèbre $A' = ((.)^{A'}, T')$ est une *restriction algébrique homomorphe* ou *sous-algèbre homomorphe* de A si :

$$\exists(\psi, \varphi) : A' \overset{\mathcal{H}}{\longmapsto} A \text{ telles que } \psi \text{ et } \varphi \text{ sont } \textit{injectives}$$

Les fonctions ψ et φ sont injectives si toutes leurs composantes sont injectives. Nous noterons une telle condition par l'écriture $A' \overset{\psi,\varphi}{\leqq} A$ ou plus simplement (lorsque ψ sera implicite) par $A' \overset{\varphi}{\leqq} A$, ou encore (lorsque les deux fonctions seront implicites) par $A' \leqq A$.

En particulier, lorsque l'interprétation des valeurs φ coïncide avec l'*identité* (pour toutes ses composantes), nous dirons que A' est une *restriction algébrique* ou *sous-algèbre* de A, notée $A' \subseteqq A$.

Remark 1.4.1. Une sous-algèbre A' de A correspond au choix d'un sous-ensemble pour tout domaine de A correspondant à une sorte ; ce qui justifie la notation $A' \subseteqq A$ évoquant l'inclusion ensembliste. Mais ce choix n'est pas arbitraire : il se doit de préserver les nouvelles opérations dans les anciennes. Par exemple, le domaine des réels avec la fonction qui multiplie par 2 son argument $f(x) = 2x$ est une algèbre $(\mathbb{R}, f : \mathbb{R} \to \mathbb{R})$ dont $(\mathbb{N}, f : \mathbb{N} \to \mathbb{N})$ est une restriction algébrique. En revanche, aucune sous-algèbre de signature $(\mathbb{N}, f : \mathbb{N} \to \mathbb{N})$ ne peut exister pour l'algèbre $(\mathbb{R}, f : \mathbb{R} \to \mathbb{R})$ si la fonction $f(x) = x/2$, cette fois-ci, divise par 2 son argument.

Il est simple de vérifier que toute sous-algèbre homomorphe A' de A est isomorphe à une sous-algèbre A'' de A :

$$A' \overset{\varphi}{\leqq} A \quad \Rightarrow \quad \exists A''.\ A'' \subseteqq A \ \wedge \ A' \overset{\varphi}{\cong} A''$$

Ainsi, il est courant de traduire la condition $A' \leqq A$ en affirmant que A *contient une image isomorphe de* A'. La notation $A' \leqq A$ souligne la condition d'infériorité, l'inclusion à un isomorphisme près, de l'algèbre A' par rapport à l'algèbre A.

Example 1.4.2. La définition classique de *sous-monoïde* M' d'un monoïde M se traduit exactement en celle de *sous-algèbre* : M' est un sous-monoïde de M si et seulement si $(M', *' : M' \times M' \to M', 0' :\to M')$ est une sous-algèbre de $(M, * : M \times M \to M, 0 :\to M)$. En effet, puisque l'homomorphisme φ concerné est l'identité, il faudra, d'une part, que $0' = 0$, et que $\varphi(x *' y) = \varphi(x) * \varphi(y)$, c'est-à-dire $x *' y = x * y$. Ainsi les deux nouvelles opérations hériteront forcément les propriétés des anciennes (associativité, élément neutre). $\square$

L'exemple précédent montre que les deux conditions souhaitées, l'inclusion $(M' \subseteq M)$ et la conservation du caractère initial (M' monoïde), se résument dans la *seule* condition de sous-algèbre. Ceci n'exclut pas que l'on puisse procéder différemment et définir une sous-structure de la façon habituelle : comme une sous-algèbre ayant le même caractère que l'algèbre la contenant. Une première raison sera liée à l'incapacité du métalangage algébrique à exprimer ce que l'on a appelé caractère de l'algèbre de départ (monoïde, ordre partiel, catégorie). Une seconde raison, est que, même lorsque le caractère pourra être codé dans une définition adéquate des opérations et de leurs types, il restera préférable, pour ne pas alourdir la définition de l'algèbre, d'énoncer la notion de sous-algèbre de façon classique. Dans ce dernier cas, nous utiliserons la notion de sous-algèbre comme une forme étendue d'inclusion ensembliste.

1.4.2. Extensions. Étant donné une algèbre $A = ((.)^A, T^A)$, une autre algèbre $A' = ((.)^{A'}, T^{A'})$ est une *extension algébrique homomorphe* de A si :

$$\exists(\psi, \varphi) : A' \overset{\mathcal{H}}{\longmapsto} A \text{ telles que } \psi \text{ et } \varphi \text{ sont } \textit{surjectives}$$

Les fonctions ψ et φ sont surjectives si toutes les composantes de leurs germes sont surjectives. À nouveau, une telle condition sera notée, en fonction du contexte, par l'écriture $A' \overset{\psi,\varphi}{\geqq} A$, ou bien par $A' \overset{\varphi}{\geqq} A$, ou encore plus simplement par $A' \geqq A$.

1.5. Interprétation catégorique

Nous allons conclure ce chapitre en justifiant, par sa connotation catégorique, le suffixe *morphisme* des mots homomorphisme et isomorphisme dont nous avons étendu le sens au cadre des algèbres typées. Étant donné un homomorphisme entre deux algèbres $(\psi, \varphi) : A \overset{\mathcal{H}}{\longmapsto} B$, et un second $(\psi', \varphi') : B \overset{\mathcal{H}}{\longmapsto} C$, nous voulons construire un nouvel homomorphisme entre la première algèbre A et la dernière C. Afin d'obtenir une catégorie, nous demandons qu'une telle composition soit associative. La construction plus naturelle, celle qu'on obtient par la composition fonctionnelle des fonctions d'interprétation $(\psi' \circ \psi, \varphi' \circ \varphi)$ est, certes, une interprétation des valeurs de A dans C, mais n'est pas forcément un homomorphisme. Nous allons vérifier qu'il s'agit d'une conséquence importante de la généralisation de la notion d'homomorphisme à des fonctions d'ordre supérieur $(\tau_1 \to \tau_2) \to \tau_3$.

LEMMA 1.5.1. *Pour tout couple d'interprétations $\psi_1, \varphi_2 : A \overset{\mathcal{H}}{\longmapsto} B$ et $\psi_2, \varphi_2 : B \overset{\mathcal{H}}{\longmapsto} C$ entre algèbres, dont la première a pour signature $\Sigma_A = (\mathcal{B}, S, F, T)$, pour tout type élémentaire $\tau \in \mathcal{T}_{\mathcal{E}}(S)$, et pour tout objet $x \in \tau^A$ nous avons l'équation $\overrightarrow{\varphi_2 \circ \varphi_1}(x) = (\vec{\varphi}_2 \circ \vec{\varphi}_1)(x)$.*

DÉMONSTRATION. *Par induction sur la syntaxe des types élémentaires qui est $\mathcal{T}_{\mathcal{E}}(S) \ni \tau ::= \pi \,|\, \sigma \,|\, \wp^{\tau} \,|\, \wp^{\tau}_{F} \,|\, \tau \times \tau \,|\, \tau \oplus \tau$ où $\pi \in \mathcal{B}$ et $\sigma \in S$.*

- *($\tau = \pi$)* $\overrightarrow{\varphi_2 \circ \varphi_1}(x) = x = \vec{\varphi}_2(x) = \vec{\varphi}_2(\vec{\varphi}_1(x)) = (\vec{\varphi}_2 \circ \vec{\varphi}_1)(x)$
- *($\tau = \sigma$)* $\overrightarrow{\varphi_2 \circ \varphi_1}(x) = (\varphi_2 \circ \varphi_1)(x) = \varphi_2(\varphi_1(x)) = \vec{\varphi}_2(\vec{\varphi}_1(x)) = (\vec{\varphi}_2 \circ \vec{\varphi}_1)(x)$
- *($\tau = \wp^{\tau'}$ ou $\tau = \wp^{\tau'}_{F}$) En appliquant l'hypothèse inductive sur les éléments constituant l'ensemble :* $\overrightarrow{\varphi_2 \circ \varphi_1}(X) = \{\overrightarrow{\varphi_2 \circ \varphi_1}(x) \mid x \in X\} = \{(\vec{\varphi}_2 \circ \vec{\varphi}_1)(x) \mid x \in X\} = (\vec{\varphi}_2 \circ \vec{\varphi}_1)(X)$

– $(\tau = \tau_1 \times \tau_2)$ *En appliquant l'hypothèse inductive sur les éléments constituant le couple :* $\overrightarrow{\varphi_2 \circ \varphi_1}(x_1, x_2) = (\overrightarrow{\varphi_2 \circ \varphi_1}(x_1), \overrightarrow{\varphi_2 \circ \varphi_1}(x_2)) = ((\vec{\varphi}_2 \circ \vec{\varphi}_1)(x_1), (\vec{\varphi}_2 \circ \vec{\varphi}_1)(x_2)) = (\vec{\varphi}_2 \circ \vec{\varphi}_1)(x_1, x_2)($

– $(\tau = \tau_1 \oplus \tau_2)$ *En supposant* $x \in \tau_1$ *et appliquant l'hypothèse inductive sur* τ_1 *:* $\overrightarrow{(\varphi_2 \circ \varphi_1)_\tau}(x) = \overrightarrow{(\varphi_2 \circ \varphi_1)_{\tau_1}}(x) = (\vec{\varphi}_2 \circ \vec{\varphi}_1)(x)$

□

LEMMA 1.5.2. *Pour tout couple d'interprétations* $\psi_1, \varphi_1 : A \overset{\mathcal{H}}{\longmapsto} B$ *et* $\psi_2, \varphi_2 : B \overset{\mathcal{H}}{\longmapsto} C$ *entre algèbres, dont la première a pour signature* $\Sigma_A = (\mathcal{B}, S, F, T)$*, pour tout type du premier ordre* $\tau \in \mathcal{T}_{\mathcal{F}}(S)$*, et pour tout objet* $x \in \tau^A$ *et* $z \in \tau^C_{\psi_2 \circ \psi_1}$*, nous avons l'implication suivante :*

$$\exists y \in \tau^B_{\psi_1}.\ \ x \underset{\tau}{\overset{\varphi_1}{\longmapsto}} y \wedge y \underset{\tau_{\psi_1}}{\overset{\varphi_2}{\longmapsto}} z \qquad \Rightarrow \qquad x \underset{\tau}{\overset{\varphi_2 \circ \varphi_1}{\longmapsto}} z$$

DÉMONSTRATION. *Par induction sur le type du premier ordre* τ *dont la syntaxe générative est, nous le rappelons,* $\mathcal{T}_{\mathcal{F}}(S) \ni \tau ::= \tau_\varepsilon \,|\, \tau_\varepsilon \to \tau \,|\, \tau_\varepsilon \multimap \tau$ *où* $\tau_\varepsilon \in \mathcal{T}_{\mathcal{E}}(S)$ *est un type élémentaire.*

– $(\tau = \tau_\varepsilon)$
Par le lemme 1.3.3, l'hypothèse se traduit en $\vec{\varphi}_1(x) = y$ *et* $\vec{\varphi}_2(y) = z$*. Donc* $z = \vec{\varphi}_2(y) = \vec{\varphi}_2(\vec{\varphi}_1(x)) = (\vec{\varphi}_2 \circ \vec{\varphi}_1)(x)$*. Par le lemme 1.5.1* $z = \overrightarrow{\varphi_2 \circ \varphi_1}(x)$ *et enfin, toujours par le lemme 1.3.3, nous avons l'énoncé* $x \underset{\tau}{\overset{\varphi_2 \circ \varphi_1}{\longmapsto}} z$*.*

– $(\tau = \tau_\varepsilon \multimap \tau')$
L'hypothèse est :

$$\exists g : (\tau_\varepsilon \multimap \tau')_{\psi_1}.\ \ f \underset{\tau_\varepsilon \multimap \tau'}{\overset{\varphi_1}{\longmapsto}} g \wedge f \underset{(\tau_\varepsilon \multimap \tau')_{\psi_1}}{\overset{\varphi_2}{\longmapsto}} h$$

Supposons que $x \in \mathrm{dom}(f)$ *et que* $x \underset{\tau_\varepsilon}{\overset{\varphi_2 \circ \varphi_1}{\longmapsto}} z$*. Puisque* τ_ε *est un type élémentaire, nous avons* $z = \overrightarrow{\varphi_2 \circ \varphi_1}(x) = (\vec{\varphi}_2 \circ \vec{\varphi}_1)(x)$ *(cf. lemme 1.5.1). L'élément* $y \triangleq \vec{\varphi}_1(x)$ *est, par définition, tel que* $x \underset{\tau_\varepsilon}{\overset{\varphi_1}{\longmapsto}} y$*. De plus, il est aussi tel que* $z = \vec{\varphi}_2(y)$*, donc tel que* $y \underset{\tau_\varepsilon}{\overset{\varphi_2}{\longmapsto}} z$*. Par hypothèse nous avons* $y \in \mathrm{dom}(g)$*, donc* $z \in \mathrm{dom}(h)$*,* $f(x) \underset{\tau'}{\overset{\varphi_1}{\longmapsto}} g(y)$ *et* $g(y) \underset{\tau'_{\psi_1}}{\overset{\varphi_2}{\longmapsto}} h(z)$*. Ainsi, pour les éléments* $f(x)$ *et* $h(z)$ *il existe un élément* $g(y)$ *qui vérifie l'hypothèse du lemme. En appliquant l'hypothèse d'induction, nous pouvons en déduire que* $f(x) \underset{\tau'}{\overset{\varphi_2 \circ \varphi_1}{\longmapsto}} h(z)$*. Pour résumer, en supposant* $x \in \mathrm{dom}(f)$ *et* $x \underset{\tau_\varepsilon}{\overset{\varphi_2 \circ \varphi_1}{\longmapsto}} z$*, nous avons pu déduire* $z \in \mathrm{dom}(h)$ *et* $f(x) \underset{\tau'}{\overset{\varphi_2 \circ \varphi_1}{\longmapsto}} h(z)$*. Autrement dit* $f \underset{\tau_\varepsilon \multimap \tau'}{\overset{\varphi_2 \circ \varphi_1}{\longmapsto}} h$*.*

– $(\tau = \tau_\varepsilon \to \tau')$
Il s'agit d'un cas particulier du précédent (toute fonction totale étant aussi une fonction partielle). L'hypothèse :

$$\exists g : (\tau_\varepsilon \to \tau')_{\psi_1}.\ \ f \underset{\tau_\varepsilon \to \tau'}{\overset{\varphi_1}{\longmapsto}} g \wedge f \underset{(\tau_\varepsilon \to \tau')_{\psi_1}}{\overset{\varphi_2}{\longmapsto}} h$$

permet donc d'obtenir $f \underset{\tau_\varepsilon \multimap \tau'}{\overset{\varphi_2 \circ \varphi_1}{\longmapsto}} h$*, qui équivaut, les fonctions* f *et* h *étant totales, à* $f \underset{\tau_\varepsilon \to \tau'}{\overset{\varphi_2 \circ \varphi_1}{\longmapsto}} h$*.*

□

Nous dirons qu'une algèbre est *du premier ordre* lorsque l'ensemble de ses opérations sera du premier ordre. Sinon, elle sera *d'ordre supérieur* comme l'une de ses opérations. Une conséquence directe du lemme précédent est que la composition d'homomorphismes entre algèbres du premier ordre est encore un homomorphisme.

COROLLARY 1.5.3. *La sous-classe des algèbres typées constituée seulement par des opérations du premier ordre est une catégorie où les morphismes sont les homomorphismes et l'identité est le couple des interprétations identiques.*

DÉMONSTRATION. *Soient* $\psi_1, \varphi_1 : A \stackrel{\mathcal{H}}{\longmapsto} B$ *et* $\psi_2, \varphi_2 : B \stackrel{\mathcal{H}}{\longmapsto} C$ *deux homomorphismes entre algèbres du premier ordre, respectivement de signature* $\Sigma_A = (\mathcal{B}, S, F, T)$, $\Sigma_B = (\mathcal{B}, S', F', T')$ *et* $\Sigma_C = (\mathcal{B}, S'', F'', T'')$. *Soit* $f \in F$. *D'une part, puisque* $\psi_1, \varphi_1 : A \stackrel{\mathcal{H}}{\longmapsto} B$, *il existe* $g = f_{\psi_1} \in F'$ *tel que* $f \overset{\varphi_1}{\underset{\tau}{\longmapsto}} g$. *D'autre part, puisque* $\psi_2, \varphi_2 : B \stackrel{\mathcal{H}}{\longmapsto} C$, *il existe* $h = (f_{\psi_1})_{\psi_2} = f_{\psi_2 \circ \psi_1} \in F''$ *tel que* $g \overset{\varphi_2}{\underset{\tau_{\psi_1}}{\longmapsto}} h$. *Par le lemme 1.5.2, le type* τ *étant du premier ordre, nous avons* $f \overset{\varphi_2 \circ \varphi_1}{\underset{\tau}{\longmapsto}} h$. *Nous avons donc prouvé que tout* $f \in F$ *vérifie la condition* $f \overset{\varphi_2 \circ \varphi_1}{\underset{\tau}{\longmapsto}} f_{\psi_2 \circ \psi_1}$. *Autrement dit,* $(\psi_2 \circ \psi_1, \varphi_2 \circ \varphi_1) : A \stackrel{\mathcal{H}}{\longmapsto} C$. *L'associativité de l'opération de composition, que nous noterons* $\cdot$, *dérive directement de l'associativité de la composition fonctionnelle :*

$$(\psi_3, \varphi_3) \cdot (\psi_2 \circ \psi_1, \varphi_2 \circ \varphi_1) = (\psi_3 \circ (\psi_2 \circ \psi_1), \varphi_3 \circ (\varphi_2 \circ \varphi_1)) =$$
$$((\psi_3 \circ \psi_2) \circ \psi_1, (\varphi_3 \circ \varphi_2) \circ \varphi_1) = (\psi_3 \circ \psi_2, \varphi_3 \circ \varphi_2) \cdot (\psi_1, \varphi_1)$$

Enfin, l'interprétation $(\psi_{id}, \varphi_{id})$ *constituée par les fonctions identiques* $\psi_{id} : S \to S \oplus F \to F$, $\psi_{id}(x) = x$, *et* $\varphi_{id} : \oplus_{D \in S} D \to D$, $\varphi(x) = x$, *est l'élément neutre de l'opération de composition des morphismes :*

$$(\psi_{id}, \varphi_{id}) \cdot (\psi, \varphi) = (\psi_{id} \circ \psi, \varphi_{id} \circ \varphi) = (\psi, \varphi) =$$
$$(\psi \circ \psi_{id}, \varphi \circ \varphi_{id}) = (\psi, \varphi) \cdot (\psi_{id}, \varphi_{id})$$

□

1.5.1. Le contre-exemple pour l'ordre supérieur. Le résultat précédent ne peut être généralisé aux algèbres d'ordre supérieur. Le contre-exemple est le suivant : supposons qu'une algèbre A soit constituée d'un seul ensemble, représenté par la sorte D, constitué à son tour de quatre éléments $D^A = \{a, b, c, d\}$. Soit B une autre structure où $D^B = \{u, v, w\}$, et C une troisième telle que $D^C = \{0, 1\}$. On peut vérifier facilement que pour qu'une fonction $f : D \to D$ admette une correspondante $f' : D_\psi \to D_\psi$, pour une interprétation des valeurs φ donnée, il est nécessaire qu'elle vérifie, pour tout x_1 et x_2, l'implication :

$$\varphi(x_1) = \varphi(x_1) \quad \Rightarrow \quad \varphi(f(x_1)) = \varphi(f(x_1))$$

Considérons alors la fonction $\hat{f} : D \to D$ définie de la façon suivante :

$$\hat{f}(x) = \begin{cases} b & \text{si} \quad x = a \\ c & \text{si} \quad x = b \\ a & \text{si} \quad x = c \\ d & \text{si} \quad x = d \end{cases}$$

et supposons qu'on ait les deux interprétations suivantes des valeurs (la première de A vers B, la seconde de B vers C) :

$$\varphi_1(x) = \begin{cases} u & \text{si} \quad x = a \\ u & \text{si} \quad x = b \\ v & \text{si} \quad x = c \\ w & \text{si} \quad x = d \end{cases} \qquad \varphi_2(y) = \begin{cases} 0 & \text{si} \quad y = u \\ 0 & \text{si} \quad y = v \\ 1 & \text{si} \quad y = w \end{cases}$$

Il existe alors un couple de valeurs (a, b) tel que $\varphi_1(a) = \varphi_1(b)$ mais $\varphi_1(\hat{f}(a)) = \varphi_1(b) = u \neq v = \varphi_1(c) = \varphi_1(\hat{f}(b))$. Pour cela $\hat{f}$ ne peut pas avoir un objet correspondant dans l'algèbre B. Cependant, elle a la fonction identité comme correspondante dans l'algèbre C. Si l'on considère la composition des interprétations $\varphi_2 \circ \varphi_1$:

$$(\varphi_2 \circ \varphi_1)(x) = \begin{cases} 0 & \text{si} \quad x = a \\ 0 & \text{si} \quad x = b \\ 0 & \text{si} \quad x = c \\ 1 & \text{si} \quad x = d \end{cases}$$

nous avons $x \overset{\varphi_2 \circ \varphi_1}{\underset{D}{\longmapsto}} 0$ si et seulement si $x \in \{a, b, c\}$, dans ce cas $\hat{f}(x) \in \{a, b, c\}$ donc $\hat{f}(x) \overset{\varphi_2 \circ \varphi_1}{\underset{D}{\longmapsto}} 0$, et $x \overset{\varphi_2 \circ \varphi_1}{\underset{D}{\longmapsto}} 1$ si et seulement si $x = d$, dans ce cas $\hat{f}(x) = d$ donc $\hat{f}(x) \overset{\varphi_2 \circ \varphi_1}{\underset{D}{\longmapsto}} 1$. Ainsi, dans les deux cas, si $x \overset{\varphi_2 \circ \varphi_1}{\underset{D}{\longmapsto}} w$, alors $\hat{f}(x)$ correspond au résultat de l'identité sur l'élément $w \in \{0, 1\}$. Autrement dit $\hat{f} \overset{\varphi_2 \circ \varphi_1}{\underset{D}{\longmapsto}} \mathrm{id}$. Nous pouvons alors définir une fonction d'ordre supérieur $F^A : (D \to D) \to D$, une seconde $F^B : (D \to D) \to D$ et une troisième $F^C : (D \to D) \to D$, de la façon suivante :

$$F^A(f) = \begin{cases} a & \text{si } f = \hat{f} \\ d & \text{sinon} \end{cases} \qquad F^B(f) = w \qquad F^C(f) = 1$$

Nous observons, d'une part, que $F^A \overset{\varphi_1}{\underset{(D \to D) \to D}{\longmapsto}} F^B$. En effet, puisqu'il n'existe pas de fonction correspondante à $\hat{f}$, si l'on suppose $f \overset{\varphi_1}{\underset{D \to D}{\longmapsto}} f'$, on aura forcément $f \neq \hat{f}$, donc $F^A(f) = d$ qui correspondra via φ_1 à $w = F^B(f')$. D'autre part, il est également vrai que $F^B \overset{\varphi_2}{\underset{(D \to D) \to D}{\longmapsto}} F^C$, puisque en supposant $g \overset{\varphi_2}{\underset{D \to D}{\longmapsto}} g'$, nous avons $F^B(g) = w$ correspond via φ_2 à $1 = F^C(g')$. Cependant, il est faux que $F^A \overset{\varphi_2 \circ \varphi_1}{\underset{(D \to D) \to D}{\longmapsto}} F^C$. Il suffit de considérer la fonction $\hat{f}$ qui correspond, comme nous avons vu, à l'identité id, pour laquelle $F^A(\hat{f}) = a$ alors que $F^C(\mathrm{id}) = 1$. En résumé, l'énoncé du lemme 1.5.2 est fausse lorsque nous ne nous limitons pas aux types du premier ordre.

CHAPITRE 2

Éléments de théorie des domaines

Les préordres, les ordres partiels, et toutes les notions corrélées de ce chapitre se retrouvent dans la littérature de la théorie des domaines. Cependant, certains sujets sont développés de façon autonome (notamment la co-finalité et co-initialité), certains résultats, utilisés par la suite, sont prouvés formellement, et l'ensemble des notions et des énoncés est abordées par le métalangage algébrique défini dans le chapitre précédent.

2.1. Préordres

Les préordres sont des algèbres constituées d'un seul domaine, donc des *structures*. Avec le domaine D, un semi-prédicat binaire $\leq : 1^{\mathsf{D}\times\mathsf{D}}$, appelé *relation de préordre*, caractérise complètement la structure. Nous verrons que l'algèbre est aussi peuplée par d'autre opérations telles que les propriétés *dirigé*, *idéal* ou *filtre*. Nous utiliserons le plus souvent la notation infixe $x \leq y$ pour signifier $\leq (x, y) = \bullet$, autrement dit pour signifier $(x, y) \in \mathrm{dom}(\leq)$. Nous noterons souvent, avec un abus de notation, la relation binaire $\mathrm{dom}(\leq)$, qui est de type $\wp^{\mathsf{D}\times\mathsf{D}}$, toujours par le symbole $\leq$, ce qui nous autorisera l'écriture $(x, y) \in\leq$. La meilleure approche, selon nous, est celle de considérer l'opération $\leq$ comme une *propriété*, ayant le type *officiel*, pour ainsi dire, de semi-prédicat $\leq : 1^{\mathsf{D}\times\mathsf{D}}$, mais pouvant être considérée en l'occurrence comme une relation $\leq : \wp^{\mathsf{D}\times\mathsf{D}}$ (d'où le nom *relation de préordre* et la notation $(x, y) \in\leq$), ou encore, comme un prédicat $\leq : 2^{\mathsf{D}\times\mathsf{D}}$ (d'où la notation $x \leq y$).

2.1.1. Préordres. Soit $\mathfrak{R}$ une relation sur un domaine D. Si les conditions suivantes sont vérifiées :

(a) $\forall x \in D.\ (x,x) \in \mathfrak{R}$

(b) $\forall x,y,z \in D.\ (x,y) \in \mathfrak{R} \wedge (y,z) \in \mathfrak{R} \Rightarrow (x,z) \in \mathfrak{R}$

nous disons que le couple $(D,\mathfrak{R})$ est un *préordre*. Nous traduisons la condition (a) en disant que $\mathfrak{R}$ est *réflexive* sur D, et la (b) en disant que $\mathfrak{R}$ est *transitive* sur D. Le *dual* du préordre $(D,\mathfrak{R})$ est le couple $(D,\mathfrak{R}^{\perp})$ où $\mathfrak{R}^{\perp} \triangleq \{(y,x) \in D \times D \mid (x,y) \in \mathfrak{R}\}$ est la relation *duale* de $\mathfrak{R}$. Si la relation est notée par un symbole asymétrique (par exemple $\leq$), l'image spéculaire du symbole (dans l'exemple $\geq$) pourra dénoter, aussi bien que $\leq^{\perp}$, la relation duale. La *relation stricte* associée à $\mathfrak{R}$ est la relation $\mathfrak{R} \setminus (D \times D)$. Toute relation de préordre contient, par définition, l'égalité sur le domaine. Quand le symbole utilisé fera référence implicite à l'égalité contenue (comme $\leq$, $\sqsubseteq$ ou $\preceq$), le symbole empêchant cette réminiscence (dans l'exemple $<$, $\sqsubset$ ou $\prec$) dénotera la relation stricte associée.

2.1.2. Symétrie et anti-symétrie. Deux propriétés additionnelles, la *symétrie* et l'*anti-symétrie* (qui ne sont pas, malgré leur nom, en opposition logique) peuvent caractériser une structure de préordre.

2.1.2.1. *Équivalences.* Lorsqu'un préordre $(D,\mathfrak{R})$ vérifie aussi la condition suivante :

(c') $\forall x,y \in D.\ (x,y) \in \mathfrak{R} \Rightarrow (y,x) \in \mathfrak{R}$

nous disons que $\mathfrak{R}$ est une relation d'*équivalence* sur D. En particulier, nous traduisons la condition (c') en disant que $\mathfrak{R}$ est *symétrique* sur D.

Étant donné une équivalence $(D,\approx)$, la *classe* (d'*équivalence*) de x est l'ensemble $[x]_{\approx} \triangleq \{y \in D \mid x \approx y\}$. Tout élément d'une classe est un *représentant* de la classe (par le biais de l'application $[.]_{\approx} : D \to \wp^{D}$). Les propriétés des équivalences (réflexive, transitive et symétrique) impliquent que la relation $x \approx y$ soit logiquement équivalente à l'égalité des classes $[x]_{\approx} = [y]_{\approx}$. Le *quotient* de D *modulo* $\approx$ est l'ensemble des classes d'équivalences : $D_{\approx} \triangleq \{\chi \in \wp^{D} \mid \chi = [x]_{\approx},\ x \in D\}$. Le quotient modulo une relation d'équivalence est une *partition* de l'ensemble D, i.e. une famille de parties $\{\chi_i\}_{i\in I}$ de D étanches ($\chi_i \cap \chi_j = \emptyset$) et dont l'union reconstitue D (i.e. $\bigcup_{i\in I} \chi_i = D$).

2.1.2.2. *Ordres partiels.* Un préordre $(D,\mathfrak{R})$ est un *ordre partiel* s'il vérifie la condition suivante, appelée propriété *anti-symétrique* :

(c) $\forall x,y \in D.\ (x,y) \in \mathfrak{R} \wedge (y,x) \in \mathfrak{R} \Rightarrow x = y$

2.1.2.3. *Préordres propres.* Nous dirons qu'un préordre est *propre* lorsqu'il ne vérifie ni la propriété symétrique ni l'anti-symétrique, en d'autres termes, lorsque il ne s'agit ni d'une équivalence ni d'un ordre partiel.

Tout préordre $(D,\leq)$ définit implicitement une relation d'équivalence par l'intersection de la relation de préordre et de sa duale :

$$\approx \triangleq \leq \cap \geq = \{(x,y) \in D \times D \mid x \leq y \wedge y \leq x\}$$

Nous dirons que l'équivalence $\approx$ est *engendrée* par le préordre. La relation qui résulte de cette construction est une relation originale d'un véritable intérêt mathématique, seulement lorsque le préordre est propre. Sinon, dans le cas d'une équivalence, l'équivalence engendrée coïncide avec son germe : $(\approx) = (\leq)$, tandis que dans le cas d'un ordre partiel, l'équivalence engendrée coïncide avec l'égalité sur le domaine : $(\approx) = (=)$. Dans les deux cas, elle existe déjà.

La relation de préordre $\leq$ peut être étendue aux classes de l'ensemble quotient $D_{\approx}$:

$$\chi_1 \leq \chi_2 \overset{\triangle}{\iff} \chi_1 = [x] \ \wedge \ \chi_2 = [y] \ \wedge \ x \leq y$$

On montre que la définition ne dépend pas des représentants x et y choisis, et que la nouvelle relation est, elle aussi, un préordre. De plus, elle vérifie la propriété anti-symétrique : $\chi_1 \leq \chi_2 \ \wedge \ \chi_1 \geq \chi_2 \ \Rightarrow \ \chi_1 = \chi_2$, ce qui confère au couple $(D_{\approx}, \leq)$ la structure d'ordre partiel. Nous dirons, de ce dernier aussi, qu'il s'agit de l'ordre partiel *engendré*.

Ainsi, un préordre propre $(D, \leq)$ définit implicitement, d'une part, une structure *symétrique*, l'équivalence $(D, \approx)$ et, d'autre part, une structure *anti-symétrique*, l'ordre partiel $(D_{\approx}, \leq)$.

2.1.3. Parties singulières d'un préordre. Étant donné un préordre $\Gamma = (D, \mathfrak{R})$, les parties $X \in \wp^D$ du domaine peuvent être classées selon des critères portant sur la relation $\mathfrak{R}$.

2.1.3.1. *Dirigés, co-dirigés.* Nous disons qu'un sous-ensemble $X \in \wp^D$, est un *dirigé* (ou *filtrant*) de Γ si : $\forall x, y \in X. \ \exists z \in X \ : \ (x, z) \in \mathfrak{R} \wedge (y, z) \in \mathfrak{R}$. Nous noterons $\Delta(\Gamma)$ l'ensemble des dirigés du préordre. Si $D \in \Delta(\Gamma)$ nous disons que le préordre Γ est, lui-même, un dirigé. De façon duale, nous disons que $X \in \wp^D$ est un *co-dirigé* de Γ si : $\forall x, y \in X. \ \exists z \in X \ : \ (z, x) \in \mathfrak{R} \wedge (z, y) \in \mathfrak{R}$. Nous notons $\nabla(\Gamma)$ l'ensemble des co-dirigés du préordre et si $D \in \nabla(\Gamma)$ nous disons que le préordre Γ est, lui-même, un co-dirigé. Nous remarquerons que l'ensemble vide $\emptyset$ est, par définition, à la fois un dirigé et un co-dirigé de tout préordre Γ. Nous noterons respectivement Δ^+ et ∇^+ les propriétés *dirigé non vide* $\Delta(\Gamma) \setminus \{\emptyset\}$ et *co-dirigé non vide* $\nabla(\Gamma) \setminus \{\emptyset\}$.[1]

2.1.3.2. *Majorants, minorants.* Un élément $x \in D$ est un *majorant* d'un sous-ensemble $X \in \wp^D$, si $\forall x' \in X. \, x'\mathfrak{R}x$. La propriété *être un majorant de* (*upper bound of*) est définie par : $\mathrm{ub}_\Gamma(x, X) \overset{\triangle}{\iff} \forall x' \in X. \, x'\mathfrak{R}x$. L'ensemble de tous les majorants (upper bounds) de $X \in \wp^D$ est : $\mathrm{ubs}_\Gamma(X) \triangleq \{y \in D \mid \mathrm{ub}_\Gamma(y, X)\}$. De façon duale, un élément $x \in D$ est un *minorant* de $X \in \wp^D$, si $\forall x' \in X. \, x\mathfrak{R}x'$. La propriété *être un minorant de* (*lower bound*) est définie par $\mathrm{lb}_\Gamma(x, X) \overset{\triangle}{\iff} \forall x' \in X. \, x\mathfrak{R}x'$. L'ensemble $\mathrm{lbs}_\Gamma(X) \triangleq \{y \in D \mid \mathrm{lb}_\Gamma(y, X)\}$ est l'ensemble de tous les minorants (lower bounds) de X. Nous considérerons les deux propriétés ub_Γ et lb_Γ, comme des semi-prédicats, donc de type $1^{D \times \wp^D}$. Les deux opérations ubs_Γ et lbs_Γ seront, en revanche, des fonctions de type $\wp^D \to \wp^D$. On remarquera que, par définition, $\forall x \in D. \ \mathrm{ub}_\Gamma(x, \emptyset) \wedge \mathrm{lb}_\Gamma(x, \emptyset)$, donc $\mathrm{ubs}_\Gamma(\emptyset) = \mathrm{lbs}_\Gamma(\emptyset) = D$.

[1] Certains auteurs excluent directement l'ensemble vide $\emptyset$ des définitions de dirigé et co-dirigé. Ce choix n'est pas sans conséquences pour les notions, définies successivement, d'ordre partiel *complet par dirigés* ou *co-dirigés* et sur celle de fonction *continue*.

2.1.3.3. *Maximaux, minimaux.* Un élément *maximal* d'un sous-ensemble donné $X \in \wp^D$ non vide $X \neq \emptyset$, est un élément $x \in X$ tel que $\forall x' \in X.\ x'\mathfrak{R}x$. D'autre part, un élément *minimal* d'un sous-ensemble $X \in 2^D$ non vide $X \neq \emptyset$, est un élément $x \in X$ tel que $\forall x' \in X.\ x\mathfrak{R}x'$. Nous noterons respectivement $\max_\Gamma X$ et $\min_\Gamma X$ l'ensemble des éléments maximaux et celui des éléments minimaux du sous-ensemble X dans le préordre $\Gamma = (D, \mathfrak{R})$. Il s'agit de deux fonctions partielles $\max_\Gamma$, $\min_\Gamma : \wp^D \multimap \wp^D$ (qui ne sont pas définies sur l'ensemble vide : $\emptyset \notin \mathrm{dom}(\min)$ et $\emptyset \notin \mathrm{dom}(\max)$).

2.1.3.4. *Clôtures vers le bas, vers le haut.* Nous disons qu'un sous-ensemble $\mathcal{L} \in 2^D$ est *clos vers le bas*[2] (resp. $\mathcal{U} \in 2^D$ est *clos vers le haut*[3]) dans Γ si : $\forall x \in \mathcal{L}. \forall y \in D.\ (y,x) \in \mathfrak{R} \Rightarrow y \in \mathcal{L}$ (resp. $\forall x \in \mathcal{U}. \forall y \in D.\ (x,y) \in \mathfrak{R} \Rightarrow y \in \mathcal{U}$). Nous noterons $\mathcal{L}(\Gamma)$ et $\mathcal{U}(\Gamma)$ l'ensemble des parties du préordre closes, respectivement, vers le bas et vers le haut. La *fermeture vers le haut* est l'opération $\uparrow_\Gamma (.) : \wp^D \rightarrow \wp^D$ définie par $\uparrow_\Gamma (X) = \{y \in D \mid \exists x \in X.\ x\mathfrak{R}y\}$. De façon duale, la *fermeture vers le bas* est l'opération $\downarrow_\Gamma (.) : \wp^D \rightarrow \wp^D$ définie par $\downarrow_\Gamma (X) = \{y \in D \mid \exists x \in X.\ y\mathfrak{R}x\}$. Par définition $\mathcal{L}(\Gamma) = \{X \in \wp^D \mid\ \downarrow_\Gamma (X) = X\}$, $\mathcal{U}(\Gamma) = \{X \in \wp^D \mid\ \uparrow_\Gamma (X) = X\}$, $\mathrm{ubs}_\Gamma(X) \subseteq \uparrow_\Gamma (X)$ et $\mathrm{lbs}_\Gamma(X) \subseteq \downarrow_\Gamma (X)$. Nous utiliserons les notations allégées $\uparrow_\Gamma x$ et $\downarrow_\Gamma x$, ou $\uparrow x$ et $\downarrow x$ (lorsque la structure Γ sera implicite dans le contexte), pour signifier respectivement $\uparrow_\Gamma \{x\}$ et $\downarrow_\Gamma \{x\}$. Tout sous-ensemble de la forme $\downarrow_\Gamma x$ sera appelé *cône* (*de* x).

2.1.3.5. *Idéaux et filtres.* Les idéaux et les filtres sont des propriétés dérivées (par intersection). Nous disons qu'un sous-ensemble $\mathcal{I} \in \wp^D$ est un *idéal* si il est à la fois dirigé et clos vers le bas. Nous noterons $\mathcal{I}(\Gamma)$ l'ensemble des idéaux du préordre : $\mathcal{I}(\Gamma) \triangleq \Delta(\Gamma) \cap \mathcal{L}(\Gamma)$. En particulier, les ensembles $\downarrow x$ seront les *idéaux principaux*. La notion duale est celle de *filtre* : un sous-ensemble $\mathcal{F} \in \wp^D$ est un *filtre* si il est à la fois co-dirigé et clos vers le haut. L'ensemble des filtres du préordre est ainsi défini par $\mathcal{F}(\Gamma) \triangleq \nabla(\Gamma) \cap \mathcal{U}(\Gamma)$. En particulier, les *filtres principaux* seront les ensembles de la forme $\uparrow x$.

2.1.4. Extension aux fonctions à valeurs dans le préordre.

Étant donné un préordre $(D, \leq : 1^{D \times D})$ et un domaine de base $X \in \mathcal{B}$, le préordre *extensionnel* entre fonctions de type $X \rightarrow D$ est le semi-prédicat $\leq^e_X$ défini par :

$$f \leq^e_X g \ \stackrel{\triangle}{\Longleftrightarrow}\ \forall x \in X.\ \ f(x) \leq g(x)$$

On vérifie facilement que la nouvelle propriété $\leq^e_X$ est, comme son germe $\leq$, symétrique et transitive. Donc, la structure $(X \rightarrow D, \leq^e_X)$ est, elle aussi, un préordre. Le choix d'un domaine de base reflète l'intention de neutraliser la traduction des valeurs pour toute interprétation du préordre $A = (D, \leq : 1^{D \times D})$ dans une autre algèbre B. Les valeurs du domaine de base X seront forcément traduites par la fonction identité dans le même ensemble X.

Nous remarquerons que le type du préordre extensionnel dépend de l'ensemble de base donné. Il s'agit donc d'une construction polymorphe :

$$\leq^e\ :\ \forall \alpha{:}\mathsf{base}.\ 1^{(\alpha \rightarrow D) \times (\alpha \rightarrow D)}$$

[2] *lower set* ou *down set*
[3] *upper set*

Non seulement $\leq^e$ est une construction polymorphe, mais les objets mathématiques résultants sont d'ordre supérieur. En effet, étant donné $X \in \mathcal{B}$, le type de $\leq^e_X$, c'est-à-dire $1^{(X \to D)\times(X \to D)} = ((X \to D)\times(X \to D)) \twoheadrightarrow 1$, n'est pas un type du premier ordre.

2.1.5. Extension aux classes co-finales et co-initiales. Étant donné un préordre $(D, \leq : 1^{D\times D})$, la *co-finalité* entre parties de D est le semi-prédicat $\leq^\flat$: $1^{(\wp^D)\times(\wp^D)}$ défini par :

$$X \leq^\flat Y \overset{\triangle}{\iff} \forall x \in X.\ \exists y \in Y.\ x \leq y$$

Si $X \leq^\flat Y$ on dit que Y est *co-final* de X. Vice versa, la *co-initialité* entre parties de D est la relation $\leq^\sharp$: $1^{(\wp^D)\times(\wp^D)}$ définie par :

$$X \leq^\sharp Y \overset{\triangle}{\iff} \forall y \in Y.\ \exists x \in X.\ x \leq y$$

Si $X \leq^\sharp Y$ on dit que X est *co-initial* de Y. Les deux semi-prédicats $\leq^\flat$ et $\leq^\sharp$ apparaissent en littérature (par exemple dans [**Taylor, 1999**]), le premier sous le nom de *lower order*. Ce nom s'explique en observant que la condition $X \leq^\flat Y$ est équivalente à l'inclusion des respectives clôtures vers le bas $\downarrow X \subseteq \downarrow Y$. De façon symétrique, la condition $X \leq^\sharp Y$ est équivalente à l'inclusion $\uparrow X \supseteq \uparrow Y$. Cependant, le nom de *lower order* est à notre avis captieux : il évoque le bas alors que, étant donné deux ensembles X et Y, leur partie initiale est ignorée : l'intérêt est tout pour leur avant-garde (pour tout $x \in X$ il existe un $y \in Y$ plus grand que x), d'où le nom de *co-finalité*. La notation musicale, nous apprend encore [**Taylor, 1999**], a été proposée par Carl Gunter. Les deux relations sont de toute évidence *réflexives* et *transitives* : l'ensemble des parties $\wp^D$, équipé d'une de ces deux relations, constitue toujours un préordre. Et elles sont apparentées : la co-finalité construite sur le préordre dual coïncide avec la co-initialité construite sur le préordre de départ : $(\leq^\perp)^\flat = \leq^\sharp$, et vice versa : $(\leq^\perp)^\sharp = \leq^\flat$. Puisque ces nouvelles relations ne sont pas la duale l'une de l'autre : $\leq^\flat \neq (\leq^\sharp)^\perp$, nous avons $(\leq^\perp)^\flat = \leq^\sharp \neq (\leq^\flat)^\perp$, et $(\leq^\perp)^\sharp = \leq^\flat \neq (\leq^\sharp)^\perp$. Ainsi, la notation des symboles renversés, $\geq^\flat$ et $\geq^\sharp$, pose un problème d'ambiguïté sur l'ordre de la construction mathématique opérée : dualité puis co-finalité (resp. co-initialité) ou co-finalité (resp. co-initialité) puis dualité. Ainsi, il est nécessaire de préciser que le symbole $\geq^\flat$ dénotera la relation duale de la co-finalité $(\leq^\flat)^\perp$ (et non pas la co-finalité de la duale $(\leq^\perp)^\flat = \leq^\sharp$), et que le symbole $\geq^\sharp$ dénotera la relation duale de la co-initialité $(\leq^\sharp)^\perp$ (et non pas la co-initialité de la duale $(\leq^\perp)^\sharp = \leq^\flat$). La *conjonction* logique des deux relations apparaît elle aussi en littérature (par exemple dans [**Abramsky & Jung, 1994**]) sous le nom d'*ordre d'Egli-Milner :*

$$X \leq^\natural Y \quad \overset{\triangle}{\iff} \quad X \leq^\flat Y\ \wedge X \leq^\sharp Y.$$

Ainsi, un couple de parties X et Y est dans l'ordre d'Egli-Milner si et seulement si Y est co-final de X et X est co-initial de Y.

2.1.5.1. *Équivalences engendrées.* Si le préordre de départ $\leq$ n'est pas trivialement contenu dans la diagonale de $D \times D$ (i.e. il existe $x, y \in D$, $x \neq y$ tel que $x \leq y$), alors les préordres de co-finalité et co-initialité sont propres. Dans cette hypothèse, les contre-exemples de la propriété anti-symétrique sont, pour le préordre $\leq^\flat$, les parties $\{x, y\}$ et $\{y\}$, et pour le préordre $\leq^\sharp$, les parties $\{x, y\}$ et $\{x\}$. Les équivalences engendrées par ces préordres, notées respectivement $\approx^\flat$ et $\approx^\sharp$, sont

TAB. 1. Signature complète d'un préordre

(D			,	*domaine*
$\leq$	:	$1^{\mathsf{D}\times\mathsf{D}}$,	,	*relation de préordre*
$\Delta(\mathsf{D},\leq)$	:	$1^{(\wp^{\mathsf{D}})}$	,	*dirigés (filtrants)*
$\nabla(\mathsf{D},\leq)$	:	$1^{(\wp^{\mathsf{D}})}$	,	*co-dirigés*
$\mathrm{lb}_{(\mathsf{D},\leq)}$	:	$1^{\mathsf{D}\times\wp^{\mathsf{D}}}$	,	*minorant (lower bound)*
$\mathrm{ub}_{(\mathsf{D},\leq)}$	:	$1^{\mathsf{D}\times\wp^{\mathsf{D}}}$	,	*majorant (upper bound)*
$\mathrm{lbs}_{(\mathsf{D},\leq)}$	:	$\wp^{\mathsf{D}}\to\wp^{\mathsf{D}}$	,	*minorants*
$\mathrm{ubs}_{(\mathsf{D},\leq)}$	:	$\wp^{\mathsf{D}}\to\wp^{\mathsf{D}}$	,	*majorants*
$\min_{(\mathsf{D},\leq)}$	:	$\wp^{\mathsf{D}}\multimap\wp^{\mathsf{D}}$	,	*minimaux*
$\max_{(\mathsf{D},\leq)}$	:	$\wp^{\mathsf{D}}\multimap\wp^{\mathsf{D}}$	,	*maximaux*
$\downarrow_{(\mathsf{D},\leq)}$	:	$\wp^{\mathsf{D}}\to\wp^{\mathsf{D}}$	,	*fermeture vers le bas*
$\uparrow_{(\mathsf{D},\leq)}$	:	$\wp^{\mathsf{D}}\to\wp^{\mathsf{D}}$	,	*fermeture vers le haut*
$\mathcal{L}(\mathsf{D},\leq)$	:	$1^{(\wp^{\mathsf{D}})}$	,	*lower sets*
$\mathcal{U}(\mathsf{D},\leq)$	:	$1^{(\wp^{\mathsf{D}})}$	,	*upper sets*
$\mathcal{I}(\mathsf{D},\leq)$	:	$1^{(\wp^{\mathsf{D}})}$	,	*idéaux*
$\mathcal{F}(\mathsf{D},\leq)$	:	$1^{(\wp^{\mathsf{D}})}$	,	*filtres*
$\leq^{e}$	:	$\forall\alpha{:}\mathsf{base}.\ 1^{(\alpha\to\mathsf{D})\times(\alpha\to\mathsf{D})}$	,	*préordre extensionnel*
$\leq^{\flat},\leq^{\sharp}$	:	$1^{(\wp^{\mathsf{D}})\times(\wp^{\mathsf{D}})}$	,	*préordres de co-finalité et co-initialité*
$\approx^{\flat},\approx^{\sharp}$	:	$1^{(\wp^{\mathsf{D}})\times(\wp^{\mathsf{D}})}$	)	*équivalences de co-finalité et co-initialité*

donc des équivalences non triviales, strictement plus *abstraites* (moins *fines*) que l'égalité sur les parties de D.

$$\approx^{\flat}\colon 1^{(\wp^{\mathsf{D}})\times(\wp^{\mathsf{D}})} \qquad \mathsf{X}\approx^{\flat}\mathsf{Y} \overset{\triangle}{\iff} \mathsf{X}\leq^{\flat}\mathsf{Y}\ \wedge\ \mathsf{Y}\geq^{\flat}\mathsf{X}$$
$$\approx^{\sharp}\colon 1^{(\wp^{\mathsf{D}})\times(\wp^{\mathsf{D}})} \qquad \mathsf{X}\approx^{\sharp}\mathsf{Y} \overset{\triangle}{\iff} \mathsf{X}\leq^{\sharp}\mathsf{Y}\ \wedge\ \mathsf{Y}\geq^{\sharp}\mathsf{X}$$

Les deux équivalences ne coïncident pas puisque les préordres $\leq^{\flat}$ et $\leq^{\sharp}$ ne sont pas le dual l'un de l'autre : $\leq^{\flat}\neq\geq^{\sharp}$ et $\leq^{\sharp}\neq\geq^{\flat}$. Nous utiliserons les mêmes symboles $\leq^{\flat}$ et $\leq^{\sharp}$ pour dénoter les relations de co-finalité et co-initialité étendues aux classes de l'équivalence respective. La structure $(\wp^{\mathsf{D}}_{\approx^{\flat}},\leq^{\flat}\colon 1^{\wp^{\mathsf{D}}_{\approx^{\flat}}\times\wp^{\mathsf{D}}_{\approx^{\flat}}})$, qui est un ordre partiel (2.1.2.3), sera l'*extension aux classes co-finales* du préordre $(\mathsf{D},\leq\colon 1^{\mathsf{D}\times\mathsf{D}})$. Vice versa, l'ordre partiel $(\wp^{\mathsf{D}}_{\approx^{\sharp}},\leq^{\sharp}\colon 1^{\wp^{\mathsf{D}}_{\approx^{\sharp}}\times\wp^{\mathsf{D}}_{\approx^{\sharp}}})$ sera son *extension aux classes co-initiales.*

NOTATION 2.1.1. Pour alléger la notation des fonctions $[.]_{\approx^{\flat}}:\wp^{\mathsf{D}}\to\wp^{\mathsf{D}}_{\approx^{\flat}}$ et $[.]_{\approx^{\sharp}}:\wp^{\mathsf{D}}\to\wp^{\mathsf{D}}_{\approx^{\sharp}}$, nous les écrirons respectivement $[.]_{\flat}$ et $[.]_{\sharp}$.

2.1.6. Signatures d'un préordre. Une façon de résumer la définition d'une algèbre est celle d'en présenter la signature. Pour les préordres la signature que nous dirons *essentielle* est $(\mathsf{D},\leq\colon 1^{\mathsf{D}\times\mathsf{D}})$, notée le plus souvent $(\mathsf{D},\leq)$. En revanche, la signature *complète*, faisant référence à toutes les opérations définies précédemment, est représentée dans la table 1.

2.2. Ordres Partiels

La principale conséquence de la propriété anti-symétrique des ordres partiels est qu'elle garantit, lorsque ils existent, l'unicité des éléments minimaux et maximaux. En d'autre termes, dans le contexte d'un ordre partiel, le $\max_{\Gamma}$ et le $\min_{\Gamma}$ deviennent des fonctions partielles de type $\wp^{\mathsf{D}} \multimap \mathsf{D}$, au lieu que de type $\wp^{\mathsf{D}} \multimap \wp^{\mathsf{D}}$ pour les préordres génériques.

2.2.1. Bornes inférieure et supérieure. Soit $\Gamma = (\mathsf{D}, \mathfrak{R})$ un ordre partiel. Si $\max_\Gamma(\mathsf{X}) = x$ (resp. $\min_\Gamma(\mathsf{X}) = x$) nous disons que x est le *maximum* (resp. *minimum*) de X. Nous disons que l'ordre partiel Γ est *pointé* (*vers le bas*) si $\exists \bot = \min_\Gamma \mathsf{D}$. Réciproquement, nous disons qu'il est *pointé vers le haut* si $\exists \top = \max_\Gamma \mathsf{D}$. La *borne supérieure de* $\mathsf{X} \in \wp^{\mathsf{D}}$ est, si elle existe, le minimum de ses majorants : $\sup_\Gamma \mathsf{X} \stackrel{\leftarrow}{=} \min_\Gamma(\mathrm{ubs}_\Gamma \mathsf{X})$. Vice versa, la *borne inférieure de* $\mathsf{X} \in \wp^{\mathsf{D}}$ est, si elle existe, le maximum de ses minorants : $\inf_\Gamma \mathsf{X} \stackrel{\leftarrow}{=} \max(\mathrm{lbs}_\Gamma \mathsf{X})$. Les fonctions partielles sup et inf sont souvent notées respectivement lub, de l'anglais *least upper bound*, et glb, de *greatest lower bound*. Quand le domaine D sera implicitement défini par le contexte, nous écrirons que le symbole de relation $\leq$ en indice des propriétés lb et ub, des fonctions ubs, lbs, et des fonctions partielles max, min, sup et inf. Ou vice versa, nous omettrons la relation en précisant seulement le domaine D. Et lorsque les deux seront implicites nous éliminerons tout indice. En particulier, dans ce dernier cas, les fonctions partielles $\sup \mathsf{A}$ et $\inf \mathsf{A}$ pourrons être notées de façon alternative, respectivement, par $\bigvee \mathsf{A}$ et $\bigwedge \mathsf{A}$. La composition de la fonction sup avec la propriété *dirigé non vide* Δ^+ (cf. section 1.1.6), est le *sup-dirigé* (ou *sup-filtrant*) : $\sqcup \triangleq \sup_{\Delta^+} = \{(\mathsf{X}, \sup_\Gamma(\mathsf{X})) \mid \mathsf{X} \in \Delta(\Gamma), \mathsf{X} \neq \emptyset\}$. D'autre part, la composition de la fonction inf avec la propriété *co-dirigé non vide* ∇^+, est le *inf-co-dirigé* : $\sqcap \triangleq \inf_{\nabla^+}$.

La co-finalité réciproque est une condition logique plus forte que l'égalité des bornes supérieures. De façon duale la co-initialité réciproque est plus forte que l'égalité de bornes inférieures.

LEMMA 2.2.1. *Soit* $(\mathsf{D}, \leq)$ *un ordre partiel. Alors, l'équivalence de co-finalité (resp. co-initialité) implique l'égalité, si elles existent, des bornes supérieures (resp. inférieures). D'une part nous avons l'implication suivante :*

$$\textit{Hp}\begin{cases} (\mathsf{D}, \leq) \textit{ ordre partiel} \\ \mathsf{X}, \mathsf{Y} \in \wp^{\mathsf{D}} \\ \mathsf{X} \approx^\flat \mathsf{Y} \end{cases} \qquad \textit{Ts}\,\{\sup \mathsf{X} = \nu \;\Leftrightarrow\; \sup \mathsf{Y} = \nu$$

et, d'autre part, nous avons l'implication duale :

$$\textit{Hp}\begin{cases} (\mathsf{D}, \leq) \textit{ ordre partiel} \\ \mathsf{X}, \mathsf{Y} \in \wp^{\mathsf{D}} \\ \mathsf{X} \approx^\sharp \mathsf{Y} \end{cases} \qquad \textit{Ts}\,\{\inf \mathsf{X} = \nu \;\Leftrightarrow\; \inf \mathsf{Y} = \nu$$

DÉMONSTRATION. *Nous démontrons uniquement la première implication, la seconde étant duale. Soit* $\mathsf{X} \approx^\flat \mathsf{Y}$. *Il suffit de montrer que* X *et* Y *partagent le même ensemble de majorants. En particulier nous avons* $\mathsf{X} \leq^\flat \mathsf{Y}$. *Soit* $\mathsf{M} \in \mathrm{ubs}_{(\mathsf{D},\leq)}\mathsf{Y}$, *donc* $\forall y \in \mathsf{Y}.\ y \leq \mathsf{M}$. *Soit* $x \in \mathsf{X}$. *Puisque* Y *est co-final de* X, *il existe* $y \in \mathsf{Y}$ *tel que* $x \leq y$. *Donc* $x \leq \mathsf{M}$, *autrement dit* $\mathsf{M} \in \mathrm{ubs}_{(\mathsf{D},\leq)}\mathsf{X}$. *En utilisant la co-finalité dans l'autre sens* ($\mathsf{Y} \leq^\flat \mathsf{X}$), *on déduit que* $\mathrm{ubs}_{(\mathsf{D},\leq)}\mathsf{X} = \mathrm{ubs}_{(\mathsf{D},\leq)}\mathsf{Y}$, *donc forcément* $\sup_{(\mathsf{D},\leq)} \mathsf{X} = \sup_{(\mathsf{D},\leq)} \mathsf{Y}$ *si l'un ou l'autre existe.* $\square$

Ainsi, aux classes de co-finalité (resp. co-initialité) on associe la borne supérieure (resp. inférieure), lorsqu'elle existe, partagée par tous leurs représentants. En d'autres termes, les fonctions partielles $\sup(.)$ et $\inf(.)$ peuvent être étendues, respectivement, aux classes de co-finalité et co-initialité : $\sup : (\wp^{\mathsf{D}} \oplus \wp^{\mathsf{D}}_{\approx_\flat}) \multimap \mathsf{D}$, $\sup \chi = \sup \mathsf{X}$ si $\chi = [\mathsf{X}]_\flat$, et $\inf : (\wp^{\mathsf{D}} \oplus \wp^{\mathsf{D}}_{\approx_\sharp}) \multimap \mathsf{D}$, $\inf \chi = \inf \mathsf{X}$ si $\chi = [\mathsf{X}]_\sharp$. Dans les deux cas la définition est indépendante du représentant X choisi pour la classe χ.

2.2.2. Complétude. Par définition, l'existence de la borne supérieure (ou inférieure) d'un sous-ensemble $\mathsf{D}' \subseteq \mathsf{D}$ est un souci double : d'une part il faut qu'il existe des majorants (resp. des minorants) de D', puis il est nécessaire qu'ils admettent un minimum (resp. un maximum). L'existence des bornes supérieures (ou inférieures) pour certaines familles de sous-ensembles du domaine D caractérise et distingue les ordres partiels. Si un ordre partiel $\Gamma = (\mathsf{D}, \mathfrak{R})$ vérifie :

(d) $\forall x, y \in \mathsf{D}.\ \{x, y\} \in \mathrm{dom}(\sup_\Gamma)$

nous disons que Γ est un *sup-demi-treillis.* De façon duale, si :

(d') $\forall x, y \in \mathsf{D}.\ \{x, y\} \in \mathrm{dom}(\inf_\Gamma)$

nous disons que Γ est un *inf-demi-treillis.* Si l'ordre partiel vérifie simultanément les deux conditions (d) et (d'), nous disons que Γ est un *treillis.* Il s'agit d'une notion de complétude par sous-ensembles binaires $\{x, y\}$. Si l'ordre partiel vérifie :

(e) $\Delta(\Gamma) \subseteq \mathrm{dom}(\sup_\Gamma)$

nous disons que $(\mathsf{D}, \mathfrak{R})$ est un *dcpo*, ou que l'ordre partiel est *complet par dirigés.* De façon duale, si :

(e') $\nabla(\Gamma) \subseteq \mathrm{dom}(\inf_\Gamma)$

nous disons que Γ est un *co-dcpo*, ou que l'ordre partiel est *complet par co-dirigés.* Si l'ordre partiel vérifie simultanément les deux conditions (e) et (e'), nous disons que Γ est un *bi-dcpo.* Si l'ordre partiel vérifie :

(f) $\mathrm{dom}(\sup_\Gamma) = \wp^{\mathsf{D}}$

nous disons que Γ est un *sup-demi-treillis complet.* De façon duale, si :

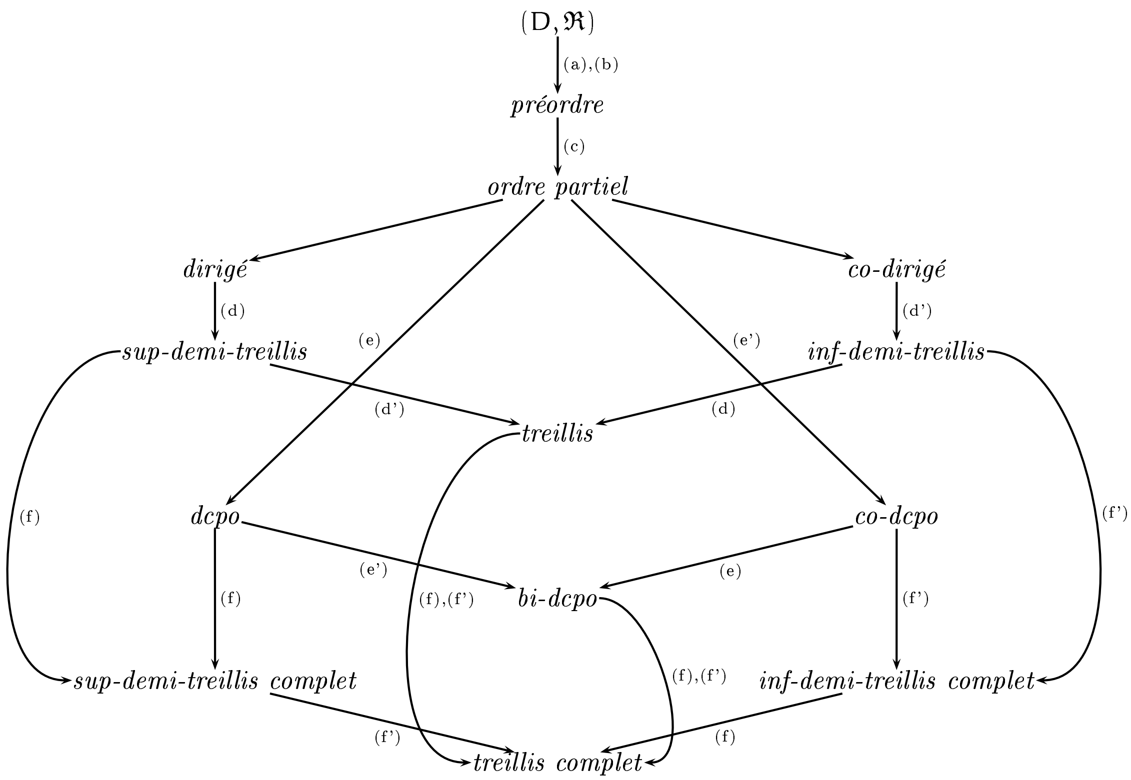
(f') $\mathrm{dom}(\inf_\Gamma) = \wp^{\mathsf{D}}$

nous disons que Γ est un *inf-demi-treillis complet.* Si l'ordre partiel vérifie simultanément les deux conditions (f) et (f'), nous disons que Γ est un *treillis complet.*

Les implications logiques entre les différentes notions de complétude ne sont pas, au premier abord, toutes évidentes : la taxinomie de la table 2 s'efforce de mettre un peu d'ordre dans les ordres.

Plus généralement, étant donné une famille de parties $\mathcal{F} : \wp^{(\wp^{\mathsf{D}})}$, nous disons que l'ordre partiel $(\mathsf{D}, \leq)$ est *complet vers le haut* (resp. *vers le bas*) par rapport à $\mathcal{F}$ ou *sup-complet* (resp. *inf-complet*) dans $\mathcal{F}$, si et seulement si $\forall \mathsf{X} \in \mathcal{F}.\ \mathsf{X} \in \mathrm{dom}(\sup_\Gamma)$ (resp. $\forall \mathsf{X} \in \mathcal{F}.\ \mathsf{X} \in \mathrm{dom}(\inf_\Gamma \mathsf{X})$). En d'autre termes, un ordre partiel est complet vers le haut (resp. bas) pour une famille $\mathcal{F}$ de parties, si la fonction *partielle* sup (resp. inf) restreinte à $\mathcal{F}$ se révèle *totale* : $\sup_{|\mathcal{F}} : \mathcal{F} \to \mathsf{D}$ (resp. $\inf_{|\mathcal{F}} : \mathcal{F} \to \mathsf{D}$).

TAB. 2. Taxinomie des notions de complétude



2.3. Homomorphismes d'ordres partiels

2.3.1. Monotonie (croissance). Étant donné deux structures d'ordres partiels $(D, \leq)$ et $(D', \leq')$, la composante ψ (interprétation des noms) de tout homomorphisme $(\psi, \varphi) : (D, \leq) \xrightarrow{\mathcal{H}} (D', \leq')$ est forcément la fonction $\psi(D) = D'$ et $\psi(\leq) = \leq'$. Nous pourrons donc toujours la considérer implicite et noter $\varphi : (D, \leq) \xrightarrow{\mathcal{H}} (D', \leq')$. Dans le contexte des ordres partiels, les homomorphismes ou, plus précisément, leur interprétation des valeurs $\varphi : D \to D'$, sont appelés *fonctions monotones* (ou *croissantes*). La spécificité de telles structures permet de réécrire la condition d'homomorphisme (cf. section 1.3.4.4), c'est-à-dire de monotonie, de la façon suivante :

$$\forall x, y \in D.\ x \leq y \ \Rightarrow\ \varphi(x) \leq' \varphi(y)$$

Comme tout homomorphisme, la fonction transpose les élément de D en élément de D' en respectant la relation d'ordre : ce qui est plus petit au départ le demeure à l'arrivée. D'une façon analogue, la notion de *décroissance* coïncide avec celle d'homomorphisme entre $(D, \leq)$ et l'ordre partiel dual $(D', \leq')^{\perp} = (D', \geq')$.

2.3.1.1. *Points fixes.* Dans le cas spécial où la fonction monotone projette dans le même ordre partiel, c'est-à-dire dans le cas où $(D, \leq)$ et $(D', \leq')$ coïncident (f est donc un endomorphisme), nous appellerons *point fixe de* f tout élément $x \in D$ tel que $f(x) = x$. Nous appellerons fix : $(D \to D) \to 1^D$ l'application qui associe à toute fonction $f : D \to D$ le semi-prédicat représentant l'ensemble de ses points fixes :

TAB. 3. Signature complète d'un ordre partiel

$(\mathsf{D}, \leq, \Delta, \nabla, \mathrm{lb}, \mathrm{ub}, \mathrm{lbs}, \mathrm{ubs}, \downarrow, \uparrow,$				
$\mathcal{L}, \mathcal{U}, \mathcal{I}, \mathcal{F}, \leq^{e}, \leq^{\flat}, \leq^{\sharp}, \approx^{\flat}, \approx^{\sharp},$				
$\min_{(\mathsf{D},\leq)}$	:	$\wp^{\mathsf{D}} \mathrel{-\!\!\circ\!\!\rightarrow} \mathsf{D}$	,	*minimum*
$\max_{(\mathsf{D},\leq)}$	:	$\wp^{\mathsf{D}} \mathrel{-\!\!\circ\!\!\rightarrow} \mathsf{D}$	,	*maximum*
$\bot$	:	$\mathrel{-\!\!\circ\!\!\rightarrow} \mathsf{D}$	,	*bottom*
$\top$	:	$\mathrel{-\!\!\circ\!\!\rightarrow} \mathsf{D}$	,	*top*
$\inf_{(\mathsf{D},\leq)}$	:	$\wp^{\mathsf{D}} \mathrel{-\!\!\circ\!\!\rightarrow} \mathsf{D}$	,	*borne inférieure*
$\sup_{(\mathsf{D},\leq)}$	:	$\wp^{\mathsf{D}} \mathrel{-\!\!\circ\!\!\rightarrow} \mathsf{D}$	,	*borne supérieure*
$\sqcap$	:	$\wp^{\mathsf{D}} \mathrel{-\!\!\circ\!\!\rightarrow} \mathsf{D}$	,	*inf-co-dirigé*
$\sqcup$	:	$\wp^{\mathsf{D}} \mathrel{-\!\!\circ\!\!\rightarrow} \mathsf{D}$	,	*sup-dirigé*
fix	:	$(\mathsf{D} \to \mathsf{D}) \to 1^{\mathsf{D}}$	,	*points fixes*
$\mu_{(\mathsf{D},\leq)}$	:	$(\mathsf{D} \to \mathsf{D}) \mathrel{-\!\!\circ\!\!\rightarrow} \mathsf{D}$	,	*plus petit point fixe*
$\nu_{(\mathsf{D},\leq)}$	:	$(\mathsf{D} \to \mathsf{D}) \mathrel{-\!\!\circ\!\!\rightarrow} \mathsf{D}$	)	*plus grand point fixe*

$\mathrm{dom}(\mathrm{fix}(f)) = \{x \in \mathsf{D} \mid f(x) = x\}$. Nous utiliserons, comme d'habitude, la notation allégée $\mathrm{fix}(f)$ aussi bien pour le semi-prédicat que pour son domaine (l'ensemble des points fixes). Nous définissons le *plus petit point fixe* $\mu(f) \stackrel{\leftarrow}{=} \min_{(\mathsf{D},\leq)} \mathrm{fix}(f)$, et le *plus grand point fixe* $\nu(f) \stackrel{\leftarrow}{=} \max_{(\mathsf{D},\leq)} \mathrm{fix}(f)$. Les fonctions partielles μ et ν de type $(\mathsf{D} \to \mathsf{D}) \mathrel{-\!\!\circ\!\!\rightarrow} \mathsf{D}$, que nous noterons aussi $\mu_{(\mathsf{D},\leq)}$ et $\nu_{(\mathsf{D},\leq)}$, sont appelées *opérateurs de point fixe*. Pour certaines familles d'ordres partiels et restreintes à certaines classes de fonctions, les opérateurs de point fixe deviennent des fonctions totales.

THEOREM 2.3.1. (TARSKI'S FIXPOINT) *Si* $(\mathsf{D}, \leq)$ *est un treillis complet* et $f : (\mathsf{D}, \leq) \xmapsto{\mathcal{H}} (\mathsf{D}, \leq)$ *est une fonction monotone (endomorphisme), alors l'ensemble des points fixes* $\mathrm{fix}(f)$ *équipé de la relation* $\leq$ *est, lui-même, un treillis complet. Par conséquent :* $\exists\, \mu(f)$ *et* $\exists\, \nu(f)$. □

2.3.1.2. *Opérateurs de clôture.* Étant donné un ordre partiel $(\mathsf{D}, \leq)$, un *opérateur de clôture* (*vers le haut*) est une fonction *monotone* $\rho : (\mathsf{D}, \leq) \xmapsto{\mathcal{H}} (\mathsf{D}, \leq)$ (*endomorphisme*) telle que $\forall x \in \mathsf{D}.\ \rho(\rho(x)) = \rho(x)$ (*idempotente*) et telle que $\forall x \in \mathsf{D}.\ x \leq \rho(x)$ (*extensive*). La notion duale, d'*opérateur de clôture vers le bas* est obtenue en remplaçant la propriété extensive par la duale, qu'on appellera *contraction* (ou *réductive*), i.e. $\forall x \in \mathsf{D}.\ \rho(x) \leq x$. La propriété d'idempotence implique dans les deux cas (vers le bas ou vers le haut) que l'ensemble des points fixes d'un opérateur de clôture coïncide avec son image : $\mathrm{fix}(\rho) = \mathrm{Im}_{\rho}\mathsf{D}$.

2.3.2. Signature d'un ordre partiel. La signature *complète* d'un ordre partiel, illustrée par la table 3, diffère de celle des préordres par le type plus précis des opérations min et max, et par la présence des nouvelles opérations sup, inf,$\sqcup$,$\sqcap$ et fix, μ, ν.

2.3.3. Étendue d'un homomorphisme d'ordres partiels. Les fonctions monotones sont des homomorphismes des structures essentielles $(\mathsf{D}, \leq)$ et $(\mathsf{D}', \leq')$. Si l'unique opération de la structure $(\mathsf{D}, \leq)$, la relation d'ordre $\leq$, est conservée par une fonction monotone, la question se pose de savoir quelles autres opérations sont *systématiquement* conservées. Parmi les opérations forcément conservées nous retrouverons les propriétés (en tant que semi-prédicats) *dirigé* $\Delta(\mathsf{D}, \leq) : 1^{(\wp^{\mathsf{D}})}$ et *co-dirigé* $\nabla(\mathsf{D}, \leq) : 1^{(\wp^{\mathsf{D}})}$, les propriétés *minorant* $\mathrm{lb} : 1^{\mathsf{D} \times 2^{\mathsf{D}}}$ et *majorant* $\mathrm{lb} : 1^{\mathsf{D} \times 2^{\mathsf{D}}}$, et les fonctions partielles *minimum* et *maximum* $\min, \max : \wp^{\mathsf{D}} \multimap \mathsf{D}$. En d'autres termes, l'image d'un dirigé est un dirigé et l'image d'un co-dirigé est un co-dirigé, l'image d'un minorant (resp. majorant) d'un ensemble est un minorant (resp. majorant) de l'image de l'ensemble, l'image du minimum (resp. maximum) d'un ensemble est le minimum (resp. maximum) de l'image de l'ensemble. La propriété *point fixe* $\mathrm{fix} : (\mathsf{D} \to \mathsf{D}) \to 1^{\mathsf{D}}$ et les *opérateurs de* (plus petit et plus grand) *point fixe* $\mu, \nu : (\mathsf{D} \to \mathsf{D}) \multimap \mathsf{D}$, comme les propriétés de *co-finalité* et *co-initialité* (préordre et équivalence) seront, elles aussi, systématiquement conservées par toute fonction monotone. Enfin, l'ordre extensionnel $\leq^e$: $\forall \alpha : \mathsf{base}.\, 1^{(\alpha \to \mathsf{D}) \times (\alpha \to \mathsf{D})}$, qui est une construction polymorphe, sera lui aussi conservé par toute application monotone.

PROPOSITION 2.3.2. *Soit* $\varphi : (\mathsf{D}, \leq) \overset{\mathcal{H}}{\longmapsto} (\mathsf{D}', \leq')$ *un homomorphisme d'ordres partiels. Alors :*

$$\varphi : \left(\begin{array}{c} \mathsf{D}, \leq, \Delta, \nabla, \mathrm{lb}, \mathrm{ub}, \min, \max, \\ \leq^e, \leq^\flat, \leq^\sharp, \approx^\flat, \approx^\sharp, \mathrm{fix}, \mu, \nu \end{array} \right) \overset{\mathcal{H}}{\longmapsto} \left(\begin{array}{c} \mathsf{D}', \leq', \Delta', \nabla', \mathrm{lb}', \mathrm{ub}', \min', \max', \\ \leq'^e, \leq'^\flat, \leq'^\sharp, \approx'^\flat, \approx'^\sharp, , \mathrm{fix}', \mu', \nu' \end{array} \right)$$

est encore un homomorphisme.

DÉMONSTRATION. (Dirigés) *Soit* $\mathsf{X} \in \Delta(\mathsf{D}, \leq)$. *On veut prouver que* $\mathsf{Y} \triangleq \mathrm{Im}_\varphi \mathsf{X}$ *est un dirigé de* $(\mathsf{D}', \leq')$. *Soient* $y_1, y_2 \in \mathsf{Y}$. *Par définition de* Y *nous avons* $y_1 = \varphi(x_1)$ *et* $y_2 = \varphi(x_2)$ *où* $x_1, x_2 \in \mathsf{X}$. *Par définition de dirigé, il existe* x_3 *tel que* $x_1 \leq_1 x_3$ *et* $x_2 \leq_1 x_3$. *Il existe donc* $y_3 = \varphi(x_3)$ *tel que, par monotonie de* φ, $y_1 \leq_1 y_3$ *et* $y_2 \leq_1 y_3$, c.q.f.d. *(Minorant) Soit* $y \in \mathsf{D}$ *et* $\mathsf{X} \in 2^{\mathsf{D}}$ *tels que* $\mathrm{lb}(y, \mathsf{X})$. *Soit* $y' \triangleq \varphi(y)$ *et* $x' \in \mathsf{X}' \triangleq \varphi(\mathsf{X})$ *choisi arbitrairement. Nous avons* $x' = \varphi(x)$ *pour un certain* $x \in \mathsf{X}$. *Puisque* $y \leq x$, *par monotonie,* $y' \leq x'$, *donc* $\mathrm{lb}(y', \mathsf{X}')$. *(Minimum) Soit* $\mathsf{X} \in \wp^{\mathsf{D}}$ *et* $m = \min \mathsf{X}$. *Soit* $m' \triangleq \varphi(m)$ *et* $\mathsf{X}' \triangleq \varphi(\mathsf{X})$. *L'appartenance* $m \in \mathsf{X}$ *implique* $m' \in \mathsf{X}'$. *De plus, pour tout* $x' \in \mathsf{X}'$, *il existe* $x \in \mathsf{X}$ *tel que* $x' = \varphi(x)$. *Et puisque* $m \leq x$, *nous avons* $m' \leq' x'$, c.q.f.d. (extensionnel) *Soit* X *un domaine de base arbitraire. Il faut montrer que* $\leq^e_{\mathsf{X}} \underset{1^{(\mathsf{X} \to \mathsf{D}) \times (\mathsf{X} \to \mathsf{D})}}{\overset{\varphi}{\longmapsto}} \leq'^e_{\mathsf{X}}$. *Supposons alors que deux fonctions* $f, g : \mathsf{X} \to \mathsf{D}$ *soient telles que* $f \leq^e_{\mathsf{X}} g$, *et qu'il existe un couple* (f', g') *correspondant dans la seconde algèbre. Dans le cas des fonctions de type élémentaire,* $f \underset{\mathsf{X} \to \mathsf{D}}{\overset{\varphi}{\longmapsto}} f'$ *signifie que* $\forall x \in \mathsf{X}.\ \varphi(f(x)) = f'(\vec{\varphi}(x))$, *et* $g \underset{\mathsf{X} \to \mathsf{D}}{\overset{\varphi}{\longmapsto}} g'$ *signifie que* $\forall x \in \mathsf{X}.\ \varphi(g(x)) = g'(\vec{\varphi}(x))$. *De plus, par définition de l'ordre extensionnel,* $\forall x \in \mathsf{X}.\ f(x) \leq g(x)$. *Donc, par monotonie de* φ *et considérant que* $\vec{\varphi}(x) = x$, *nous avons, pour tout* $x \in \mathsf{X}$, $f'(x) = f'(\vec{\varphi}(x)) = \varphi(f(x)) \leq' \varphi(g(x)) = g'(\vec{\varphi}(x)) = g'(x)$, c.q.f.d. (co-finalité) *Soit* $\mathsf{X} \leq^\flat \mathsf{Y}$. *On veut prouver que* $\mathsf{X}' \triangleq \mathrm{Im}_\varphi \mathsf{X}$ *et* $\mathsf{Y}' \triangleq \mathrm{Im}_\varphi \mathsf{Y}$ *sont dans la relation analogue :* $\mathsf{X}' \leq'^\flat \mathsf{Y}'$. *Soit* $x' \in \mathsf{X}'$. *Il existe alors* $x \in \mathsf{X}$ *tel que* $x' = \varphi(x)$. *Puisque* $\mathsf{X} \leq^\flat \mathsf{Y}$, *il existe* $y \in \mathsf{Y}$

tel que $x \le y$*. Il existe donc* $y' = \varphi(y) \in Y'$ *tel que, par monotonie de* φ, $x' \le' y'$ c.q.f.d. *(Point fixes) Supposons* $f \underset{D \to D}{\overset{\varphi}{\longmapsto}} f'$*, ce qui équivaut à la condition* $\forall x \in D.\ \varphi(f(x)) = f'(\varphi(x))$*. Il nous faut prouver que* $\mathrm{fix}(f) \underset{D \multimap 1}{\overset{\varphi}{\longmapsto}} \mathrm{fix}'(f')$*, autrement dit, que* φ *transforme tout point fixe de* f *en un point fixe de* f'*. Soit donc* $x = f(x)$*. En appliquant l'hypothèse,* $\varphi(x) = \varphi(f(x)) = f'(\varphi(x))$, c.q.f.d. *Les preuves pour les co-dirigés, le majorant, le maximum et le préordre de co-initialité sont les duales des précédentes. Les équivalences* $\approx^\flat$, $\approx^\sharp$ *sont dérivées, par intersection, d'opérations qui se conservent (cf. prop. 1.3.8). Les opérateurs de point fixe* μ *et* ν *sont eux aussi dérivés, mais par composition fonctionnelle (respectivement de* fix *et* min*, et de* fix *et* max*), d'opérations qui se conservent (cf. prop. 1.3.8).* □

Les autres opérations des ordres partiels, résumées dans les tables 1 et 3, ne se conservent pas forcément. La structure $(\{1,2,3\}, \le)$ où $\le = \{(1,3),(2,3)\}$ est une extension homomorphe de $(\{1,2\}, \le')$ où $\le' = \{(1,3)\}$. L'association monotone et surjective est $\varphi(1) = \varphi(2) = 1$, $\varphi(3) = 3$. Les *majorants* de $X = \{1,2\}$ sont $Y = \{3\}$, mais les majorants de l'image de X, i.e. $\{1,3\}$ ne coïncident pas avec l'image de Y qui est $\{3\}$. Si on ajoute à la seconde structure un élément 0 plus petit que les autres, l'image de la *clôture vers le bas* sera $\varphi(\downarrow X) = \{1\}$ tandis que $\downarrow \varphi(X) = \{0,1\}$. De même, X est un *idéal* et il est *clos vers le bas* (*lower set*), tandis que $\varphi(X)$ n'est ni l'un ni l'autre. Enfin, si l'image de la *borne supérieure* de X est égale à $\varphi(\sup X) = \varphi(3) = 3$, la borne supérieure de l'image est, en revanche, égale à $\sup(\varphi(X)) = \sup\{1\} = 1$. En retournant les deux ordres (l'association reste monotone) les parties sélectionnées précédemment nous donnent les contre-exemples pour les opérations duales telles que les *minorants*, la *clôture vers le haut*, les *filtres,* les *clos vers le haut* (*upper sets*) et la *borne inférieure.*

La conservation des propriétés dirigé, co-dirigé, minorant, majorant et point fixe motive le statut de semi-prédicat que nous leurs avons affecté. La monotonie d'une fonction $\varphi : D \to D$ est une condition qui se traduit comme la conservation du *semi-prédicat* $\le$: $1^{D \times D}$. Ce rang, qui implique au départ la plus faible des conditions de conservation pour une propriété (cf. section 1.3.4.3), n'aurait pas suffit pour obtenir, à l'arrivée, qu'une propriété telle que celle de dirigé $\Delta : 1^{(\wp^D)}$ se conserve en tant que prédicat $\Delta : 2^{(\wp^D)}$ ou en tant que partie $\Delta : \wp^{(\wp^D)}$.

2.3.4. Étendue d'un isomorphisme d'ordres partiels. Les isomorphismes d'ordres partiels (de leurs structures *essentielles*) sont systématiquement des isomorphismes de leurs structures *complètes.*

Proposition 2.3.3. *Soit* $(D, \le) \overset{\varphi}{\cong} (D', \le')$ *un isomorphisme d'ordres partiels. Alors, les deux structures complètes*

$$\Sigma = (D, \le, \Delta, \nabla, \mathrm{lb}, \mathrm{ub}, \mathrm{lbs}, \mathrm{ubs}, \min, \max, \downarrow, \uparrow, \mathcal{L}, \mathcal{U}, \mathcal{I}, \mathcal{F},$$
$$\inf, \sup, \sqcap, \sqcup, \le^e, \le^\flat, \le^\sharp, \approx^\flat, \approx^\sharp, \mathrm{fix}, \mu, \nu)$$

et

$$\Sigma' = (D', \le', \Delta', \nabla', \mathrm{lb}', \mathrm{ub}', \mathrm{lbs}', \mathrm{ubs}', \min', \max', \downarrow', \uparrow', \mathcal{L}', \mathcal{U}', \mathcal{I}', \mathcal{F}',$$
$$\inf', \sup', \sqcap', \sqcup', \le'^e, \le'^\flat, \le'^\sharp, \approx'^\flat, \approx'^\sharp, \mathrm{fix}', \mu', \nu')$$

sont, elles aussi, isomorphes par la même bijection :

$$\Sigma \overset{\varphi}{\cong} \Sigma'$$

DÉMONSTRATION. *En premier lieu, nous devons montrer que* φ *est un homomorphisme de* Σ *vers* Σ'*. Si la Prop. 2.3.2 assure la conservation des opérations* $\Delta, \nabla, \mathrm{lb}, \mathrm{ub}, \min, \max, \leq^e, \leq^\flat, \leq^\sharp, \approx^\flat, \approx^\sharp, \mathrm{fix}, \mu, \nu$*, il reste encore à le prouver, cas par cas, pour les autres opérations* $\mathrm{lbs}, \mathrm{ubs}, \downarrow, \uparrow, \mathcal{L}, \mathcal{U}, \mathcal{I}, \mathcal{F}, \inf, \sup$*. (Minorants) Il faut montrer que* $\varphi(\mathrm{lbs}\, X) = \mathrm{lbs}(\varphi(X))$*. L'inclusion* $\subseteq$ *est assurée par la Prop. 2.3.2 puisque l'image d'un minorant d'un ensemble appartient aux minorants de l'image de l'ensemble (conservation de* lb*). Soit* $y' \in \mathrm{lbs}\, \varphi(X)$*. Donc* $y' \leq' x'$ *pour tout* $x' \in \varphi(X)$*. En appliquant aux deux membres la fonction inverse* φ^{-1} *(qui est elle aussi monotone) on obtient* $\varphi^{-1}(y') \leq \varphi^{-1}(x')$*, donc pour tout* $x \in X$ *l'élément* $y = \varphi^{-1}(y')$ *est tel que* $y \leq x$*. Autrement dit,* $y \in \mathrm{lbs}\, X$*, donc* $y' = \varphi(y) \in \varphi(\mathrm{lbs}\, X)$ c.q.f.d. *Les preuves pour les opérations* $\downarrow$ *et* $\mathcal{L}$ *se construisent de façon analogue en s'appuyant sur la monotonie des deux fonctions inverses. La propriété* idéaux $\mathcal{I}$ *est l'intersection de deux opérations qui se conservent, les* dirigés *(cf. prop. 2.3.2) et les* lower sets*, donc elle se conserve (cf. prop. 1.3.8). La* borne inférieure *est la composition (*$\inf = \max \circ \mathrm{lbs}$*) de deux opérations qui se conservent, le* maximum *(cf. prop. 2.3.2) et les* minorants*, donc elle se conserve (cf. prop. 1.3.8). Le* sup-dirigé *est la composition (*$\sqcup = \sup_{\Delta^+}$*) de la fonction* sup *qui se conserve et de la propriété dirigé non vide* Δ^+ *qui se conserve aussi (intersection de deux propriétés qui se conservent), donc elle se conserve (cf. prop. 1.3.8). La conservation des autres opérations est prouvée par dualité. En deuxième lieu, il faut prouver que l'inverse* φ^{-1} *est, elle aussi, un homomorphisme. Puisque la fonction inverse est monotone, nous pouvons appliquer la Prop. 2.3.2. Pour les opérations restantes, les arguments utilisés pour* φ *demeurent valides, par symétrie, pour l'inverse* φ^{-1}*, qui est donc un homomorphisme.* □

2.4. Dcpo et co-Dcpo

Une conséquence importante de la définition donné de dcpo est que tout dcpo $(D, \leq)$ est *pointé vers le bas*[4]. En effet, puisque $\emptyset \in \Delta(D, \leq)$, nous avons, par complétude, qu'il existe $\bot \triangleq \sup_{(D,\leq)} \emptyset = \min_{(D,\leq)}(\mathrm{ubs}_{(D,\leq)} \emptyset) = \min_{(D,\leq)} D$. Ainsi, la constante partielle $\bot \multimap D$, devient pour un dcpo une constante $\bot :\to D$. Pour des raisons duales, tout co-dcpo $(D, \leq)$ est *pointé vers le haut* : $\top \triangleq \inf_{(D,\leq)} \emptyset = \max_{(D,\leq)}(\mathrm{lbs}_{(D,\leq)} \emptyset) = \max_{(D,\leq)} D$, donc la constante partielle $\top \multimap D$ devient une constante $\top :\to D$. Une deuxième conséquence est que la fonction partielle sup-dirigé $\sqcup : \wp^D \multimap D$ devient totale dans le sous-domaine des dirigés non vides :

$$\sqcup : \Delta^+(D, \leq) \to D$$

[4] Ceci n'est pas forcément vrai pour la définition alternative de dirigé que l'on peut retrouver dans d'autres travaux et qui n'inclut pas l'ensemble vide.

2.4.1. Continuité. Si le théorème de Tarski assure l'existence d'un plus petit et d'un plus grand point fixe, il ne préfigure aucune façon de les calculer. Un autre inconvénient, plus fastidieux, est que le statut de treillis complet requis à l'ordre partiel n'est pas une condition des plus fréquentes.

2.4.1.1. *Continuité vers le haut.* Une fonction monotone $f:(D,\leq) \xrightarrow{\mathcal{H}} (D',\leq')$ entre deux *dcpo* est *continue* (ou *continue vers le haut*) si :

$$\forall X \in \Delta^{+}(D,\leq). \qquad f(\sqcup X) = \sqcup' f(X)$$

Les deux membres de l'équation existent toujours. D'un coté X est un dirigé et $(D,\leq)$ un dcpo. De l'autre, la Prop. 2.3.2 assure que $f(X)$ est un dirigé, alors que $(D',\leq')$ est lui aussi un dcpo. De plus, par définition d'image, il s'agit d'un dirigé non vide, étant X non vide.

On exclut le dirigé vide de la définition de continuité pour ne pas exiger que la fonction soit *stricte*, i.e. $f(\perp) = \perp'$. Une autre manière d'expliquer la continuité d'une fonction entre dcpo est celle de dire que l'*homomorphisme* entre $(D,\leq)$ et $(D',\leq')$ reste tel lorsqu'on ajoute aux structures l'opération sup-dirigé :

$$f : (D, \leq, \sqcup : \wp^{D} \multimap D) \xrightarrow{\mathcal{H}} (D', \leq', \sqcup' : \wp^{D'} \multimap D')$$

De la même façon, une fonction est stricte si, en ajoutant la constante bottom aux deux structures $(D,\leq)$ et $(D',\leq')$, l'application est encore un homomorphisme : $f : (D,\leq,\perp) \xrightarrow{\mathcal{H}} (D',\leq',\perp')$. Une fonction à la fois stricte et continue est donc un homomorphisme de type $f : (D,\leq,\perp,\sqcup) \xrightarrow{\mathcal{H}} (D',\leq',\perp',\sqcup')$. On remarquera que, par la Prop. 2.3.3, tout isomorphisme entre dcpo sera forcément une fonction *continue* et *stricte*.

2.4.1.2. *Continuité vers le bas.* De façon duale, une fonction monotone $f : (D,\leq) \xrightarrow{\mathcal{H}} (D',\leq')$ entre deux *co-dcpo* est *co-continue (*ou *continue vers le bas)* si :

$$\forall X \in \nabla^{+}(D,\leq). \qquad f(\sqcap X) = \sqcap' f(X)$$

À nouveau, l'hypothèse que les deux ordres partiels soient des co-dcpo et la Prop. 2.3.2 assurent l'existence des deux membres de l'équation. On exclut le co-dirigé vide pour ne pas exiger que la fonction soit *co-stricte*, i.e. $f(\top) = \top'$. Traduisant, une fois de plus, la définition en termes algébriques, une fonction *co-continue* f sur deux co-dcpo est un *homomorphisme* de type :

$$f : (D, \leq, \sqcap : \wp^{D} \multimap D) \xrightarrow{\mathcal{H}} (D', \leq', \sqcap' : \wp^{D'} \multimap D')$$

Une fonction co-stricte est un homomorphisme de type $f : (D,\leq,\top) \xrightarrow{\mathcal{H}} (D',\leq',\top')$. Une fonction à la fois co-stricte et co-continue est un homomorphisme de type $f : (D,\leq,\top,\sqcap) \xrightarrow{\mathcal{H}} (D',\leq',\top',\sqcap')$. La Prop. 2.3.3 implique alors que tout isomorphisme entre co-dcpo sera forcément une fonction *co-continue* et *co-stricte*.

2.4.1.3. *Théorème de Kleene.* Le théorème de Kleene est, par rapport au théorème de Tarsky, d'une majeure utilité pratique : nous pouvons construire le plus petit point fixe par les puissances naturelles de la fonction continue (i.e. son application réitérée, section 1.1.5).

THEOREM 2.4.1. (KLEENE'S FIXPOINT) *Si* $(D, \leq, \bot, \sqcup)$ *est un dcpo et la fonction* $f : (D, \leq, \bot, \sqcup)) \overset{\mathcal{H}}{\longmapsto} (D, \leq, \bot, \sqcup))$ *est continue, alors* $\exists\, \mu(f) = \sqcup\{f^n(\bot) \mid n \in \mathbb{N}\}$ *où* f^n *est la puissance* n *de la fonction* f. *De façon duale, si* $(D, \leq, \top, \sqcap)$ *est un co-dcpo et* $f : (D, \leq, \top, \sqcap) \overset{\mathcal{H}}{\longmapsto} (D, \leq, \top, \sqcap)$ *est une fonction co-continue, alors* $\exists\, \nu(f) = \sqcap\{f^n(\top) \mid n \in \mathbb{N}\}$. □

2.4.1.4. *Principe d'induction des points fixes.* On peut établir un principe d'induction, basé sur le théorème de Kleene, qui est de grande utilité pratique quand il s'agit de prouver une propriété d'un objet mathématique défini comme plus petit (resp. plus grand) point fixe d'une fonction continue (resp. co-continue). Étant donné un ordre partiel $\Gamma = (D, \leq)$ vérifiant les hypothèses du théorème de Kleene, pour démontrer que le plus petit point fixe $\mu(f)$ vérifie une propriété $p : \wp^D \oplus 2^D \oplus 1^D$, il suffira de montrer que :

(1) $p(\bot)$

(2) $p(x) \;\Rightarrow\; p(f(x))$

(3) $\forall s : D^{\omega}.\ (\forall i \in \mathbb{N}.\ p(s_i)) \;\wedge\; (s \text{ croissante (i.e. } s_i \leq s_{i+1})) \;\Rightarrow\; p(\sup_{\Gamma}(\mathrm{cod}\, s))$

D'autre part, dans les hypothèses duales du théorème, pour démontrer que le plus grand point fixe $\nu(f)$ vérifie p, il suffira de montrer que :

(1) $p(\top)$

(2) $p(x) \;\Rightarrow\; p(f(x))$

(3) $\forall s : D^{\omega}.\ (\forall i \in \mathbb{N}.\ p(s_i)) \;\wedge\; (s \text{ décroissante (i.e. } s_{i+1} \leq s_i)) \;\Rightarrow\; p(\inf_{\Gamma}(\mathrm{cod}\, s))$

2.4.2. Compacts, algébricité, domaines de Scott. Étant donné un dcpo $(D, \leq, \bot, \sqcup)$, un élément $k \in D$ est *compact* si :

$$\forall X \in \Delta^+(D, \leq). \quad \sqcup X \geq k \;\Rightarrow\; \exists x \in X.\ x \geq k$$

L'ensemble des éléments compacts du dcpo sera noté $\mathcal{K}(D, \leq)$. La *fermeture compacte vers le bas* est l'opération $\not\downarrow_{(D,\leq)}(.) : \wp^D \to \wp^D$ définie par $\not\downarrow_{(D,\leq)}(X) \triangleq \mathcal{K}(D, \leq) \cap \downarrow_{(D,\leq)}(X)$. L'écriture $\not\downarrow_{(D,\leq)} x$, où $x \in D$, dénote le sous-ensemble $\not\downarrow_{(D,\leq)}\{x\}$ appelé *cône compacts* de x. La composition de la propriété compacte avec, d'une part, celle de dirigé et, d'autre part, celle d'idéal, est d'un intérêt particulier. Les *dirigés de compacts* sont les dirigés de la restriction de l'ordre partiel au sous-ensemble des éléments compacts : $\Delta\mathcal{K}(D, \leq) \triangleq \Delta(\mathcal{K}(D, \leq), \leq_{|\mathcal{K}(D,\leq)})$. Autrement dit, il s'agit de l'intersection des dirigés avec les parties compactes : $\Delta\mathcal{K}(D, \leq) = \Delta(D, \leq) \cap \wp^{\mathcal{K}(D,\leq)}$. De la même manière, les *idéaux de compacts* sont l'ensemble $\mathcal{IK}(D, \leq) \triangleq \mathcal{I}(\mathcal{K}(D, \leq), \leq_{|\mathcal{K}(D,\leq)})$, autrement dit, $\mathcal{IK}(D, \leq) = \mathcal{I}(D, \leq) \cap \wp^{\mathcal{K}(D,\leq)}$. La *partie compacte,* qui est une propriété de type $\wp^D$, les *idéaux de compacts* et les *dirigés de compacts*, de type $1^{(\wp^D)}$, ainsi que la *fermeture compacte vers le bas* et la constante *bottom*, s'ajoutent à la signature complète d'un ordre partiel, lorsque ce dernier est un dcpo (Table 4).

La borne supérieure d'un cône compact $\not\downarrow_{(D,\leq)} x$, si elle existe, est appelée *sommet*. Un dcpo est *algébrique* si tout élément est le sommet de son propre cône compact, qui est un dirigé :

$$\forall x \in D. \quad (\not\downarrow x) \in \Delta(D, \leq) \;\wedge\; \sqcup(\not\downarrow x) = x$$

TAB. 4. Signature complète d'un dcpo

$(\mathsf{D}, \leq$	:	$1^{\mathsf{D}\times\mathsf{D}}$	,	
$\Delta, \nabla, \mathcal{L}, \mathcal{U}, \mathcal{I}, \mathcal{F}$	:	$1^{(\wp^{\mathsf{D}})}$	,	
$\leq^\flat, \leq^\sharp, \approx^\flat, \approx^\sharp$	:	$1^{(\wp^{\mathsf{D}})\times(\wp^{\mathsf{D}})}$	,	
$\leq^e$	:	$\forall\alpha{:}\mathsf{base}.\ 1^{(\alpha\to\mathsf{D})\times(\alpha\to\mathsf{D})}$	,	
lb, ub	:	$1^{\mathsf{D}\times\wp^{\mathsf{D}}}$	,	
$\mathrm{lbs}, \mathrm{ubs}, \downarrow, \uparrow$	:	$\wp^{\mathsf{D}} \to \wp^{\mathsf{D}}$	,	
$\min, \max$	:	$\wp^{\mathsf{D}} \multimap \mathsf{D}$	,	
$\inf, \sup, \sqcap, \sqcup$	:	$\wp^{\mathsf{D}} \multimap \mathsf{D}$	,	
fix	:	$(\mathsf{D} \to \mathsf{D}) \to 1^{\mathsf{D}}$	,	
μ, ν	:	$(\mathsf{D} \to \mathsf{D}) \multimap \mathsf{D}$	,	
$\top$	:	$\multimap \mathsf{D}$	,	
$\bot$	:	$\to \mathsf{D}$	,	*bottom*
$\mathcal{K}(\mathsf{D}, \leq)$	:	$\wp^{\mathsf{D}}$	,	*partie compacte*
$\Delta\mathcal{K}(\mathsf{D}, \leq)$	:	$1^{(\wp^{\mathsf{D}})}$	,	*dirigés de compactes*
$\mathcal{IK}(\mathsf{D}, \leq)$	:	$1^{(\wp^{\mathsf{D}})}$	,	*idéaux de compactes*
$\not\downarrow_{(\mathsf{D},\leq)}$	:	$\wp^{\mathsf{D}} \to \wp^{\mathsf{D}}$	)	*fermeture compacte*

Un *domaine de Scott* est un dcpo $(\mathsf{D}, \leq)$ algébrique tel que tout sous-ensemble majoré a une borne supérieure :

$$\forall \mathsf{X} \in 2^{\mathsf{D}}.\ \ \mathrm{ubs}_{(\mathsf{D},\leq)}\mathsf{X} \neq \emptyset \ \Rightarrow\ \exists \sup_{(\mathsf{D},\leq)} \mathsf{X}$$

Les notions duales, pour un co-dcpo $\Gamma = (\mathsf{D}, \leq, \top, \sqcap)$ sont définies en faisant référence à la structure duale $\Gamma^\perp \triangleq (\mathsf{D}, \geq, \bot, \sqcup)$ où $\bot \triangleq \top$ et $\sqcup \triangleq \sqcap$, qui, elle, est un dcpo. Un *co-compact* de Γ est un compact de $\Gamma^\perp$. L'ensemble des co-compacts sera noté $\mathcal{K}^{\mathrm{op}}(\mathsf{D}, \leq)$. L'ensemble des co-dirigés de co-compacts sera noté $\nabla\mathcal{K}(\mathsf{D}, \leq) \triangleq \nabla(\mathsf{D}, \leq) \cap \wp^{\mathcal{K}^{\mathrm{op}}(\mathsf{D},\leq)}$. L'ensemble des filtres de co-compacts sera noté $\mathcal{FK}(\mathsf{D}, \leq) = \mathcal{F}(\mathsf{D}, \leq) \cap \wp^{\mathcal{K}^{\mathrm{op}}(\mathsf{D},\leq)}$. De toute évidence nous avons $\nabla\mathcal{K}(\mathsf{D}, \leq) = \Delta\mathcal{K}(\mathsf{D}, \geq)$ et $\mathcal{FK}(\mathsf{D}, \leq) = \mathcal{IK}(\mathsf{D}, \geq)$. Nous utiliserons la notation des dirigés non vide Δ^+, pour toute autre classe de parties : les écritures $\mathcal{K}^+, \mathcal{IK}^+, \Delta\mathcal{K}^+, \mathcal{K}^{\mathrm{op}+}, \mathcal{FK}^+, \nabla\mathcal{K}^+$ dénoteront la classe correspondante privée de la partie vide. On dit que Γ est un *co-dcpo algébrique* lorsque $\Gamma^\perp$ est un dcpo algébrique. Un *co-domaine de Scott* est un co-dcpo tel que le dual est un domaine de Scott.

Si en général la co-finalité réciproque est une condition suffisante (cf. lemme 2.2.1) mais pas nécessaire à l'égalité des bornes supérieures, les deux conditions logiques deviennent équivalentes lorsque les parties sont des dirigés de points compacts.

LEMMA 2.4.2. *Soit $(\mathsf{D}, \leq)$ un dcpo. Les dirigés non vides de compacts dont les bornes supérieures sont corrélées, sont dans un rapport de co-finalité :*

$$Hp \begin{cases} (\mathsf{D}, \leq) \ \textit{dcpo} \\ \mathsf{X}, \mathsf{Y} \in \Delta\mathcal{K}^+(\mathsf{D}, \leq) \\ \sup \mathsf{X} \leq \sup \mathsf{Y} \end{cases} \qquad Ts \left\{ \ \mathsf{X} \leq^\flat \mathsf{Y} \right.$$

Vice versa, si $(\mathsf{D}, \leq)$ est un co-dcpo, les co-dirigés non vides de co-compacts dont les bornes inférieures sont corrélées, sont dans un rapport de co-initialité :

$$Hp \begin{cases} (\mathsf{D}, \leq) \textit{ co-dcpo} \\ \mathsf{X}, \mathsf{Y} \in \nabla\mathcal{K}^+(\mathsf{D}, \leq) \\ \inf \mathsf{X} \leq \inf \mathsf{Y} \end{cases} \qquad Ts \left\{ \; \mathsf{X} \leq^\sharp \mathsf{Y} \right.$$

DÉMONSTRATION. *Nous ne démontrons que la première implication, la seconde étant duale. Soit* $x \in X$. *Nous avons* $x \leq \sup X \leq \sup Y$. *Puisque* Y *est un dirigé non vide,* x *est un point compact, et* $\sup Y \geq x$, *nous avons* $\exists y \in Y.\ y \geq x$, c.q.f.d. □

COROLLARY 2.4.3. *Soit* $(D, \leq)$ *un dcpo (resp. co-dcpo). Alors l'égalité des bornes supérieures (resp. inférieures), si elles existent, implique l'équivalence de co-finalité (resp. co-initialité). Autrement dit, nous avons, d'une part, l'implication suivante :*

$$Hp \begin{cases} (\mathsf{D}, \leq) \textit{ dcpo} \\ \mathsf{X}, \mathsf{Y} \in \Delta\mathcal{K}^+(\mathsf{D}, \leq) \\ \sup \mathsf{X} = \sup \mathsf{Y} \end{cases} \qquad Ts \left\{ \; \mathsf{X} \approx^\flat \mathsf{Y} \right.$$

et, d'autre part, l'implication duale :

$$Hp \begin{cases} (\mathsf{D}, \leq) \textit{ co-dcpo} \\ \mathsf{X}, \mathsf{Y} \in \nabla\mathcal{K}^+(\mathsf{D}, \leq) \\ \inf \mathsf{X} = \inf \mathsf{Y} \end{cases} \qquad Ts \left\{ \; \mathsf{X} \approx^\sharp \mathsf{Y} \right.$$

□

LEMMA 2.4.4. *Soit* $(D, \leq)$ *un domaine de Scott. Alors, toute partie non vide a une borne inférieure.* □

Ainsi, lorsque le domaine de Scott $(D, \leq)$ est pointé vers le haut, il s'agit forcément d'un inf-demi-treillis complet.

EXAMPLE 2.4.5. La table suivante, extraite de [**Davey & Priestley, 1990**], illustre quelques exemples canoniques de dcpo algébriques et de leurs points compacts.

hypothèse	construction	compacts $\mathcal{K}(.)$
A ensemble	$(\wp^A, \subseteq)$	$\wp_F^A$
$(D, \leq)$ ordre partiel	$(\mathcal{L}(D, \leq), \subseteq)$	$\{\downarrow X \mid X \in \wp_F^D\}$
A ensemble	$(A^{\leq\omega}, \leq)$	A^*
A, B ensembles	$(A \multimap B, \subseteq)$	$\{f : A \multimap B \mid \mathrm{dom}(f) \in \wp_F^A\}$

□

2.4.3. Isomorphismes de dcpo. Les isomorphismes entre dcpo, c'est-à-dire de leurs structures *essentielles*, sont systématiquement des isomorphismes de leurs structures *complètes*. En d'autres termes, les opérations spécifiques aux dcpo telle que *bottom*, *compacts*, *dirigés de compacts*, *idéaux de compacts* et *fermeture compacte* sont automatiquement conservées dans les deux sens.

PROPOSITION 2.4.6. *Soit* $(\mathsf{D}, \leq) \overset{\varphi}{\cong} (\mathsf{D}', \leq')$ *un isomorphisme entre dcpo. Alors, les deux structures complètes (comprenant* $\mathcal{K}$, $\Delta\mathcal{K}$, $\mathcal{IK}$ *et* $\downarrow$*) sont, elles aussi, isomorphes par la même bijection* φ.

DÉMONSTRATION. *Similaire à celle de la Prop. 2.3.3.* □

COROLLARY 2.4.7. *Soit* $(\mathsf{D}, \leq) \overset{\varphi}{\cong} (\mathsf{D}', \leq')$ *un isomorphisme entre ordres partiels. Si l'un est un dcpo l'autre l'est aussi, et si l'un est algébrique l'autre l'est aussi.*

2.5. Complétion

2.5.1. Treillis complets engendrées. L'inclusion ensembliste est une relation qui joue un rôle important dans la construction de structures avec de bonnes propriétés de complétude. Si au départ nous avons un ensemble D, éventuellement équipé d'une relation d'ordre partiel $\leq$, nous aurons à l'arrivée une structure $(\mathcal{F}, \subseteq)$ où $\mathcal{F}$ sera une *famille* de parties de D, $\mathcal{F} \in \wp^{(\wp^{\mathsf{D}})}$, équipée de l'inclusion ensembliste $\subseteq$. Ce procédé typique en théorie des domaines, réduit le problème de la construction d'un domaine *complet* $(\mathcal{X}, \leq)$ à la définition d'une famille avec des propriétés adéquates, fixant définitivement la relation d'ordre à l'inclusion ensembliste : $\leq = \subseteq$. La famille de toutes les parties possibles $\mathcal{F} = \wp^{\mathsf{D}}$, est un exemple canonique d'une telle construction.

2.5.1.1. *Familles closes par union et intersection.* Les opérations ensemblistes d'union $\bigcup_{i \in I} A_i$ et d'intersection $\bigcap_{i \in I} A_i$ se disent *finies* lorsque l'ensemble des indices I est fini. Sinon elles sont *arbitraires*.

LEMMA 2.5.1. *Étant donné un ensemble* A*, toute famille d'ensemble* $\mathcal{F} \subseteq \wp^{\mathsf{A}}$ *close par union (resp. intersection)* finie *constitue, équipée de l'inclusion ensembliste* $(\mathcal{F}, \subseteq)$*, un sup-demi-treillis (resp. inf-demi-treillis) où la borne supérieure (resp. inférieure) coïncide avec l'union (resp. intersection).* □

Donc, une famille $\mathcal{F}$ close par union et intersection *finies*, forme avec l'inclusion un treillis $(\mathcal{F}, \subseteq)$. Si tel est le cas, nous disons, pour souligner le caractère des bornes (égales à l'union et l'intersection), que la structure $(\mathcal{F}, \subseteq)$ est un *treillis d'ensembles*.

LEMMA 2.5.2. *Étant donné un ensemble* A*, toute famille d'ensemble* $\mathcal{F} \subseteq \wp^{\mathsf{A}}$ *close par union (resp. intersection)* arbitraire *constitue, équipée de l'inclusion ensembliste* $(\mathcal{F}, \subseteq)$*, un sup-demi-treillis* complet *(resp. inf-demi-treillis* complet*) où la borne supérieure (resp. inférieure) coïncide avec l'union (resp. intersection).* □

Ainsi, quand la famille $\mathcal{F}$ est close à la fois par unions et intersections *arbitraires*, le couple $(\mathcal{F}, \subseteq)$ est d'un treillis complet. Si tel est le cas, nous disons, encore pour souligner le caractère des bornes, que la structure $(\mathcal{F}, \subseteq)$ est un *treillis complet d'ensembles*. En général, le suffixe *"d'ensembles"*, peut être utilisé aussi quand la propriété de complétude est plus faible (par exemple un dcpo ou co-dcpo), lorsque les bornes supérieure et/ou inférieure coïncident encore, respectivement, avec l'union et l'intersection ensembliste. Nous pourrons dire, par exemple, qu'une structure est un dcpo (ou un co-dcpo) *d'ensembles*.

Les lemmes précédents peuvent être appliqués à plusieurs constructions que nous avons déjà définies au cours du chapitre.

2.5.1.2. *La famille des parties.* La famille de toutes les parties possibles $(\wp^{A}, \subseteq)$ est close par union et intersection arbitraire, donc est un treillis complet.

2.5.1.3. *La famille des down-sets et des upper-sets.* Si $(D, \leq)$ est un préordre, alors $(\mathcal{L}(D, \leq), \subseteq)$ et $(\mathcal{U}(D, \leq), \subseteq)$ sont clos par union et intersection arbitraires. Il s'agit donc de treillis complets.

2.5.1.4. *Systèmes de clôture.* Une famille $\mathcal{F} : \wp^{(\wp^{A})}$ de parties d'un ensemble A close par intersection arbitraire est une *structure d'intersection*[5] sur A. Un *système de* clôture[6] est une structure d'intersection $\mathcal{F}$ contenant la partie toute entière : $A \in \mathcal{F}$. L'application $\rho_{\mathcal{F}} : \wp^{A} \to \wp^{A}$ définie par $\rho_{\mathcal{F}}(X) \triangleq \bigcap\{F \in \mathcal{F} \mid X \subseteq F\}$ est un opérateur de clôture sur la famille de toutes les parties possibles $\rho_{\mathcal{F}} : (\wp^{A}, \subseteq) \xmapsto{\mathcal{H}} (\wp^{A}, \subseteq)$. Étant donné $X \in \wp^{A}$, l'élément $\rho_{\mathcal{F}}(X)$ est la clôture *de* X *dans* $\mathcal{F}$.

THEOREM 2.5.3. *Tout système de clôture équipé de l'inclusion ensembliste* $(\mathcal{F}, \subseteq)$ *est un treillis complet où la borne inférieure est l'intersection ensembliste, et la borne supérieure est la clôture de l'union ensembliste.* □

Vice versa, étant donné un opérateur de clôture $\rho : (\wp^{A}, \subseteq) \xmapsto{\mathcal{H}} (\wp^{A}, \subseteq)$, son image (qui coïncide avec l'ensemble de ses points fixes) constitue un système de clôture $\mathcal{F} \triangleq \{X \in \wp^{A} \mid \rho(X) = X\}$, dont ρ est l'opérateur de clôture associé : $\rho = \rho_{\mathcal{F}}$. Il existe donc une *bijection* entre l'ensemble des systèmes de clôture $\mathcal{F} : \wp^{(\wp^{A})}$ et l'ensemble des opérateurs de clôture $\rho : (\wp^{A}, \subseteq) \xmapsto{\mathcal{H}} (\wp^{A}, \subseteq)$.

2.5.2. Complétion par idéaux et co-finalité. La *complétion par idéaux* est une technique de construction d'un dcpo algébrique à partir d'un préordre.

THEOREM 2.5.4. *Si* $(D, \leq)$ *est un préordre, alors* $(\mathcal{I}(D, \leq), \subseteq)$ *est un dcpo algébrique dont les élément compacts coïncident avec les idéaux principaux du préordre :* $\mathcal{K}(\mathcal{I}(D, \leq), \subseteq) = \{X \in \wp^{D} \mid \exists x \in D.\ X = \downarrow_{(D, \leq)} x\}$. *De plus, si* $(D, \leq)$ *est un dcpo algébrique, alors la structure des idéaux de compacts, équipés de l'inclusion ensembliste, est isomorphe (par l'application borne supérieure) à la structure de départ :* $(\mathcal{IK}(D, \leq), \subseteq) \overset{\sqcup}{\cong} (D, \leq)$. *La fonction inverse de cet isomorphisme est la clôture compacte vers le bas.*

[5] *intersection structure* ou $\bigcap$-*structure*
[6] *closure system* ou *topped intersection structure*

DÉMONSTRATION. *cf.* [**Amadio & Curien, 1998**] *par exemple.* □

La *complétion par filtres* est la construction duale. Nous allons vérifier que les deux peuvent êtres expliquées, respectivement, par les propriétés de co-finalité et co-initialité.

Le passage aux idéaux, qui sont clos vers le bas, et l'adoption de l'ordre d'inclusion ensembliste révèlent l'intention d'utiliser les dirigés en faisant abstraction de leur dénouement vers le bas. Dans le même esprit, qui ne nous fait regarder que l'avant-garde des dirigés, nous aurions pu considérer l'ensemble des dirigés modulo l'équivalence de co-finalité. La proposition suivante prouve l'équivalence des deux approches : la construction des ensembles clos vers le bas (lower sets) correspond au passage aux parties co-finales.

PROPOSITION 2.5.5. *Soit* $(\mathsf{D}, \leq)$ *un préordre. L'ensemble des parties closes vers le bas, équipés de l'inclusion ensembliste, est isomorphe (par l'application qui associe la classe de co-finalité) à l'ensemble des parties modulo co-finalité, équipé de l'ordre de co-finalité entre classes :*

$$(\mathcal{L}(\mathsf{D}, \leq), \subseteq) \overset{[.]_\flat}{\cong} (\wp^{\mathsf{D}}_{\approx^\flat}, \leq^\flat)$$

L'application inverse est la clôture vers le bas d'un représentant arbitraire de la classe : $\downarrow : \wp^{\mathsf{D}}_{\approx^\flat} \to \mathcal{L}(\mathsf{D}, \leq)$*, définie par* $\downarrow \chi \triangleq \downarrow_{(\mathsf{D},\leq)} \mathsf{X}$ *où* $[\mathsf{X}]_\flat = \chi$.

DÉMONSTRATION. *Il nous faut démontrer, d'une part, que l'application* $[.]_\flat : \mathcal{L}(\mathsf{D}, \leq) \to \wp^{\mathsf{D}}_{\approx^\flat}$ *est injective et surjective, d'autre part, qu'elle conserve l'inclusion ensembliste et que son inverse conserve la relation de co-finalité. (injective) Soit* $[\mathsf{X}]_\flat = [\mathsf{Y}]_\flat$ *et soit* $x \in \mathsf{X}$*. Par définition de classes co-finales* $\mathsf{X} \approx^\flat \mathsf{Y}$*, donc* $\exists y \in \mathsf{Y}.\, x \leq y$*. Mais puisque* Y *est clos vers le bas,* $x \in \mathsf{Y}$*. Par symétrie nous pouvons obtenir* $\mathsf{Y} \subseteq \mathsf{X}$*, donc* $\mathsf{X} = \mathsf{Y}$*. (surjective) Soit* $\chi \in \wp^{\mathsf{D}}_{\approx^\flat}$*. Par définition* $\exists \mathsf{X} \in 2^{\mathsf{D}}.\, \chi = [\mathsf{X}]_\flat$*. Considérons la clôture vers le bas* $\mathsf{X}' = \downarrow_{(\mathsf{D},\leq)} \mathsf{X}$*. Nous voulons montrer que* $\mathsf{X}' \approx^\flat \mathsf{X}$*. D'une part,* $\mathsf{X} \leq^\flat \mathsf{X}'$ *est évident puisque* X' *contient* X*. D'autre part, soit* $x' \in \mathsf{X}'$ *; par définition de clôture vers le bas, il existe* $x \in \mathsf{X}$ *tel que* $x' \leq x$*, donc* $\mathsf{X}' \leq^\flat \mathsf{X}$*. Nous avons ainsi obtenu que* $\exists \mathsf{X}' \in \mathcal{L}(\mathsf{D}, \leq).\, [\mathsf{X}']_\flat = [\mathsf{X}]_\flat = \chi$*. (conservation de* $\subseteq$*) Soit* $\chi = [\mathsf{X}]_\flat$*,* $\chi' = [\mathsf{X}']_\flat$*. L'inclusion ensembliste* $\mathsf{X} \subseteq \mathsf{X}'$ *implique trivialement* $\mathsf{X} \leq^\flat \mathsf{X}'$ *donc* $\chi \leq^\flat \chi'$*. (conservation de* $\leq^\flat$*) La fonction inverse, qui surgit de la preuve de surjectivité, est la clôture vers le bas d'un représentant quelconque de classe. Soit* $\chi \leq^\flat \chi'$ *où* $\chi = [\mathsf{X}]_\flat$ *et* $\chi' = [\mathsf{X}']_\flat$*. Nous devons montrer que* $\mathsf{Y} = \downarrow \mathsf{X}$ *et* $\mathsf{Y}' = \downarrow \mathsf{X}'$ *sont tels que* $\mathsf{Y} \subseteq \mathsf{Y}'$*. Soit* $y \in \mathsf{Y}$*. Donc* $\exists x \in \mathsf{X}.\, y \leq x$*. Puisque* $\mathsf{X} \leq^\flat \mathsf{X}'$ *nous avons que* $\exists x' \in \mathsf{X}'.\, x \leq x'$*. La conjonction des deux conditions et la transitivité du préordre nous permettent d'affirmer que* $\exists x' \in \mathsf{X}'.\, y \leq x'$*. Enfin, puisque* Y' *est la clôture vers le bas de* X'*, nous avons* $y \in \mathsf{Y}'$, c.q.f.d. □

Une conséquence directe du résultat précédent est que les deux constructions correspondantes, celle des lower sets, et celle des parties co-finales, peuvent êtres limitées aux parties dirigés. Nous obtenons d'un côté la complétion par idéaux, de l'autre ce que nous appellerons la construction *bémol*.

Corollary 2.5.6. *D'une part, si* $(\mathsf{D},\leq)$ *un préordre, alors :*

$$(\mathcal{I}(\mathsf{D},\leq),\subseteq) \quad \overset{[\,.\,]_\flat}{\cong} \quad (\Delta(\mathsf{D},\leq)_{\approx^\flat},\leq^\flat)$$

où la fonction inverse est la clôture vers le bas $\downarrow$ *d'un représentant arbitraire de la classe. D'autre part, si* $(\mathsf{D},\leq)$ *est un dcpo, alors :*

$$(\mathcal{I}\mathcal{K}(\mathsf{D},\leq),\subseteq) \quad \overset{[\,.\,]_\flat}{\cong} \quad (\Delta\mathcal{K}(\mathsf{D},\leq)_{\approx^\flat},\leq^\flat)$$

où la fonction inverse est la clôture compacte vers le bas $\mathrel{\rlap{\downarrow}{\,\raisebox{0.3ex}{-}}}$ *d'un représentant arbitraire de la classe.* □

Definition 2.5.7. (bémol $\mathsf{D}^\flat$ et dièse $\mathsf{D}^\sharp$ d'un préordre) *Étant donné un préordre* $(\mathsf{D},\leq)$*, nous appellerons* bémol de D*, noté* $\mathsf{D}^\flat$*, l'ensemble des dirigés de* D *modulo co-finalité :* $\mathsf{D}^\flat \triangleq \Delta(\mathsf{D},\leq)_{\approx^\flat}$*. D'autre part, nous appellerons* dièse de D*, noté* $\mathsf{D}^\sharp$*, l'ensemble des co-dirigés de* D *modulo co-initialité :* $\mathsf{D}^\sharp \triangleq \nabla(\mathsf{D},\leq)_{\approx^\sharp}$*. Lorsque* $(\mathsf{D},\leq)$ *sera un dcpo, le* $\mathcal{K}$*-bémol de* D*, noté* $\mathsf{D}^\flat_{\mathcal{K}}$*, sera l'ensemble des dirigés de compacts modulo co-finalité* $\mathsf{D}^\flat_{\mathcal{K}} \triangleq \Delta\mathcal{K}(\mathsf{D},\leq)_{\approx^\flat}$*. Lorsque* $(\mathsf{D},\leq)$ *sera un co-dcpo, le* $\mathcal{K}$*-dièse de* D*, noté* $\mathsf{D}^\sharp_{\mathcal{K}}$*, sera l'ensemble des co-dirigés de co-compacts modulo co-initialité* $\mathsf{D}^\sharp_{\mathcal{K}} \triangleq \nabla\mathcal{K}(\mathsf{D},\leq)_{\approx^\sharp}$*.* □

Par le corollaire 2.5.6, d'une part le bémol d'un préordre correspond à la *complétion par idéaux*, d'autre part, le dièse correspond à la *complétion par filtres*. Ainsi, ces deux ensembles équipés respectivement de la relation de co-finalité $(\mathsf{D}^\flat,\leq^\flat)$ et de co-initialité $(\mathsf{D}^\sharp,\leq^\sharp)$, formeront toujours, le premier, un dcpo et, le second, un co-dcpo. Nous pouvons à présent réécrire le théorème 2.5.4.

Corollary 2.5.8. *Si* $(\mathsf{D},\leq)$ *est un préordre, alors* $(\mathsf{D}^\flat,\leq^\flat)$ *est un dcpo algébrique dont les éléments compacts sont les classes d'appartenance des singletons du préordre :*

$$\mathcal{K}(\mathsf{D}^\flat,\leq^\flat) \;=\; \{[\{x\}]_\flat \in \mathsf{D}^\flat \mid x \in \mathsf{D}\}$$

De plus, si $(\mathsf{D},\leq)$ *est un dcpo algébrique, alors la structure des dirigés de compacts modulo co-finalité, est isomorphe (par l'application borne supérieure) à la structure de départ :*

$$(\mathsf{D}^\flat_{\mathcal{K}},\leq^\flat) \overset{\sqcup}{\cong} (\mathsf{D},\leq)$$

La fonction inverse de tel isomorphisme est la classe de co-finalité de la clôture compacte de l'élément : $x \mapsto [\mathrel{\rlap{\downarrow}{\,\raisebox{0.3ex}{-}}} x]_\flat$*.*

> Démonstration. *Par le théorème 2.5.4 et le corollaire 2.5.6, nous avons* $\mathcal{K}(\mathsf{D}^\flat,\leq^\flat) = \{[\downarrow x]_\flat \in \mathsf{D}^\flat \mid x \in \mathsf{D}\}$*. La classe de co-finalité de* $\downarrow x$ *étant égale à celle du singleton* $\{x\}$*, nous obtenons le résultat.* □

La première partie de l'énoncé signifie que, pour tout préordre $(\mathsf{D},\leq)$ non seulement $(\mathsf{D}^\flat,\leq^\flat)$ est un dcpo, mais aussi que le plongement naturel de D dans $\mathsf{D}^\flat$, c'est-à-dire l'application $[\{.\}]_\flat$, qui à tout $x \in \mathsf{D}$ associe la classe de co-finalité du singleton $\{x\}$, définit précisément, par son image, l'ensemble des compacts dans $\mathsf{D}^\flat$.

EXAMPLE 2.5.9. (Suites finies ou infinies) Étant donné un ensemble A, l'ensemble des suites finies ou infinies $A^{\leq\omega}$ sur A peut être équipé de la relation *préfixe*, définie par $x \leq y$ si et seulement si $x = y_{|_I}$ où I est l'ensemble $\{i \in \mathbb{N}_+ \mid 1 \leq_{\mathbb{N}} i \leq_{\mathbb{N}} |x|\}$ ou bien $I = \mathbb{N}_+$. On peut facilement prouver que $(A^{\leq\omega}, \leq)$ est un dcpo algébrique dont les compacts sont les suites finies de A^*. Donc, il s'agit d'un domaine isomorphe à la construction $\mathcal{K}$-bémol $((A^{\leq\omega})^\flat_{\mathcal{K}}, \leq^\flat)$. Intuitivement, toute suite infinie $x : \mathbb{N}_+ \to A$ pourra être considérée (modulo l'isomorphisme) comme la collection (la classe) de toutes les successions croissantes (dirigés) de suites *finies* convergentes à x. □

Le résultat dual du théorème 2.5.4 est que, si $(D, \leq)$ est un préordre, alors $(D^\sharp, \leq^\sharp)$ est un co-dcpo algébrique dont les éléments co-compacts sont :

$$\mathcal{K}^{op}(D^\sharp, \leq^\sharp) \;=\; \{[\{x\}]_\sharp \in D^\sharp \mid x \in D\}$$

Et que, si $(D, \leq)$ est un co-dcpo, alors nous avons l'isomorphisme :

$$(D^\sharp_{\mathcal{K}}, \leq^\sharp) \stackrel{\sqcap}{\cong} (D, \leq)$$

2.6. Sous-structures

Dans le contexte des préordres ou des ordres partiels, pour éviter toute ambiguïté entre la notion de *restriction d'une relation* (1.1.4) et celle de *restriction algébrique* (1.4.1), nous préférerons indiquer la seconde avec le terme *sous-algèbre*. Il existe un lien entre les deux notions : restreindre la relation de préordre ou d'ordre $\leq$ à un sous-ensemble D' du domaine D, permet d'obtenir une sous-algèbre spécifique $(D', \leq')$.

2.6.1. Sous-préordres. Étant donné un préordre $(D, \leq: 1^{D\times D})$, nous appellerons *sous-préordre* toute sous-algèbre $(D', \leq') \subseteqq (D, \leq)$ qui soit, elle-même, un préordre. Pour dénoter la conjonction des deux conditions orthogonales, l'inclusion algébrique de deux structures étant des préordres, nous utiliserons la notation $(D', \leq') \stackrel{pre}{\subseteqq} (D, \leq)$. Un exemple particulier de sous-préordre est le choix d'un sous-ensemble $D' \subseteq D$ équipé de la restriction de relation $\leq$ au sous-domaine D'.

LEMMA 2.6.1. $(D', \leq_{|_{D'}}) \stackrel{pre}{\subseteqq} (D, \leq)$ *pour tout* $D' \subseteq D$.

DÉMONSTRATION. *D'une part, nous avons* $(D', \leq_{|_{D'}}: 1^{D'\times D'}) \subseteqq (D, \leq: 1^{D\times D})$, *puisque l'identité* $\varphi : D' \to D$ $\varphi(x) = x$ *est telle que* $x' \leq_{|_{D'}} y' \Rightarrow \varphi(x') \leq \varphi(y')$. *D'autre part, la relation* $\leq_{|_{D'}}$ *hérite de* $\leq$ *les propriétés réflexive et transitive.* □

On remarquera que $(D', \leq_{|_{D'}})$ est le plus grand préordre, aux sens de l'inclusion algébrique $\subseteqq$, qui peut être bâti après la sélection d'un sous-ensemble $D' \subseteq D$. Tout autre sous-préordre de $(D, \leq)$ sera forcément un sous-préordre de $(D', \leq_{|_{D'}})$.

Comme pour les *sous-monoïdes,* où la condition de sous-algèbre suffit pour obtenir les propriétés des monoïdes (associative et élément neutre, voir l'exemple 1.4.2), nous aurions pu espérer pouvoir coder les propriétés *réflexive* et *transitive* dans la seule condition de sous-algèbre. Le lemme 2.6.1 indique, d'une façon indirecte, une définition alternative de sous-préordre. En effet, on peut vérifier facilement que, étant donné un sous-ensemble $D' \subseteq D$, il existe une *unique* sous-algèbre $(D', \leq')$ de $(D, \leq)$ si nous considérons $\leq$ et $\leq'$ non pas comme des *semi-prédicats* $\leq : 1^{D\times D}$ et $\leq : 1^{D'\times D'}$, mais comme des *prédicats* $\leq : 2^{D\times D}$ et $\leq' : 2^{D'\times D'}$. Cette algèbre est précisément $(D', \leq_{|_{D'}})$ (il suffit de remarquer que la condition $(\leq') \underset{2^{D'\times D'}}{\xmapsto{\mathrm{id}}} (\leq')$ équivaut à la condition $\leq' = \leq_{|_{D'\times D'}}$). La définition que nous avons donnée est, donc, plus générale.

2.6.2. Sous-ordres partiels. Étant donné un ordre partiel $(D, \leq)$, nous appellerons *sous-ordre partiel* toute sous-algèbre $(D', \leq') \subseteq (D, \leq)$ qui soit, elle-même, un ordre partiel. D'une façon analogue aux sous-préordres, nous noterons $(D', \leq') \overset{\mathrm{po}}{\subseteq} (D, \leq)$. Tout sous-préordre d'un ordre partiel est systématiquement un sous-ordre partiel, ce qui équivaut à dire que la propriété *anti-symétrique* des ordres partiels est directement codée dans la condition de sous-algèbre.

LEMMA 2.6.2. *Soit* $(D, \leq)$ *un ordre partiel et soit* $(D', \leq') \overset{pre}{\subseteq} (D, \leq)$. *Alors :*

$$(D', \leq') \overset{po}{\subseteq} (D, \leq).$$

DÉMONSTRATION. *Il suffit de montrer que* $\leq'$ *est forcément anti-symétrique. Soit* $x \leq' y$ *et* $y \leq' z$. *Puisque* $\leq'$ *se conserve en* $\leq$, *nous avons* $x \leq' y$ *et* $y \leq' z$, *donc* $x = z$ *par anti-symétrie de* $\leq$. □

REMARK 2.6.3. L'application combinée des lemmes 2.6.1 et 2.6.2, nous permet d'affirmer que $(D', \leq_{|_{D'}}) \overset{\mathrm{po}}{\subseteq} (D, \leq)$ pour tout $D' \subseteq D$. L'ordre partiel $(D', \leq_{|_{D'}})$ n'est pas le seul sous-ordre partiel de $(D, \leq)$, mais il est, en certitude, le plus grand au sens de l'inclusion algébrique. Nous l'appellerons *restriction (maximale) de* $(D, \leq)$ *au sous-domaine* $D' \subseteq D$.

2.6.2.1. *Bornes d'un sous-ordre partiel.* Dans le passage à un sous-ordre partiel, les bornes supérieures ont tendance à croître par rapport aux bornes supérieures du contexte initial. De façon duale, les bornes inférieures ont tendance à fléchir.

LEMMA 2.6.4. *Soit* $(D', \leq') \overset{po}{\subseteq} (D, \leq)$. *Alors :*

$$\begin{array}{ll} \forall X \in \mathrm{dom}(\sup') \cap \mathrm{dom}(\sup). & \sup'(X) \geq \sup(X) \\ \forall X \in \mathrm{dom}(\inf') \cap \mathrm{dom}(\inf). & \inf'(X) \leq \inf(X) \end{array}$$

DÉMONSTRATION. *Il suffit de montrer que* $v \triangleq \sup_{(D', \leq')} X$ *appartient à* $\mathrm{ubs}_{(D,\leq)} X$. *Soit* $x \in X$. *Nous avons* $x \leq' v$, *donc* $x \leq v$ *puisque* $(D', \leq') \overset{po}{\subseteq} (D, \leq)$. □

Le lemme précédent suggère une façon de calculer les bornes supérieures ou inférieures dans un sous-ordre partiel : si une ancienne borne supérieure est un majorant dans le nouvel ordre, il s'agit forcément de la nouvelle borne supérieure.

LEMMA 2.6.5. *Soit* $(\mathsf{D}', \leq') \overset{po}{\subseteq} (\mathsf{D}, \leq)$. *Alors :*

$$\begin{array}{lccc} \forall \mathsf{X} \in \mathsf{D}' \cap \mathrm{dom}(\sup). & \mathrm{ub}'(\sup \mathsf{X}, \mathsf{X}) & \Rightarrow & \sup'(\mathsf{X}) = \sup(\mathsf{X}) \\ \forall \mathsf{X} \in \mathsf{D}' \cap \mathrm{dom}(\inf). & \mathrm{lb}'(\inf \mathsf{X}, \mathsf{X}) & \Rightarrow & \inf'(\mathsf{X}) = \inf(\mathsf{X}) \end{array}$$

DÉMONSTRATION. *Supposons, par l'absurde, qu'il existe un autre majorant* M' *de* X, *tel que* $\mathsf{M}' \leq' \mathsf{M} = \sup \mathsf{X}$. *Puisque* $(\mathsf{D}', \leq') \overset{po}{\subseteq} (\mathsf{D}, \leq)$ *on a* $\mathsf{M}' \leq \mathsf{M}$. *D'autre part, pour tout* $x \in \mathsf{D}'$, $x \leq' \mathsf{M}'$, *donc aussi* $x \leq \mathsf{M}'$. *Mais si* M' *est un majorant de* X *dans* $(\mathsf{D}, \leq)$, *alors* $\mathsf{M}' = \sup(\mathsf{X})$. □

Une restriction maximale $\Gamma' \triangleq (\mathsf{D}', \leq_{|_{\mathsf{D}'}})$ d'un ordre partiel $\Gamma \triangleq (\mathsf{D}, \leq)$ n'hérite pas forcément les propriétés de complétudes vérifiées par la structure de départ Γ. Intuitivement, en éliminant des éléments, le risque de supprimer des bornes, supérieures ou inférieures, est conséquent. En revanche, une preuve de complétude de la nouvelle structure peut exploiter celle de la structure de départ.

REMARK 2.6.6. Soit F la famille de sous-ensembles de D (sous-ensembles binaires, dirigés, co-dirigés ou parties quelconques) pour laquelle $(\mathsf{D}, \mathfrak{R})$ est complet vers le haut ($\forall \mathsf{X} \in \mathsf{F}.\ \exists \sup_\Gamma \mathsf{X}$) ou vers le bas ($\forall \mathsf{X} \in \mathsf{F}.\ \exists \inf_\Gamma \mathsf{X}$). Puisque $\mathsf{D}' \subseteq \mathsf{D}$, la famille correspondante F' dans D' sera incluse dans F : $\mathsf{F}' \subseteq \mathsf{F}$ (par exemple les dirigés de D' seront des dirigés de D). Donc, un X choisi arbitrairement dans F' appartiendra aussi à F. En appliquant alors l'hypothèse de complétude, nous aurons $\exists \sup_\Gamma \mathsf{X}$ (si complet vers le haut) ou $\exists \inf_\Gamma \mathsf{X}$ (si complet vers le bas). Dans le premier cas nous aurons un minimum parmi les majorants de X au sens de la structure $(\mathsf{D}, \mathfrak{R})$. Soit $\mathsf{S} \triangleq \sup_\Gamma \mathsf{X} = \min_\Gamma \mathrm{ubs}_\Gamma \mathsf{X}$. Puisque $\mathsf{D}' \subseteq \mathsf{D}$ nous avons $\mathrm{ubs}_{\Gamma'} \mathsf{X} \subseteq \mathrm{ubs}_\Gamma \mathsf{X}$. Minimiser sur un ensemble plus petit (moins de choix) nous donne un résultat possiblement plus grand, donc : $\mathsf{S}' \triangleq \sup_{\Gamma'} \mathsf{X} = \min_{\Gamma'} \mathrm{ubs}_{\Gamma'} \mathsf{X}$ sera tel que $\mathsf{S} \mathfrak{R} \mathsf{S}'$. Cette remarque nous amène à une condition suffisante à la complétude de Γ' vis-à-vis de F' : la condition $\mathsf{S} \in \mathsf{D}'$. En effet, dans une telle hypothèse, S serait aussi un majorant de X au sens de l'ordre $(\mathsf{D}', \mathfrak{R}_{|_{\mathsf{D}'}})$ donc, on aurait aussi $\mathsf{S}' \mathfrak{R} \mathsf{S}$ et, par anti-symétrie, $\mathsf{S}' = \mathsf{S}$. Le cas de la complétude vers le bas est symétrique. Ainsi, pour résumer, des critères suffisants pour prouver une propriété de complétude d'une restriction sont les suivants :

$$\begin{array}{lcl} \forall \mathsf{X} \in \mathsf{F}'.\ \sup_\Gamma \mathsf{X} \in \mathsf{D}' & \Rightarrow & \forall \mathsf{X} \in \mathsf{F}'.\ \exists \sup_{\Gamma'} \mathsf{X} = \sup_\Gamma \mathsf{X} \\ \forall \mathsf{X} \in \mathsf{F}'.\ \inf_\Gamma \mathsf{X} \in \mathsf{D}' & \Rightarrow & \forall \mathsf{X} \in \mathsf{F}'.\ \exists \inf_{\Gamma'} \mathsf{X} = \inf_\Gamma \mathsf{X} \end{array}$$

2.6.3. Sous-dcpo. Étant donné un dcpo $(\mathsf{D}, \leq, \perp, \sqcup)$, un *sous-dcpo* est une sous-algèbre qui est, elle aussi, un dcpo. Nous noterons une telle relation de la façon suivante :

$$(\mathsf{D}', \leq', \perp', \sqcup') \overset{\mathrm{dcpo}}{\subseteq} (\mathsf{D}, \leq, \perp, \sqcup)$$

En d'autres termes, un sous-dcpo est un choix double, celui d'un sous-ensemble $\mathsf{D}' \subseteq \mathsf{D}$ et celui d'une sous-relation $(\leq') \subseteq (\leq)$, tels que :

(1) $(\mathsf{D}', \leq', \perp', \sqcup')$ soit un ordre partiel

(2) $\perp'() = \perp()$

(3) $\forall \mathsf{X} \in \Delta^+(\mathsf{D}', \leq').\quad \sqcup'(\mathsf{X}) = \sqcup(\mathsf{X})$

Si dans un sous-ordre partiel les bornes supérieures ont tendance à croître (cf. lemme 2.6.4), la condition demandée à un sous-dcpo est précisément celle de conserver, pour les dirigés, leurs anciennes bornes supérieures. La notion duale, celle de *sous-co-dcpo*, est une sous-algèbre $(\mathsf{D}', \leq', \top', \sqcap')$ de $(\mathsf{D}, \leq, \top, \sqcap)$ qui soit, elle aussi, un co-dcpo. Un *sous-domaine de Scott* est un sous-dcpo qui soit, lui aussi, un domaine de Scott.

LEMMA 2.6.7. *Soit* $(\mathsf{D}', \leq', \perp', \sqcup') \overset{dcpo}{\subseteqq} (\mathsf{D}, \leq, \perp, \sqcup)$. *Alors, les compacts de* $(\mathsf{D}, \leq)$ *se trouvant dans* D' *sont des compacts du sous-dcpo :*

$$\mathcal{K}(\mathsf{D}, \leq) \cap \mathsf{D}' \;\subseteq\; \mathcal{K}(\mathsf{D}', \leq')$$

DÉMONSTRATION. *Soit* $\mathsf{k} \in \mathcal{K}(\mathsf{D}, \leq)$. *Soit* $\mathsf{X}' \in \Delta(\mathsf{D}', \leq')$ *choisi arbitrairement parmi les dirigés tels que* $\sqcup'(\mathsf{X}') \geq \mathsf{k}$. *L'identité étant ici un homomorphisme, elle conserve les dirigés (cf. prop. 2.3.2). Donc* $\mathsf{X}' \in \Delta(\mathsf{D}, \leq)$. *Par définition de sous-dcpo, nous avons* $\sqcup(\mathsf{X}') \geq \mathsf{k}$. *Puisque* k *est compact en* $(\mathsf{D}, \leq)$, *il existe* $x' \in \mathsf{X}'$ *tel que* $x' \geq \mathsf{k}$, c.q.f.d. □

En revanche, l'équation $\mathcal{K}(\mathsf{D}, \leq) \cap \mathsf{D}' = \mathcal{K}(\mathsf{D}', \leq')$ n'est pas vraie en général. Intuitivement, la sélection opérée sur D pour obtenir D' peut éliminer une quantité de points suffisante pour que de nouveaux compacts apparaissent.

LEMMA 2.6.8. *L'union arbitraire de cônes d'un dcpo constitue un sous-dcpo :*

$$Hp \begin{cases} (\mathsf{D}, \leq, \perp, \sqcup) \text{ dcpo} \\ \mathsf{D}' = \bigcup_{x \in \mathrm{I}} \downarrow_{(\mathsf{D}, \leq)} x \\ \leq' = \leq|_{\mathsf{D}' \times \mathsf{D}'} \end{cases} \qquad Ts \begin{cases} (\mathsf{D}', \leq', \perp', \sqcup') \overset{dcpo}{\subseteqq} (\mathsf{D}, \leq, \perp, \sqcup) \end{cases}$$

DÉMONSTRATION. *Puisque* $\leq'$ *est la restriction de* $\leq$ *à un sous-ensemble* D' *d'un ordre partiel, la structure* $(\mathsf{D}', \leq')$ *est, elle-même, un ordre partiel. Puisque* $\perp$ *appartient à tous les cônes de* D, *il sera aussi le plus petit élément de* D'. *Soit* $\mathsf{X} \in \Delta(\mathsf{D}', \leq')$. *Par définition de* D' *tout élément* $x \in \mathsf{X}$ *est majoré par un sommet* $v \in \mathrm{I}$. *Nous voulons prouver qu'il existe un sommet* $\bar{v}$ *qui majore tous les éléments du dirigé. Supposons par l'absurde qu'aucun sommet* $v \in \mathrm{I}$ *majorant un élément* $x \in \mathsf{X}$ *puisse être aussi un majorant d'un autre élément* $y \in \mathsf{X}$. *Puisque* X *est un dirigé, il existera* $z \in \mathsf{X}$ *tel que* $x \leq z$ *et* $y \leq z$. *Donc tout sommet majorant* z *(et il en existe forcément un) sera un majorant à la fois de* x *et de* y, *ce qui prouve la contradiction. Ainsi, puisque* X *est aussi un dirigé de* $(\mathsf{D}, \leq)$, *qui est un dcpo, nous avons* $\exists \sqcup \mathsf{X} = s$. *Mais* s *est le plus petit des majorants de* X, *donc il est plus petit que le sommet* $\bar{v}$ *qui borne* X *dans* D'. *Autrement dit* $s \leq \bar{v}$, *et donc* $s \in \mathsf{D}'$. □

Deuxième partie

Théorie des jeux combinatoires

CHAPITRE 3

Théorie des arbres et des termes

Ce chapitre est consacré à l'étude des arbres et des termes définis comme fonctions partielles d'un domaine structuré, l'*arborescence*, et à valeurs dans un codomaine arbitraire, un ensemble éventuellement sans structure où les éléments seront des simples symboles appelés *étiquettes*. L'arborescence, qui représente la structure essentielle de l'arbre, matérialise l'abstraction qui vise la forme de l'arbre oubliant sa décoration. L'étude débutera donc par l'analyse des arborescences, de leurs opérations et de leurs relations d'ordre, et sera à la base du reste du chapitre, ou l'on analysera les mêmes opérations et relations d'ordre étendues aux arbres et aux termes. Par la nature des relations sur les arbres de jeu, nécessaires par la suite, nous utiliserons une définition d'arbre décalée par rapport à l'habituelle (cf. par exemple [**Courcelle, 1983**]), et nous focaliserons l'attention sur les propriétés de complétude des relations d'ordre engendrées. Les algèbres des arborescences, des arbres et des termes seront décrite à l'aide du métalangage introduit dans le premier chapitre.

3.1. L'algèbre des arborescences

3.1.1. Alphabet, mots. Soit A un ensemble, que nous appelons *alphabet*, dont nous dénoterons les éléments, appelés *caractères*, par des lettres $\mathtt{a}, \mathtt{b}, \mathtt{c}$ etc. Nous supposons l'alphabet A muni d'un ordre total $\preceq$, noté aussi $\preceq_A$, appelé *ordre lexical*. Un *mot*, dénoté par les lettres u, v, w etc, est une suite finie de caractères. Comme toute suite, la *longueur* d'un mot u est noté $|u|$, et son i-ème caractère est noté u_i. L'ensemble de tous les mots possibles sur l'alphabet est l'ensemble des suites bâties sur A, qui est noté A^* ; le mot vide est, comme la suite vide, noté par ε, tandis que A^+ dénote l'ensemble $A^* \setminus \{\varepsilon\}$. Le *produit* ou *concaténation* de deux mots est la concaténation des suites (cf. section 1.1.8), noté $u.v$ ou uv, c'est-à-dire le mot obtenu en composant séquentiellement les lettres de u et après celles de v. Étant donné un mot u, le produit de k mots identiques à u est noté u^k, et nous notons $u^0 \triangleq \varepsilon$. A^* est muni de la relation d'ordre *préfixe* $\leq$ définie par :

$$u \leq v \quad \overset{\triangle}{\Longleftrightarrow} \quad \exists w \in A^*.\ \ u.w = v$$

La relation préfixe est un ordre partiel sur A^*. Si $u \leq v$ nous disons que u est un *préfixe* de v et que v est un *suffixe* de u. En particulier, si $v = u.\mathtt{a}$ avec $\mathtt{a} \in A$, nous disons que v est un suffixe *immédiat* de u et, symétriquement, que u est un préfixe immédiat de v. Si l'ordre préfixe est partiel et ne dépend pas de l'ordre primitif $\preceq_A$ défini sur l'alphabet, l'ordre *lexicographique* est, en revanche, l'extension de son homologue aux suites de caractères et, comme son homologue, il est total. L'ordre lexicographique, noté $\preceq_{A^*}$, est défini récursivement comme la plus petite relation (aux sens de l'inclusion ensembliste, dans le treillis complet $A^* \times A^*$) telle que :

$$\frac{\mathtt{a} \preceq_A \mathtt{b}}{\mathtt{a}.u \preceq_{A^*} \mathtt{b}.u} \ \text{ où } \mathtt{a},\mathtt{b} \in A,\ u \in A^* \qquad\qquad \frac{u \preceq_{A^*} v}{\mathtt{a}.u \preceq_{A^*} \mathtt{a}.v} \ \text{ où } \mathtt{a} \in A,\ u,v \in A^*$$

Dans la notation, il arrive souvent de confondre l'ordre $\preceq_A$ avec son extension aux suites de caractères $\preceq_{A^*}$, en utilisant le symbole $\preceq$ privé de l'indice.

3.1.2. Langages. Un *langage* sur l'alphabet A est un sous-ensemble quelconque de A^*. Nous noterons les langages par des lettres majuscules. Le *produit* ou *concaténation* de deux langages U et V, noté $U.V$, est le langage $\{u.v \mid (u,v) \in U \times V\}$. Par U^k nous dénoterons l'ensemble des mots obtenus par le produit de k mots de U : $U^k \triangleq \{u.v \mid (u,v) \in U^{k-1} \times U\}$, $U^0 \triangleq \{\varepsilon\}$, que nous appelerons *puissance* k de U. L'*étoile* (ou *puissances finies*) du langage U, noté U^*, est le langage $\bigcup_{k \geq 0} U^k$. Par convention, la priorité des opérateurs définis, pour des expressions dénotant des langages, est, du plus au moins prioritaire, l'étoile ou la puissance, le produit et l'union. Quand cela n'engendrera pas d'ambiguïté, nous noterons avec la lettre u le langage $\{u\}$ contenant le seul mot u. L'ensemble des préfixes $\mathrm{Pref}(U)$ et des suffixes $\mathrm{Suff}(U)$ d'un langage sont la clôture, respectivement, par préfixes et suffixes des mots de U : $\mathrm{Pref}(U) \triangleq \{v \in A^* \mid v \leq u\,,\ u \in U\}$, $\mathrm{Suff}(U) \triangleq \{v \in A^* \mid u \leq v\,,\ u \in U\}$. En d'autre termes, il s'agit respectivement des ensembles des minorants et des majorants de U vis-à-vis de la structure d'ordre partiel $(A^*, \leq)$: $\mathrm{Pref}(U) = \mathrm{lbs}_{(A^*,\leq)}U$ et $\mathrm{Suff}(U) = \mathrm{ubs}_{(A^*,\leq)}U$. Un ensemble de suffixes d'un intérêt particulier est celui des *suffixes immédiats*, ou *successeurs*, d'un mot u dans un langage U : $\mathrm{Succ}(U,u) \triangleq \{\mathtt{a} \in A \mid u.\mathtt{a} \in U\}$. L'ordre lexical $\preceq$ sur l'alphabet A, qui est un ordre total, permet d'ordonner in extenso les successeurs d'un mots dans un langage : pour toute adresse $u \in A^*$ et tout langage $U \subseteq A^*$, la *suite des suffixes immédiats* (ou *successeurs*) de u dans U, notée $\mathrm{Succ}_k^{(U,u)}$, est définie de façon inductive :

$$\begin{cases} \mathrm{Succ}_1^{(U,u)} \mathrel{\overset{\leftarrow}{=}} \min_{\preceq}\{\mathtt{a} \in A \mid u.\mathtt{a} \in U\} \\[2ex] \mathrm{Succ}_{k+1}^{(U,u)} \mathrel{\overset{\leftarrow}{=}} \min_{\preceq} \left\{ \mathtt{a} \in A \mid u.\mathtt{a} \in U \setminus \{u.\mathtt{b} \mid \mathtt{b} = \mathrm{Succ}_h^{(U,u)}\,,\ h \leq_{\mathbb{N}} k\} \right\} \end{cases}$$

Ainsi, le premier suffixe immédiat de l'adresse u dans U est $\mathrm{Succ}_1^{(U,u)}$, le second est $\mathrm{Succ}_2^{(U,u)}$, le troisième est $\mathrm{Succ}_3^{(U,u)}$, etc. Puisque l'ordre lexical est total, le procédé inductif peut s'achever, pour un certain $k \geq 1$, seulement si l'ensemble dont on calcule le minimum est vide. Dans ce cas la suite est finie et elle s'achève à l'indice $k-1$ (quand $k-1=0$ nous écrirons $\mathrm{Succ}^{(U,u)}=\varepsilon$). Sinon elle sera infinie (et l'alphabet, lui aussi, sera forcément infini).

3.1.3. Expressions régulières. Les expressions régulières, et les langages qu'elles dénotent, sont définies de façon inductive :

- 0, 1 et $\mathtt{a}$ sont des expressions régulières qui dénotent respectivement le langage vide $\emptyset$, $\{\varepsilon\}$ et $\{\mathtt{a}\}$ pour tout $\mathtt{a}\in A$.
- si $\mathbf{u}$ et $\mathbf{v}$ sont des expressions régulières dénotant respectivement les langages réguliers U et V, alors $\mathbf{u}+\mathbf{v}$, $\mathbf{u}.\mathbf{v}$, $\mathbf{u}^*$ sont des expressions régulières dénotant respectivement les langages réguliers $U\cup V$, $U.V$ et U^*.

Par convention, la priorité décroissante des opérations dans des expressions régulières est, comme celles sur les langages, l'étoile ($*$), le produit (.) et la somme ($*$). Le langage décrit par l'expression régulière est noté $\mathtt{Lang}(\mathbf{u})$. Les langages susceptibles d'être décrits par une expression régulière sont une classe plutôt restreinte, celle des langages *réguliers*, qui sont reconnus par des automates finis. Malgré leur faible envergure, les expressions régulières sont un outil que nous pourrons utiliser par la suite, spécialement pour fournir des exemples (ou contre-exemples) de langage vérifiant ou pas une propriété. Ces propos intéressent spécialement la classe des langages clos par préfixes, les *arborescences*, que nous introduisons à présent.

3.1.4. Arborescences. Pour la définition des arborescences, donc des arbres, nous utiliserons l'ensemble des entiers naturels (non nuls) comme alphabet de base. Soit $\mathbb{N}$ l'ensemble des entiers naturels, $\mathbb{N}_+$ l'ensemble des entiers naturels non nuls, $\mathbb{N}_+^*$ le langage de tous les mots possibles sur l'alphabet $\mathbb{N}_+$. Chaque $\mathtt{i}\in\mathbb{N}_+$ est identifié à la séquence $\mathtt{i}\in\mathbb{N}_+^*$ (composée du seul caractère $\mathtt{i}$). L'ordre lexical est celui habituel des entiers naturels : $\preceq\,=\,\leq_{\mathbb{N}}$. Nous dénotons par le symbole $\leq$ l'ordre préfixe sur $\mathbb{N}_+^*$.

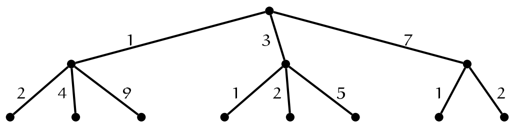
DEFINITION 3.1.1. *Une* arborescence *(ou* ensemble de positions d'un arbre) *est un langage sur les entiers naturels clos par préfixe. Nous noterons* IT *l'ensemble de toutes les possibles arborescences :*

$$\mathrm{IT}\triangleq\{P\subseteq\mathbb{N}_+^*\mid \mathrm{Pref}(P)=P\}$$

Les mots d'une arborescence sont appelés adresses*. Une collection d'arborescences* $F\in 2^{\mathrm{IT}}$ *est appelé* forêt*. Nous noterons* T *l'ensemble des arborescences finies, i.e. composées d'un nombre fini d'adresses.* □

REMARK 3.1.2. Le domaine des arborescences coïncide avec l'ensemble des parties closes vers le bas de l'ordre partiel $(\mathbb{N}_+^*,\leq)$: $\mathrm{IT}=\mathcal{L}(\mathbb{N}_+^*,\leq)$.

Pour les mots d'une arborescence, nous préférons l'appellation *adresse*, au lieu de *position*, traditionnellement utilisée, pour éviter de possibles ambiguïtés avec les *positions d'un jeu* que nous définirons par la suite. La notion d'arborescence est préliminaire à celle d'arbre : elle forge, pourrait-on dire, structure des ramifications et leurs priorités, définies implicitement par l'ordre lexical des entiers naturels. Ce dernier sera noté de la façon habituelle, par le symbole $\leq_{\mathbb{N}}$. La propriété caractéristique des arborescences, la clôture par préfixe, permet de les représenter graphiquement d'une façon avantageuse et bien connue. Le dessin suivant correspond à l'arborescence $\{1, 3, 7, 1.2, 1.4, 1.9, 3.1, 3.2, 3.5, 7.1, 7.2\}$.

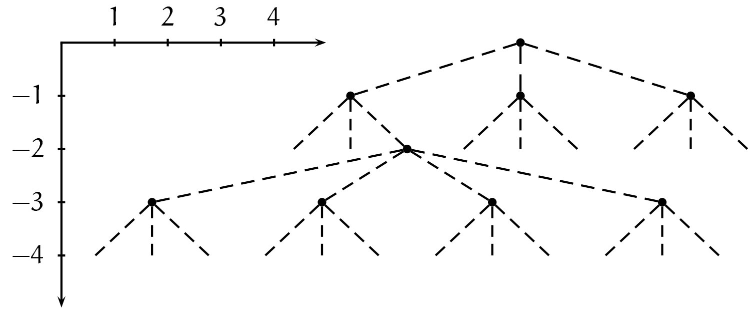


Cet exemple attire l'attention sur une différence importante entre la définition 3.1.1 et celle qui est utilisée habituellement (par exemple dans [**Gécseg & Steinby, 1997**] et [**Courcelle, 1983**]), où l'on ajoute la condition :

$$\mathtt{u.i} \in \mathsf{P} \;\Rightarrow\; \mathtt{u.j} \in \mathsf{P} \;\; \forall \mathtt{j}.\, 1 \leq_{\mathbb{N}} \mathtt{j} \leq_{\mathbb{N}} \mathtt{i}$$

La définition 3.1.1 est donc moins restrictive : elle ne prétend pas une numérotation continue des suffixes immédiats d'une adresse. Par exemple, si $\mathsf{P} = \{\varepsilon, 1, 2, 2.1, 2.2\}$ est une arborescence pour les deux définitions, le langage $\mathsf{P}' = \{\varepsilon, 1, 3, 3.1, 3.4\}$ ne le serait pas au sens habituel. La raison pour laquelle nous préférons la 3.1.1 est qu'elle permettra aux *élagages d'un arbre*, notion qui sera définie par la suite, d'appartenir eux aussi à la famille des arbres. Bien entendu, il existe une transformation triviale qui associe à toute arborescence dans le nouveau sens, une arborescence dans l'ancien. Il suffit de renommer les adresses dans la progression des entiers naturels. De cette façon, l'espace P' serait reconverti en P qui, oubliant les adresses réelles, ne représenterait que la structure essentielle, autrement dit, les ramifications et leur ordre. Nous dirons que P est la *forme normale* de l'arborescence P'.

La définition d'une arborescence comme un langage clos par préfixes, sans autres conditions, autorise deux infinités orthogonales : d'une part chaque mot du langage peut avoir une infinité de suffixes immédiats, d'autre part, les mots du langage peuvent être d'une longueur illimitée (tout en étant, chacun, de longueur finie). Intuitivement, cela revient à dire que la largeur des arborescences peut être infinie, tout comme leur profondeur.



Par exemple, la plus grande arborescence (au sens de l'inclusion ensembliste), le langage *exhaustif* $\mathbb{N}_+^*$, est infini en profondeur et infini en largeur à tous ses niveaux.

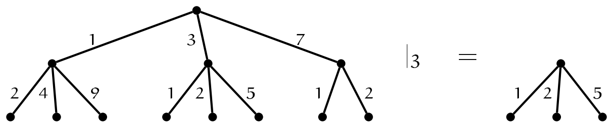
3.1.5. Racines, feuilles, noeuds d'arborescences. Un certaine terminologie qui concerne les arbres, comme les notions de *feuille*, *noeud*, *branche* et *élagage*, appartient d'abord au cadre théorique des arborescences et s'exporte ensuite aux arbres. Puisqu'il s'agit d'un langage clos par préfixe, le mot vide ε appartient par définition à toute arborescence non vide, et sera appelé *racine*. Toutes les adresses maximales pour l'ordre préfixe dans une arborescence P seront appelées *feuilles*. Toute adresse $u \in P$ non maximale pour l'ordre préfixe sera appelée *noeud*. La cardinalité des successeurs de l'adresse u dans le langage P, c'est-à-dire $\mathrm{Succ}(P, u)$, est appelée *degré (de branchement)* de P à l'adresse u, noté $\mathrm{degré}(P, u)$. La cardinalité de $\mathrm{Succ}(P, u)$ peut être finie ou infinie. Ainsi, la fonction de branchement est de type : $\mathrm{degré} : \mathrm{IT} \times \mathbb{N}_+^* \to \mathbb{N}_\infty$, où le codomaine $\mathbb{N}_\infty$ est l'ensemble des entiers naturels étendus par le symbole $+\infty$, plus grand que tout autre élément. Nous disons qu'une arborescence est à *branchement fini*, quand à aucune adresse le degré de branchement est infini : $\forall u \in P.\ \mathrm{degré}(u) \in \mathbb{N}$. Munis de la fonction degré, nous pouvons définir les ensembles des feuilles et des noeuds d'une arborescence d'une façon plus formelle :

$$
\begin{array}{lcl}
\mathrm{Feuilles}(P) & \triangleq & \{u \in P \mid \mathrm{degré}(P, u) = 0\} \\
\mathrm{Noeuds}(P) & \triangleq & \{u \in P \mid \mathrm{degré}(P, u) > 0\}
\end{array}
$$

Il existe une terminologie spécifique aux arborescences (et aux arbres) qui évoque des notions déjà définies pour les langages. Par exemple, le mot *chemin* est synonyme d'adresse. La *profondeur* d'un chemin n'est autre que la longueur de l'adresse. Le mot *ancêtres* est synonyme de préfixe, tout comme *descendant* qui est synonyme de suffixe. Aussi, le *père* d'une adresse est son (unique) préfixe immédiat, et le *fils* d'une adresse est un de ses suffixes immédiats. Dans ce nouveau vocabulaire, par exemple, nous pouvons dire que le degré est le nombre de fils d'une adresse. En revanche, une nouvelle notion, propre aux arborescences, est celle de *hauteur*. La hauteur d'une arborescence P est la longueur maximale des adresses de P : $\mathrm{hauteur}(P) \stackrel{\leftarrow}{=} max_{\leq_{\mathbb{N}}}\{|u| \mid u \in P\}$. Si l'arborescence est infinie, nous disons que la hauteur est, elle aussi, *infinie* ou *illimitée*. Une autre notion propre aux arborescences est celle de *frontière*. La frontière d'une arborescence est l'ensemble des adresses maximales pour l'ordre préfixe, autrement dit, l'ensemble de ses feuilles : $\mathrm{Feuilles}(P) = \{u \in P \mid \nexists v \in P \text{ t.q. } u \leq v\}$. On dira alors qu'un langage L est une frontière s'il existe une arborescence P telle que $L = \mathrm{Feuilles}(P)$. Nous noterons $\mathrm{Front} : 2^{\mathbb{N}_+^*}$ le prédicat correspondant sur l'ensemble des langages. En d'autres termes, un langage L est une frontière si, par rapport à l'ordre préfixe, l'entropie est maximale : un élément de L n'est en relation avec aucun autre élément de L (à part lui-même) : $(L \times L) \cap \leq = \{(x, x) \mid x \in L\}$.

3.1.6. Branches d'arborescences. Les arborescences ont un caractère récursif que les informaticiens connaissent bien : on peut construire une arborescence en fournissant d'abord une racine et, ensuite, les arborescences, les *branches*, qui doivent être, pour ainsi dire, greffées sur la racine.

DEFINITION 3.1.3. *Étant donné une arborescence* $P \in \mathrm{IT}$ *et une adresse* $u \in \mathbb{N}_+^*$, *la* branche de t à l'adresse u, *est le langage* $P|_u \triangleq \{v \in \mathbb{N}_+^* \mid u.v \in P\}$. *Si* $|u| = 1$ *on dit que* $P|_u$ *est une branche* immédiate *ou une branche* fille *de* P. *La relation* branche $\preceq_{\mathrm{IT}} \subseteq (\mathbb{N}_+^* \times \mathrm{IT})$ *est définie par la condition* $L \preceq_{\mathrm{IT}} P \stackrel{\triangle}{\Longleftrightarrow} \exists u \in P.\ L = P|_u$.
$\square$



La première question qui se pose est celle de cerner le codomaine de l'opération branche ainsi que le domaine de la relation homonyme.

PROPOSITION 3.1.4. *Toute branche est, elle aussi, une arborescence.*

DÉMONSTRATION. *Il faut prouver que le langage* $\mathsf{P}|_u$ *est clos par préfixe. S'il est vide, il est forcément clos :* $\mathrm{Pref}(\emptyset) = \emptyset$*. Sinon, soit* $v \in \mathsf{P}|_u$ *et soit* $w \leq v$*. Par définition de branche* $u.v \in \mathsf{P}$*, et par définition de préfixe* $u.w \leq u.v$*. L'hypothèse* $\mathrm{Pref}(\mathsf{P}) = \mathsf{P}$ *implique* $u.w \in \mathsf{P}$*, donc* $w \in \mathsf{P}|_u$. □

La proposition précédente précise le type de l'opération *branche* : $(.)|_{(.)} : \mathrm{IT} \times \mathbb{N}_+^* \to \mathrm{IT}$, et précise aussi le périmètre de relation homonyme, qui est donc une relation entre arborescences : $\preceq_{\mathrm{IT}} \subseteq (\mathrm{IT} \times \mathrm{IT})$. On peut remarquer que, par définition, si $u \in \mathsf{P}$, alors $\mathsf{P}|_u$ contient au moins le mot vide ε. Tout comme les successeurs d'un mot dans un langage, les branches filles peuvent être rangées dans l'ordre lexical des entiers naturels. La suite des arborescences filles de l'arborescence P, $\mathrm{Filles}(\mathsf{P}) : \mathrm{IT}^*$, est la suite définie par $\mathrm{Filles}_i(\mathsf{P}) \doteq \mathsf{P}|_{\mathrm{Succ}_i^{(\mathsf{P},\varepsilon)}}$. Toute arborescence pourra s'exprimer en termes de ses branches filles :

$$\mathsf{P} = \{\varepsilon\} \cup \bigcup_{i \in \mathrm{Succ}(\mathsf{P},\varepsilon)} i.\mathrm{Filles}_i(\mathsf{P})$$

Il est intéressant de connaître les propriétés que la structure $(\mathrm{IT}, \preceq_{\mathrm{IT}})$ vérifie.

PROPOSITION 3.1.5. *Le couple* $(\mathrm{IT}, \preceq_{\mathrm{IT}})$ *est un préordre.*

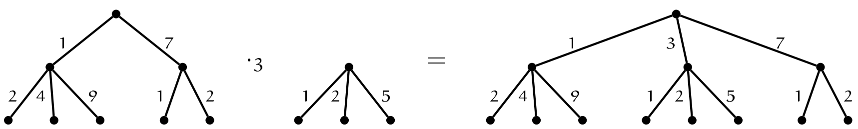
DÉMONSTRATION. *Une arborescence non vide coïncide avec sa propre branche à l'adresse* ε*, donc* $\preceq_{\mathrm{IT}}$ *est réflexive. Soient* $\mathsf{P}_1 \preceq_{\mathrm{IT}} \mathsf{P}_2 \preceq_{\mathrm{IT}} \mathsf{P}_3$*. Nous avons* $z \in \mathsf{P}_1 \Rightarrow \exists u.\, u.z \in \mathsf{P}_2 \Rightarrow \exists v.\, v.u.z \in \mathsf{P}_3 \Rightarrow \exists x = (v.u).\, x.z \in \mathsf{P}_3 \Rightarrow \mathsf{P}_1 \preceq_{\mathrm{IT}} \mathsf{P}_3$*, donc* $\preceq_{\mathrm{IT}}$ *est transitive.* □

La relation branche n'est pas plus qu'un préordre. Prenons, par exemple, l'arborescence définie par l'expression régulière $\mathsf{P} = (1.2)^*$. La branche à l'adresse 1 est l'arborescence $\mathsf{P}' = 2.(1.2)^*$. Donc P est lui-même une branche de P', celle à l'adresse 2. Nous avons donc, $\mathsf{P} \preceq_{\mathrm{IT}} \mathsf{P}'$ et $\mathsf{P}' \preceq_{\mathrm{IT}} \mathsf{P}$ mais $\mathsf{P} \neq \mathsf{P}'$. Donc la relation branche n'est pas anti-symétrique, autrement dit, le préordre $(\mathrm{IT}, \preceq_{\mathrm{IT}})$ n'est pas un ordre partiel. Il est simple de constater que la composante anti-réflexive, notée $\prec_{\mathrm{IT}}$, est une relation bien fondée sur le domaine des arborescences finies T.

3.1.7. Greffes d'arborescences. Si l'opération branche $(.)|_{(.)}$ permet d'extraire une branche d'une arborescence, l'opération suivante permet, au contraire, d'en greffer.

DEFINITION 3.1.6. *Étant donné deux arborescences* $P_1, P_2 \in \mathrm{IT}$ *et une adresse* $u \in \mathbb{N}_+^*$, *la* greffe de P_2 sur P_1 à l'adresse u *est le langage* $P_1 \cdot_u P_2 \triangleq (P_1 \setminus \mathrm{Suff}(u)) \cup u.P_2$. □

L'opération de greffe $(.) \cdot_u (.) : \mathrm{IT} \times \mathrm{IT} \to 2^{\mathbb{N}_+^*}$ ne restitue pas toujours une arborescence, mais on vérifie facilement que $P_1 \cdot_u P_2 \in \mathrm{IT}$ quand il existe au moins un préfixe immédiat de u dans P_1. Donc, en particulier, quand $u \in P_1$. Le rapport entre la notion de branche et celle de greffe est que $X = P|_u$ est une solution (l'unique) de l'équation $P = P \cdot_u X$. Autrement dit, si on extrait une branche pour la greffer à nouveau au même endroit, nous reconstituons l'arborescence d'origine.



L'opération de greffe peut être généralisée, d'abord, à une frontière : $(.) \cdot_U (.) : \mathrm{IT} \times \mathrm{IT} \to 2^{\mathbb{N}_+^*}$, si $\mathrm{Front}(U)$, de la façon classique, par l'union ensembliste des résultats de la greffe sur toutes les adresses $u \in U$:

$$P_1 \cdot_U P_2 \triangleq \bigcup_{u \in U} P_1 \cdot_u P_2$$

Elle restituera une arborescence lorsque $(\mathrm{Pref}(U) \setminus U) \subseteq P_1$. Ensuite, au lieu de greffer toujours la même branche P_2, nous pouvons l'étendre en recueillant les greffons dans une famille $F : \mathbb{N}_+^* \to \mathrm{IT}$:

$$P \cdot_U F \triangleq \bigcup_{u \in U} P \cdot_u F(u)$$

L'opération sera donc du type d'ordre supérieur $(.) \cdot_U (.) : \mathrm{IT} \times (\mathbb{N}_+^* \to \mathrm{IT}) \to \wp^{\mathbb{N}_+^*}$.

3.1.8. Arborescences rationnelles. La notion de branche permet de caractériser les arborescences qui, n'étant pas finies, ont néanmoins une structure finie.

DEFINITION 3.1.7. *Nous disons qu'une arborescence est* rationnelle *si elle contient un nombre fini de branches distinctes. Nous noterons* RT *l'ensemble des arborescences rationnelles.* □

PROPOSITION 3.1.8. *Une arborescence est rationnelle si et seulement si elle est un langage régulier sur l'alphabet* $\mathbb{N}_+$.

DÉMONSTRATION. *cf.* **[Gécseg & Steinby, 1997]** *ou* **[Perrin & Pin, 2001]**, *par exemple.* □

Ainsi, toute arborescence rationnelle pourra être décrite par une expression régulière. Nous verrons par la suite que cette équivalence ne s'applique pas aux arbres dans les deux sens : un arbre rationnel aura, certes, une arborescence rationnelle mais un arbre constitué d'une arborescence rationnelle ne sera pas, en revanche, forcément rationnel.

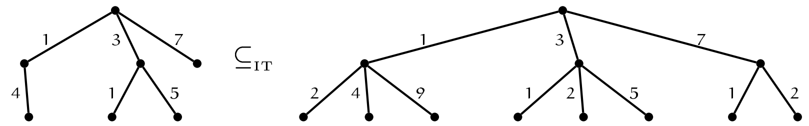
3.1.9. Élagages d'arborescences. Nous introduisons à présent une notion, l'*élagage,* qui est cruciale dans la théorie des jeux combinatoires, précisément dans les méthodes de résolution d'un jeu, comme l'algorithme Alpha-Bêta dont l'analyse sera faite dans le chapitre 6. Quand nous nous poserons la question de savoir qui est-ce qui gagne un jeu, nous aurons, comme il arrive souvent en informatique, l'ambition de répondre à la question le plus rapidement possible. Une optimisation du calcul aura lieu et elle se fera par des élagages corrects (sans perte d'information) de l'arbre de jeu. La représentation des arborescences comme langages clos par préfixes permet de définir l'élagage d'une façon bien avantageuse : une arborescence est un élagage d'une autre si, tout simplement, elle y est incluse.

Definition 3.1.9. *Étant donné une arborescence* $\mathsf{P} \in \mathrm{IT}$ *, un* élagage de P *est un langage* $\mathsf{P}' \subseteq \mathsf{P}$ *tel que* $\mathsf{P}' \in \mathrm{IT}$. *La relation d'élagage* $\subseteq_{\mathrm{IT}}$ *est la restriction de l'inclusion ensembliste* $\subseteq$ *au carré des arborescences* $\subseteq_{\mathrm{IT}} \triangleq \subseteq_{|\mathrm{IT}} = \subseteq \cap(\mathrm{IT} \times \mathrm{IT})$. □

Étant donné $\mathsf{P}' \subseteq \mathsf{P}$, nous pouvons affirmer qu'il existe P'' tel que $\mathsf{P}' = \mathsf{P} \setminus \mathsf{P}''$. Le mot *élagage* retrouve ici son sens habituel : il s'agit d'un choix d'un ensemble de branches, la forêt P'', à éliminer. Quand il s'agit d'élaguer une arborescence à une adresse donnée, de multiples possibilités se presentent. Nous allons baptiser, parmi les choix possibles, des façons spéciales comme celles où l'on élimine toutes les branches immédiates existantes, ou aucune de ces branches, ou toutes sauf une etc.

Definition 3.1.10. (Types d'élagages) *Considérons une adresse d'une arborescence :* $\mathsf{u} \in \mathsf{P}$. *Nous disons qu'un élagage* $\mathsf{P}' \subseteq_{\mathrm{IT}} \mathsf{P}$ *est* complet à l'adresse u *s'il contient* tous *les suffixes immédiats (fils) de* P *:* $\mathrm{Succ}(\mathsf{P}', \mathsf{u}) = \mathrm{Succ}(\mathsf{P}, \mathsf{u})$. *Nous disons qu'il est* déterministe à l'adresse u *s'il contient* un et un seul *suffixe immédiat ou* u *est une feuille de* P *:* $\mathrm{degré}(\mathsf{P}', \mathsf{u}) = 1 \vee \mathsf{u} \in \mathrm{Feuilles}(\mathsf{P})$. *Nous disons qu'il est* radical à l'adresse u *si* $\mathsf{u} \in \mathrm{Feuilles}(\mathsf{P}')$. *Nous disons qu'il est* partiel à l'adresse u *s'il n'est pas complet ni radical à la même adresse.* □

Les types d'élagage définis ne sont pas disjoints. En effet, par définition, un élagage P' de P est à la fois *complet*, *radical* et *déterministe* aux feuilles de P qui sont aussi contenues dans P', i.e. aux adresses de l'intersection $\mathrm{Feuilles}(\mathsf{P}') \cap \mathrm{Feuilles}(\mathsf{P})$. Ainsi, pour qu'un élagage soit partiel à une certaine adresse u, il est nécessaire que u soit un noeud de l'arbre élagué. La distinction des types d'élagages porte donc sur les noeuds de P. Reprenons notre premier exemple d'arborescence et considérons l'élagage suivant :



Nous avons ici tous les types d'élagages définis. À l'adresse ε l'élagage est complet, à l'adresse 1 il est déterministe, à l'adresse 3 il est partiel et à l'adresse 7 il est radical. Intuitivement, on peut considérer l'élagage comme une visite de l'arborescence qui ignore (élague) certains arcs et, par conséquent, toutes les branches qui en découlent. Derrière cette intuition se cache, à l'abri de toute ambiguïté sur la notion de *visite*, une définition mathématique précise. On peut considérer la relation d'élagage $\subseteq_{\mathrm{IT}}$ comme la plus grande relation (au sens de l'inclusion ensembliste sur l'ensemble des parties $2^{\mathrm{IT}\times\mathrm{IT}}$) telle que $\mathsf{P}_1 \subseteq_{\mathrm{IT}} \mathsf{P}_2$ implique :

$$(\varepsilon \in \mathsf{P}_1 \Rightarrow \varepsilon \in \mathsf{P}_2) \quad \wedge \quad \forall \mathtt{i} \in \mathbb{N}.\ \mathsf{P}_1|_{\mathtt{i}} \subseteq_{\mathrm{IT}} \mathsf{P}_2|_{\mathtt{i}}$$

En d'autres termes, une arborescence P_1 est un élagage de P_2 si le rapport d'inclusion apparaît à la racine et si, récursivement, toutes les branches immédiates de la première sont des élagages des branches correspondantes de la seconde. On peut montrer qu'il s'agit d'une définition de point fixe maximum d'une fonction croissante dans l'ensemble de parties $2^{\mathbb{N}^*_+\times\mathbb{N}^*_+}$ qui, lui, est un treillis complet d'ensembles. L'équivalence des deux définitions d'élagage se démontre trivialement. En ce qui concerne la structure $(\mathrm{IT}, \subseteq_{\mathrm{IT}})$, la relation d'élagage étant la restriction de l'inclusion ensembliste $\subseteq$ sur le carré $\mathrm{IT} \times \mathrm{IT}$, nous pouvons nous attendre à d'excellentes propriétés de complétude.

PROPOSITION 3.1.11. *La structure* $(\mathrm{IT}, \subseteq_{\mathrm{IT}})$ *est un treillis complet d'ensembles, avec un minimum* $\perp \triangleq \emptyset$ *et un maximum* $\top \triangleq \mathbb{N}^*_+$. *Il s'agit aussi d'un dcpo algébrique où les compacts sont les arborescences finies.*

DÉMONSTRATION. *La structure* $(\mathrm{IT}, \subseteq_{\mathrm{IT}})$ *est celle des ensembles clos vers le bas* $(\mathcal{L}(\mathbb{N}^*_+, \leq), \subseteq)$. *Il s'agit donc d'un treillis complet d'ensembles (cf. section 2.5.1.3). Il s'agit aussi d'un dcpo algébrique dont les compacts sont les arborescences finiment engendrées (exemple 2.4.5) :*

$$\mathcal{K}(\mathrm{IT}, \subseteq_{\mathrm{IT}}) = \{\mathsf{P} \in \mathrm{IT} \mid \mathsf{P} = \downarrow \mathsf{X},\ \mathsf{X} \in \wp_{\mathrm{F}}^{(\mathbb{N}^*_+)}\}$$

Une arborescence compacte aura donc un nombre fini de chemins, autrement dit, elle sera finie. Le langage vide $\perp \triangleq \emptyset$ *est contenu dans toute arborescence, le langage exhaustif* $\top \triangleq \mathbb{N}^*_+$ *contient toute arborescence, et les deux sont clos par préfixe, donc* $\perp, \top \in \mathrm{IT}$ *sont les éléments, respectivement, minimum et maximum de l'ordre.* □

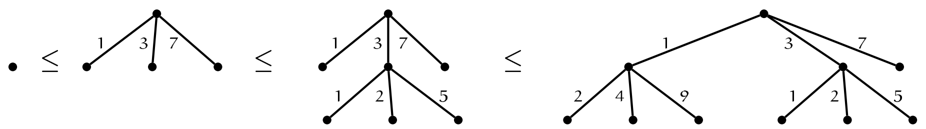
3.1.10. Préfixes d'arborescences. L'élagage d'une arborescence peut être méthodique : nous pouvons exiger que toutes les adresses soient visitées et éventuellement élaguées avec la même discipline. Un exemple de discipline, qui nous amène à la notion de *préfixe*, est celle qui oblige à élaguer complètement un noeud, ou bien, à le laisser intact. En visitant l'arborescence nous choisissons, à chaque adresse, de visiter tous les suffixes immédiats (élagage *complet*) de l'adresse, ou bien, sans demi-mesures, d'arrêter la visite en les ébranchant tous (élagage *radical*).

DEFINITION 3.1.12. *Un* préfixe P' *d'une arborescence* P *est un élagage partout* complet *ou* radical *(ou, autrement dit, jamais* partiel*). La relation préfixe, notée par le symbole* $\leq_{\text{IT}}$*, est une relation sur le carré* $\text{IT} \times \text{IT}$*, définie par la condition :*

$$\mathsf{P}' \leq_{\text{IT}} \mathsf{P} \overset{\triangle}{\iff} \forall u \in \mathsf{P}'.\ u \in \mathsf{P} \ \wedge\ (\text{Succ}(\mathsf{P}', u) = \text{Succ}(\mathsf{P}, u) \ \vee\ u \in \text{Feuilles}(\mathsf{P}'))$$

□

Soit $\mathsf{P}' \leq_{\text{IT}} \mathsf{P}$. La première condition de la définition ($\forall u \in \mathsf{P}'.\ u \in \mathsf{P}$) implique que P' soit un sous-ensemble de P. Puisque la relation est restreinte, par définition, au carré $\text{IT} \times \text{IT}$, elle ne concerne que les arborescences, donc $\mathsf{P}' \subseteq_{\text{IT}} \mathsf{P}$. Autrement dit, la relation préfixe est incluse dans celle d'élagage. La deuxième condition implique que l'élagage soit complet ($\text{Succ}(\mathsf{P}', u) = \text{Succ}(\mathsf{P}, u)$) ou radical ($u \in \text{Feuilles}(\mathsf{P}')$). L'élagage représenté à la page 81, n'est pas un préfixe de son arborescence, puisqu'il contient des élagages partiels. Les élagages suivants sont, en revanche, des préfixes de l'arborescence concernée :



La notion de préfixe est partagée par les mots et les arborescences, ce qui entraîne une ambiguïté potentielle qui sera toujours résolue par le contexte. D'autre part, nous utilisons le même symbole $\leq$ pour l'ordre préfixe entre mots et celui entre arborescences. Il ne s'agit pas d'un hasard, au contraire les deux notions sont bien apparentées : il se trouve que l'ordre préfixe sur les arborescences peut être défini dans le même style utilisé pour l'ordre préfixe sur les mots : $\mathsf{P}' \leq_{\text{IT}} \mathsf{P}$ si et seulement si il existe une frontière U (finie ou pas) et une famille d'arborescences $\mathsf{P}'' : \mathbb{N}_+^* \to \text{IT}$ telles que $\mathsf{P} = \mathsf{P}'.{}_{\mathsf{U}}\mathsf{P}''$. L'opération $(.) \cdot_{\mathsf{U}} (.) : \text{IT} \times (\mathbb{N}_+^* \to \text{IT}) \to 2^{\mathbb{N}_+^*}$ joue donc le rôle du produit (composition) entre mots. Elle greffe sur l'arborescence P' les branches de la famille P'' aux adresses définies par la frontière U. Dans ce sens, il s'agit d'une extension naturelle de la concaténation aux arborescences.

La notion de préfixe, comme celle d'élagage, peut s'interpréter comme une visite de l'arborescence qui, dans le cas du préfixe, ignore toute branche ou les visite toutes. À nouveau, nous pouvons préciser mathématiquement ce qu'est la visite avec une telle discipline. Sans faire appel à la définition d'élagage, on peut redéfinir la relation préfixe $\leq_{\text{IT}}$ directement comme la plus grande relation (au sens de l'inclusion ensembliste sur l'ensemble des parties $2^{\text{IT} \times \text{IT}}$) telle que $\mathsf{P}_1 \leq_{\text{IT}} \mathsf{P}_2$ implique :

$$\mathsf{P}_1 = \emptyset \quad \vee \quad (\mathsf{P}_1 = \{\varepsilon\} \ \wedge\ \varepsilon \in \mathsf{P}_2) \quad \vee$$
$$(\text{Succ}(\mathsf{P}_1, \varepsilon) = \text{Succ}(\mathsf{P}_2, \varepsilon) \ \wedge\ \forall i \in \text{Succ}(\mathsf{P}_1, \varepsilon).\ \mathsf{P}_1|_i \leq_{\text{IT}} \mathsf{P}_2|_i)$$

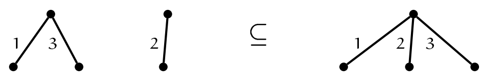
En d'autres termes, une arborescence P_1 est un préfixe de P_2 si elle est vide, ou bien, si elle ne contient pas plus que la racine de P_2, ou bien si, récursivement, toutes les branches immédiates de P_2 sont préfixées par les branches correspondantes de P_1. Il s'agit, là aussi, d'une définition de point fixe maximum d'une fonction croissante dans l'ensemble de parties $2^{\mathbb{N}_+^* \times \mathbb{N}_+^*}$. L'équivalence des deux définitions se démontre facilement.

À présent, nous allons étudier, comme nous l'avons fait pour l'élagage, les propriétés d'ordre partiel de la relation préfixe.

PROPOSITION 3.1.13. *La structure* $(\mathrm{IT}, \leq_{\mathrm{IT}})$ *est un sous-ordre partiel de* $(\mathrm{IT}, \subseteq_{\mathrm{IT}})$*, dont le minimum est encore* $\bot \triangleq \emptyset$.

> DÉMONSTRATION. *Une arborescence* P *est un préfixe d'elle-même, puisqu'il s'agit d'un élagage complet à toutes ses adresses. Donc la relation* $\leq_{\mathrm{IT}}$ *est* réflexive. *Soient* P_1, P_2, P_3 *trois arborescences telles que* $P_1 \leq_{\mathrm{IT}} P_2$ *et* $P_2 \leq_{\mathrm{IT}} P_3$. *Nous voulons prouver que* $P_1 \leq_{\mathrm{IT}} P_3$. *Considérons une certaine adresse* $u \in P_1$. *Par inclusion de* $\leq_{\mathrm{IT}}$ *dans* $\subseteq_{\mathrm{IT}}$ *nous avons* $P_1 \subseteq_{\mathrm{IT}} P_3$ *donc* $u \in P_3$. *Puisque* $P_1 \leq_{\mathrm{IT}} P_2$, *nous avons aussi* $u \in P_2$. *En appliquant les conditions de la définition, nous obtenons, dans un premier temps :*
>
> – $((1)\ \mathrm{Succ}(P_1, u) = \mathrm{Succ}(P_2, u) \quad \vee \quad (2)\ u \in \mathrm{Feuilles}(P_2)) \quad \wedge$
> – $((3)\ \mathrm{Succ}(P_2, u) = \mathrm{Succ}(P_3, u) \quad \vee \quad (4)\ u \in \mathrm{Feuilles}(P_3))$
>
> *Ensuite, en distribuant le* $\wedge$ *sur le* $\vee$, *nous obtenons les cas suivants :*
>
> – $(1) \wedge (3) \;\Rightarrow\; \mathrm{Succ}(P_1, u) = \mathrm{Succ}(P_3, u)$
> – $(1) \wedge (4) \;\Rightarrow\; u \in \mathrm{Feuilles}(P_3)$
> – $(2) \wedge (3) \;\Rightarrow\; u \in \mathrm{Feuilles}(P_2) \wedge \mathrm{Succ}(P_2, u) = \mathrm{Succ}(P_3, u)$
> $\qquad\qquad\;\Rightarrow\; u \in \mathrm{Feuilles}(P_3)$
> – $(2) \wedge (4) \;\Rightarrow\; u \in \mathrm{Feuilles}(P_3)$
>
> *Donc, dans tous les cas, la condition* $\mathrm{Succ}(P_1, u) = \mathrm{Succ}(P_3, u) \vee u \in \mathrm{Feuilles}(P_3)$ *est vérifiée par le* u *choisi arbitrairement dans* P_1. *Autrement dit,* $P_1 \leq_{\mathrm{IT}} P_3$, *donc la relation* $\leq_{\mathrm{IT}}$ *est* transitive. *Par le lemme 2.6.2,* $(\mathrm{IT}, \leq_{\mathrm{IT}}) \overset{pre}{\subseteq} (\mathrm{IT}, \subseteq_{\mathrm{IT}})$ *implique, puisque le second est un ordre partiel, que* $(\mathrm{IT}, \leq_{\mathrm{IT}}) \overset{po}{\subseteq} (\mathrm{IT}, \subseteq_{\mathrm{IT}})$. *Enfin, le langage vide* $\bot \triangleq \emptyset$, *qui est contenu dans toute arborescence, est un préfixe de toute arborescence (la quantification universelle est toujours vraie sur l'ensemble vide) ; il s'agit donc du minimum de la structure d'ordre partiel.* □

Le langage exhaustif $\top \triangleq \mathbb{N}_+^*$, qui contient toutes les autres arborescences, ne pourra pas être un majorant, au sens de l'ordre préfixe, des arborescences à branchement fini contenant plus qu'une adresse (plus que la racine ε). Donc l'ordre $(\mathrm{IT}, \leq_{\mathrm{IT}})$, à la différence de $(\mathrm{IT}, \subseteq_{\mathrm{IT}})$, ne possède pas un élément maximal (s'il existait, il serait forcément plus grand que $\top$, ce qui est absurde). Pire, nous ne pouvons pas attendre de la relation préfixe des propriétés de complétude aussi bonnes que celles de la structure $(\mathrm{IT}, \subseteq_{\mathrm{IT}})$. Prenons, par exemple, les trois arborescences suivantes :



Certes, les deux premières trouvent dans leur union, la troisième, leur plus petit majorant (au sens de l'élagage), mais il ne sont pas pour autant des préfixes de leur union. Aucun majorant, au sens de l'ordre $\leq_{\mathrm{IT}}$, peut exister : ils ne pourront pas être des élagages complets à la racine, les deux à la fois, d'une arborescence qui doit forcément les contenir, donc contenir leur union. Par conséquent, le couple $(\mathrm{IT}, \leq_{\mathrm{IT}})$ n'est pas un dirigé, condition nécessaire aux treillis (voir schéma de la page 53). En revanche, il se trouve que $(\mathrm{IT}, \leq_{\mathrm{IT}})$ est complet par dirigés et co-dirigés.

PROPOSITION 3.1.14. *La structure* $(\mathrm{IT}, \leq_{\mathrm{IT}})$ *est un bi-dcpo d'ensembles.*

DÉMONSTRATION. *La preuve se décompose en deux parties : celle où l'on montre que* $(\mathrm{IT}, \leq_{\mathrm{IT}})$ *est un dcpo d'ensembles et celle où l'on montre qu'il s'agit aussi d'un co-dcpo d'ensembles.*

(1) $(\mathrm{IT}, \leq_{\mathrm{IT}})$ *est un dcpo d'ensembles.*
Soit $D \in \Delta(\mathrm{IT}, \leq_{\mathrm{IT}})$. *Puisque* $(\mathrm{IT}, \subseteq_{\mathrm{IT}})$ *est un treillis d'ensembles (cf. prop. 3.1.11),* $U \triangleq \sup_{\subseteq_{\mathrm{IT}}} X = \bigcup_{P \in D} P$. *Par le lemme 2.6.5, il suffira de prouver que* U *est aussi un majorant au sens de l'ordre préfixe. Supposons alors, par l'absurde, qu'il existe un élément* $Y \in D$ *tel que* $Y \not\leq_{\mathrm{IT}} U$. *Par définition* U *contient* Y, *autrement dit,* Y *est un élagage de* U. *Supposer qu'il ne soit pas un préfixe équivaut à dire que* Y *est partiel (ni complet, ni radical) à une quelconque adresse* $u \in Y$. *Puisque* $u \notin \mathrm{Feuilles}(Y)$ *il faudra alors que* $\mathrm{Succ}(Y,u) \subset \mathrm{Succ}(U,u)$. *Soient* $j_1,...,j_n$ *les suffixes entiers de l'adresse* u *que* U *contient et qui, en revanche, sont absents de* Y : $\{j_1, \dots j_n\} \triangleq \mathrm{Succ}(U,u) \setminus \mathrm{Succ}(Y,u)$. *Puisque* U *est l'union du dirigé, il doit nécessairement exister des éléments* $X_1, \dots, X_n \in D$ *contenant respectivement les adresses* $u.j_1,...,u.j_n$. *Par définition de dirigé,* $\exists Z \in D$ *tel que* $Y \leq_{\mathrm{IT}} Z$ *et, pour tout* $i = 1 \dots n$, $X_i \leq_{\mathrm{IT}} Z$. *Par construction,* Z *est complet à l'adresse* u *par rapport à* U, *puisqu'il contient tous les suffixes immédiats de* u. *Donc* Y *est, à la même adresse, forcément partiel vis-à-vis de* Z, *en contradiction avec la condition* $Y \leq_{\mathrm{IT}} Z$.

(2) $(\mathrm{IT}, \leq_{\mathrm{IT}})$ *est un co-dcpo d'ensembles.*
Soit $D \in \nabla(\mathrm{IT}, \leq_{\mathrm{IT}})$. *Puisque* $(\mathrm{IT}, \subseteq_{\mathrm{IT}})$ *est un treillis d'ensembles (cf. prop. 3.1.11),* $I \triangleq \inf_{\subseteq_{\mathrm{IT}}} X = \bigcap_{P \in D} P$. *Toujours par le lemme 2.6.5, il suffira de prouver que* I *est aussi un minorant au sens de l'ordre préfixe. Supposons par l'absurde qu'il existe un élément* $Y \in D$ *tel que* $I \not\leq_{\mathrm{IT}} Y$. *Par définition d'intersection* Y *contient* I, *autrement dit,* I *est un élagage de* Y. *Supposer qu'il ne soit pas un préfixe, équivaut à dire que* I *est partiel (ni complet, ni radical) à une certaine adresse* $u \in I$. *Puisque* $u \notin \mathrm{Feuilles}(I)$ *il faudra alors que* $\mathrm{Succ}(I,u) \subset \mathrm{Succ}(Y,u)$. *Soient* $j_1,...,j_n$ *les suffixes entiers de l'adresse* u *que* Y *contient et qui, en revanche, sont absents de* I : $\{j_1, \dots j_n\} \triangleq \mathrm{Succ}(Y,u) \setminus \mathrm{Succ}(I,u)$. *Puisque* I *est l'intersection du co-dirigé, il doit nécessairement exister des éléments* $X_1, \dots, X_n \in D$ *ne contenant pas, respectivement, les adresses* $u.j_1,...,u.j_n$. *Par définition de co-dirigé,* $\exists Z \in D$ *tel que* $Z \leq_{\mathrm{IT}} Y$ *et, pour tout* $i = 1 \dots n$, $Z \leq_{\mathrm{IT}} X_i$. *Par construction* Z *ne peut être complet à l'adresse* u *vis-à-vis de* Y. *Donc il sera radical. Donc* $I \subseteq_{\mathrm{IT}} Z$ *est, à la même adresse, forcément radical vis-à-vis de* Y, *en contradiction avec ce qui avait été supposé précédemment.*

□

Un type particulier de préfixe correspond à élaguer radicalement une arborescence à niveaux différents de profondeur.

DEFINITION 3.1.15. (PRÉFIXES RÉGULIERS) *Étant donné* $P \in \mathrm{IT}$*, nous définissons l'*élagage de P au niveau $k \in \mathbb{N}$*, comme le sous-ensemble* $P/^k = \{u \in P \mid |u| \leq_{\mathbb{N}} k\}$ *et nous disons que* P' *est un* préfixe régulier *de* P*. La relation* préfixe régulier *entre arborescences, notée par le symbole* $\sqsubseteq_{\mathrm{IT}}$*, est définie par la condition* $P' \sqsubseteq_{\mathrm{IT}} P \overset{\triangle}{\iff} \exists k \in \mathbb{N}.\ P' = P/^k$. □

Il est évident que la relation $\sqsubseteq_{\mathrm{IT}}$ est incluse dans l'ordre préfixe $\leq_{\mathrm{IT}}$: l'élagage au niveau k d'une arborescence est, par définition, un élagage radical aux adresses de longueur égale à k et complet aux adresses de longueur inférieure à k. Il s'agit donc d'un préfixe. La succession $\left(P/^k\right)_{k\in\mathbb{N}}$ est une chaîne ascendante pour l'ordre préfixe $\leq_{\mathrm{IT}}$. La structure $(\mathrm{IT}, \leq_{\mathrm{IT}})$ étant un dcpo d'ensembles, la limite supérieure de la succession au sens de l'ordre $\leq_{\mathrm{IT}}$ est l'union de tous ses éléments, c'est-à-dire P. En revanche, la structure $(\mathrm{IT}, \sqsubseteq_{\mathrm{IT}})$ n'est pas un ordre partiel, ni un préordre, la propriété réflexive étant impossible pour les arborescences infinies à branchement fini (tout préfixe régulier d'une telle arborescence serait forcément fini).

3.1.11. Arborescences à branchement fini. La classe des arborescences à branchement fini est l'ensemble $\mathrm{BT} \triangleq \{P \in \mathrm{IT} \mid \forall u \in P.\ \text{degré}(P, u) \in \mathbb{N}\}$. Pour de telles arborescences, une des deux causes orthogonales d'infinitude est éliminée : seule la hauteur de l'arborescence, i.e. la longueur de ses mots, demeure sans contrainte et peut donc impliquer la non finitude de l'arborescence. Restreindre l'objet du discours à ce sous-ensemble du domaine IT aura un effet sensible sur les propriétés de complétude de la relation d'élagage. Considérons la succession d'arborescences $f : \mathbb{N} \to \mathrm{BT}$ telle que $f(x) \triangleq \{\varepsilon\} \cup \{i \in \mathbb{N} \mid i \leq_{\mathbb{N}} x\}$. Il s'agit d'une succession croissante, donc un dirigé, au sens de l'ordre d'élagage. Mais la borne supérieure du dirigé, qui est l'union des éléments du dirigé, n'est pas une arborescence à branchement fini. Ainsi, la structure $(\mathrm{BT}, \subseteq_{\mathrm{BT}})$ où $\subseteq_{\mathrm{BT}} \triangleq \subseteq_{|\mathrm{BT}\times\mathrm{BT}}$, n'est pas un dcpo. En revanche, tout comme $(\mathrm{BT}, \leq_{\mathrm{BT}})$ où $\leq_{\mathrm{BT}} \triangleq \leq_{|\mathrm{BT}\times\mathrm{BT}}$, elle conserve la complétude vers le bas.

PROPOSITION 3.1.16. *L'intersection arbitraire d'ensembles est close dans* BT.

> DÉMONSTRATION. *Soit* $I = \bigcap_{P\in F} P$ *où* F *est une famille arbitraire d'arborescences à branchement fini* $F \subseteq 2^{\mathrm{BT}}$*. Supposons, par l'absurde que* $I \notin \mathrm{BT}$*. Il existe alors une adresse* $u \in I$ *telle que* $\text{degré}(I, u) = +\infty$*. Mais* I *est contenue dans toute autre arborescence de* F *donc, pour tout* $P \in F$*,* $\text{degré}(P, u) = +\infty$*, en contradiction avec l'hypothèse* $F \subseteq \wp^{\mathrm{BT}}$. □

COROLLARY 3.1.17. *La structure des arborescences à branchement fini avec l'ordre d'élagage* $(\mathrm{BT}, \subseteq_{\mathrm{BT}})$ *et celle avec l'ordre préfixe* $(\mathrm{BT}, \leq_{\mathrm{BT}})$*, sont des co-dcpo d'ensembles.*

DÉMONSTRATION. *La remarque 2.6.3 nous assure qu'il s'agit d'ordres partiels. D'autre part, la 2.6.6 nous donne un critère suffisant pour prouver l'énoncé : montrer que la borne inférieure d'un co-dirigé au sens, respectivement, de* $(\mathrm{IT}, \subseteq_{\mathrm{IT}})$ *et de* $(\mathrm{IT}, \leq_{\mathrm{IT}})$*, appartient à* BT*. Dans les deux cas, la borne inférieure coïncide avec l'intersection du dirigé (cf. prop. 3.1.11 et Prop. 3.1.14) qui appartient, par la proposition précédente, au domaine* BT. □

On peut remarquer que le contre-exemple prouvant la non clôture de l'union dans BT, la suite $f(x) \triangleq \{\varepsilon\} \cup \{i \in \mathbb{N} \mid i \leq_{\mathbb{N}} x\}$, est, certes, un dirigé de $(\mathrm{BT}, \subseteq_{\mathrm{BT}})$ mais, en revanche, n'est pas un dirigé de $(\mathrm{BT}, \leq_{\mathrm{BT}})$. Un dirigé au sens de l'ordre préfixe admet toujours une borne supérieure dans BT.

PROPOSITION 3.1.18. *La structure* $(\mathrm{BT}, \leq_{\mathrm{BT}})$ *est un dcpo d'ensembles et un dcpo algébrique dont les compacts sont les arborescences finies.*

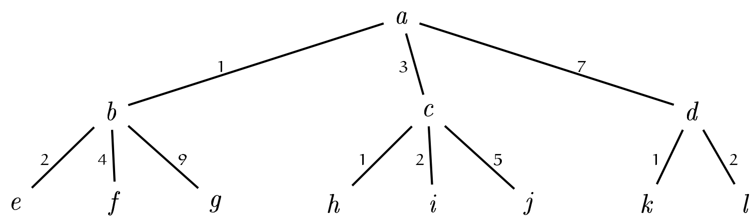
DÉMONSTRATION. *Pour montrer que* $(\mathrm{BT}, \leq_{\mathrm{BT}})$ *est un dcpo d'ensembles, il suffit de montrer que l'union d'un dirigé est à branchement fini. Soit* $D \in \Delta(\mathrm{BT}, \leq_{\mathrm{BT}})$ *et* $U = \bigcup_{P \in D} P$*. Supposons, par l'absurde que* $U \notin \mathrm{BT}$*. Il existe alors une adresse* $u \in U$ *telle que* $\text{degré}(U, u) = +\infty$*. La proposition 3.1.14 nous assure que tout élément du dirigé est un préfixe de l'union. Donc, à l'adresse* u *aucun ne sera partiel : il seront ou bien radicaux ou bien complets. Ils ne pourront pas être tous radicaux, sinon* u *serait une feuille de l'union. Alors, il en existera au moins un complet, en contradiction avec l'hypothèse que* D *soit un dirigé d'arborescences à branchement fini. Puisque* $(\mathrm{BT}, \leq_{\mathrm{BT}}) \overset{dcpo}{\subseteqq} (\mathrm{IT}, \subseteq_{\mathrm{IT}})$*, les arborescences finies sont compactes en* $(\mathrm{BT}, \leq_{\mathrm{BT}})$ *(cf. lemme 2.6.7). D'autre part, aucune arborescence infinie* P *ne peut être compacte dans* $(\mathrm{BT}, \leq_{\mathrm{BT}})$ *: le dirigé de ses préfixes réguliers serait un exemple de dirigés* D *tel que* $P = \sup_{\leq_{\mathrm{BT}}} D$*, et il est impossible qu'il existe* $X \in D$ *tel que* $P \leq_{\mathrm{BT}} X$*. Donc* $\mathcal{K}(\mathrm{BT}, \leq_{\mathrm{BT}}) = \mathrm{T}$*. De plus, toute arborescence à branchement fini est l'union de ses préfixes finis, donc le dcpo est algébrique.* □

En particulier, résumant les résultats des propositions précédentes, nous pouvons aussi affirmer que la structure $(\mathrm{BT}, \leq_{\mathrm{BT}})$ est un *bi-dcpo d'ensembles*.

3.2. L'algèbre des arbres

Nous traiterons par la suite ce qui, dans la théorie des graphes, est appelé arbre *orienté*, sans pour autant dériver la définition de celle d'un graphe.

3.2.1. Arbres. Les arbres peuvent être considérés comme des arborescences décorées, à toute adresse, par des étiquettes d'une nature quelconque. Il s'agit d'une extension du concept d'arborescence qui nous permettra de définir des notions et des relations analogues et de réutiliser les résultats prouvés pour le domaine des arborescences.

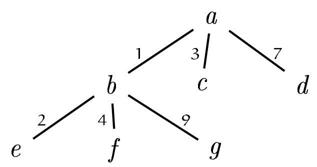


DEFINITION 3.2.1. *Étant donné un ensemble de symboles* F*, un* arbre étiqueté *dans* F *est une fonction partielle* $\mathsf{t} : \mathbb{N}_+^* \multimap\!\!\to \mathsf{F}$ *telle que* $\mathrm{dom}(\mathsf{t}) \triangleq \mathsf{P}$ *est une arborescence. Nous disons que* P *est l'*arborescence de l'arbre t*. L'ensemble des arbres sur un ensemble d'étiquettes* F *est noté* $\mathrm{IT}(\mathsf{F})$*, de l'anglais* Infinite Trees on F :

$$\mathrm{IT}(\mathsf{F}) \triangleq \{\mathsf{t} : \mathbb{N}_+^* \multimap\!\!\to \mathsf{F} \mid \mathrm{dom}(\mathsf{t}) \in \mathrm{IT}\}$$

Nous noterons $\mathrm{T}(\mathsf{F})$ *l'ensemble des arbres finis.* □

Les arbres finis $\mathsf{t} \in \mathrm{T}(\mathsf{F})$ auront forcément une arborescence finie $\mathsf{P} \triangleq \mathrm{dom}(\mathsf{t}) \in \mathrm{T}$. Dans ce cas nous pourrons représenter l'arbre en entier par un ensemble fini d'associations, tel que $\{(\varepsilon, \mathsf{a}), (1, \mathsf{b}), (3, \mathsf{c}), (7, \mathsf{d}), (1.2, \mathsf{e}), (1.4, \mathsf{f}), (1.9, \mathsf{g})\}$. Dans l'exemple suivant nous considérons $\mathsf{P} = \{\varepsilon, 1, 3, 7, 1.2, 1.4, 1.9\}$ et $\mathsf{F} = \{\mathsf{a}, \mathsf{b}, \mathsf{c}, \mathsf{d}, \mathsf{e}, \mathsf{f}, \mathsf{g}\}$.



Le choix d'une numérotation non strictement continue des adresses dans les arborescences (Def. 3.1.1) affecte le sens de la notation *syntaxique* des arbres, telle que $\mathsf{f}(\mathsf{g}(\mathsf{a}, \mathsf{b}), \mathsf{c})$. Si dans l'ancien sens, une telle écriture correspond exactement à l'arbre $\mathsf{t} = \{(\varepsilon, \mathsf{f}), (1, \mathsf{g}), (1.1, \mathsf{a}), (1.2, \mathsf{b}), (2, \mathsf{c})\}$, à présent elle peut dénoter aussi bien l'arbre t (qui est aussi un arbre dans le sens actuel), que la classe des arbres dont la structure est reconductible à t. En d'autres termes, dans une telle écriture nous retrouvons l'abstraction qui oublie les adresses spécifiques pour ne décrire que les ramifications et leur ordre.

3.2.2. Racines, feuilles, noeuds d'arbres. Les définitions de *racine*, *feuille*, *noeud*, etc. s'étendent d'une façon très naturelle des arborescences aux arbres. Un couple (u, e) d'un arbre t est appelé *racine*, *feuille* ou *noeud* si, respectivement, u est une racine, une feuille ou un noeud pour l'arborescence $\mathrm{dom}(\mathsf{t})$. Donc, les ensembles des feuilles et des noeuds *d'un arbre* t sont la restriction de la relation t au sous-domaine, respectivement, des adresses feuilles et des adresses noeuds : $\mathrm{Feuilles}(\mathsf{t}) = \{(\mathsf{u}, \mathsf{e}) \in \mathsf{t} \mid \mathsf{u} \in \mathrm{Feuilles}(\mathrm{dom}\,\mathsf{t})\}$, $\mathrm{Noeuds}(\mathsf{t}) = \{(\mathsf{u}, \mathsf{e}) \in \mathsf{t} \mid \mathsf{u} \in \mathrm{Noeuds}(\mathrm{dom}\,\mathsf{t})\}$. Pour tout $\mathsf{u} \in \mathbb{N}_+^*$, l'*ensemble des suffixes immédiats,* ou *successeurs,* de u dans t est l'ensemble $\mathrm{Succ}(\mathsf{t}, \mathsf{u}) \triangleq \{(\mathsf{i}, \mathsf{e}) \in \mathbb{N}_+ \times \mathsf{F} \mid (\mathsf{u}.\mathsf{i}, \mathsf{e}) \in \mathsf{t},\ \mathsf{i} \in \mathrm{Succ}(\mathrm{dom}\,\mathsf{t}, \mathsf{u})\}$. De la même façon, la *suite des suffixes immédiats,* ou *successeurs,* de u dans t est une suite de couples (i, e) où i est un des successeurs de u dans l'arborescence $\mathrm{dom}(\mathsf{t})$ et e est l'étiquette associée dans t à ce successeur : $\mathrm{Succ}_{\mathsf{k}}^{(\mathsf{t},\mathsf{u})} \triangleq (\mathsf{i}_{\mathsf{k}}, \mathsf{e}_{\mathsf{k}})$

où $i_k = \mathrm{Succ}_k^{(\mathrm{dom}\,t,u)}$ et $e_k = t(u.i_k)$. Le *degré de branchement* de l'arbre t à l'adresse u, noté degré(t,u), est le degré de branchement de l'arborescence dom(t) à la même adresse : degré$(t,u) \triangleq$ degré(dom t,u). La fonction de branchement est donc de type : degré : $\mathrm{IT}(F) \times \mathbb{N}_+^* \rightarrow \mathbb{N}_\infty$. Un arbre est à *branchement fini*, si son arborescence est à branchement fini. La *hauteur* d'un arbre est celle de son arborescence : hauteur$(t) \triangleq$ hauteur(dom t). Nous dirons qu'une relation $\mathfrak{R} \subseteq \mathbb{N}_+^* \times F$ est une *frontière* (*d'arbre*) s'il existe un arbre t tel que $\mathfrak{R} =$ Feuilles(t). Si $\mathfrak{R}$ est une frontière d'arbre, il est simple de vérifier que dom$(\mathfrak{R})$ est une frontière d'arborescence. Nous noterons Front : $2^{\mathbb{N}_+^* \times F}$ le prédicat correspondant sur le produit cartésien $\mathbb{N}_+^* \times F$.

3.2.3. Translation d'un arbre. La concaténation entre mots d'un langage dans A^* peut s'étendre aux relations $\mathfrak{R}$ de type $A^* \times B$ en considérant de concaténer un mot $u \in A^*$ avec la première composante de tous les couples $(v,b) \in \mathfrak{R}$:

$$u.\mathfrak{R} \triangleq \{(u.v, b) \mid (v,b) \in \mathfrak{R}\}$$

En général, une telle opération, qu'on appellera *translation* de $\mathfrak{R}$ à l'adresse u, renvoit un résultat encore de type $A^* \times B$. Dans le cas des arbres, qui sont des fonctions partielles $t : \mathbb{N}_+^* \rightharpoonup F \subseteq \mathbb{N}_+^* \times F$, la translation prendra en argument un mot $u \in \mathbb{N}_+^*$ et un arbre t mais ne restituera pas forcément un arbre (sauf dans le cas de la translation nulle $u = \varepsilon$). Cependant, les formules du genre :

$$\{(\varepsilon, e)\} \cup i.t$$

restitueront toujours un arbre, pour toute étiquette $e \in F$, pour tout entier non nul $i \in \mathbb{N}_+$, et pour tout arbre $t \in \mathrm{IT}(F)$.

Nous allons, à présent, étendre les notions de *branche*, *élagage* et *préfixe* au cadre des arbres, en examinant, comme nous l'avons fait précédemment pour les arborescences, les propriétés des relations engendrées. L'enchaînement des définitions et des résultats concernant les arbres suivra donc le même cours que celui de la section précédente.

3.2.4. Branches d'arbres. L'idée de *branche* d'une arborescence se retrouve avec les arbres, sous le nom traditionnellement utilisé de *sous-arbre*. Elle permet de caractériser, d'une façon à la fois simple et rigoureuse, l'ensemble des arbres appelés *rationnel*, qui ne sont pas finis mais qui ont, en revanche, une structure cyclique donc une représentation finie.

DEFINITION 3.2.2. *Étant donné un arbre* $t \in \mathrm{IT}(F)$ *et une adresse* $u \in \mathbb{N}_+^*$, *la* branche *(ou* sous-arbre*)* de t à l'adresse u, noté $t|_u$, *est la relation* $t|_u \subseteq \mathbb{N}_+^* \times F$, *définie par :*

$$t|_u \triangleq \{(v,e) \mid (u.v,e) \in t\}$$

Si $|u| = 1$ *on dit que* $t|_u$ *est une branche* immédiate *ou un* fils *de* t. *La relation* branche *(entre arbres)* $\preceq_{\mathrm{IT}(F)} \subseteq \big((\mathbb{N}_+^* \times F) \times \mathrm{IT}(F)\big)$ *est définie par la condition* $t' \preceq_{\mathrm{IT}(F)} t \overset{\triangle}{\iff} \exists u \in \mathbb{N}_+^*.\ t' = t|_u$. □

Par définition, pour tout $\nu \in \mathrm{dom}(t|_u)$, nous avons $t|_u(\nu) = t(u.\nu)$. Puisque $t : \mathbb{N}_+^* \rightharpoonup F$ est une fonction partielle, le sous-arbre, lui aussi, sera une fonction partielle : $t|_u : \mathbb{N}_+^* \rightharpoonup F$. De plus, le rapport entre les deux notions de branche, celle sur les arborescences et celle sur les arbres, est évident : le domaine d'un sous-arbre est une branche du domaine de l'arbre : $\mathrm{dom}(t|_u) = \mathrm{dom}(t)|_u$. Donc, étant la branche d'une autre arborescence, par la Prop. 3.1.4, le domaine du sous-arbre est, lui aussi, une arborescence. En conclusion, l'appellation est bien méritée : un sous-arbre est, lui aussi, un arbre. Et pour autant, la relation branche est limitée au carré des arbres : $\preceq_{\mathrm{IT(F)}} \subseteq (\mathrm{IT}(F) \times \mathrm{IT}(F))$. Aussi, l'opération *branche* (d'un arbre) est de type : $(.)|_{(.)} : \mathrm{IT}(F) \times \mathbb{N}_+^* \to \mathrm{IT}(F)$. Pour illustrer la définition, le sous-arbre de $f(g(a,b),c)$ à l'adresse 1 est l'arbre $g(a,b)$. Un second exemple est que la branche u d'une translation d'un arbre t à l'adresse u est égale à l'arbre t lui-même : $(u.t)|_u = t$.

Les fils d'un arbre sont donc des arbres. Comme les branches filles, nous pouvons les ranger dans l'ordre des entiers naturels. La suite des fils d'un arbre, $\mathrm{Fils}(t) : \mathrm{IT}(F)^*$, est la suite définie par $\mathrm{Fils}_i(t) \stackrel{\leftarrow}{=} t|_{\mathrm{Succ}_i^{(\mathrm{dom}\, t, \varepsilon)}}$. Tout arbre peut donc s'exprimer en termes de ses fils :

$$t = \{(\varepsilon, e)\} \cup \bigcup_{i \in \mathrm{Succ}(\mathrm{dom}\, t, \varepsilon)} i.\mathrm{Fils}_i(t)$$

La relation branche, définie cette fois sur $\mathrm{IT}(F)$, inaugure la série des relations homologues. Le rapport entre ces relations et leurs correspondantes des arborescences, définies auparavant, sera toujours le même : deux arbres dans la relation notée $\mathfrak{R}_{\mathrm{IT}}$ auront leurs domaines dans l'homologue, notée $\mathfrak{R}_{\mathrm{IT(F)}}$. C'est bien le cas de la relation *branche* pour laquelle nous avons, par définition :

$$t_1 \preceq_{\mathrm{IT(F)}} t_2 \;\Rightarrow\; \mathrm{dom}(t_1) \preceq_{\mathrm{IT}} \mathrm{dom}(t_2)$$

PROPOSITION 3.2.3. *Le couple* $(\mathrm{IT}(F), \preceq_{\mathrm{IT(F)}})$ *est un préordre.*

> DÉMONSTRATION. *Un arbre coïncide avec sa propre branche à l'adresse* ε, *donc* $\preceq_{\mathrm{IT(F)}}$ *est réflexive. Soient* $t_1 \preceq_{\mathrm{IT(F)}} t_2 \preceq_{\mathrm{IT(F)}} t_3$. *En particulier nous avons* $\mathrm{dom}(t_1) \preceq_{\mathrm{IT}} \mathrm{dom}(t_2) \preceq_{\mathrm{IT}} \mathrm{dom}(t_3)$. *La relation analogue sur les arborescences est un préordre (cf. prop. 3.1.5), ce qui implique* $\mathrm{dom}(t_1) \preceq_{\mathrm{IT}} \mathrm{dom}(t_3)$ *à une certaine adresse* u. *Soient* $u_2 \in \mathrm{dom}(t_3)$ *l'adresse de la branche* $\mathrm{dom}(t_2)$ *dans l'arborescence* $\mathrm{dom}(t_3)$ *et* $u_1 \in \mathrm{dom}(t_2)$ *l'adresse de* $\mathrm{dom}(t_1)$ *dans* $\mathrm{dom}(t_2)$. *Nous avons* $u = u_2.u_1$. *Il reste à prouver que* $\forall \nu \in \mathrm{dom}(t_1).\ t_1(\nu) = t_3(u.\nu)$. *Soit* $\nu \in \mathrm{dom}(t_1)$. *Par hypothèse* $t_1(\nu) = t_2(u_1.\nu) = t_3(u_2.u_1.\nu)$ *donc* $t_1(\nu) = t_3(u.\nu)$, *autrement dit* $\preceq_{\mathrm{IT(F)}}$ *est transitive.* □

Comme son homologue, la relation branche ne peut pas être plus qu'un préordre. Il est évident que la structure $(\mathrm{IT}, \preceq_{\mathrm{IT}})$ est isomorphe à $(\mathrm{IT}(F), \preceq_{\mathrm{IT(F)}})$ quand l'on choisit une seule façon de décorer les noeuds des arbres, autrement dit, quand F est un singleton : $F = \{e\}$. Un contre-exemple prouvant, dans $(\mathrm{IT}, \preceq_{\mathrm{IT}})$, l'absence d'une propriété sera donc aussi bien probant pour la structure $(\mathrm{IT}(F), \preceq_{\mathrm{IT(F)}})$. Il est donc impossible que $(\mathrm{IT}(F), \preceq_{\mathrm{IT(F)}})$ puisse vérifier la propriété anti-symétrique et, pour autant, qu'il puisse être un ordre partiel. En revanche, la composante anti-réflexive de cette relation, notée $\prec_{IT(F)}$, est, comme son homologue, une relation bien fondée sur le domaine des arbres finis $T(F)$.

3.2.5. Greffes d'arbres. Comme pour les arborescences, nous avons une opération duale à celle d'extraction de branche, qui assemble un arbre avec une branche.

Definition 3.2.4. *Étant donné deux arbres* $t_1, t_2 \in IT(F)$ *et une adresse* $u \in \mathbb{N}_+^*$*, la* greffe de t_2 sur t_1 à l'adresse u *est la relation :*

$$t_1 \cdot_u t_2 \triangleq \{(v, e) \mid v \in \mathrm{dom}\, t_1 \cdot_u \mathrm{dom}\, t_2,\ \text{si}\, u \leq v\, \text{alors}\, e = t_2(u.v)\, \text{sinon}\, e = t_1(u)\}$$

□

Comme son analogue, l'opération de greffe, ici de type $(.) \cdot_u (.) : IT(F) \times IT(F) \to \wp^{\mathbb{N}_+^* \times F}$, ne restitue pas toujours un arbre, mais on vérifie facilement que $t_1 \cdot_u t_2 \in IT(F)$ quand $\mathrm{dom}\, t_1 \cdot_u \mathrm{dom}\, t_2 \in IT$, c'est-à-dire quand il existe au moins un préfixe immédiat de u dans $\mathrm{dom}\, t_1$. Ici aussi, l'arbre $X = t|_u$ est la solution unique de l'équation $t = t \cdot_u X$. L'opération de greffe d'arbres peut être, elle aussi, d'abord généralisée à une frontière d'arborescence : $(.) \cdot_U (.) : IT(F) \times IT(F) \to 2^{\mathbb{N}_+^* \times F}$, si $U \in \wp^{\mathbb{N}_+^*}$ et $\mathrm{Front}(U)$, par l'union ensembliste des résultats de la greffe sur toutes les adresses $u \in U$:

$$t_1 \cdot_U t_2 \triangleq \bigcup_{u \in U} t_1 \cdot_u t_2$$

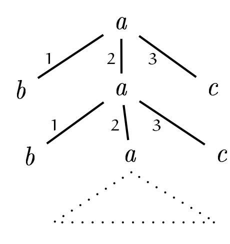
Ensuite, pour ne pas greffer constamment la branche t_2, nous pouvons l'étendre en recueillant les greffons dans une famille d'arbres $G : \mathbb{N}_+^* \to IT(F)$:

$$t \cdot_U F \triangleq \bigcup_{u \in U} t \cdot_u G(u)$$

L'opération sera donc de type $(.) \cdot_U (.) : IT(F) \times (\mathbb{N}_+^* \to IT(F)) \to \wp^{\mathbb{N}_+^* \times F}$.

3.2.6. Arbres rationnels. La définition des arbres rationnels est calquée sur celle des arborescences rationnelles, où la notion de branche est, bien entendu, celle des arbres.

Definition 3.2.5. *Un arbre est* rationnel *si il contient un nombre fini de branches (d'arbres) distinctes. Nous noterons* RT(F)*, de l'anglais* Rational Trees on F*, l'ensemble des arbres rationnels sur l'ensemble d'étiquettes* F. □



Dans cet exemple, il existe seulement trois branches distinctes de l'arbre : la branche à l'adresse ε, celle à l'adresse 1 et celle à l'adresse 3. La branche à l'adresse 2 coïncide avec l'arbre lui-même, c'est-à-dire avec la branche à l'adresse ε. Le rapport entre les deux notions de rationnalité est établi par le résultat suivant.

PROPOSITION 3.2.6. *Si un arbre* t *est rationnel alors son arborescence* $\mathrm{dom}(t)$ *est rationnelle.*

DÉMONSTRATION. *Si l'arbre a un nombre fini de branches, alors il existe une frontière* U *finie et une famille de greffons* G *telles que* $t = t \cdot_U G$. *Donc* $\mathrm{dom}(t) = \mathrm{dom}\,(t \cdot_U G) = \mathrm{dom}\left(\bigcup_{u \in U} t \cdot_u G(u)\right) = \bigcup_{u \in U} \mathrm{dom}\,(t \cdot_u G(u)) = \bigcup_{u \in U} \mathrm{dom}(t) \cdot_u \mathrm{dom}\,(G(u)) = \mathrm{dom}(t) \cdot_U G'$ *où* $G : \mathbb{N}_+^* \to \mathrm{IT}$, $G'(u) = \mathrm{dom}\,(G(u))$. *Il existe une famille d'arborescences qui peut être greffée à un nombre fini d'emplacements de* $\mathrm{dom}(t)$, *donc l'énoncé.* $\square$

Ainsi, l'arborescence d'un arbre rationnel est un langage régulier sur l'alphabet $\mathbb{N}_+$ (cf. prop. 3.1.8). La réciproque n'est pas vraie : l'arborescence peut être un langage régulier sans que l'arbre soit rationnel. Le contre-exemple est simple : il suffit d'étiquetter l'arborescence *rationnelle* $P = 1^*$ avec un ensemble infini, par exemple avec les entiers naturels $F = \mathbb{N}$, en associant à toute adresse sa longueur : $t = \{(u, |u|) \mid u \in P\}$.

3.2.7. Extension des relations d'ordre aux arbres. Le schéma d'extension que nous avons utilisé pour définir les arbres avec, à la base, la notion d'arborescence, mérite une abstraction qui sera utile pour démontrer les propriétés de l'ordre d'élagage et de l'ordre préfixe étendus aux arbres. Le choix d'une arborescence correspond au choix d'un ensemble clos par préfixe de mots d'un langage A (dans notre cas $A = \mathbb{N}_+^*$). Or, supposons qu'on veuille construire un nouveau domaine (celui des arbres) où les éléments seraient essentiellement des mots du langage A mais, cette fois, associés à des étiquettes d'un ensemble B. Cette construction ajouterait un choix : celui des étiquettes. Une façon de formaliser le choix, qui à présent serait double, est celle d'utiliser les fonctions partielles $A \rightharpoonup B$: on peut considérer de construire une fonction partielle $f : A \rightharpoonup B$ en sélectionnant les éléments de son domaine, c'est-à-dire en construisant $\mathrm{dom}(f)$, puis en leur associant les symboles, c'est-à-dire en définissant l'application $\mathrm{dom}(f) \to B$. Borner le premier choix à une famille $\mathfrak{F} \in 2^A$, par exemple celle des arborescences, signifie définir une famille dans le nouveau domaine : $\mathfrak{F}' = \{f : A \rightharpoonup B \mid \mathrm{dom}(f) \in \mathfrak{F}\}$, dans notre cas, la famille des arbres. Il est alors intéressant d'observer comme des propriétés de complétude peuvent être héritées par la nouvelle structure, une fois équipée de relations en rapport strict avec celles définies sur l'ancienne structure $\mathfrak{F}$.

PROPOSITION 3.2.7. *Soit* $\mathfrak{F}$ *une famille de parties d'un ensemble* A, *et soit* $\mathfrak{F}' = \{f : A \to B \mid \mathrm{dom}(f) \in \mathfrak{F}\}$ *une famille dans* $A \rightharpoonup B$. *Soit* $\mathfrak{R}'$ *une relation sur le carré* $\mathfrak{F}' \times \mathfrak{F}'$ *telle qu'il existe une relation* $\mathfrak{R}$ *sur le carré* $\mathfrak{F} \times \mathfrak{F}$ *et que le rapport des deux relations soit :*

$$y_1 \mathfrak{R}' y_2 \quad \Leftrightarrow \quad \mathrm{dom}(y_1) \mathfrak{R} \mathrm{dom}(y_2) \ \wedge \ y_1 \subseteq y_2$$

Sous ces hypothèses, si le couple $(\mathfrak{F}, \mathfrak{R})$ est un dcpo (resp. co-dcpo) d'ensembles, alors le couple $(\mathfrak{F}', \mathfrak{R}')$ est, lui aussi, un dcpo (resp. co-dcpo) d'ensembles.

DÉMONSTRATION. *La relation $\mathfrak{R}'$ est réflexive puisque $\mathfrak{R}$ et $\subseteq$ le sont. Pour la même raison, $\mathfrak{R}'$ est transitive. Et puisqu'elle implique l'inclusion ensembliste $\subseteq$, elle est aussi anti-symétrique. Donc le couple $(\mathfrak{F}', \mathfrak{R}')$ est un ordre partiel. Il reste à prouver que si $(\mathfrak{F}, \mathfrak{R})$ est clos par dirigé (resp. co-dirigé), alors $(\mathfrak{F}', \mathfrak{R}')$ l'est aussi (1) (resp. (2)).*

(1) $(\mathfrak{F}, \mathfrak{R})$ *dcpo d'ensembles* $\Rightarrow$ *pour tout dirigé* $D' \in \Delta(\mathfrak{F}', \mathfrak{R}')$ *il existe* $\sup_{\mathfrak{R}'} D' = \bigcup_{y \in \Delta'} y$
Soient $D' \in \Delta(\mathfrak{F}', \mathfrak{R}')$ *et* $U = \bigcup_{y \in \Delta'} y$. *Puisque* $\sup_{\mathfrak{R}} D$, *s'il existe, contiendra tous les éléments du dirigé* $y \in D'$, *il contiendra forcément leur union* U. *Donc, il suffit de montrer que* U *est un majorant de* D' *pour l'ordre* $\mathfrak{R}'$. *Soit* $y \in D'$. *Nous voulons prouver que* $y\mathfrak{R}'U$. *Par définition d'union,* $y \subseteq U$, *donc il reste à prouver que* $\mathrm{dom}(y)\mathfrak{R}\mathrm{dom}(U)$. *Calculons le domaine de l'union :* $\mathrm{dom}(U) = \mathrm{dom}\left(\bigcup_{y \in D'} y\right) = \{a \in A \mid \exists (a,b) \in y,\ y \in D'\} = \{a \in A \mid a \in \mathrm{dom}(y),\ y \in D'\} = \bigcup_{y \in D'} \mathrm{dom}(y)$.
Donc $\mathrm{dom}(U) = \bigcup_{x \in \mathrm{dom}(D')} x$ *où* $\mathrm{dom}(D')$ *est l'image de la fonction* $\mathrm{dom}(.)$ *sur le dirigé* D'. *On peut prouver que* $D \triangleq \mathrm{dom}(D')$ *est un dirigé dans* $\mathfrak{F}$ *: soient* $x_1, x_2 \in D$. *Par définition d'image, il existe* $y_1, y_2 \in D'$ *tels que* $x_1 = \mathrm{dom}(y_1)$ *et* $x_2 = \mathrm{dom}(y_2)$. *Puisque* D' *est un dirigé, il existe* $y_3 \in D$ *tel que* $y_1\mathfrak{R}'y_3$ *et* $y_2\mathfrak{R}'y_3$. *Donc il existe* $x_3 \in D$, $x_3 = \mathrm{dom}(y_3)$, *tel que* $x_1\mathfrak{R}x_3$ *et* $x_2\mathfrak{R}x_3$. *Autrement dit,* D *est un dirigé dans la structure* $(\mathfrak{F}, \mathfrak{R})$. *Par hypothèse l'union du dirigé* D *est la borne supérieure du dirigé au sens de l'ordre* $\mathfrak{R}$. *Donc* $\mathrm{dom}(y)\mathfrak{R}\mathrm{dom}(U)$.

(2) $(\mathfrak{F}, \mathfrak{R})$ *co-dcpo d'ensembles* $\Rightarrow$ *pour tout co-dirigé* $D' \in \nabla(\mathfrak{F}', \mathfrak{R}')$ *il existe* $\inf_{\mathfrak{R}'} D' = \bigcap_{y \in D'} y$
Soient $D' \in \nabla(\mathfrak{F}', \mathfrak{R}')$ *et* $I = \bigcap_{y \in D'} y$. *Puisque* $\inf_{\mathfrak{R}'} D$, *s'il existe, sera contenu dans tout élément du co-dirigé* $y \in D'$, *il contiendra forcément leur intersection* U. *Donc, il suffit de montrer que* I *est un minorant de* D' *pour l'ordre* $\mathfrak{R}'$. *Soit* $y \in D'$. *Nous voulons prouver que* $I\mathfrak{R}'y$. *Par définition d'intersection* $I \subseteq y$, *il reste à prouver que* $\mathrm{dom}(I)\mathfrak{R}\mathrm{dom}(y)$.
- *Nous voulons prouver, à présent, que :* $\mathrm{dom}(I) = \bigcap_{y \in D'} \mathrm{dom}(y)$.
  * *($\subseteq$)* $a \in \mathrm{dom}(I) \Rightarrow \exists (a,b) \in I \Rightarrow \forall y \in D'.\ (a,b) \in y \Rightarrow \forall y \in D'.\ a \in \mathrm{dom}(y) \Rightarrow a \in \bigcap_{y \in D'} \mathrm{dom}(y)$.
  * *($\supseteq$)* $a \in \bigcap_{y \in D'} \mathrm{dom}(y) \Rightarrow \forall y \in D'.\ a \in \mathrm{dom}(y) \Rightarrow \forall y \in D'. \exists b_t.(a, b_t) \in y$. *Or, supposons par l'absurde qu'il existe deux éléments* $y_1, y_2 \in D'$ *tels que* $b_1 \neq b_2$. *Par définition de co-dirigé, il existerait* $y_3 \in D'$ *tel que* $y_3\mathfrak{R}'y_1$ *et* $y_3\mathfrak{R}'y_2$. *Nous savons que* $\exists b_3.(a, b_3) \in y_3$. *Donc* y_1 *et* y_3 *devraient contenir cette association. Comme* y_1 *et* y_3 *sont des fonctions partielles* ($D' \subseteq \mathfrak{F}'$), *on doit avoir* $b_3 = b_2 = b_1$. *Il existe donc une étiquette unique* $b \in B$ *telle que* $\forall y \in D'.\ (a,b) \in y \Rightarrow \exists (a,b) \in I \Rightarrow a \in \mathrm{dom}(I)$.

Donc $\mathrm{dom}(I) = \bigcap_{x \in \mathrm{dom}(D')} x$ *où* $\mathrm{dom}(D')$ *est l'image de la fonction* $\mathrm{dom}(.)$ *sur le co-dirigé* D'. *On peut prouver que* $D \triangleq \mathrm{dom}(D')$ *est un dirigé dans* $\mathfrak{F}$ *: soient* $x_1, x_2 \in D$. *Par définition d'image il existe*

$y_1, y_2 \in D'$ *tels que* $x_1 = \mathrm{dom}(y_1)$ *et* $x_2 = \mathrm{dom}(y_2)$. *Puisque* D' *est un co-dirigé, il existe* $y_3 \in D$ *tel que* $y_3 \mathfrak{R}' y_1$ *et* $y_3 \mathfrak{R}' y_2$. *Donc il existe* $x_3 \in D$, $x_3 = \mathrm{dom}(y_3)$, *tel que* $x_3 \mathfrak{R} x_1$ *et* $x_3 \mathfrak{R} x_1$. *Autrement dit,* D *est un co-dirigé dans la structure* $(\mathfrak{F}, \mathfrak{R})$. *Par hypothèse, l'intersection du co-dirigé* D *est la borne inférieure du co-dirigé au sens de l'ordre* $\mathfrak{R}$. *Donc* $\mathrm{dom}(I) \mathfrak{R} \mathrm{dom}(y)$.

□

3.2.8. Élagages d'arbres. L'élagage est une autre notion qui s'étend naturellement aux arbres : il suffit de considérer un élagage de l'arborescence en utilisant, pour les adresses rescapées, les anciennes étiquettes.

Definition 3.2.8. *Étant donné un arbre* $t \in \mathrm{IT}(F)$, *un* élagage de l'arbre t *est une restriction de la fonction partielle* t *à un élagage (d'arborescences) de* $\mathrm{dom}(t)$. *Autrement dit, la relation* d'élagage *(entre arbres)* $\subseteq_{\mathrm{IT}(F)} \subseteq \big((\mathbb{N}_+^* \multimap F) \times \mathrm{IT}(F)\big)$ *est définie par la condition :*

$$t' \subseteq_{\mathrm{IT}(F)} t \;\overset{\triangle}{\Longleftrightarrow}\; \mathrm{dom}(t') \subseteq_{\mathrm{IT}} \mathrm{dom}(t) \;\wedge\; \forall u \in \mathrm{dom}(t').\ t'(u) = t(u).$$

□

Puisque t' est une fonction partielle $t' : \mathbb{N}_+^* \multimap F$ et que $\mathrm{dom}(t') \in \mathrm{IT}$, tout élagage d'un arbre est, lui aussi, un arbre. Ainsi la relation d'élagage est contenue dans le carré des arbres : $\subseteq_{\mathrm{IT}(F)} \subseteq (\mathrm{IT}(F) \times \mathrm{IT}(F))$. Le rapport entre la relation *d'élagage* des arborescences et l'homologue des arbres suit la trame habituelle, ce qui est vrai pour l'arbre est vrai pour son arborescence :

$$t_1 \subseteq_{\mathrm{IT}(F)} t_2 \;\Rightarrow\; \mathrm{dom}(t_1) \subseteq_{\mathrm{IT}} (t_2)$$

Les types d'élagage concernent la structure de l'arbre, c'est-à-dire son arborescence, non pas son étiquetage.

Definition 3.2.9. (Types d'élagages) *L'élagage* t' *d'un arbre* t *se dit* complet, radical, partiel *ou* déterministe *à une adresse* u *si l'élagage (d'arborescences)* $\mathrm{dom}(t')$ *est, respectivement, complet, radical, partiel ou déterministe vis-à-vis de* $\mathrm{dom}(t)$ *à la même adresse.* □

Comme pour les arborescences, nous utiliserons le symbole d'inclusion ensembliste $\subseteq_{\mathrm{IT}(F)}$ pour dénoter la relation d'élagage, cette fois, entre arbres. Le résultat suivant justifie ce choix.

Proposition 3.2.10. *Soient* $t, t' \in \mathrm{IT}(F)$. *Alors* $t' \subseteq_{\mathrm{IT}(F)} t$ *si et seulement si* t' *est un sous-ensemble de la relation* t, *i.e.* $t' \subseteq t$.

DÉMONSTRATION. *($\Rightarrow$) Soit* $(u, e) \in t'$. *Par définition* $\mathrm{dom}(t') \subseteq_{\mathrm{IT}} \mathrm{dom}(t)$ *et l'association des étiquettes est la même, donc* $(u, e) \in t$. *($\Leftarrow$) Puisque* $t' \in \mathrm{IT}(F)$ *nous avons* $t' : \mathbb{N}_+^* \multimap F$, $\mathrm{dom}(t') \in \mathrm{IT}$. *De plus, si* $t' \subseteq t$ *alors, d'une part* $\mathrm{dom}(t') \subseteq \mathrm{dom}(t)$ *donc* $\mathrm{dom}(t') \subseteq_{\mathrm{IT}} \mathrm{dom}(t)$, *et d'autre part si* $u \in \mathrm{dom}(t')$ *alors il existe une unique étiquette* $e \in F$ *telle que* $(u, e) \in t'$, *ce qui implique* $(u, e) \in t$, *autrement dit,* $t'(u) = t(u)$. □

Donc, la relation d'élagage est la restriction de l'inclusion ensembliste (définie sur $\mathbb{N}_+^* \times F$) aux arbres : $\subseteq_{\mathrm{IT}(F)} = \subseteq_{|\mathrm{IT}(F)}$. La structure $(\mathrm{IT}(F), \subseteq_{\mathrm{IT}(F)})$ n'est pas, comme son homologue $(\mathrm{IT}, \subseteq_{\mathrm{IT}})$, un treillis complet d'ensembles. Les arbres étant des fonctions partielles, ils associent au plus une étiquette à toute adresse. Par conséquent, les deux arbres $t_1 \triangleq \{(\varepsilon, a)\}$ et $t_2 \triangleq \{(\varepsilon, b)\}$ où $a \neq b$, ne trouvent aucun majorant possible dans $\mathrm{IT}(F)$. Autrement dit, le couple $(\mathrm{IT}(F), \subseteq_{\mathrm{IT}(F)})$ n'est pas un dirigé, condition nécessaire pour qu'il soit un sup-demi-treillis. En général, nous remarquons que les opérations d'union et d'intersection ensembliste appliqués aux arbres peuvent ne pas rendre un résultat qui soit un arbre, et cela pour différentes raisons. Le domaine du résultat peut, dans le cas de l'union, ne plus être une fonction partielle, à la différence de ses arguments. Ou bien, dans le cas de l'intersection, le résultat peut ne pas être clos par préfixe (par exemple $t_1 \triangleq \{(\varepsilon, a), (1, b)\}$ et $t_2 \triangleq \{(\varepsilon, b), (1, b)\}$ où $a \neq b$). On remarquera aussi un défaut de symétrie : $(\mathrm{IT}(F), \subseteq_{\mathrm{IT}(F)})$ est un co-dirigé puisque tout couple d'arbres compte au moins l'arbre vide parmi ses minorants. Cette discussion nous amène à la proposition suivante, qui nous rassure quant aux propriétés de complétude de l'ordre d'élagage sur les arbres : les sous-ensembles dirigés ou co-dirigés de $(\mathrm{IT}(F), \subseteq_{\mathrm{IT}(F)})$, atteignent leur bornes supérieure et inférieure, respectivement, à l'union et l'intersection ensemblistes de leurs éléments.

PROPOSITION 3.2.11. *Le couple* $(\mathrm{IT}(F), \subseteq_{\mathrm{IT}(F)})$ *est un bi-dcpo d'ensembles, avec un minimum* $\perp \triangleq \emptyset$. *Il s'agit aussi d'un dcpo algébrique où les compacts sont les arbres finis.*

DÉMONSTRATION. *Le rapport entre la relation* $\subseteq_{\mathrm{IT}(F)}$ *et* $\subseteq_{\mathrm{IT}}$ *est celui des hypothèses de la proposition 3.2.7, qu'on appliquera dans les deux sens,* $(\mathrm{IT}, \subseteq_{\mathrm{IT}})$ *étant à la fois un dcpo et un co-dcpo d'ensembles. De plus, la table de l'exemple 2.4.5, nous indique qu'il s'agit d'un dcpo algébrique dont les compacts sont les arbres ayant une arborescence finie. Autrement dit, les arbres finis.* □

3.2.9. Préfixes d'arbres. L'enchaînement des définitions sur les arbres se poursuit avec la notion de préfixe qui dérive elle aussi de son analogue sur les arborescences.

DEFINITION 3.2.12. *Étant donné un arbre* $t \in \mathrm{IT}(F)$, *un* préfixe de l'arbre t *est la restriction de* t *à un préfixe (d'arborescences) de son domaine. La relation* préfixe *(entre arbres), notée* $\leq_{\mathrm{IT}(F)}$, *est la relation contenue dans* $\big((\mathbb{N}_+^* \multimap F) \times \mathrm{IT}(F)\big)$ *définie de la façon suivante :*

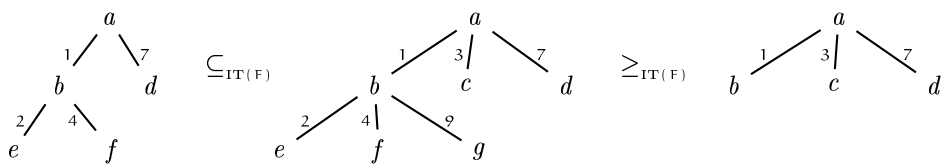
$$t' \leq_{IT(F)} t \overset{\triangle}{\iff} \mathrm{dom}(t') \leq_{IT} \mathrm{dom}(t) \ \wedge \ \forall u \in \mathrm{dom}(t').\ t'(u) = t(u).$$

□

Puisqu'un préfixe d'arborescence est un élagage d'arborescence, un préfixe d'arbre est un élagage d'arbre, donc un arbre. En d'autres termes, la relation *préfixe* entre arbres est limitée au carré des arbres $IT(F) \times IT(F)$ et incluse dans celle d'*élagage*. Nous retrouvons aussi le rapport habituel entre la relation sur les arbres et son homologue des arborescences :

$$t_1 \leq_{IT(F)} t_2 \ \Rightarrow \ \mathrm{dom}(t_1) \leq_{IT} \mathrm{dom}(t_2)$$

EXAMPLE 3.2.13. Considérons l'arbre défini par $t = \{\varepsilon \to a, 1 \to b, 3 \to c, 7 \to d, 1.2 \to e, 1.4 \to f, 1.9 \to g\}$. Un exemple d'élagage (qui n'est pas un préfixe) est l'arbre défini par le choix du sous-ensemble d'adresses $\{\varepsilon, 1, 7, 1.2, 1.4\} = \mathrm{dom}(t) \setminus \{3, 1.9\}$, qui conserve les associations de t. En revanche, un exemple de préfixe est l'arbre correspondant au choix des adresses $\{\varepsilon, 1, 3, 7\} = P \setminus \{1.2, 1.4, 1.9\}$ (tous les suffixes de l'adresse 1 sont élagués). Dans la représentation suivante, l'arbre se trouve au centre, l'élagage à gauche et le préfixe à droite.



□

DEFINITION 3.2.14. (PRÉFIXES RÉGULIERS D'ARBRES) *Étant donné* $t \in IT(F)$*, nous définissons l'*élagage de t au niveau $k \in \mathbb{N}$*, noté* $t/^k$*, comme la restriction de* t *au préfixe régulier de niveau* k *de son arborescence* $\mathrm{dom}(t)$*. En d'autres termes :* $t/^k \triangleq \{(u, e) \in t \mid u \in \mathrm{dom}(t)/^k\}$*. La relation* préfixe régulier *entre arbres, notée par le symbole* $\sqsubseteq_{IT(F)}$*, est définie par la condition* $t' \sqsubseteq_{IT(F)} t \overset{\triangle}{\iff} \exists k \in \mathbb{N}.\ t' = t/^k$.
□

La structure $(IT(F), \sqsubseteq_{IT(F)})$, comme son homologue $(IT, \sqsubseteq_{IT})$, n'est pas un ordre partiel, ni un préordre, la propriété réflexive étant impossible pour les arbres infinis à branchement fini (tout préfixe régulier d'un tel arbre serait forcément fini).

Le passage, chez les arborescences, de la relation d'élagage à celle de préfixe comporte une perte des propriétés de complétude ($(IT, \subseteq_{IT})$ est un treillis d'ensembles alors que $(IT, \leq_{IT})$ est un bi-dcpo d'ensembles). En revanche, auprès des arbres, la propriété de complétude ne diminue pas.

PROPOSITION 3.2.15. *Le couple* $(IT(F), \leq_{IT(F)})$ *est un bi-dcpo d'ensembles, avec la relation vide comme minimum* $\perp \triangleq \emptyset$*. Il s'agit aussi d'un dcpo algébrique où les compacts sont les arbres finis.*

DÉMONSTRATION. *Le rapport entre la relation* $\leq_{\mathrm{IT}(\mathsf{F})}$ *et* $\leq_{\mathrm{IT}}$ *est celui des hypothèses de la proposition 3.2.7, qu'on appliquera dans les deux sens,* $(\mathrm{IT}, \leq_{\mathrm{IT}})$ *étant à la fois un dcpo et un co-dcpo d'ensembles. Puisque les compacts de* $(\mathrm{IT}(\mathsf{F}), \subseteq_{\mathrm{IT}(\mathsf{F})})$ *sont les arbres finis, ces derniers seront encore compacts dans* $(\mathrm{IT}(\mathsf{F}), \leq_{\mathrm{IT}(\mathsf{F})})$*. Vice versa, un compact* t *de* $(\mathrm{IT}(\mathsf{F}), \leq_{\mathrm{IT}(\mathsf{F})})$ *ne pourra pas être infini. S'il l'était, la suite de ses préfixes réguliers serait un dirigé dont la borne supérieure (l'union) serait égale à* t*, mais aucun préfixe régulier, étant fini, ne pourrait dépasser* t*. D'autre part, tout arbre infini pourra s'écrire comme l'union (borne supérieure) de ses préfixes finis, qui sont compacts. Autrement dit, le dcpo* $(\mathrm{IT}(\mathsf{F}), \leq_{\mathrm{IT}(\mathsf{F})})$ *est algébrique.* □

3.2.10. Arbres à branchement fini. La classe des *arbres à branchement fini*, noté $\mathrm{BT}(\mathsf{F})$ est l'ensemble des arbres avec une arborescence à branchement fini :

$$\mathrm{BT}(\mathsf{F}) \triangleq \{\mathsf{t} \in \mathrm{IT}(\mathsf{F}) \mid \mathrm{dom}(\mathsf{t}) \in \mathrm{BT}\}$$

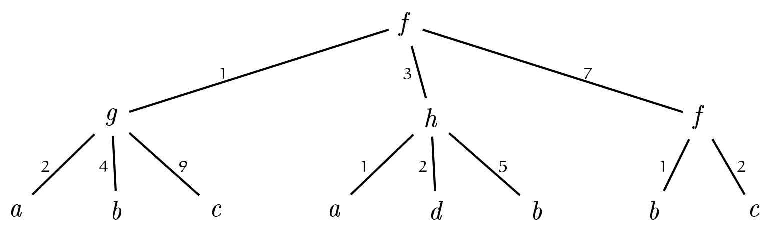
L'union arbitraire d'arborescences à branchement fini n'est pas forcément à branchement fini (3.1.11). Ainsi, nous ne pouvons pas espérer que $(\mathrm{BT}(\mathsf{F}), \subseteq_{\mathrm{BT}(\mathsf{F})})$ où $\subseteq_{\mathrm{BT}(\mathsf{F})} \triangleq \subseteq_{|\mathrm{BT}(\mathsf{F}) \times \mathrm{BT}(\mathsf{F})}$, soit un dcpo d'ensembles. Le rapport entre la structure des arborescences à branchement fini et celle des arbres à branchement fini est tel que les relations correspondantes d'élagage et préfixe sont dans les hypothèses de la proposition 3.2.7, c'est-à-dire :

$$\begin{array}{lcl} \mathsf{t}' \subseteq_{|\mathrm{BT}(\mathsf{F})} \mathsf{t} & \Leftrightarrow & \mathrm{dom}(\mathsf{t}') \subseteq_{|\mathrm{BT}} \mathrm{dom}(\mathsf{t}) \quad \wedge \quad \mathsf{t}' \subseteq \mathsf{t} \\ \mathsf{t}' \leq_{|\mathrm{BT}(\mathsf{F})} \mathsf{t} & \Leftrightarrow & \mathrm{dom}(\mathsf{t}') \leq_{|\mathrm{BT}} \mathrm{dom}(\mathsf{t}) \quad \wedge \quad \mathsf{t}' \subseteq \mathsf{t} \end{array}$$

Pour autant, les propriétés de complétude sont les mêmes que celles du domaine BT : la structure $(\mathrm{BT}(\mathsf{F}), \subseteq_{\mathrm{BT}(\mathsf{F})})$ est un co-dcpo d'ensembles tandis que la structure $(\mathrm{BT}(\mathsf{F}), \leq_{\mathrm{BT}(\mathsf{F})})$ où $\leq_{\mathrm{BT}(\mathsf{F})} \triangleq \leq_{|\mathrm{BT}(\mathsf{F}) \times \mathrm{BT}(\mathsf{F})}$, est un bi-dcpo d'ensembles. De plus, $(\mathrm{BT}(\mathsf{F}), \leq_{\mathrm{BT}(\mathsf{F})})$ est, par les arguments utilisés précédemment pour $(\mathrm{IT}(\mathsf{F}), \leq_{\mathrm{IT}(\mathsf{F})})$, un dcpo algébrique dont les compacts sont les arbres finis.

3.3. Les algèbres des termes

Les termes sont un cas particulier des arbres à branchement fini. Dans ce cadre, les étiquettes représenteront toujours une fonction ayant un nombre éventuellement *variable* d'arguments, mais toujours un nombre *fini*. Si pour les arbres génériques l'étiquetage est arbitraire, pour les *termes*, la manière d'étiqueter l'arborescence est limitée par la *signature* des symboles utilisés. Dans ce sens, les termes sont des arbres bien typés, où la signature est le type utilisé.



Dans cet exemple, les étiquettes f, g, h évoquent des *fonctions*, tandis que les symboles a, b, c, d évoquent des *constantes*. Les constantes sont d'un certain type (la *sorte*) et doivent forcément être des feuilles de l'arbre. Les fonctions, elles aussi, ont un type qui consiste en un domaine, qui est un produit cartésien de sortes, et un codomaine, qui est, lui aussi, une sorte. Le nombre et la sorte des branches découlants d'un symbole de fonction devra alors respecter le type de la fonction. Nous admettrons les fonctions d'arité variables, i.e. avec un nombre variable d'arguments. Dans l'exemple précédent, le symbole f ne pourra être qu'un symbole d'arité variable, étant présent aux adresses ε et 7 avec un nombre d'arguments (branches immédiates) différent. Le nombre d'argument d'une telle fonction sera arbitraire mais *fini*. Autrement dit, les termes seront toujours des arbres à branchement fini.

3.3.1. Types arborescents. Parmi les types algébriques (cf. section 1.2.1) nous qualifierons d'*arborescent* ceux qui dénotent des fonctions à zero ou plusieurs arguments sortés et restituant une valeur sortée ou une valeur d'un type de base. En effet, les *compositions* des fonctions de ce type (éventuellement polymorphes mais du premier ordre) peuvent être représentées sous forme d'arbre.

DEFINITION 3.3.1. (TYPES ARBORESCENTS) *Étant donné un ensemble de types de base $\mathcal{B} \supseteq \{1, 2, \mathbb{N}, \mathbb{N}_+, \mathbb{N}_\infty\}$, un ensemble de sortes $S = \{D_i\}_{i \in I}$ disjoint de $\mathcal{B}$, et un ensemble de variables $\mathcal{V}$ disjoint de $\mathcal{B}$ et de S, l'ensemble des* types arborescents ouverts *sur $\mathcal{B}$ et S, noté $\widetilde{\mathcal{T}}_{arb}(\mathcal{B}, S)$, est l'ensemble défini par la syntaxe $(\{\tau, \delta, \eta, \vartheta, \pi, \sigma, \chi\}, \mathcal{B} \cup S \cup \mathcal{V} \cup C \cup \{\times, \rightarrow, *, \forall, :, \mathsf{sort}, \{, ,, \}, .\}, \tau, ::=)$, où $::=$ est la relation suivante :*

$$
\begin{array}{rcl}
\widetilde{\mathcal{T}}_{arb}(\mathcal{B}, S) \ni \tau & ::= & \forall \alpha{:}\chi.\, \tau \mid \delta \rightarrow \vartheta \\
\delta & ::= & 1 \mid \eta \mid \eta_1 \times \cdots \times \eta_n \mid \eta^* \\
\eta & ::= & \sigma \mid \alpha \\
\vartheta & ::= & \pi \mid \sigma \\
\chi & ::= & \mathsf{sort} \mid \{\sigma_1, \cdots, \sigma_n\}
\end{array}
$$

où $\pi \in \mathcal{B}$, $\sigma \in S$ et $\alpha \in \mathcal{V}$. L'ensemble des types arborescents *(*clos*), noté $\mathcal{T}_{arb}(\mathcal{B}, S)$, est l'ensemble des types arborescents sans variables libres :*

$$\mathcal{T}_{arb}(\mathcal{B}, S) \triangleq \{\tau \in \widetilde{\mathcal{T}}_{arb}(\mathcal{B}, S) \mid \mathsf{FV}(\tau) = \emptyset\}$$

□

Les types arborescents représentent donc des fonctions (éventuellement polymorphes) dont le domaine δ peut être 1 (les fonctions constantes), ou bien une sorte η (exprimée éventuellement par une variable), ou bien un produit cartésien de sortes $\eta_1 \times \cdots \times \eta_n$ (exprimées éventuellement par des variables), ou bien l'ensemble des suites η^* sur une sorte (eventuellement exprimée par une variable). D'autre part, le codomaine ϑ peut être seulement un type de base π ou bien une sorte σ. Nous verrons par la suite que cela permet de classifier les fonctions entre constructeurs de *termes* et constructeurs de *primitives*.

Tout type arborescent τ a un domaine (le domaine des fonctions représentées) et un codomaine (le codomaine des fonctions représentées). En abusant de la notation déjà utilisée pour les fonctions, les deux seront notés respectivement $\mathrm{dom}(\tau)$ et $\mathrm{cod}(\tau)$, et définis de façon inductive :

$$\left\{\begin{array}{lcl} \mathrm{dom}(\forall\alpha{:}\chi.\,\tau) & = & \forall\alpha{:}\chi.\,\mathrm{dom}(\tau) \\ \mathrm{dom}(\delta \to \vartheta) & = & \delta \end{array}\right. \qquad \left\{\begin{array}{lcl} \mathrm{cod}(\forall\alpha{:}\chi.\,\tau) & = & \mathrm{cod}(\tau) \\ \mathrm{cod}(\delta \to \vartheta) & = & \vartheta \end{array}\right.$$

Par la suite nous pourrons éviter d'écrire les quantifications des variables, quand cela ne sera pas relevant, présentes dans un type ou dans le domaine d'un type.

3.3.2. Signatures arborescentes. Les termes sont des arbres particuliers associés à des signatures dont les types des opérations sont inclus dans le sous-langage des types arborescents.

DEFINITION 3.3.2. (SIGNATURE ARBORESCENTE, CONSTRUCTEURS) *Une signature algébrique* $\Sigma = (\mathcal{B}, S, F, T)$ *est* arborescente (polymorphe du premier ordre) *si tous les types impliqués par les opérations algébriques sont des types arborescents polymorphes du premier ordre :*

$$\{T(f) \mid f \in F\} \subseteq \mathcal{T}_{\mathrm{arb}}(\mathcal{B}, S)$$

Pour tout $f \in F$*, le* codomaine *du symbole est* $\mathrm{cod}_T(f) \triangleq \mathrm{cod}(T(f))$*, son* domaine*, noté* $\mathrm{dom}_T(f) \triangleq \mathrm{dom}(T(f))$*. L'ensemble des symboles de* F *est divisé en deux partie :*

$$F = F_S \oplus F_{\mathcal{B}}$$

où F_S *est l'ensemble des symboles de fonctions à valeur dans une sorte* $F_S = \{f \in F \mid \mathrm{cod}_T(f) \in S\}$*, et* $F_{\mathcal{B}}$ *est l'ensemble des fonctions à valeur dans un type de base* $F_{\mathcal{B}} = \{f \in F \mid \mathrm{cod}_T(f) \in \mathcal{B}\}$*. Les symboles de* F_S *seront appelés* constructeurs de termes*, ceux de* $F_{\mathcal{B}}$ *seront appelés* constructeurs de primitives*.* ☐

Ainsi, dans une signature arborescente, les opérations ne peuvent être que des fonctions à zéro (constantes), un ou plusieurs arguments. Dans le cas où $\mathrm{dom}_T(f) = \eta^*$ le nombre d'arguments pourra être variable et on dira qu'un symbole $f \in F$ de ce type est d'*arité variable* (éventuellement nulle). Sinon il sera d'*arité fixe,* égale au nombre d'arguments et noté $|f|$:

$$|f| \triangleq \left\{\begin{array}{lll} 0 & \text{si} & \mathrm{dom}_T(f) = 1 \\ 1 & \text{si} & \mathrm{dom}_T(f) = \eta \\ k & \text{si} & \mathrm{dom}_T(f) = \eta_1 \times \cdots \times \eta_k \end{array}\right.$$

Si une signature arborescente correspond à la description de la structure d'une algèbre constituée de fonctions, les *termes*, d'un autre coté, représentent les compositions *fonctionnelles* possibles (finies ou infinies), autrement dit, les compositions des fonctions de l'algèbres, compatibles vis-à-vis de l'arité et du type des arguments.

DEFINITION 3.3.3. (ARBRES D'UNE SIGNATURE ARBORESCENTE, TERMES, PRIMITIVES) *Étant donné une signature arborescente* $\Sigma = (\mathcal{B}, S, F, T)$*, un* arbre de la signature Σ *est un arbre étiqueté dans* F*,* $t \in \mathrm{BT}(F)$*, à branchement fini et tel que pour toute adresse* $u \in \mathrm{dom}(t)$ *l'arbre vérifie les conditions suivantes :*

(1) *Si* $(u, f) \in \mathrm{Feuilles}(t)$*, alors :*

   (a) $\mathrm{dom}_T(f) = 1$ *ou* $\mathrm{dom}_T(f) = \eta^*$

(2) *Si* $(u, f) \in \mathrm{Noeuds}(t)$, $\mathrm{Succ}^{(t,u)} = (i_1, f_1)..(i_n, f_n)$, *alors :*

(a) $\mathrm{dom}_T(f) = \eta_1 \times \cdots \times \eta_n$ *ou* $\mathrm{dom}_T(f) = \eta^*$

(b) *si* $f : \sigma_1 \times \cdots \times \sigma_n \to \vartheta$ *(où* $n \geq 1$*), alors :*

$$\forall j = 1..n.\ \ \mathrm{cod}_T(f_j) = \sigma_j$$

(c) *si* $f : \sigma^* \to \vartheta$, *alors :*

$$\forall j = 1..n.\ \ \mathrm{cod}_T(f_j) = \sigma$$

(d) *si* $f : \forall \alpha_1 {:} \chi_1. \ldots \forall \alpha_k {:} \chi_k. \delta \to \vartheta$, *alors :*
$\exists s_1 \in \chi_1, .., s_k \in \chi_k$ *tels que* $\delta' = \delta[{}^{s_1}/_{\alpha_1}] \cdots [{}^{s_k}/_{\alpha_k}]$
vérifie la condition :

$$\begin{cases} \forall j = 1..n.\ \mathrm{cod}_T(f_j) = \sigma_j & si \quad \delta' = \sigma_1 \times \cdots \times \sigma_n \\ \forall j = 1..n.\ \mathrm{cod}_T(f_j) = \sigma & si \quad \delta' = \sigma^* \end{cases}$$

Si le symbole à la racine est un constructeur de termes, l'arbre est un terme*, si en revanche il s'agit d'un constructeur de primitive, l'arbre est une* primitive*. La* sorte d'un terme *est le codomaine du symbole à la racine du terme. Les ensembles des* arbres arbitraires, rationnels *ou* finis de la signature Σ, *seront notés respectivement* $\mathrm{IT}(\Sigma)$, $\mathrm{RT}(\Sigma)$, $\mathrm{T}(\Sigma)$ *; les ensembles des termes arbitraires, rationnels ou finis de sorte* $\sigma \in S$ *seront notés* $\mathrm{IT}(\Sigma, \sigma)$, $\mathrm{RT}(\Sigma, \sigma)$, $\mathrm{T}(\Sigma, \sigma)$, *et de la même façon, les ensembles des primitives arbitraires, rationnelles ou finies de type* $\pi \in \mathcal{B}$ *seront notés* $\mathrm{IT}(\Sigma, \pi)$, $\mathrm{RT}(\Sigma, \pi)$, $\mathrm{T}(\Sigma, \pi)$. *Les ensembles de tous les termes (indépendemment de la sorte) arbitraires, rationnels ou finis seront notés* $\mathrm{IT}(\Sigma, S)$, $\mathrm{RT}(\Sigma, S)$, $\mathrm{T}(\Sigma, S)$ *; les ensembles de toutes les primitives arbitraires, rationnelles ou finies seront notés* $\mathrm{IT}(\Sigma, \mathcal{B})$, $\mathrm{RT}(\Sigma, \mathcal{B})$, $\mathrm{T}(\Sigma, \mathcal{B})$. □

On remarquera que, s'il en existe, les fils d'un arbre de la signature sont forcément des termes, puisqu'ils doivent avoir comme codomaine une sorte. En effet, dans la construction d'un arbre de la signature, les symboles primaires peuvent être utilisés seulement à la racine[1].

La distinction entre arité *fixe* ou *variable* que l'on fait sur les symboles d'une signature se retrace sur les adresses d'un terme : quand le symbole $t(u) \in F$ est à arité fixe nous disons que l'*adresse* u est *à arité fixe*, sinon nous disons qu'elle est *à arité variable*. Nous disons qu'une adresse u d'un arbre t est de *type* $\vartheta \in S$ si $f \triangleq t(u)$ est un symbole de F dont le codomaine est ϑ. L'écriture $\mathrm{IT}(\Sigma)$ ne doit pas faire oublier qu'il s'agit d'arbres à branchement fini : $\mathrm{IT}(\Sigma) \subseteq \mathrm{BT}(F \cup \mathcal{B})$ où $\Sigma = (\mathcal{B}, S, F, T)$. L'ambiguïté avec la notation qui classifie les arbres est résolue par le symbole de signature qui remplace, dans les parenthèses, un simple alphabet d'étiquettes.

[1] Nous nous intéresserons à ce genre d'arbre dans le chapitre 8, où les *algèbres logiques* contiennent des primitives, les *prédicats logiques*, certains de type monomorphe $\sigma_1 \times \ldots \times \sigma_n \to 2$, d'autres, comme le prédicat d'égalité $=: \forall \alpha{:}\mathsf{sort}.\ \alpha \times \alpha \to 2$, de type polymorphe.

3.3.3. La famille des algèbres des termes finis. Les *algèbres des termes finis* (*sur une signature*) constituent une famille d'algèbres canoniques qui sont associées à toute signature arborescente.

DEFINITION 3.3.4. (ALGÈBRES DES TERMES FINIS SUR UNE SIGNATURE) *Étant donné une signature arborescente* $\Sigma = (\mathcal{B}, \mathsf{S}, \mathsf{F}, \mathsf{T})$*, la famille des* algèbres des termes (finis) sur Σ*, notée* $\mathsf{TAlg}(\Sigma)$*, est la famille des algèbres* $(\hat{\cdot} : \mathcal{B} \to \hat{\mathcal{B}} \oplus \mathsf{S} \to \hat{\mathsf{S}} \oplus \mathsf{F_S} \to \hat{\mathsf{F}}_\mathsf{S} \oplus \mathsf{F}_\mathcal{B} \to \hat{\mathsf{F}}_\mathcal{B}, \mathsf{T})$ *telles que :*

(1) *la classe des domaines est constituée par les domaines des termes finis :*

$$\hat{\mathsf{S}} = \{\mathrm{T}(\Sigma, \sigma) \mid \sigma \in \mathsf{S}\}$$

(2) *tout constructeur de terme* $\hat{\mathsf{f}} \in \hat{\mathsf{F}}_\mathsf{S} = \{\hat{\mathsf{f}} \mid \mathrm{cod}_\mathsf{T}(\mathsf{f}) \in \mathsf{S}\}$ *est la fonction qui construit le terme fini ayant le symbole* f *à la racine, et autant de fils que d'arguments :*

$$\begin{cases} \hat{\mathsf{f}}() & = \{(\varepsilon, \mathsf{f})\} & si \;\; \mathrm{dom}(\mathsf{f}) = \mathbb{1} \\ \hat{\mathsf{f}}(\mathsf{t}) & = \{(\varepsilon, \mathsf{f})\} \cdot_1 \mathsf{t} & si \;\; \mathrm{dom}(\mathsf{f}) = \eta \\ \hat{\mathsf{f}}(\mathsf{t}_1, .., \mathsf{t}_n) & = \{(\varepsilon, \mathsf{f})\} \cdot_1 \mathsf{t}_1 \; \cdots \; \cdot_n \mathsf{t}_n & si \;\; \mathrm{dom}(\mathsf{f}) = \eta_1 \times .. \times \eta_n \\ \hat{\mathsf{f}}\,\mathsf{t}_1..\mathsf{t}_n & = \{(\varepsilon, \mathsf{f})\} \cdot_1 \mathsf{t}_1 \; \cdots \; \cdot_n \mathsf{t}_n & si \;\; \mathrm{dom}(\mathsf{f}) = \eta^* \end{cases}$$

□

Dans la famille des algèbres des termes, si la classe des domaines $\hat{\mathsf{S}}$ et l'ensemble des *opérations sortées* $\hat{\mathsf{F}}_\mathsf{S}$ est complètement défini, en revanche, l'ensemble des *opérations primitives* $\hat{\mathsf{F}}_\mathcal{B}$ est arbitraire. Un exemple typique de famille d'algèbres des termes est celle des *interprétations de Herbrand*, que nous discuterons au cours du chapitre 8. Lorsque la signature ne contient que des constructeurs de termes ($\mathsf{F} = \mathsf{F_S}$), et une fois établis les domaines de base $\mathcal{B}$, il existe une seule algèbre des termes possible, qui est donc appelée *l'algèbre des termes*.

Il existe une façon canonique d'interpréter toute algèbre des termes vers une algèbre compatible et de la même signature.

DEFINITION 3.3.5. (INTERPRÉTATION CANONIQUE D'UNE ALGÈBRE DES TERMES) *Étant donné une algèbre* $\mathsf{A} = ((.)^\mathsf{A} : \mathcal{B} \to \mathcal{B}^\mathsf{A} \;\oplus\; \mathsf{S} \to \mathsf{S}^\mathsf{A} \;\oplus\; \mathsf{F} \to \mathsf{F}^\mathsf{A}, \mathsf{T})$ *sur la signature arborescente* $\Sigma = (\mathcal{B}, \mathsf{S}, \mathsf{F}, \mathsf{T})$*, l'*interprétation canonique *de toute algèbre des termes finis* $\mathsf{A_t} \in \mathsf{TAlg}(\Sigma)$ *vers l'algèbre* $\mathsf{A} \in \mathsf{Alg}(\Sigma)$*, notée* $[\![.]\!]_\mathsf{A} : \forall\alpha\!:\!\mathsf{S}.\,\alpha \to \alpha$*, est l'interprétation algébrique (cf. section 1.3.2) définie de façon inductive :*

$$[\![\mathsf{t}]\!]_\mathsf{A} \triangleq \begin{cases} \mathsf{f}^\mathsf{A}() & si \;\; \mathsf{t} = \{(\varepsilon, \mathsf{f})\} \\ \mathsf{f}^\mathsf{A}([\![\mathsf{t}_1]\!]_\mathsf{A}) & si \;\; \mathsf{t} = \{(\varepsilon, \mathsf{f})\} \cdot_1 \mathsf{t}_1 \\ \mathsf{f}^\mathsf{A}([\![\mathsf{t}_1]\!]_\mathsf{A}, .., [\![\mathsf{t}_n]\!]_\mathsf{A}) & si \;\; \mathsf{t} = \{(\varepsilon, \mathsf{f})\} \cdot_1 \mathsf{t}_1 \; \cdots \; \cdot_n \mathsf{t}_n \,,\; \mathrm{dom}(\mathsf{f}) = \eta_1 \times .. \times \eta_n \\ \mathsf{f}^\mathsf{A}\,[\![\mathsf{t}_1]\!]_\mathsf{A} .. [\![\mathsf{t}_n]\!]_\mathsf{A} & si \;\; \mathsf{t} = \{(\varepsilon, \mathsf{f})\} \cdot_1 \mathsf{t}_1 \; \cdots \; \cdot_n \mathsf{t}_n \,,\; \mathrm{dom}(\mathsf{f}) = \eta^* \end{cases}$$

□

Les opérations algébriques étant de type élémentaire, l'interprétation canonique conserve, par définition, tous les constructeurs (cf. corollaire 1.3.4). Donc, si les primitives sont, elles aussi conservées, l'interprétation canonique est un homomorphisme (et il s'agit de l'unique possible, cf. par exemple [**Tucker & Zucker, 2000**]).

3.3.4. Termes avec variables. Il est courant d'ajouter des symboles spéciaux, les *variables*, aux langages des termes et des primitives sur une signature. Entre autres, cela permet d'exprimer, par un seul arbre avec variables, la multitude d'arbres qui peuvent s'obtenir par des greffes aux positions des variables.

DEFINITION 3.3.6. (ARBRES, TERMES ET PRIMITIVES AVEC VARIABLES) *Étant donné une signature arborescente* $\Sigma = (\mathcal{B}, \mathsf{S}, \mathsf{F}, \mathsf{T})$*, et un ensemble* $\mathsf{V} = \biguplus_{s\in \mathsf{S}} \mathsf{V}_s$ *de symboles de variables disjoint de* S *et de* $\mathcal{B}$*, la* signature augmentée des variables*, notée* $\Sigma{+}\mathsf{V}$*, est définie par :*

$$\Sigma{+}\mathsf{V} \triangleq (\mathsf{S},\ \mathsf{F} \oplus \mathsf{V}, \mathsf{T}\ \oplus\ \biguplus_{s\in \mathsf{S}} \mathsf{V}_s \times \{\to s\})$$

Un arbre avec variables *est un arbre de* $\mathrm{IT}(\Sigma{+}\mathsf{V})$*. Un* terme avec variables *est un terme de* $\mathrm{IT}(\Sigma{+}\mathsf{V}, \mathsf{S})$*. Une* primitive avec variables *est une primitive de* $\mathrm{IT}(\Sigma{+}\mathsf{V}, \mathcal{B})$*.*
□

Dans une signature $\Sigma{+}\mathsf{V}$, d'une part l'ensemble des symboles de fonctions est augmenté, par rapport à Σ, par l'ensemble des symboles de variables pour chaque sorte ; d'autre part, toute variable est une constante de la sorte correspondante. L'ensemble des variables d'un arbre *fini* (terme ou primitive) $t \in \mathrm{T}(\Sigma{+}\mathsf{V})$ peut être défini de façon inductive :

$$\mathrm{vars}(t) \triangleq \begin{cases} \{\mathsf{X}\} & \text{si} \quad t = \{(\varepsilon, \mathsf{X})\}\,,\ \mathsf{X} \in \mathsf{V} \\ \\ \mathrm{vars}(t_1) \cup \cdots \cup \mathrm{vars}(t_n) & \text{si} \quad \mathrm{Fils}(t) = t_1..t_n \end{cases}$$

3.3.4.1. *Substitutions.* La notion de substitution permet, pour un arbre (terme ou primitive) donné, de formaliser le remplacement des variables par d'autres arbres. Ce derniers seront forcément des termes. En effet, puisque les variables sont, comme les termes, des constructeurs (dans la signature augmentée), elles ne peuvent être remplacées que par des termes (et de la même sorte). Les termes remplaçant les variables pourront, à leur tour, contenir des variables. Ainsi, l'effet d'une substitution n'impliquera pas forcément l'élimination des variables du terme ou de la primitive.

DEFINITION 3.3.7. (SUBSTITUTION) *Étant donné une signature arborescente* $\Sigma = (\mathcal{B}\,\mathsf{S}, \mathsf{F}, \mathsf{T})$*, et un ensemble* $\mathsf{V} = \biguplus_{s\in \mathsf{S}} \mathsf{V}_s$ *de symboles de variables disjoint de* S *et de* $\mathcal{B}$*, une* substitution *est une fonction partielle polymorphe :*

$$\theta\ :\ \forall\alpha{:}\mathsf{S}.\, \mathsf{V}_\alpha \,\dashrightarrow\, \mathrm{T}(\Sigma{+}\mathsf{V}, \alpha)$$

telle que le domaine *de la substitution, i.e. l'ensemble* $\mathrm{dom}(\theta) = \cup_{\alpha\in \mathsf{S}} \mathrm{dom}(\theta_\alpha)$*, est fini. La substitution est qualifiée de* ground *si elle appartient au type, plus précis, suivant :*

$$\theta\ :\ \forall\alpha{:}\mathsf{S}.\, \mathsf{V}_\alpha \,\dashrightarrow\, \mathrm{T}(\Sigma, \alpha)$$

L'ensemble des substitutions sera noté Subst. □

Une substitution est donc une fonction qui associe à toute variable de type σ, un terme du même type, éventuellement avec variables. Lorsque les termes associés par la substitution ne contiennent pas de variables, la substitution est qualifiée de *ground*. L'application d'une substitution à un terme t permet de greffer les termes indiqués par la substitution au adresses de t où se trouvent les variables.

Definition 3.3.8. (Application d'une substitution, extension aux arbres) *Étant donné une signature arborescente* $\Sigma = (\mathcal{B}, S, F, T)$*, un ensemble* $V = \biguplus_{s \in S} V_s$ *de symboles de variables disjoint de* S *et de* $\mathcal{B}$*, et une substitution* $\theta : \forall\alpha : S.\, V_\alpha \twoheadrightarrow T(\Sigma{+}V, \alpha)$*, l'*application de θ à un arbre fini $t \in T(\Sigma{+}V)$ *(ou* Subst-application*), notée* $t\,\theta$*, est la définie inductivement par :*

$$t\,\theta \triangleq \begin{cases} \theta(X) & \textit{si} \quad t = \{(\varepsilon, X)\},\ X \in \mathrm{dom}(\theta) \\ \\ \{(\varepsilon, t\,\varepsilon)\} \cdot_1 (t_1\,\sigma) \cdots \cdot_n (t_n\,\sigma) & \textit{si} \quad \mathrm{Fils}(t) = t_1..t_n \end{cases}$$

La composition *de deux substitutions* $\theta_1, \theta_2 : \forall\alpha : S.\, V_\alpha \twoheadrightarrow T(\Sigma{+}V, \alpha)$ *(ou* Subst-composition*) est la substitution notée* $\theta_1 \cdot \theta_2$ *et définie par* $(\theta_1 \cdot \theta_2)(X) \triangleq (X\,\theta_1)\,\theta_2$*. L'*extension *de la substitution* θ aux arbre finis *est la fonction polymorphe totale :*

$$\hat{\theta} \ : \ \forall\alpha : (S \oplus \mathcal{B}).\, T(\Sigma{+}V, \alpha) \to T(\Sigma{+}V, \alpha)$$

définie par $\hat{\theta}(t) \triangleq t\,\theta$. □

L'extension aux arbres (termes ou primitives) finis permet de justifier les noms utilisés pour les opérations d'*application* (Subst-application) et de *composition* de substitutions (Subst-composition). En effet la Subst-application dénote l'*application fonctionnelle* de l'extension de la substitution :

$$\hat{\theta}(t) = t\,\theta$$

tandis que la Subst-composition de substitutions dénote la *composition fonctionnelle* des correspondantes extensions :

$$\widehat{\theta_1 \cdot \theta_2} = \hat{\theta}_2 \circ \hat{\theta}_1$$

L'ensemble des variables d'une substitution θ, que nous noterons $\mathrm{vars}(\theta)$, est l'ensemble des variables qu'elle introduit pas son application :

$$\mathrm{vars}(\theta) \triangleq \bigcup_{X \in \mathrm{dom}(\theta)} \mathrm{vars}(X \cdot \theta)$$

Definition 3.3.9. (Substitution idempotente) *Étant donné une signature arborescente* $\Sigma = (\mathcal{B}\, S, F, T)$*, et un ensemble* $V = \biguplus_{s \in S} V_s$ *de symboles de variables disjoint de* S *et de* $\mathcal{B}$*, une substitution* θ *est* idempotente *lorsque la fonction* $\lambda t.\, t \cdot \theta : T(\Sigma{+}V) \to T(\Sigma{+}V)$ *est idempotente, c'est-à-dire lorsque* $\forall t \in T(\Sigma{+}V).\ (t \cdot \theta) \cdot \theta = t \cdot \theta$*. L'ensemble des substitutions idempotentes sera noté* ISubst. □

Il est simple à vérifier qu'une substitution est idempotente si et seulement si $\mathrm{dom}(\theta) \cap \mathrm{vars}(\theta) = \emptyset$.

DEFINITION 3.3.10. (RENOMMAGE) *Étant donné une signature arborescente $\Sigma = (\mathcal{B}\,\mathsf{S}, \mathsf{F}, \mathsf{T})$, et un ensemble $\mathsf{V} = \biguplus_{s \in \mathsf{S}} \mathsf{V}_s$ de symboles de variables disjoint de S et de $\mathcal{B}$, un* renommage *est une substitution de type $\theta \;:\; \forall\alpha : \mathsf{S}.\, \mathsf{V}_\alpha \multimap \mathsf{V}_\alpha$ idempotente et injective.* □

En d'autres termes, une substitution θ est un renommage si son extension aux termes finis $\hat{\theta}$ est une fonction bijective.

DEFINITION 3.3.11. (PRÉORDRE ET ÉQUIVALENCE DE FILTRAGE) *Étant donné une signature arborescente $\Sigma = (\mathcal{B}\,\mathsf{S}, \mathsf{F}, \mathsf{T})$, et un ensemble $\mathsf{V} = \biguplus_{s \in \mathsf{S}} \mathsf{V}_s$ de symboles de variables disjoint de S et de $\mathcal{B}$, une substitution θ_1 est* plus générale *qu'une autre θ_2, noté $\theta_1 \leq \theta_2$, s'il existe une troisième substitution θ telle que $\theta_2 = \theta_1 \cdot \theta$. Une substitution θ_1 est* équivalente *à une autre θ_2, noté $\theta_1 \approx \theta_2$, s'il existe un renommage θ tel que $\theta_2 = \theta_1 \cdot \theta$.* □

L'équivalence de filtrage mérite bien son nom : il s'agit de l'équivalence engendrée par la relation $\leq$ qui, elle, est un relation de préordre (propre) sur Subst. Ainsi, la structure $(\mathsf{ISubst}_{\approx}, \leq)$ est un ordre partiel (cf. section 2.1.2.3).

THEOREM 3.3.12. ([**Eder, 1985**]) *La structure des substitutions idempotentes $(\mathsf{ISubst}^{\top}_{\approx}, \leq)$ modulo renommage et équipée de l'ordre de filtrage et d'un élément adjoint $\top$, plus grand que tous les autres, est un treillis complet. L'élément maximum est $\top$, l'élément minimum est la classe de la substitution vide $\emptyset_{\approx}$.* □

Ce théorème affirme, entre autres, que la classe des substitutions plus générales est celle de la subsitution vide, c'est-à-dire celle des substitution θ telles que leur extension aux arbres $\hat{\theta}$ est la fonction identité. Autrement dit, il n'y a pas de substitution plus générale d'une substitution qui laisse intacts tous les arbres auxquels elle s'applique. Le théorème autorise aussi la définition d'*unificateur plus général* ou *mgu* (de l'anglais Most General Unifier) entre deux arbres finis.

DEFINITION 3.3.13. (MGU) *Étant donné une signature arborescente $\Sigma = (\mathcal{B}\,\mathsf{S}, \mathsf{F}, \mathsf{T})$, et un ensemble $\mathsf{V} = \biguplus_{s \in \mathsf{S}} \mathsf{V}_s$ de symboles de variables disjoint de S et de $\mathcal{B}$, la fonction partielle :*

$$\mathrm{mgu} \,:\, \mathrm{T}(\Sigma{+}\mathsf{V}) \times \mathrm{T}(\Sigma{+}\mathsf{V}) \multimap \mathsf{ISubst}_{\approx}$$

restitue la classe des unificateurs plus généraux *ou (*principaux*) d'un couple d'arbres finis, qui est la borne inférieure, si elle existe, de tous les classes d'unificateurs du couple :*

$$\mathrm{mgu}(\mathsf{t}_1, \mathsf{t}_2) \stackrel{\leftharpoonup}{=} \inf_{(\mathsf{ISubst}_{\approx}, \leq)} \{[\theta]_{\approx} \mid \mathsf{t}_1\,\theta \;=\; \mathsf{t}_2\,\theta\}$$

Toute substitution idempotente $\theta \in \mathrm{mgu}(\mathsf{t}_1, \mathsf{t}_2)$ est un unificateur principal *des arbres finis t_1 et t_2.* □

Nous pouvons donc affirmer que l'unificateur principal d'un couple d'arbres finis est unique modulo renommage. Pour le théorème 3.3.12, la seule cause possible de non existence de la borne inférieure est l'absence d'unificateurs entre les deux arbres, c'est-à-dire la condition $\{\theta \mid t_1\,\theta = t_2\,\theta\} = \emptyset$. Si les arbres sont *unifiables* (il existe des unificateurs), alors il existe un unificateur principal (modulo rennomage). Pour résumer, la condition :

$$\theta \in \mathrm{mgu}(t_1, t_2)$$

signifie :

(1) que les arbres t_1 et t_2 sont unifiables

(2) que θ est l'unificateur plus général, unique modulo rennomage

3.3.4.2. *Environnements.* Étant donné une algèbre A sur la signature arborescente Σ, pour donner un sens aux termes avec variables $t \in T(\Sigma{+}V)$, il suffit de savoir interpréter les variables sortées par des valeurs de l'algèbre.

Definition 3.3.14. (Environnement) *Étant donné une algèbre A sur une signature arborescente $\Sigma = (\mathcal{B}\, S, F, T)$, et un ensemble $V = \biguplus_{s\in S} V_s$ de symboles de variables disjoint de S et de $\mathcal{B}$, un* environnement *dans A est une fonction totale polymorphe :*

$$\xi \;:\; \forall\alpha{:}S.\, V_\alpha \to \alpha^A$$

□

Autrement dit, un *environnement* dans A est une quelconque fonction $\xi : \forall\alpha{:} S.\, V_\alpha \to \alpha^A$ affectant à toute variable sortée $X \in V_s$ une dénotation $\xi(X) \in s^A$ dans le domaine algèbrique interprétant la sorte. Nous noterons ENV_A l'ensemble des environnements à valeurs dans l'algèbre A. Tout couple (A, ξ), avec $\xi \in \mathrm{ENV}_A$, définit complétement l'interprétation des termes $t \in T(\Sigma{+}V)$ et sera appelé *interprétation implicite* des termes avec variables.

Definition 3.3.15. (Interprétations implicites) *L'ensemble des* interprétations implicites *des termes avec variables sur la signature $\Sigma + V$, est défini par :*

$$\mathsf{IAlg}(\Sigma, V) \triangleq \{(A, \xi) \mid A \in \mathsf{LAlg}(\Sigma),\ \xi \in \mathrm{ENV}_A\}$$

□

3.3.4.3. *Interprétation d'un terme avec variables.* La donnée d'un couple (A, ξ), où A est une algèbre sur Σ et ξ un environnement dans A, correspond à la définition d'une interprétation d'une quelconque algèbre des termes avec variables $A_t \in \mathsf{TAlg}(\Sigma{+}V)$ vers l'algèbre A.

DEFINITION 3.3.16. (INTERPRÉTATION CANONIQUE D'UNE ALGÈBRE DES TERMES AVEC VARIABLES) *Étant donné une algèbre* $A = ((.)^A : \mathcal{B} \to \mathcal{B}^A \oplus S \to S^A \oplus F \to F^A, T)$ *sur la signature arborescente* $\Sigma = (\mathcal{B}, S, F, T)$*, et un environnement* $\xi \in \mathrm{ENV}_A$*, l'*interprétation canonique *de toute algèbre des termes finis avec variables* $A_t \in \mathsf{TAlg}(\Sigma{+}V)$ *vers l'algèbre* $A \in \mathsf{Alg}(\Sigma)$*, notée* $[\![.]\!]_{A,\xi} : \forall\alpha{:}S.\,\alpha \to \alpha$*, est l'interprétation algébrique (cf. section 1.3.2) définie de façon inductive :*

$$[\![t]\!]_{A,\xi} \triangleq \begin{cases} \xi(X) & \textit{si } t{=}\{(\varepsilon,X)\},\ X \in V \\ f^A() & \textit{si } t{=}\{(\varepsilon,f)\},\ f \in F \\ f^A([\![t_1]\!]_{A,\xi}) & \textit{si } t{=}\{(\varepsilon,f)\} \cdot_1 t_1 \\ f^A([\![t_1]\!]_{A,\xi}, .., [\![t_n]\!]_{A,\xi}) & \textit{si } t{=}\{(\varepsilon,f)\} \cdot_1 t_1 \cdots \cdot_n t_n,\ \mathrm{dom}(f){=}\eta_1 \times .. \times \eta_n \\ f^A\, [\![t_1]\!]_{A,\xi} .. [\![t_n]\!]_{A,\xi} & \textit{si } t{=}\{(\varepsilon,f)\} \cdot_1 t_1 \cdots \cdot_n t_n\ ,\ \mathrm{dom}(f){=}\eta^* \end{cases}$$

□

En d'autres termes, l'interprétation canonique des algèbres des termes avec variables $A_t \in \mathsf{TAlg}(\Sigma{+}V)$ vers A coïncide avec l'interprétation canonique, au sens de la définition 3.3.5, de A_t vers l'algèbre A' sur la signature $\Sigma{+}V$, obtenue en ajoutant à l'algèbre A autant de constantes $X^{A'}$ que de variables $X \in V$, et dont la valeur est définie par l'environnement ξ :

$$X^{A'}() = \xi(X)$$

La donnée d'une interprétation implicite (A, ξ) ou d'une algèbre A' contenant A et sur la signature $\Sigma{+}V$, représentent deux façons équivalentes de définir l'interprétation canonique des termes avec variables.

L'interprétation d'un terme fini avec variables dans une algèbre A peut aussi être définie par une simple composition fonctionnelle : en premier lieu on éliminera les variables par l'application d'une substitution le permettant (ground), et en second lieu on utilisera l'interprétation canonique des termes finis (sans variables) dans l'algèbre sémantique A :

$$[\![t]\!]_{A,\sigma} = [\![t \cdot \sigma]\!]_A$$

Ce type d'interprétation correspond à la donnée d'une interprétation implicite (T, σ), où $T = \mathrm{T}(\Sigma)$ est l'algèbre des termes sans variables, et σ est un environnement dans T, c'est-à-dire une substitution ground. L'interprétation vers l'algèbre A s'obtient alors par l'interprétation vers les termes finis $\mathrm{T}(\Sigma)$, composée avec l'interprétation canonique des termes finis vers A.

3.3.5. Élagages et préfixes de termes.

Les termes héritent aussi les définitions de *sous-arbre* (nous dirons *sous-terme*), d'*élagage* et de *préfixe*. Si les sous-termes respectent forcément la signature, et il est simple de le vérifier, les élagages et les préfixes peuvent, en revanche, sortir de la classe des termes.

DEFINITION 3.3.17. (ÉLAGAGES CORRECTS) *Étant donné une signature* $\Sigma = (\mathcal{B}, S, F, T)$ *et un terme* $t \in \mathrm{IT}(\Sigma, S)$*, nous disons qu'un élagage* $t' \in \mathrm{IT}(F)$ *de* t *est* correct vis-à-vis de Σ*, s'il est complet à toutes les adresses de* $\mathrm{dom}(t')$ *d'arité fixe :*

$$\forall u \in \mathrm{dom}(t').\quad t(u) = f \ \wedge\ \mathrm{dom}_T(f) = \eta_1 \times \ldots \times \eta_n \quad \Rightarrow\ \mathrm{Succ}(t', u) = \mathrm{Succ}(t, u)$$

□

PROPOSITION 3.3.18. *Un élagage d'un terme est correct si et seulement si il est un terme.*

DÉMONSTRATION. *($\Rightarrow$) Soit* $t' \in \mathrm{IT}(F)$ *un élagage correct de* $t \in \mathrm{IT}(\Sigma, S)$ *vis-à-vis de* $\Sigma = (\mathcal{B}, S, F, T)$. *Nous devons prouver que* t' *respecte, lui aussi, les conditions de la définition 3.3.3. Soit* $u \in \mathrm{dom}(t')$. *Il y a deux cas :*

(1) *Le cas* $(u, f) \in \mathrm{Feuilles}(t')$. *Donc* $(u, f) \in \mathrm{Feuilles}(t)$. *Puisque* $t' \subseteq t$, *nous avons, d'une part,* $t'(u) = t(u)$ *et, de l'autre,* t *est un terme. Donc* $t'(u) = f$ *est un symbole de* Σ *d'arité fixe nulle ou d'arité variable.*

(2) *Le cas* $(u, f) \in \mathrm{Noeuds}(t')$ *et* $\mathrm{Succ}^{(t,u)} = (i_1, f_1) \cdots (i_m, f_m)$. *Soit* $n = \mathrm{degré}(t, u)$. *Deux cas se présentent :*

   (a) $t'(u)$ *est un symbole de domaine* $\eta_1 \times \ldots \eta_n$ *et codomaine* ϑ. *L'arbre* t' *étant complet aux adresses d'arité fixe nous avons* $m = n$, *et puisque* $t(u) = t'(u)$ *et que* t *est un terme, nous avons que pour tout* $j = 1 \ldots n$ *le symbole* f_j *a comme codomaine* ϑ.

   (b) $t'(u)$ *est un symbole de domaine* η^* *et codomaine* ϑ. *Il se peut que* $m < n$ *mais, de toute façon,* t *étant un terme, nous avons que pour tout* $j = 1 \ldots m$, *le symbole* $f_j = t'(u.i_j) = t(u.i_j)$ *a comme codomaine* ϑ.

($\Leftarrow$) Si un élagage d'un terme est un terme, par définition même de terme, il ne pourra pas être partiel aux adresses d'arité fixe. $\square$

COROLLARY 3.3.19. *Tout préfixe radical seulement aux adresses d'arité variable est un terme.*

Pour ne pas sortir de la classe des termes, tout en construisant un préfixe, il faudra donc ébrancher (radicalement) seulement les adresses portant sur un symbole d'arité variable.

Le domaine des termes équipé des relations (restreintes aux termes) branche, élagage et préfixe, définies sur les arbres à branchement fini $\mathrm{BT}(F)$, constitue respectivement les structures $(\mathrm{IT}(\Sigma, S), \ll_{\mathrm{IT}(\Sigma,S)})$, $(\mathrm{IT}(\Sigma, S), \subseteq_{\mathrm{IT}(\Sigma,S)})$ et $(\mathrm{IT}(\Sigma, S), \leq_{\mathrm{IT}(\Sigma,S)})$. Étant des restrictions, ces structures conservent au moins les propriétés réflexive, transitive et anti-symétrique (remarque 2.6.3, page 67). Donc la première est un préordre, et les deux autres sont (au moins) des ordres partiels. Le contre-exemple prouvant la non complétude par dirigés des arborescences à branchement fini pour l'ordre d'élagage, peut être adapté au cas des termes : soit $t : \mathbb{N} \to \mathrm{IT}(\Sigma, S)$ telle que $t(x) \triangleq \{(\varepsilon, f)\} \cup \{(i, f) \in \mathbb{N} \times F \mid i \leq_{\mathbb{N}} x\}$ où f est un symbole d'arité variable dans $\Sigma = (\mathcal{B}, S, F, T)$. Pour tout $n \in \mathbb{N}$, $t(n)$ est un terme de Σ de hauteur 1 avec un degré de branchement égal à n à la racine. Toutefois, aucun majorant d'un tel dirigé ne pourra être à branchement fini, donc un terme. Cependant, les propriétés des ordres d'élagage et préfixe sont les mêmes que celle du domaine des arbres à branchement fini.

PROPOSITION 3.3.20. *Le couple* $(\mathrm{IT}(\Sigma, S), \subseteq_{\mathrm{IT}(\Sigma,S)})$ *est un co-dcpo d'ensembles.*

DÉMONSTRATION. *Par la remarque 2.6.6, il nous suffit de prouver que l'intersection des termes d'un co-dirigé de* $(\mathrm{IT}(\Sigma, S), \subseteq_{\mathrm{IT}(\Sigma,S)})$ *est encore un terme. Soit* $I \triangleq \bigcap_{t \in \nabla} t$ *où* $\nabla \in \nabla(\mathrm{IT}(\Sigma, S), \subseteq_{\mathrm{IT}(\Sigma,S)})$. *Il suffit de montrer que* I *est un élagage correct de tout* t_1 *choisi arbitrairement dans* ∇. *Si elle ne l'était pas, elle serait partielle, vis-à-vis de* t_1, *à une certaine adresse* $u \in \mathrm{dom}(I)$ *d'arité fixe.* I *étant l'intersection du co-dirigé, il existerait un* t_2 *n'ayant pas, lui non plus, les branches de* t_1 *manquantes dans* I. *Par définition de co-dirigé, il existerait un* t_3 *tel que* $t_3 \subseteq_{\mathrm{IT}(\Sigma,S)} t_1$ *et* $t_3 \subseteq_{\mathrm{IT}(\Sigma,S)} t_2$. *Alors* t_3 *serait correct par rapport à* t_2 *et donc complet à l'adresse* u *en contradiction avec la condition* $t_3 \subseteq_{\mathrm{IT}(\Sigma,S)} t_2$. □

PROPOSITION 3.3.21. *Le couple* $(\mathrm{IT}(\Sigma, S), \leq_{\mathrm{IT}(\Sigma,S)})$ *est un bi-dcpo d'ensembles.*

DÉMONSTRATION. *Il faut prouver qu'il s'agit d'un dcpo (1) et d'un co-dcpo (2) d'ensembles.*

(1) $(\mathrm{IT}(\Sigma, S), \leq_{\mathrm{IT}(\Sigma,S)})$ *dcpo d'ensembles.*
Il suffit de prouver que l'union des termes d'un dirigé de $(\mathrm{IT}(\Sigma, S), \leq_{\mathrm{IT}(\Sigma,S)}$ $)$ *est encore un terme. Soit* $U \triangleq_{t \in \Delta} t$ *où* $\Delta \in \Delta(\mathrm{IT}(\Sigma, S), \leq_{\mathrm{IT}(\Sigma,S)})$. *Tout élément du dirigé est, à la fois, un terme et un préfixe de* U. *Supposons, par l'absurde, que* U *ne soit pas un terme. Puisque* $(\mathrm{BT}(F), \leq_{\mathrm{BT}(F)})$ *est un dcpo d'ensembles, l'arbre* U *est à branchement fini. Si* $(u, f)\mathrm{Feuilles}(U)$, *alors il existe un élément du dirigé qui contient* (u, f) *ce qui garantit la condition de terme. Si* $(u, f) \in \mathrm{Noeuds}(U)$, *alors il existe un nombre fini de successeurs de* u *dans* U : $\mathrm{Succ}^{(U,u)} = (i_1, f_1) \cdots (i_m, f_m)$. *Si* f *est un symbole d'arité variable* $\mathrm{dom}_T(f) = \eta^*$ *alors les* f_i *ont le codomaine* η *puisque* U *est l'union de termes. Si* f *est un symbole d'arité fixe* $\mathrm{dom}_T(f) = \eta_1 \times \ldots \times \eta_n$, *alors tout élément* t *du dirigé tel que* $(u, f) \in \mathrm{Noeuds}(t)$ *doit forcément avoir les mêmes successeurs de* U. *Donc* $\forall j = 1 \ldots n$, *le symbole* f_j *a comme codomaine* η_j *dans* Σ. *Donc* U *respecte, à toute adresse, la condition de terme.*

(2) $(\mathrm{IT}(\Sigma, S), \leq_{\mathrm{IT}(\Sigma,S)})$ *co-dcpo d'ensembles.*
Un co-dirigé au sens du préfixe est (par inclusion logique) aussi un co-dirigé au sens de l'élagage. La proposition 3.3.20 nous assure alors que l'intersection d'un tel co-dirigé est un terme.

□

3.4. Récapitulatif

Les tables 1 et 2 résument respectivement l'algèbre des arborescences et celle des arbres par leurs signatures.

Nous avons pu constater, tout au long de ce chapitre, que les opérations définies d'abord sur les arborescences on été ensuite étendues aux arbres en respectant la règle suivante : oubliant les étiquettes, l'opération sur les arbres coïncide avec son homologue sur les arborescences. En d'autre termes, nous avons construit une

TAB. 1. Signature des arborescences

(IT			,	*arborescences*
T			,	*arborescences finies*
RT			,	*arborescences rationnelles*
BT			;	*arboresc. à branchement fini*
Feuilles	:	$\mathrm{IT} \to \wp^{\mathbb{N}_+^*}$	,	*ensemble des feuilles*
Noeuds	:	$\mathrm{IT} \to \wp^{\mathbb{N}_+^*}$	,	*ensemble des noeuds*
hauteur	:	$\forall\alpha{:}\mathsf{sort}.\ \alpha \to \mathsf{if}\ \alpha = \mathrm{T}\ \mathsf{then}\ \mathbb{N}\ \mathsf{else}\ \mathbb{N}_\infty$	,	*hauteur*
Front	:	$2^{\wp(\mathbb{N}_+^*)}$	,	*prédicat frontière*
degré	:	$\forall\alpha{:}\mathsf{sort}.\ \alpha \times \mathbb{N}_+^* \to \mathsf{if}\ \alpha = \mathrm{IT}\ \mathsf{then}\ \mathbb{N}_\infty\ \mathsf{else}\ \mathbb{N}$	,	*degré de branchement*
Succ	:	$\mathrm{IT} \times \mathbb{N}_+^* \to \wp^{\mathbb{N}_+^*}$	,	*ensemble des successeurs*
$\mathrm{Succ}^{(.,.)}$	:	$\mathrm{IT} \times \mathbb{N}_+^* \to (\mathbb{N}_+^*)^\infty$	,	*suite des successeurs*
$(.)\|_{(.)}$	:	$\forall\alpha{:}\mathsf{sort}.\ \alpha \times \mathbb{N}_+^* \to \alpha$	,	*branche d'une arborescence*
Filles	:	$\forall\alpha{:}\mathsf{sort}.\ \alpha \to \mathsf{if}\ \alpha = \mathrm{IT}\ \mathsf{then}\ \alpha^\infty\ \mathsf{else}\ \alpha^*$	,	*filles d'une arborescence*
$(.)\cdot_{(.)}(.)$	:	$\mathbb{N}_+^* \to (\mathrm{IT} \times \mathrm{IT}) \to \wp^{\mathbb{N}_+^*}$	,	*greffe à une adresse*
		$\oplus\ \wp^{\mathbb{N}_+^*} \to (\mathrm{IT} \times \mathrm{IT}) \to \wp^{\mathbb{N}_+^*}$		*greffe à plusieurs adresses*
		$\oplus\ \wp^{\mathbb{N}_+^*} \to \mathrm{IT} \times (\mathbb{N}_+^* \to \mathrm{IT}) \to \wp^{\mathbb{N}_+^*}$		*greffe en fonct. de l'adresse*
$(.).(.)$	:	$\mathbb{N}_+^* \times \mathrm{IT} \to \wp^{\mathbb{N}_+^*}$	,	*translation à une adresse*
$\preceq$	:	$\forall\alpha{:}\mathsf{sort}.\ 1^{\alpha\times\alpha}$	,	*semi-prédicat branche*
$\prec$	:	$\forall\alpha{:}\mathsf{sort}.\ 1^{\alpha\times\alpha}$	,	*semi-prédicat branche stricte*
$\subseteq$	:	$\forall\alpha{:}\mathsf{sort}.\ 1^{\alpha\times\alpha}$	,	*semi-prédicat élagage*
$\leq$	:	$\forall\alpha{:}\mathsf{sort}.\ 1^{\alpha\times\alpha}$	,	*semi-prédicat préfixe*
$\sqsubseteq$	:	$\forall\alpha{:}\mathsf{sort}.\ 1^{\alpha\times\alpha}$	)	*semi-prédicat préfixe régulier*

extension homomorphe (cf. section 1.4.2) de l'algèbre des arborescences. D'une façon rigoureuse, il faudrait considérer, dans l'algèbre des arbres, le domaine $\mathbb{N}_+^* \times \mathsf{F}$ comme une sorte correspondante à la sorte $\mathbb{N}_+^*$ des arborescences. L'homomorphisme (surjectif) en question serait alors l'application φ associant à tout arbre son arborescence (i.e. son domaine) et aux couples $(u, e) \in \mathbb{N}_+^* \times \mathsf{F}$ leur première composante :

$$\varphi : \mathrm{IT}(\mathsf{F}) \to \mathrm{IT} \ \oplus\ (\mathbb{N}_+^* \times \mathsf{F}) \to \mathbb{N}_+^*, \qquad \begin{cases} \varphi(t) = \mathrm{dom}\, t & \text{si } t \in \mathrm{IT}(\mathsf{F}) \\ \varphi(u, e) = u & \text{si } (u, e) \in \mathbb{N}_+^* \times \mathsf{F} \end{cases}$$

TAB. 2. Signature des arbres

$(\mathsf{F}$			;	*étiquettes (domaine de base)*
$\mathrm{IT}(\mathsf{F})$			,	*arbres*
$\mathrm{T}(\mathsf{F})$			,	*arbres finis*
$\mathrm{RT}(\mathsf{F})$			,	*arbres rationnels*
$\mathrm{BT}(\mathsf{F})$			;	*arbres à branchement fini*
Feuilles	:	$\mathrm{IT}(\mathsf{F}) \to \wp^{\mathbb{N}_+^* \times \mathsf{F}}$	,	*ensemble des feuilles*
Noeuds	:	$\mathrm{IT}(\mathsf{F}) \to \wp^{\mathbb{N}_+^* \times \mathsf{F}}$	,	*ensemble des noeuds*
hauteur	:	$\forall \alpha{:}\mathsf{sort}.\ \alpha \to \mathsf{if}\ \alpha = \mathrm{T}(\mathsf{F})\ \mathsf{then}\ \mathbb{N}\ \mathsf{else}\ \mathbb{N}_\infty$	,	*hauteur*
Front	:	$2^{\wp^{\mathbb{N}_+^* \times \mathsf{F}}}$	,	*prédicat frontière*
degré	:	$\forall \alpha{:}\mathsf{sort}.\ \alpha \times \mathbb{N}_+^* \to \mathsf{if}\ \alpha = \mathrm{IT}(\mathsf{F})\ \mathsf{then}\ \mathbb{N}_\infty\ \mathsf{else}\ \mathbb{N}$	,	*degré de branchement*
Succ	:	$\mathrm{IT}(\mathsf{F}) \times \mathbb{N}_+^* \to \wp^{\mathbb{N}_+^* \times \mathsf{F}}$	,	*ensemble des successeurs*
$\mathrm{Succ}^{(.,.)}$	:	$\mathrm{IT}(\mathsf{F}) \times \mathbb{N}_+^* \to (\mathbb{N}_+^* \times \mathsf{F})^\infty$	,	*suite des successeurs*
$(.)\vert_{(.)}$	:	$\forall \alpha{:}\mathsf{sort}.\ \alpha \times \mathbb{N}_+^* \to \alpha$	,	*branche d'un arbre*
Fils	:	$\forall \alpha{:}\mathsf{sort}.\ \alpha \to \mathsf{if}\ \alpha = \mathrm{IT}(\mathsf{F})\ \mathsf{then}\ \alpha^\infty\ \mathsf{else}\ \alpha^*$	,	*fils d'un arbre*
$(.) \cdot_{(.)} (.)$	:	$\mathbb{N}_+^* \to \mathrm{IT}(\mathsf{F}) \times \mathrm{IT}(\mathsf{F}) \to \wp^{\mathbb{N}_+^* \times \mathsf{F}}$	,	*greffe à une adresse*
		$\oplus\ \wp^{\mathbb{N}_+^*} \to \mathrm{IT}(\mathsf{F}) \times \mathrm{IT}(\mathsf{F}) \to \wp^{\mathbb{N}_+^* \times \mathsf{F}}$		*greffe à plusieurs adresses*
		$\oplus\ \wp^{\mathbb{N}_+^*} \to \mathrm{IT}(\mathsf{F}) \times (\mathbb{N}_+^* \to \mathrm{IT}(\mathsf{F})) \to \wp^{\mathbb{N}_+^* \times \mathsf{F}}$		*greffe en fonct. de l'adresse*
$(.).(.)$	:	$\mathbb{N}_+^* \times \mathrm{IT}(\mathsf{F}) \to \wp^{\mathbb{N}_+^* \times \mathsf{F}}$	,	*translation à une adresse*
$\preceq$	:	$\forall \alpha{:}\mathsf{sort}.\ 1^{\alpha \times \alpha}$	,	*semi-prédicat branche*
$\prec$	:	$\forall \alpha{:}\mathsf{sort}.\ 1^{\alpha \times \alpha}$	,	*semi-prédicat branche stricte*
$\subseteq$	:	$\forall \alpha{:}\mathsf{sort}.\ 1^{\alpha \times \alpha}$	,	*semi-prédicat élagage*
$\leq$	:	$\forall \alpha{:}\mathsf{sort}.\ 1^{\alpha \times \alpha}$	,	*semi-prédicat préfixe*
$\sqsubseteq$	:	$\forall \alpha{:}\mathsf{sort}.\ 1^{\alpha \times \alpha}$	)	*semi-prédicat préfixe régulier*

Les propriétés de complétude des relations définies dans le chapitre sont résumées, pour toutes les combinaisons *domaine-relation*, dans la Table 3.

TAB. 3. Propriétés des structures d'ordre définies

	branche $\ll$	élagage $\subseteq$	préfixe $\leq$
Arborescences IT	$\preceq_{\mathrm{IT}}$ préordre	$\subseteq_{\mathrm{IT}}$ treillis complet d'ensembles $\bot = \emptyset \quad \top = \mathbb{N}_+^*$	$\leq_{\mathrm{IT}}$ bi-dcpo d'ensembles $\bot = \emptyset$
Arborescences à branchement fini BT	$\ll_{\mathrm{BT}}$ préordre	$\subseteq_{\mathrm{BT}}$ co-dcpo d'ensembles $\bot = \emptyset$	$\leq_{\mathrm{BT}}$ bi-dcpo d'ensembles $\bot = \emptyset$
Arbres $\mathrm{IT}(\mathsf{F})$	$\preceq_{\mathrm{IT}(\mathsf{F})}$ préordre	$\subseteq_{\mathrm{IT}(\mathsf{F})}$ bi-dcpo d'ensembles $\bot = \emptyset$	$\leq_{\mathrm{IT}(\mathsf{F})}$ bi-dcpo d'ensembles $\bot = \emptyset$
Arbres $\mathrm{BT}(\mathsf{F})$	$\ll_{\mathrm{BT}(\mathsf{F})}$ préordre	$\subseteq_{\mathrm{BT}(\mathsf{F})}$ co-dcpo d'ensembles $\bot = \emptyset$	$\leq_{\mathrm{BT}(\mathsf{F})}$ bi-dcpo d'ensembles $\bot = \emptyset$
Termes $\mathrm{BT}(\mathsf{F})$	$\ll_{\mathrm{IT}(\Sigma,\mathrm{S})}$ préordre	$\subseteq_{\mathrm{IT}(\Sigma,\mathrm{S})}$ co-dcpo d'ensembles $\bot = \emptyset$	$\leq_{\mathrm{IT}(\Sigma,\mathrm{S})}$ bi-dcpo d'ensembles $\bot = \emptyset$

CHAPITRE 4

Syntaxe des jeux à deux joueurs

La notion de jeu en théorie des jeux combinatoires est celle, très générale, à laquelle nous sommes habitués dans le contexte des jeux de hasard. À partir d'une position initiale, il y a une suite de coups effectués par les joueurs, chaque joueur choisissant parmi plusieurs possibilités ; à certain moment dans cette suite peuvent intervenir des coups de hasard (ou aléatoires), tels que le lancer d'un dé ou le mélange d'un paquet de cartes. Des exemples de tels jeux sont les *Échecs*, où il n'y a pas de coups aléatoires, le *Bridge*, où le hasard joue un bien plus grand rôle, et la *Roulette*, qui n'est qu'un jeu de chance où la stratégie est inexistante. Nous allons nous occuper des jeux du premier type, comme les *Échecs*, où le jeu s'explique par les choix rationnels des joueurs et par la convergence ou divergence de leurs objectifs.

De ces jeux, nous allons distinguer deux aspects : le premier qu'on appellera *syntaxique*, porte sur la façon dont le jeu peut déployer tous ses chemins possibles ou parties (matches), le second qu'on dira *sémantique*, concerne les façons d'interpréter le jeu selon les intérêts des joueurs.

La définition de jeu à n personnes, proposée par von Neumann dans son article fondateur [**von Neumann, 1928**], et largement acceptée depuis, met l'accent sur la structure arborescente d'un jeu : un jeu est fondamentalement l'arbre des possibilités de jeu, c'est-à-dire l'arbre de toutes les parties possibles. Cette présentation n'est pas très économique : la taille des arbres croît très vite dans tous les jeux réels. C'est la raison pour laquelle, pour décrire un jeu, on donne plus souvent la position initiale (ou les positions initiales) et les règles du jeu, qui permettent de passer d'une position à l'autre. Cette discussion rappelle la définition des langages informatiques indirectement définis par des grammaires formelles, tout comme la définition des sémantiques opérationnelles structurées de langages non-déterministes ou concurrents. Dans un calcul de processus, par exemple, la sémantique est exprimée par un nombre fini de règles logiques qui décrivent les évolutions possibles de l'état d'un système, tandis que ce qui nous importe vraiment est l'arbre de dérivation global d'un processus, c'est-à-dire l'arbre de toutes les évolutions possibles à partir de son état initial.

Les notions présentes dans ce chapitre sont en partie anciennes (*arbres de jeu*, *stratégies*), et en partie nouvelles (*préfixes du jeu*, *élagages du joueur et de l'opposant*, *algèbres syntaxiques*). Lorsque elles sont anciennes, elles sont redéfinies et réadaptées au cadre des jeux arbitraires, où les arbres de jeux peuvent être finis ou infinis, et les propriétés de ces nouvelles versions sont prouvées de façon formelle.

4.1. Arènes de jeu

À la différence de von Neumann, nous allons définir un jeu, ou plus précisément, la syntaxe d'un jeu, comme une structure mathématique définissant indirectement un ou plusieurs arbres de jeu. L'approche est donc celle des travaux sur la sémantique des jeux pour la logique linéaire (voir par exemple [**Blass, 1992**] et [**Danos & Harmer, 2000**]).

DEFINITION 4.1.1. (SYNTAXE D'UN JEU COMBINATOIRE À DEUX JOUEURS) *La* syntaxe *(ou* arène*) d'un jeu combinatoire à deux joueurs (*syntaxe de jeu *ou* arène de jeu *par la suite) est un quadruplet* $\mathcal{S} = (\mathbb{P}, \lambda, \rightarrow, \mathbb{P}_I)$ *où :*

- $\mathbb{P}$ *est l'ensemble des positions du jeu*
- $\lambda : \mathbb{P} \rightarrow \{\text{Player}, \text{Opponent}\}$ *est la fonction qui classifie les positions en celles du* joueur *(*Player*) et celles de l'*opposant *(*Opponent*)*
- $\rightarrow : (\mathbb{P} \times \mathbb{N}_+) \multimap \mathbb{P}$ *est la relation de transition représentant les* coups *(ou* mouvements*) possibles pour toute position du jeu. L'écriture* $\pi \xrightarrow{\ell} \pi'$ *signifie* $((\pi, \ell), \pi') \in \rightarrow$*, et indique que dans la position* π *un mouvement possible, étiqueté par* $\ell \in \mathbb{N}_+$*, conduit le jeu dans la nouvelle position* π'*. On demande à cette relation d'être* localement finie*, i.e. pas plus qu'un nombre fini de coups possibles dans toutes positions du jeu :*
$$\forall \pi \in \mathbb{P} \ \exists n \in \mathbb{N}_+ \ \text{ s.t. } \pi \xrightarrow{\ell} \pi' \Rightarrow \ell \leq n$$
- $\mathbb{P}_I \subseteq \mathbb{P}$ *est l'ensemble des positions initiales du jeu, c'est-à-dire celles à partir desquelles le jeu peut démarrer.*

□

Si la numérotation des coups, comme pour les arbres (déf. 3.1.1, page 75), n'est pas nécessairement continue (il peut y avoir, par exemple, trois coups numérotés $1, 4, 9$), elle suffit pour les ranger dans l'ordre habituel des entiers naturels. Nous pourrons dire avoir un premier coup, un deuxième, un troisième et ainsi de suite. Quand un tel ordre n'aura pas d'importance, nous pourrons faire abstraction de l'étiquette entière, et considérer la relation de transition entre positions comme étant de type $\wp^{\mathbb{P}\times\mathbb{P}}$. Nous noterons cette autre relation, avec un abus de notation, encore avec le symbole $\rightarrow$:

$$\pi \rightarrow \pi' \overset{\triangle}{\iff} \exists i \in \mathbb{N}_+ .\ \pi \overset{i}{\rightarrow} \pi'$$

NOTATION 4.1.2. Nous écrirons $\pi \overset{\ell_i}{\longrightarrow} \pi'$ pour indiquer la disponibilité d'un i-ème coup possédant l'étiquette ℓ_i. Dans l'exemple précédent, le 3-ème coup disponible sera donc celui ayant l'étiquette 9. Par un schéma du genre :

$$\pi \begin{array}{l} \nearrow \pi_1 \\ \dashrightarrow \mid \\ \searrow \pi_n \end{array}$$

nous représenterons une position ayant n coups disponibles, numérotés d'une quelconque façon, où le i-ème coup disponible conduit à la position du jeu π_i. En d'autres termes, nous considérerons implicitement que la numérotation des positions d'arrivée correspond à l'ordre des coups impliqués.

Le *Morpion*[1] est, par sa simplicité, un jeu qui permet de bien illustrer la définition 4.1.1.

EXAMPLE 4.1.3. **Arène du Morpion.** Dans ce jeu très populaire, les deux joueurs écrivent, tour à tour, sur les cases d'un damier de 3×3 cellules, leurs symboles associés, tels que $\times$ et $\circ$.

$\times$	$\circ$	
$\times$	$\times$	$\circ$
$\circ$		$\times$

Au départ le damier est vide et le but du jeu est celui de cocher en premier, de son propre symbole, trois cases consécutives, à l'horizontale, à la verticale ou à la diagonale. La syntaxe du jeu peut être décrite formellement comme un quadruplet $\mathcal{S} = (\mathbb{P}, \lambda, \rightarrow, \mathbb{P}_I)$. L'ensemble des positions est le produit cartésien $\mathbb{P} = \text{Initiative} \times \text{Damier}$. Les positions sont donc des couples où la première composante indique qui des deux joueurs a la main : $\text{Initiative} = \{\times, \circ\}$, et la seconde représente l'état du damier : $\text{Damier} = \{1, 2, 3, 4, 5, 6, 7, 8, 9\} \rightarrow \{\text{vide}, \times, \circ\}$, considérant la numérotation des cases suivantes :

1	2	3
4	5	6
7	8	9

Pour ce jeu nous associons le *joueur* et l'*opposant* respectivement aux signes $\times, \circ$. La fonction $\lambda : \mathbb{P} \rightarrow \{\text{Player}, \text{Opponent}\}$ de classification des positions est donc définie par : $\lambda(\times, d) = \text{Player}$ et $\lambda(\circ, d) = \text{Opponent}$ pour tout $d \in \text{Damier}$. La relation exprimant les mouvements possibles $\rightarrow \subseteq \mathbb{P} \times \mathbb{N}_+ \times \mathbb{P}$ est définie comme la plus petite relation respectant les règles suivantes :

[1] Parfois appelé *tic-tac-toe*

$$\frac{\mathrm{d}(i) = \text{vide} \quad \neg\text{final}(\mathrm{d})}{(\times, \mathrm{d}) \xrightarrow{i} (\circ, \mathrm{d}[i \mapsto \times])} \qquad \frac{\mathrm{d}(i) = \text{vide} \quad \neg\text{final}(\mathrm{d})}{(\circ, \mathrm{d}) \xrightarrow{i} (\times, \mathrm{d}[i \mapsto \circ])}$$

Où le prédicat final(d) est vrai lorsque trois cases consécutives (les *directions*) ont été remplies par le même joueur. Formellement, l'ensemble des directions et le prédicat sont définis de la façon suivante :

$$\begin{array}{rcl} \text{Directions} & \triangleq & \{(1,2,3),(4,5,6),(7,8,9),(1,4,7),(2,5,8),(3,6,9),(1,5,9),(3,5,7)\} \\ \text{final}(\mathrm{d}) & \stackrel{\triangle}{\Longleftrightarrow} & \exists \vec{x} \in \text{Directions}.\ \ \mathrm{d}(x_1) = \mathrm{d}(x_2) = \mathrm{d}(x_3) \ \wedge\ \mathrm{d}(x_1) \neq \text{vide} \end{array}$$

Enfin, l'ensemble des positions initiales :

$$\mathbb{P}_{\mathrm{I}} = \{(\times, \mathrm{d}_v) \in \mathbb{P} \mid \mathrm{d}_v = \{(i, \text{vide}) \mid 1 \leq i \leq 9\}\}$$

modélise la situation où, devant le damier vide, c'est le joueur associé au signe $\times$, donc le Player, qui doit prendre l'initiative et débuter la partie. □

4.1.1. Dualité. Dans la théorie des jeux il arrive souvent qu'on observe et discute un jeu du point de vue d'un des joueurs. Ceci est aussi le cas, lors de la définition d'un jeu combinatoire à deux joueurs, quand on décide qui est le joueur qu'on appellera *Player*, et celui qu'on appellera *Opponent*. Bien entendu, cet arbitre laisse entrevoir le choix opposé, qui aurait lieu en échangeant le nom des joueurs. La dualité des jeux à deux joueurs se retrouve aussi dans leur interprétation sémantique : le plus souvent l'intérêt des joueurs est, là aussi, *opposé* ou, autrement dit, *dual*.

DEFINITION 4.1.4. *Étant donné une arène de jeu* $\mathcal{S} = (\mathbb{P}, \lambda, \rightarrow, \mathbb{P}_{\mathrm{I}})$*, nous définissons la* duale de $\mathcal{S}$*, noté* $\mathcal{S}^{\perp}$*, comme la syntaxe de jeu* $\mathcal{S}^{\perp} = (\mathbb{P}, \lambda^{\perp}, \rightarrow, \mathbb{P}_{\mathrm{I}})$*, où la fonction de classification des coups* $\lambda^{\perp} : \mathbb{P} \rightarrow \{\text{Player}, \text{Opponent}\}$ *est l'inverse de* λ *:*

$$\lambda^{\perp}(x) = \text{Player} \quad \Leftrightarrow \quad \lambda(x) = \text{Opponent}$$

□

La syntaxe duale est donc, tout simplement, un renommage des joueurs. Dans l'exemple 4.1.3 du Morpion, la duale est celle qui associe le *joueur* au symbole $\circ$, et l'*opposant* au symbole $\times$. Dans l'arène duale, c'est donc l'*opposant* qui débute la partie. On peut remarquer que la duale de la duale d'une arène est égale à l'arène originale : $\left(\mathcal{S}^{\perp}\right)^{\perp} = \mathcal{S}$.

4.2. Arbres de jeu

Nous allons, à présent, définir formellement plusieurs notions faisant partie du vocabulaire de la théorie des jeux combinatoires. L'arène d'un jeu est une structure mathématique qui, une fois définie, exprime un certain nombre de notions subordonnées comme celle, fondamentale, d'*arbres de jeu.* Dans ces arbres, les noeuds sont décorés par les positions du jeu tandis que la relation $\rightarrow$ établit les arcs entre ces noeuds. A toute position du jeu sera associé un arbre, certes, mais ce que nous appellerons précisément *arbres légaux* du jeu seront seulement ceux décorés à la racine par une position initiale, de l'ensemble $\mathbb{P}_{\mathrm{I}}$. L'hypothèse de la définition 4.1.1, que la relation $\rightarrow$ soit localement finie, implique que les arbres de jeu seront à branchement fini.

DEFINITION 4.2.1. (ARBRE D'UNE POSITION) *Étant donné une arène de jeu* $\mathcal{S} = (\mathbb{P}, \lambda, \rightarrow, \mathbb{P}_{\mathrm{I}})$*, et un arbre* $t \in \mathrm{IT}(\mathbb{P})$ *étiqueté dans* $\mathbb{P}$*, le* déroulement élémentaire de t selon $\rightarrow$ *(ou* selon le jeu $\mathcal{S}$*), est l'arbre :*

$$\mathrm{step}_{\rightarrow}(t) \triangleq t \cup \{(u.\ell, \pi') \mid (u, \pi) \in \mathrm{Feuilles}(t),\ \pi \xrightarrow{\ell} \pi'\}.$$

Soit $\pi \in \mathbb{P}$ *une position du jeu. L'*arbre de la position π*, noté* $t_\pi^{\mathcal{S}}$ *(ou plus simplement* t_π *lorsque la syntaxe* $\mathcal{S}$ *est implicite), est la borne supérieure, au sens de l'ordre préfixe* $\leq$ *sur* $\mathrm{IT}(\mathbb{P})$*, de la suite croissante :*

$$\begin{cases} t_\pi^0 & \triangleq \ \{(\varepsilon, \pi)\} \\ t_\pi^{k+1} & \triangleq \ \mathrm{step}_{\rightarrow}(t_\pi^k) \end{cases}$$

$t_\pi \triangleq \sup_{\leq}\{t_\pi^k \mid k \in \mathbb{N}\}$ □

Intuitivement, l'arbre d'une position π est le déroulement limite, selon les règles du jeu S, de l'arbre au départ constitué de la seule racine (ε, π).

PROPOSITION 4.2.2. *La définition d'arbre d'une position* t_π *est correcte et il s'agit d'un arbre à branchement fini.*

DÉMONSTRATION. *L'opération* $\mathrm{step}_{\rightarrow}(t)$ *rallonge l'arbre* t *en ajoutant des couples* $(u.\ell, \pi')$ *tels que* $(u, \pi) \in \mathrm{Feuilles}(t)$*, où* $\ell \in \mathbb{N}_+$*. Donc le domaine du résultat d'une telle opération sera, comme celui de* t*, clos par préfixe, c'est-à-dire une arborescence. De plus, puisque la relation qui exprime les mouvements du jeu est une fonction partielle :* $\rightarrow : (\mathbb{P} \times \mathbb{N}_+) \rightharpoonup \mathbb{P}$*, l'opération* $\mathrm{step}_{\rightarrow}(t)$ *ne pourra pas ajouter deux couples* $(u.\ell, \pi_1)$ *et* $(u.\ell, \pi_2)$ *partageant la même adresse. Autrement dit, nous aurons forcément* $\pi_1 \neq \pi_2$*. Ainsi, le déroulement d'un arbre est encore une fonction partielle avec une arborescence comme domaine, c'est-à-dire un arbre. L'opération effectuée est donc une fonction de type* $\mathrm{step} : \mathrm{IT}(\mathbb{P}) \rightarrow \mathrm{IT}(\mathbb{P})$*. Par définition de déroulement, l'arbre* t *est un préfixe de* $\mathrm{step}_{\rightarrow}(t)$ *:* t *est complet ou radical vis-à-vis de* $\mathrm{step}_{\rightarrow}(t)$ *à toutes les adresses de* $\mathrm{dom}(t)$*. Donc* $(t_\pi^k)_{k \in \mathbb{N}}$ *est effectivement une suite croissante. Enfin, considérant que la structure* $(\mathrm{IT}(\mathbb{P}), \leq_{\mathrm{IT}(\mathbb{P})})$ *est complète par dirigé (cf. prop. 3.2.15), la définition 4.2.3 est correcte. Puisque la relation de transition du jeu* $\rightarrow$ *est localement finie, l'opération de déroulement élémentaire ne pourra pas ajouter un nombre infini de branches* *aux feuilles de son argument. Donc, restreinte au domaine des arbres à branchement fini, l'opération* step *est de type* $\mathrm{BT}(\mathbb{P}) \to \mathrm{BT}(\mathbb{P})$. *Puisqu'au départ l'arbre* t^0_π *est à branchement fini, la suite* $(\mathrm{t}^k_\pi)_{k\in\mathbb{N}}$ *sera croissante (donc un dirigé) dans la structure* $(\mathrm{BT}(\mathbb{P}), \leq_{\mathrm{BT}(\mathbb{P})})$ *qui est un dcpo d'ensembles (cf. section 3.2.10). Ainsi, l'arbre* t_π *d'une position, qui est la borne supérieure de la suite, sera toujours à branchement fini.* □

Definition 4.2.3. (Arbres du jeu) *Étant donné une arène de jeu* $\mathcal{S} = (\mathbb{P}, \lambda, \to, \mathbb{P}_\mathrm{I})$, *l'ensemble des* arbres de la syntaxe *(ou* du jeu*)* $\mathcal{S}$, *notés* $\mathrm{IT}(\mathcal{S})$, *est l'image de l'application* $\mathrm{t}^{\mathcal{S}}_{(.)} : \mathbb{P} \to \mathrm{IT}(\mathbb{P})$ *qui associe à toute position son arbre* $\mathrm{t}^{\mathcal{S}}_{(\pi)} \triangleq \mathrm{t}^{\mathcal{S}}_\pi$ *:*

$$\mathrm{IT}(\mathcal{S}) \triangleq \mathrm{t}^{\mathcal{S}}_{(\mathbb{P})} = \mathrm{Im}_{\mathrm{t}^{\mathcal{S}}_{(.)}} \mathbb{P}$$

Parmi ces derniers, nous noterons par $\mathrm{T}(\mathcal{S})$ *l'ensemble des arbres de jeu* finis, *et par* $\mathrm{RT}(\mathcal{S})$ *l'ensemble des arbres de jeu* rationnels *:*

$$\mathrm{T}(\mathcal{S}) \triangleq \mathrm{IT}(\mathcal{S}) \cap \mathrm{T}(\mathbb{P})$$
$$\mathrm{RT}(\mathcal{S}) \triangleq \mathrm{IT}(\mathcal{S}) \cap \mathrm{RT}(\mathbb{P})$$

□

Étant constitués d'arbres à branchement fini (cf. prop. 4.2.2), l'ensemble $\mathrm{IT}(\mathcal{S})$ coïncide avec $\mathrm{BT}(\mathcal{S}) \triangleq \mathrm{IT}(\mathcal{S}) \cap \mathrm{BT}(\mathbb{P})$. Nous avons utilisé, comme pour les termes $\mathrm{IT}(\Sigma)$, $\mathrm{T}(\Sigma)$, $\mathrm{RT}(\Sigma)$, un abus de notation : quand l'argument est une syntaxe de jeu $\mathcal{S}$, les écritures $\mathrm{IT}(\mathcal{S})$, $\mathrm{T}(\mathcal{S})$, $\mathrm{RT}(\mathcal{S})$, dénotent des ensembles d'arbres (tous les arbres, les arbres finis, les arbres rationnels) du jeu $\mathcal{S}$ et non pas les ensembles correspondants d'arbres étiquetés dans $\mathcal{S}$ (ce qui n'aurait aucun sens puisqu'une syntaxe de jeu n'est pas un ensemble mais un quadruplet d'éléments).

Il n'existe donc pas un arbre de jeu mais, au contraire, autant d'arbres que de positions du jeu. Bien entendu, si dans le jeu il existe une seule position initiale ($\mathbb{P}_\mathrm{I} = \{\bar{\pi}\}$ est un singleton), nous pourrons dire *l'arbre du jeu* pour évoquer l'arbre $\mathrm{t}^{\mathcal{S}}_{\bar{\pi}}$ de cette unique position. Plus généralement, les *arbres* l*égaux*, évoqueront l'image de la fonction $\mathrm{t}^{\mathcal{S}}_{(.)}$ sur l'ensemble $\mathbb{P}_\mathrm{I}$. L'ensemble des arbres de jeu est comparable à celui des termes sur une signature : si les termes sont des arbres étiquetés d'une façon qui respecte le type et l'arité des symboles de fonction utilisés, les arbres du jeu sont des arbres étiquetés d'une façon qui respecte les règles du jeu, c'est-à-dire les mouvements légaux dans le jeu.

Notation 4.2.4. Quand la référence à une syntaxe de jeu $\mathcal{S}$ donnée sera évidente, pour alléger la notation, nous écrirons $\mathrm{t}_{(.)}$ au lieu de $\mathrm{t}^{\mathcal{S}}_{(.)}$.

Il est évident qu'il existe un rapport entre l'arbre d'une position π et celui d'une position suivante π' dans le déroulement du jeu : l'arbre de la seconde est une branche de l'arbre de la première.

Proposition 4.2.5. *Pour toute position* π, *si* $\pi \xrightarrow{\ell} \pi'$ *alors* $(\mathrm{t}_\pi)|_\ell = \mathrm{t}_{\pi'}$.

DÉMONSTRATION. *On démontre, par induction sur* $k \in \mathbb{N}$, *que* $(t_\pi^{k+1})|_\ell = t_{\pi'}^{k}$.

- $k = 0$

$$\begin{array}{rcll} (t_\pi^1)|_\ell & = & (\mathrm{step}_\rightarrow t_\pi^0)|_\ell & (a) \\ & = & (\{(\varepsilon,\pi)\} \cup \{(\ell',\pi'') \mid \pi \xrightarrow{\ell'} \pi''\})|_\ell & (b) \\ & = & \{(\varepsilon,\pi')\} & (c) \\ & = & t_{\pi'}^0 & (d) \end{array}$$

(a) par définition de t_π^k *; (b) par définition de* $\mathrm{step}_\rightarrow(.)$ *; (c) par définition de branche; (d) par définition de* $t_{\pi'}^0$.

- $k \Rightarrow k+1$

$$\begin{array}{rcll} (t_\pi^{k+1})|_\ell & = & (\mathrm{step}_\rightarrow t_\pi^k)|_\ell & (a) \\ & = & (t_\pi^k \cup \{(u.\ell',\pi_2) \mid (u,\pi_1) \in \mathrm{Feuilles}(t_\pi^k),\ \pi_1 \xrightarrow{\ell'} \pi_2\})|_\ell & (b) \\ & = & t_\pi^k|_\ell \cup \{(u.\ell',\pi_2) \mid (\ell.u,\pi_1) \in \mathrm{Feuilles}(t_\pi^k),\ \pi_1 \xrightarrow{\ell'} \pi_2\} & (c) \\ & = & t_\pi^k|_\ell \cup \{(u.\ell',\pi_2) \mid (u,\pi_1) \in \mathrm{Feuilles}(t_\pi^k|_\ell),\ \pi_1 \xrightarrow{\ell'} \pi_2\} & (d) \\ & = & t_{\pi'}^{k-1} \cup \{(u.\ell',\pi_2) \mid (u,\pi_1) \in \mathrm{Feuilles}(t_{\pi'}^{k-1}),\ \pi_1 \xrightarrow{\ell'} \pi_2\} & (e) \\ & = & \mathrm{step}_\rightarrow t_{\pi'}^{k-1} & (f) \\ & = & t_{\pi'}^k & (g) \end{array}$$

(a) par définition de t_π^{k-1} *; (b) par définition de* $\mathrm{step}_\rightarrow(.)$ *; (c) par linéarité de l'opération branche vis-à-vis de l'union ensembliste; (d) puisque la condition* $(\ell.u,\pi_1) \in \mathrm{Feuilles}(t_\pi^k)$ *équivaut à* $(u,\pi_1) \in \mathrm{Feuilles}(t_\pi^k|_\ell)$ *; (e) par hypothèse inductive; (f) par définition de* $\mathrm{step}_\rightarrow(.)$ *; (g) par définition de* $t_{\pi'}^k$.

Puisque les deux suites sont co-finales, leurs bornes supérieures coïncident (cf. lemme 2.2.1). □

La proposition précédente implique que l'ensemble des branches immédiates d'un arbre d'une position π coïncide avec la forêt des arbres des positions suivantes à π dans le jeu. On peut donc écrire l'arbre d'une position en fonction des arbres des positions suivantes, en utilisant la fonction de translation $(.).(.) : \mathbb{N}_+^* \times \mathrm{IT}(\mathsf{POS}) \rightarrow \mathrm{IT}(\mathsf{POS})$ (cf. section 3.2.3, page 88) pour les décaler à la bonne adresse.

COROLLARY 4.2.6. *L'application* $t_{(\pi)}^{\mathcal{S}} \triangleq t_\pi^{\mathcal{S}}$ *qui associe à toute position son arbre est l'unique fonction* $t_{(.)} : \mathbb{P} \rightarrow \mathrm{IT}(\mathbb{P})$ *telle que :*

$$\forall \pi \in \mathbb{P}. \quad t_{(\pi)} = \{(\varepsilon,\pi)\} \cup \bigcup_{\pi \xrightarrow{\ell} \pi'} \ell.t_{(\pi')}$$

DÉMONSTRATION. *La Prop. 4.2.5 assure que, pour tout* $\pi \in \mathbb{P}$, *l'arbre* $t_{(\pi)}^{\mathcal{S}}$ *est une solution de l'équation ci-dessus. Pour prouver l'unicité, supposons qu'il existe deux solutions* $f_{(.)}, g_{(.)} : \mathbb{P} \rightarrow \mathrm{IT}(\mathbb{P})$ *vérifiant la condition ci-dessus. Soit* $\pi \in \mathbb{P}$ *une position arbitraire, et* $u \in \mathbb{N}_+^*$ *une adresse arbitraire. Nous voulons montrer que supposer une des deux telle que* $f_{(\pi)}(u) = \pi''$ *impliquera* $g_{(\pi)}(u) = \pi''$. *Nous procédons par induction sur la longueur* $|u|$ *de l'adresse.*

- $|u| = 0$

  Dans le cas $u = \varepsilon$, *l'équation implique* $f_{(\pi)}(u) = f_{(\pi)}(\varepsilon) = \pi = g_{(\pi)}(\varepsilon) = g_{(\pi)}(u)$

– $|u| \Rightarrow |u| + 1$
Soit $u = i.v$ *où* $i \in \mathbb{N}_+$. *Si l'on suppose* $f_{(\pi)}(u) = \pi''$, *utilisant l'équation, il existe forcément un mouvement du jeu* $\pi \xrightarrow{i} \pi'$ *tel que* $f_{(\pi)}(u) = f_{(\pi)}(i.v) = f_{(\pi')}(v)$. *Si un tel mouvement existe, nous pouvons écrire pour l'autre fonction aussi que* $g_{(\pi)}(u) = g_{(\pi)}(i.v) = g_{(\pi')}(v)$. *Par hypothèse d'induction* $f_{(\pi')}(v) = g_{(\pi')}(v)$, *d'où la conclusion.*

□

EXAMPLE 4.2.7. **Arbres du Morpion**. Malgré la simplicité du jeu, nous ne pouvons pas illustrer l'arbre complet du Morpion, qui a un nombre de noeuds de l'ordre de 9!. Dans la figure 4.2.1 nous nous limitons donc à montrer l'arbre d'une position donnée (donc une branche de l'arbre du jeu), proche de la fin du match, telle que $\pi = (\times, d)$ où :

$$d = \{(1, \times), (2, \times), (3, \circ), (4, \circ), (5, \circ), (6, \text{vide}), (7, \times), (8, \text{vide}), (9, \text{vide})\}.$$

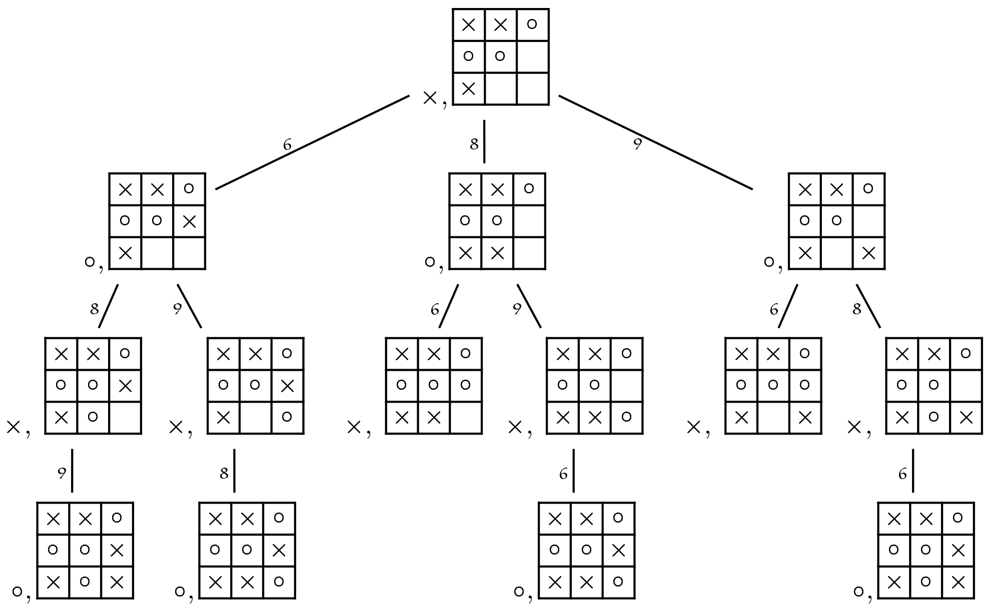
×	×	○
○	○	
×		

Les coups possibles dans la position π correspondent à cocher une des trois cases encore vides : la 6, la 8 ou la 9. D'après le corollaire 4.2.6, l'arbre de la position π est égal à l'ensemble $\{(\varepsilon, \pi)\} \cup 6.t_{\pi_6} \cup 8.t_{\pi_8} \cup 9.t_{\pi_9}$, où π_6,π_8 et π_9 sont les positions d'arrivée correspondantes, respectivement, aux mouvements 6,8 et 9. En d'autres termes, si l'étiquette π décore la racine de t_π, la décoration du reste de l'arbre renvoie à celle de ses sous-arbres : $t_\pi(6.u) = t_{\pi_6}(u)$, $t_\pi(8.u) = t_{\pi_8}(u)$ et $t_\pi(9.u) = t_{\pi_9}(u)$, et ainsi de suite, récursivement.

On remarquera que, suite aux mouvements 8 et 9, l'*opposant* ($\circ$) a la possibilité de gagner le match. Jusqu'à présent nous n'avons pas formalisé ce genre de notion. La syntaxe du jeu ne décèle pas *qui* gagne le jeu ni *combien* il gagne : la définition des positions gagnantes et celle de gain reviendra à la discussion sur la sémantique du jeu, non pas à son développement combinatoire. □

4.2.1. Le vocabulaire des jeux. La plupart des notions de la théorie des jeux combinatoires trouve un correspondant dans la théorie des arbres. Étant donné une arène de jeu $\mathcal{S} = (\mathbb{P}, \lambda, \rightarrow, \mathbb{P}_{\mathrm{I}})$ nous disons qu'une position $\pi \in \mathbb{P}$ est *terminale* si, à partir d'elle, il n'existe pas de transitions (coups) possibles : $\nexists \pi'.\ \pi \rightarrow \pi'$, sinon elle est *non terminale*. En termes d'arbres, une position est terminale si son arbre t_π est constitué de la seule feuille (ε, π). Nous noterons $\mathbb{P}_{\mathrm{T}}$ l'ensemble des positions terminales et $\mathbb{P}_{\mathrm{NT}}$ l'ensemble des positions non terminales : $\mathbb{P}_{\mathrm{NT}} = \mathbb{P} \setminus \mathbb{P}_{\mathrm{T}}$. La suite des *positions succédant* à π, notée $\mathrm{Succ}(\pi) : \mathbb{P}^*$, coïncide avec celle des successeurs de l'arbre t_π à la racine en faisant abstraction des adresses impliquées : $\mathrm{Succ}(\pi)_i \stackrel{\leftarrow}{=} \pi_i$ où $(\ell_i, \pi_i) = \mathrm{Succ}_i^{(t_\pi, \varepsilon)}$. Le *nombre de coups disponibles* dans une position π est la cardinalité de l'ensemble $\{\ell \in \mathbb{N} \mid \pi \xrightarrow{\ell} \pi'\}$, c'est-à-dire le degré de branchement $\text{degré}(t_\pi, \varepsilon)$. La distinction entre positions du joueur et positions de l'opposant se retrouve sur les noeuds et les feuilles d'un arbre t_π : une adresse u ou un élément

FIG. 4.2.1. Une branche du Morpion



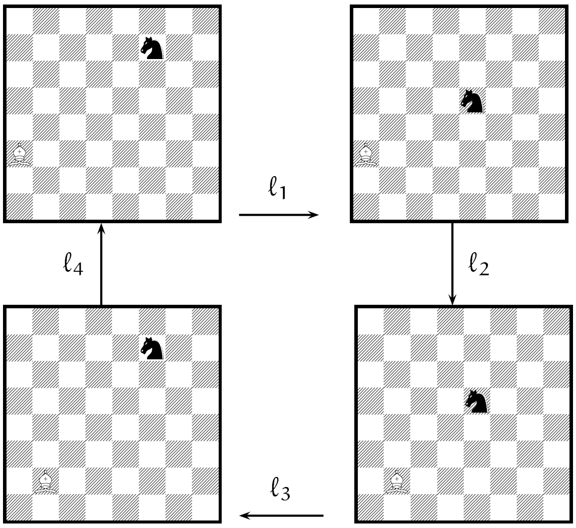
$(u, t_\pi(u))$, noeud ou feuille, de l'arbre t_π est dit *du joueur* ou *de l'opposant* si, respectivement, $\lambda(t_\pi(u)) = \text{Player}$ ou $\lambda(t_\pi(u)) = \text{Opponent}$. Quand une adresse est d'un joueur, on dit aussi qu'il *a la main* ou que, à cette adresse, *c'est à lui de jouer*.

NOTATION 4.2.8. Dans les représentations graphiques des arbres de jeu, il est courant de dessiner différemment les noeuds du *joueur* et ceux de l'*opposant*. Par la suite, suivant cette pratique, nous renfermerons dans un cercle les positions du *joueur* et dans un rectangle celles de l'*opposant*.

REMARK 4.2.9. Il ne faut pas confondre les *adresses d'un joueur,* qui sont des mots $u \in \mathbb{N}_+^*$, avec les *positions d'un joueur*, qui appartiennent à l'ensemble $\mathbb{P}$ défini par la syntaxe de jeu. Le vocabulaire courant en théorie des termes, qui appelle *position* ce que nous appelons *adresse*, peut entretenir une certaine confusion. Une position d'un arbre, nous disons adresse, n'est pas une position du jeu mais, on peut dire plutôt, qu'elle *montre* une position du jeu, celle qui se présenterait aux joueurs après la séquence de coups que l'adresse indique. En effet, rien n'empêche une position de paraître à plusieurs adresses d'un arbre de jeu. L'exemple de la figure 4.2.2, tiré des Échecs[2], illustre comme une même position puisse se retrouver à plusieurs endroits dans l'arbre, en particulier dans un même chemin. Si on appelle, dans l'ordre d'apparition, les positions de la boucle π_1, π_2, π_3 et π_4, et les étiquettes des coups impliqués ℓ_1, ℓ_2, ℓ_3 et ℓ_4, il est évident que la position π_1 se trouvant à l'adresse u de l'arbre, se retrouve aussi à toutes les adresses, une infinité, définies par l'expression régulière $u.(\ell_1.\ell_2.\ell_3.\ell_4)^*$.

[2]Nous considérons ne pas utiliser les règles modernes qui évitent que les positions se répètent indéfiniment.

FIG. 4.2.2. Boucles dans les Échecs



4.2.2. Arbres de jeu et dualité. On remarquera que la définition des arbres d'une syntaxe de jeu $\mathcal{S} = (\mathbb{P}, \lambda, \rightarrow, \mathbb{P}_{\mathrm{I}})$ (Def. 4.2.3, page 116) ne dépend pas de la fonction de classification λ. Par conséquent, la fonction qui associe à toute position du jeu son arbre : $\mathrm{t}^{\mathcal{S}}_{(.)} : \mathbb{P} \rightarrow \mathrm{IT}(\mathbb{P})$ est identique dans la syntaxe et sa duale : $\mathrm{t}^{\mathcal{S}}_{(.)} = \mathrm{t}^{\mathcal{S}^{\perp}}_{(.)}$. Intuitivement cela ne laissait aucun doute : regarder le match au dos d'un joueur où au dos de son adversaire ne conditionne en aucun cas le déroulement du jeu.

4.3. Préfixes du jeu

L'ensemble des préfixes du jeu est une construction essentielle à l'évaluation d'un jeu où les arbres légaux peuvent êtres infinis.

DEFINITION 4.3.1. (PRÉFIXES DU JEU) *Étant donné une arène de jeu* $\mathcal{S} = (\mathbb{P}, \lambda, \rightarrow, \mathbb{P}_{\mathrm{I}})$*, l'ensemble des* préfixes du jeu $\mathcal{S}$*, noté* $\widehat{\mathrm{IT}}(\mathcal{S})$*, est la clôture vers le bas de l'ensemble des arbres du jeu, dans le domaine des arbres étiquetés dans* $\mathbb{P}$ *équipé de l'ordre préfixe :*

$$\widehat{\mathrm{IT}}(\mathcal{S}) \triangleq \downarrow_{(\mathrm{IT}(\mathbb{P}),\leq)} \mathrm{IT}(\mathcal{S})$$

Parmi ces derniers, nous noterons par $\widehat{\mathrm{T}}(\mathcal{S})$ *l'ensemble des préfixes de jeu* finis*, et par* $\widehat{\mathrm{RT}}(\mathcal{S})$ *l'ensemble des préfixes de jeu* rationnels *:*

$$\widehat{\mathrm{T}}(\mathcal{S}) \triangleq \widehat{\mathrm{IT}}(\mathcal{S}) \cap \mathrm{T}(\mathbb{P})$$
$$\widehat{\mathrm{RT}}(\mathcal{S}) \triangleq \widehat{\mathrm{IT}}(\mathcal{S}) \cap \mathrm{RT}(\mathbb{P})$$

□

La construction utilisée pour définir les préfixes d'un jeu est l'union, pour tous les arbres de jeu possibles, des correspondants cônes dans $(\mathrm{IT}(\mathbb{P}), \leq)$:

$$\widehat{\mathrm{IT}}(\mathcal{S}) = \bigcup_{\mathrm{t}\in\mathrm{IT}(\mathcal{S})} \downarrow_{(\mathrm{IT}(\mathbb{P}),\leq)} \mathrm{t}$$

NOTATION 4.3.2. Pour les préfixes finis $\widehat{\mathrm{T}}(\mathcal{S})$ ou infinis $\widehat{\mathrm{IT}}(\mathcal{S})$ d'un jeu, nous indiquerons respectivement par l'écriture $\widehat{\mathrm{T}}(\mathcal{S},\pi)$ et $\widehat{\mathrm{IT}}(\mathcal{S},\pi)$ les sous-ensembles relatifs à la position π :

$$\widehat{\mathrm{T}}(\mathcal{S},\pi) = \{\mathrm{t} \in \widehat{\mathrm{T}}(\mathcal{S}) \mid \mathrm{t}(\varepsilon) = \pi\}$$
$$\widehat{\mathrm{IT}}(\mathcal{S},\pi) = \{\mathrm{t} \in \widehat{\mathrm{IT}}(\mathcal{S}) \mid \mathrm{t}(\varepsilon) = \pi\}$$

PROPOSITION 4.3.3. *La structure* $(\widehat{\mathrm{IT}}(\mathcal{S}), \leq: 1^{\widehat{\mathrm{IT}}(\mathcal{S})\times\widehat{\mathrm{IT}}(\mathcal{S})})$ *est un sous-dcpo de* $(\mathrm{IT}(\mathbb{P}), \leq: 1^{\mathrm{IT}(\mathbb{P})\times\mathrm{IT}(\mathbb{P})})$*, donc un dcpo d'ensembles.*

DÉMONSTRATION. *Conséquence directe du lemme 2.6.8.* □

PROPOSITION 4.3.4. *Le domaine des préfixes du jeu est isomorphe aux dirigés de préfixes finis du jeu modulo co-finalité :*

$$(\widehat{\mathrm{IT}}(\mathcal{S}), \leq) \;\cong\; (\Delta(\widehat{\mathrm{T}}(\mathcal{S}), \leq)_{\approx^\flat}, \leq^\flat) \;=\; (\widehat{\mathrm{T}}(\mathcal{S})^\flat, \leq^\flat)$$

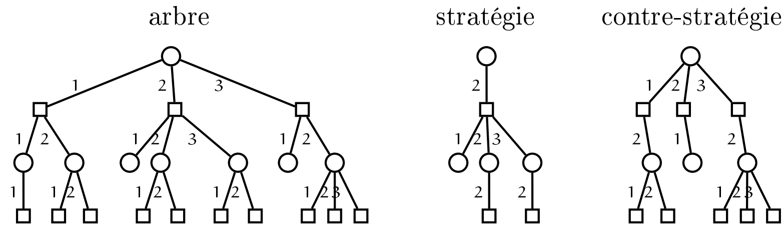
DÉMONSTRATION. *Les éléments compacts de* $(\widehat{\mathrm{IT}}(\mathcal{S}), \leq)$ *sont les préfixes finis d'arbres de jeu. La structure* $(\widehat{\mathrm{IT}}(\mathcal{S}), \leq)$ *est un dcpo algébrique puisque tout élément peut s'exprimer comme limite de ses approximants finis (compacts). Ainsi, l'énoncé est conséquence directe du lemme 2.5.8.* □

4.4. Stratégies

Une des notions les plus importantes de la théorie des jeux est certainement celle de *stratégie*. La signification intuitive d'une stratégie est celle d'un plan pour jouer des parties d'un jeu : on pense à un joueur qui se dit : si ceci arrive, j'agis comme cela, tandis que si cela arrive, etc. Dans les jeux combinatoires, une stratégie pour un joueur peut être représentée par un élagage particulier (cf. Def. 3.1.10) de l'arbre de jeu : à chaque noeud où c'est à lui de jouer, il a fait le choix de son coup, donc il n'y a qu'un descendant.

DEFINITION 4.4.1. *Étant donné une arène de jeu* $\mathcal{S} = (\mathbb{P}, \lambda, \rightarrow, \mathbb{P}_{\mathrm{I}})$ *et une position* $\pi \in \mathbb{P}$ *du jeu, une* stratégie du joueur *(ou* stratégie*) pour la position* π*, est un élagage de l'arbre* t_π *complet dans les noeuds de l'*opposant *et déterministe dans les noeuds du* joueur*. De façon symétrique, une* stratégie de l'opposant *(ou* contre-stratégie*) pour la position* π*, est un élagage de l'arbre* t_π *complet dans les noeuds du* joueur *et déterministe dans les noeuds de l'*opposant*. En d'autre termes, une* contre-stratégie *est une* stratégie *dans l'arène duale* $\mathcal{S}^\perp$. □

TAB. 1. Exemple de stratégie et contre-stratégie



La Table 1 illustre la définition précédente avec un exemple de stratégie et un exemple de contre-stratégie.

4.4.1. Élagages du joueur et de l'opposant. Dans la construction d'une stratégie, non seulement l'élagage dans les noeuds de l'opposant est interdit, mais il est forcé d'être déterministe dans les noeuds du joueur. En conservant, de ces deux conditions, seulement celle qui proscrit les élagages dans les positions de l'opposant, nous obtenons la notion d'*élagage du joueur*, qui est plus générale que celle de stratégie. De façon duale, une notion plus générale que celle de contre-stratégie est celle d'*élagage de l'opposant*, qui est un élagage complet dans les noeuds du joueur.

DEFINITION 4.4.2. (ÉLAGAGES DES JOUEURS) *Étant donné une arène de jeu* $\mathcal{S} = (\mathbb{P}, \lambda, \to, \mathbb{P}_{\mathrm{I}})$, *la relation d'*élagage du joueur, *notée* $\subseteq_{\mathrm{p}}$, *et la relation d'*élagage de l'opposant, *notée* $\subseteq_{\mathrm{o}}$, *sont définies sur le carré* $\mathrm{IT}(\mathbb{P}) \times \mathrm{IT}(\mathbb{P})$ *de la façon suivante :*

$$\begin{array}{lll} \mathrm{t}' \subseteq_{\mathrm{p}} \mathrm{t} & \stackrel{\triangle}{\iff} & \textit{est un élagage de } \mathrm{t} \textit{ complet dans les noeuds de l'opposant} \\ \mathrm{t}' \subseteq_{\mathrm{o}} \mathrm{t} & \stackrel{\triangle}{\iff} & \textit{est un élagage de } \mathrm{t} \textit{ complet dans les noeuds du joueur} \end{array}$$

L'ensemble des élagages du joueur, *noté* $\widetilde{\mathrm{IT}}_{\mathrm{p}}(\mathcal{S})$, *et l'ensemble des* élagages de l'opposant, *noté* $\widetilde{\mathrm{IT}}_{\mathrm{o}}(\mathcal{S})$, *sont la clôture vers le bas dans* $\mathrm{IT}(\mathbb{P})$ *avec la respective relation d'élagage (du joueur ou de l'opposant) :*

$$\widetilde{\mathrm{IT}}_{\mathrm{p}}(\mathcal{S}) \triangleq \downarrow_{(\mathrm{IT}(\mathbb{P}),\subseteq_{\mathrm{p}})} \mathrm{IT}(\mathcal{S})$$
$$\widetilde{\mathrm{IT}}_{\mathrm{o}}(\mathcal{S}) \triangleq \downarrow_{(\mathrm{IT}(\mathbb{P}),\subseteq_{\mathrm{o}})} \mathrm{IT}(\mathcal{S})$$

Parmi ces derniers, nous noterons $\widetilde{\mathrm{T}}_{\mathrm{p}}(\mathrm{S}) \triangleq \widetilde{\mathrm{IT}}_{\mathrm{p}}(\mathcal{S}) \cap \mathrm{T}(\mathbb{P})$ *l'ensemble des* élagages finis du joueur, $\widetilde{\mathrm{T}}_{\mathrm{o}}(\mathcal{S}) \triangleq \widetilde{\mathrm{IT}}_{\mathrm{o}}(\mathcal{S}) \cap \mathrm{T}(\mathbb{P})$ *l'ensemble des* élagages finis de l'opposant, $\mathrm{IS}_{\mathrm{p}}(\mathcal{S})$ *l'ensemble des* stratégies, *et* $\mathrm{IS}_{\mathrm{o}}(\mathcal{S})$ *l'ensemble des* contre-stratégies. □

Nous avons, par définition, $\mathrm{IS}_{\mathrm{p}}(\mathcal{S}) \subseteq \widetilde{\mathrm{IT}}_{\mathrm{p}}(\mathcal{S})$ et $\mathrm{IS}_{\mathrm{o}}(\mathcal{S}) \subseteq \widetilde{\mathrm{IT}}_{\mathrm{o}}(\mathcal{S})$. La construction utilisée pour définir les ensembles $\widetilde{\mathrm{IT}}_{\mathrm{p}}(\mathcal{S})$ et $\widetilde{\mathrm{IT}}_{\mathrm{o}}(\mathcal{S})$ est l'union, pour tous les arbres de jeu possibles, des cônes correspondants dans, respectivement, $(\mathrm{IT}(\mathbb{P}), \subseteq_{\mathrm{p}})$ et $(\mathrm{IT}(\mathbb{P}), \subseteq_{\mathrm{o}})$:

$$\widetilde{\mathrm{IT}}_{\mathrm{p}}(\mathcal{S}) = \bigcup_{\mathrm{t} \in \mathrm{IT}(\mathcal{S})} \downarrow_{(\mathrm{IT}(\mathbb{P}),\subseteq_{\mathrm{p}})} \mathrm{t}$$
$$\widetilde{\mathrm{IT}}_{\mathrm{o}}(\mathcal{S}) = \bigcup_{\mathrm{t} \in \mathrm{IT}(\mathcal{S})} \downarrow_{(\mathrm{IT}(\mathbb{P}),\subseteq_{\mathrm{o}})} \mathrm{t}$$

PROPOSITION 4.4.3. *Les structures* $(\widetilde{\mathrm{IT}}_{\mathrm{p}}(\mathcal{S}), \subseteq_{\mathrm{p}}: 1^{\widetilde{\mathrm{IT}}_{\mathrm{p}}(\mathcal{S}) \times \widetilde{\mathrm{IT}}_{\mathrm{p}}(\mathcal{S})})$ *et* $(\widetilde{\mathrm{IT}}_{\mathrm{o}}(\mathcal{S}), \subseteq_{\mathrm{o}}: 1^{\widetilde{\mathrm{IT}}_{\mathrm{o}}(\mathcal{S}) \times \widetilde{\mathrm{IT}}_{\mathrm{o}}(\mathcal{S})})$ *sont des sous-dcpo de la structure* $(\mathrm{IT}(\mathbb{P}), \subseteq: 1^{\mathrm{IT}(\mathbb{P}) \times \mathrm{IT}(\mathbb{P})})$*, et sont donc des dcpo d'ensembles.*

DÉMONSTRATION. *Corollaire du lemme 2.6.8.* □

PROPOSITION 4.4.4. *Les structures* $(\widetilde{\widetilde{\mathrm{IT}}}_{\mathrm{p}}(\mathcal{S}), \subseteq_{\mathrm{p}}: 1^{\widetilde{\mathrm{IT}}_{\mathrm{p}}(\mathcal{S}) \times \widetilde{\mathrm{IT}}_{\mathrm{p}}(\mathcal{S})})$ *et* $(\widetilde{\widetilde{\mathrm{IT}}}_{\mathrm{o}}(\mathcal{S}), \subseteq_{\mathrm{o}}: 1^{\widetilde{\mathrm{IT}}_{\mathrm{o}}(\mathcal{S}) \times \widetilde{\mathrm{IT}}_{\mathrm{o}}(\mathcal{S})})$ *sont isomorphes aux dirigés d'élagages finis (respectivement du joueur et de l'opposant) modulo co-finalité :*

$$\begin{array}{rcl} (\widetilde{\widetilde{\mathrm{IT}}}_{\mathrm{p}}(\mathcal{S}), \subseteq_{\mathrm{p}}) & \cong & (\Delta(\widetilde{\mathrm{T}}_{\mathrm{p}}(\mathcal{S}), \subseteq_{\mathrm{p}})_{\approx^\flat}, \subseteq^\flat_{\mathrm{p}}) \;=\; (\widetilde{\mathrm{T}}_{\mathrm{p}}(\mathcal{S})^\flat, \subseteq^\flat_{\mathrm{p}}) \\ (\widetilde{\widetilde{\mathrm{IT}}}_{\mathrm{o}}(\mathcal{S}), \subseteq_{\mathrm{o}}) & \cong & (\Delta(\widetilde{\mathrm{T}}_{\mathrm{o}}(\mathcal{S}), \subseteq_{\mathrm{o}})_{\approx^\flat}, \subseteq^\flat_{\mathrm{o}}) \;=\; (\widetilde{\mathrm{T}}_{\mathrm{o}}(\mathcal{S})^\flat, \subseteq^\flat_{\mathrm{o}}) \end{array}$$

DÉMONSTRATION. *D'une part, les éléments compacts de la structure* $(\widetilde{\mathrm{IT}}_{\mathrm{p}}(\mathcal{S}), \subseteq_{\mathrm{p}})$ *sont les élagages finis du joueurs. D'autre part, la structure* $(\widetilde{\mathrm{IT}}_{\mathrm{p}}(\mathcal{S}), \subseteq_{\mathrm{p}})$ *est un dcpo algébrique puisque tout élément peut s'exprimer comme limite de ses approximants finis (compacts). L'énoncé est donc conséquence directe du lemme 2.5.8. Pour la structure* $(\widetilde{\mathrm{IT}}_{\mathrm{o}}(\mathcal{S}), \subseteq_{\mathrm{o}})$*, nous pouvons réutiliser les mêmes arguments.* □

4.5. Algèbres syntaxiques

4.5.1. Algèbre syntaxique complète. L'algèbre syntaxique d'un jeu $\mathcal{S}$ est constituée par les domaines et les opérations que nous avons définis dans les sections précédentes. Les contreparties sémantiques, que nous appellerons algèbres *d'évaluation* du jeu, donneront un sens précis (gagnant, perdant, match null ou autre) aux éléments syntaxiques et, en particulier, aux positions du jeu.

La table 2 résume l'ensemble des domaines et des opérations constituant l'algèbre syntaxique (complète) du jeu, qui sera notée $\mathsf{A}_{\mathcal{S}}$. La notation utilisée entend être mnémonique. Un nom de domaine coiffé par les symboles $\widetilde{\ }$ ou $\widehat{\ }$ dénotera toujours une clôture vers le bas du domaine impliqué. L'ordre utilisé pour la clôture pourra être l'ordre d'élagage $\subseteq$, évoqué par la sinuosité du premier symbole, ou bien l'ordre préfixe $\leq$, évoqué par l'angularité du second. En raison des isomorphismes évoqués précédemment, les opérations polymorphes, appliquées au domaine des compacts, seront notées avec le même symbole $\leq$, $\subseteq_{\mathrm{p}}$ et $\subseteq_{\mathrm{o}}$, alors que, appliquées au domaine des éléments arbitraires, elles seront notées $\leq^\flat$, $\subseteq^\flat_{\mathrm{p}}$ et $\subseteq^\flat_{\mathrm{o}}$.

4.5.2. Sous-algèbres syntaxiques. Nous allons définir trois sous-algèbres de l'algèbre syntaxique complète, la sous-algèbre *principale*, la sous-algèbre *du joueur* et celle *de l'opposant*. La vocation de chacune sera liée à l'interprétation sémantique du jeu. Elles seront définies par l'ensemble des domaines et des opérations sélectionné dans l'algèbre syntaxique complète (l'interprétation des valeurs étant ici l'identité, cf. section 1.4.1). Ainsi, la signature suffira pour exprimer les domaines et les opérations élues pour constituer la sous-algèbre.

TAB. 2. Algèbre syntaxique complète

$(\mathbb{P}$		,	*positions du jeu*
$\mathbb{P}_{\mathrm{T}}$		,	*positions terminales*
$\mathbb{P}_{\mathrm{NT}}$		,	*positions non terminales*
$\mathrm{T}(\mathbb{P})$		,	*arbres finis étiquetés dans* $\mathbb{P}$
$\mathrm{T}(\mathcal{S})$		,	*arbres finis du jeu*
$\mathrm{IT}(\mathcal{S})$		,	*arbres finis ou infinis du jeu*
$\widehat{\mathrm{T}}(\mathcal{S})$		,	*préfixes finis du jeu*
$\widehat{\mathrm{IT}}(\mathcal{S})$		,	*préfixes arbitraires du jeu* $(\cong \widehat{\mathrm{T}}(\mathcal{S})^{\flat})$
$\widetilde{\mathrm{T}}_{\mathrm{p}}(\mathcal{S})$		,	*élagages finis du joueur*
$\widetilde{\mathrm{IT}}_{\mathrm{p}}(\mathcal{S})$		,	*élagages arbitraires du joueur* $(\cong \widetilde{\mathrm{T}}_{\mathrm{p}}(\mathcal{S})^{\flat})$
$\mathrm{IS}_{\mathrm{p}}(\mathcal{S})$		,	*stratégies (du joueur)*
$\widetilde{\mathrm{T}}_{\mathrm{o}}(\mathcal{S})$		,	*élagages finis de l'opposant*
$\widetilde{\mathrm{IT}}_{\mathrm{o}}(\mathcal{S})$		,	*élagages arbitraires de l'opposant* $(\cong \widetilde{\mathrm{T}}_{\mathrm{o}}(\mathcal{S})^{\flat})$
$\mathrm{IS}_{\mathrm{o}}(\mathcal{S})$		;	*contre-stratégies (de l'opposant)*
$\leq$	$: \forall\alpha{:}\{\widehat{\mathrm{T}}(\mathcal{S}), \widehat{\mathrm{IT}}(\mathcal{S})\}.\,1^{\alpha\times\alpha}$	,	*ordre préfixe*
$\subseteq_{\mathrm{p}}$	$: \forall\alpha{:}\{\widetilde{\mathrm{T}}_{\mathrm{p}}(\mathcal{S}), \widetilde{\mathrm{IT}}_{\mathrm{p}}(\mathcal{S})\}.\,1^{\alpha\times\alpha}$	,	*ordre élagage du joueur*
$\subseteq_{\mathrm{o}}$	$: \forall\alpha{:}\{\widetilde{\mathrm{T}}_{\mathrm{o}}(\mathcal{S}), \widetilde{\mathrm{IT}}_{\mathrm{o}}(\mathcal{S})\}.\,1^{\alpha\times\alpha}$	)	*ordre élagage de l'opposant*

TAB. 3. Sous-algèbre syntaxique principale $\mathsf{A}^{\mathrm{e}}_{\mathcal{S}}$

$(\mathbb{P}_{\mathrm{T}}$		,	*positions terminales*
$\mathbb{P}_{\mathrm{NT}}$		,	*positions non terminales*
$\mathrm{T}(\mathbb{P})$		,	*arbres finis étiquetés dans* $\mathbb{P}$
$\widehat{\mathrm{T}}(\mathcal{S})$		,	*préfixes finis du jeu*
$\widehat{\mathrm{IT}}(\mathcal{S})$		,	*préfixes arbitraires du jeu* $(\cong \widehat{\mathrm{T}}(\mathcal{S})^{\flat})$
$\mathrm{IT}(\mathcal{S})$		;	*arbres finis ou infinis du jeu*
$\leq$	$: \forall\alpha{:}\{\widehat{\mathrm{T}}(\mathcal{S}), \widehat{\mathrm{IT}}(\mathcal{S})\}.\,1^{\alpha\times\alpha}$	,	*ordre préfixe*

4.5.2.1. *Sous-algèbre principale.* La définition de la sous-algèbre syntaxique *principale* ou *essentielle* a le but de cerner l'ensemble des éléments nécessaires à l'évaluation des arbres de $\mathrm{IT}(\mathcal{S})$, principale cible de l'évaluation sémantique. Cette sous-algèbre, notée $\mathsf{A}^{\mathrm{e}}_{\mathcal{S}}$, comprend les domaines des préfixes finis ou infinis du jeu, avec les ensembles de positions terminales $\mathbb{P}_{\mathrm{T}}$ et non terminales $\mathbb{P}_{\mathrm{NT}}$, et le semi-prédicat polymorphe *préfixe* $\leq$. En définissant une évaluation de tous les arbres de $\mathrm{T}(\mathbb{P})$, nous obtiendrons, en particulier, une évaluation des préfixes finis, puisque $\widehat{\mathrm{T}}(\mathbb{P}) \subseteq \mathrm{T}(\mathbb{P})$. Par les préfixes finis, nous pourrons ensuite évaluer les préfixes infinis dont les arbres de jeu font partie. La table 3 résume donc les ingrédients nécessaires à l'évaluation de $\mathrm{IT}(\mathcal{S})$.

TAB. 4. Sous-algèbre syntaxique du joueur

$(\mathbb{P}_{\mathrm{T}}$		,	*positions terminales*
$\mathbb{P}_{\mathrm{NT}}$		,	*positions non terminales*
$\mathrm{T}(\mathbb{P})$		,	*arbres finis étiquetés dans* $\mathbb{P}$
$\widetilde{\mathrm{T}}_{\mathrm{p}}(\mathcal{S})$		,	*élagages finis du joueur*
$\widetilde{\mathrm{IT}}_{\mathrm{p}}(\mathcal{S})$		,	*élagages arbitraires du joueur* $(\cong \widetilde{\mathrm{T}}_{\mathrm{p}}(\mathcal{S})^{\flat})$
$\mathrm{IS}_{\mathrm{p}}(\mathcal{S})$		;	*stratégies (du joueur)*
$\subseteq_{\mathrm{p}}$	$: \ \forall\alpha{:}\{\widetilde{\mathrm{T}}_{\mathrm{p}}(\mathcal{S}), \widetilde{\mathrm{IT}}_{\mathrm{p}}(\mathcal{S})\}.\, 1^{\alpha\times\alpha}$	,	*ordre élagage du joueur*

TAB. 5. Sous-algèbre syntaxique de l'opposant

$(\mathbb{P}_{\mathrm{T}}$		,	*positions terminales*
$\mathbb{P}_{\mathrm{NT}}$		,	*positions non terminales*
$\mathrm{T}(\mathbb{P})$		,	*arbres finis étiquetés dans* $\mathbb{P}$
$\widetilde{\mathrm{T}}_{\mathrm{o}}(\mathcal{S})$		,	*élagages finis de l'opposant*
$\widetilde{\mathrm{IT}}_{\mathrm{o}}(\mathcal{S})$		,	*élagages arbitraires de l'opposant* $(\cong \widetilde{\mathrm{T}}_{\mathrm{o}}(\mathcal{S})^{\flat})$
$\mathrm{IS}_{\mathrm{o}}(\mathcal{S})$		;	*contre-stratégies (de l'opposant)*
$\subseteq_{\mathrm{o}}$	$: \ \forall\alpha{:}\{\widetilde{\mathrm{T}}_{\mathrm{o}}(\mathcal{S}), \widetilde{\mathrm{IT}}_{\mathrm{o}}(\mathcal{S})\}.\, 1^{\alpha\times\alpha}$	)	*ordre élagage de l'opposant*

4.5.2.2. *Sous-algèbre du joueur.* La table 4 définit les domaines et les opérations de la sous-algèbre syntaxique *du joueur*. Par cette sous-algèbre, notée $\mathsf{A}^{\mathrm{p}}_{\mathcal{S}}$, nous entendons cerner les éléments syntaxiques nécessaires à l'évaluation des *stratégies* des joueurs (finies ou infinies). Comme pour la sous-algèbre principale, l'évaluation des approximations finies, les compacts de $\widetilde{\mathrm{T}}_{\mathrm{p}}(\mathcal{S})$, nous permettra de définir l'évaluation des éléments arbitraires (compact ou pas) de $\widetilde{\mathrm{IT}}_{\mathrm{p}}(\mathcal{S})$.

4.5.2.3. *Sous-algèbre de l'opposant.* La sous-algèbre syntaxique *de l'opposant*, notée $\mathsf{A}^{\mathrm{o}}_{\mathcal{S}}$, définie dans la table 5, permet en revanche de délimiter les éléments syntaxiques nécessaires à l'évaluation des contre-stratégies. Comme pour les stratégies du joueur, l'évaluation des éléments compacts de $\widetilde{\mathrm{T}}_{\mathrm{o}}(\mathcal{S})$, permettra de définir l'évaluation des éléments arbitraires de $\widetilde{\mathrm{IT}}_{\mathrm{o}}(\mathcal{S})$.

4.6. Critères de classification syntaxique

4.6.1. Finitude. La classification des arbres *finis*, *infinis* et *rationnels*, se renouvèle sur les jeux, ces derniers étant des expressions indirectes (par l'image de l'application $\mathrm{t}^{\mathcal{S}}_{(.)}$) d'une forêt d'arbres.

DEFINITION 4.6.1. *Un jeu est* fini *si tous ses arbres de jeu légaux* $\mathrm{t}^{\mathcal{S}}_{(\mathbb{P}_{\mathrm{I}})}$ *le sont. En revanche, si au moins un de ces arbres est infini on dit que le jeu est* infini. *Et si tous ses arbres sont rationnels* $\mathrm{t}^{\mathcal{S}}_{(\mathbb{P}_{\mathrm{I}})} \subseteq \mathrm{RT}(\mathbb{P})$, *on dit que le jeu est* rationnel. $\square$

Le Morpion est un exemple de jeu fini, comme les Dames et les Échecs avec les règles modernes[3]. En revanche, dans leurs anciennes règles, n'imposant pas la conclusion du jeu, les Échecs représentent un bon exemple de jeu rationnel. En effet, comme nous l'avons observé dans 4.2.9, page 119, les positions du jeu peuvent paraître cycliquement à une infinité d'adresses de l'arbre des Échecs. Donc l'arbre est infini. Mais puisqu'il n'y a pas un nombre infini de positions du jeu (il y a un nombre fini de pièces et, donc, de configurations du damier), tout chemin de l'arbre aboutit inévitablement sur une position déjà atteinte dans le chemin. En d'autre termes, l'arbre de jeu a un nombre fini de sous-arbres ou, autrement dit, il est rationnel. Ce type d'argument, qui est valide pour les arbres de jeu, ne peut pas être utilisé en général pour les arbres construits arbitrairement sur un ensemble d'étiquettes F. Dans les jeux, puisque les règles ne changent pas, nous somme sûrs que deux noeuds distincts d'un arbre (u, π) et (v, π) partageant la même étiquette, auront la même suite, c'est-à-dire les mêmes successeurs. Donc, aboutir à l'adresse v, sur une position π, déjà atteinte auparavant à l'adresse u, implique l'égalité des branches u et v de l'arbre de jeu.

4.6.2. Alternance. La définition d'arène de jeu (cf. Def. 4.1.1) ne demande pas l'alternance stricte des joueurs. En effet, même si cela est le cas le plus fréquent, il existe aussi des jeux où les joueurs alternent, certes, mais le font sans régularité : il y a des positions où un joueur a le droit, après un premier coup, d'en jouer un second sans rendre la main à son adversaire.

Definition 4.6.2. (Jeux alternés) *Étant donné une arène de jeu $\mathcal{S} = (\mathbb{P}, \lambda, \rightarrow, \mathbb{P}_{\mathrm{I}})$ nous disons que les joueurs* alternent *ou que $\mathcal{S}$ est un jeu (*strictement*)* alterné *ou un jeu* tour à tour *si :*

$$\forall \pi \in \mathbb{P}. \qquad \pi \xrightarrow{\ell} \pi' \;\;\Rightarrow\;\; \lambda(\pi) \neq \lambda(\pi')$$

□

Le *Morpion*, les *Dames*, les *Échecs* et *Go*, entre autres, sont des exemples bien connus de jeux tour à tour. Dans d'autres jeux, en revanche, les règles impliquent une condition plus faible, une sorte d'alternance à long terme.

Definition 4.6.3. (Jeux inévitablement alternés) *Un jeu $\mathcal{S} = (\mathbb{P}, \lambda, \rightarrow, \mathbb{P}_{\mathrm{I}})$ nous disons que $\mathcal{S}$ est un jeu* inévitablement alterné *si :*

$$\forall \pi \in \mathbb{P}.\ \exists \delta \in \mathbb{N}. \quad \left\{ \begin{array}{l} \exists\, \pi \xrightarrow{u}{}^{*} \pi' \\ |u| \geq \delta \end{array} \right. \quad \Rightarrow \quad \pi' \in \mathbb{P}_{\mathrm{T}} \;\vee\; \left\{ \begin{array}{l} \exists\, \pi \xrightarrow{u'}{}^{*} \pi'' \xrightarrow{u''}{}^{*} \pi' \\ u = u'.u'' \\ \lambda(\pi) \neq \lambda(\pi'') \end{array} \right.$$

□

[3]Dans le passé ce n'était pas le cas, mais des règles on été ajoutées pour forcer la terminaison du jeu.

Dans un jeu inévitablement alterné il n'existe pas de séquences infinies de positions du même joueur. Nous remarquerons donc, que tout jeu fini est inévitablement alterné. *Othello* est un exemple de jeu fini, donc inévitablement alterné, mais pas strictement alterné. Dans ce jeu célèbre, il existe une règle pour laquelle tout mouvement des joueurs doit impliquer le retournement d'un pion de l'adversaire. Un joueur pourra ainsi se retrouver sans aucun mouvement possible, et il sera forcé de passer la main. Autrement dit, son adversaire jouera (au moins) deux coups à la suite. Nous retrouvons ce principe dans certains jeux de carte.

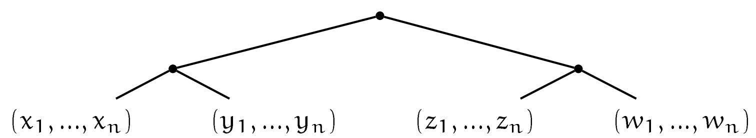
CHAPITRE 5

Sémantique des jeux à deux joueurs

Dans ce chapitre nous définissons la sémantique d'un jeu combinatoire à deux joueurs. Dans la théorie des jeux habituelle, où l'on assume que les arbres de jeu sont finis, la sémantique d'un jeu est fournie par une simple (et parfois implicite) fonction d'évaluation *inductive*. Dans cette hypothèse, le sens des stratégies et des contre-stratégies est également très simple à définir. En revanche, le relâchement de l'hypothèse de finitude complique, d'une part, la définition de sémantique d'un arbre de jeu complet, donc la sémantique d'une position, et, d'autre part, la définition de sémantique d'un élagage de l'arbre de jeu complet, tel qu'une stratégie ou tel qu'une contre-stratégie.

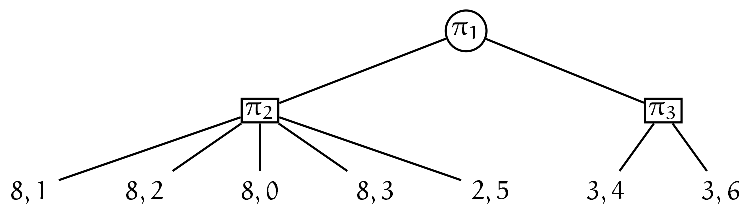
5.1. Introduction

À la fin d'une partie, il y a normalement soit une récompense soit, au contraire, un enjeu à payer pour les joueurs (sous forme d'argent, de prestige ou de satisfaction), qui dépend de la progression de la partie. On peut penser formaliser une telle situation avec une fonction qui assigne un coût, appelé traditionnellement *payoff*, à chaque position terminale du jeu. Dans le cas de n joueurs, le payoff est un n-uplet composé par les coûts de chaque participant.



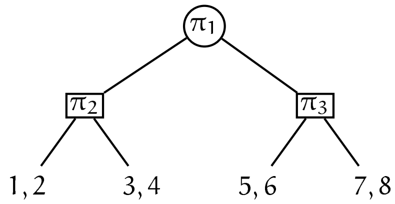
Le plus souvent l'ensemble des valeurs utilisées pour représenter un payoff individuel est très simple. Par exemple, l'ensemble des booléens $\{0, 1\}$, autrement dit, *gagné* (1) ou *perdu* (0). Ou alors, comme dans le *Morpion*, les *Dames* et les *Échecs*, l'ensemble $\{-1, 0, 1\}$ c'est-à-dire *gagné* (1), *perdu* (-1) ou *match nul* (0). Pour plusieurs jeux de cartes nous aurons besoin des entiers $\mathbb{Z}$ pour traduire les gains ou les pertes en argent des joueurs.

Quels que soient les domaines $D_1, \ldots, D_n$ utilisés pour les payoff individuels des n joueurs, la fonction de payoff assigne un coût dans le produit cartésien $D = D_1 \times \cdots \times D_n$. Le jeu est toujours étudié sur la base d'une présomption fondamentale de la théorie des jeux, l'*hypothèse de rationalité*, selon laquelle les joueurs choisiront, à tout moment du match, leur meilleur coup. Dans cette hypothèse, les joueurs sont non seulement rationnels mais, aussi, *égoïstes* : ils ont à coeur leur seul intérêt qui est *indépendant* de celui des autres. En d'autres termes, chaque participant fera le *maximum* vis-à-vis de son payoff, tout en sachant que ses adversaires se comporteront de la même façon quand ce sera leur tour. D'un point de vue mathématique, cela correspond à équiper les domaines d'une relation de préférence $\leq$ et d'une opération, que nous appellerons le $\mathtt{max}$, qui permettra de sélectionner le meilleur des coûts, donc le meilleur des coups disponibles. C'est pourquoi, nous réfléchirons sur des structures de type $(D_1, \leq_1, \max_1), \ldots, (D_n, \leq_n, \max_n)$. Pour illustrer cette hypothèse cardinale de la théorie, supposons qu'on aie un jeu à deux joueurs (où $D_1 = D_2 = \mathbb{N}$) se déployant de la façon suivante :

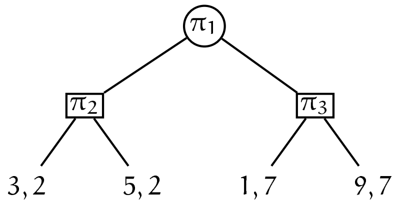


Le *joueur* se trouvant dans la position π_1, peut choisir entre deux coups, celui conduisant le jeu dans la position π_2 de l'*opposant*, ou celui conduisant le jeu dans la position π_3, toujours de l'*opposant*. Quel choix est le plus rentable dans la position π_1 ? On pourrait penser au premier, puisque la plupart des coups ensuite disponibles à l'*opposant* conduiraient le *joueur* vers son gain maximal, c'est-à-dire 8. Hélas, le *joueur* ne peut se permettre de faire un tel choix, il doit savoir que l'*opposant* ne joue pas au hasard mais tente, lui aussi, de maximiser son gain. Si l'*opposant* se retrouvait dans la position π_2 il n'hésiterait pas un seul instant et opterait pour son coup le plus rentable, pour lequel il gagnerait 5 faisant gagner 2 et non 8 au *joueur*. L'hypothèse de rationalité implique donc que le *joueur* choisira le second coup lui permettant d'assurer le gain 3.

Il peut arriver, accidentellement, que les intérêts des joueurs convergent, sans que l'hypothèse qui veut que les joueurs soient rationnels et égoïstes ne soit remise en discussion :



Dans cet exemple le match aboutit au payoff $(7,8)$ qui est maximal pour les deux joueurs. Si la convergence fortuite des intérêts ne pose pas de véritables problèmes, un second aspect peut compliquer l'évaluation d'un jeu. Si plusieurs coups sont équivalents, à un moment donné, pour le joueur qui a le choix (le joueur *actif*), mais ils ne le sont pas pour celui qui subit ce choix (le *passif*), nous aurons des difficultés pour estimer la valeur de payoff du joueur passif. Considérons l'exemple suivant :



Devant le *joueur* un choix difficile : s'il choisit à gauche, l'*opposant* se trouvera dans une position où ses deux coups disponibles seront équivalents à la valeur 2 ; de la même manière, s'il choisit à droite, l'*opposant* aura deux coups équivalents à 7. Qu'est-ce que fera l'*opposant* ? Il sera *indifférent* et choisira au *hasard* ? Il sera *gentil* et viendra au secours ? Est-ce qu'il profitera *méchamment* de toutes les occasions qui se présenteront pour affliger son concurrent ? En d'autre termes, s'il est clair que *l'opposant* fera le max sur sa composante, qu'est-ce qu'il fera sur l'autre ? La moyenne (indifférent), le max (gentil), ou le min (méchant) ?

Ce type de jeu trouve donc des réponses extrêmes, celles du *pire des cas* (adversaire méchant) et du *meilleur des cas* (adversaire gentil), ou la réponse intermédiaire, le *cas moyen* (adversaire indifférent). Il s'agit d'une classe de jeux qui peut être aussi abordée par la théorie, extrêmement développée, des jeux *répétés* ou *stochastiques*, où les joueurs réitèrent les parties et, ce faisant, il sont gentils ou méchants, c'est-à-dire coopératifs ou pas, selon l'expérience du comportement de leur adversaire observé dans le passé.

Nous ne nous occuperons plus de ce genre de jeu, où les intérêts peuvent converger, mais, nous nous placerons dans le cadre théorique des jeux dits *à somme nulle*, pour lesquels la *somme* des payoffs individuels des joueurs est *nulle*. Si la somme est nulle, les payoffs sont l'un l'opposé de l'autre, tout comme les intérêts des joueurs. D'un point de vue mathématique, cela revient à dire que, dans le cas de deux joueurs, les deux structures $(D_1, \leq_1, \max_1)$ et $(D_2, \leq_2, \max_2)$ ont le même domaine $D_1 = D_2$ mais avec l'ordre de préférence renversé : $x \leq_1 y \;\Leftrightarrow\; y \leq_2 x$, ce qui implique $\max_1 = \min_2$ et $\max_2 = \min_1$.

Dans les jeux à somme nulle, on peut évidemment se contenter de donner les $n-1$ premières composantes du payoff, ce qui est particulièrement utile pour les jeux à $n = 2$ joueurs. En effet, si la seconde composante des payoffs est toujours l'opposé de la première, nous pouvons la passer sous silence et ne nous occuper que de la première. Nous aurons alors un seul domaine d'évaluation, que nous appellerons D, avec un seul ordre de préférence, celui du *joueur*, que nous noterons $\leq$ et les deux opérations de sélection, celle du *joueur*, noté max, et celle de l'*opposant*, noté min. Pour cette raison, les jeux à deux joueurs à somme nulle sont appelés jeux *min-max*.

5.2. Évaluation traditionnelle des jeux finis

Pour l'évaluation d'un jeu fini, dont la syntaxe est $\mathcal{S} = (\mathbb{P}, \lambda, \rightarrow, \mathbb{P}_{\mathrm{I}})$, on peut imaginer que la technique que l'on met habituellement en oeuvre consiste à décorer l'arbre de jeu (fini) qui découle d'une position $\pi \in \mathbb{P}$, par les fonctions min et max sur les noeuds non terminaux, et par le *payoff* de la position terminale sur les feuilles de l'arbre. Ensuite, on imaginera un mécanisme de rétro-propagation des valeurs vers la racine par le calcul des opérations affectées aux noeuds de l'arbre. D'un point de vue mathématique cela correspond précisément à une définition inductive, où le calcul de la valeur d'un noeud renvoie au calcul des valeurs de ses fils. Nous supposerons que nous disposerons toujours d'une fonction indiquant le payoff, par exemple un entier naturel, de toute position terminale :

$$p : \mathbb{P}_{\mathrm{T}} \rightarrow \mathbb{N}$$

Sur cette fonction nous pouvons bâtir l'évaluation d'un jeu fini. En effet, ce qui prend traditionnellement le nom de *valeur* d'un jeu est une fonction $val : \mathbb{P} \rightarrow \mathbb{N}$ définie de la façon suivante :

$$val(\pi) = \begin{cases} p(\pi) & \text{si } \pi \in \mathbb{P}_{\mathrm{T}} \\ \max\limits_{i=1..n} \; val(\pi_i) & \text{si } \mathrm{Succ}(\pi) = \pi_1..\pi_n, \;\; \lambda(\pi) = \text{Player} \\ \min\limits_{i=1..n} \; val(\pi_i) & \text{si } \mathrm{Succ}(\pi) = \pi_1..\pi_n, \;\; \lambda(\pi) = \text{Opponent} \end{cases}$$

Une version équivalente nous reporte à la décoration imaginaire de l'arbre de jeu suivi du mécanisme de rétro-propagation des valeurs. Nous définissons la fonction $val : \mathrm{T}(\mathcal{S}) \rightarrow \mathbb{N}$ non pas sur les positions du jeu mais sur les arbres finis du jeu :

$$val(t) = \begin{cases} p(\pi) & \text{si } t = \{(\varepsilon, \pi)\} \\ \max\limits_{i=1..n} \; val(t_i) & \text{si } \mathrm{Fils}(t) = t_1..t_n, \;\; \lambda(t(\varepsilon)) = \text{Player} \\ \min\limits_{i=1..n} \; val(t_i) & \text{si } \mathrm{Fils}(t) = t_1..t_n, \;\; \lambda(t(\varepsilon)) = \text{Opponent} \end{cases}$$

La fonction $val : \mathbb{P} \rightarrow \mathbb{N}$ sur les positions pourra alors être redéfinie par l'intermédiaire de son analogue des arbres : $val(\pi) \triangleq val(t_\pi)$.

Dans les deux cas, les définitions sont, bien entendu, limitées au cas des jeux finis, où les *feuilles* des arbres constituent la base inductive. La motivation principale de ce chapitre est, en revanche, celle de donner une signification ou, autrement dit, une valeur, à tout type de jeu, fini ou infini. Dans ce sens, la deuxième technique est plus intéressante puisqu'elle suggère une façon d'évaluer un arbre infini : celle de considérer la valeur *limite* d'un processus qui, selon le schéma précédent, évaluerait des préfixes *finis* de l'arbre infini. Finalement, avant de discuter l'évaluation des jeux infinis, nous pouvons formaliser l'évaluation des jeux finis, où $\mathrm{IT}(\mathcal{S}) = \mathrm{T}(\mathcal{S})$.

5.2.1. Structures d'évaluation élémentaires. Dans le procédé d'abstraction des propriétés requises à la donnée d'une sémantique des jeux, nous commençons par présenter les éléments sémantiques nécessaires aux jeu finis.

DEFINITION 5.2.1. (STRUCTURE D'ÉVALUATION ÉLÉMENTAIRE) *Une* structure d'évaluation élémentaire *est une algèbre $\mathcal{D}$ sur la signature :*

$$\begin{array}{llll} (\mathsf{D}, & & & \textit{domaine d'évaluation} \\ \quad \uparrow & : \mathsf{D}^* \to \mathsf{D} & , & \textit{fonction du joueur} \\ \quad \downarrow & : \mathsf{D}^* \to \mathsf{D} & , & \textit{fonction de l'opposant} \\ \quad) & & & \end{array}$$

□

Les opérations $\uparrow$: $\mathsf{D}^* \to \mathsf{D}$ et $\downarrow$: $\mathsf{D}^* \to \mathsf{D}$, d'arité variable, qui généralisent le max et le min, sont appelées respectivement fonction du *joueur* et fonction de l'*opposant*. Une structure d'évaluation élémentaire permet, avec une fonction de *payoff* $p : \mathbb{P}_T \to \mathsf{D}$ évaluant les noeuds terminaux, d'attribuer une valeur dans D à tout jeu fini.

DEFINITION 5.2.2. (VALEUR D'UN JEU FINI) *Étant donné une arène de jeu* $\mathcal{S} = (\mathbb{P}, \lambda, \to, \mathbb{P}_I)$*, une structure d'évaluation élémentaire* $(\mathsf{D}, \uparrow, \downarrow)$*, et une fonction de payoff* $p : \mathbb{P}_T \to \mathsf{D}$*, la* valeur d'un arbre fini *du jeu est la fonction* $val : \mathrm{T}(\mathcal{S}) \to \mathsf{D}$ *définie de la façon suivante :*

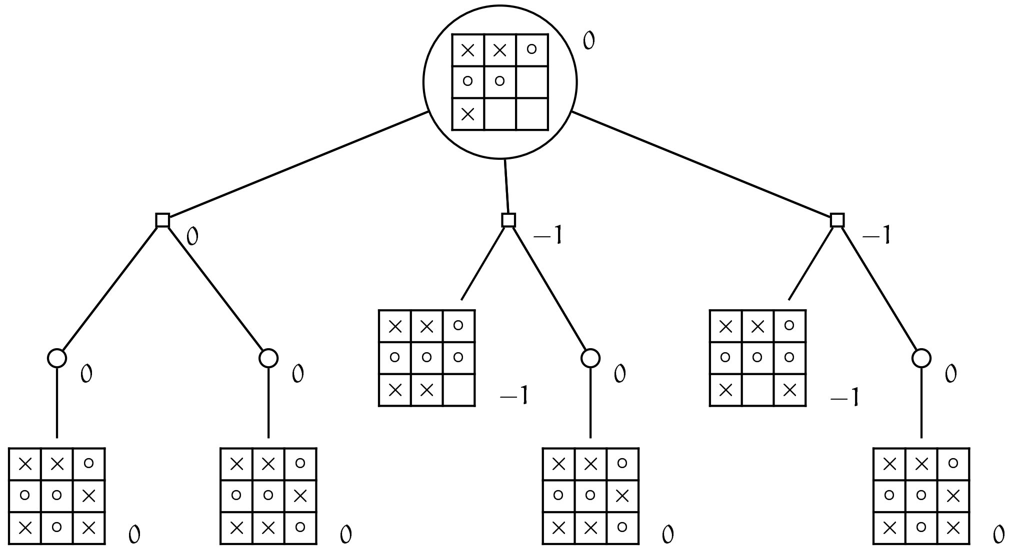
$$val(t) \triangleq \begin{cases} \bot & \text{si } t = \emptyset \\ p(\pi) & \text{si } t = \{(\varepsilon, \pi)\} \\ \underset{i=1..n}{\uparrow} val(t_i) & \text{si Fils}(t) = t_1..t_n,\ \ \lambda(t(\varepsilon)) = \text{Player} \\ \underset{i=1..n}{\downarrow} val(t_i) & \text{si Fils}(t) = t_1..t_n,\ \ \lambda(t(\varepsilon)) = \text{Opponent} \end{cases}$$

□

EXAMPLE 5.2.3. Dans le cas d'un jeu comme le Morpion, la structure d'évaluation est composée de trois valeurs seulement : gagné (+1), perdu (−1) et match nul (0). Les opérations des deux joueurs sont le max et le min (restreintes à l'ensemble des trois valeurs du jeu). Dans la position initiale de la figure 5.2.1, le joueur n'a pas de stratégie gagnante, au contraire il doit choisir le coup à gauche sur la figure pour espérer décrocher le match nul. En d'autres termes, la valeur de la position initiale de l'arbre, obtenue par le mécanisme de rétro-propagation des valeurs, est égale à 0, qui dénote le match nul.

□

FIG. 5.2.1. Valeur d'une branche du Morpion



5.3. Interprétation d'un jeu arbitraire

Supposons qu'on veuille évaluer un préfixe fini $\mathtt{t}'$ d'un arbre de jeu $\mathtt{t}$: $\mathtt{t}' \leq \mathtt{t}$. Par définition de préfixe, l'arbre $\mathtt{t}'$ est un élagage complet ou radical à toutes ses adresses. En d'autres termes, les feuilles de $\mathtt{t}$ sont forcément des feuilles de $\mathtt{t}'$, mais pas réciproquement, les feuilles de $\mathtt{t}'$ pouvant correspondre à un élagage radical de $\mathtt{t}$ aux adresses concernées. Pour évaluer un préfixe, il faudra alors savoir évaluer des *feuilles* ne correspondant pas à des positions *terminales* du jeu. Autrement dit, il faudra savoir approximer la valeur des positions non terminales du jeu.

Le problème de l'approximation de la valeur d'une position d'un jeu n'est pas une question qui se pose exclusivement pour les jeux infinis. On la rencontre aussi, dans la théorie des jeux combinatoires, pour estimer la valeur d'un jeu fini dont l'arbre de jeu est trop grand pour être évalué précisément (ceci est le cas, par exemple, des Échecs avec les règles modernes). Dans un tel contexte, un algorithme explorant l'arbre à la recherche d'une stratégie gagnante (tel que *Deep Blue* pour les Échecs), contraint à répondre dans un temps considéré comme raisonnable pour un être humain, sera souvent forcé d'arrêter ses recherches à une profondeur inférieure à la hauteur de l'arbre. Il sera alors obligé d'*estimer* la valeur d'une branche inexplorée, au lieu de l'évaluer avec précision.

Si dans la problématique du robot joueur d'Échecs nous pouvons imaginer plusieurs façons heuristiques d'estimer la valeur d'une position, plus au moins correctes vis-à-vis de la véritable valeur, dans la question d'attribuer une sémantique à tout jeu infini, nous devons, en revanche, procéder d'une façon rigoureuse d'un point de vue mathématique. La façon d'approximer les positions non terminales respectera, intuitivement, une règle dictée par un bon sens mathématique : plus l'exploration de l'arbre avancera en largeur et profondeur, et plus l'approximation de sa valeur sera fidèle. Plus précisément, nous dirons que les heuristiques devront respecter la relation préfixe entre arbres : en évaluant deux préfixes $\mathtt{t}_1$ et $\mathtt{t}_2$ d'un arbre infini $\mathtt{t}$, eux-mêmes dans la relation préfixe $t_1 \leq t_2$, nous nous attendrons à ce que l'approximation résultant de la visite de t_1 soit moins informative que celle correspondant à t_2. Autrement dit, nous nous attendrons à ce que les valeurs soient, elles aussi, dans une relation $\leq_{\mathrm{D}}$ définie sur le domaine d'évaluation D et *homomorphe* à la relation préfixe :

$$t_1 \leq t_2 \quad \Rightarrow \quad v(t_1) \leq_{\mathrm{D}} v(t_2)$$

La notion d'*interprétation d'un jeu* est donc celle, habituelle, d'interprétation algébrique. Au départ de cette projection, que nous demandons à être un homomorphisme, nous retrouvons l'*algèbre syntaxique principale* du jeu, dont l'objectif est celui de définir les éléments nécessaires à l'évaluation des arbres de jeu $t \in \mathrm{IT}(\mathcal{S})$;

$$(\mathbb{P}_{\mathrm{T}}, \mathbb{P}_{\mathrm{NT}}, \mathrm{T}(\mathbb{P}), \widehat{\mathrm{T}}(\mathcal{S}), \widehat{\mathrm{IT}}(\mathcal{S}), \mathrm{IT}(\mathcal{S})\,;\, \leq)$$

Tous les éléments syntaxiques seront traduits dans l'algèbre *sémantique* du jeu, une algèbre d'évaluation arbitraire dont la signature sera :

$$\mathcal{D} = (\mathrm{D}, \mathrm{D}^{\flat}\,;\, \leq, \uparrow, \downarrow, \perp)$$

Si nous appelons D_π l'ensemble des valeurs résultant de l'évaluation d'un préfixe quelconque de la position π, nous verrons que tout arbre de jeu infini $t \in \mathrm{IT}(\mathcal{S}) \subseteq \widehat{\mathrm{IT}}(\mathcal{S}) \cong \widehat{\mathrm{T}}(\mathcal{S})^{\flat}$ a un signifié dans l'ensemble des dirigés de D_π modulo co-finalité (ou, autrement dit, dans la complétion par idéaux de D_π, cf. corollaire 2.5.6). Et lorsque la sous-algèbre $(\mathrm{D}_\pi, \leq_{\mathrm{D}_\pi})$ sera un dcpo algébrique, un tel signifié pourra être considéré, à un isomorphisme près, comme étant une valeur de D_π. Nous allons donc définir l'ensemble des fonctions composant l'interprétation des valeurs (qui, nous le rappelons, est une construction polymorphe $\varphi : \forall\alpha{:}\mathrm{S}.\, \alpha \to \psi(\alpha)$, où S est l'ensemble des sortes, cf. section 1.3.2), c'est-à-dire :

(1) l'interprétation des étiquettes (les positions terminales $\mathbb{P}_{\mathrm{T}}$ ou non terminales $\mathbb{P}_{\mathrm{NT}}$) ;

(2) l'interprétation des arbres étiquetés dans $\mathbb{P}$, qui incluent les préfixes finis ;

(3) l'interprétation des préfixes infinis, qui incluent $\mathrm{IT}(\mathcal{S})$.

Pour construire un tel homomorphisme on demande à la structure d'évaluation de contenir une relation de comparaison entre valeurs et des opérations des joueurs avec des propriétés de monotonie.

5.3.1. Interprétation des étiquettes. Le couple constitué de l'interprétation des positions terminales (fonction de payoff) et de l'interprétation des positions non terminales d'un jeu (fonction heuristique), doit constituer la partie essentielle de l'homomorphisme de l'algèbre syntaxique vers l'algèbre d'évaluation du jeu.

DEFINITION 5.3.1. (PAYOFF, HEURISTIQUE) *Étant donné une arène de jeu* $\mathcal{S} = (\mathbb{P}, \lambda, \to, \mathbb{P}_{\mathrm{I}})$*, et un domaine de valeurs* D*, une* fonction de payoff *est une fonction de type* $p : \mathbb{P}_{\mathrm{T}} \to \mathrm{D}$*, une* fonction heuristique *est une fonction de type* $h : \mathbb{P}_{\mathrm{NT}} \to \mathrm{D}$.
□

5.3.2. Interprétation des arbres finis décorés. Les opérations des joueurs $\uparrow$ et $\downarrow$ définies dans une *structure d'évaluation élémentaire* permettent, avec les fonctions d'évaluation des positions terminales (le *payoff*) et non terminales (l'*heuristique*), d'évaluer tout arbre fini étiqueté dans $\mathbb{P}$ avec l'habituel mécanisme de rétro-propagation des valeurs, à partir des feuilles, vers la racine de l'arbre. Si le noeud est terminal, la position est évaluée avec l'heuristique ou avec la fonction de payoff selon le statut (terminale ou non terminale) de la position qui décore le noeud. Si le noeud est non terminal, le statut terminal ou non terminal (dans le jeu) de la position qui le décore est ignoré. L'évaluation du noeud dépend exclusivement du joueur qui a la main dans la position du noeud. La fonction associée à ce joueur ($\uparrow$ ou bien $\downarrow$) est appliquée pour composer les évaluations des fils de l'arbre.

DEFINITION 5.3.2. (RÉTRO-PROPAGATION MIN-MAX)*Étant donné une arène de jeu* $\mathcal{S} = (\mathbb{P}, \lambda, \to, \mathbb{P}_I)$*, une structure d'interprétation élémentaire* $(D, \uparrow, \downarrow)$*, une fonction de payoff* $p : \mathbb{P}_T \to D$ *et une heuristique* $h : \mathbb{P}_{NT} \to D$*,* l'interprétation inductive de $T(\mathbb{P})$ *(ou* rétro-propagation min-max*) est la fonction* $v_{p,h} : T(\mathbb{P}) \to D$ *définie de la façon suivante :*

$$v_{p,h}(t) \triangleq \begin{cases} \perp & \text{si } t = \emptyset \\ p(\pi) & \text{si } t = \{(\varepsilon, \pi)\},\ \pi \in \mathbb{P}_T \\ h(\pi) & \text{si } t = \{(\varepsilon, \pi)\},\ \pi \in \mathbb{P}_{NT} \\ \underset{i=1..n}{\uparrow} v_{p,h}(t_i) & \text{si } \mathrm{Fils}(t) = t_1..t_n,\ \ \lambda(t(\varepsilon)) = \text{Player} \\ \underset{i=1..n}{\downarrow} v_{p,h}(t_i) & \text{si } \mathrm{Fils}(t) = t_1..t_n,\ \ \lambda(t(\varepsilon)) = \text{Opponent} \end{cases}$$

□

Ce mécanisme inductif permet, en particulier, d'évaluer aussi bien les arbres finis du jeu $T(\mathcal{S})$ que les préfixes finis du jeu $\widehat{T}(\mathcal{S})$ et les élagages finis du joueur $\widetilde{T}_p(\mathcal{S}, \pi)$ et de l'opposant $\widetilde{T}_o(\mathcal{S}, \pi)$.

DEFINITION 5.3.3. (VALEURS D'UNE POSITION) *Étant donné une arène de jeu* $\mathcal{S} = (\mathbb{P}, \lambda, \to, \mathbb{P}_I)$*, une structure d'interprétation élémentaire* $(D, \uparrow, \downarrow)$*, une fonction de payoff* $p : \mathbb{P}_T \to D$ *et une heuristique* $h : \mathbb{P}_{NT} \to D$*, l'ensemble des* valeurs (des préfixes finis) *d'une position* $\pi \in \mathbb{P}$*, noté* $D_\pi^{p,h}$*, est l'image de la rétro-propagation min-max de tous les préfixes finis de l'arbre de la position :*

$$D_\pi^{p,h} \triangleq \{v_{p,h}(t) \mid t \in \widehat{T}(\mathcal{S}, \pi)\}$$

L'ensemble des valeurs des élagages finis (du joueur) *dans* π*, noté* $\widetilde{D}_\pi^{p,h}$*, est l'image de la rétro-propagation min-max de tous les élagages finis du joueur :*

$$\widetilde{D}_\pi^{p,h} \triangleq \{v_{p,h}(t) \mid t \in \widetilde{T}_p(\mathcal{S}, \pi)\}$$

□

NOTATION 5.3.4. Pour alléger la notation, lorsque il n'y aura pas d'ambiguïté possible sur les fonctions de payoff et heuristique utilisées, nous écrirons D_π et $\widetilde{D}_\pi$ au lieu de $D_\pi^{p,h}$ et $\widetilde{D}_\pi^{p,h}$. Nous ferons de même avec la rétro-propagation min-max, que nous noterons plus simplement par l'écriture v.

Si l'ensemble des valeurs des préfixes finis D_π permet de caracteriser les structures d'évaluation qui donnent un sens aux arbres de jeu infinis, l'ensemble des valeurs des élagages finis du joueur $\widetilde{D}_\pi$ permet, en revanche, de caractériser les structures qui donnent un sens aux élagages du joueur et donc, en particulier, à ces stratégies.

5.3.3. Structures d'évaluation monotones. Nous allons voir comment les valeurs des préfixes finis d'une position π peuvent converger vers une limite que nous appellerons *valeur* (ou *sémantique*) de π. Nous pouvons généraliser une propriété des fonctions max et min utilisées habituellement pour l'évaluation d'un jeu.

DEFINITION 5.3.5. (MONOTONIE DANS LES COMPOSANTES) *Étant donné un préordre* $(D, \leq)$*, une fonction* $f : D^* \to D$ *est* monotone *(ou* croissante*)* dans toutes ses composantes *si :*

$$\forall u, v \in D^*.\ \forall x, y \in D. \quad x \leq y \quad \Rightarrow \quad f(u.x.v) \leq f(u.y.v)$$

□

Une fonction $f : D^* \to D$ est monotone dans toutes les composantes, si pour tout couple de suites $u, v \in D^*$ identiques à l'exception d'une composante : $w = u.x.v$ et $w' = u.y.v$, où $x \leq y$, nous avons $f(w) \leq f(w')$. Le nom que nous avons utilisé pour cette propriété a une signification précise : il s'agit bien de la propriété de monotonie habituelle (homomorphisme) des fonctions entre préordres, où la relation en question est l'extension produit de $\leq$ aux suites de D (cf. section 1.1.8), que nous noterons $\leq_*$. Donc, une fonction est monotone dans toutes ses composantes s'il s'agit d'un homomorphisme de type :

$$f : (D^*, \leq_*) \xmapsto{\mathcal{H}} (D, \leq)$$

Cette forme de monotonie est la première et principale propriété que nous attendons des fonctions généralisant le max et le min. Il est cependant inutile, pour un jeu donné, de demander une telle propriété sur toutes les valeurs possibles du domaine d'évaluation. Une version plus faible de la monotonie dans les composantes fait référence aux seules valeurs du jeu.

DEFINITION 5.3.6. (STRUCTURE D'ÉVALUATION MONOTONE POUR UN JEU) *Étant donné :*

- *une arène de jeu* $\mathcal{S} = (\mathbb{P}, \lambda, \to, \mathbb{P}_I)$,
- *une structure d'inteprétation élémentaire* $(D, \uparrow, \downarrow)$,
- *une fonction de payoff* $p : \mathbb{P}_T \to D$ *et une heuristique* $h : \mathbb{P}_{NT} \to D$,

une structure d'évaluation (monotone) pour le jeu $\mathcal{S}$*, est une algèbre* $\mathcal{D}$*, contentant la structure élémentaire, de signature :*

$$\begin{array}{llll} (D, & & & \textit{domaine d'évaluation} \\ \quad \leq_D & : & 1^{D\times D} \quad , & \textit{relation sur les valeurs} \\ \quad \uparrow & : & D^* \to D \quad , & \textit{fonction du joueur} \\ \quad \downarrow & : & D^* \to D \quad , & \textit{fonction de l'opposant} \\ \quad) & & & \end{array}$$

et telle que :

(1) *pour toute position* $\pi \in \mathbb{P}$ *du jeu* $(D_\pi, \leq_{D_\pi})$ *est un préordre*

(2) $\uparrow$ *et* $\downarrow$ *sont monotones dans toutes leurs composantes par rapport au valeurs du jeu, i.e. pour toute position non terminale* $\pi \in \mathbb{P}_{NT}$*, où* $\mathrm{Succ}(\pi) = \pi_1..\pi_n$*, les fonctions vérifient la propriété :*

$$\forall u \in D_{\pi_1}..D_{\pi_{k-1}}, v \in D_{\pi_{k+1}}..D_{\pi_n}\ .$$

$$x \leq_{D_{\pi_k}} y \Rightarrow \begin{cases} \uparrow (u.x.v) \leq_{D_\pi} \uparrow (u.y.v) \\ \\ \downarrow (u.x.v) \leq_{D_\pi} \downarrow (u.y.v) \end{cases}$$

(3) *l'heuristique est* correcte*, c'est-à-dire, pour toute position* $\pi \in \mathbb{P}_{NT}$*, telle que* $\mathrm{Succ}(\pi) = \pi_1..\pi_n$ $(n \geq 1)$*, la fonction vérifie les conditions suivantes :*

$$\lambda(\pi) = \text{Player} \quad \Rightarrow \quad h(\pi) \leq_{D_\pi} \left(\mathop{\uparrow}_{i=1..n} p \oplus h(\pi_i) \right)$$

$$\lambda(\pi) = \text{Opponent} \quad \Rightarrow \quad h(\pi) \leq_{D_\pi} \left(\mathop{\downarrow}_{i=1..n} p \oplus h(\pi_i) \right)$$

La famille des algèbres d'évaluation monotones sur un jeu $\mathcal{S}$*, une structure élémentaire* $(D, \uparrow, \downarrow)$*, un payoff* p *et une heuristiques* h*, sera noté* $\mathsf{Estruct}(\mathcal{S}, D, \uparrow, \downarrow, p, h)$*. Dans le cas particulier où la relation* $\leq_D$ *est un préordre, un ordre partiel ou un dcpo, nous dirons que* $\mathcal{D}$ *est, respectivement, un* préordre d'évaluation*, un* ordre d'évaluation *ou un* dcpo d'évaluation*. Si* $(D, \leq_D)$ *est un treillis et les opérations* $\uparrow$ *et* $\downarrow$ *coïncident respectivement avec la borne supérieure et la borne inférieure du treillis, alors la structure d'évaluation est appelée* treillis d'évaluation*.* □

Pour qu'une algèbre soit une structure d'évaluation, il faudra, en particulier, que l'heuristique vérifie la propriété de correction vis-à-vis des préordres $(D_\pi, \leq_{D_\pi})$. Autrement dit, pour les feuilles d'un arbre qui ne contiennent pas une position terminale du jeu, l'heuristique doit fournir une information moins précise (inférieure) par rapport à celle qu'on pourrait obtenir en développant en peu plus l'arbre à ces mêmes adresses.

Si pour les jeux finis une structure d'évaluation élémentaire suffit pour attribuer un sens aux positions du jeu, dans le cas général des jeux infinis, nous aurons besoin de valeurs correspondant aux arbres infinis. Une structure d'évaluation monotone n'est toujours pas suffisante. Nous avons remarqué que les arbres infinis sont isomorphes à la complétion par idéaux des arbres finis (cf. prop. 4.3.4). La construction sémantique analogue sera la complétion par idéaux, c'est-à-dire l'extension bémol, du préordre $(D_\pi, \leq_{D_\pi})$ constituant une structure d'évaluation. Nous obtiendrons ainsi une algèbre qui, par rapport à la structure d'évaluation, aura un nouvel ensemble $D^\flat_\pi$ et une nouvelle opération, le semi-prédicat $\leq^\flat_{D_\pi}$ sur les classes de dirigés co-finaux.

Intuitivement, si les préfixes du jeu constituent un domaine ayant de bonnes propriétés algébriques (il s'agit d'un dcpo d'ensembles), la *structure* d'évaluation pourrait, elle, ne pas avoir d'aussi bonnes propriétés. La construction bémol, qui donne lieu à l'*algèbre* d'évaluation, nous mettra à l'abri d'un tel manque de propriétés de complétude[1].

DEFINITION 5.3.7. (ALGÈBRE D'ÉVALUATION MONOTONE POUR UN JEU) *Étant donné une structure d'évaluation monotone* $\mathcal{D} \in \mathsf{Estruct}(\mathcal{S}, \mathrm{D}, \uparrow, \downarrow, \mathrm{p}, \mathrm{h})$, $\mathcal{D} = (\mathrm{D}, \leq_{\mathrm{D}}, \uparrow, \downarrow)$, *où* $\mathcal{S} = (\mathbb{P}, \lambda, \to, \mathbb{P}_{\mathrm{I}})$, l'algèbre d'évaluation (monotone) du jeu $\mathcal{S}$, *est l'algèbre* $\mathcal{D}^\flat$ *contenant la sous-algèbre* $\mathcal{D}$, *de signature :*

$$
\begin{array}{llllll}
(\{\mathrm{D}, \mathrm{D}^\flat\}, & \{ & & & & \\
 & \leq & : & \forall\alpha\!:\mathsf{sort}.\ 1^{\alpha\times\alpha} & , & \textit{relation sur } \mathrm{D} \textit{ et } \mathrm{D}^\flat \\
 & \uparrow & : & \mathrm{D}^* \to \mathrm{D} & , & \textit{fonction du joueur} \\
 & \downarrow & : & \mathrm{D}^* \to \mathrm{D} & , & \textit{fonction de l'opposant} \\
 & \}) & & & &
\end{array}
$$

où la sous-structure $(\mathrm{D}^\flat, \leq^\flat)$ *est la réunion des extensions bémol des différentes positions :*

$$
\mathrm{D}^\flat \triangleq \bigcup_{\pi\in\mathbb{P}} \mathrm{D}^\flat_\pi
$$

$$
\leq^\flat \triangleq \bigcup_{\pi\in\mathbb{P}} \leq^\flat_\pi
$$

où pour toute position du jeu $\pi \in \mathbb{P}$, *la structure* $(\mathrm{D}^\flat_\pi, \leq^\flat_{\mathrm{D}_\pi})$, *où* $(\leq^\flat_{\mathrm{D}_\pi}) \triangleq (\leq_{\mathrm{D}^\flat_\pi})$, *est l'extension bémol du préordre* $(\mathrm{D}_\pi, \leq_{\mathrm{D}_\pi})$.

Dans le cas spécifique où $(\mathrm{D}, \leq_{\mathrm{D}}, \uparrow, \downarrow)$ *est un treillis d'évaluation, l'algèbre d'évaluation correspondante est appelée* treillis d'évaluation étendu. □

La sous-algèbre $\mathcal{D} = (\mathrm{D}, \leq_{\mathrm{D}}, \uparrow, \downarrow)$, qui définit complètement $\mathcal{D}^\flat$, est le *germe* de l'*algèbre* d'évaluation $\mathcal{D}^\flat = (\mathrm{D}, \mathrm{D}^\flat; \leq_{\mathrm{D}}, \leq^\flat_{\mathrm{D}}, \uparrow, \downarrow)$. Lorsque tout préordre $(\mathrm{D}_\pi, \leq_{\mathrm{D}_\pi})$ est un dcpo algébrique, son extension bémol $(\mathrm{D}^\flat_\pi, \leq^\flat_{\mathrm{D}_\pi})$ lui est isomorphe. Cela arrive, en particulier, lorsque la sous-structure $(\mathrm{D}, \leq_{\mathrm{D}})$ est un dcpo algébrique. Dans les deux cas, l'algèbre d'évaluation n'a pas d'intérêt majeur par rapport à son germe.

5.3.4. Heuristiques correctes. La proposition suivante montre que la définition d'heuristique correcte généralise la définition habituelle d'heuristique correcte (cf. [**Pearl, 1990**]), qui est, elle, limitée au cadre des jeux finis (et interprétés dans la structure des entiers avec les opérations max et min).

PROPOSITION 5.3.8. *Dans une structure d'évaluation monotone* $\mathcal{D} \in \mathsf{Estruct}(\mathcal{S}, \mathrm{D}, \uparrow, \downarrow, \mathrm{p}, \mathrm{h})$, *l'heuristique correcte* $\mathrm{h} : \mathbb{P}_{\mathrm{NT}} \to \mathrm{D}$ *est telle que* $\mathrm{h}(\pi) \leq_{\mathrm{D}_\pi} v(\mathrm{t})$ *pour tout préfixe fini d'un arbre de jeu.*

DÉMONSTRATION. *Par induction sur la hauteur de l'arbre* t *qui, nous le rappelons, est la longueur maximale des adresses de l'arbre.*

[1] Ce sera effectivement le cas du domaine des contraintes pour la Programmation Logique avec Contraintes (cf. chapitre 8)

– $\text{hauteur}(t_\pi) = 1$

Dans ce cas $\forall\pi \to \pi'.\ \pi' \in \mathbb{P}_T$, *donc* $p \oplus h(\pi') = p(\pi')$. *Supposons* $\lambda(\pi) = \text{Player}$, *l'autre cas étant dual, et* $\text{Succ}(\pi) = \pi_1..\pi_n$. *Puisque l'heuristique est correcte, nous avons :*

$$h(\pi) \leq_{D_\pi} \mathop{\uparrow}_{i=1..n} p \oplus h(\pi_i) = \mathop{\uparrow}_{i=1..n} p(\pi_i) = v(t)$$

– $\text{hauteur}(t_\pi) > 1$

Soit $\text{Succ}(\pi) = \pi_1..\pi_n$ *et supposons* $\lambda(\pi) = \text{Player}$. *Puisque l'heuristique est correcte, nous avons* $h(\pi) \leq_D \mathop{\uparrow}_{i=1..n} p \oplus h(\pi_i)$. *Pour chaque argument de* $\uparrow$ *deux cas mutuellement exclusifs sont possibles. Le premier est que* $\pi_i \in \mathbb{P}_T$, *donc* $p \oplus h(\pi_i) = p(\pi_i) = v(\pi_i)$. *Le second est que* $\pi_i \in \mathbb{P}_{NT}$, *donc* $p \oplus h(\pi_i) = h(\pi_i) \leq_D v(\pi_i)$ *par hypothèse d'induction. Dans les deux cas nous avons* $p \oplus h(\pi_i) \leq_D v(\pi_i)$ *ce qui implique l'énoncé par monotonie de la fonction* $\uparrow$. *Le cas dual* $\lambda(\pi) = \text{Opponent}$ *s'appuie sur la monotonie de* inf.

□

COROLLARY 5.3.9. *Dans une structure d'évaluation monotone* $\mathcal{D} \in \mathsf{Estruct}(\mathcal{S}, D, \uparrow, \downarrow, p, h)$, *l'heuristique correcte* $h : \mathbb{P}_{NT} \to D$ *est telle que, si l'arbre* t_π *d'une position non terminale est fini, alors* $h(\pi) \leq_{D_\pi} v(t_\pi)$. □

En d'autre termes, si la correction de l'heuristique est une condition *locale* (i.e. portant sur la valeur engendrée par le plus petit préfixe supérieur à la feuille), par effet de la monotonie des opérations des joueurs elle devient *globale* (i.e. portant sur toutes les valeurs engendrées par la position). Cela signifie que la valeur heuristique d'une position non terminale $\pi \in \mathbb{P}_{NT}$ est précisément l'élément minimum de l'ensemble des valeurs de la position :

$$h(\pi) = \bot_\pi = \min_{\leq_{D_\pi}} D_\pi.$$

Si la structure est un préordre d'évaluation avec un élément minimal $\bot$, et l'heuristique est telle que $h(\pi) = \bot$, nous disons qu'il s'agit de l'heuristique *canonique* ou *banale* (elle sera trivialement correcte).

Il existe une notion de qualité évidente pour une fonction heuristique. Plus la valeur estimée approche la valeur réelle, meilleure est l'heuristique.

DEFINITION 5.3.10. (HEURISTIQUES PLUS INFORMÉES) *Soit* $\mathcal{S} = (\mathbb{P}, \lambda, \to, \mathbb{P}_I)$ *une arène de jeu et* $\mathcal{D} = (D, \leq_D, \uparrow, \downarrow, \bot)$ *une structure qui est à la fois une structure d'évaluation pour deux heuristiques différentes :*

$$\mathcal{D} \in \mathsf{Estruct}(\mathcal{S}, D, \uparrow, \downarrow, p, h_1)$$
$$\mathcal{D} \in \mathsf{Estruct}(\mathcal{S}, D, \uparrow, \downarrow, p, h_2)$$

Alors, si le couple d'heuristiques (h_1, h_2) *est dans l'ordre extensionnel des fonctions de type* $\mathbb{P}_{NT} \to D$, *nous disons que* h_2 *est* plus informée *que* h_1, *noté* $h_1 \leq h_2$. □

Autrement dit, h_2 est est plus informée que h_1, si $\forall\pi \in \mathbb{P}_{NT}.\ h_1(\pi) \leq_D h_2(\pi)$. Nous verrons par la suite que lorsque deux heuristiques en relation $h \leq h'$ sont correctes vis-à-vis d'une fonction de payoff, il se trouve que l'heuristique plus informée h' permet d'accélérer l'évaluation d'une position et, parfois, de terminer l'évaluation alors qu'en utilisant l'heuristique moins informée h nous n'aurions pu répondre en temps fini.

5.3.5. Interprétation des préfixes finis. La notion d'heuristique correcte est cruciale pour que la fonction v, restreinte aux préfixes finis du jeu, soit un homomorphisme. Lorsque deux arbres sont dans la relation préfixe $t \leq t'$, nous pouvons voir t' comme une meilleure approximation, par rapport à t, de l'arbre (possiblement infini) d'une position du jeu. À un effort majeur de développement de l'arbre correspondra nécessairement, si l'heuristique est correcte, une meilleure évaluation de la position. Autrement dit, la valeur $v(t')$ sera supérieure, au sens de l'ordre sur le domaine d'évaluation, à la valeur $v(t)$.

DEFINITION 5.3.11. *Étant donné une structure d'évaluation monotone* $\mathcal{D} \in \mathsf{Estruct}(\mathcal{S}, D, \uparrow, \downarrow, p, h)$, *l'*interprétation des préfixes finis *est la restriction* $v : \widehat{T}(\mathcal{S}) \to D$ *de l'interprétation inductive de* $T(\mathbb{P})$ *au sous-ensemble des préfixes finis du jeu.* □

PROPOSITION 5.3.12. *Soit* $\mathcal{D} \in \mathsf{Estruct}(\mathcal{S}, D, \uparrow, \downarrow, p, h)$ *une structure d'évaluation monotone. Alors, l'interprétation des préfixes finis est une fonction monotone :*

$$v : (\widehat{T}(\mathcal{S}, \pi), \leq) \overset{\mathcal{H}}{\longmapsto} (D_\pi, \leq_{D_\pi})$$

DÉMONSTRATION. *Nous pouvons comparer les arbres par leur hauteur. La relation* $\prec \triangleq \{(t', t) \mid \text{hauteur}(t) <_{\mathbb{N}} \text{hauteur}(t)\}$ *est une relation bien fondée sur les préfixes finis* $\prec : \wp^{\widehat{T}(\mathcal{S}) \times \widehat{T}(\mathcal{S})}$. *Son extension produit aux couples de préfixes finis* $\prec_2 : \wp^{\widehat{T}(\mathcal{S})^2 \times \widehat{T}(\mathcal{S})^2}$ *est alors, elle aussi, une relation bien fondée (cf. section 1.1.11). Nous prouvons l'énoncer par le principe d'induction bien fondée sur* $\prec_2$. *Soient* $t', t \in \widehat{T}(\mathcal{S}, \pi)$ *tels que* $t' \leq t$. *Nous voulons montrer que* $v(t') \leq_{D_\pi} v(t)$. *Si* t *est vide, alors* t' *doit l'être aussi (tout préfixe est un élagage :* $t' \subseteq t$*), donc l'énoncé est vrai. Si* $t = \{(\varepsilon, \pi)\}$, *alors soit* $t' = \emptyset$, *soit* $t' = t$ *; dans les deux cas* $v(t') \leq_{D_\pi} v(t)$. *Si* $\text{Fils}(t) = t_1..t_n$, *supposons* $\lambda(t(\varepsilon)) = \text{Player}$. *Par définition, un préfixe est un élagage complet ou radical à toute adresse. Il y a alors trois cas. Pour le premier,* $t' = \emptyset$, *l'énoncé est trivialement vrai. Pour le second,* $t' = \{(\varepsilon, \pi)\}$ *(élagage radical à l'adresse* ε*), nous avons* $v(t') = h(\pi)$. *Puisque l'heuristique est correcte, nous avons* $h(\pi) \leq_{D_\pi} \underset{i=1..n}{\uparrow} p \oplus h(\pi_i)$ *où* $\text{Succ}(\pi) = \pi_1..\pi_n$. *Pour toute position* π_i *nous avons, par dé"finition de* v, $p \oplus h(\pi_i) = v(\{(\varepsilon, \pi_i)\})$. *En définissant* $t'_i = \{(\varepsilon, \pi_i)\}$, *nous obtenons un préfixe* t'_i *pour chaque fils* t_i *de* t. *Soit* $t'' = t' \cup \bigcup_{\pi \xrightarrow{\ell} \pi_i} \ell.\{(\varepsilon, \pi_i)\}$. *Par définition de* t'', *nous avons* $\text{Fils}(t'') = t'_1..t'_n$. *Puisque l'heuristique est correcte, nous avons* $v(t') \leq_{D_\pi} v(t'')$. *De plus, par hypothèse d'induction (*$(t'_i, t_i) \prec_2 (t'', t)$*), nous avons* $v(t'_i) \leq_{D_{\pi_i}} v(t_i)$. *En appliquant l'hypothèse de monotonie de* $\uparrow$, *nous obtenons :*

$$v(t') \leq_{D_\pi} v(t'') = \underset{i=1..n}{\uparrow} v(t'_i) \leq_{D_\pi} \underset{i=1..n}{\uparrow} v(t_i) = v(t)$$

Donc, par la propriété transitive des préordres $(D_\pi, \leq_{D_\pi})$, *on obtient* $v(t') \leq_{D_\pi} v(t)$. *Le troisième cas est celui où* t' *a lui aussi des arbres fils :* $\text{Fils}(t') = t'_1..t'_n$. *Puisque* $t'_i \leq t_i$, *par hypothèse d'induction* $((t'_i, t_i) \prec_2 (t', t))$, *nous avons* $v(t'_i) \leq_{D_{\pi_i}} v(t_i)$. *Nous obtenons alors l'énoncé, une fois de plus, par la monotonie de* $\uparrow$. *La preuve pour le cas* $\lambda(\pi) = \text{Opponent}$ *s'appuie, en revanche, sur la monotonie de* $\downarrow$. □

5.3.6. Interprétation des préfixes infinis. Si l'interprétation des préfixes finis a lieu dans une structure d'interprétation, en revanche, celle des préfixes infinis nécessite, en général, une algèbre d'interprétation, autrement dit, une structure d'évaluation étendue par la complétion idéaux. De cette manière, nous ne serons pas obligés de prétendre que la structure $(D_\pi, \leq_{D_\pi})$ des valeurs d'une positions soit complète par dirigés (dcpo) : la sémantique d'un arbre infini sera, d'une façon très générale, la classe de co-finalité des évaluations de ses approximants finis.

Definition 5.3.13. *Étant donné une structure d'évaluation monotone* $\mathcal{D} \in \mathsf{Estruct}(\mathcal{S}, D, \uparrow, \downarrow, p, h)$, *germe de l'algèbre d'évaluation* $\mathcal{D}^\flat = (\{D, D^\flat\}, \{\leq, \uparrow, \downarrow\})$, *l'*interprétation des préfixes arbitraires *(*finis ou infinis*) est la fonction* $w : \widehat{\mathrm{IT}}(\mathcal{S}, \pi) \to D^\flat_\pi$ *définie de la façon suivante :*

$$w(t) \triangleq [X]_\flat \quad \textit{où } X = \{v(t/^k) \mid k \in \mathbb{N}\}$$

□

L'interprétation d'un préfixe arbitraire t est donc la classe de co-finalité du dirigé constitué par l'évaluation des préfixes finis *réguliers* de t. Ce choix n'est pas critique ; nous verrons que tout autre dirigé convergent à l'arbre t conduit à la même interprétation. En revanche, l'hypothèse de correction de l'heuristique est essentielle. Puisque l'évaluation des préfixes finis est un homomorphisme (cf. Prop. 5.3.12), les dirigés de préfixes d'arbres seront transformés en dirigés de valeurs, autrement dit, en éléments de l'extension bémol $D^\flat_\pi$ (où π est la position à la racine).

Proposition 5.3.14. *L'interprétation des préfixes arbitraires est correcte et il s'agit d'une fonction monotone :* $w : (\widehat{\mathrm{IT}}(\mathcal{S}, \pi), \leq) \overset{\mathcal{H}}{\longmapsto} (D^\flat_\pi, \leq^\flat_{D_\pi})$

Démonstration. *Par la Prop. 5.3.12, la fonction* v *est un homomorphisme. Elle transforme alors les dirigés de* $(\widehat{\mathrm{T}}(\mathcal{S}, \pi), \leq_\pi)$ *en dirigés de* $(D_\pi, \leq_{D_\pi} : 1^{D \times D})$. *L'ensemble* $X = \{v(t/^k) \mid k \in \mathbb{N}\}$ *est l'image de* v *sur le dirigé* $\{t/^k \mid k \in \mathbb{N}\}$. *Il s'agit donc d'un dirigé. Supposons* $t' \leq t$. *Soient* $X' = \{v(t'/^k) \mid k \in \mathbb{N}\}$ *et* $X = \{v(t/^k) \mid k \in \mathbb{N}\}$. *Nous voulons montrer que* $[X']_\flat \leq^\flat_\pi [X]_\flat$. *Il suffit de prouver que* $X' \leq^\flat_\pi X$. *Soit* $x' \in X'$. *Par définition de* X', *nous avons* $x' = v(t'/^j)$. *Soit* $x = v(t/^j) \in X$. *Puisque* $t' \leq t$, *nous avons* $t'/^k \leq t/^k$ *pour tout* $k \in \mathbb{N}$. *En particulier* $t'/^j \leq t/^j$ *donc, par monotonie de* v, $x' \leq_{D_\pi} x$, c.q.f.d. □

La proposition suivante montre que le choix des préfixes *réguliers* $t/^k$, pour l'approximation d'un arbre infini $t \in \widehat{IT}(\mathcal{S})$, n'est pas un choix fondamental : toute autre façon d'approcher l'arbre t par des préfixes finis de t constituant un dirigé, se révèle équivalente. Autrement dit, tout autre dirigé de points compacts approchant l'arbre t, compact ou non, permettra de donner à t la même sémantique.

PROPOSITION 5.3.15. *Pour tout dirigé* $T \in \Delta(\widehat{T}(\mathcal{S}), \leq)$ *tel que* $\sup_{\leq} T = t$, *nous avons l'équation :*

$$w(t) = [v(T)]_\flat$$

DÉMONSTRATION. *Soit* $R = \{t/^k \mid k \in \mathbb{N}\}$. *Étant des dirigés de compacts et partageant la même borne supérieure* t, *les ensembles* T *et* R *sont co-finaux (cf. lemme 2.4.3) . La fonction* v, *qui est monotone, transforme les ensembles co-finaux en ensembles co-finaux et les dirigés en dirigés (cf. 2.3.2). Autrement dit, la classe* $[v(T)]_\flat$ *coïncide avec la classe* $[v(R)]_\flat = w(t)$. □

REMARK 5.3.16. Si nous considérons l'extension aux classes co-finales des opérations des joueurs, définies de la façon suivantes :

$$\overset{n}{\underset{i=1}{\uparrow}} \chi_i \triangleq [\{\overset{n}{\underset{i=1}{\uparrow}} x_i \mid x_1 \in \chi_1, .., x_n \in \chi_n\}]_\flat$$

$$\overset{n}{\underset{i=1}{\downarrow}} \chi_i \triangleq [\{\overset{n}{\underset{i=1}{\downarrow}} x_i \mid x_1 \in \chi_1, .., x_n \in \chi_n\}]_\flat$$

nous remarquerons que l'interprétation des préfixes arbitraires w vérifie les même équations (rapportées à l'extension bémol) que l'interprétation des préfixes finis v vérifie par définition :

$$w(t) = \begin{cases} [\{\perp\}]_\flat & \text{si } t = \emptyset \\ [\{p(\pi)\}]_\flat & \text{si } t = \{(\varepsilon, \pi)\},\ \pi \in \mathbb{P}_T \\ [\{h(\pi)\}]_\flat & \text{si } t = \{(\varepsilon, \pi)\},\ \pi \in \mathbb{P}_{NT} \\ \overset{n}{\underset{i=1}{\uparrow}} w(t_i) & \text{si Fils}(t) = t_1..t_n,\ \lambda(t(\varepsilon)) = \text{Player} \\ \overset{n}{\underset{i=1}{\downarrow}} w(t_i) & \text{si Fils}(t) = t_1..t_n,\ \lambda(t(\varepsilon)) = \text{Opponent} \end{cases}$$

Les cas de base sont trivialement vrais par définition de w. En ce qui concerne l'équation $w(t) = \overset{n}{\underset{i=1}{\uparrow}} w(t_i)$ (l'autre équation étant duale), il suffit d'appliquer les différentes définitions :

$$\begin{aligned} w(t) &= [\{v(t/^k) \mid k \in \mathbb{N}\}]_\flat \\ &= [\{\overset{n}{\underset{i=1}{\uparrow}} v(t_i/^{k-1}) \mid k \in \mathbb{N}\}]_\flat \\ &= \overset{n}{\underset{i=1}{\uparrow}} [\{v(t_i/^{k-1}) \mid k \in \mathbb{N}\}]_\flat \\ &= \overset{n}{\underset{i=1}{\uparrow}} [\{v(t_i/^{k}) \mid k \in \mathbb{N}\}]_\flat \\ &= \overset{n}{\underset{i=1}{\uparrow}} w(t_i) \end{aligned}$$

Cela renforce l'idée que la fonction définie w soit une extension naturelle aux arbres infinis du procédé d'évaluation traditionnel, la rétro-propagation des valeurs aux feuilles, utilisé pour les arbres finis. □

Tenant compte de l'isomorphisme $(\widehat{\mathrm{T}}(\mathcal{S},\pi)^\flat, \leq^\flat_\pi) \overset{\sqcup}{\cong} (\widehat{\mathrm{IT}}(\mathcal{S},\pi), \leq)$ (cf. prop. 4.3.4), l'interprétation des préfixes arbitraires pourra être considérée, au besoin, comme une fonction de type $\widehat{\mathrm{T}}(\mathcal{S},\pi)^\flat \to \mathrm{D}^\flat_\pi$, que nous noterons $w^\flat$:

$$w^\flat(\chi) \triangleq [v(\mathsf{X})]_\flat \quad \text{si } \chi = [\mathsf{X}]_\flat,\ \mathsf{X} \in \Delta(\widehat{\mathrm{T}}(\mathcal{S},\pi), \leq)$$

La proposition 5.3.15 nous assure que la définition ne dépend pas du représentant X choisi pour la classe de co-finalité. En d'autres termes, les deux fonctions coïncident modulo l'isomorphisme tracé par la fonction borne supérieure :

$$w^\flat(\chi) = w(\sup \chi)$$

En effet, $w^\flat(\chi)$ est égale, par définition, à la classe $[v(\mathsf{T})]_\flat$ pour un représentant arbitraire $\mathsf{T} \in \Delta(\widehat{\mathrm{T}}(\mathcal{S},\pi), \leq)$ tel que $\chi = [\mathsf{T}]_\flat$. Soit $\mathsf{t} = \sup \mathsf{T}$. Par la prop. 5.3.15 $w(\mathsf{t}) = [v(\mathsf{T})]_\flat$. Donc $w^\flat(\chi) = [v(\mathsf{T})]_\flat = w(\mathsf{t}) = w(\sup \mathsf{T}) = w(\sup \chi)$.

5.3.7. Interprétation d'un jeu. L'interprétation d'un jeu est ainsi complètement définie par la donnée d'une structure d'évaluation monotone $\mathcal{D} \in \mathsf{Estruct}(\mathcal{S}, \mathrm{D}, \uparrow, \downarrow, \mathrm{p}, \mathrm{h})$. En effet, puisque l'heuristique est correcte, la fonction d'évaluation des préfixes infinis w peut être définie par le biais de la fonction d'évaluation v des préfixes finis. Ces deux applications complètent ainsi le cadre de l'interprétation des domaines de la sous-algèbre syntaxique *principale* (cf. section 4.5.2.1).

Definition 5.3.17. *Étant donné une arène de jeu* $\mathcal{S} = (\mathbb{P}, \lambda, \to, \mathbb{P}_{\mathrm{I}})$ *dont la sous-algèbre syntaxique principale est*

$$(\mathbb{P}_{\mathrm{T}}, \mathbb{P}_{\mathrm{NT}}, \mathrm{T}(\mathbb{P}), \widehat{\mathrm{T}}(\mathcal{S}), \widehat{\mathrm{IT}}(\mathcal{S}), \mathrm{IT}(\mathcal{S})\,;\ \leq : \forall\alpha{:}\{\widehat{\mathrm{T}}(\mathcal{S}), \widehat{\mathrm{IT}}(\mathcal{S})\}.\ 1^{\alpha\times\alpha})$$

et une structure d'évaluation monotone $\mathcal{D} \in \mathsf{Estruct}(\mathcal{S}, \mathrm{D}, \uparrow, \downarrow, \mathrm{p}, \mathrm{h})$*, germe de l'algèbre d'évaluation :*

$$\mathcal{D}^\flat = (\mathrm{D}, \mathrm{D}^\flat\,;\ \leq : \forall\alpha{:}\mathsf{sort}.\ 1^{\alpha\times\alpha}, \uparrow, \downarrow)$$

l'interprétation du jeu *est l'interprétation* (ψ, φ) *de l'algèbre syntaxique principale du jeu vers* $\mathcal{D}^\flat$*, définie de la façon suivante :*

(1) *l'interprétation des noms* ψ *est l'application suivante :*

(a) *pour les noms des domaines :*

$$\psi(\sigma) = \begin{cases} \mathrm{D} & \text{si} \quad \sigma \in \{\mathbb{P}_{\mathrm{T}}, \mathbb{P}_{\mathrm{NT}}, \mathrm{T}(\mathbb{P}), \widehat{\mathrm{T}}(\mathcal{S})\} \\ \mathrm{D}^\flat & \text{si} \quad \sigma \in \{\widehat{\mathrm{IT}}(\mathcal{S}), \mathrm{IT}(\mathcal{S})\} \end{cases}$$

(b) *pour les noms des opérations :*

$$\psi(\leq) \triangleq (\leq)$$

(2) *l'interprétation des valeurs est la construction polymorphe :*

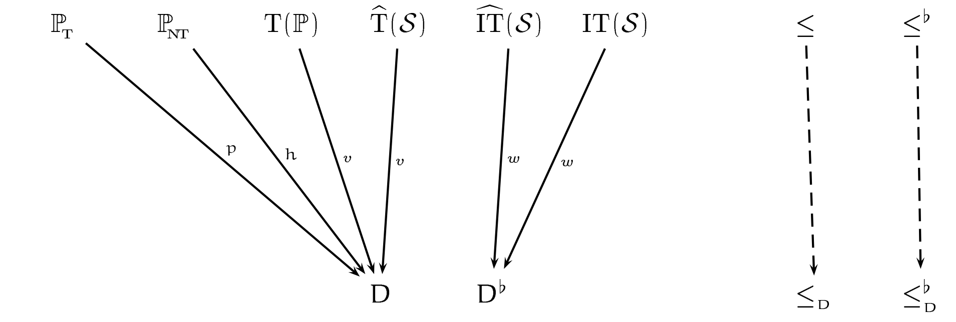
$$\varphi : \forall\alpha{:}\{\mathbb{P}_{\mathrm{T}}, \mathbb{P}_{\mathrm{NT}}, \mathrm{T}(\mathbb{P}), \widehat{\mathrm{T}}(\mathcal{S}), \widehat{\mathrm{IT}}(\mathcal{S}), \mathrm{IT}(\mathcal{S})\}.\ \alpha \to \psi(\alpha)$$

définie par : $\varphi_{\mathbb{P}_{\mathrm{T}}} \triangleq \mathrm{p}$, $\varphi_{\mathbb{P}_{\mathrm{NT}}} \triangleq \mathrm{h}$, $\varphi_{\mathrm{T}(\mathbb{P})} \triangleq v$, $\varphi_{\widehat{\mathrm{T}}(\mathcal{S})} \triangleq v_{|\widehat{\mathrm{T}}(\mathcal{S})}$, $\varphi_{\widehat{\mathrm{IT}}(\mathcal{S})} \triangleq w$, $\varphi_{\mathrm{IT}(\mathcal{S})} \triangleq w_{|\mathrm{IT}(\mathcal{S})}$*, où :*

(a) *la fonction* $v : \mathrm{T}(\mathbb{P}) \to \mathrm{D}$ *est l'interprétation des arbres étiquetés dans* $\mathbb{P}$ *(avec* p *et* h*)*

FIG. 5.3.1. Interprétation d'un jeu

SOUS-ALGÈBRE SYNTAXIQUE PRINCIPALE



ALGÈBRE D'ÉVALUATION SÉMANTIQUE

(b) *la fonction* $w : \widehat{\mathrm{IT}}(\mathcal{S}) \to \mathrm{D}^\flat$ *est l'interprétation des préfixes arbitraires du jeu (avec* p, h *et* v*)*

□

L'interprétation du jeu pourra être notée par les quatre composantes principales de l'interprétation des valeurs $\varphi = (\mathrm{p}, \mathrm{h}, v, w)$, faisant omission de l'association implicite des noms. Nous évoquerons l'interprétation d'un jeu $\mathcal{S}$ dans une *structure* d'évaluation $\mathcal{D}$, sous-entendant l'interprétation de $\mathcal{S}$ dans l'*algèbre* d'évaluation ayant pour germe la structure $\mathcal{D}$. Étant une somme de fonctions, l'interprétation des valeurs φ traduit forcément les éléments de façon univoque. Pour prouver que la définition précédente est correcte, il suffit donc de montrer que l'interprétation des noms ψ est correcte vis-à-vis des types.

PROPOSITION 5.3.18. *La définition d'interprétation d'un jeu est correcte et il s'agit d'un homomorphisme.*

DÉMONSTRATION. *La traduction du type* $1^{\widehat{\mathrm{T}}(\mathcal{S})\times\widehat{\mathrm{T}}(\mathcal{S})}$ *de* $\leq_{\widehat{\mathrm{T}}(\mathcal{S})}$ *est précisément le type* $1^{\mathrm{D}\times\mathrm{D}}$ *de* $\psi(\leq)_{\mathrm{D}}$. *De même, la traduction du type* $1^{\widehat{\mathrm{IT}}(\mathcal{S})\times\widehat{\mathrm{IT}}(\mathcal{S})}$ *de* $\leq^\flat = \leq_{\widehat{\mathrm{IT}}(\mathcal{S})}$ *est précisément le type* $1^{\mathrm{D}^\flat\times\mathrm{D}^\flat}$ *de* $\psi(\leq^\flat)_{\mathrm{D}^\flat} = \leq^\flat_{\mathrm{D}}$. *Donc la construction polymorphe syntaxique* $\leq$ *est compatible avec l'opération sémantique correspondante (cf. section 1.3.2). Autrement dit, l'interprétation des noms est correcte vis-à-vis des types. Par ailleurs, l'interprétation des valeurs est une fonction (traduction univoque) sur l'union des domaines syntaxiques et à valeurs dans l'union des domaines sémantiques. Enfin, en ce qui concerne la conservation des opérations, d'une part* $\leq$: $1^{\widehat{\mathrm{T}}(\mathcal{S})\times\widehat{\mathrm{T}}(\mathcal{S})}$ *est conservée dans* $\psi(\leq)_{\mathrm{D}} = \leq_{\mathrm{D}}$ *par la fonction monotone* v *(cf. Prop. 5.3.12) et, d'autre part,* $\leq^\flat$: $1^{\widehat{\mathrm{IT}}(\mathcal{S})\times\widehat{\mathrm{IT}}(\mathcal{S})}$ *est conservée dans* $\psi(\leq^\flat) = \leq^\flat_{\mathrm{D}}$ *par la fonction monotone* w *(cf. prop. 5.3.14).* □

La figure 5.3.1 représente d'une façon succincte l'homomorphisme de la sous-algèbre syntaxique principale d'un jeu vers une algèbre d'évaluation. D'une part, les lignes continues représentent les composantes de l'interprétation des valeurs φ et, implicitement, l'association des noms ψ pour les domaines. D'autre part, les lignes pointillées représentent aussi bien l'association des noms que la relation de *conservation* des opérations algébriques.

NOTATION 5.3.19. Puisque l'interprétation d'un jeu $\mathcal{S}$ est complètement définie par une structure d'évaluation monotone $\mathcal{D} \in \mathsf{Estruct}(\mathcal{S}, \mathsf{D}, \uparrow, \downarrow, \mathsf{p}, \mathsf{h})$, par la suite nous la noterons par $\varphi_{\mathcal{D}}$.

5.4. Sémantique des positions

Puisque l'interprétation d'un jeu est complètement définie par sa syntaxe $\mathcal{S}$ et par la structure $\mathcal{D} \in \mathsf{Estruct}(\mathcal{S}, \mathsf{D}, \uparrow, \downarrow, \mathsf{p}, \mathsf{h})$ qui constitue le germe de l'algèbre d'évaluation $\mathcal{D}^\flat$, nous pouvons définir une fonction sémantique à valeurs dans $\mathsf{D}^\flat$ pour les positions du jeu. Étant donné une position du jeu π, nous pouvons considérer, en premier lieu, son arbre de jeu t_π et, en second lieu, l'évaluation de cet arbre par la fonction d'interprétation des préfixes arbitraires. La composition fonctionnelle $w \circ \mathsf{t}_{(.)}$, que nous noterons $(\!|\,.\,|\!)^\flat$, sera donc la sémantique de référence pour les positions du jeu. Cependant, il existe une projection naturelle, la borne supérieure, de l'extension bémol vers son germe. Autrement dit, il existe une façon canonique, bien que partielle (puisque la borne n'existe pas toujours), de traduire le sens d'une position dans le domaine D de départ. Cela nous amène à définir deux fonctions sémantiques pour les positions : une *totale* $(\!|\,.\,|\!)^\flat$, à valeurs dans $\mathsf{D}^\flat$, et une *partielle* $(\!|\,.\,|\!)$, à valeurs dans D.

DEFINITION 5.4.1. (SÉMANTIQUE MONOSÉMIQUE) *Étant donné une arène de jeu* $\mathcal{S} = (\mathbb{P}, \lambda, \rightarrow, \mathbb{P}_{\mathrm{I}})$ *et une structure d'évaluation monotone* $\mathcal{D} \in \mathsf{Estruct}(\mathcal{S}, \mathsf{D}, \uparrow, \downarrow, \mathsf{p}, \mathsf{h})$*, la* sémantique monosémique bémol *des positions est la fonction totale* $(\!|\,.\,|\!)^\flat_{\mathcal{D}} : \mathbb{P} \rightarrow \mathsf{D}^\flat$ *définie par :*

$$(\!|\pi|\!)^\flat_{\mathcal{D}} \triangleq w(\mathsf{t}_\pi)$$

où $\varphi_{\mathcal{D}} = (\mathsf{p}, \mathsf{h}, v, w)$*. En revanche, la* sémantique monosémique *des positions est la fonction partielle* $(\!|\,.\,|\!)_{\mathcal{D}} : \mathbb{P} \rightharpoonup \mathsf{D}'$ *définie par :*

$$(\!|\pi|\!)_{\mathcal{D}} \leftleftharpoons \sup\, (\!|\pi|\!)^\flat_{\mathcal{D}}$$

□

où D' est D si $(\mathsf{D}_\pi, \leq_\pi)$ est une famille d'ordres partiels, ou bien D' est la famille $(\mathsf{D}_{\pi_\approx}, \leq_\pi)$ des ordres partiels obtenus par le passage aux classes d'équivalence des respectives structures de préordre. Par la suite nous écrirons $(\!|\,.\,|\!)_{\mathcal{D}} : \mathbb{P} \rightharpoonup \mathsf{D}$, sous-entendant le passage aux classes d'équivalences lorsqu'il y en aura besoin. Nous écrirons $(\!|\pi|\!)^\flat$ et $(\!|\pi|\!)$, omettant les indices, lorsque cela ne causera aucune ambiguïté. La portée de l'appellation *monosémique*[2] sera évidente par la suite, lorsque

[2] En linguistique, un mot est *monosémique* quand il n'a qu'un seul sens, par opposition aux mots qui ont plusieurs sens (*polysémiques*). Dans la catégorie des polysémiques, un mot est *bisémique* lorsqu'il prend deux sens différents selon les contextes.

nous discuterons la notion de co-évaluation d'un jeu. Si les sémantiques monosémiques attribuent un seul signifié à toute position du jeu (dans $D^\flat$ ou bien dans D), en revanche, les sémantiques *bisémiques* en apporteront deux : le premier sens pourra être interprété comme la valeur *par défaut* de la position du jeu, et le second comme la valeur *par excès*.

REMARK 5.4.2. Par la remarque 5.3.16 nous pouvons réécrire les équations habituelle pour la sémantique monosémique bémol :

$$(\!|\pi|\!)^\flat = \begin{cases} [\{p(\pi)\}]_\flat & \text{si}\, \pi \in \mathbb{P}_T \\ [\{h(\pi)\}]_\flat & \text{si}\, t = \{(\varepsilon,\pi)\},\ \pi \in \mathbb{P}_{NT} \\ \overset{n}{\underset{i=1}{\uparrow}} (\!|\pi_i|\!)^\flat & \text{si}\, \mathrm{Succ}(\pi) = \pi_1..\pi_n,\ \ \lambda(\pi) = \text{Player} \\ \overset{n}{\underset{i=1}{\downarrow}} (\!|\pi_i|\!)^\flat & \text{si}\, \mathrm{Succ}(\pi) = \pi_1..\pi_n,\ \ \lambda(\pi) = \text{Opponent} \end{cases}$$

□

La sémantique monosémique devient une fonction totale dans le cadre d'une structure d'évaluation qui est un dcpo algébrique.

PROPOSITION 5.4.3. *Soit* $\mathcal{D} \in \mathsf{Estruct}(\mathcal{S}, D, \uparrow, \downarrow, p, h)$ *une structure d'évaluation monotone pour un jeu* $\mathcal{S} = (\mathbb{P}, \lambda, \rightarrow, \mathbb{P}_I)$. *Si pour toute position* $\pi \in \mathbb{P}$, *le préordre* $(D_\pi, \leq_{D_\pi})$ *est un dcpo algébrique, alors la sémantique monosémique des positions est une fonction totale :*

$$(\!|\,.\,|\!)_\mathcal{D} : \mathbb{P} \rightarrow D$$

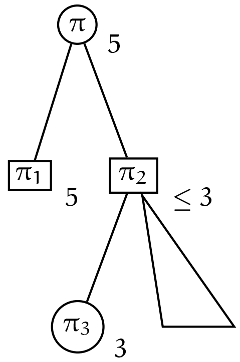
DÉMONSTRATION. *L'énoncé est conséquence directe de l'isomorphisme, via l'application borne supérieure, entre tout dcpo algébrique et son extension bémol (cf. prop. 2.5.8)* □

Le rapport entre, d'une part, la sémantique monosémique des positions et, d'autre part, les fonctions auxiliaires p, pour l'interprétation des positions terminales, et v, pour l'interprétation des positions dont l'arbre de jeu est fini, est celui auquel nous pouvons nous attendre : la première généralise les secondes permettant l'évaluation de tout type de position.

PROPOSITION 5.4.4. *Soit* $\varphi_\mathcal{D} = (p, h, v, w)$ *l'interprétation d'un jeu* $\mathcal{S} = (\mathbb{P}, \lambda, \rightarrow, \mathbb{P}_I)$ *engendrée par une structure d'évaluation monotone* $\mathcal{D} \in \mathsf{Estruct}(\mathcal{S}, D, \uparrow, \downarrow, p, h)$. *Alors la sémantique monosémique des positions coïncide avec la fonction de payoff* p *sur les positions terminales, et coïncide avec la composition* $v \circ t_{(.)}$ *sur les positions dont l'arbre de jeu est fini :*

$$\begin{array}{lrcl} \forall \pi \in \mathbb{P}_T. & p(\pi) & = & (\!|\pi|\!)_\mathcal{D} \\ \forall t_\pi \in T(\mathcal{S}). & v(t_\pi) & = & (\!|\pi|\!)_\mathcal{D} \end{array}$$

FIG. 5.4.1. Sémantique monosémique d'un jeu infini



DÉMONSTRATION. *Par définition* $(\!|\pi|\!)_{\mathcal{D}} \leftleftharpoons \sup(w(t_\pi))$. *Si* $\pi \in \mathbb{P}_T$, *alors* $t_\pi = \{(\varepsilon,\pi)\} = t_\pi/^0$. *Donc* $\sup(w(t_\pi)) = \sup([\{v(t_\pi/^k) \mid k \in \mathbb{N}\}]_\flat) = \sup([\{v(t_\pi/^0)\}]_\flat) = \sup([\{v(t_\pi)\}]_\flat) = \sup([\{p(\pi)\}]_\flat) = p(\pi)$. *De la même façon, si l'arbre de jeu est fini il existe un préfixe régulier tel que* $t_\pi/^j = t_\pi$. *Donc* $\sup(w(t_\pi)) = \sup([\{v(t_\pi/^k) \mid k \in \mathbb{N}\}]_\flat) = \sup([\{v(t_\pi/^j)\}]_\flat) = \sup([\{v(t_\pi)\}]_\flat) = v(t_\pi) = v \circ t_{(.)}(\pi)$. □

Par conséquent, dans le cas particulier où les ordres partiels $(D_\pi, \leq_{D_\pi})$ sont des dcpo algébriques, la sémantique monosémique des positions d'un jeu est une fonction totale qui étend les fonctions p et v à toutes les positions du jeu. Toujours en ce qui concerne le rapport entre la sémantique monosémique et l'interprétation du jeu $\varphi_{\mathcal{D}} = (p, h, v, w)$, nous remarquerons que la fonction v d'interprétation des préfixes finis peut être utilisée pour approximer la fonction sémantique $(\!|\,.\,|\!)_{\mathcal{D}}$, lorsque cette dernière existe.

LEMMA 5.4.5. *Soit* $\varphi_{\mathcal{D}} = (p, h, v, w)$ *l'interprétation d'un jeu* $\mathcal{S} = (\mathbb{P}, \lambda, \rightarrow, \mathbb{P}_I)$ *engendrée par une structure d'évaluation monotone* $\mathcal{D} \in \mathsf{Estruct}(\mathcal{S}, D, \uparrow, \downarrow, p, h)$. *Alors :*

$$\forall \pi \in \mathrm{dom}((\!|\,.\,|\!)_{\mathcal{D}}).\ \forall t \in \widehat{T}(\mathcal{S}, \pi). \qquad v(t) \leq_{D_\pi} (\!|\pi|\!)_{\mathcal{D}}$$

DÉMONSTRATION. *Par définition de sémantique monosémique, nous avons :*

$$(\!|\pi|\!)_{\mathcal{D}} \ = \ \sup\ w(t_\pi) \ = \ \sup([\{v(t_\pi/^k) \mid k \in \mathbb{N}\}]_\flat)$$

Si on suppose $t \leq t_\pi$, *alors il existe un préfixe régulier de* t_π *qui dépasse* t. *Soit* $t_\pi/^j$ *un tel préfixe. En utilisant la monotonie de la fonction* v *et de la borne supérieure (vis-à-vis de l'inclusion ensembliste), nous avons :*

$$\begin{array}{lll} v(t) & \leq_{D_\pi} & v(t_\pi/^j) \leq_{D_\pi} \sup\{v(t_\pi/^j)\} \leq_{D_\pi} \sup\{v(t_\pi/^k) \mid k \in \mathbb{N}\} \\ & = & \sup([\{v(t_\pi/^k) \mid k \in \mathbb{N}\}]_\flat) \ = \ (\!|\pi|\!)_{\mathcal{D}} \end{array}$$

Donc, par la propriété transitive des préordres $(D_\pi, \leq_{D_\pi})$, *nous obtenons l'énoncé* $v(t) \leq_{D_\pi} (\!|\pi|\!)_{\mathcal{D}}$. □

La relation entre la sémantique monosémique et la fonction d'évaluation des préfixes finis v ne se limite pas aux conditions des lemmes précédents, qui ne donnent aucune indication sur l'*existence* de la valeur sémantique. Il existe aussi une sorte de critère d'existence (et de terminaison du calcul), pour la sémantique monosémique, dont l'utilité sera manifeste dans le cadre des co-évaluations.

LEMMA 5.4.6. *Soit* $\varphi_{\mathcal{D}} = (p, h, v, w)$ *l'interprétation d'un jeu* $\mathcal{S} = (\mathbb{P}, \lambda, \rightarrow, \mathbb{P}_I)$ *engendrée par une structure d'évaluation monotone* $\mathcal{D} \in \mathsf{Estruct}(\mathcal{S}, D, \uparrow, \downarrow, p, h)$. *Alors :*

$$\frac{\exists \bar{t} \in \widehat{T}(\mathcal{S},\pi).\ \forall t \in \widehat{T}(\mathcal{S},\pi).\ \ t \geq \bar{t} \ \Rightarrow\ v(t) = x}{(\!|\pi|\!)_{\mathcal{D}} = x}$$

DÉMONSTRATION. *Dans les hypothèse du critère, nous avons :*

$$\begin{array}{rcll}
(\!|\pi|\!)_{\mathcal{D}} & \stackrel{\leftarrow}{=} & \sup(\ [\{v(t_\pi/^k) \mid k \in \mathbb{N}\}]_\flat\ ) & (\textit{déf. de } (\!|\pi|\!)_{\mathcal{D}}\ ) \\
 & \stackrel{\leftarrow}{=} & \sup(\ [\{v(\bar{t})\}]_\flat) & (\textit{déf. de classe et monotonie de } v) \\
 & \stackrel{\leftarrow}{=} & \sup\{v(\bar{t})\} & (\textit{def. de } \sup\ \textit{d'une classe}) \\
 & = & v(\bar{t}) & \\
 & = & x & (\textit{hypothèse})
\end{array}$$

□

Ce résultat nous conduit vers la notion de position finiment évaluable. La figure 5.4.1 illustre le cas d'un jeu min-max sur les entiers naturels, où la sémantique d'une position π existe en dépit de la dimension infinie de son arbre. Dans la position initiale, le joueur a un coup (à gauche de la figure) qui conduit le jeu dans une position terminale π_1 dont le payoff est l'entier 5. En revanche, pour l'autre coup possible (à droite), l'opposant aura à son tour, dans la position π_2, un coup conduisant à une autre position terminale π_3 dont le payoff est 3. Il est ainsi certain que, l'opposant faisant le min, le résultat (forcément inférieur ou égal à 3) ne pourra jamais conditionner la valeur à la racine de l'arbre (où le joueur fait le max avec 5). En d'autres termes, on peut prouver que tout préfixe régulier $t_\pi/^k$ où $k \geq 2$, a comme valeur l'entier 5, qui est donc la borne supérieure de l'évaluation des préfixes finis.

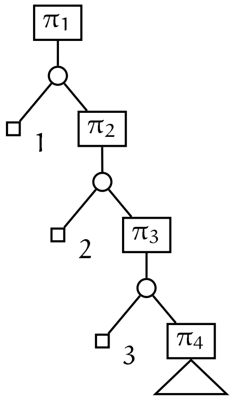
DEFINITION 5.4.7. (POSITIONS FINIMENT ÉVALUABLES) *Étant donné une structure d'évaluation monotone* $\mathcal{D} \in \mathsf{Estruct}(\mathcal{S}, D, \uparrow, \downarrow, p, h)$ *pour un jeu* $\mathcal{S} = (\mathbb{P}, \lambda, \rightarrow, \mathbb{P}_I)$*, une position* $\pi \in \mathbb{P}$ *est* finiment évaluable *si sa sémantique monosémique existe et coïncide avec l'évaluation d'un préfixe fini de son arbre de jeu :*

$$\exists k \in \mathbb{N}. \qquad (\!|\pi|\!)_{\mathcal{D}} = v(t_\pi/^k)$$

□

Les positions non finiment évaluables sont telles que toutes les approximations $\{v(t_\pi/^k)\}_{k \in \mathbb{N}}$ approchent progressivement la valeur sémantique $(\!|\pi|\!)$ sans jamais l'atteindre. Si le cas de la figure 5.4.1 est celui d'une position finiment évaluable, au contraire, celui de la figure 5.4.2 est un exemple de position non finiment évaluable dans un jeu min-max sur les entiers naturels, interprété avec l'heuristique canonique

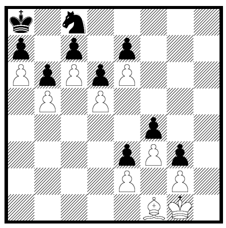
FIG. 5.4.2. Position non finiment évaluable



$h(\pi) = \perp = 0$. Pour chaque position π_i non terminale de l'opposant, le joueur a un premier coup conduisant à une position terminale dont le payoff est i, et un second conduisant à une position π_{i+1} de l'opposant similaire à la précédente (à la différence du payoff suivant qui est égal à $i+1$). L'évaluation du jeu à tous les niveaux pairs $i = 2k$ est donc égale à i. La sémantique bémol d'une telle position est la classe d'équivalence des dirigés tendant à l'infini. Si l'on considère que la valeur $+\infty$ appartient à notre domaine d'évaluation, la sémantique des positions π_i n'est jamais atteinte par ses approximations finies. Il s'agit donc de positions non finiment évaluables. Nous remarquerons que, dans le cas général, une position finiment évaluable peut avoir des branches qui ne le sont pas. Une telle situation peut être provoquée en greffant l'arbre d'une position non finiment évaluable sur la partie qui n'a aucune influence dans l'arbre de la figure 5.4.1.

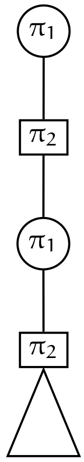
5.5. Sémantique bisémique

Dans l'exemple de la figure 5.4.2, si nous avions choisi l'heuristique $h(\pi) = +\infty$ au lieu de l'heuristique canonique, la position π aurait dû être considérée finiment évaluable. Considérons la position des Échecs suivante :



Le jeu est traditionnellement interprété dans le domaine des valeurs $\{-1, 0, +1\}$ avec les opérations min et max. Dans la position illustrée, les blancs et les noirs sont bloqués, à l'exception des deux rois, qui peuvent se déplacer horizontalement dans l'espace de deux seules cases. L'arbre de cette position a donc la forme de la figure 5.5.1. N'importe quelle fonction constante $h_1(\pi) = -1$ ou $h_2(\pi) = 0$ ou $h_3(\pi) = +1$ est correcte et peut donc être utilisée pour l'interprétation d'un jeu

FIG. 5.5.1. Arbre de jeu linéaire et rationnel



avec un seul arbre tel que celui de la figure 5.5.1. À chaque heuristique correspondra une valeur différente de la sémantique monosémique de la position : pour le choix h_1 la sémantique sera -1 (perdant), pour h_2 la sémantique sera 0 (match null), pour h_3 la sémantique sera $+1$ (gagnant). Dans les trois cas, nous aboutissons à la valeur sémantique dès la première approximation, au niveau de la racine de l'arbre. En d'autre termes, dans les trois cas la position est finiment évaluable.

Aussi bien la sémantique des positions que la propriété, d'une position, d'être finiment évaluable ou pas dépendent ainsi de l'heuristique considérée pour l'interprétation du jeu. Cependant, un exemple précédent, celui de la figure 5.4.1, nous montre le contraire : il nous suggère que certaines évaluations du jeu ne dépendent en aucune façon de l'heuristique utilisée pour l'approximation des positions non terminales. En effet, la valeur 5 de la position π est exclusivement impliquée par les propriétés algébriques, notamment la monotonie, des opérations min et max associées aux joueurs. Cet exemple nous amène vers une considération fondamentale : en évaluant les positions non terminales avec deux valeurs opposées, la plus petite $\perp$ et la plus grande $\top$ de notre domaine d'évaluation, nous pourrions obtenir deux approximations, une *pessimiste* et l'autre *optimiste*, de la sémantique d'une position. Éventuellement, en observant l'égalité des deux approximations, nous pourrions affirmer, sans incertitudes, avoir calculé l'*unique* possible sémantique de la position donnée. En généralisant l'approche, équipés de deux façons arbitraires d'évaluer les noeuds non terminaux, une pessimiste, l'autre optimiste, non nécessairement égales à $\perp$ et $\top$, nous pourrions obtenir deux signifiés pour toute position du jeu. Et en observant, pour une certaine position, l'égalité des deux signifiés, le signifié pessimiste et le signifié optimiste, nous aurions saisi une unique sémantique de la position. Il se pourra donc, à partir d'une sémantique bisémique, de retrouver un seul signifié, donc une sémantique monosémique, pour certaines positions du jeu.

DEFINITION 5.5.1. (DUALE D'UNE STRUCTURE D'ÉVALUATION) *Étant donné une structures d'évaluation* $\mathcal{D} = (\mathsf{D}, \leq_{\mathsf{D}}, \uparrow, \downarrow)$*, la structure* duale *est l'algèbre* $\mathcal{D}^{\perp} = (\mathsf{D}, \geq_{\mathsf{D}}, \uparrow, \downarrow)$ *où* $\geq_{\mathsf{D}}$ *est la relation duale de* $\leq_{\mathsf{D}}$ *:*

$$x \leq_{\mathrm{D}} y \quad \Leftrightarrow y \geq_{\mathrm{D}} x$$

□

Il peut s'avérer qu'une structure d'évaluation $\mathcal{D} \in \mathsf{Estruct}(\mathcal{S}, \mathrm{D}, \uparrow, \downarrow, p, h)$ puisse fournir une double évaluation des préfixes du jeu, et cela lorsque sa duale $\mathcal{D}^{\perp}$ est, elle même, une structure d'évaluation $\mathcal{D}^{\perp} \in \mathsf{Estruct}(\mathcal{S}, \mathrm{D}, \uparrow, \downarrow, p, h')$, bâtie sur la même fonction de payoff et sur une seconde fonction heuristique. Nous allons voir sous quelles conditions cela est vrai.

Definition 5.5.2. (Valeurs pessimistes et optimistes d'une position) *Étant donné une arène de jeu* $\mathcal{S} = (\mathbb{P}, \lambda, \rightarrow, \mathbb{P}_{\mathrm{I}})$*, une structure d'inteprétation élémentaire* $(\mathrm{D}, \uparrow, \downarrow)$*, une fonction de payoff* $p : \mathbb{P}_{\mathrm{T}} \rightarrow \mathrm{D}$ *et deux heuristiques* $h_1, h_2 : \mathbb{P}_{\mathrm{NT}} \rightarrow \mathrm{D}$, l'ensemble des valeurs pessimiste et optimistes *d'une position* $\pi \in \mathbb{P}$*, noté* $\mathrm{D}_{\pi}^{p,h_1 h_2}$*, est l'union des valeurs des préfixes finis obtenues avec* h_1 *et avec* h_2 *:*

$$\mathrm{D}_{\pi}^{p,h_1,h_2} \triangleq \mathrm{D}_{\pi}^{p,h_1} \cup \mathrm{D}_{\pi}^{p,h_2}$$

De la même façon, l'ensemble des valeurs pessimiste et optimistes des élagages finis (du joueur) *dans* π*, noté* $\widetilde{\mathrm{D}}_{\pi}^{p,h_1,h_2}$*, est l'union des valeurs des élagages finis obtenues avec* h_1 *et avec* h_2 *:*

$$\widetilde{\mathrm{D}}_{\pi}^{p,h_1,h_2} \triangleq \widetilde{\mathrm{D}}_{\pi}^{p,h_1} \cup \widetilde{\mathrm{D}}_{\pi}^{p,h_2}$$

□

Notation 5.5.3. Pour alléger la notation, lorsque il n'y aura pas d'ambiguïté possible sur les fonctions de payoff et les deux heuristiques utilisées, nous écrirons $\mathrm{D}_{\pi}^{\pm}$ pour $\mathrm{D}_{\pi}^{p,h_1,h_2}$, et la notation $\widetilde{\mathrm{D}}_{\pi}^{\pm}$ pour $\widetilde{\mathrm{D}}_{\pi}^{p,h_1,h_2}$.

Les valeurs pessimiste et optimiste de l'ensemble $\mathrm{D}_{\pi}^{\pm}$ sont utilisées pour caractériser les structures permettant une double évaluation des arbres de jeu, finis ou infinis. En revanche, nous verrons par la suite que les valeurs pessimiste et optimiste des élagages finis, c'est-à-dire les valeurs de l'ensemble $\widetilde{\mathrm{D}}_{\pi}^{\pm}$, seront utilisées pour caractériser les structures permettant une double évaluation des stratégies finies ou infinies.

Definition 5.5.4. (Structures de co-évaluation monotones pour un jeu) *Étant donné :*

- *une arène de jeu* $\mathcal{S} = (\mathbb{P}, \lambda, \rightarrow, \mathbb{P}_{\mathrm{I}})$*,*
- *une structure d'inteprétation élémentaire* $(\mathrm{D}, \uparrow, \downarrow)$*,*
- *une fonction de payoff* $p : \mathbb{P}_{\mathrm{T}} \rightarrow \mathrm{D}$ *et deux heuristiques* $h_1, h_2 : \mathbb{P}_{\mathrm{NT}} \rightarrow \mathrm{D}$*,*

une structure de co-évaluation (monotone) pour le jeu $\mathcal{S}$*, est une algèbre* $\mathcal{D}$*, contenant la structure élémentaire, de signature :*

(D,				*domaine d'évaluation*
$\leq_{\mathrm{D}}$	:	$1^{\mathrm{D}\times\mathrm{D}}$	,	*relation sur les valeurs*
$\uparrow$	:	$\mathrm{D}^* \rightarrow \mathrm{D}$	,	*fonction du joueur*
$\downarrow$	:	$\mathrm{D}^* \rightarrow \mathrm{D}$	,	*fonction de l'opposant*
)				

et telle que :

(1) *pour toute position* $\pi \in \mathbb{P}$ *du jeu* $(D^{\pm}_{\pi}, \leq_{D^{\pm}_{\pi}})$ *est un préordre*

(2) $\uparrow$ *et* $\downarrow$ *sont monotones dans toutes leurs composantes par rapport au valeurs pessimistes ou optimistes du jeu, i.e. pour toute position non terminale* $\pi \in \mathbb{P}_{NT}$*, où* $\mathrm{Succ}(\pi) = \pi_1..\pi_n$*, les fonctions vérifient la propriété :*

$$\forall u \in D^{\pm}_{\pi_1}..D^{\pm}_{\pi_{k-1}}, v \in D^{\pm}_{\pi_{k+1}}..D^{\pm}_{\pi_n}.$$

$$x \leq_{D^{\pm}_{\pi_k}} y \Rightarrow \begin{cases} \uparrow (u.x.v) \leq_{D^{\pm}_{\pi}} \uparrow (u.y.v) \\ \\ \downarrow (u.x.v) \leq_{D^{\pm}_{\pi}} \downarrow (u.y.v) \end{cases}$$

(3) *les heuristiques sont* correctes, h_1 *dans la structure d'évaluation* $\mathcal{D}$*, et* h_2 *dans la structure d'évaluation* $\mathcal{D}^{\perp}$*, c'est-à-dire, pour toute position* $\pi \in \mathbb{P}_{NT}$*, telle que* $\mathrm{Succ}(\pi) = \pi_1..\pi_n$ $(n \geq 1)$*, elles vérifient les conditions suivantes :*

$$\lambda(\pi) = \text{Player} \quad \Rightarrow \begin{cases} h_1(\pi) \leq_{D^{\pm}_{\pi}} \left(\underset{i=1..n}{\uparrow} p \oplus h_1(\pi_i) \right) \\ \\ h_2(\pi) \geq_{D^{\pm}_{\pi}} \left(\underset{i=1..n}{\uparrow} p \oplus h_2(\pi_i) \right) \end{cases}$$

$$\lambda(\pi) = \text{Opponent} \quad \Rightarrow \begin{cases} h_1(\pi) \leq_{D^{\pm}_{\pi}} \left(\underset{i=1..n}{\downarrow} p \oplus h_1(\pi_i) \right) \\ \\ h_2(\pi) \geq_{D^{\pm}_{\pi}} \left(\underset{i=1..n}{\downarrow} p \oplus h_2(\pi_i) \right) \end{cases}$$

(4) *les heuristiques sont dans le préordre extensionnel :*

$$\forall \pi \in \mathbb{P}. \quad h_1(\pi) \leq_{D^{\pm}_{\pi}} h_2(\pi)$$

La fonction h_1 *est la* heuristique pessimiste, *la fonction* h_2 *est la* heuristique optimiste. *Nous noterons* $\mathsf{Cstruct}(\mathcal{S}, D, \uparrow, \downarrow, p, h_1, h_2)$ *l'ensemble des structures de co-évaluation bâties sur le jeu* $\mathcal{S}$*, la structure d'évaluation élémentaire* $(D, \uparrow, \downarrow)$*, le payoff* p *et les heuristiques* h_1 *et* h_2. □

On vérifie simplement que toute structure de co-évaluation monotone $\mathcal{D} \in \mathsf{Cstruct}(\mathcal{S}, D, \uparrow, \downarrow, p, h_1, h_2)$ est aussi une structure d'évaluation monotone $\mathcal{D} \in \mathsf{Estruct}(\mathcal{S}, D, \uparrow, \downarrow, p, h_1)$ sur l'heuristique h_1, et que sa structure duale est, elle aussi, une structure d'évaluation monotone $\mathcal{D}^{\perp} \in \mathsf{Estruct}(\mathcal{S}, D, \uparrow, \downarrow, p, h_2)$, mais sur l'heuristique h_2. Autrement dit, le caractère pessimiste ou optimiste d'une heuristique est une notion qui dépend de la direction que l'on donne à la relation sur les valeurs. La réciproque n'est pas vraie, les conditions de monotonie et de correction portant sur l'ensemble des valeurs pessimistes ou optimiste $D^{\pm}_{\pi}$, et non pas sur les ensembles, plus petits, D^{p,h_1}_{π} et D^{p,h_2}_{π}.

Par monotonie des opérations des joueurs, la rétro-propagation des valeurs, propage la relation $h_1(\pi) \leq_{D^{\pm}_{\pi}} h_2(\pi)$ entre les feuilles jusqu'à la racine de tout préfixe du jeu :

$$\forall t \in \widehat{T}(S, \pi).\ v_{p,h_1}(t) \leq_{D^{\pm}_{\pi}} v_{p,h_2}(t)$$

Cela permet de définir l'ensemble des intervalles d'une position.

DEFINITION 5.5.5. (INTERVALLES D'UNE POSITION) *Étant donné une structure de co-évaluation* $\mathcal{D} \in \mathsf{Cstruct}(\mathcal{S}, D, \uparrow, \downarrow, p, h, h')$ *pour un jeu* $\mathcal{S} = (\mathbb{P}, \lambda, \rightarrow, \mathbb{P}_I)$*, l'ensemble des* intervalles d'une position $\pi \in \mathbb{P}$*, noté* $\ddot{D}_\pi$*, est l'image de la fonction qui construit les deux approximations, la pessimiste et l'optimiste, par la rétro-propagation min-max de tous les préfixes finis de l'arbre de la position :*

$$\ddot{D}_\pi \triangleq \{(v_{p,h}(t), v_{p,h'}(t)) \mid t \in \widehat{T}(\mathcal{S}, \pi)\}$$

Pour toute position $\pi \in \mathbb{P}$*, le préordre sur les intervalles de* π *est défini de la façon suivante :*

$$(x, y) \leq_{\ddot{D}_\pi} (x', y') \quad \stackrel{\triangle}{\Longleftrightarrow} \quad x \leq_{D^{\pm}_{\pi}} x' \ \wedge \ y' \leq_{D^{\pm}_{\pi}} y$$

□

NOTATION 5.5.6. Pour alléger la notation la relation de préordre sur les intervalles $\leq_{\ddot{D}_\pi}$ sera notée plus simplement $\ddot{\leq}_\pi$. Tout élément de $\ddot{D}_\pi$ pourra être noté aussi bien (x, y) que $[x, y]$. Nous écrirons $z \in [x, y]$ pour signifier $x \leq_{D^{\pm}_{\pi}} z \leq_{D^{\pm}_{\pi}} y$.

Le préordre $\ddot{\leq}_\pi$ sur les intervalles est celui habituel de l'analyse mathématique. Intuitivement un premier intervalle $[x, y]$ est de qualité inférieure à un second $[x', y']$, nous noterons $[x, y] \ddot{\leq}_\pi [x', y']$, si le premier est plus large, donc moins informatif, que le second. Autrement dit, si le but des intervalles est celui de cerner un ensemble de valeurs, un intervalle est meilleur qu'un autre s'il cerne les valeurs d'une façon plus précise. La relation $\leq_{\ddot{D}_\pi}$ hérite trivialement les propriétés réflexive et transitive de la relation de préordre $\leq_{D^{\pm}_{\pi}}$ définies sur les valeurs pessimistes ou optimistes de la position π.

À nouveau, nous allons voir comment les intervalles d'une position π convergent vers une limite que nous appellerons *sémantique bisémique* de π. En effet, sur une structure de co-évaluation nous pouvons bâtir une *algèbre de co-évaluation* pour l'évaluation bisémique d'un jeu.

DEFINITION 5.5.7. (ALGÈBRE DE CO-ÉVALUATION POUR UN JEU) *Étant donné une structure de co-évaluation monotone* $\mathcal{D} \in \mathsf{Cstruct}(\mathcal{S}, D, \uparrow, \downarrow, p, h, h')$*, où* $\mathcal{D} = (D, \leq_D, \uparrow, \downarrow)$*, pour un jeu* $\mathcal{S} = (\mathbb{P}, \lambda, \rightarrow, \mathbb{P}_I)$*, l'*algèbre de co-évaluation *pour* $\mathcal{S}$ *, est la structure* $\ddot{\mathcal{D}}$ *sur la signature :*

$$
\begin{array}{llll}
(\mathsf{D} & & , & \textit{valeurs} \\
\ddot{\mathsf{D}} & & , & \textit{intervalles de valeurs} \\
\mathsf{D}^\flat & & , & \textit{classes co-finales} \\
\mathsf{D}^\sharp & & , & \textit{classes co-initiales} \\
\ddot{\mathsf{D}}^\natural & & ; & \textit{intervalles de classes} \\
\\
\leq & : \forall\alpha{:}\mathsf{sort}.\,1^{\alpha\times\alpha} & , & \textit{relations} \\
\geq & : \forall\alpha{:}\mathsf{sort}.\,1^{\alpha\times\alpha} & , & \textit{relations duales} \\
\uparrow & : \forall\alpha{:}\{\mathsf{D},\ddot{\mathsf{D}}\}.\,\alpha^* \to \alpha & , & \textit{fonction polymorphe du joueur} \\
\downarrow & : \forall\alpha{:}\{\mathsf{D},\ddot{\mathsf{D}}\}.\,\alpha^* \to \alpha & & \textit{fonction polymorphe de l'opposant} \\
)
\end{array}
$$

où :

– *la structure* $(\ddot{\mathsf{D}}, \ddot{\leq})$ *est la réunion des préordres sur les intervalles :*

$$\ddot{\mathsf{D}} \triangleq \bigcup_{\pi\in\mathbb{P}} \ddot{\mathsf{D}}_\pi \qquad \ddot{\leq} \triangleq \bigcup_{\pi\in\mathbb{P}} \ddot{\leq}_\pi$$

– *les structures* $(\mathsf{D}^\flat, \leq_{\mathsf{D}^\flat})$ *et* $(\mathsf{D}^\sharp, \leq_{\mathsf{D}^\sharp})$ *sont respectivement la réunion des extensions bémol et dièse :*

$$\mathsf{D}^\flat = \bigcup_{\pi\in\mathbb{P}} (\mathsf{D}^{\pm}_\pi)^\flat \qquad \leq_{\mathsf{D}^\flat} = \bigcup_{\pi\in\mathbb{P}} \leq_{(\mathsf{D}^{\pm}_\pi)^\flat}$$

$$\mathsf{D}^\sharp = \bigcup_{\pi\in\mathbb{P}} (\mathsf{D}^{\pm}_\pi)^\sharp \qquad \leq_{\mathsf{D}^\sharp} = \bigcup_{\pi\in\mathbb{P}} \leq_{(\mathsf{D}^{\pm}_\pi)^\sharp}$$

(les deux unions seront aussi notés, respectivement, $\leq^\flat_{\mathsf{D}}$ *et* $\leq^\sharp_{\mathsf{D}}$*)*

– *l'ensemble* $\ddot{\mathsf{D}}^\natural$ *est la réunion des* intervalles de classes *:*

$$\ddot{\mathsf{D}}^\natural = \bigcup_{\pi\in\mathbb{P}} \ddot{\mathsf{D}}^\natural_\pi$$

définis de la façon suivante :

$$\ddot{\mathsf{D}}^\natural_\pi \triangleq \{(\chi_1,\chi_2) \in (\mathsf{D}^{\pm}_\pi)^\flat \times (\mathsf{D}^{\pm}_\pi)^\sharp \mid \chi_1 \leq_{\wp^2(\mathsf{D}^{\pm}_\pi)} \chi_2\}$$

où l'ordre $\leq_{\wp^2(\mathsf{D}^{\pm}_\pi)}$ *entre les classes est l'extension de l'ordre* $\leq_{\mathsf{D}^{\pm}_\pi}$ *aux classes de parties de* $\mathsf{D}^{\pm}_\pi$ *(cf. section 1.1.9.2, page 23) ; pour les éléments de ce domaine, comme pour ceux de* $\ddot{\mathsf{D}}$*, nous utiliserons les notations équivalentes* (χ_1,χ_2) *et* $[\chi_1,\chi_2]$ *; et nous écrirons à nouveau* $\chi \in [\chi_1,\chi_2]$ *pour signifier* $\chi_1 \leq_{\wp^2(\mathsf{D}^{\pm}_\pi)} \chi \leq_{\wp^2(\mathsf{D}^{\pm}_\pi)} \chi_2$.

– *la relation* $\leq_{\ddot{\mathsf{D}}^\natural}$*, noté aussi* $\ddot{\leq}^\natural$*, est la réunion des relations* $\ddot{\leq}^\natural_\pi$ *sur les intervalles de classes :*

$$\ddot{\leq}^\natural = \bigcup_{\pi\in\mathbb{P}} \ddot{\leq}^\natural_\pi$$

définies de la façon suivante :

$$[\chi_1,\chi_2]\ \ddot{\leq}^\natural_\pi\ [\chi_1',\chi_2'] \quad \stackrel{\triangle}{\iff} \quad \chi_1 \leq^\flat_{\mathsf{D}_\pi} \chi_1' \ \wedge\ \chi_2' \leq^\sharp_{\mathsf{D}_\pi} \chi_2$$

– *les fonctions des joueurs sur les intervalles, notées* $\ddot{\uparrow}$ *et* $\ddot{\downarrow}$ *sont définies par :*

$$\ddot{\uparrow}_{i=1..n} [x_i, y_i] \triangleq [\uparrow_{i=1..n} x_i\,,\ \uparrow_{i=1..n} y_i]$$

$$\ddot{\downarrow}_{i=1..n} [x_i, y_i] \triangleq [\downarrow_{i=1..n} x_i\,,\ \downarrow_{i=1..n} y_i]$$

Les deux composantes $\mathcal{D}_{\_} = \mathcal{D}$ *et* $\mathcal{D}_{+} = \mathcal{D}^{\perp}$ *sont appelées, respectivement, structure d'évaluation ou projection* pessimiste*, notée aussi* $\ddot{\mathcal{D}}_1$*, et structure d'évaluation ou projection* optimiste*, notée aussi* $\ddot{\mathcal{D}}_2$*. Si* $(\mathsf{D}, \leq_{\mathsf{D}})$ *est un treillis et les opérations* $\uparrow$ *et* $\downarrow$ *coïncident respectivement avec les bornes, supérieure et inférieure, du treillis, alors la structure* $\ddot{\mathcal{D}}$ *est appelée* treillis de co-évaluation*.* ☐

Les fonctions des joueurs $\downarrow$ et $\uparrow$ étant *monotones* dans toutes les composantes, les extensions respectives $\ddot{\downarrow}$ et $\ddot{\uparrow}$ aux intervalles, seront des fonctions restituant encore des intervalles. Considérant alors les fonctions des joueurs comme des constructions polymorphes, leur type sera $\forall\alpha{:}\{\mathsf{D},\ddot{\mathsf{D}}\}.\ \alpha^* \to \alpha$.

Nous remarquerons que la structure $((\mathsf{D}^{\pm}_{\pi})^{\sharp}, \leq_{(\mathsf{D}^{\pm}_{\pi})^{\sharp}})$ coïncide avec l'extension bémol de $(\mathsf{D}^{\pm}_{\pi}, \geq_{\mathsf{D}^{\pm}_{\pi}})$. Intuitivement, les deux composantes, l'algèbre pessimiste $\mathcal{D}_{\_}$ et l'algèbre optimiste $\mathcal{D}_{+}$, représentent, en quelque sorte, un couple d'objets chiraux (énantiomorphe*)*, images l'un de l'autre dans un miroir (comme la main gauche et la main droite), mais n'étant pas égaux. Réunies, elles permettent une double évaluation d'un jeu où les deux évaluations sont en rapport : la plus petite, au sens de l'ordre $\leq_{\mathsf{D}^{\pm}_{\pi}}$, est *l'évaluation pessimiste*, la plus grande est *l'évaluation optimiste* ; et les deux sont fournies par les composantes algébriques homologues. Lorsque tout préordre $(\mathsf{D}^{\pm}_{\pi}, \leq_{\mathsf{D}^{\pm}_{\pi}})$ est, à la fois, un dcpo algébrique et un co-dcpo algébrique, les deux extensions, la bémol et la dièse, lui sont isomorphes. Dans ce cas, l'algèbre de co-évaluation $\ddot{\mathcal{D}}$ n'aura pas un intérêt majeur par rapport à la structure de co-évaluation $\mathcal{D}$ constituant son germe.

Réunissant les deux interprétations d'un jeu, celle dans la projection pessimiste $\mathcal{D}^{\flat}$, et celle dans la projection optimiste $\mathcal{D}^{\sharp} = (\mathcal{D}^{\perp})^{\flat}$, nous obtiendrons une co-évaluation du jeu. Intuitivement cela nous donnera un intervalle de valeurs cernant la sémantique du jeu à un point tel que, parfois, l'intervalle pourra se réduire à un seul élément, une forme de signifié absolu. À nouveau, tous les éléments de la sous-algèbre syntaxique principale du jeu :

$$(\mathbb{P}_{\mathrm{T}}, \mathbb{P}_{\mathrm{NT}}, \mathrm{T}(\mathbb{P}), \widehat{\mathrm{T}}(\mathcal{S}), \widehat{\mathrm{IT}}(\mathcal{S}), \mathrm{IT}(\mathcal{S})\,;\, \leq)$$

seront traduits dans l'algèbre de co-évaluation du jeu.

Example 5.5.8. Un modèle très simple d'algèbre de co-évaluation est celui des nombres rationnels $\mathbb{Q}$, illustré dans la figure 5.5.2, où l'on considère que les ensembles $\mathbb{Q}$ et $\mathbb{R}$ (les réels) contiennent aussi les limites $-\infty$ et $+\infty$. D'une part, puisque l'ordre est total, tous les sous-ensembles sont, à la fois, dirigés et co-dirigés. D'autre part, tout nombre irrationnel peut s'exprimer comme limite d'un dirigé (ou co-dirigé) de rationnels (il suffit d'approcher l'irrationnel x par une recherche dichotomique à partir de deux rationnels a et b l'encadrant, c'est-à-dire tels que $x \in$

FIG. 5.5.2. L'algèbre de co-évaluation des rationnels

D	$=$	$\mathbb{Q}$		
$\ddot{\mathsf{D}}$	$=$	$\{(x,y) \in \mathbb{Q} \times \mathbb{Q} \mid x \leq y\}$		
$\mathsf{D}^\flat$	$=$	$\Delta(\mathbb{Q}, \leq)_{\approx^\flat}$	$\cong$	$\mathbb{R}$
$\mathsf{D}^\sharp$	$=$	$\nabla(\mathbb{Q}, \leq)_{\approx^\sharp}$	$\cong$	$\mathbb{R}$
$\ddot{\mathsf{D}}^\natural$	$=$	$\{(\chi_1,\chi_2) \in \mathsf{D}^\flat \times \mathsf{D}^\sharp \mid \chi_1 \leq_{\wp^2(\mathsf{D})} \chi_2\}$	$\cong$	$\{(x,y) \in \mathbb{R} \times \mathbb{R} \mid x \leq y\}$
$\uparrow$	$=$	$\max$		
$\downarrow$	$=$	$\min$		
$\perp$	$=$	$-\infty$		
$\top$	$=$	$+\infty$		
$\ddot{\perp}$	$=$	$[-\infty, +\infty]$		

$[a, b]$). L'extension bémol et l'extension dièse du domaine des rationnels, sont alors isomorphes au domaine des réels. Nous verrons que, dans une telle structure, un jeu infini sera évalué à un couple de valeurs $[x, y]$, qui ne seront pas systématiquement rationnels. La valeur gauche, que nous qualifions de pessimiste, pourra être soit un réel, soit un rationnel, et de même pour la valeur droite, que nous qualifions d'optimiste. Ainsi, l'attribution d'une sémantique bisémique à une position racine d'un arbre de jeu infini, pourra se faire dans le produit cartésien de l'ensemble initial $\mathbb{Q} \times \mathbb{Q}$, ou dans un produit cartésien mixte $\mathbb{Q} \times \mathbb{R}$ ou $\mathbb{R} \times \mathbb{Q}$, si l'on nécessite la complétion idéaux pour la limite pessimiste ou pour celle optimiste, ou bien dans le carré $\mathbb{R} \times \mathbb{R}$, si l'on nécessite la complétion idéaux pour capturer les deux limites. □

5.5.1. Co-interprétation d'un jeu. Une structure de co-évaluation monotone $\mathcal{D}$ suffit pour définir l'interprétation du jeu dans l'algèbre de co-évaluation $\ddot{\mathcal{D}}$, c'est-à-dire pour définir ce que nous appellerons co-interprétation du jeu.

DEFINITION 5.5.9. (CO-INTERPRÉTATION D'UN JEU) *Soit $\mathcal{D} \in \mathsf{Cstruct}(\mathcal{S}, \mathsf{D}, \uparrow, \downarrow, p, h_1, h_2)$ une structure de co-évaluation monotone, où $\mathcal{D} = (\mathsf{D}, \leq_{\mathsf{D}}, \uparrow, \downarrow)$, pour un jeu $\mathcal{S} = (\mathbb{P}, \lambda, \rightarrow, \mathbb{P}_I)$. Soit $\varphi_1 : \mathsf{A}^p_{\mathcal{S}} \xmapsto{\mathcal{H}} \mathcal{D}^\flat_1$, où $\varphi_1 = (p, h_1, v_1, w_1)$, l'interprétation pessimiste du jeu ($\varphi_1 = \varphi_{\mathcal{D}}$), et soit $\varphi_2 : \mathsf{A}^p_{\mathcal{S}} \xmapsto{\mathcal{H}} \mathcal{D}^\flat_2$, où $\varphi_2 = (p, h_2, v_2, w_2)$, l'interprétation optimiste du jeu ($\varphi_1 = \varphi_{\mathcal{D}^\perp}$). Alors, la* co-interprétation du jeu *$(\ddot{\psi}, \ddot{\varphi}) : \mathsf{A}^p_{\mathcal{S}} \xmapsto{\mathcal{H}} \ddot{\mathcal{D}}$, où*

$$\ddot{\mathcal{D}}{=}(\mathsf{D}, \mathsf{D}^2, \ddot{\mathsf{D}}, \mathsf{D}^\flat, \mathsf{D}^\sharp, \ddot{\mathsf{D}}^\natural\,; \leq, \geq, \uparrow, \downarrow)$$

est définie par :

(1) *l'interprétation des noms $\ddot{\psi}$ est l'application suivante :*

(a) *pour les noms des domaines :*

$$\ddot{\psi}(\sigma) = \begin{cases} \mathsf{D} & \text{si} \quad \sigma = \mathbb{P}_{\mathrm{T}} \\ \ddot{\mathsf{D}} & \text{si} \quad \sigma \in \{\mathbb{P}_{\mathrm{NT}}, \mathrm{T}(\mathbb{P}), \widehat{\mathrm{T}}(\mathcal{S})\} \\ \ddot{\mathsf{D}}^\natural & \text{si} \quad \sigma \in \{\widehat{\mathrm{IT}}(\mathcal{S}), \mathrm{IT}(\mathcal{S})\} \end{cases}$$

(b) *pour les noms des opérations :*

$$\psi(\leq) \triangleq (\leq)$$

(2) *l'interprétation des valeurs est la construction polymorphe :*

$$\ddot{\varphi} \,:\, \forall\alpha{:}\{\mathbb{P}_{\mathrm{T}}, \mathbb{P}_{\mathrm{NT}}, \mathrm{T}(\mathbb{P}), \widehat{\mathrm{T}}(\mathcal{S}), \widehat{\mathrm{IT}}(\mathcal{S}), \mathrm{IT}(\mathcal{S})\}.\, \alpha \to \psi(\alpha)$$

définie par :

(a) $\ddot{\varphi}_{\mathbb{P}_{\mathrm{T}}} \triangleq \mathrm{p}$

(b) $\ddot{\varphi}_{\mathbb{P}_{\mathrm{NT}}}(\pi) \triangleq [\mathrm{h}_1(\pi), \mathrm{h}_2(\pi)]$ *est la* co-heuristique*, notée aussi* $\ddot{\mathrm{h}}$ *; les* projections $\ddot{\mathrm{h}}_1 \triangleq \mathrm{h}_1$ *et* $\ddot{\mathrm{h}}_2 \triangleq \mathrm{h}_2$ *seront respectivement la heuristique pessimiste et la heuristique optimiste*

(c) $\ddot{\varphi}_{\mathrm{T}(\mathbb{P})}(\mathrm{t}) \triangleq [v_1(\mathrm{t}), v_2(\mathrm{t})]$*, notée aussi* $\ddot{v}$*, est la* co-évaluation *des arbres étiquetés ;* $\ddot{v}_1 \triangleq v_1$ *est l'évaluation* pessimiste*, et* $\ddot{v}_2 \triangleq v_2$ *est l'évaluation* optimiste *des arbres finis étiquetés*

(d) $\ddot{\varphi}_{\widehat{\mathrm{T}}(\mathcal{S})} \triangleq (\ddot{\varphi}_{\mathrm{T}(\mathbb{P})})_{|\widehat{\mathrm{T}}(\mathcal{S})}$ *est la* co-évaluation *des préfixes finis*

(e) $\ddot{\varphi}_{\widehat{\mathrm{IT}}(\mathcal{S})} \triangleq [w_1(\mathrm{t}), w_2(\mathrm{t})]$*, notée aussi* $\ddot{w}$*, est la* co-évaluation *des préfixes arbitraires ;* $\ddot{w}_1 \triangleq w_1$ *est l'évaluation* pessimiste*, et* $\ddot{w}_2 \triangleq w_2$ *est l'évaluation* optimiste *des préfixes arbitraires*

(f) $\ddot{\varphi}_{\mathrm{IT}(\mathcal{S})} \triangleq (\ddot{\varphi}_{\widehat{\mathrm{IT}}(\mathcal{S})})_{|\mathrm{IT}(\mathcal{S})}$ *est la* co-évaluation *des arbres de jeu*

□

Une co-interprétation pourra être notée par ses quatre composantes principales $\ddot{\varphi}_{\mathcal{D}} = (\mathrm{p}, \ddot{\mathrm{h}}, \ddot{v}, \ddot{w})$. S'il existe un élément minimal $\bot$ et un maximal $\top$ pour la structure $(\mathrm{D}, \leq)$ et si la co-heuristique est la fonction $\lambda\pi.(\bot, \top)$ nous la qualifierons de *canonique* (ou *banale*). La co-heuristique banale restitue toujours l'intervalle moins significatif pour une position donnée.

5.5.1.1. *Preuve de correction.* Nous allons prouver, par le biais de plusieurs lemmes, que la co-interprétation d'un jeu est, comme son nom l'indique, une *interprétation algébrique* de la structure syntaxique principale vers l'algèbre de co-évaluation. Nous prouverons ensuite qu'il s'agit d'un *homomorphisme*.

Le lemme suivant confirme l'intuition qui accompagne les co-heuristiques correctes : dans le cas des arbres finis, elles peuvent être utilisées, pour toute position non terminale, pour élaborer deux approximations de la valeur exacte du jeu, une pessimiste (approximation *par défaut*) et une optimiste (approximation *par excès*).

LEMMA 5.5.10. *Dans une structure de co-évaluation monotone* $\mathcal{D} \in \mathsf{Cstruct}(\mathcal{S}, \mathrm{D}, \uparrow, \downarrow, \mathrm{p}, \ddot{\mathrm{h}}_1, \ddot{\mathrm{h}}_2)$*, la co-heuristique* $\ddot{\mathrm{h}} : \mathbb{P}_{\mathrm{NT}} \to \ddot{\mathrm{D}}$*, est telle que, si* $\mathrm{t}_\pi \in \mathrm{T}(\mathcal{S})$*, alors :*

$$val(\mathrm{t}_\pi) \in \ddot{\mathrm{h}}(\pi).$$

DÉMONSTRATION. *L'énoncé se traduit, en considérant les projections pessimiste et optimiste, par les deux inégalités suivantes :*

$$\ddot{h}_1(\pi) \;\leq_{D^{\pm}_{\pi}}\; \mathit{val}(t_\pi) \;\leq_{D^{\pm}_{\pi}}\; \ddot{h}_2(\pi)$$

Et les deux sont conséquences directes de la correction des composantes $\ddot{h}_1$ et $\ddot{h}_2$ respectivement dans les structures d'évaluation pessimiste $\mathcal{D}$ et optimiste $\mathcal{D}^{\perp}$ (cf. prop. 5.3.9). □

L'interprétation bisémique d'un préfixe fini est l'assemblage de deux interprétations monosémiques duales, une *pessimiste* $\ddot{v}_1$, obtenue grâce à l'heuristique pessimiste, et une *optimiste* $\ddot{v}_2$, obtenue grâce à l'heuristique optimiste.

LEMMA 5.5.11. *La co-interprétation des préfixes finis est une fonction à valeurs dans les intervalles :*

$$\ddot{v} : \widehat{T}(\mathcal{S}, \pi) \to \ddot{D}_\pi.$$

DÉMONSTRATION. *L'énoncé se traduit, en termes de projections pessimiste et optimiste, par la condition $\ddot{v}_1(t) \leq_{D^{\pm}_{\pi}} \ddot{v}_2(t)$ pour tout $t \in \widehat{T}(\mathcal{S}, \pi)$. Cette condition est prouvée par une simple induction sur la taille de l'arbre fini, en utilisant, pour le cas de base, l'hypothèse :*

$$h_1(\pi) \leq_{D^{\pm}_{\pi}} h_2(\pi)$$

de la définition de structure de co-évaluation monotone (cf. déf. 5.5.4). □

Nous pouvons remarquer à présent que, par définition, la co-évaluation des préfixes finis peut être formulée d'une façon semblable à ses projections, par le biais des fonctions des joueurs sur les intervalles :

$$\ddot{v}(t) = \begin{cases} \ddot{\bot} & \text{si } t = \emptyset \\ \ddot{p}(\pi) & \text{si } t = \{(\varepsilon, \pi)\},\ \pi \in \mathbb{P}_T \\ \ddot{h}(\pi) & \text{si } t = \{(\varepsilon, \pi)\},\ \pi \in \mathbb{P}_{NT} \\ \ddot{\uparrow}_{i=1..n}\ \ddot{v}(t_i) & \text{si Fils}(t) = t_1..t_n,\ \lambda(t(\varepsilon)) = \text{Player} \\ \ddot{\downarrow}_{i=1..n}\ \ddot{v}(t_i) & \text{si Fils}(t) = t_1..t_n,\ \lambda(t(\varepsilon)) = \text{Opponent} \end{cases}$$

La structure de $\ddot{v}$ demeure celle d'une simple fonction d'évaluation. Seul le type des valeurs est différent, les intervalles de valeurs ayant remplacé les valeurs singulières. En outre, la co-interprétation des préfixes finis vérifie, comme ses deux projections, la propriété de monotonie. Si pour les projections le préordre est celui des valeurs $\leq_{D^{\pm}_{\pi}}$, pour la co-interprétation le préordre que la fonction conserve est celui des intervalles $\ddot{\leq}_\pi$.

LEMMA 5.5.12. *La co-interprétation des préfixes finis est une fonction monotone $\ddot{v} : (\widehat{T}(\mathcal{S}, \pi), \leq) \xrightarrow{\mathcal{H}} (\ddot{D}^{\natural}_{\pi}, \ddot{\leq}^{\natural}_{\pi})$.*

DÉMONSTRATION. *L'énoncé découle directement de la proposition 5.3.12. Soit* $t \leq t'$ *deux préfixes de jeu de la position* π. *La projection* pessimiste *de la co-heuristique est correcte dans la projection* pessimiste $\mathcal{D}_-$ *de la structure de co-évaluation, donc* $\ddot{v}_1(t) \leq_{D^{\pm}_{\pi}} \ddot{v}_1(t')$. *De même, la projection* optimiste *de la co-heuristique est correcte dans la projection optimiste* $\mathcal{D}_+$, *donc* $\ddot{v}_2(t) \geq_{D^{\pm}_{\pi}} \ddot{v}_2(t')$. *Autrement dit :*

$$\ddot{v}(t) \quad = \quad [\ddot{v}_1(t), \ddot{v}_2(t)] \quad \ddot{\leq}_{\pi} \quad [\ddot{v}_1(t'), \ddot{v}_2(t')] \quad = \quad \ddot{v}(t')$$

□

L'interprétation bisémique des préfixes infinis est, elle aussi, la réunion des interprétations monosémiques correspondantes aux deux germes de l'algèbre de co-évaluation. La co-interprétation des préfixes arbitraires est l'intervalle composé de la classe de co-finalité des évaluations pessimiste et de la classe de co-initialité des évaluations optimistes :

$$\ddot{w}(t) = [\ [\{v_-(t/^k) \mid k \in \mathbb{N}\}]_\flat\ ,\ [\{v_+(t/^k) \mid k \in \mathbb{N}\}]_\sharp\ ]$$

PROPOSITION 5.5.13. *La co-évaluation des préfixes arbitraire est une fonction à valeurs dans les intervalles de classes :*

$$\ddot{w} : \widehat{\mathrm{IT}}(\mathcal{S}, \pi) \to \ddot{\mathrm{D}}^{\natural}_{\pi}.$$

DÉMONSTRATION. *Soit* $(\chi_1, \chi_2) = \ddot{w}(t)$ *pour* $t \in \widehat{\mathrm{IT}}(\mathcal{S}, \pi)$. *Nous voulons montrer que* $\chi_1 \leq_{\wp^2(D^{\pm}_{\pi})} \chi_2$. *Soit* $X \in \chi_1$ *et* $Y \in \chi_2$. *Il nous faut prouver que* $X \leq_{\wp(D^{\pm}_{\pi})} Y$. *Soit* $x \in X$ *et* $y \in Y$ *deux éléments arbitraires. Puisque* X *est co-final de* $\{\ddot{v}_1(t/^k) \mid k \in \mathbb{N}\}$, *il existe* $i \in \mathbb{N}$ *tel que* $x \leq_{D^{\pm}_{\pi}} \ddot{v}_1(t/^i)$. *D'autre part, puisque* Y *est co-initial de* $\{\ddot{v}_2(t/^k) \mid k \in \mathbb{N}\}$, *il existe* $j \in \mathbb{N}$ *tel que* $\ddot{v}_2(t/^j) \leq_{D^{\pm}_{\pi}} y$. *Soit* $k = \max\{i, j\}$. *Puisque* $\ddot{v}_1$ *est monotone pour l'ordre* $\leq_{D^{\pm}_{\pi}}$ *et que* $\ddot{v}_2$ *est monotone pour l'ordre* $\geq_{D^{\pm}_{\pi}}$, *nous avons :*

$$x \leq_{D^{\pm}_{\pi}} \ddot{v}_1(t/^i) \leq_{D^{\pm}_{\pi}} \ddot{v}_1(t/^k) \leq_{D^{\pm}_{\pi}} \ddot{v}_2(t/^k) \leq_{D^{\pm}_{\pi}} \ddot{v}_2(t/^j) \leq_{D^{\pm}_{\pi}} y$$

où la condition $\ddot{v}_1(t/^k) \leq_{D^{\pm}_{\pi}} \ddot{v}_2(t/^k)$ *est garantie par le lemme 5.5.11. L'énoncé est donc obtenu par la propriété transitive des préordres* $(D^{\pm}_{\pi}, \leq_{D^{\pm}_{\pi}})$ *constituant une structure de co-évaluation.* □

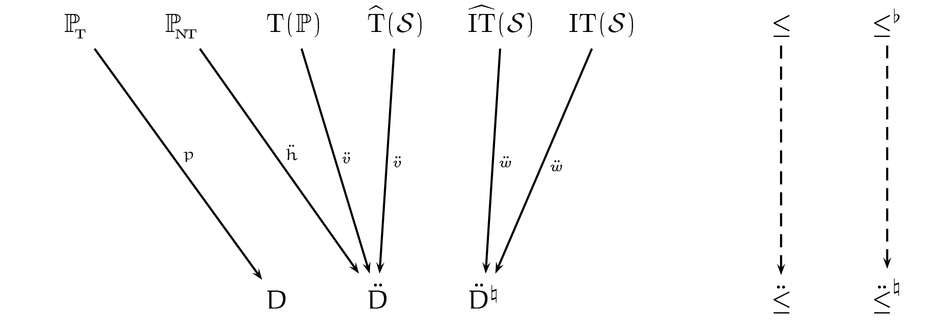
La co-évaluation des préfixes arbitraires, comme celle des préfixes finis, est une fonction monotone vis-à-vis de l'ordre sur les intervalles.

LEMMA 5.5.14. *La co-interprétation des préfixes finis est une fonction monotone* $\ddot{v} : (\widehat{\mathrm{T}}(\mathcal{S}, \pi), \leq) \xrightarrow{\mathcal{H}} (\ddot{\mathrm{D}}^{\natural}_{\pi}, \ddot{\leq}^{\natural}_{\pi})$.

DÉMONSTRATION. *La preuve est identique à celle de la proposition 5.5.12, à la différence près que l'argument initial, i.e. la monotonie des projections, est fourni, cette fois-ci, par la proposition 5.3.14.* □

FIG. 5.5.3. Co-interprétation d'un jeu

SOUS-ALGÈBRE SYNTAXIQUE PRINCIPALE



ALGÈBRE DE CO-ÉVALUATION

PROPOSITION 5.5.15. *La définition de co-interprétation d'un jeu est correcte et il s'agit d'un homomorphisme.*

DÉMONSTRATION. *La traduction du type* $1^{\widehat{\mathrm{T}}(\mathcal{S})\times\widehat{\mathrm{T}}(\mathcal{S})}$ *de* $\leq_{\widehat{\mathrm{T}}(\mathcal{S})}$ *est précisément le type* $1^{\ddot{\mathrm{D}}\times\ddot{\mathrm{D}}}$ *de* $\ddot{\psi}(\leq)_{\ddot{\psi}(\widehat{\mathrm{T}}(\mathcal{S}))} = \ddot{\psi}(\leq)_{\ddot{\mathrm{D}}} = \ddot{\leq}$. *De même, la traduction du type* $1^{\widehat{\mathrm{IT}}(\mathcal{S})\times\widehat{\mathrm{IT}}(\mathcal{S})}$ *de* $\leq_{\widehat{\mathrm{IT}}(\mathcal{S})}$ *est précisément le type* $1^{\ddot{\mathrm{D}}^\natural\times\ddot{\mathrm{D}}^\natural}$ *de* $\ddot{\psi}(\leq)_{\ddot{\psi}(\widehat{\mathrm{IT}}(\mathcal{S}))} = \ddot{\psi}(\leq)_{\ddot{\mathrm{D}}^\natural} = \ddot{\leq}^\natural$. *Autrement dit, l'interprétation des noms est correcte vis-à-vis des types. La conservation des opérations est garantie, comme dans le cas d'une interprétation du jeu (cf. prop. 5.3.18), par les propriétés de monotonie démontrées auparavant : le semi-prédicat* $\leq_{\widehat{\mathrm{T}}(\mathcal{S})}: 1^{\widehat{\mathrm{T}}(\mathcal{S})\times\widehat{\mathrm{T}}(\mathcal{S})}$ *est conservé dans* $\ddot{\psi}(\leq)_{\ddot{\psi}(\widehat{\mathrm{T}}(\mathcal{S}))} = \ddot{\leq}$ *par* $\ddot{v}$ *qui est monotone (cf. lemme 5.5.12), et le semi-prédicat* $\leq_{\widehat{\mathrm{IT}}(\mathcal{S})}$: $1^{\widehat{\mathrm{IT}}(\mathcal{S})\times\widehat{\mathrm{IT}}(\mathcal{S})}$ *est conservé dans* $\ddot{\psi}(\leq)_{\ddot{\psi}(\widehat{\mathrm{IT}}(\mathcal{S}))} = \ddot{\leq}^\natural$ *par* $\ddot{w}$ *qui est monotone (cf. lemme 5.5.14).* □

Les composantes de l'interprétation des valeurs (lignes continues) et les relations de conservation des opérations (lignes pointillées) impliquées par la co-interprétation d'un jeu sont résumées dans la figure 5.5.3.

5.5.2. Sémantiques bisémiques des positions. Nous avons vu que, étant donné une structure d'évaluation $\mathcal{D}$ dont le domaine est D, son extension bémol $\mathcal{D}^\flat$, nous permet d'évaluer les positions du jeu dans les classes de co-finalité des dirigés de D. La fonction d'interprétation sémantique correspondante, la *sémantique monosémique bémol* $(\!|\,.\,|\!)^\flat_{\mathcal{D}} : \mathbb{P} \to \mathrm{D}^\flat$, associe alors à toute position du jeu une classe de co-finalité. Par ailleurs, la *sémantique monosémique* $(\!|\,.\,|\!)_{\mathcal{D}} : \mathbb{P} \multimap \mathrm{D}$ des positions est la borne supérieure, si elle existe, de la sémantique monosémique bémol. En ce qui concerne la co-évaluation, nous allons procéder d'une façon similaire : en regroupant deux structures d'évaluation compatibles et duales (l'ordre des valeurs étant dual), nous obtiendrons une structure de co-évaluation nous permettant d'évaluer une position dans un intervalle de classes. Ainsi, lorsque les deux classes auront une borne, respectivement supérieure et inférieure, l'intervalle constitué par ces bornes pourra être considéré comme la (double) valeur sémantique de la position du jeu. De plus, lorsque les deux bornes coïncideront à une valeur $\nu \in \mathsf{D}$, nous pourrons considérer cette dernière comme étant l'*unique* sémantique de la position.

Definition 5.5.16. (Sémantique bisémique) *Étant donné une structure de co-évaluation monotone* $\mathcal{D} \in \mathsf{Cstruct}(\mathcal{S}, \mathsf{D}, \uparrow, \downarrow, p, h_1, h_2)$ *pour un jeu* $\mathcal{S} = (\mathbb{P}, \lambda, \rightarrow, \mathbb{P}_{\mathrm{I}})$*, la* sémantique bisémique bécarre *des positions est la fonction totale* $[\![\,.\,]\!]^{\natural}_{\mathcal{D}} : \mathbb{P} \rightarrow \ddot{\mathsf{D}}^{\natural}$ *définie par :*

$$[\![\,\pi\,]\!]^{\natural}_{\mathcal{D}} \triangleq \ddot{w}(t_\pi)$$

où $\ddot{\varphi}_{\mathcal{D}} = (p, \ddot{h}, \ddot{v}, \ddot{w})$ *est la co-interprétation du jeu dans l'algèbre de co-évaluation* $\ddot{\mathcal{D}}$ *:*

$$\ddot{\mathcal{D}} = (\mathsf{D}, \mathsf{D}^2, \ddot{\mathsf{D}}, \mathsf{D}^\flat, \mathsf{D}^\sharp, \ddot{\mathsf{D}}^\natural\,;\, \leq, \geq, \uparrow, \downarrow)$$

La sémantique bécarre pessimiste, *notée* $\lfloor\!\lfloor\,.\,\rfloor\!\rfloor^{\flat}_{\mathcal{D}}$*, est la projection de la première composante de la sémantique bisémique bécarre. De même, la* sémantique bécarre optimiste, *notée* $\lceil\!\lceil\,.\,\rceil\!\rceil^{\sharp}_{\mathcal{D}}$*, est la projection de la seconde composante :*

$$\lfloor\!\lfloor\,\pi\,\rfloor\!\rfloor^{\flat}_{\mathcal{D}} = \chi_1 \quad \wedge \quad \lceil\!\lceil\,\pi\,\rceil\!\rceil^{\sharp}_{\mathcal{D}} = \chi_2 \quad \overset{\triangle}{\Longleftrightarrow} \quad [\![\,\pi\,]\!]^{\natural}_{\mathcal{D}} = [\chi_1, \chi_2]$$

La sémantique bisémique *des positions est la fonction partielle* $[\![\,.\,]\!]_{\mathcal{D}} : \mathbb{P} \rightharpoonup \ddot{\mathsf{D}}$ *définie par :*

$$[\![\,\pi\,]\!]_{\mathcal{D}} \overset{\leftharpoonup}{=} \sup_{\ddot{\mathsf{D}}} [\![\,\pi\,]\!]^{\natural}_{\mathcal{D}}$$

La sémantique pessimiste, *notée* $\lfloor\!\lfloor\,.\,\rfloor\!\rfloor_{\mathcal{D}}$ *et la* sémantique optimiste, *notée* $\lceil\!\lceil\,.\,\rceil\!\rceil_{\mathcal{D}}$*, sont égales à la sémantique monosémique dans les projections, respectivement, pessimistes et optimistes de la structure de co-évaluation* $\mathcal{D}$ *et de la co-interprétation* $\ddot{\varphi}_{\mathcal{D}}$ *:*

$$\begin{cases} \lfloor\!\lfloor\,\pi\,\rfloor\!\rfloor_{\mathcal{D}} \triangleq (\!|\,\pi\,|\!)_{\mathcal{D}} \\ \lceil\!\lceil\,\pi\,\rceil\!\rceil_{\mathcal{D}} \triangleq (\!|\,\pi\,|\!)_{\mathcal{D}^\perp} \end{cases}$$

□

En ce qui concerne la notation des fonctions sémantiques bisémiques et de leurs projections, nous omettrons les indices lorsque cela ne pourra pas être cause d'ambiguïté. Nous remarquerons que la sémantique pessimiste $\lfloor\!\lfloor\,.\,\rfloor\!\rfloor_{\mathcal{D}}$, et l'optimiste $\lceil\!\lceil\,.\,\rceil\!\rceil_{\mathcal{D}}$ coïncident avec les deux projections de sémantique bisémique lorsque cette dernière existe :

$$\lfloor\!\lfloor\,\pi\,\rfloor\!\rfloor_{\mathcal{D}} = x_1 \quad \wedge \quad \lceil\!\lceil\,\pi\,\rceil\!\rceil_{\mathcal{D}} = x_2 \quad \Leftrightarrow \quad [\![\,\pi\,]\!]_{\mathcal{D}} = [x_1, x_2]$$

Cependant, la sémantique bisémique peut ne pas exister pour une certaine position π du jeu, alors qu'une sémantique entre la pessimiste et l'optimiste peut être définie sur π (mais pas les deux simultanément).

Proposition 5.5.17. *Étant donné une structure de co-évaluation monotone* $\mathcal{D} \in \mathsf{Cstruct}(\mathcal{S}, \mathsf{D}, \uparrow, \downarrow, p, h_1, h_2)$ *pour un jeu* $\mathcal{S} = (\mathbb{P}, \lambda, \rightarrow, \mathbb{P}_{\mathrm{I}})$*, nous avons :*

(1) $\lfloor\!\lfloor\,\pi\,\rfloor\!\rfloor^{\flat}_{\mathcal{D}} = (\!|\,\pi\,|\!)^{\flat}_{\mathcal{D}}$

(2) $\lceil\!\lceil\,\pi\,\rceil\!\rceil^{\sharp}_{\mathcal{D}} = (\!|\,\pi\,|\!)^{\flat}_{\mathcal{D}^\perp}$

(3) $[\![\,\pi\,]\!]_{\mathcal{D}} = [x_1, x_2] \quad \Leftrightarrow \quad \sup(\!|\,\pi\,|\!)^{\flat}_{\mathcal{D}} = x_1 \;\wedge\; \inf(\!|\,\pi\,|\!)^{\flat}_{\mathcal{D}} = x_2$

TAB. 1. Mémento des fonctions sémantiques

$$\text{avec} \begin{cases} \varphi = (p, h, v, w) \\ \ddot{\varphi} = (p, \ddot{h}, \ddot{v}, \ddot{w}) \end{cases}$$

Nom	Notation	Type		Définition
monosémique bémol	$(\!\|\pi\|\!)^\flat$	$\mathbb{P} \longrightarrow D^\flat$	*totale*	$w(t_\pi)$
monosémique	$(\!\|\pi\|\!)$	$\mathbb{P} \multimap D$	*partielle*	$\sup w(t_\pi)$
bisémique bécarre	$[\![\pi]\!]^\natural$	$\mathbb{P} \longrightarrow \ddot{D}^\natural$	*totale*	$\ddot{w}(t_\pi)$
pessimiste bémol	$\llcorner\pi\lrcorner^\flat$	$\mathbb{P} \longrightarrow D^\flat$	*totale*	$\ddot{w}_1(t_\pi)$
optimiste dièse	$\ulcorner\pi\urcorner^\sharp$	$\mathbb{P} \longrightarrow D^\sharp$	*totale*	$\ddot{w}_2(t_\pi)$
bisémique	$[\![\pi]\!]$	$\mathbb{P} \multimap \ddot{D}$	*partielle*	$\sup \ddot{w}(t_\pi)$
pessimiste	$\llcorner\pi\lrcorner$	$\mathbb{P} \multimap D$	*partielle*	$\sup \ddot{w}_1(t_\pi)$
optimiste	$\ulcorner\pi\urcorner$	$\mathbb{P} \multimap D$	*partielle*	$\inf \ddot{w}_2(t_\pi)$

DÉMONSTRATION. *Le deux premiers points sont des conséquences directes des définitions :*

$$\llcorner\pi\lrcorner^\flat_{\mathcal{D}} = \left([\![\pi]\!]^\natural_{\mathcal{D}}\right)_1 = \ddot{w}_1(t_\pi) = (\!|\pi|\!)^\flat_{\mathcal{D}}$$
$$\ulcorner\pi\urcorner^\sharp_{\mathcal{D}} = \left([\![\pi]\!]^\natural_{\mathcal{D}}\right)_2 = \ddot{w}_2(t_\pi) = (\!|\pi|\!)^\flat_{\mathcal{D}^\perp}\ .$$

Le dernier point est une propriété de l'ordre sur les intervalles. La borne supérieure d'un dirigé d'intervalles coïncide avec le couple des bornes supérieures des deux projections du dirigé :

$$\begin{aligned} [\![\pi]\!]_{\mathcal{D}} = [x_1, x_2] \quad & \Leftrightarrow \quad \sup [\![\pi]\!]^\natural_{\mathcal{D}} = [x_1, x_2] \\ & \Leftrightarrow \quad \sup(\!|\pi|\!)^\flat_{\mathcal{D}} = x_1 \ \wedge\ \sup(\!|\pi|\!)^\flat_{\mathcal{D}^\perp} = x_2 \\ & \Leftrightarrow \quad \sup(\!|\pi|\!)^\flat_{\mathcal{D}} = x_1 \ \wedge\ \inf(\!|\pi|\!)^\flat_{\mathcal{D}} = x_2 \end{aligned}$$

□

REMARK 5.5.18. La proposition précédente et la remarque 5.4.2 nous permettent de réécrire la fonction sémantique bisémique selon le schéma habituel, inauguré par la définition de valeur d'une position d'un jeu fini, et qui reste valide, comme nous l'avons constaté, pour l'interprétation des préfixes arbitraires w et pour la sémantique monosémique bémol $(\!|\,.\,|\!)^\flat$:

$$[\![\pi]\!]^\natural = \begin{cases} (\ [\{p(\pi)\}]_\flat\,,\ [\{p(\pi)\}]_\sharp\ ) & \text{si}\ \pi \in \mathbb{P}_T \\ (\ [\{\ddot{h}_1(\pi)\}]_\flat\,,\ [\{\ddot{h}_2(\pi)\}]_\sharp\ ) & \text{si}\ t = \{(\varepsilon, \pi)\},\ \pi \in \mathbb{P}_{NT} \\ \ddot{\uparrow}_{i=1}^{n} [\![\pi_i]\!]^\natural & \text{si}\ \mathrm{Succ}(\pi) = \pi_1..\pi_n,\ \ \lambda(\pi) = \text{Player} \\ \ddot{\downarrow}_{i=1}^{n} [\![\pi_i]\!]^\natural & \text{si}\ \mathrm{Succ}(\pi) = \pi_1..\pi_n,\ \ \lambda(\pi) = \text{Opponent} \end{cases}$$

□

La table 1 résume l'ensemble de toutes les fonctions sémantiques définies pour les positions d'un jeu. Le schéma de la remarque 5.5.18 reste valide pour toutes les fonctions sémantiques répertoriées. Mais pour les fonctions partielles $(\!|\,.\,|\!)_{\mathcal{D}}$, $[\![\,.\,]\!]_{\mathcal{D}}$, $\llcorner\,.\,\lrcorner_{\mathcal{D}}$ et $\ulcorner\,.\,\urcorner_{\mathcal{D}}$, il faudra plus précisément substituer le symbole d'égalité par le symbole $\stackrel{\leftarrow}{=}$ (égal s'il existe), puisque l'existence du membre gauche (la sémantique d'une position), est impliquée par l'existence du membre droit (les sémantiques des positions suivantes).

Si la structure de co-évaluation est, à la fois, un dcpo algébrique et un co-dcpo co-algébrique (pour toute position π), la sémantique des positions d'un jeu devient une fonction *totale*.

COROLLARY 5.5.19. *Soit* $\mathcal{D} \in \mathsf{Cstruct}(\mathcal{S}, \mathsf{D}, \uparrow, \downarrow, \mathsf{p}, \mathsf{h}_1, \mathsf{h}_2)$ *une structure de co-évaluation monotone pour un jeu* $\mathcal{S} = (\mathbb{P}, \lambda, \rightarrow, \mathbb{P}_{\mathrm{I}})$*. Si pour toute position* $\pi \in \mathbb{P}$ *la structure* $(\mathsf{D}^{\pm}_{\pi}, \leq_{\mathsf{D}^{\pm}_{\pi}})$ *est à la fois un dcpo algébrique et un co-dcpo co-algébrique, alors la sémantique bisémique des positions du jeu est une fonction totale :*

$$[\![\,.\,]\!] : \mathbb{P} \rightarrow \ddot{\mathsf{D}}$$

DÉMONSTRATION. *L'énoncé est une conséquence directe des propositions 5.4.3 et 5.5.17.* □

Le rapport entre la co-interprétation $\ddot{v}$ et la sémantique bisémique $[\![\,.\,]\!]_{\mathcal{D}}$ respecte la règle énoncée, dans le cadre des évaluations, par le lemme 5.4.5 : la fonction $\ddot{v}$ peut être utilisée pour calculer des approximations de $[\![\,.\,]\!]_{\mathcal{D}}$, lorsque cette dernière existe.

LEMMA 5.5.20. *Soit* $\mathcal{D} \in \mathsf{Cstruct}(\mathcal{S}, \mathsf{D}, \uparrow, \downarrow, \mathsf{p}, \mathsf{h}_1, \mathsf{h}_2)$ *une structure de co-évaluation monotone pour un jeu* $\mathcal{S} = (\mathbb{P}, \lambda, \rightarrow, \mathbb{P}_{\mathrm{I}})$*. Alors*

$$\forall \pi \in \mathrm{dom}\, [\![\,.\,]\!]_{\mathcal{D}} \,.\; \forall \mathsf{t} \in \widehat{\mathsf{T}}(\mathcal{S}, \pi). \qquad \ddot{v}(\mathsf{t}) \mathrel{\ddot{\leq}_{\pi}} [\![\,\pi\,]\!]_{\mathcal{D}}$$

DÉMONSTRATION. *Si la sémantique bisémique existe, alors les sémantiques pessimiste et optimiste existent (cf. prop. 5.5.17). En utilisant deux fois le lemme 5.4.5 (dans la structure pessimiste* $\mathcal{D}$ *et dans celle optimiste* $\mathcal{D}^{\perp}$*, où l'ordre sur les valeurs de* π *est* $\geq_{\mathsf{D}^{\pm}_{\pi}}$*), nous avons :*

$$\begin{aligned} \ddot{v}_1(\mathsf{t}) &\leq_{\mathsf{D}^{\pm}_{\pi}} \llfloor \pi \rrfloor_{\mathcal{D}} \\ \ddot{v}_2(\mathsf{t}) &\geq_{\mathsf{D}^{\pm}_{\pi}} \llceil \pi \rrceil_{\mathcal{D}} \end{aligned}$$

Donc on conclut :

$$\ddot{v}(\mathsf{t}) \;=\; [\,\ddot{v}_1(\mathsf{t}), \ddot{v}_2(\mathsf{t})\,] \;\ddot{\leq}_{\pi}\; [\,\llfloor \pi \rrfloor_{\mathcal{D}}, \llceil \pi \rrceil_{\mathcal{D}}\,] \;=\; [\![\,\pi\,]\!]_{\mathcal{D}}$$

□

Le lemme précédent a une implication importante pour le calcul en temps fini de la sémantique bisémique d'une position π du jeu : s'il existe un préfixe *fini* de l'arbre de π dont la co-évaluation est un intervalle de dimension nulle, i.e. de la forme $[x, x]$, alors la sémantique bisémique coïncide avec cet intervalle.

COROLLARY 5.5.21. *Sous les hypothèses du lemme 5.5.20, nous avons le critère suivant :*

$$\frac{\forall \mathsf{t} \in \widehat{\mathsf{T}}(\mathcal{S}, \pi). \quad \ddot{v}(\mathsf{t}) = [x, x] \in \ddot{\mathsf{D}}_{\pi}}{[\![\,\pi\,]\!]_{\mathcal{D}} = [x, x]}$$

DÉMONSTRATION. *Par le lemme 5.5.20, si la sémantique bisémique existe, elle sera forcément supérieure à la valeur* $[x, x]$. *Mais ce dernier est un point maximal dans le préordre* $(\ddot{D}_\pi, \ddot{\lesssim}_\pi)$ *(rien de comparable ne pourrait être plus précis), donc* $[\![\, \pi \,]\!]_{\mathcal{D}} = [x, x]$. *Il reste donc à montrer que la sémantique bisémique existe. Cela est assuré par le critère de terminaison du lemme 5.4.6 appliqué aux deux composantes, la pessimiste et l'optimiste, de la fonction de co-évaluation* $\ddot{v}$. □

5.5.3. Indépendance vis-à-vis des heuristiques. La notion d'heuristique *plus informée* s'étend de façon naturelle aux cas des co-heuristiques.

DEFINITION 5.5.22. (CO-HEURISTIQUES PLUS INFORMÉES) *Soit* $\mathcal{S} = (\mathbb{P}, \lambda, \to, \mathbb{P}_{\mathrm{I}})$ *une arène de jeu et* $\mathcal{D} = (\mathsf{D}, \le_{\mathsf{D}}, \uparrow, \downarrow, \perp)$ *une structure qui est à la fois une structure de co-évaluation pour deux co-heuristiques différentes :*

$$\mathcal{D} \in \mathsf{Cstruct}(\mathcal{S}, \mathsf{D}, \uparrow, \downarrow, p, h_1, h_2)$$
$$\mathcal{D} \in \mathsf{Cstruct}(\mathcal{S}, \mathsf{D}, \uparrow, \downarrow, p, {h_1}', {h_2}')$$

Alors, si les co-heuristiques $\ddot{h} = (h_1, h_2)$ *et* $\ddot{h}' = ({h_1}', {h_2}')$ *sont dans l'ordre extensionnel des fonctions de type* $\mathbb{P}_{\mathrm{NT}} \to \ddot{\mathsf{D}}$, *nous disons que* $\ddot{h}'$ *est* plus informée *que* $\ddot{h}$, *noté* $\ddot{h} \le \ddot{h}'$. □

Autrement dit, une co-heuristique $\ddot{h}'$ est plus informée qu'une autre co-heuristique $\ddot{h}$, si $\forall \pi \in \mathbb{P}_{\mathrm{NT}}.\ \ddot{h}(\pi) \ddot{\lesssim}_\pi \ddot{h}'(\pi)$. Le rapport entre un couple de co-heuristiques dans l'ordre extensionnel et les projections correspondantes est évident : une co-heuristique $\ddot{h}'$ est plus informée qu'une autre $\ddot{h}$, si et seulement si la première projection de $\ddot{h}'$ est plus informée que la première projection de $\ddot{h}$ dans la structure pessimiste $\mathcal{D}$, et la seconde projection de $\ddot{h}'$ est plus informée que la seconde projection de $\ddot{h}$ dans la structure optimiste $\mathcal{D}^\perp$.

Étant donné une fonction de payoff, l'ordre *plus informé* entre co-heuristiques permet de définir un ordre de précision entre co-interprétations.

DEFINITION 5.5.23. (CO-INTERPRÉTATIONS PLUS PRÉCISES) *Soit* $\mathcal{S} = (\mathbb{P}, \lambda, \to, \mathbb{P}_{\mathrm{I}})$ *une arène de jeu, et soient* $\ddot{\varphi} : \mathsf{A}^p_{\mathcal{S}} \overset{\mathcal{H}}{\longmapsto} \ddot{\mathcal{D}}$ *et* $\ddot{\varphi}' : \mathsf{A}^p_{\mathcal{S}} \overset{\mathcal{H}}{\longmapsto} \ddot{\mathcal{D}}$, *où* $\ddot{\varphi} = (p, \ddot{h}, \ddot{v}, \ddot{w})$ *et* $\ddot{\varphi}' = (p', \ddot{h}', \ddot{v}', \ddot{w}')$, *deux interprétations du jeu dans la même algèbre de co-évaluation* $\ddot{\mathcal{D}}$. *Nous disons que* $\ddot{\varphi}'$ *est* plus précise *(ou* meilleure*) que* $\ddot{\varphi}$, *noté* $\ddot{\varphi} \ddot{\le} \ddot{\varphi}'$, *si les deux fonctions de payoff coïncident et la co-heuristique* $\ddot{h}'$ *est plus informée que* $\ddot{h}$ *:*

$$\ddot{\varphi} \ddot{\le} \ddot{\varphi}' \quad \overset{\triangle}{\Longleftrightarrow} \quad p = p' \ \wedge\ \ddot{h} \ddot{\le} \ddot{h}'$$

□

L'idée d'une plus grande précision, évoquée par la définition précédente, s'explique par les implications d'une telle condition sur les fonctions sémantiques. Puisque par définition de co-heuristiques plus informées, nous supposons à la fois $\mathcal{D} \in \mathsf{Cstruct}(\mathcal{S}, \mathsf{D}, \uparrow, \downarrow, p, h_1, h_2)$ et $\mathcal{D} \in \mathsf{Cstruct}(\mathcal{S}, \mathsf{D}, \uparrow, \downarrow, p, {h_1}', {h_2}')$, nous distinguerons les co-interprétations d'un jeu dans $\mathcal{D}$, donc les fonctions sémantiques impliquées, en remplaçant, dans la notation des fonctions sémantiques, l'indice $\mathcal{D}$ par la co-interprétation impliquée.

LEMMA 5.5.24. *Soit $\mathcal{S} = (\mathbb{P}, \lambda, \rightarrow, \mathbb{P}_{\mathrm{I}})$ une arène de jeu, et $\ddot{\varphi}, \ddot{\varphi}'$ deux co-interprétations du jeu dans $\ddot{\mathcal{D}}$ telles que $\ddot{\varphi} \mathrel{\ddot{\leq}} \ddot{\varphi}'$. Alors, pour toute position $\pi \in \mathbb{P}$, nous avons les relations suivantes entres les fonctions sémantiques correspondant aux deux co-interprétations :*

	$[\![\, \pi \,]\!]^{\natural}_{\ddot{\varphi}} \ \ddot{\leq}^{\natural}_{\pi} \ [\![\, \pi \,]\!]^{\natural}_{\ddot{\varphi}'}$,	bisémiques bécarre
	$\lfloor\!\lfloor\, \pi \,\rfloor\!\rfloor^{\flat}_{\ddot{\varphi}} \ \leq^{\flat}_{\pi} \ \lfloor\!\lfloor\, \pi \,\rfloor\!\rfloor^{\flat}_{\ddot{\varphi}'}$,	pessimistes bémole
	$\lceil\!\lceil\, \pi \,\rceil\!\rceil^{\sharp}_{\ddot{\varphi}} \ \geq^{\sharp}_{\pi} \ \lceil\!\lceil\, \pi \,\rceil\!\rceil^{\sharp}_{\ddot{\varphi}'}$,	optimistes dièse
(si elles existent)	$[\![\, \pi \,]\!]_{\ddot{\varphi}} \ \ddot{\leq}_{\pi} \ [\![\, \pi \,]\!]_{\ddot{\varphi}'}$,	bisémiques
(si elles existent)	$\lfloor\!\lfloor\, \pi \,\rfloor\!\rfloor_{\ddot{\varphi}} \ \leq_{\mathrm{D}^{\pm}_{\pi}} \ \lfloor\!\lfloor\, \pi \,\rfloor\!\rfloor_{\ddot{\varphi}'}$,	pessimistes
(si elles existent)	$\lceil\!\lceil\, \pi \,\rceil\!\rceil_{\ddot{\varphi}} \ \geq_{\mathrm{D}^{\pm}_{\pi}} \ \lceil\!\lceil\, \pi \,\rceil\!\rceil_{\ddot{\varphi}'}$,	optimistes

DÉMONSTRATION. *Conséquences directes de la monotonie des fonctions polymorphes des joueurs. Les preuves pour les arbres finis se font par induction en utilisant, pour les cas de base, l'hypothèse $p = p'$ et $\ddot{h} \mathrel{\ddot{\leq}} \ddot{h}'$.* □

La principale conséquence du lemme précédent est un critère de terminaison pour le calcul des sémantiques bisémiques. Dans un ordre partiel $(\mathrm{D}, \leq_{\mathrm{D}})$ pointé vers le bas et vers le haut, si nous obtenons une valeur maximale $[x, x]$ avec les heuristiques banales $\perp$ et $\top$, nous obtiendrons forcément le même résultat avec toute autre co-heuristique.

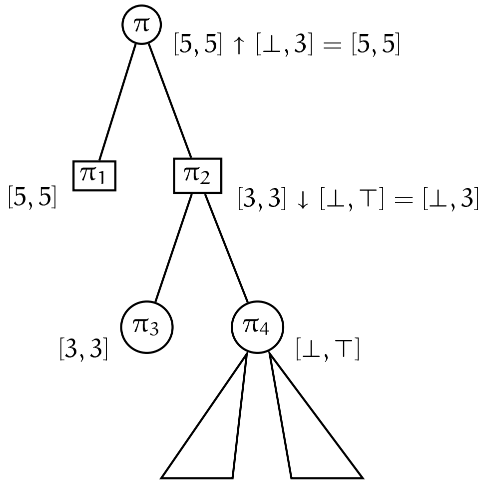
COROLLARY 5.5.25. *Soit $\mathcal{S} = (\mathbb{P}, \lambda, \rightarrow, \mathbb{P}_{\mathrm{I}})$ une arène de jeu et $\mathcal{D} = (\mathrm{D}, \leq_{\mathrm{D}}, \uparrow, \downarrow, \perp)$ une structure de co-évaluation telle que $(\mathrm{D}, \leq_{\mathrm{D}})$ est un ordre partiel avec un minimum $\perp$ et un maximum $\top$. Soit $\ddot{\varphi} = (p, \ddot{h}, \ddot{v}, \ddot{w})$ une co-interprétation bâtie sur la co-heuristique banale $\ddot{h} = \lambda\pi.(\perp, \top)$, et soit $\ddot{\varphi}' = (p, \ddot{h}', \ddot{v}', \ddot{w}')$ une seconde co-interprétation bâtie sur la même fonction de payoff. Nous avons alors l'implication suivante :*

$$\frac{[\![\, \pi \,]\!]_{\ddot{\varphi}} = [x, x]}{[\![\, \pi \,]\!]_{\ddot{\varphi}'} = [x, x]}$$

DÉMONSTRATION. *Pour le lemme 5.5.24 nous avons $[\![\, \pi \,]\!]_{\ddot{\varphi}} \mathrel{\ddot{\leq}} [\![\, \pi \,]\!]_{\ddot{\varphi}'}$. D'autre part, le point $[x, x]$ est maximal (le plus précis possible) dans $\ddot{\mathrm{D}}$, donc $[\![\, \pi \,]\!]_{\ddot{\varphi}'} \mathrel{\ddot{\leq}} [x, x] = [\![\, \pi \,]\!]_{\ddot{\varphi}'}$. L'énoncé est ainsi conséquence de la propriété anti-symétrique de $(\ddot{\mathrm{D}}, \ddot{\leq})$, propriété qui est héritée de l'ordre partiel $(\mathrm{D}, \leq)$.* □

Ainsi, lorsque la sémantique bisémique $[\![\, \pi \,]\!]_{\ddot{\varphi}}$ d'une position du jeu, calculée avec la co-heuristique banale $\ddot{h} = \lambda\pi.[\perp, \top]$, sera égale à une valeur maximale $[x, x]$, nous dirons que x est la *sémantique absolue* de la position π. Le qualificatif *absolue* évoquera l'indépendance d'une telle valeur par rapport aux approximations heuristiques utilisées.

FIG. 5.5.4. Sémantique absolue d'une position



5.5.3.1. *Exemple de sémantique absolue.* Considérons à nouveau l'exemple illustré par la figure 5.4.1, page 148. Supposons que dans la position π_2 l'opposant ait deux seuls coups possibles, le premier conduisant à la position terminale π_3, et le second conduisant à la position non terminale π_4. Supposons qu'on évalue la position initiale en développant son arbre jusqu'au niveau des positions π_3 et π_4. En d'autres termes, considérons le préfixe régulier $t_\pi/^2$ de l'arbre de la position π. Pour la position non terminale π_4 nous utiliserons la co-heuristique banale, pour les positions terminales π_1 et π_3 nous utiliserons le co-payoff dérivé de la même fonction de payoff de la figure 5.4.1. Le résultat de la rétro-propagation des valeurs est représenté dans la figure 5.5.4. Dans la position π_2, l'opposant minimise chacune des composantes de l'intervalle. Il obtient ainsi la valeur $[\perp, 3]$. Intuitivement, cela signifie que, dans le pire des cas pour lui, il sera obligé de concéder le payoff 3 (dans le meilleur $\perp$). Mais de toute façon, maximisant son gain dans la position π, le joueur déterminera que la valeur du préfixe $t_\pi/^2$ est $[5, 5]$. Ce dernier étant un intervalle maximal, il s'agit bien de l'unique valeur sémantique de la position π (cf. corollaire 5.5.21).

5.5.4. Classification sémantique des positions. Les résultats énoncés précédemment peuvent être résumés par une taxinomie des types de position d'un jeu donné. Soit $\mathcal{S} = (\mathbb{P}, \lambda, \rightarrow, \mathbb{P}_I)$ une arène de jeu, et soit $\ddot{\varphi} = (p, \ddot{h}, \ddot{v}, \ddot{w})$ une co-interprétation à valeur dans l'algèbre de co-évaluation $\ddot{\mathcal{D}} = (D, D^2, \ddot{D}, D^\flat, D^\sharp, \ddot{D}^\natural ; \leq, \uparrow, \downarrow, \perp, \top, \ddot{\perp})$. Une position π du jeu peut être :

(1) certainement *co-évaluable en* $\ddot{D}^\natural$,

la fonction bisémique bécarre $[\![\pi]\!]^\natural_{\ddot{\varphi}}$ étant totale ;

(2) éventuellement *évaluable en* $\ddot{D}_1$,

si son interprétation pessimiste $\lfloor\!\lfloor \pi \rfloor\!\rfloor_{\ddot{\varphi}}$ existe ;

(3) éventuellement *évaluable en* $\ddot{D}_2$

si son interprétation optimiste $\lceil\!\lceil \pi \rceil\!\rceil_{\ddot{\varphi}}$ existe ;

FIG. 5.5.5. Taxinomie des positions

$$\begin{tikzcd}
& \textit{strictement co-évaluable en } \mathsf{D} \arrow[d] & \\
& \textit{co-évaluable en } \mathsf{D} \arrow[d] & \\
& \textit{co-évaluable en } \ddot{\mathsf{D}} \arrow[dl] \arrow[dr] & \\
\textit{évaluable en } \ddot{\mathsf{D}}_1 \arrow[dr] & & \textit{évaluable en } \ddot{\mathsf{D}}_2 \arrow[dl] \\
& \textit{co-évaluable en } \ddot{\mathsf{D}}^{\natural} &
\end{tikzcd}$$

(4) éventuellement *co-évaluable en* $\ddot{\mathsf{D}}$

si ses deux interprétations, pessimiste et optimiste, existent ; autrement dit, si l'interprétation bisémique $[\![\, \pi \,]\!]_{\ddot{\varphi}}$ existe ;

(5) éventuellement *co-évaluable en* D

si ses deux interprétations, pessimiste et optimiste, existent et coïncident ; autrement dit, si $[\![\, \pi \,]\!]_{\ddot{\varphi}} = [x, x]$;

(6) éventuellement *strictement co-évaluable en* D

si $(\mathsf{D}, \leq)$ est un ordre partiel avec un minimum $\perp$ et un maximum $\top$, et si ses deux interprétations, pessimiste et optimiste, existent et coïncident même lorsqu'on utilise les heuristiques banales ; autrement dit, si la position a une sémantique absolue :
$[\![\, \pi \,]\!]_{\ddot{\varphi}'} = [x, x]$ où $\ddot{\varphi}' = (p, \ddot{h}', \ddot{v}', \ddot{w}')$, $\ddot{h}' = \lambda\pi.[\perp, \top]$.

Dans le treillis des propriétés de (1) à (6), où l'ordre est l'implication logique, la propriété (1) fait figure de propriété minimum (toujours vraie), et la (6) de propriété maximum. La totalité des implications logiques est représentée dans la figure 5.5.5.

Un second type de classification concerne l'existence d'un préfixe *fini* dont l'évaluation coïncide avec celle de l'arbre de la position dans son intégralité. Selon l'approche, pessimiste ou optimiste, nous avons une notion de finitude différente : si la position est finiment évaluable, au sens des sémantiques monosémiques (déf. 5.4.7), à la fois pour la structure pessimiste et pour l'optimiste, nous pouvons dire que la position est *finiment co-évaluable*. Les deux critères de classification sont orthogonaux. Par exemple, une position co-évaluable en D, n'est pas forcément finiment co-évaluable.

EXAMPLE 5.5.26. Soit $\mathcal{S} = (\mathbb{P}, \lambda, \rightarrow, \mathbb{P}_{\text{I}})$ un jeu où l'ensemble des positions est celui des entiers naturels $\mathbb{P} = \mathbb{N}$, la relation de transition est la relation successeur $(\pi \rightarrow \pi' \;\Leftrightarrow\; \pi' = \pi + 1)$, la fonction de classification des positions affecte les positions paires au joueur et les impaires à l'opposant, et $\mathbb{P}_{\mathrm{I}} = \{1\}$. Supposons d'évaluer le jeu dans la structure des nombres rationnels $\mathbb{Q}$ avec les heuristiques suivantes :

$$\ddot{h}_1(\pi) = -1/\pi \qquad\qquad \ddot{h}_2(\pi) = +1/\pi$$

L'arbre de jeu étant linéaire, les deux évaluations sémantiques convergent, pour π tendant à $+\infty$, à la valeur 0. Autrement dit, la position initiale $\pi_0 = 1$ est *co-évaluable en* D. Mais elle ne l'est pas finiment, puisque la limite 0 peut être seulement approchée mais jamais atteinte par la co-évaluation des préfixes finis. □

5.6. Sémantique des stratégies

Dans l'exemple de la figure 5.5.4, la stratégie consistant à choisir le coup conduisant à la position π_1 est une stratégie *optimale* pour le joueur. Les stratégies étant une façon d'affaiblir le joueur (on lui impose un choix unique à chacune de ses positions), la valeur de toute stratégie sera forcément inférieure à la valeur de la position. Ainsi, quand leur évaluation coïncidera avec celle de la position initiale, nous dirons que la stratégie est optimale. Nous allons, au cours de cette section, développer et formaliser les concepts liés à l'évaluation des stratégies des joueurs.

5.6.1. Structures d'évaluation régulières. La monotonie (dans toutes les composantes et par rapport aux valeurs des positions $\mathsf{D}_\pi^{p,h}$) est l'unique condition demandée aux structures d'évaluation pour définir une sémantique des positions du jeu. Pour interpréter les stratégies, qui sont des élagages particuliers du joueur (cf. section 4.4.1), nous demanderons que la structure d'évaluation $\mathcal{D} = (\mathsf{D}, \leq_{\mathsf{D}}, \uparrow, \downarrow)$ vérifie la même propriété de monotonie, mais par rapport aux valeurs des élagages finis du joueur $\widetilde{\mathsf{D}}_\pi^{p,h}$ (cf. définition 5.3.3), et qu'elle vérifie des propriétés additionnelles.

Dans les structures d'évaluation habituelles les opérations des joueurs sont le max et le min. Tous deux sont des fonctions qui n'ont aucun effet lorsqu'on les applique à un seul argument ($\max x = x = \min x$) ou à plusieurs répliques du même argument ($\max x..x = x = \min x..x$). Puisqu'elles nécessitent au moins deux arguments pour produire un effet, c'est-à-dire pour opérer une véritable transformation de leurs arguments, nous dirons qu'il s'agit de fonctions *plurielles.* Dans tout treillis d'évaluation $\mathcal{D} = (\mathsf{D}, \leq_{\mathsf{D}}, \sup, \inf)$, les fonctions sup et inf sont plurielles.

Definition 5.6.1. (Structures d'évaluation régulières) *Une structure d'évaluation monotone* $\mathcal{D} = (\mathsf{D}, \leq_{\mathsf{D}}, \uparrow, \downarrow)$, $\mathcal{D} \in \mathsf{Estruct}(\mathcal{S}, \mathsf{D}, \uparrow, \downarrow, p, h)$ *est* régulière *si :*

(1) *pour toute position* $\pi \in \mathbb{P}$ *du jeu* $(\widetilde{\mathsf{D}}_\pi, \leq_{\widetilde{\mathsf{D}}_\pi})$ *est un préordre*

(2) $\uparrow$ *et* $\downarrow$ *sont monotones dans toutes leurs composantes par rapport au valeurs des élagages, i.e. pour toute position non terminale* $\pi \in \mathbb{P}_{\mathrm{NT}}$*, où* $\mathrm{Succ}(\pi) = \pi_1..\pi_n$*, les fonctions vérifient la propriété :*

$$\forall u \in \widetilde{D}_{\pi_1}..\widetilde{D}_{\pi_{k-1}}, v \in \widetilde{D}_{\pi_{k+1}}..\widetilde{D}_{\pi_n}.$$

$$x \leq_{\widetilde{D}_{\pi_k}} y \Rightarrow \begin{cases} \uparrow (u.x.v) \leq_{\widetilde{D}_\pi} \uparrow (u.y.v) \\ \\ \downarrow (u.x.v) \leq_{\widetilde{D}_\pi} \downarrow (u.y.v) \end{cases}$$

(3) *l'heuristique est* fortement correcte*, c'est-à-dire, pour toute position* $\pi \in \mathbb{P}_{\text{NT}}$*, telle que* $\text{Succ}(\pi) = \pi_1..\pi_n$ *(*$n \geq 1$*), la fonction vérifie les conditions suivantes :*

$$\lambda(\pi) = \text{Player} \quad \Rightarrow \quad \forall \pi_i.\ \ h(\pi) \leq_{\widetilde{D}_\pi} p \oplus h(\pi_i)$$

$$\lambda(\pi) = \text{Opponent} \quad \Rightarrow \quad h(\pi) \leq_{\widetilde{D}_\pi} \left(\underset{i=1..n}{\downarrow} p \oplus h(\pi_i) \right)$$

(4) *la fonction du joueur est une fonction* plurielle *par rapport au valeurs des élagages, i.e. :*

$$\forall x \in \widetilde{D}_\pi. \quad x = \uparrow x..x$$

La famille des algèbres d'évaluation régulières sur un jeu $\mathcal{S}$*, une structure élémentaire* $(D, \uparrow, \downarrow)$*, un payoff* p *et une heuristiques* h*, sera notée* $\mathsf{REstruct}(\mathcal{S}, D, \uparrow, \downarrow, p, h)$. □

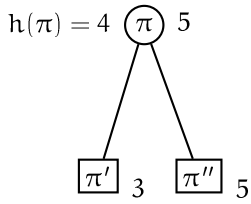
Dans le cadre des structures d'évaluation régulières, la correction de l'heuristique utilisée est une condition qui est renforcée. La nouvelle condition, qu'on appelle *correction forte*, permet de donner une sémantique aux élagages du joueur, donc aux stratégies. La condition sur les positions de l'opposant est identique à celle de correction simple. En revanche, la condition sur les positions du joueur demande que l'heuristique appliquée à la position considérée soit inférieure à l'heuristique (ou le payoff) appliquée individuellement aux positions suivantes dans le jeu.

On remarquera que l'heuristique canonique est à la fois correcte et fortement correcte. Si en général les deux notions de correction sont orthogonales, dans le cadre spécifique d'une structure d'évaluation régulière, la condition de correction est impliquée par celle de correction forte. Ainsi, pour prouver qu'une structure d'évaluation est monotone et régulière, on demontrera seulement la propriété de correction forte.

PROPOSITION 5.6.2. *Si la structure d'évaluation est régulière, alors toute heuristique fortement correcte est une heuristique correcte.*

DÉMONSTRATION. *Soit* $\mathcal{D} = (D, \leq_D, \uparrow, \downarrow)$ *une structure d'évaluation régulière, et soit* $\pi \in \mathbb{P}$*. Il suffit de montrer que* $h(\pi) \leq_{\widetilde{D}_\pi} \overset{n}{\underset{i=1}{\uparrow}} p \oplus h(\pi_i)$ *où* $\text{Succ}(\pi) = \pi_1..\pi_n$.

FIG. 5.6.1. Heuristique correcte $\not\Rightarrow$ fortement correcte



$$
\begin{array}{rcll}
h(\pi) & = & \overset{n}{\underset{i=1}{\uparrow}}\ h(\pi)..h(\pi) & (\uparrow \textit{ plurielle}) \\
 & \leq_{\widetilde{D}_\pi} & \overset{n}{\underset{i=1}{\uparrow}}\ p \oplus h(\pi_i) & (\uparrow \textit{ monotone}, \ h \textit{ fortement correcte})
\end{array}
$$

□

La réciproque n'est pas vraie. Dans un jeu min-max sur les entiers naturels (ce qui constitue un treillis d'évaluation), supposons qu'on ait une position π avec seulement deux coups conduisant à des positions terminales π' et π'', dont le payoff est, respectivement, l'entier 3 et l'entier 5. La situation est illustrée dans la figure 5.6.1. Une heuristique h estimant à 4 la valeur de π est correcte mais pas fortement correcte, puisque $\pi \to \pi'$ alors que $h(\pi) = 4 \not\leq_{\mathbb{N}} 3 = p(\pi')$.

5.6.2. Interprétation des élagages du joueur. Nous allons construire l'interprétation des élagages des joueurs, donc des stratégies et des contre-stratégies, par la même technique que celle utilisée pour l'interprétation des préfixes finis. En premier lieu nous allons montrer que la fonction v est *monotone*, sous certaines hypothèses, lorsque elle est appelée à l'évaluation des élagages *finis* (*compact*) du joueur. Ensuite, nous pourrons la vouer à l'évaluation de tous les arbres de $\widetilde{\mathrm{IT}}_p(\mathcal{S})$, compacts ou non. L'interprétation des élagages de l'opposant, en particulier des contre-stratégies, sera définie par dualité, les élagages de l'opposant étant des élagages du joueur dans la syntaxe de jeu duale. Une condition *sine qua non* pour une telle interprétation est que le jeu soit inévitablement alterné.

LEMMA 5.6.3. *Dans un jeu $\mathcal{S} = (\mathbb{P}, \lambda, \to, \mathbb{P}_I)$ inévitablement alterné, la fonction* $\mathrm{small}_{(.)}$, $\mathrm{small}_\pi \triangleq \min_{\subseteq}\{t \in \widetilde{\mathrm{IT}}_p(\mathcal{S}) \mid t \neq \emptyset \ \wedge \ t \subseteq t_\pi\}$ *est une fonction totale* $\mathbb{P} \to \widetilde{\mathrm{IT}}_p(\mathcal{S})$.

DÉMONSTRATION. *Soit $\pi \in \mathbb{P}$. Si $\lambda(\pi) =$* Player *alors $\{(\varepsilon, \pi)\}$ est un élagage du joueur pour la position π et il s'agit forcément du plus petit non vide, donc* $\mathrm{small}_\pi = \{(\varepsilon, \pi)\}$. *Si $\lambda(\pi) =$* Opponent, *puisque le jeu $\mathcal{S}$ est inévitablement alterné, il n'existe pas de séquences infinies de positions du même joueur :*

$$
\exists \delta_\pi \in \mathbb{N}. \ \left\{ \begin{array}{l} \exists\, \pi \xrightarrow{u}{}^{*} \pi' \\ |u| \geq \delta_\pi \end{array} \right. \quad \Rightarrow \quad \pi' \in \mathbb{P}_T \ \vee \left\{ \begin{array}{l} \exists\, \pi \xrightarrow{u'}{}^{*} \pi'' \xrightarrow{u''}{}^{*} \pi' \\ u = u'.u'' \\ \lambda(\pi) \neq \lambda(\pi'') \end{array} \right.
$$

Nous prouvons l'énoncé par induction sur δ_π.

- $\delta_\pi = 1$
  Dans ce cas la condition ci-dessus se simplifie et devient $\forall \pi \xrightarrow{\ell_i} \pi_i.\ \lambda(\pi_i) = \text{Player}$. *L'arbre* $t = \{(\varepsilon, \pi), (\ell_1, \pi_1), .., (\ell_n, \pi_n)\}$ *est un élagage du joueur. Puisque tout autre élagage* t' *non vide du joueur devra contenir* (ε, π) *et devra être complet à l'adresse* ε *de l'opposant, l'arbre* t *sera forcément contenu dans* t'. *Autrement dit* $\text{small}_\pi = t$.
- $\delta_\pi = k + 1$
  Pour tout $\pi_i \in \text{Succ}(\pi)$ *nous avons* $\delta_{\pi_i} \leq n$. *Par hypothèse d'induction,* small_{π_i} *existe. Nous définissons alors l'arbre :*

$$t = \{(\varepsilon, \pi)\} \cup \bigcup_{\pi \xrightarrow{\ell_i} \pi_i} \ell_i.\text{small}_{\pi_i}$$

  L'arbre t *est complet à l'adresse* ε *de l'opposant et tous ses fils sont des élagages du joueur. Donc* $t \in \widetilde{\text{IT}}_p(\mathcal{S})$. *Puisque tout autre élagage* t' *non vide du joueur devra être complet à la racine, les fils de* t' *seront, par définition de* small, *contenus dans les fils de* t *correspondants. Autrement dit,* $\text{small}_\pi = t$.

□

Nous utiliserons l'existence d'un plus petit élagage non vide du joueur, pour prouver la monotonie de la fonction v restreinte au domaine des élagages du joueur $\widetilde{\text{IT}}_p(\mathcal{S})$ équipé de l'ordre d'élagage. Un autre résultat préliminaire est le suivant.

LEMMA 5.6.4. *Si* $\mathcal{D} = (D, \leq_D, \uparrow, \downarrow)$ *est une structure d'évaluation régulière* $\mathcal{D} \in \mathsf{REstruct}(\mathcal{S}, D, \uparrow, \downarrow, p, h)$, *pour un jeu* $\mathcal{S} = (\mathbb{P}, \lambda, \to, \mathbb{P}_I)$ *inévitablement alterné, alors :*

$$\forall \pi \in \mathbb{P}.\quad p \oplus h(\pi) \ \leq_{\widetilde{D}_\pi}\ v_{p,h}(\text{small}_\pi)$$

DÉMONSTRATION. *Soit* $\pi \in \mathbb{P}$. *Par induction sur la hauteur* h *de l'arbre fini* small_π.

Si $\lambda(\pi) = \text{Player}$ *alors* $\{(\varepsilon, \pi)\}$ *est un élagage du joueur pour la position* π *et il s'agit forcément du plus petit non vide, donc* $\text{small}_\pi = \{(\varepsilon, \pi)\}$. *Si* $\lambda(\pi) = \text{Opponent}$,

- $h = 1$
  Dans ce cas $\text{small}_\pi = \{(\varepsilon, \pi)\}$ *donc* $p \oplus h(\pi) = v(\text{small}_\pi)$
- $h = k + 1$
  Si small_π *a une hauteur supérieure à* 1, *la position* π *est forcément de l'opposant et non terminale. Soit* $\text{Succ}(\pi) = \pi_1..\pi_n$ *Donc :*

$$\begin{array}{rcll} p \oplus h(\pi) & = & h(\pi) & (\pi \in \mathbb{P}_{NT}) \\ & \leq_{\widetilde{D}_\pi} & \underset{i=1..n}{\downarrow}\ p \oplus h(\pi_i) & (h \textit{ correcte}) \\ & \leq_{\widetilde{D}_\pi} & \underset{i=1..n}{\downarrow}\ v(\text{small}_{\pi_i}) & (\textit{induction},\ \downarrow\ \textit{monotone}) \\ & = & v(\text{small}_\pi) & (\textit{définition de } \text{small}) \end{array}$$

□

PROPOSITION 5.6.5. *Si* $\mathcal{D} = (\mathsf{D}, \leq_{\mathsf{D}}, \uparrow, \downarrow)$ *est une structure d'évaluation régulière* $\mathcal{D} \in \mathsf{REstruct}(\mathcal{S}, \mathsf{D}, \uparrow, \downarrow, \mathsf{p}, \mathsf{h})$*, pour un jeu* $\mathcal{S} = (\mathbb{P}, \lambda, \rightarrow, \mathbb{P}_{\mathrm{I}})$ *inévitablement alterné, alors la fonction* $v\colon \mathrm{T}(\mathbb{P}) \rightarrow \mathrm{D}$*, restreinte aux élagages finis du joueur, est monotone vis-à-vis de l'ordre d'élagage :*

$$v\,:\,(\widetilde{\mathrm{T}}_{\mathrm{p}}(\mathcal{S}), \subseteq) \xrightarrow{\mathcal{H}} (\mathsf{D}, \leq_{\mathsf{D}})$$

DÉMONSTRATION. *Comme dans la preuve de la proposition 5.3.12, nous comparons les arbres par leur hauteur. La relation :*

$$\prec \triangleq \{(\mathsf{t}', \mathsf{t}) \mid \mathrm{hauteur}(\mathsf{t}) <_{\mathbb{N}} \mathrm{hauteur}(\mathsf{t})\}$$

est bien fondée sur les élagage finis du joueur $\prec\colon \wp^{\widetilde{\mathrm{T}}_{\mathrm{p}}(\mathcal{S})\times\widetilde{\mathrm{T}}_{\mathrm{p}}(\mathcal{S})}$*, et son extension produit aux couples d'élagages finis* $\prec_2\colon \wp^{\widetilde{\mathrm{T}}_{\mathrm{p}}(\mathcal{S})^2\times\widetilde{\mathrm{T}}_{\mathrm{p}}(\mathcal{S})^2}$ *est, elle aussi, une relation bien fondée (cf. section 1.1.11). Nous pouvons alors prouver l'énoncé par le principe d'induction bien fondée sur* $\prec_2$*. Soient* $\mathsf{t}', \mathsf{t} \in \widetilde{\mathrm{T}}_{\mathrm{p}}(\mathcal{S})$ *tels que* $\mathsf{t}' \subseteq \mathsf{t}$*. La propriété à prouver est* $v(\mathsf{t}') \leq_{\widetilde{\mathrm{D}}_\pi} v(\mathsf{t})$*. Nous procédons par cas.*

(1) $\mathsf{t} = \emptyset$

Si t *est vide, alors* t' *doit l'être aussi (puisque* $\mathsf{t}' \subseteq \mathsf{t}$*), donc l'énoncé est vrai.*

(2) $\mathsf{t} = \{(\varepsilon, \pi)\}$

Si t *est une feuille, alors soit* $\mathsf{t}' = \emptyset$*, soit* $\mathsf{t}' = \mathsf{t}$ *; dans les deux cas* $v(\mathsf{t}') \leq_{\widetilde{\mathrm{D}}_\pi} v(\mathsf{t})$*.*

(3) $\mathrm{Fils}(\mathsf{t}) = \mathsf{t}_1..\mathsf{t}_n \quad \wedge \quad \lambda(\pi) = \mathrm{Player}$ *(où* $\pi = \mathsf{t}(\varepsilon)$ *et* $n \geq 1$*)*

Si t *a comme fils* $\mathsf{t}_1..\mathsf{t}_n$*, et la position initiale* π *est du joueur. Puisque* $n \geq 1$ *il s'agit d'une position non terminale. Considérons trois sous-cas possibles.*

(a) $\mathsf{t}' = \emptyset$

l'énoncé est, à nouveau, trivialement vrai.

(b) $\mathsf{t}' = \{(\varepsilon, \pi)\}$

Les deux arbres t, t' *sont des élagages (du joueur) d'un arbre de jeu* t_π*. Si l'élagage* t' *est radical à l'adresse* ε*, donc* $v(\mathsf{t}') = \mathsf{h}(\pi)$*, l'élagage* t *est, à la même adresse, partiel ou, éventuellement complet. Nous avons :*

$$\begin{array}{lll}
v(\mathsf{t}') = & \mathsf{h}(\pi) & (\pi \in \mathbb{P}_{\mathrm{NT}}) \\
\leq_{\widetilde{\mathrm{D}}_\pi} & \uparrow \mathsf{h}(\pi) & (\uparrow\ \textit{extensive}) \\
\leq_{\widetilde{\mathrm{D}}_\pi} & \underset{i=1..n}{\uparrow}\ \mathsf{h}(\pi) & (\uparrow\ \subseteq^{*}\textit{-croissante}) \\
\leq_{\widetilde{\mathrm{D}}_\pi} & \underset{i=1..n}{\uparrow}\ \mathsf{p} \oplus \mathsf{h}(\pi_i) & (\mathsf{h}\ \textit{fort. correcte},\ \uparrow\ \textit{monotone}) \\
\leq_{\widetilde{\mathrm{D}}_\pi} & \underset{i=1..n}{\uparrow}\ v(\mathrm{small}_{\pi_i}) & (\textit{lemme } 5.6.4) \\
= & v(\mathsf{t}'') &
\end{array}$$

où l'arbre t'' *est défini de la façon suivante :*

$$\mathsf{t}'' = \{(\varepsilon, \pi)\} \cup \bigcup_{(\ell_i, \pi_i) \in \mathsf{t}} \ell_i.\mathrm{small}_{\pi_i}$$

Il reste à montrer que $v(t'') \leq_{\widetilde{D}_\pi} v(t)$. *Puisque tous les fils de* t'' *sont plus petits, par définition, que les fils correspondants de* t, *par hypothèse d'induction* ($(t''_i, t_i) \prec_2 (t'', t)$), *nous avons* $v(t''_i) \leq_{\widetilde{D}_\pi} v(t_i)$. *Donc, par monotonie de* $\uparrow$, *nous avons l'énoncé.*

(4) $\mathrm{Fils}(t) = t_1..t_n \quad \wedge \quad \lambda(\pi) = \mathrm{Opponent} \quad$ (*où* $\pi = t(\varepsilon)$ *et* $n \geq 1$)
Dans ce dernier cas, soit t' *est vide, donc l'énoncé est vrai, soit* t' *est, lui aussi, complet à l'adresse* ε *de l'opposant. Nous avons alors que* $\mathrm{Fils}(t') = t'_1..t'_n$ *avec* $t'_i \subseteq t_i$, *À nouveau, par hypothèse d'induction* ($(t'_i, t_i) \prec_2 (t', t)$), *nous avons* $v(t'_i) \leq_{\widetilde{D}_\pi} v(t_i)$, *et nous pouvons conclure par monotonie de* $\uparrow$.

□

DEFINITION 5.6.6. *Étant donné une arène de jeu* $\mathcal{S} = (\mathbb{P}, \lambda, \rightarrow, \mathbb{P}_I)$ *et une structure d'évaluation régulière* $\mathcal{D} = (D, \leq_D, \uparrow, \downarrow)$, $\mathcal{D} \in \mathsf{REstruct}(\mathcal{S}, D, \uparrow, \downarrow, p, h)$, *l'*interprétation des élagages du joueur *est la fonction* $w_p : \widetilde{\mathrm{IT}}_p(\mathcal{S}) \rightarrow D^\flat$ *définie de la façon suivante :*

$$w_p(t) \triangleq [\{v(t') \mid t' \in T\}]_\flat \quad \textit{où } T \in \Delta(\widetilde{T}_p(\mathcal{S}), \subseteq),\ \sup_{(\widetilde{T}_p(\mathcal{S}), \subseteq)} T = t$$

□

L'interprétation d'un élagage du joueur t est donc la classe de co-finalité de n'importe quel dirigé de compacts approximant l'arbre t.

PROPOSITION 5.6.7. *La définition d'interprétation des élagages du joueur est correcte et il s'agit d'une fonction monotone :* $w_p : (\widetilde{\mathrm{IT}}_p(\mathcal{S}), \subseteq) \xrightarrow{\mathcal{H}} (D^\flat, \leq^\flat_D)$.

DÉMONSTRATION. *Nous voulons montrer que la définition de* $w_p(t)$ *ne dépend pas du représentant* T *choisi pour approximer l'arbre* t. *En effet, puisque* $(\widetilde{\mathrm{IT}}_p(\mathcal{S}), \subseteq)$ *est un dcpo, l'égalité des bornes supérieures implique la co-finalité. Donc tout autre représentant* T' *sera co-final de* T, *et son image* $v(T')$ *sera, elle aussi, co-finale de* $v(T)$, *la fonction* v *étant un homomorphisme. En ce qui concerne la monotonie de* w_p, *les dirigés de* $(\widetilde{T}_p(\mathcal{S}), \subseteq)$ *sont transformés par* v *en dirigés de* $(D, \leq_D : 1^{D \times D})$. *Supposons* $t' \subseteq t$ *où l'arbre* t *est représenté par le dirigé de compacts* T *et* t' *est représenté par* T'. *Puisque* v *conserve aussi la propriété de co-finalité, il suffit de prouver que* $T \subseteq^\flat T'$. T *et* T' *étant des dirigés de compacts dont les bornes supérieures sont dans la relation d'ordre* $\sup T \subseteq \sup T'$, *ils sont forcément dans le rapport de co-finalité correspondant* $T \subseteq^\flat T'$ *(cf. lemme 2.4.2).* □

Nous avons, à présent, tous les ingrédients nécessaires à l'interprétation de la sous-algèbre syntaxique *du joueur* (cf. section 4.5.2.2).

DEFINITION 5.6.8. (INTERPRÉTATION DES STRATÉGIES) *Étant donné un jeu* $\mathcal{S} = (\mathbb{P}, \lambda, \rightarrow, \mathbb{P}_I)$ *inévitablement alterné, dont la sous-algèbre syntaxique du joueur est*

$$(\mathbb{P}_{\mathrm{T}}, \mathbb{P}_{\mathrm{NT}}, \mathrm{T}(\mathbb{P}), \widetilde{\mathrm{T}}_{\mathrm{p}}(\mathcal{S}), \widetilde{\mathrm{IT}}_{\mathrm{p}}(\mathcal{S}), \mathrm{IS}_{\mathrm{p}}(\mathcal{S})\,;\, \subseteq_{\mathrm{p}} : \forall\alpha{:}\{\widetilde{\mathrm{T}}_{\mathrm{p}}(\mathcal{S}), \widetilde{\mathrm{IT}}_{\mathrm{p}}(\mathcal{S})\}.\, 1^{\alpha\times\alpha})$$

et étant donné une structure d'évaluation $\mathcal{D} = (\mathsf{D}, \leq_{\mathsf{D}}, \uparrow, \downarrow)$ *régulière* $\mathcal{D} \in \mathsf{REstruct}(\mathcal{S}, \mathsf{D}, \uparrow, \downarrow, \mathtt{p}, \mathtt{h})$, *et germe de l'algèbre d'évaluation :*

$$\mathcal{D}^\flat = (\mathsf{D}, \mathsf{D}^\flat\,;\, \leq : \forall\alpha{:}\mathsf{sort}.\, 1^{\alpha\times\alpha}, \uparrow, \downarrow),$$

l'interprétation des stratégies *est l'interprétation algébrique* (ψ, φ) *de l'algèbre syntaxique du joueur vers l'algèbre d'évaluation* $\mathcal{D}^\flat$, *définie de la façon suivante :*

(1) *l'interprétation des noms* ψ *est l'application suivante :*

(a) *pour les noms des domaines :*

$$\psi(\sigma) = \begin{cases} \mathsf{D} & \text{si} \quad \sigma \in \{\mathbb{P}_{\mathrm{T}}, \mathbb{P}_{\mathrm{NT}}, \mathrm{T}(\mathbb{P}), \widetilde{\mathrm{T}}_{\mathrm{p}}(\mathcal{S})\} \\ \mathsf{D}^\flat & \text{si} \quad \sigma \in \{\widetilde{\mathrm{IT}}_{\mathrm{p}}(\mathcal{S}), \mathrm{IS}_{\mathrm{p}}(\mathcal{S})\} \end{cases}$$

(b) *pour les noms des opérations :*

$$\psi(\subseteq_{\mathrm{p}}) \triangleq (\leq)$$

(2) *l'interprétation des valeurs est la construction polymorphe :*

$$\varphi : \forall\alpha{:}\{\mathbb{P}_{\mathrm{T}}, \mathbb{P}_{\mathrm{NT}}, \mathrm{T}(\mathbb{P}), \widetilde{\mathrm{T}}_{\mathrm{p}}(\mathcal{S}), \widetilde{\mathrm{IT}}_{\mathrm{p}}(\mathcal{S}), \mathrm{IS}_{\mathrm{p}}(\mathcal{S})\}.\, \alpha \to \psi(\alpha)$$

définie par : $\varphi_{\mathbb{P}_{\mathrm{T}}} \triangleq \mathtt{p}$, $\varphi_{\mathbb{P}_{\mathrm{NT}}} \triangleq \mathtt{h}$, $\varphi_{\mathrm{T}(\mathbb{P})} \triangleq v$, $\varphi_{\widetilde{\mathrm{T}}_{\mathrm{p}}(\mathcal{S})} \triangleq v_{|\widetilde{\mathrm{T}}_{\mathrm{p}}(\mathcal{S})}$, $\varphi_{\widetilde{\mathrm{IT}}_{\mathrm{p}}(\mathcal{S})} \triangleq w_{\mathrm{p}}$, $\varphi_{\mathrm{IS}_{\mathrm{p}}(\mathcal{S})} \triangleq w_{\mathrm{p}|\mathrm{IS}_{\mathrm{p}}(\mathcal{S})}$, *où :*

(a) *la fonction* $v : \mathrm{T}(\mathbb{P}) \to \mathsf{D}$ *est l'interprétation des arbres étiquetés dans* $\mathbb{P}$ *(avec* $\mathtt{p}$ *et* $\mathtt{h}$*)*

(b) *la fonction* $w_{\mathrm{p}} : \widetilde{\mathrm{IT}}_{\mathrm{p}}(\mathcal{S}) \to \mathsf{D}^\flat$ *est l'interprétation des élagages arbitraires du joueur (avec* $\mathtt{p}$, $\mathtt{h}$ *et* v*)*

□

Comme pour l'interprétation d'un jeu, nous noterons l'interprétation des stratégies par ses quatre composantes principales $\varphi = (\mathtt{p}, \mathtt{h}, v, w_{\mathrm{p}})$, faisant omission de l'association des noms.

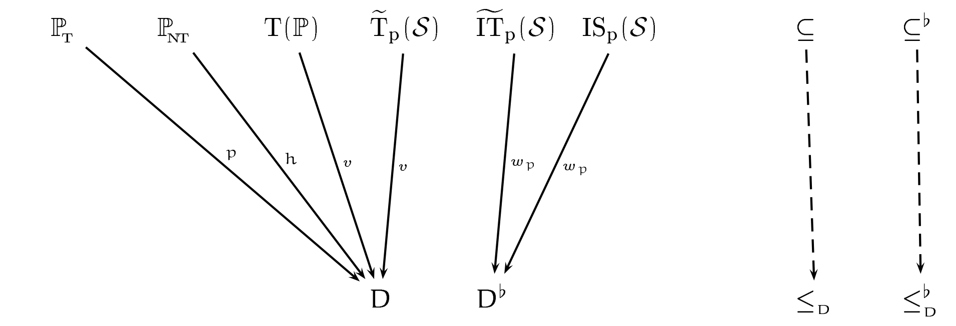
PROPOSITION 5.6.9. *La définition d'interprétation des stratégies d'un jeu est correcte et il s'agit d'un homomorphisme.*

DÉMONSTRATION. *La preuve est calquée sur la preuve analogue de correction de la définition d'interprétation d'un jeu.* □

L'homomorphisme entre la sous-algèbre syntaxique du joueur et l'algèbre d'évaluation régulière est résumé par la figure 5.6.2. Les contre-stratégies d'un jeu $\mathcal{S}$ coïncident avec les stratégies du joueur dans le jeu dual $\mathcal{S}^\perp$. Pour donner un sens aux contre-stratégies, il suffit donc d'interpréter les stratégies du jeu dual, à condition que la structure d'évaluation soit toujours régulière, même lorsqu'on échange le rôle des joueurs. Autrement dit, pour donner un sens au contre-stratégies finies ou infinies, il faudra que la structure d'évaluation $\mathcal{D} \in \mathsf{Estruct}(\mathcal{S}, \mathsf{D}, \uparrow, \downarrow, \mathtt{p}, \mathtt{h})$ soit telle que :

FIG. 5.6.2. interprétation d'un jeu

SOUS-ALGÈBRE SYNTAXIQUE DU JOUEUR



ALGÈBRE D'ÉVALUATION SÉMANTIQUE (RÉGULIÈRE)

$$\mathcal{D} \in \mathsf{REstruct}(\mathcal{S}^\perp, \mathsf{D}, \downarrow, \uparrow, \mathsf{p}, \mathsf{h})$$

Dans cette dernière écriture, nous remarquerons que l'ordre des opérations est échangé par rapport à la condition $\mathcal{D} \in \mathsf{Estruct}(\mathcal{S}, \mathsf{D}, \uparrow, \downarrow, \mathsf{p}, \mathsf{h})$. Cela permet d'associer aux joueurs leur ancienne opération.

DEFINITION 5.6.10. (INTERPRÉTATION DES CONTRE-STRATÉGIES) *Étant donné un jeu* $\mathcal{S} = (\mathbb{P}, \lambda, \rightarrow, \mathbb{P}_{I})$ *inévitablement alterné, dont la sous-algèbre syntaxique de l'opposant est :*

$$(\mathbb{P}_{T}, \mathbb{P}_{NT}, T(\mathbb{P}), \widetilde{T}_o(\mathcal{S}), \widetilde{IT}_o(\mathcal{S}), IS_o(\mathcal{S})\,;\, \subseteq_o : \forall\alpha{:}\{\widetilde{T}_p(\mathcal{S}), \widetilde{IT}_p(\mathcal{S})\}.\, 1^{\alpha\times\alpha})$$

et étant donné une structure d'évaluation $\mathcal{D} = (\mathsf{D}, \leq_{\mathsf{D}}, \uparrow, \downarrow)$*, régulière pour le jeu dual* $\mathcal{D} \in \mathsf{REstruct}(\mathcal{S}^\perp, \mathsf{D}, \downarrow, \uparrow, \mathsf{p}, \mathsf{h})$*, et dont la duale* $\mathcal{D}^\perp$ *est le germe de l'algèbre d'évaluation :*

$$\mathcal{D}^\sharp = (\mathsf{D}, \mathsf{D}^\sharp\,;\, \geq : \forall\alpha{:}\mathsf{sort}.\, 1^{\alpha\times\alpha}, \uparrow, \downarrow)$$

l'interprétation des contre-stratégies *est l'interprétation algébrique* (ψ, φ) *définie de la façon suivante :*

(1) *l'interprétation des noms* ψ *est l'application suivante :*

   (a) *pour les noms des domaines :*

$$\psi(\sigma) = \begin{cases} \mathsf{D} & \text{si} \quad \sigma \in \{\mathbb{P}_{T}, \mathbb{P}_{NT}, T(\mathbb{P}), \widetilde{T}_o(\mathcal{S})\} \\ \mathsf{D}^\sharp & \text{si} \quad \sigma \in \{\widetilde{IT}_o(\mathcal{S}), IS_o(\mathcal{S})\} \end{cases}$$

   (b) *pour les noms des opérations :*

$$\psi(\subseteq_o) \triangleq (\geq)$$

(2) *l'interprétation des valeurs est la construction polymorphe :*

$$\varphi\ :\ \forall\alpha{:}\{\mathbb{P}_{T}, \mathbb{P}_{NT}, T(\mathbb{P}), \widetilde{T}_o(\mathcal{S}), \widetilde{IT}_o(\mathcal{S}), IS_o(\mathcal{S})\}.\, \alpha \rightarrow \psi(\alpha)$$

définie par : $\varphi_{\mathbb{P}_T} \triangleq p$, $\varphi_{\mathbb{P}_{NT}} \triangleq h$, $\varphi_{T(\mathbb{P})} \triangleq v$, $\varphi_{\tilde{T}_o(\mathcal{S})} \triangleq v_{|\tilde{T}_o(\mathcal{S})}$, $\varphi_{\widetilde{IT}_o(\mathcal{S})} \triangleq w_o$, $\varphi_{IS_o(\mathcal{S})} \triangleq w_{o|IS_o(\mathcal{S})}$

où $\varphi' = (p, h, v, w'_p)$ *est l'interprétation des stratégies de* $\mathcal{S}^\perp$ *vers* $\mathcal{D}^\perp$ *et* $w_o = w'_p$.

□

PROPOSITION 5.6.11. *La définition d'interprétation des contre-stratégies d'un jeu est correcte et il s'agit d'un homomorphisme.*

DÉMONSTRATION. *La fonction d'interprétation* w_p *des élagages arbitraires de l'opposant est celle du joueur dans la syntaxe duale. Puisque* $\widetilde{IT}_o(\mathcal{S}) = \widetilde{IT}_p(\mathcal{S}^\perp)$, *les fonctions* $\varphi_{\widetilde{IT}_o(\mathcal{S})}$ *et* $\varphi_{IS_o(\mathcal{S})}$ *sont définies correctement. Puisque* $\varphi' : A^p_{\mathcal{S}^\perp} \xrightarrow{\mathcal{H}} \mathcal{D}^\perp$ *est un homomorphisme, la relation* $\subseteq_o$: $1^{\tilde{T}_o(\mathcal{S})\times\tilde{T}_o(\mathcal{S})}$ *est conservée dans* $\geq$: $1^{D\times D}$, *et la relation* $\subseteq^\flat_o$: $1^{\widetilde{IT}_o(\mathcal{S})\times\widetilde{IT}_o(\mathcal{S})}$ *est conservée dans* $\geq^\flat$: $1^{\Delta(D,\geq)_{\approx_\flat}\times\Delta(D,\geq)_{\approx_\flat}}$. *Puisque* $\Delta(D,\geq)_{\approx_\flat} = D^\sharp$ *(et* $\geq^\flat = \leq^\sharp$*) nous avons que* $\geq^\flat$: $1^{D^\sharp\times D^\sharp}$ *est de type compatible, via* ψ, *avec le type* $1^{\widetilde{IT}_o(\mathcal{S})\times\widetilde{IT}_o(\mathcal{S})}$ *de l'opération* $\subseteq_{\widetilde{IT}_o(\mathcal{S})}$. *L'interprétation algébrique est donc correcte par rapport aux types et hérite de* φ' *la conservation de l'opération polymorphe* $\subseteq_o$. □

Lorsque une structure d'évaluation $\mathcal{D} = (D, \leq_D, \uparrow, \downarrow)$ sera à la fois régulière pour un jeu $\mathcal{S}$ et régulière pour son dual (en échangeant les opérations des joueurs) :

$$\begin{cases} \mathcal{D} \in \mathsf{REstruct}(\mathcal{S}\ \ , D, \uparrow, \downarrow, p, h) \\ \mathcal{D} \in \mathsf{REstruct}(\mathcal{S}^\perp, D, \downarrow, \uparrow, p, h) \end{cases}$$

nous la qualifierons d'*ambivalente*. Nous avons vu que toute structure régulière ambivalente se prête aussi bien à l'interprétation des stratégies que à celle des contre-stratégies. Autrement dit, toute structure régulière ambivalente permet l'interprétation homomorphe de l'algèbre syntaxique *complète*.

5.6.3. Suite des stratégies approximantes. Tout préfixe fini $t \in \widehat{T}(\mathcal{S}, \pi)$ d'un arbre d'une position π, est une approximation de la valeur de l'arbre. Dans ce procédé d'approximation, on peut remplacer, sous certaines hypothèses, les préfixes finis par les stratégies du joueur. Supposons que les opérations des joueurs sont distributives, c'est-à-dire :

$$\uparrow u.(\downarrow y_1..y_n).v = \downarrow (\uparrow u.y_1.v)..(\uparrow u.y_n.v)$$
$$\downarrow u(\uparrow y_1..y_n).v = \uparrow (\downarrow u.y_1.v)..(\downarrow u.y_n.v)$$

Dans l'évaluation d'un arbre de jeu, la rétro-propagation des valeurs peut être considérée comme l'évaluation d'un terme où les fonctions des joueurs remplacent les positions non terminales et le payoff remplace les positions terminales. Considerons alors les propriétés distributives comme une relation de transformation entre termes équivalents. Si au départ les opérations des joueurs sont imbriquées dans ce terme, par l'application de la propriété distributive nous pourrions faire "remonter" progressivement tous les symboles ↑ du joueur, "repoussant" les symboles ↓ de l'opposant vers le bas du terme. Par une série de transformations de ce genre, nous pourrions finalement écrire un terme équivalent de profondeur 2, avec le symbole ↑

du joueur à la racine, des symboles $\downarrow$ au niveau 1, et des valeurs au niveau 2. La valeur de chaque branche de la racine correspondrait alors à la valeur d'une stratégie du joueur. Autrement dit, on aurait caractérisé la valeur de l'arbre de jeu comme le résultat de l'application de la fonction du joueur $\uparrow$ sur les valeurs de toutes les stratégies du même joueur.

Par la suite, nous supposerons aussi que les opérations soient associatives, i.e. :

$$\begin{aligned}\uparrow u.(\uparrow v).z &= \uparrow u.v.z \\ \downarrow u.(\downarrow v).z &= \downarrow u.v.z\end{aligned}$$

DEFINITION 5.6.12. (PRÉ-STRATÉGIES) *Étant donné un jeu $\mathcal{S}$, les* pré-stratégies *sont les préfixes finis des stratégies arbitraires du joueur :*

$$\widehat{S}(\mathcal{S},\pi) = \{\hat{s} \in T(\mathbb{P}) \mid \exists s \in IS_p(\mathcal{S},\pi).\ \hat{s} \leq s\}$$

Soit $t \in \widehat{IT}(\mathcal{S},\pi)$ *un préfixe du jeu. Les* pré-stratégies dans t *sont les pré-stratégies incluses dans l'arbre* t *:*

$$\widehat{S}(\mathcal{S},t) = \{\hat{s} \in \widehat{S}(\mathcal{S},\pi) \mid \pi = t(\varepsilon),\ \hat{s} \subseteq t\}$$

$\square$

La valeur d'un préfixe fini du jeu est égale, dans une structure d'évaluation distributive, au résultat de l'opération du joueur appliquée à toutes les pré-stratégies incluses dans le préfixe.

PROPOSITION 5.6.13. *Étant donné une structure d'évaluation $\mathcal{D} = (D, \leq_D, \uparrow, \downarrow)$, $\mathcal{D} \in \mathsf{Cstruct}(\mathcal{S}, D, \uparrow, \downarrow, p, h)$ pour un jeu $\mathcal{S}$, si les opérations des joueurs sont distributives, associatives et l'opération du joueur $\uparrow$ est plurielle, alors :*

$$v_{p,h}(t) = \mathop{\uparrow}_{\hat{s}\in\widehat{S}(\mathcal{S},t)} v_{p,h}(\hat{s})$$

DÉMONSTRATION. *Par induction sur la profondeur* k *de l'arbre* t.

– $k = 0$

Si $t = \{(\varepsilon,\pi)\}$ *alors la seule stratégie et pré-stratégie possible est l'arbre* t. *Donc :*

$$\begin{aligned} v_{p,h}(t) &= \uparrow v_{p,h}(t) && \uparrow \textit{ plurielle} \\ &= \mathop{\uparrow}_{s\in\widehat{S}(\mathcal{S},t)} v_{p,h}(s) && \widehat{S}(\mathcal{S},t) = \{t\}\end{aligned}$$

– $k \Rightarrow k+1$ *avec* $\lambda(\pi) = \text{Player}$

L'arbre t *a* $n \geq 1$ *fils situés aux adresses* $\ell_1, .., \ell_n$. *Une pré-stratégie dans* t *se construit par la greffe à l'arbre* $\{(\varepsilon,\pi)\}$, *où* $\pi = t(\varepsilon)$, *d'une quelconque pré-stratégie d'un fils* t_i *:*

$$\widehat{S}(\mathcal{S},t) = (\{(\varepsilon,\pi)\}\cdot_{\ell_1} \widehat{S}(\mathcal{S},t_1)) \cup .. \cup (\{(\varepsilon,\pi)\}\cdot_{\ell_n} \widehat{S}(\mathcal{S},t_n)) \qquad (i)$$

Alors :

$$\begin{array}{rcll} v_{p,h}(t) & = & \overset{n}{\underset{i=1}{\uparrow}} v_{p,h}(t_i) & \textit{def. de } v_{p,h} \\ & = & \overset{n}{\underset{i=1}{\uparrow}} \Big(\underset{s \in \widehat{S}(\mathcal{S},t_i)}{\uparrow} v_{p,h}(s) \Big) & \textit{Hp. d'induction} \\ & = & \underset{s \in U}{\uparrow} v_{p,h}(s) & \uparrow \textit{ associative, } U = \bigcup_{i=1..n} \widehat{S}(\mathcal{S},t_i) \\ & = & \underset{s' \in \widehat{S}(\mathcal{S},t)}{\uparrow} v_{p,h}(s') & \textit{def. de } U \textit{ et } (i) \end{array}$$

– $k \Rightarrow k+1$ *avec* $\lambda(\pi) = \text{Opponent}$

L'arbre t *a* $n \geq 1$ *fils situés aux adresses* $\ell_1, .., \ell_n$*. Une pré-stratégie dans* t *se construit par l'assemblage de* n *pré-stratégies, chacune relative à un fils de* t *:*

$$\widehat{S}(\mathcal{S}, t) = \{(\varepsilon, \pi)\} \cdot_{\ell_1} \widehat{S}(\mathcal{S}, t_1) \cdot ... \cdot_{\ell_n} \widehat{S}(\mathcal{S}, t_n) \qquad (i)$$

Alors :

$$\begin{array}{rcll} v_{p,h}(t) & = & \overset{n}{\underset{i=1}{\downarrow}} v_{p,h}(t_i) & \textit{def. de } v_{p,h} \\ & = & \overset{n}{\underset{i=1}{\downarrow}} \Big(\underset{s \in \widehat{S}(\mathcal{S},t_i)}{\uparrow} v_{p,h}(s) \Big) & \textit{Hp. d'induction} \\ & = & \underset{\substack{s_1 \in \widehat{S}(\mathcal{S},t_1) \\ \cdots \\ s_n \in \widehat{S}(\mathcal{S},t_n)}}{\uparrow} (v_{p,h}(s_1) \downarrow .. \downarrow v_{p,h}(s_n)) & \textit{distributive} \\ & = & \underset{s \in \widehat{S}(\mathcal{S},t)}{\uparrow} v_{p,h}(s) & (i) \end{array}$$

□

Ainsi, lorsque les opérations des joueurs sont associatives et distributives, et la structure d'évaluation est régulière, les pré-stratégies permettent d'approximer la valeur du jeu, autant que le font les préfixes.

COROLLARY 5.6.14. *Étant donné une structure d'évaluation* $\mathcal{D} = (D, \leq_D, \uparrow, \downarrow)$, $\mathcal{D} \in \mathsf{Cstruct}(\mathcal{S}, D, \uparrow, \downarrow, p, h)$ *pour un jeu* $\mathcal{S}$*, si les opérations des joueurs sont distributives, associatives et l'opération du joueur* $\uparrow$ *est plurielle, alors :*

$$(\!|\pi|\!)^{\flat}_{\mathcal{D}} = \sup\{ \underset{s \in \widehat{S}(\mathcal{S},\pi)_k}{\uparrow} v_{p,h}(s) \mid k \in \mathbb{N}\}$$

où $\widehat{S}(\mathcal{S}, \pi)_k$ *est l'ensemble des pré-stratégies de profondeur* k*.* □

Puisque les pré-stratégies, donc les stratégies, peuvent jouer le rôle d'approximant fini dans l'évaluation sémantique d'un jeu, il est intéressant d'observer que la retro-propagation min-max appliquée à une pré-stratégie donnée peut être calculée d'une manière bien spéciale : il suffit d'appliquer l'opération de l'opposant à l'évaluation (par la fonction de payoff ou par l'heuristique) de toutes les feuilles de la pré-stratégie.

PROPOSITION 5.6.15. *Étant donné une structure d'évaluation* $\mathcal{D} = (D, \leq_D, \uparrow, \downarrow)$, $\mathcal{D} \in \mathsf{Cstruct}(\mathcal{S}, D, \uparrow, \downarrow, p, h)$ *pour un jeu* $\mathcal{S}$*, si les opérations des joueurs sont associatives, les opérations* $\downarrow$ *et* $\uparrow$ *sont plurielles, et* $s \in \widehat{S}(\mathcal{S}, \pi)$*, alors :*

$$v_{p,h}(s) = \downarrow_{\pi' \in \mathrm{Feuilles}(s)} p \oplus h(\pi')$$

DÉMONSTRATION. *Par induction sur la profondeur* k *de la pré-stratégie* s.

- $k = 0$

  Si $s = \{(\varepsilon, \pi)\}$ *alors* $\mathrm{Feuilles}(s) = \{\pi\}$, *donc :*

  $$\begin{array}{rcll} v_{p,h}(s) & = & p \oplus h(\pi) & \textit{def. de } v_{p,h} \\ & = & \downarrow p \oplus h(\pi) & \downarrow \textit{ plurielle} \\ & = & \downarrow_{\pi' \in \mathrm{Feuilles}(s)} p \oplus h(\pi') & \end{array}$$

- $k \Rightarrow k+1$ *avec* $\lambda(\pi) = \mathrm{Player}$

  La pré-stratégie s *a un seul fils* s' *dont la valeur est, par hypothèse d'induction, égale à :*

  $$v_{p,h}(s') = \downarrow_{\pi' \in \mathrm{Feuilles}(s')} p \oplus h(\pi')$$

  Puisque $\uparrow$ *est plurielle et que* $\mathrm{Feuilles}(s) = \mathrm{Feuilles}(s')$ *nous avons l'énoncé :*

  $$v_{p,h}(s) = \uparrow v_{p,h}(s') = v_{p,h}(s') = \downarrow_{\pi' \in \mathrm{Feuilles}(s)} p \oplus h(\pi')$$

- $k \Rightarrow k+1$ *avec* $\lambda(\pi) = \mathrm{Opponent}$

  La pré-stratégie s *a autant de fils que l'opposant a de coup dans la position* π. *Par induction, tous les fils* s_i *de* s *sont des pré-stratégies telles que :*

  $$v_{p,h}(s_i) = \downarrow_{\pi' \in \mathrm{Feuilles}(s_i)} p \oplus h(\pi')$$

  Alors :

  $$\begin{array}{rcll} v_{p,h}(s) & = & \downarrow_{i=1}^{n} v_{p,h}(s_i) & \textit{def. de } v_{p,h} \\ & = & \downarrow_{i=1}^{n} \left(\downarrow_{\pi' \in \mathrm{Feuilles}(s_i)} p \oplus h(\pi') \right) & \textit{Hp. d'induction} \\ & = & \downarrow_{\pi' \in \mathrm{Feuilles}(s)} p \oplus h(\pi') & \textit{associative}, \downarrow \textit{ plurielle} \end{array}$$

□

5.7. Conclusion

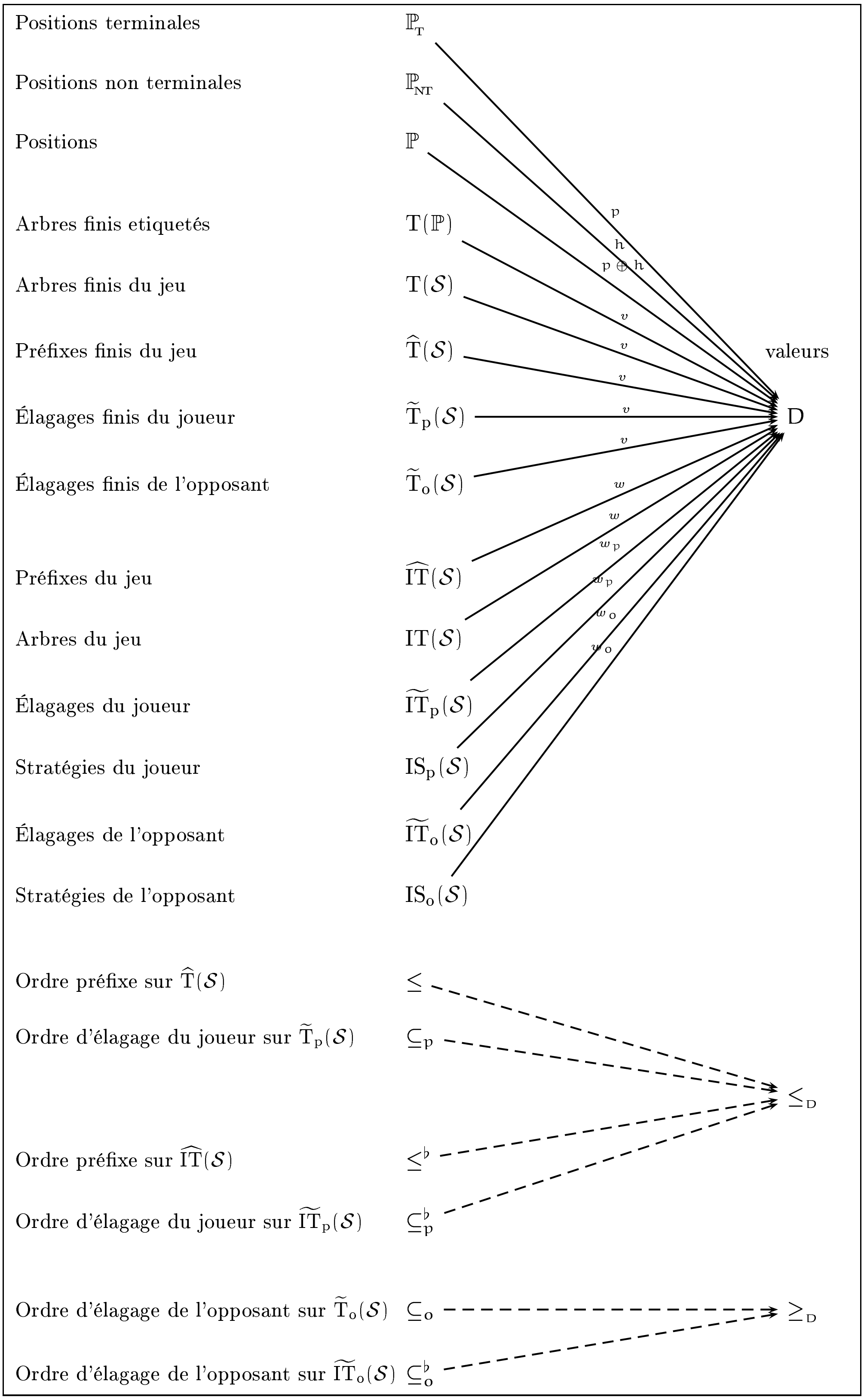
Nous avons construit progressivement un homomorphisme de l'algèbre syntaxique complète du jeu vers une algèbre de co-évaluation bâtie sur deux structures d'évaluation régulières. L'interprétation homomorphe est résumée dans la figure 5.7.1. Dans la figure 5.7.2 est représenté, en revanche, le cas d'une structure d'évaluation non seulement régulière, mais aussi telle que le préordre $(D, \leq_D)$ est, à la fois, un dcpo et un co-dcpo. Dans ce cas, tous les domaines d'arbres de la syntaxe complète peuvent être interprétés directement dans D (les extensions bémol et dièse étant isomorphes à la structure initiale $(D, \leq_D)$).

Lorsque la structure d'évaluation $\mathcal{D}$ et sa duale $\mathcal{D}^{\perp}$ ne seront pas régulières respectivement pour le jeu $\mathcal{S}$ et pour le dual $\mathcal{S}^{\perp}$, nous pourrons interpréter exclusivement l'algèbre syntaxique *principale* du jeu. Cela nous permettra de donner une sémantique (monosémique ou bisémique) à toutes les positions du jeu, mais nous resterons en défaut d'une fonction sémantique pour les stratégies des joueurs.

FIG. 5.7.1. Interprétation de l'algèbre syntaxique complète dans une structure d'évaluation régulière ambivalente

Positions terminales	$\mathbb{P}_{\mathrm{T}}$	$\xrightarrow{\mathrm{p}}$	D
Positions non terminales	$\mathbb{P}_{\mathrm{NT}}$	$\xrightarrow{\mathrm{h}}$	D
Positions	$\mathbb{P}$	$\xrightarrow{\mathrm{p} \oplus \mathrm{h}}$	D
			valeurs
Arbres finis etiquetés	$\mathrm{T}(\mathbb{P})$	$\xrightarrow{v}$	D
Arbres finis du jeu	$\mathrm{T}(\mathcal{S})$	$\xrightarrow{v}$	D
Préfixes finis du jeu	$\widehat{\mathrm{T}}(\mathcal{S})$	$\xrightarrow{v}$	D
Élagages finis du joueur	$\widetilde{\mathrm{T}}_{\mathrm{p}}(\mathcal{S})$	$\xrightarrow{v}$	D
Élagages finis de l'opposant	$\widetilde{\mathrm{T}}_{\mathrm{o}}(\mathcal{S})$	$\xrightarrow{v}$	D
			classes co-finales
Préfixes du jeu	$\widehat{\mathrm{IT}}(\mathcal{S})$	$\xrightarrow{w}$	$\mathsf{D}^\flat$
Arbres du jeu	$\mathrm{IT}(\mathcal{S})$	$\xrightarrow{w}$	$\mathsf{D}^\flat$
Élagages du joueur	$\widetilde{\mathrm{IT}}_{\mathrm{p}}(\mathcal{S})$	$\xrightarrow{w_{\mathrm{p}}}$	$\mathsf{D}^\flat$
Stratégies du joueur	$\mathrm{IS}_{\mathrm{p}}(\mathcal{S})$	$\xrightarrow{w_{\mathrm{p}}}$	$\mathsf{D}^\flat$
Élagages de l'opposant	$\widetilde{\mathrm{IT}}_{\mathrm{o}}(\mathcal{S})$	$\xrightarrow{w_{\mathrm{o}}}$	$\mathsf{D}^\sharp$
Stratégies de l'opposant	$\mathrm{IS}_{\mathrm{o}}(\mathcal{S})$	$\xrightarrow{w_{\mathrm{o}}}$	$\mathsf{D}^\sharp$
			classes co-initiales
Ordre préfixe sur $\widehat{\mathrm{T}}(\mathcal{S})$	$\leq$	$\dashrightarrow$	$\leq_{\mathrm{D}}$
Ordre d'élagage du joueur sur $\widetilde{\mathrm{T}}_{\mathrm{p}}(\mathcal{S})$	$\subseteq_{\mathrm{p}}$	$\dashrightarrow$	$\leq_{\mathrm{D}}$
Ordre préfixe sur $\widehat{\mathrm{IT}}(\mathcal{S})$	$\leq^\flat$	$\dashrightarrow$	$\leq^\flat_{\mathrm{D}}$
Ordre d'élagage du joueur sur $\widetilde{\mathrm{IT}}_{\mathrm{p}}(\mathcal{S})$	$\subseteq^\flat_{\mathrm{p}}$	$\dashrightarrow$	$\leq^\flat_{\mathrm{D}}$
Ordre d'élagage de l'opposant sur $\widetilde{\mathrm{T}}_{\mathrm{o}}(\mathcal{S})$	$\subseteq_{\mathrm{o}}$	$\dashrightarrow$	$\geq_{\mathrm{D}}$
Ordre d'élagage de l'opposant sur $\widetilde{\mathrm{IT}}_{\mathrm{o}}(\mathcal{S})$	$\subseteq^\flat_{\mathrm{o}}$	$\dashrightarrow$	$\leq^\sharp_{\mathrm{D}} = \geq^\flat_{\mathrm{D}}$

FIG. 5.7.2. Interprétation de l'algèbre syntaxique complète dans une structure d'évaluation régulière ambivalente, à la fois **dcpo** et **co-dcpo**



CHAPITRE 6

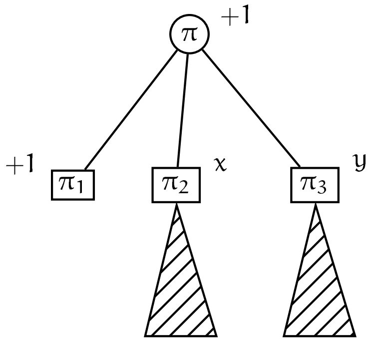
L'algorithme Alpha-Bêta

Le but de ce chapitre est celui d'étudier les propriétés de l'algorithme Alpha-Bêta, défini pour la structure d'évaluation des entiers, et de l'adapter à un plus grand spectre de structures d'évaluations. Nous introduirons aussi un nouvel algorithme, la méthode Alpha-Bêta-Gamma-Delta, plus prudent, donc moins efficace que son prédécesseur, mais toujours correct, même lorsqu'on relâchera certaines hypothèses sur la structure d'évaluation utilisée.

6.1. Introduction

L'algorithme Alpha-Bêta est une méthode de calcul de la valeur d'un jeu, présentée pour la première fois dans [**Hart & Edwards, 1961**], et largement étudiée depuis dans la littérature spécialisée des jeux combinatoires (cf. par exemple [**Knuth & Moore, 1975**] et [**Fuller et al., 1973**]). Le jeu est supposé être évalué dans la structure des entiers avec les opérations min et max. Pour résoudre ce type de jeu, c'est-à-dire pour en connaître la valeur (perdant ou gagnant, et combien), il est souvent inutile de visiter *entièrement* l'arbre de jeu découlant d'une position initiale. Une portion significative de l'arbre de jeu peut à l'occasion suffire à déterminer la valeur *exacte* de la position étudiée. Par exemple, durant l'évaluation d'un noeud du joueur (où l'on maximise le gain), si une branche est évaluée à la valeur maximale $\top$ du domaine d'évaluation D, il est bien inutile de calculer les valeurs des branches non encore visitées : la valeur ne pourra être supérieure, de toute manière, à la valeur déjà atteinte. D'une façon duale, dans un noeud de l'opposant (où l'on minimise le gain), une fois la valeur $\perp$ atteinte, c'est-à-dire une fois obtenu tout ce que l'opposant pouvait espérer de mieux, nous pouvons aussitôt interrompre le procédé d'évaluation, évitant ainsi de parcourir d'autres branches.

FIG. 6.1.1. Coupures *minimax*



6.1.1. Coupures minimax. La figure 6.1.1 illustre le cas d'une position du joueur, en évaluation dans la structure $D = \{-1, 0, +1\}$ (perdant, match null, gagnant), où l'analyse du premier coup du joueur permet de répondre $+1$, et cela indépendamment des valeurs inconnues x et y des autres deux coups possibles.

Algorithm 1 *minimax*

```
fonction minimax(π : ℙ) : D

    1. si π ∈ ℙ_T alors retourner p(π)

    2. sinon soit π_1,...,π_n la liste des successeurs de π

    3. si λ(π) = Player alors
           3.1) v ← ⊥
           3.2) pour i allant de 1 à n faire
                v ← v max minimax(π_i)

    4. si λ(π) = Opponent alors
           4.1) v ← ⊤
           4.2) pour i allant de 1 à n faire
                v ← v min minimax(π_i)

    5. retourner v
```

L'évaluation *exhaustive* de l'arbre de jeu est la procédure qui, sur le modèle de la fonction *v* d'évaluation des jeux finis (cf. section), calcule la valeur d'un jeu en parcourant l'arbre de jeu dans sa totalité (cf. algorithme 1). Ce type de visite est appelé aussi algorithme *minimax*, évoquant la rétro-propagation des valeurs, à partir des feuilles vers la racine, par l'application des fonctions min et max des joueurs. Le critère d'élagage décrit auparavant, où l'on vérifie l'obtention du gain maximal ($\top$ dans un noeud du joueur, $\perp$ dans un noeud de l'opposant), correspond à une remarque tellement simple que la recherche exhaustive équipée d'un tel test est souvent appelée, une fois de plus, méthode minimax. Cependant, Judea Pearl, auteur de l'ouvrage de référence sur les techniques heuristiques (cf. [**Pearl, 1990**]), lui réserve le nom de méthode *solve* (cf. algorithme 2).

Algorithm 2 *Solve* (minimax avec test du gain maximal)

fonction minimax(π : $\mathbb{P}$) : D

1. si $\pi \in \mathbb{P}_T$ alors retourner $p(\pi)$

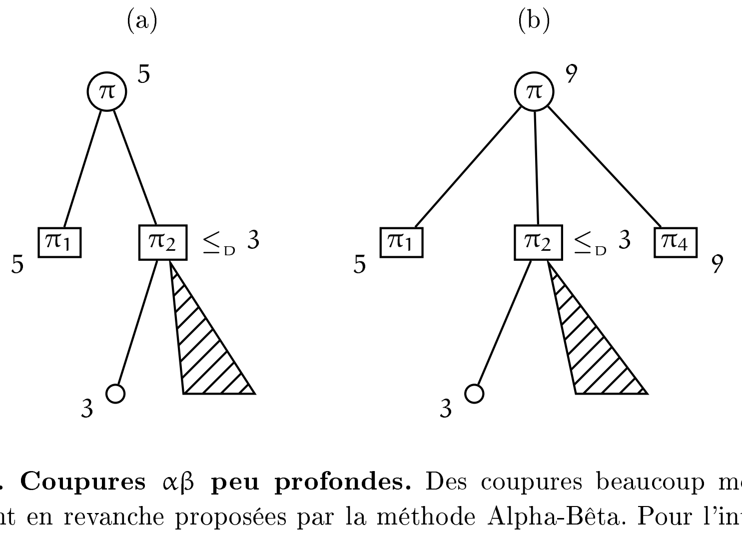
2. sinon soit $\pi_1, \ldots, \pi_n$ la liste des successeurs de π

3. si $\lambda(\pi) = \text{Player}$ alors
   3.1) v $\leftarrow \bot$
   3.2) pour i allant de 1 à n
        et tant que v $<_D \top$ faire
        v $\leftarrow$ v max minimax(π_i)

4. si $\lambda(\pi) = \text{Opponent}$ alors
   4.1) v $\leftarrow \top$
   4.2) pour i allant de 1 à n
        et tant que $\bot <_D$ v faire
        v $\leftarrow$ v min minimax(π_i)

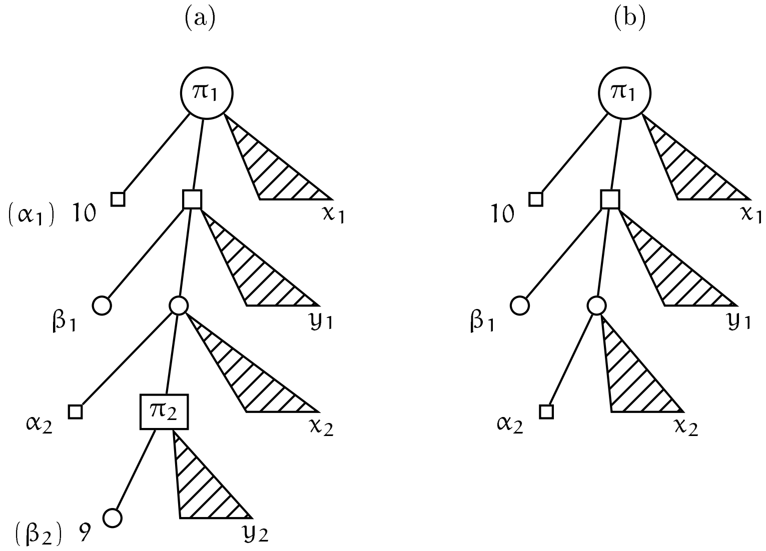
5. retourner v

FIG. 6.1.2. Coupures $\alpha\beta$ peu profondes



6.1.2. Coupures $\alpha\beta$ peu profondes. Des coupures beaucoup moins évidentes sont en revanche proposées par la méthode Alpha-Bêta. Pour l'introduire, supposons qu'on veuille calculer la valeur, dans la structure d'évaluation des entiers naturels $D = \mathbb{N}$ (avec les fonctions min et max), d'une position dont l'arbre aura la forme représentée dans la figure 6.1.2.(a). Dans la position π, le joueur aura un premier coup conduisant à la position terminale π_1 dont le payoff sera 5, et il aura un second coup conduisant vers une position π_2 de l'opposant, cette fois-ci non terminale. Dans cette dernière position, l'opposant aura, à son tour, un premier coup conduisant à un payoff égal à 3, et d'éventuels autres mouvements possibles. Malgré l'information incomplète sur l'évaluation de chacune de ses parties, nous connaissons d'ores et déjà la valeur exacte de l'arbre tout entier. En effet, puisque la fonction min renvoie une valeur inférieure à tous ses arguments, la valeur de

FIG. 6.1.3. Coupures $\alpha\beta$ profondes



la position π_2 sera forcément inférieure à 3. Ainsi, dans la position à la racine, le joueur devra au bout du compte choisir entre un gain égal à 5 et un gain qui sera certainement inférieur à 3, donc inférieur à 5. Nous pouvons donc prévoir que le joueur choisira son premier coup (la valeur de la position sera 5), et cela sans visiter les autres coups à disposition de l'opposant dans la position π_2.

L'exemple précédent ne doit pas induire en erreur : la coupure n'est pas justifiée par le fait que le premier coup analysé est, par un heureux hasard, le meilleur coup possible pour le joueur dans la position à la racine. En effet, même si le premier coup examiné n'avait pas été le meilleur (comme dans la figure 6.1.2.(b)), la valeur de la position π_2 n'aurait eu, de toute manière, aucune influence sur la valeur de la position π.

Ce type de coupures, dont la cause se situe un niveau seulement au dessus du noeud où elles surviennent, seront qualifiées de *peu profondes*. Les coupures qui se justifient, en revanche, par l'analyse de plusieurs niveaux au-dessus du noeud impliqué dans l'arbre de jeu, seront appelées coupures (ou ruptures) *profondes*.

6.1.3. Coupures $\alpha\beta$ profondes. Considérons la situation représentée dans la figure 6.1.3.(a). Le joueur dispose, dans la position initiale π_1, d'un coup dont le payoff est 10. Nous pouvons montrer, indépendamment des valeurs x_1, β_1, y_1, α_2 et x_2, que si l'opposant dispose à son tour, dans la position π_2, d'un coup dont le payoff est 9 (donc inférieur à 10), alors la valeur de la position π_2 ne pourra conditionner la valeur de la position à la racine. En effet, la formule qui calcule la valeur de π_1 est la suivante (nous écrivons $\uparrow$ pour signifier la fonction max, et $\downarrow$ pour signifier la fonction min) :

$$10 \uparrow \{\beta_1 \downarrow [\alpha_2 \uparrow (9 \downarrow y_2) \uparrow x_2] \downarrow y_1\} \uparrow x_1$$

Pour faciliter la lecture et retrouver une notation familière, nous pouvons remplacer les symboles $\uparrow$ et $\downarrow$ respectivement avec les symboles $\cdot$ et $+$, utilisés habituellement en algèbre pour dénoter des opérations distributives :

$$10 \cdot \{\beta_1 + [\alpha_2 \cdot (9 + y_2) \cdot x_2] + y_1\} \cdot x_1$$

En appliquant justement la propriété distributive, nous vérifierons que la formule ci-dessus coïncide avec la suivante :

$$\{10 \cdot \beta_1 + [10 \cdot \alpha_2 \cdot (9 + y_2) \cdot x_2] + 10 \cdot y_1\} \cdot x_1$$

Nous remarquerons à présent que $(9 + y2)$ est forcément inférieur à 9 donc à 10, ce qui permet d'éliminer cet argument de la formule (puisque la fonction $\cdot =\uparrow=$ max ignore les arguments plus petits que d'autres) :

$$\{10 \cdot \beta_1 + [10 \cdot \alpha_2 \cdot x_2] + 10 \cdot y_1\} \cdot x_1$$

Enfin, en appliquant la propriété distributive dans le sens inverse, nous obtenons la formule :

$$10 \cdot \{\beta_1 + [\alpha_2 \cdot x_2] + y_1\} \cdot x_1$$

qui correspond précisément à l'évaluation de l'arbre de la figure 6.1.3.(b). En d'autres termes, la valeur de la position π_2 n'a aucun intérêt dans le contexte où elle se trouve : du point de vue du joueur, qui examine ses possibilités à la position π_1, tout se passe comme si elle n'existait pas. Cela implique, entre autres, qu'après avoir calculé la valeur 9 du premier coup de l'opposant dans la position π_2, il n'y aura aucune raison d'évaluer les autres coups disponibles, la valeur inconnue y_2 étant aussi insignifiante que la valeur 9 établie.

6.1.4. L'algorithme Alpha-Bêta conventionnel. La méthode Alpha-Bêta, qui réalise les deux types de coupures examinées, est habituellement présentée sous la forme d'une fonction récursive dans un pseudo-langage de programmation impératif (cf. algorithme 3). La fonction `Alpha-Bêta` analyse une position $\pi \in \mathbb{P}$ du jeu avec les paramètres additionnels α et β, représentant une sorte de *seuil* à ne pas dépasser dans le calcul graduel de la valeur de la position π, faute de quoi, la progression du calcul sera interrompue. Comme illustré dans la figure 6.1.3.(a), dans un état quelconque de l'évolution de l'algorithme, nous nous trouverons dans un noeud d'un arbre de jeu avec des ancêtres (du joueur ou de l'opposant), dont l'évaluation aura été suspendue dans l'attente du résultat de l'évaluation d'un noeud fils. Pour tout ancêtre correspondant à la position du jeu π_i, nous pourrons discerner :

(1) la valeur déjà calculée (α_i si la position est du joueur, β_i sinon), que nous appellerons valeur *inachevée*,
(2) la valeur du fils actuellement en évaluation, que nous appellerons valeur *d'actualité*,
(3) la valeur totale des autres fils dont l'évaluation n'a toujours pas débuté (x_i si la position est du joueur, y_i sinon), que nous appellerons valeur *inconnue*.

Avec cette terminologie, nous pouvons mieux préciser le sens des paramètres α et β de la fonction `Alpha-Bêta` :

Algorithm 3 *Alpha-Bêta conventionnel*

```
fonction Alpha-Bêta(π : ℙ, α,β : D) : D

    1. si π ∈ ℙ_T alors retourner p(π)

    2. sinon soit π_1,...,π_n la liste des successeurs de π

    3. si λ(π) = Player alors
          3.1) v ← ⊥
          3.2) pour i allant de 1 à n
                 et tant que (v <_D β) faire
                 v ← v max Alpha-Bêta(π_i, α max v, β)

    4. si λ(π) = Opponent alors
          4.1) v ← ⊤
          4.2) pour i allant de 1 à n
                 et tant que (α <_D v) faire
                 v ← v min Alpha-Bêta(π_i, α, β min v)

    5. retourner v
```

– le seuil α est le résultat de l'application de la fonction *du joueur* (max) à toutes les valeurs *inachevées* des noeuds ancêtres *du joueur*

– le seuil β est le résultat de l'application de la fonction de l'opposant (min) à toutes les valeurs *inachevées* des noeuds ancêtres *de l'opposant*

En effet, dans les noeuds *du joueur*, l'appel récursif :

$$\texttt{Alpha-Bêta}(\pi_i, \alpha \max \mathtt{v}, \beta)$$

propage le total des valeurs inachevées *du joueur* ($\alpha \max \mathtt{v}$) à l'évaluation de la branche fille. De façon duale, dans les noeuds *de l'opposant*, l'appel récursif :

$$\texttt{Alpha-Bêta}(\pi_i, \alpha, \beta \min \mathtt{v})$$

propage le total des valeurs inachevées *de l'opposant* à l'évaluation d'une branche fille. Dans un noeud du joueur en cours d'évaluation, si la valeur inachevée v du noeud dépasse déjà, malgré son imperfection (puisqu'elle pourrait s'agrandir encore), le total des valeurs inachevées de l'opposant, c'est-à-dire le seuil β, alors l'évaluation du noeud peut être interrompue. De façon duale, si la valeur inachevée v d'un noeud de l'opposant en cours d'évaluation dépasse déjà, malgré son imperfection (puisqu'elle pourrait s'amincir encore), le total des valeurs inachevées du joueur, c'est-à-dire le seuil α, alors l'évaluation du noeud peut être interrompue. Une seule condition de rupture, le dépassement des seuils α et β, regroupe et explique les deux types différents de coupures, les peu profondes et les profondes, que nous avons examinées dans les exemples précédents (cf. figure 6.1.2, 6.1.3.(a)).

L'assimilation des deux types de coupures s'expliquera par la propriété *distributive* des opérations min et max sur les entiers. En revanche, dans d'autres structures possibles, la distinction des deux types de coupures sera pertinente, la correction des coupures profondes n'étant pas toujours correcte. Nous n'expliquerons pas cela en détails, mais toujours grâce à la propriété distributive du min et du max, la méthode Alpha-Bêta est parfois présentée d'une façon où l'on ajoute systématiquement le seuil des ancêtres (α dans les noeuds du joueur, β dans ceux de l'opposant) à la valeur en construction. Cela donne un algorithme un peu plus concis et élégant, la condition de rupture ($\alpha \geq_D \beta$) étant la même dans les deux types de noeud (cf. algorithme 4).

Algorithm 4 *Alpha-Bêta conventionnel (deuxième version)*

fonction Alpha-Bêta(π : $\mathbb{P}$, α, β : D) : D

1. si $\pi \in \mathbb{P}_T$ alors retourner $p(\pi)$

2. sinon soit $\pi_1, \ldots, \pi_n$ la liste des successeurs de π

3. si $\lambda(\pi) = \text{Player}$ alors
   (3.1) pour i allant de 1 à n
   et tant que ($\alpha <_D \beta$) faire
   $\alpha \leftarrow \alpha \max$ Alpha-Bêta(π_i, α, β)
   (3.2) retourner α

4. si $\lambda(\pi) = \text{Opponent}$ alors
   (4.1) pour i allant de 1 à n
   et tant que ($\alpha <_D \beta$) faire
   $\beta \leftarrow \beta \min$ Alpha-Bêta(π_i, α, β)
   (4.2) retourner β

Dans les deux cas, l'évaluation d'une position démarre par un appel à la fonction `Alpha-Bêta` sur la position concernée et avec la fenêtre la plus large possible, c'est-à-dire avec les paramètres $\alpha = \perp$ et $\beta = \top$ (dans le cas des entiers, ce sera avec les valeurs limites $\alpha = -\infty$ et $\beta = +\infty$). La fonction visitera les profondeurs de l'arbre de jeu en resserrant progressivement la fenêtre $[\alpha, \beta]$, et en augmentant, de cette manière, la probabilité des coupures.

6.2. Structures d'évaluations compatibles

La correction des coupures illustrées dans les exemples précédents, c'est-à-dire la correction de la méthode Alpha-Bêta, s'appuie sur des propriétés implicites des fonctions min et max. Nous allons, dans un premier temps, essayer d'isoler ces propriétés toujours dans le cadre de la structure des entiers avec min et max. Ensuite, nous ferons abstraction de la structure d'évaluation utilisée, nous définirons l'ensemble des propriétés souhaitées et, dans la prochaine section, nous prouverons que ces propriétés abstraites autorisent les coupures. Dans cette démarche, l'objectif sera, bien entendu, celui de deviner les conditions les plus générales, c'est-à-dire plus faibles d'un point de vue logique. Elles devront nous permettre, malgré leur faiblesse, de calculer plus rapidement, mais de façon exacte, la valeur d'une position du jeu dans une structure d'évaluation arbitraire $\mathcal{D} = (D, \leq_D, \uparrow, \downarrow)$.

6.2.1. Analyse des coupures $\alpha\beta$ peu profondes. Les propriétés qui justifient les coupures peu profondes des exemples précédents peuvent être résumées de la façon suivante :

(I) la fonction min de l'opposant renvoie un résultat *plus petit* que ses arguments ;

(II) la fonction max du joueur ignore les arguments *plus petits* que d'autres, nous dirons qu'elle est *élitiste* :

$$x \leq_D y_i \quad \Rightarrow \quad \max y_1..y_k \,.\, x \,.\, y_{k+1}..y_n \;=\; \max y_1..y_k.y_{k+1}..y_n$$

Le qualificatif *élitiste*, se justifie étant donné que lorsque un argument x (dans l'exemple $x \leq 3$) de la fonction max du joueur est plus petit qu'un second argument y (dans l'exemple $y = 5$, donc $x \leq y$), le premier n'a aucune influence sur le résultat de la fonction. Autrement dit, en éliminant x et en appliquant la fonction à tous les autres arguments, nous obtiendrons le même résultat. Dans ce cas, nous disons que x est *couvert* par y.

La façon duale d'opérer des coupures peu profondes (dans les noeuds du *joueur*, avec une information partielle de la valeur au niveau supérieur de l'*opposant*) est, bien entendu, correcte par des considérations duales :

(III) la fonction max du joueur renvoie un résultat *plus grand* que ses arguments ;

(IV) la fonction min de l'opposant ignore les arguments *plus grands* que d'autres ; nous disons qu'elle est *co-élitiste* :

$$x \geq_D y_i \quad \Rightarrow \quad \min y_1..y_k \,.\, x \,.\, y_{k+1}..y_n \;=\; \min y_1..y_k.y_{k+1}..y_n$$

Lorsqu'un argument x de la fonction min de l'opposant est *plus grand* qu'un second argument y, le premier n'a aucune influence sur le résultat de la fonction et nous pouvons donc éliminer x des arguments appliqués à la fonction.

En soulignant la dualité des propriétés (II) et (IV), nous pouvons les considérer comme une seule condition. En effet, dans les deux cas, les opérations (celle du joueur max ou celle de l'opposant min) permettent de supprimer ce qui n'est pas essentiel. Dans ce sens, nous dirons que les fonctions des joueurs sont *simplifiables*.

6.2.2. Analyse des coupures $\alpha\beta$ profondes. Pour ce type de coupure, les propriétés algébriques énoncées précédemment ne suffisent pas. En effet, nous avons expliqué la coupure par la propriété *distributive*, qui est une propriété supplémentaire des fonctions min et max sur les entiers :

(V) la fonction max du joueur (en notation infixe) se distribue sur celle de l'opposant :

$$x \max (y \min z) \;=\; (x \max y) \min (x \max z)$$

(VI) la fonction min de l'opposant (en notation infixe) se distribue sur celle du joueur :

$$x \min (y \max z) = (x \min y) \max (x \min z)$$

Nous avons vu que la (V) permet les coupures, dans les noeuds de l'*opposant,* justifiées par une valeur calculée dans un noeud antécédent du *joueur*. Vice versa, (VI) permettra les coupures, dans les noeuds du *joueur*, justifiés par une valeur calculée dans un noeud antécédent de l'*opposant*.

6.2.3. Extension d'une opération binaire aux suites. Les fonctions min et max définies sur les suites, $\min, \max : \mathbb{Z}^* \to \mathbb{Z}$, peuvent être considérées comme les extensions des versions binaires correspondantes. Plus précisément, considérons l'opération $\min_2 : \mathbb{Z} \times \mathbb{Z} \to \mathbb{Z}$, que nous appellerons *noyaux binaires* de min, définie par :

$$\min_2(x, y) = \begin{cases} x & \text{si } x \leq_{\mathbb{Z}} y \\ y & \text{sinon} \end{cases}$$

Le minimum d'un seul argument est l'argument lui-même. Donc, la version unaire $\min_1 : \mathbb{Z} \to \mathbb{Z}$ de la fonction minimum est l'identité : $\min_1 x = x$. Cette dernière peut être définie de façon équivalente en terme du noyau binaire : $\min_1 x \triangleq \min_2(x, x)$. Ainsi, le noyau binaire du min est suffisant pour définir inductivement la fonction sur les suites *finies* de longueur arbitraire :

$$(\star) \qquad \begin{cases} \min u \triangleq \min_1 u & \text{si } |u| = 1 \\ \min a.u \triangleq a \min_2 (\min u) & \text{si } |u| > 1 \end{cases}$$

A priori, nous pourrions considérer que la version unaire soit indépendante de la version binaire, et que les deux versions ensemble constituent la base inductive, dans le style de la définition $(\star)$, des opérations $\downarrow$ et $\uparrow$ des joueurs (dont le domaine d'application est celui des suites finies d'éléments de D). L'opération *unaire* définirait explicitement l'application pour un argument, c'est-à-dire pour les suites de longueur 1, l'opération *binaire* (associative et commutative) définirait explicitement l'application pour deux arguments, c'est-à-dire pour les suites de longueur 2. Les deux opérations ensemble définiraient l'application pour un nombre fini mais arbitraire d'arguments. En réalité, les conditions de simplification (élitisme et coélitisme), requises par les fonctions des joueurs, ne permettent pas une définition de l'opération unaire indépendante de la version binaire. Pour deux arguments x et y, tels que $x \leq y$, la fonction binaire du joueur, que nous noterons $\overset{2}{\uparrow}$, devra éliminer l'option x. Cela revient à énoncer la propriété suivante, où $\overset{1}{\uparrow}$ est la version unaire de l'opération du joueur :

$$\text{(A)} \qquad x \leq y \;\Rightarrow\; x \overset{2}{\uparrow} y = \overset{1}{\uparrow} y$$

Considérant que les opérations binaires utilisées sont *associatives* et *commutatives*, nous ne serons pas obligés d'énoncer la condition symétrique :

$$x \leq y \;\Rightarrow\; y \overset{2}{\uparrow} x = \overset{1}{\uparrow} y$$

Dans le cas particulier $x = y$, nous obtiendrons $\overset{1}{\uparrow} x = x \overset{2}{\uparrow} x$. L'opération unaire n'est donc pas essentielle, elle aussi est déterminée par la version binaire.

Les axiomes de simplification ont d'autres conséquences immédiates. Pour trois arguments, l'élitisme de la fonction du joueur implique la condition suivante :

$$\text{(B)} \qquad x \leq y \;\;\Rightarrow\;\; x \overset{2}{\uparrow} y \overset{2}{\uparrow} z = y \overset{2}{\uparrow} z$$

Or, nous pouvons prouver que, dans l'hypothèse (A), la condition (B) est équivalente à la suivante :

$$\text{(B')} \qquad x \overset{2}{\uparrow} y = (\overset{1}{\uparrow} x) \overset{2}{\uparrow} y$$

Cette condition, que nous appellerons *insertion*, exprime le fait que l'opération binaire peut être considérée comme une sorte d'extension de l'opération unaire. En effet, appliquer l'opération unaire aux arguments avant d'appliquer l'opération binaire ne change en rien le résultat : la binaire inclut déjà, dans un certain sens, la transformation opérée par l'opération unaire. La preuve que (B) implique (B') est la suivante :

$$\begin{array}{rcll} x \overset{2}{\uparrow} y & = & x \overset{2}{\uparrow} x \overset{2}{\uparrow} y & \qquad \text{(B)}, x \leq x \\ & = & (\overset{1}{\uparrow} x) \overset{2}{\uparrow} y & \qquad \text{(A)} \end{array}$$

Vice versa, la preuve que (B') implique (B) est la suivante :

$$\begin{array}{rcll} x \overset{2}{\uparrow} y \overset{2}{\uparrow} z & = & (\overset{1}{\uparrow} y) \overset{2}{\uparrow} z & \qquad \text{(A)}, x \leq y \\ & = & y \overset{2}{\uparrow} z & \qquad \text{(B')} \end{array}$$

Ces considérations expliquent la définition de noyau binaire.

Definition 6.2.1. (Noyau binaire) *Une fonction* $\overset{2}{\mathsf{f}} : \mathsf{D} \times \mathsf{D} \to \mathsf{D}$ *(en notation infixe) est un* noyau binaire*, si elle vérifie, avec son* extension unaire $\overset{1}{\mathsf{f}} : \mathsf{D} \to \mathsf{D}$*, définie par* $\overset{1}{\mathsf{f}}(x) \triangleq x \overset{2}{\mathsf{f}} x$*, les conditions suivantes :*

$$\begin{array}{lll} \text{(A1)} & a \overset{2}{\mathsf{f}} (b \overset{2}{\mathsf{f}} c) = (a \overset{2}{\mathsf{f}} b) \overset{2}{\mathsf{f}} c & \qquad \textit{(associative)} \\ \text{(A2)} & a \overset{2}{\mathsf{f}} b = b \overset{2}{\mathsf{f}} a & \qquad \textit{(commutative)} \\ \text{(A3)} & a \overset{2}{\mathsf{f}} b = (\overset{1}{\mathsf{f}} b) \overset{2}{\mathsf{f}} a & \qquad \textit{(insertion)} \end{array}$$

□

Sur le modèle de la définition (⋆), nous pouvons étendre un noyau binaire aux suites finies mais de longueur arbitraire.

Definition 6.2.2. (Extension d'un noyau binaire aux suites) *Étant donné un noyau binaire* $\overset{2}{\mathsf{f}} : \mathsf{D} \times \mathsf{D} \to \mathsf{D}$*, dont l'extension unaire est* $\overset{1}{\mathsf{f}} : \mathsf{D} \to \mathsf{D}$*, une fonction sur les suites* $\mathsf{f} : \mathsf{D}^* \to \mathsf{D}$ *est l'*extension du noyau binaire à un nombre arbitraire d'arguments*, noté* $\mathsf{f} = \mathsf{ext}(\overset{2}{\mathsf{f}})$*, si :*

$$\begin{array}{llcll} \text{(A5)} & \mathsf{f}(a) & = & \overset{1}{\mathsf{f}}(a) & \quad \textit{si} \quad |a| = 1 \\ \text{(A5)} & \mathsf{f}(a.u) & = & a \overset{2}{\mathsf{f}} (\mathsf{f}\, u) & \quad \textit{si} \quad |a.u| > 1 \end{array}$$

□

La définition d'extension ne concerne pas la valeur de la fonction sur la suite vide ε, qui peut donc être définie arbitrairement[1].

EXAMPLE 6.2.3. Pour illustrer la définition, considérons l'extension d'un noyau $\downarrow = \mathsf{ext}(\overset{2}{\downarrow})$ appliquée à une suite de longueur 3 (autrement dit, appliquée à trois arguments) :

$$\begin{array}{rcll}
\downarrow x.y.z & = & x \overset{2}{\downarrow} (\downarrow y.z) & \text{(A5)} \\
& = & x \overset{2}{\downarrow} (y \overset{2}{\downarrow} (\downarrow z)) & \text{(A5)} \\
& = & x \overset{2}{\downarrow} (y \overset{2}{\downarrow} (\overset{1}{\downarrow} z)) & \text{(A4)} \\
& = & x \overset{2}{\downarrow} (y \overset{2}{\downarrow} z) & \text{(A3)} \\
& = & x \overset{2}{\downarrow} y \overset{2}{\downarrow} z & \text{(A1),(A2)}
\end{array}$$

□

En termes du noyau binaire, la propriété (I), pour laquelle $\downarrow$ doit rendre un résultat plus petit (non plus grand) que ses arguments, s'énonce de la façon suivante :

$$\text{(C)} \qquad x \overset{2}{\downarrow} y \leq_D x$$

Dans l'hypothèse de l'axiome d'insertion (B'), la condition (C) équivaut à la conjonction des suivantes :

$$\begin{array}{ll}
\text{(C1)} & \overset{1}{\downarrow} x \leq_D x \\
\text{(C2)} & x \overset{2}{\downarrow} y \leq_D \overset{1}{\downarrow} x
\end{array}$$

La preuve que (C2) et (C1) impliquent (C) est simple : à partir de $x \overset{2}{\downarrow} y$ il suffira d'appliquer dans l'ordre (C2) puis (C1). La preuve que (C) implique (C1) est aussi simple, puisque (x, x) est un cas particulier de couple (x, y) où $y = x$. La preuve que (C) implique (C2) nécessite l'axiome d'insertion :

$$\begin{array}{rcll}
x \overset{2}{\downarrow} y & = & (\overset{1}{\downarrow} x) \overset{2}{\downarrow} y & \text{(Insertion)} \\
& \leq_D & (\overset{1}{\uparrow} x) & \text{(C)}
\end{array}$$

6.2.4. Structures de co-évaluation $\alpha\beta$-ordinaires. Dans une structure d'évaluation $\alpha\beta$-ordinaires, les fonctions des joueurs sont chacune l'extension d'un noyau binaire et les composantes de leur noyau vérifient les propriétés algébriques identifiées dans la section précédente, permettant à la fois les coupures *peu profondes* et les coupures *profondes*. Si d'ordinaire ces propriétés sont vérifiées par la structure d'évaluation utilisée, il se peut, dans certains cas, qu'elles ne soient pas vérifiés et que, cependant, les coupures soient toutefois justifiées. Par leur simplicité et leur caractère commun et intuitif, les structures $\alpha\beta$-ordinaires, dont l'exemple plus simple est celui des entiers naturels avec le min et le max, méritent d'être étudiées séparément. Nous découvrirons ainsi que cette classe est celle des treillis d'évaluation distributif, naturelle généralisation de la structure traditionnelle, où la borne supérieure remplace le max et la borne inférieure remplace le min. En

[1] pour le propos de cette thèse, les fonctions des joueurs auraient pu aussi bien être définies sur les domaines des suites non vides

revanche, nous prouverons la correction de la méthode Alpha-Bêta pour une classe de structures plus large, que nous qualifierons de $\alpha\beta$-*normales.*

DEFINITION 6.2.4. (STRUCTURE D'ÉVALUATION $\alpha\beta$-ORDINAIRES) *Une structure d'évaluation* $\mathcal{D} = (D, \leq_D, \uparrow, \downarrow)$ *est* $\alpha\beta$-ordinaire *si elle vérifie les conditions suivantes :*

(1) $\leq_D$ *est une relation de préordre sur* D

(2) *les fonctions des joueurs sont générées par un noyau binaire,* $\downarrow = \mathsf{ext}(\overset{2}{\downarrow})$ *et* $\uparrow = \mathsf{ext}(\overset{2}{\uparrow})$, *tous deux monotones (croissantes) pour la relation* $\leq_D$

(3) *l'opération* $\uparrow$ *renvoie des valeurs déplorables pour l'opposant, et vice-versa, l'opération* $\downarrow$ *renvoie des valeurs déplorables pour le joueur, c'est-à-dire :*

$$(x \overset{2}{\downarrow} y) \ \leq_D \ \overset{1}{\downarrow} x \ \leq_D \ x \ \leq_D \ \overset{1}{\uparrow} x \ \leq_D \ (x \overset{2}{\uparrow} y)$$

(4) *les opérations* $\uparrow$ *et* $\downarrow$ *sont simplifiable lorsqu'on détécte des arguments plus petit que d'autres :*

$$\begin{cases} x \leq_D y & \Rightarrow \quad x \overset{2}{\uparrow} y = \overset{1}{\uparrow} y \\ x \leq_D y & \Rightarrow \quad x \overset{2}{\downarrow} y = \overset{1}{\downarrow} x \end{cases}$$

(5) *les fonctions (binaires) des joueurs sont distributives :*

$$\begin{cases} x \overset{2}{\downarrow} (y \overset{2}{\uparrow} z) & = & (x \overset{2}{\downarrow} y) \overset{2}{\uparrow} (x \overset{2}{\downarrow} z) \\ \\ x \overset{2}{\uparrow} (y \overset{2}{\downarrow} z) & = & (x \overset{2}{\uparrow} y) \overset{2}{\downarrow} (x \overset{2}{\uparrow} z) \end{cases}$$

Lorsque la relation $\leq_D$ *est anti-symétrique, nous disons que* $\mathcal{D}$ *est un* ordre partiel $\alpha\beta$-ordinaire. □

6.2.5. Caractérisation des ordres partiels $\alpha\beta$-ordinaires. Pour la classe des ordres partiels $\alpha\beta$-ordinaires nous allons donner une caractérisation plus précise : nous allons prouver qu'elle coïncide exactement avec la classe des treillis d'évaluation distributifs. Par définition, tout ordre partiel $\alpha\beta$-ordinaire vérifie les axiomes de la table 1.

Le lemme suivant permet de remarquer que si deux points x et y, dans un ordre partiel $\alpha\beta$-ordinaire, ont une borne supérieure $z = \sup\{x, y\}$, et si la version unaire de l'opération du joueur ne modifie pas cette borne ($\overset{1}{\uparrow} z = z$), alors la version binaire appliquée aux deux points $x \overset{2}{\uparrow} y$ est égale à la borne z.

LEMMA 6.2.5. *Soit* $\mathcal{D} = (D, \leq_D, \uparrow, \downarrow)$ *un ordre partiel* $\alpha\beta$*-ordinaire. Alors :*

$$\begin{array}{llcl} (i) & z = \sup\{x, y\} \ \wedge \ \overset{1}{\uparrow} z = z & \Rightarrow & x \overset{2}{\uparrow} y = z \\ (ii) & w = \inf\{x, y\} \ \wedge \ \overset{1}{\downarrow} w = w & \Rightarrow & x \overset{2}{\downarrow} y = w \end{array}$$

TAB. 1. Axiomes des structures $\alpha\beta$-ordinaires

$$\mathcal{D} = (D, \leq_D, \uparrow, \downarrow)$$

(A6)	$\downarrow = \mathsf{ext}(\overset{2}{\downarrow})$	et	$\uparrow = \mathsf{ext}(\overset{2}{\uparrow})$	(noyaux binaire)
(A7)	$\overset{1}{\downarrow} x \leq_D x$	et	$\overset{1}{\uparrow} x \geq_D x$	(réductive, extensive)
(A8)	$x \overset{2}{\downarrow} y \leq_D \overset{1}{\downarrow} x$	et	$x \overset{2}{\uparrow} y \geq_D \overset{1}{\uparrow} x$	($\subseteq^*$-monotonie)
(A9)	$x \leq_D y \Rightarrow x \overset{2}{\downarrow} y = \overset{1}{\downarrow} x$	et	$x \leq_D y \Rightarrow x \overset{2}{\uparrow} y = \overset{1}{\uparrow} x$	(simplification)
(A10)	$x \overset{2}{\downarrow} (y \overset{2}{\uparrow} z) = (x \overset{2}{\downarrow} y) \overset{2}{\uparrow} (x \overset{2}{\downarrow} z)$	et	$x \overset{2}{\uparrow} (y \overset{2}{\downarrow} z) = (x \overset{2}{\uparrow} y) \overset{2}{\downarrow} (x \overset{2}{\uparrow} z)$	(distributives)

DÉMONSTRATION. *Par dualité nous nous limitons à la preuve de (i). Dans une structure d'évaluation, les opérations des joueurs sont, par définition, monotones (croissantes) dans tous les arguments. Ainsi :*

$$\begin{array}{rcll} x \overset{2}{\uparrow} y & \leq_D & \sup\{x,y\} \overset{2}{\uparrow} y & \overset{2}{\uparrow} \textit{croissante}, \quad x \leq_D \sup\{x,y\} \\ & = & \overset{1}{\uparrow} (\sup\{x,y\}) & \text{(A9)}, \quad y \leq_D \sup\{x,y\} \\ & = & \sup\{x,y\} & \textit{hypothèse} \end{array}$$

D'autre part, le résultat $x \overset{2}{\uparrow} y$ est supérieur aux deux arguments x et y. Il s'agit donc d'un majorant commun, qui sera forcément supérieur ou égal à $\sup\{x,y\}$, qui est par définition le plus petit des majorants. La seule condition possible est alors l'égalité $x \overset{2}{\uparrow} y = \sup\{x,y\}$. □

Dans le cadre d'un treillis $(D, \leq_D)$, ce lemme nous indique donc que si jamais les opérations unaires étaient égales à la fonction identité $\lambda x.x$, les opérations des joueurs coïncideraient avec les bornes du treillis. Nous allons prouver que l'effet de la propriété distributive est justement celui d'imposer que les versions unaires coïncident avec l'identité fonctionnelle.

PROPOSITION 6.2.6. *Soit $\mathcal{D} = (D, \leq_D, \uparrow, \downarrow)$ un ordre partiel $\alpha\beta$-ordinaire, où $\uparrow = \mathsf{ext}(\overset{2}{\uparrow})$ et $\downarrow = \mathsf{ext}(\overset{2}{\downarrow})$. Alors, les versions unaires des deux opérations sont forcément l'identité fonctionnelle : $\overset{1}{\downarrow} = \overset{1}{\uparrow} = \lambda x.x$.*

DÉMONSTRATION. *Nous commençons par prouver que chacune des opérations unaires $\overset{1}{\uparrow}$ et $\overset{1}{\downarrow}$ est l'identité sur l'image de l'autre, c'est-à-dire $\overset{1}{\downarrow} (\overset{1}{\uparrow} x) = \overset{1}{\uparrow} x$ et $\overset{1}{\uparrow} (\overset{1}{\downarrow} x) = \overset{1}{\downarrow} x$:*

$$\begin{array}{rcll} \overset{1}{\uparrow} x & = & x \overset{2}{\uparrow} (x \overset{2}{\downarrow} x) & \text{(A9)}, (x \overset{2}{\downarrow} x) \leq_D x \\ & = & (x \overset{2}{\uparrow} x) \overset{2}{\downarrow} (x \overset{2}{\uparrow} x) & \text{(A10)} \\ & = & \overset{1}{\downarrow} (\overset{1}{\uparrow} x) & \textit{def. de } \overset{1}{\downarrow} \end{array}$$

L'opération unaire $\rho = \overset{1}{\uparrow}$ est une fonction monotone (croissante), extensive ($\rho(x) \geq_D x$) et idempotente ($\rho(x) = x \overset{2}{\uparrow} x = (\overset{1}{\uparrow} x) \overset{2}{\uparrow} (\overset{1}{\uparrow} x) = \rho(\rho(x))$). Il s'agit donc d'un opérateur de clôture vers le haut (cf. Section 2.3.1.2). De même, l'opération unaire de l'opposant $\sigma = \overset{1}{\downarrow}$ est un opérateur de clôture mais, cette fois, vers le bas ($\sigma(x) \leq_D x$). Chacun des *deux coïncide donc avec l'identité tout au moins sur l'image de l'opérateur dual. Supposons alors, par l'absurde, qu'il existe un point* $\mathtt{b}$ *tel que* $\rho(\mathtt{b}) \neq \mathtt{b}$. *Puisque* ρ *est un opérateur de clôture vers le haut, nous aurons forcément* $\rho(\mathtt{b}) >_{\mathtt{D}} \mathtt{b}$. *Soit* $\mathtt{c} = \rho(\mathtt{b})$. *Puisque* $\mathtt{c}$ *est image de* ρ, *nous aurons* $\sigma(\mathtt{c}) = \mathtt{c}$ *et donc, pour la même raison,* $\rho(\mathtt{c}) = \mathtt{c}$. *D'autre part, puisque* $\rho(\mathtt{b}) >_{\mathtt{D}} \mathtt{b}$, *nous aurons forcément* $\sigma(\mathtt{b}) \neq \mathtt{b}$ *(sinon* $\rho(\mathtt{b})$ *serait égal à* $\mathtt{b}$*). Puisque* σ *est un opérateur de clôture vers le bas, nous aurons forcément* $\sigma(\mathtt{b}) <_{\mathtt{D}} \mathtt{b}$. *Soit* $\mathtt{a} = \sigma(\mathtt{b})$. *Puisque* $\mathtt{a}$ *est image de* σ, *nous aurons* $\rho(\mathtt{a}) = \mathtt{a}$, *donc* $\sigma(\mathtt{a}) = \mathtt{a}$. *Nous avons ainsi les faits suivants :*

$$\left\{\begin{array}{lllllllll} \mathtt{a} \overset{2}{\uparrow} \mathtt{b} & = & \overset{1}{\uparrow} \mathtt{b} & = & \rho(\mathtt{b}) & = & \mathtt{c} & & (\text{A9}),\ \mathtt{a} <_{\mathtt{D}} \mathtt{b} \\ \mathtt{a} \overset{2}{\uparrow} \mathtt{c} & = & \overset{1}{\uparrow} \mathtt{c} & = & \rho(\mathtt{c}) & = & \mathtt{c} & & (\text{A9}),\ \mathtt{a} <_{\mathtt{D}} \mathtt{c} \\ \mathtt{b} \overset{2}{\downarrow} \mathtt{c} & = & \overset{1}{\downarrow} \mathtt{b} & = & \sigma(\mathtt{b}) & = & \mathtt{a} & & (\text{A9}),\ \mathtt{b} <_{\mathtt{D}} \mathtt{c} \end{array}\right.$$

Nous avons alors déterminé un triplé $(\mathtt{a}, \mathtt{b}, \mathtt{c})$ *qui ne vérifie pas la propriété distributive, ce qui est absurde :*

$$\left\{\begin{array}{lllllll} \mathtt{a} \overset{2}{\uparrow} (\mathtt{b} \overset{2}{\downarrow} \mathtt{c}) & = & \mathtt{a} \overset{2}{\uparrow} \mathtt{a} & = & \rho(\mathtt{a}) & = & \mathtt{a} \\ (\mathtt{a} \overset{2}{\uparrow} \mathtt{b}) \overset{2}{\downarrow} (\mathtt{a} \overset{2}{\uparrow} \mathtt{c}) & = & \mathtt{c} \overset{2}{\downarrow} \mathtt{c} & = & \sigma(\mathtt{c}) & = & \mathtt{c} \end{array}\right.$$

□

Les deux résultats précédents permettent d'affirmer que, si la sous-structure $(\mathtt{D}, \leq_{\mathtt{D}})$ est un treillis (les noyaux binaires de sup et inf sont des fonctions totales), alors les opérations des joueurs d'un ordre partiel $\alpha\beta$-ordinaire $\mathcal{D} = (\mathtt{D}, \leq_{\mathtt{D}}, \uparrow, \downarrow)$ coïncident forcément, à cause des axiomes requis, avec les bornes du treillis. Il s'agira donc d'un treillis distributif. Une autre question se pose à présent : serait-il possible qu'une structure $\alpha\beta$-ordinaire existe sans pour autant que sa sous-structure $(\mathtt{D}, \leq_{\mathtt{D}})$ soit un treillis ? En d'autre termes, si la sous-structure $(\mathtt{D}, \leq_{\mathtt{D}})$ n'était pas un treillis, serait-il possible de compléter les fonctions $\inf, \sup : \mathtt{D} \times \mathtt{D} \rightharpoonup \mathtt{D}$ (strictement partielles) de façon à construire des opérations des joueurs $\overset{2}{\downarrow}$ et $\overset{2}{\uparrow}$ respectant les axiomes ? La proposition suivante répond négativement à la question en caractérisant complètement les ordres partiels $\alpha\beta$-ordinaires.

PROPOSITION 6.2.7. *Soit* $\mathcal{D} = (\mathtt{D}, \leq_{\mathtt{D}}, \uparrow, \downarrow)$ *une structure d'évaluation. Alors les deux conditions suivantes sont équivalentes :*

(1) $\mathcal{D}$ *est un ordre partiel* $\alpha\beta$*-ordinaire*

(2) $(\mathtt{D}, \leq_{\mathtt{D}})$ *est un treillis distributif où* $\sup = \uparrow$ *et* $\inf = \downarrow$

DÉMONSTRATION. *Les opérations* sup *et* inf *vérifient trivialement les axiomes de (A1) à (A10), ce qui prouve l'implication (2)* $\Rightarrow$ *(1). Vice versa, supposons que* $\mathcal{D}$ *soit une structure* $\alpha\beta$*-ordinaire. Supposons par l'absurde que* $(\mathtt{D}, \leq_{\mathtt{D}})$ *ne soit pas un treillis. Supposons qu'il existe un couple de points* $\mathtt{a}$ *et* $\mathtt{b}$ *tel que la borne supérieure n'existe pas (le cas de la borne inférieure est dual). Soit alors* $\mathtt{c} = \mathtt{a} \overset{2}{\uparrow} \mathtt{b}$ *(la fonction du joueur est, par définition, une fonction totale). Par les axiomes (A8) et (A9), le point* $\mathtt{c}$ *est un majorant à la fois de* $\mathtt{a}$ *et de* $\mathtt{b}$. *Puisqu'il existe des majorants, la non existence de la borne supérieure s'explique par la non existence d'un minimum parmi ces derniers. Il existe alors un majorant*

$d \in \mathrm{ubs}_{(D,\leq_D)}\{a, b\}$ *tel que* c *et* d *n'ont pas un minorant commun dans* $\mathrm{ubs}_{(D,\leq_D)}\{a, b\}$*. Cela implique, en particulier, que le couple* (c, d) *n'appartient pas à l'ordre partiel* $\leq_D$*. Grâce à la propriété distributive, nous pouvons alors prouver l'équation* $c \overset{2}{\downarrow} d = c$ *:*

$$\begin{array}{rcll}
c \overset{2}{\downarrow} d & = & (a \overset{2}{\uparrow} b) \overset{2}{\downarrow} d & c = a \overset{2}{\uparrow} b \\
& = & (a \overset{2}{\uparrow} b) \overset{2}{\downarrow} (a \overset{2}{\uparrow} d) & a \leq_D d,\ (A9)\ ,\ \ \overset{1}{\uparrow} = \lambda x.x \\
& = & a \overset{2}{\uparrow} (b \overset{2}{\downarrow} d) & (A10) \\
& = & a \overset{2}{\uparrow} b & b \leq_D d,\ (A9)\ ,\ \ \overset{1}{\downarrow} = \lambda x.x \\
& = & c &
\end{array}$$

qui viole l'axiome (A8), le résultat c *n'étant pas inférieur ou égal à l'argument* d *de l'opération* $\overset{2}{\downarrow}$*.* □

6.3. Théorème Alpha-Bêta polymorphe

Dans la construction d'un algorithme Alpha-Bêta polymorphe (cf. algorithme 6), utilisable pour une structure $\alpha\beta$-ordinaire quelconque, nous devrons non seulement remplacer, bien entendu, les fonctions max et min par les génériques $\uparrow$ et $\downarrow$, mais nous devrons aussi remplacer les conditions du type $(\nu <_{\mathbb{Z}} \beta)$ et $(\nu >_{\mathbb{Z}} \alpha)$, qui autorisent la continuité du calcul (`tant que [...] faire [...]`), par des conditions du type $(\nu \not\geq_D \beta)$ et $(\nu \not\leq_D \alpha)$ qui correspondent, en revanche, à la négation des conditions d'arrêt du calcul $(\nu \geq_D \beta)$ et $(\nu \leq_D \alpha)$. La différence n'est pas négligeable si, en faisant abstraction de l'ordre sur les entiers, l'on considère la relation $\leq_D$ comme une relation *partielle*, non forcément *totale* comme l'ordre sur les entiers. Si deux valeurs ne sont pas comparables, nous ne devrons pas pour autant interrompre le calcul. Autrement dit, la relation de préordre $\leq_D$ représentera précisément l'ensemble des couples vérifiant la condition de coupure : $x \leq_D y$ si et seulement si y est préférable à x pour le *joueur* et, vice-versa, y est préférable à x pour l'*opposant*.

Une autre modification simple, cependant lourde de conséquences, peut être apportée à la méthode. En effet, la présence des heuristiques permet d'initialiser la valeur inachevée v du noeud, directement à la valeur fournie par l'heuristique concernée (*pessimiste* dans les noeuds du joueur : $v \leftarrow h_1(\pi)$ au lieu de $v \leftarrow \bot$, *optimiste* dans ceux de l'opposant : $v \leftarrow h_2(\pi)$ au lieu de $v \leftarrow \top$). S'inscrivant dans la logique de coupure Alpha-Bêta, cela permettra d'interrompre l'évaluation d'un noeud sans l'avoir jamais commencée. Si en Recherche Opérationnelle on considère certains problèmes combinatoires, notamment celui du chemin de coût minimal, comme des jeux à deux joueurs (même si l'opposant est un joueur fictif qui accepte simplement les choix de l'autre), on découvre que les algorithmes de résolution bien connus *branch&bound*, coïncident en réalité avec la méthode Alpha-Bêta où l'initialisation se fait par les valeurs heuristiques[2]. Ces améliorations sont prises en compte dans l'algorithme 5.

Cependant, l'inconvénient de cette solution est que la valeur de retour de l'algorithme est une synthèse de valeurs *approximatives* (celles des heuristiques) et

[2]La présentation formelle de cet isomorphisme se développe sans difficultés majeures mais sort du cadre de cette thèse.

Algorithm 5 *Alpha-Bêta polymorphe, avec initialisations heuristiques*

fonction Alpha-Bêta(π : $\mathbb{P}$, α, β : D) : D

1. si $\pi \in \mathbb{P}_T$ alors retourner $p(\pi)$

2. sinon soit $\pi_1, \ldots, \pi_n$ la liste des successeurs de π

3. si $\lambda(\pi) = \text{Player}$ alors
   3.1) $v \leftarrow h_1(\pi)$
   3.2) pour i allant de 1 à n
   et tant que ($v \uparrow \alpha \not\leq_D \beta$) faire
   $v \leftarrow v \uparrow$ Alpha-Bêta($\pi_i, \alpha \uparrow v, \beta$)

4. si $\lambda(\pi) = \text{Opponent}$ alors
   4.1) $v \leftarrow h_2(\pi)$
   4.2) pour i allant de 1 à n
   et tant que ($v \downarrow \beta \not\leq_D \alpha$) faire
   $v \leftarrow v \downarrow$ Alpha-Bêta($\pi_i, \alpha, \beta \downarrow v$)

5. retourner v

de valeurs *exactes* des positions du jeu. Pour ne pas entremeler les deux types de valeurs, une solution équivalente, que nous retiendrons, consiste, d'une part, à initialiser les valeurs inachevées par un symbole ι, representant l'indéfini, et, d'autre part, à vérifier la condition de coupure tenant compte de la valeur heuristique de la position. On obtient alors l'algorithme 6, où l'écriture $\iota \uparrow x$ dénote, tout comme l'écriture $\iota \downarrow x$, la valeur x :

$$\iota \uparrow x \;=\; x \uparrow \iota \;=\; \iota \downarrow x \;=\; x \downarrow \iota \;=\; x$$

Nous noterons D_ι l'ensemble des valeurs augmenté du symbole indéfini :

$$D_\iota \triangleq D \oplus \{\iota\}$$

6.3.1. Généralisation du test de coupure. Les conditions de coupure ($v \uparrow \alpha \uparrow h_1(\pi) \leq_D \beta$) dans les noeuds du joueur, et ($v \downarrow \beta \downarrow h_2(\pi) \leq_D \alpha$) dans les noeuds de l'opposant, peuvent être remplacées respectivement par :

– dans les noeuds du joueur

$$\alpha \uparrow \beta \;=\; \alpha \uparrow [\beta \downarrow (v \uparrow h_1(\pi))]$$

– dans les noeuds de l'opposant

$$\beta \downarrow \alpha \;=\; \beta \downarrow [\alpha \uparrow (v \downarrow h_2(\pi))]$$

On vérifiera facilement par la suite que, pour une structure $\alpha\beta$-ordinaire, les anciens tests de coupure impliquent les nouveaux par effet des axiomes (A7), (A8) et (A9). Il s'agit donc de conditions plus "faciles" qui ne peuvent qu'augmenter les possibilités d'élagage. De plus, l'intérêt d'une telle modification concerne aussi les structures

Algorithm 6 *Alpha-Bêta polymorphe (sans initialisations)*

```
fonction Alpha-Bêta(π : ℙ, α,β : D) : D

  1. si π ∈ ℙ_T alors retourner p(π)

  2. sinon soit π_1,...,π_n la liste des successeurs de π

  3. si λ(π) = Player alors
       3.1) v ← ι
       3.2) pour i allant de 1 à n
              et tant que (v ↑ α ↑ h_1(π) ≰_D β) faire
              v ← v ↑ Alpha-Bêta(π_i,α↑v,β)

  4. si λ(π) = Opponent alors
       4.1) v ← ι
       4.2) pour i allant de 1 à n
              et tant que (v ↓ β ↓ h_2(π) ≰_D α) faire
              v ← v ↓ Alpha-Bêta(π_i,α,β↓v)

  5. retourner v
```

qui, n'étant pas $\alpha\beta$-ordinaires, ont toutefois des propriétés suffisantes pour garantir la correction des coupures. Nous remarquerons, en effet, que les nouveaux tests, qui sont des équations, permettent de s'affranchir de la relation sur les valeurs du jeu $\leq_D$ qui, pour autant, ne devra pas avoir de propriétés particulières outre celle de garantir la définition de la sémantique des arbres de jeu infinis. Le *prédicat de coupure* $\mathrm{cut} : \mathbb{P} \times D_\iota \times D_\iota \times D_\iota \to 2$, exprimant à la fois la condition de coupure dans les noeuds du joueur et la condition de coupure dans les noeuds de l'opposant, est défini de la façon suivante :

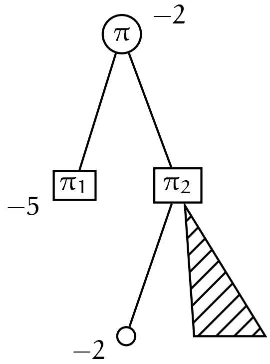
$$\mathrm{cut}(\pi,\alpha,\beta,x) \triangleq \begin{cases} 1 \text{ si} & \alpha \uparrow \beta = \alpha \uparrow [\beta \downarrow (x \uparrow h_1(\pi))] \neq \iota \text{ et } \lambda(\pi) = \text{Player} \\ 1 \text{ si} & \beta \downarrow \alpha = \beta \downarrow [\alpha \uparrow (x \downarrow h_2(\pi))] \neq \iota \text{ et } \lambda(\pi) = \text{Opponent} \\ 0 \text{ sinon} & \end{cases}$$

6.3.2. Le calcul de valeurs abstraites. Dans l'optique d'agrandir le plus possible le spectre d'application de l'algorithme, nous pouvons aussi remettre en question la notion de correction du calcul, qui vise l'*égalité* des valeurs. Dans une structure $\alpha\beta$-ordinaire, si la relation de préordre $\leq_D$ n'est pas un ordre partiel mais un préordre propre, alors l'équivalence engendrée ne coïncide pas avec l'égalité sur le domaine. On peut alors demander que les équations qui doivent être respectées, comme par exemple la propriété distributive, soient reécrites en termes de l'équivalence et non de l'égalité :

$$\begin{cases} x \overset{2}{\downarrow} (y \overset{2}{\uparrow} z) & \approx & (x \overset{2}{\downarrow} y) \overset{2}{\uparrow} (x \overset{2}{\downarrow} z) \\ \\ x \overset{2}{\uparrow} (y \overset{2}{\downarrow} z) & \approx & (x \overset{2}{\uparrow} y) \overset{2}{\downarrow} (x \overset{2}{\uparrow} z) \end{cases}$$

Cela permet d'obtenir un résultat de correction de l'algorithme encore plus général. En effet, l'égalité n'est qu'un *cas particulier* de l'équivalence engendrée par

FIG. 6.3.1. Coupures $\alpha\beta$ correctes dans le domaine abstrait



le préordre, lorsque ce dernier est un ordre partiel. On peut expliquer l'effet de ce relachement d'hypothèses en affirmant que par l'algorithme Alpha-Bêta nous pouvons calculer, à la manière de l'Interprétation Abstraite de programmes (cf. [**Cousot & Cousot, 1977**]), des valeurs *abstraites*, c'est-à-dire des valeurs qui sont exactes modulo une fonction d'abstraction. Par rapport aux techniques propres de l'Interprétation Abstraite, dans le cas d'Alpha-Bêta, les valeurs manipulées seront toujours des valeurs *concrètes*, mais les simplification réalisées, c'est-à-dire les coupures, seront, elles, justifiées dans le domaine abstrait. Autrement dit, l'algorithme sera correct par rapport à l'équivalence définie. Dans le cas particulier où l'équivalence sera l'égalité des valeurs, l'algorithme calculera exactement la valeur d'une position. Sinon, il calculera une valeur équivalente, c'est-à-dire égale dans le domaine implicite des valeurs abstraites.

EXAMPLE 6.3.1. Considérons le cas de la figure 6.3.1. Le jeu est interprété dans le domaine des entiers relatifs $\mathbb{Z}$ avec les fonctions habituelles min et max. On peut penser, par exemple, que le gain soit de l'argent et que n'importe quelle somme peut être gagnée ou perdue par les joueurs au cours d'une partie. Dans la position π de ce jeu, le joueur a un premier coup conduisant à une position π_1 dont la valeur est l'entier négatif -5, et un second coup conduisant à la position π_2 de l'opposant. Dans cette position, l'opposant a un premier coup dont la valeur est -2. Cette information n'est certainement pas suffisante pour terminer l'évaluation de la position π_2. En effet, le joueur peut encore espérer que les autres coups de l'opposant ne soient pas pire que son premier ou, tout au moins, qu'ils ne soient pas pire que -5. Dans ce cas la perte correspondante à son second coup serait, pour le joueur, bien meilleure comparée à la perte -5 de son premier coup. Autrement dit, la coupure n'est pas justifiée dans le domaine (concret) des valeurs utilisées : pour évaluer correctement la position à la racine il est indispensable d'évaluer les autres coups de l'opposant dans la position π_2. En revanche, si on se pose la question de savoir qui gagne la partie, faisant *abstraction* de la quantité gagnée ou perdue, la coupure devient correcte. En effet, nous savons que dans la position π_2 l'opposant choisira un coup conduisant, de toute façon, à une défaite du joueur. Nous pourrons donc interrompre l'évaluation et répondre -2, valeur à laquelle nous donnerons le sens (abstrait) de *perdu*. □

DEFINITION 6.3.2. (INTERPRÉTATION ABSTRAITE D'UN JEU) *Étant donné un jeu* $\mathcal{S} = (\mathbb{P}, \lambda, \to, \mathbb{P}_I)$, *et une structure* $\mathcal{D} = (\mathsf{D}, \leq_{\mathsf{D}}: 1^{\mathsf{D}\times\mathsf{D}}, \uparrow, \downarrow: \mathsf{D}^* \to \mathsf{D})$ *de co-évaluation monotone* $\mathcal{D} \in \mathsf{Cstruct}(\mathcal{S}, \mathsf{D}, \uparrow, \downarrow, p, h_1, h_2)$, *une* interprétation abstraite du jeu *est la donnée d'une structure* $\mathcal{D}^{\mathrm{abs}} = (\mathsf{D}^{\mathrm{abs}}, \leq_{\mathsf{D}}^{\mathrm{abs}}: 1^{\mathsf{D}^{\mathrm{abs}}\times\mathsf{D}^{\mathrm{abs}}}, \uparrow^{\mathrm{abs}}, \downarrow^{\mathrm{abs}} : (\mathsf{D}^{\mathrm{abs}})^* \to \mathsf{D}^{\mathrm{abs}})$ *et d'une fonction* $\varphi : \mathsf{D} \to \mathsf{D}^{\mathrm{abs}}$, *constituant un homomorphisme de* $\mathcal{D}$ *vers* $\mathcal{D}^{\mathrm{abs}}$. □

Par définition d'homomorphisme, les opérations abstraites sont telles que :

$$\begin{cases} x \leq_{\mathsf{D}} y & \Rightarrow & \varphi(x) \leq_{\mathsf{D}}^{\mathrm{abs}} \varphi(y) \\ \varphi(x \uparrow y) & = & \varphi(x) \uparrow^{\mathrm{abs}} \varphi(y) \\ \varphi(x \downarrow y) & = & \varphi(x) \downarrow^{\mathrm{abs}} \varphi(y) \end{cases}$$

Si les fonctions de payoff et les heuristiques sont définies sur le domaine abstrait en faisant appel aux fonctions correspondantes du domaine concret, c'est-à-dire :

$$p^{\mathrm{abs}}(\pi) \triangleq \varphi(p(\pi)) \qquad {h_1}^{\mathrm{abs}}(\pi) \triangleq \varphi(h_1(\pi)) \qquad {h_2}^{\mathrm{abs}}(\pi) \triangleq \varphi(h_2(\pi))$$

alors, on vérifie trivialement que $\mathcal{D}^{\mathrm{abs}}$ est, elle même, une structure de co-évaluation monotone :

$$\mathcal{D}^{\mathrm{abs}} \in \mathsf{Cstruct}(\mathcal{S}, \mathsf{D}^{\mathrm{abs}}, \uparrow^{\mathrm{abs}}, \downarrow^{\mathrm{abs}}, p^{\mathrm{abs}}, {h_1}^{\mathrm{abs}}(\pi), {h_2}^{\mathrm{abs}}(\pi))$$

où l'évaluation inductive des arbres finis est, elle aussi, telle que :

$${v_{p,h_1}}^{\mathrm{abs}}(t) = \varphi(v_{p,h_1}(t)) \qquad {v_{p,h_2}}^{\mathrm{abs}}(t) = \varphi(v_{p,h_2}(t))$$

Munis d'une interprétation abstraite d'un jeu, nous pouvons définir l'équivalence entre valeur par le bias de l'égalité après abstraction :

$$x \approx y \overset{\triangle}{\iff} \varphi(x) = \varphi(y)$$

Il s'agira, par la définition d'homomorphisme, d'une *congruence*, i.e. :

$$x \approx y \quad \Rightarrow \quad \begin{cases} x \uparrow z & \approx & y \uparrow z \\ x \downarrow z & \approx & y \downarrow z \end{cases}$$

6.3.2.1. *Le prédicat de coupure généralisé.* Nous définissons une nouvelle version du prédicat de coupure cut : $\mathbb{P} \times \mathsf{D}_\iota \times \mathsf{D}_\iota \times \mathsf{D}_\iota \times \wp^{\mathsf{D}\times\mathsf{D}} \to 2$ tenant compte d'une relation d'équivalence sur D (la pseudo-valeur ι est donc forcément exclue du test de coupure) :

$$\mathrm{cut}(\pi,\alpha,\beta,x,\approx) \triangleq \begin{cases} 1 \text{ si} & \alpha \uparrow \beta \approx \alpha \uparrow [\beta \downarrow (x \uparrow h_1(\pi))] \text{ et } \lambda(\pi) = \text{Player} \\ 1 \text{ si} & \beta \downarrow \alpha \approx \beta \downarrow [\alpha \uparrow (x \downarrow h_2(\pi))] \text{ et } \lambda(\pi) = \text{Opponent} \\ 0 \text{ sinon} & \end{cases}$$

6.3.3. Formalisation de la méthode. Avant de prouver la correction de la méthode, nous allons décrire l'algorithme d'une façon plus rigoureuse par rapport à sa formulation habituelle dans un pseudo-langage impératif. Nous utiliserons le style de définition des sémantiques à grands pas (*big steps*), qui nous permettra de développer la preuve de façon plus formelle. Pour l'évaluation Alpha-Bêta d'une position π nous utiliserons la notation :

$$\llbracket \pi \rrbracket^{\alpha,\beta}_{\mathcal{D}\approx}$$

qui évoque la sémantique bisémique des positions si l'on oublie les paramètres α et β. Cela ne tient pas au hasard et nous étudierons par la suite le rapport entre les deux notions.

6.3.3.1. *Arbres amputés du jeu.* La principale caractéristique de l'algorithme Alpha-Bêta est celle de calculer la valeur de certaines branches de l'arbre de jeu en utilisant une sorte de résumé essentiel, les valeurs α et β, des branches visitées auparavant. Pour expliquer d'une façon plus formelle le passage de l'information d'une branche à l'autre, nous introduisons une notion, celle d'arbre *amputé*, qui nous servira également pour la preuve de correction de l'algorithme.

DEFINITION 6.3.3. (ARBRES AMPUTÉS DU JEU) *Étant donné un jeu* $\mathcal{S} = (\mathbb{P}, \lambda, \rightarrow, \mathbb{P}_{\mathrm{I}})$*, un arbre* $\mathrm{t} \in \mathrm{IT}(\mathbb{P})$*, étiqueté dans* $\mathbb{P}$*, est un arbre* amputé *(ou* partiel*) du jeu s'il existe un arbre de jeu* $\mathrm{t}_\pi \in \mathrm{IT}(\mathcal{S})$ *tel que* $\mathrm{t} = \mathrm{t}_\pi = \{(\varepsilon, \pi)\}$ *ou bien* t *est un élagage de* t_π *partiel mais pas radical à l'adresse* ε*, et complet à toute autre adresse. L'ensemble de tous les arbres amputés du jeu sera noté* $\mathrm{IT}(\mathcal{S})^\partial$*, celui des arbres amputés d'une position* π *sera noté* $\mathrm{IT}(\mathcal{S},\pi)^\partial$*. L'ensemble des préfixes finis (non radicaux) d'arbres amputé d'une position du jeu sera noté* $\widehat{\mathrm{T}}(\mathcal{S},\pi)^\partial$ *:*

$$\widehat{\mathrm{T}}(\mathcal{S},\pi)^\partial = \{\mathrm{t}' \in \mathrm{T}(\mathbb{P}) \mid \mathrm{t}' \leq \mathrm{t}\,,\ \mathrm{t}' \in \mathrm{IT}(\mathcal{S},\pi)^\partial,\ \mathrm{t}' \neq \{(\varepsilon,\pi)\}\}$$

□

En d'autres termes, un arbre amputé du jeu t est un arbre de jeu si la position à la racine est terminale. Sinon un arbre amputé du jeu est l'élagage d'un véritable arbre de jeu t_π, tel que tous les fils t_i sont, eux aussi, des véritables arbres de jeu. Par définition, si la position $\pi \in \mathbb{P}_{\mathrm{NT}}$, un arbre amputé du jeu $\mathrm{t} \in \mathrm{IT}(\mathcal{S},\pi)^\partial$ ou un de ses préfixes $\mathrm{t}' \in \widehat{\mathrm{T}}(\mathcal{S},\pi)^\partial$, aura toujours au moins un fils (l'élagage ne peut être radical, ni à la racine ni ailleurs).

DEFINITION 6.3.4. (COMPOSITION D'ARBRES AMPUTÉS) *Étant donné un jeu* $\mathcal{S} = (\mathbb{P}, \lambda, \rightarrow, \mathbb{P}_{\mathrm{I}})$ *et une position non terminale* $\pi \in \mathbb{P}_{\mathrm{NT}}$*, un arbre* $\mathrm{t} \in \mathrm{IT}(\mathcal{S},\pi)^\partial$ *est la* composition *de deux arbres amputés, notée* $\mathrm{t} = \mathrm{t}_1 \bowtie \mathrm{t}_2$*, si* $\mathrm{t} = \mathrm{t}_1 \cup \mathrm{t}_2$ *et* $\mathrm{t}_1 \cap \mathrm{t}_2 = \{(\varepsilon,\pi)\}$. □

La relation de composition $\bowtie$ est une fonction partielle de type $\mathrm{IT}(\mathcal{S},\pi)^\partial \times \mathrm{IT}(\mathcal{S},\pi)^\partial \rightharpoonup \mathrm{IT}(\mathcal{S},\pi)^\partial$, qui est trivialement associative et commutative. Tout arbre de jeu ayant au moins deux fils pourra être considéré comme la composition de deux arbres amputés du jeu. Une fois choisis les deux arbres amputés t_1 et t_2 composant un arbre $\mathrm{t} = \mathrm{t}_1 \bowtie \mathrm{t}_2$, nous pourrons débuter le calcul sur le premier arbre t_1, et le continuer sur le second t_2 avec l'information α ou β (selon le type de noeud) synthétisée en visitant le premier. La composition de deux arbres disjoints correspond, après évaluation, à la construction de deux valeurs que nous dirons *indépendantes.*

DEFINITION 6.3.5. (VALEURS D'UN ARBRE AMPUTÉ) *Étant donné un jeu* $\mathcal{S} = (\mathbb{P}, \lambda, \rightarrow, \mathbb{P}_I)$, *une structure* $\mathcal{D} = (D, \leq_D, \uparrow, \downarrow)$ *de co-évaluation* $\mathcal{D} \in \mathsf{Cstruct}(\mathcal{S}, D, \uparrow, \downarrow, p, h_1, h_2)$, *et un arbre amputé d'une position* $t \in \mathrm{IT}(\mathcal{S}, \pi)^\partial$, *l'ensemble des valeurs de* t, *noté* $D_\pi(t)$, *est l'ensemble des évaluations pessimiste ou optimiste d'un quelconque élagage de* t *:*

$$D_\pi(t) \triangleq \{v_{p,h}(\tilde{t}) \mid \tilde{t} \subseteq t\,,\ \tilde{t} \in \widetilde{\mathrm{T}}(\mathcal{S}, \pi)\,,\ h \in \{h_1, h_2\}\}$$

□

L'ensemble des valeurs d'un arbre amputé est donc une application de type $D_\pi(.) : \mathrm{IT}(\mathcal{S}, \pi)^\partial \rightarrow D$.

6.3.3.2. *Valeurs des élagages.* Les arbres amputés sont des cas particuliers d'élagages de l'arbre d'une position. L'algorithme Alpha-Bêta construit implicitement un élagage dont la valeur coïncide, dans un sens que nous préciserons par la suite, avec la valeur de la position.

DEFINITION 6.3.6. (VALEURS DES ÉLAGAGES) *L'ensemble des* valeurs des élagages *d'une position* π *est l'ensemble des évaluations pessimiste ou optimiste d'un quelconque élagage de* t_π *:*

$$E_\pi \triangleq \{v_{p,h}(\tilde{t}) \mid \tilde{t} \in \widetilde{\mathrm{T}}(\mathcal{S}, \pi)\,,\ h \in \{h_1, h_2\}\}$$

□

Durant l'execution de l'algorithme Alpha-Bêta, nous pouvons affirmer que le calcul d'une valeur x, qui est la valeur d'un élagage, permet, dans certains cas, d'éviter le calcul d'une autre valeur de l'ensemble E_π.

6.3.3.3. *Le système d'inférence Alpha-Bêta.* Étant donné une arène de jeu $\mathcal{S} = (\mathbb{P}, \lambda, \rightarrow, \mathbb{P}_I)$ et une structure de co-évaluation $\mathcal{D} = (D, \leq_D, \uparrow, \downarrow)$, $\mathcal{D} \in \mathsf{Cstruct}(\mathcal{S}, D, \uparrow, \downarrow, p, h_1, h_2)$, les règles d'inférence permettant de déduire (c'est-à-dire de calculer) que l'évaluation Alpha-Bêta *converge* vers une valeur $v \in D$, notée $[\![\, t \,]\!]^{\alpha,\beta,x}_{\mathcal{D}_\approx} \Downarrow v$, sont regroupées dans la table 2, où t est un arbre amputé du jeu et les trois autres paramètres sont des valeurs de D ou bien le symbole indéfini ι :

$$t \in \mathrm{IT}(\mathcal{S}, \pi)^\partial \qquad \text{et} \qquad \alpha, \beta, x \in D_\iota$$

La relation exprimant le calcul Alpha-Bêta, noté $[\![\, \pi \,]\!]^{\alpha,\beta}_{\mathcal{D}_\approx} \Downarrow v$, sur une position du jeu donnée, est définie par :

$$[\![\, \pi \,]\!]^{\alpha,\beta}_{\mathcal{D}_\approx} \Downarrow v \quad \stackrel{\triangle}{\Longleftrightarrow} \quad [\![\, t_\pi \,]\!]^{\alpha,\beta,\iota}_{\mathcal{D}_\approx} \Downarrow v$$

TAB. 2. Algorithme Alpha-Bêta polymorphe (*à grands pas*)

Pour un jeu $\mathcal{S} = (\mathbb{P}, \lambda, \rightarrow, \mathbb{P}_{\mathrm{I}})$
Interprété dans $\mathcal{D} \in \mathsf{Cstruct}(\mathcal{S}, D, \uparrow, \downarrow, p, h_1, h_2)$

$$\frac{t = \{(\varepsilon, \pi)\} \quad \pi \in \mathbb{P}_{\mathrm{T}}}{[\![\, t \,]\!]^{\alpha,\beta,\iota}_{\mathcal{D},\approx} \Downarrow p(\pi)} \quad \text{(base)}$$

$$\frac{t = t_1 \bowtie t_2 \qquad [\![\, t_1 \,]\!]^{\alpha,\beta,x}_{\mathcal{D},\approx} \Downarrow \nu_1 \qquad [\![\, t_2 \,]\!]^{\alpha,\beta,\nu_1}_{\mathcal{D},\approx} \Downarrow \nu_2 \qquad \nu = \nu_2}{[\![\, t \,]\!]^{\alpha,\beta,x}_{\mathcal{D},\approx} \Downarrow \nu} \quad \text{(more)}$$

$$\frac{\begin{array}{l}\lambda(t\,\varepsilon) = \text{Player} \\ \text{Fils}(t) = (\ell_1.t_1)\end{array} \qquad [\![\, t_1 \,]\!]^{(\alpha\uparrow x),\beta,\iota}_{\mathcal{D},\approx} \Downarrow \nu_1 \qquad \nu = x \uparrow \nu_1}{[\![\, t \,]\!]^{\alpha,\beta,x}_{\mathcal{D},\approx} \Downarrow \nu} \quad \text{(one)}$$

$$\frac{\begin{array}{l}\lambda(t\,\varepsilon) = \text{Opponent} \\ \text{Fils}(t) = (\ell_1.t_1)\end{array} \qquad [\![\, t_1 \,]\!]^{\alpha,(\beta\downarrow x),\iota}_{\mathcal{D},\approx} \Downarrow \nu_1 \qquad \nu = x \downarrow \nu_1}{[\![\, t \,]\!]^{\alpha,\beta,x}_{\mathcal{D},\approx} \Downarrow \nu} \quad \text{(one)}$$

$$\frac{\mathrm{cut}(\pi, \alpha, \beta, x, \approx)}{[\![\, t \,]\!]^{\alpha,\beta,x}_{\mathcal{D},\approx} \Downarrow x} \quad \text{(cut)}$$

Nous simplifierons la notation en écrivant $[\![\, \pi \,]\!]^{\alpha,\beta,x}_{\mathcal{D}} \Downarrow \nu$ et $[\![\, t \,]\!]^{\alpha,\beta,x}_{\mathcal{D}} \Downarrow \nu$, lorsqu'il n'y aura pas d'ambiguité possible sur la relation d'équivalence utilisée dans le calcul. En particulier, nous utiliserons cette notation lorsque l'équivalence coincidera avec l'égalité sur le domaine.

Le système d'inférence est composé de quatre types de règles, une règle (base), une règle (cut) et une règle (more), valident à la fois pour les noeuds du joueur et pour ceux de l'opposant, et un dernier type (one) composé d'une règle applicable aux noeuds du joueur et d'une seconde applicable aux noeuds de l'opposant.

(base) est la règle pour les positions terminales, à la *base* du procédé récursif. Si l'arbre de jeu est composé de la seule racine, étiquetée par $\pi \in \mathbb{P}_{\mathrm{T}}$, nous pouvons conclure que l'évaluation est égale au payoff de cette position (indépendamment des paramètres α et β).

(more) est la règle qui réalise la boucle d'évaluation pour un arbre t ayant *plusieurs* fils. En effet, tant que l'arbre possède au moins deux fils, il peut être décomposé en deux arbres amputés du jeu. Une fois l'évaluation du premier terminée, la valeur de retour ν_1 pourra être utilisée pour mettre à jour la valeur inachevée du père ($x' = \nu_1$), et pour aborder ensuite l'évaluation du second. Cette règle exprime le non-déterminisme de l'algorithme dans le choix des branches à visiter. Pour n fils du noeud concerné, il existe $n!$ façons possibles d'ordonnancer les branches pour procéder à leur visite séquentielle. En divisant l'ensemble des fils en deux groupes, correspondants aux deux arbres amputés, et en répétant ce geste pour chaque groupe, la règle permet indirectement de définir la séquence d'évaluation des fils du noeud impliqué.

(one) sont les règles d'évaluation d'un arbre avec un *seul* fils. Par leur dualité, nous décrivons seulement la règle applicable aux noeuds du joueur. L'évaluation de cet unique fils aura lieu sur une fenêtre $[\alpha', \beta]$ différente par rapport à la fenêtre $[\alpha, \beta]$ en cours d'utilisation pour le père. En effet, puisque x est la valeur inachevée du père, et que le père lui-même deviendra un ancêtre aussitôt que nous démarrerons l'évaluation du fils, le total des valeurs inachevées des ancêtres (du joueur) sera, pour le fils, $\alpha' = \alpha \uparrow x$. La valeur inachevée du fils est indéfinie au départ de la boucle d'évaluation, donc affectée à ι. Une fois l'évaluation de son unique fils terminée, celle du père aussi peut aboutir. La valeur de retour du père est alors égale à $x \uparrow v'$, c'est-à-dire à la valeur jusque-là inachevée du père, accomplie par la valeur du dernier fils.

(cut) est la règle de *rupture* d'une boucle d'évaluation. Lorsque la condition de coupure est vérifiée, le noeud en évaluation ne pourra conditionner la valeur de l'un de ses ancêtres. L'exécution interrompue, le noeud peut être évalué à la valeur inachevée x, de toute façon non influente dans l'actuel contexte d'évaluation.

Nous remarquerons que le premier et principal argument de relation définie par le système d'inférence, l'arbre t, peut ne pas être un véritable arbre de jeu. En effet, la règle (more) décrète que l'évaluation d'un arbre avec plusieurs fils, devra passer par l'évaluation d'un premier groupe de fils, puis par l'évaluation d'un second groupe. Si au départ nous avions un arbre de jeu, après cette décomposition nous n'aurons plus un véritable arbre de jeu. La décomposition permet de construire deux arbres amputés du jeu à partir d'un arbre de jeu ou, plus généralement, à partir d'un arbre amputé du jeu.

6.3.3.4. *Lemme de caractérisation.* Une remarque importante sur le comportement de l'algorithme est que, pour une position initiale π donnée, si l'algorithme termine alors la valeur retournée appartiennent à l'ensemble $D_\pi(t_\pi)$ des valeurs de l'arbre de la position (qui est un cas particulier d'arbre amputé). En d'autres termes, l'algorithme renvoie une valeur qui correspond à l'évaluation d'un élagage de l'arbre de la position donnée.

LEMMA 6.3.7. (CARACTÉRISATION) *Si* $[\![\, t \,]\!]^{\alpha,\beta,x}_{\mathcal{D},\approx} \Downarrow v$ *alors :*

$$\begin{array}{lccl} (1) & x = \iota & \Rightarrow & v \in D_\pi(t) \oplus \{\iota\} \\ (2) & \exists t'' = t' \bowtie t\,,\ x \in D_\pi(t') & \Rightarrow & v \in D_\pi(t'') \end{array}$$

DÉMONSTRATION. *Par induction sur la longueur de la preuve de* $[\![\, t \,]\!]^{\alpha,\beta,x}_{\mathcal{D},\approx} \Downarrow v$. *(1) Supposons* $x = \iota$. *Pour la preuve de* $[\![\, t \,]\!]^{\alpha,\beta,x}_{\mathcal{D},\approx} \Downarrow v$ *la dernière règle utilisée appartient au sistème d'inférence Alpha-Bêta. Faisons donc tous les cas possibles.*

(base) *dans ce cas* $\nu = p(\pi)$, *donc* $\nu \in D_\pi(t)$.

(one) *l'évaluation du fils de* t *demarre sur le paramètre* $x' = \iota$. *Donc, par hypothèse d'induction, nous avons* $\nu_1 \in D_{\pi_1}(t_1) \oplus \{\iota\}$, *où* $\pi_1 = (t_1 \varepsilon)$. *Puisque* t_1 *est l'unique fils de* t, *nous avons* $D_{\pi_1}(t_1) \subseteq D_\pi(t)$. *Donc* $\nu = x \uparrow \nu_1 = \iota \uparrow \nu_1 = \nu_1$ *si la position est du joueur, sinon* $\nu = x \downarrow \nu_1 = \iota \downarrow \nu_1 = \nu_1$. *Dans les deux cas* $\nu = \nu_1 \in D_\pi(t)$.

(cut) *dans ce cas* $\nu = x = \iota$

(more) *par hypothèse d'induction sur* $[\![\, t_1 \,]\!]^{\alpha,\beta,x}_{\mathcal{D},\approx} \Downarrow \nu_1$ *nous avons* $\nu_1 \in D_\pi(t_1) \oplus \{\iota\}$. *Il y a deux possibilités ;*

- $\nu_1 = \iota$ *alors par hypothèse d'induction sur* $[\![\, t_2 \,]\!]^{\alpha,\beta,\nu_1}_{\mathcal{D},\approx} \Downarrow \nu_2$ *nous avons* $\nu = \nu_2 \in D_\pi(t_2) \oplus \{\iota\} \subseteq D_\pi(t) \oplus \{\iota\}$ *(puisque* $t_2 \subseteq t$*)*
- $\nu_1 \in D_\pi(t_1)$ *alors il existe* $t = t_1 \bowtie t_2$ *tel que* $x' = \nu_1 \in D_\pi(t_1)$. *Par hypothèse d'induction sur* $[\![\, t_2 \,]\!]^{\alpha,\beta,x'}_{\mathcal{D},\approx} \Downarrow \nu_2$, *nous avons* $\nu = \nu_2 \in D_\pi(t)$, c.q.f.d.

(2) Supposons que ;

$$\exists t'' = t' \bowtie t \,,\, x \in D_\pi(t')$$

Nous devons prouver que $\nu \in D_\pi(t'')$. *La position* π *est forcément non terminale. Nous faisons encore une fois tous les cas possible en ce qui concerne la dernière règle utilisée pour la preuve de* $[\![\, t \,]\!]^{\alpha,\beta,x}_{\mathcal{D},\approx} \Downarrow \nu$.

(base) *ce cas est impossible puisque* $\pi \in \mathbb{P}_{NT}$

(one) *l'évaluation du fils de* t *demarre sur le paramètre* $x' = \iota$. *Donc, par hypothèse d'induction, nous avons* $\nu_1 \in D_{\pi_1}(t_1) \oplus \{\iota\}$, *où* $\pi_1 = (t_1 \varepsilon)$. *Puisque* t_1 *est l'unique fils de* t, *nous avons* $D_{\pi_1}(t_1) \subseteq D_\pi(t)$. *Si* $\nu_1 = \iota$ *nous avons* $\nu = x \uparrow \nu_1$ *si la position est du joueur, sinon* $\nu = x \downarrow \nu_1$. *Dans les deux cas* $\nu = x \in D_\pi(t') \subseteq D_\pi(t'')$. *En revanche, si* $\nu_1 \in D_{\pi_1}(t_1)$ *alors, d'une part* $\nu_1 \in D_\pi(t)$ *et, d'autre part,* $x \in D_\pi(t')$. *Donc* $\nu = x \uparrow \nu_1$ *si le noeud est du joueur, sinon* $\nu = x \downarrow \nu_1$. *Dans les deux cas* $\nu \in D_\pi(t' \bowtie t) = D_\pi(t'')$.

(cut) *dans ce cas* $\nu = x \in D_\pi(t') \subseteq D_\pi(t'')$

(more) *l'évaluation de* t_1 *se fait sur le même paramêtre* x. *Il existe* $t_1'' = t' \bowtie t_1$ *tel que* $x \in D_\pi(t')$. *Alors, par hypothèse d'induction sur* $[\![\, t_1 \,]\!]^{\alpha,\beta,x}_{\mathcal{D},\approx} \Downarrow \nu_1$ *nous avons* $\nu_1 \in D_\pi(t_1'')$. *En revanche, l'évaluation de* t_2 *se fait sur le paramêtre* $x' = \nu_1$. *Par hypothèse nous savons qu'il existe* t'' *tel que :*

$$t'' = t' \bowtie t = t' \bowtie (t_1 \bowtie t_2) = (t' \bowtie t_1) \bowtie t_2 = t_1'' \bowtie t_2$$

Puisque $x' \in D_\pi(t_1'')$, *nous pouvons appliquer l'hypothèse d'induction sur* t_2, *donc* $\nu = \nu_2 \in D_\pi(t'')$, c.q.f.d.

$\square$

TAB. 3. Axiomes des structures $\alpha\beta$-normales

$$\mathcal{D} \in \mathsf{Cstruct}(D, \uparrow, \downarrow, p, h_1, h_2) \qquad \mathcal{D} = (D, \leq_D, \uparrow, \downarrow)$$

(1) les fonctions des joueurs sont générées par un noyau binaire, $\downarrow = \mathsf{ext}(\overset{2}{\downarrow})$ et $\uparrow = \mathsf{ext}(\overset{2}{\uparrow})$:

(A1)	$x \overset{2}{\downarrow} (y \overset{2}{\downarrow} z) = (x \overset{2}{\downarrow} y) \overset{2}{\downarrow} z$	$x \overset{2}{\uparrow} (y \overset{2}{\uparrow} z) = (x \overset{2}{\uparrow} y) \overset{2}{\uparrow} z$	(associative)
(A2)	$x \overset{2}{\downarrow} y = y \overset{2}{\downarrow} x$	$x \overset{2}{\uparrow} y = y \overset{2}{\uparrow} x$	(commutative)
(A3)	$x \overset{2}{\downarrow} x = x$	$x \overset{2}{\uparrow} x = x$	(identité)
(A4)	$\downarrow x = x \overset{2}{\downarrow} x$	$\uparrow x = x \overset{2}{\uparrow} x$	(extension)
(A5)	$\downarrow x.u = x \overset{2}{\downarrow} (\downarrow u)$	$\uparrow x.u = x \overset{2}{\uparrow} (\uparrow u)$	(extension)

(2) il existe une famille de relations d'équivalence $\{\approx_i\}_{i \in I}$ qui est une famille congruente, i.e. :

$$\text{(A6)} \quad \left\{ \begin{array}{c} \dfrac{x \approx_j x' \qquad x \uparrow y \approx_i z}{x' \uparrow y \approx_i z} \\[2ex] \dfrac{x \approx_j x' \qquad x \downarrow y \approx_i z}{x' \downarrow y \approx_i z} \end{array} \right.$$

(3) les axiomes de correction des coupures sont vérifiés :

$$\text{(A789p)} \quad \frac{\lambda(\pi) = \text{Player} \qquad \text{cut}(\pi, \alpha, \beta, x, \approx_i) \qquad \alpha, \beta \in D \oplus \{\iota\} \qquad x, y \in E_\pi \oplus \{\iota\}}{\alpha \uparrow (\beta \downarrow x) \ \approx_i \ \alpha \uparrow [\beta \downarrow (x \uparrow y)]}$$

$$\text{(A789o)} \quad \frac{\lambda(\pi) = \text{Opponent} \qquad \text{cut}(\pi, \alpha, \beta, x, \approx_i) \qquad \alpha, \beta \in D \oplus \{\iota\} \qquad x, y \in E_\pi \oplus \{\iota\}}{\beta \downarrow (\alpha \uparrow x) \ \approx_i \ \beta \downarrow [\alpha \uparrow (x \downarrow y)]}$$

(4) les fonctions des joueurs sont distributives :

$$\text{(A10)} \quad \left\{ \begin{array}{lcl} x \overset{2}{\downarrow} (y \overset{2}{\uparrow} z) & \approx_i & (x \overset{2}{\downarrow} y) \overset{2}{\uparrow} (x \overset{2}{\downarrow} z) \\ x \overset{2}{\uparrow} (y \overset{2}{\downarrow} z) & \approx_i & (x \overset{2}{\uparrow} y) \overset{2}{\downarrow} (x \overset{2}{\uparrow} z) \end{array} \right.$$

6.3.4. Structures $\alpha\beta$-normales. Les conditions requises à une structure $\alpha\beta$-ordinaire se simplifient de façon appréciable lorsque les opérations unaires des deux joueurs coïncident, comme dans le cas des ordres partiels $\alpha\beta$-ordinaires, avec la fonction identité ($\overset{1}{\downarrow} x = \overset{1}{\uparrow} x = x$). La table 3 résume l'ensemble des conditions requises aux structures de co-évaluation $\alpha\beta$-*normales*, où les opérations unaires des joueurs sont l'identité et les conditions d'élagage portent sur des équations et non pas sur la relation $\leq_D$ comme dans le cadre des structures $\alpha\beta$-ordinaires. La numérotation des axiomes sera utilisée dans la preuve du théorème de correction Alpha-Bêta, où l'on supposera que la structure de co-évaluation du jeu soit $\alpha\beta$-normale. Nous disons que la structure est $\alpha\beta$-*normale du joueur* si elle ne vérifie pas forcément l'axiome

(A789o) mais elle vérifie tous les autres. Vice-versa, nous disons que la structure est $\alpha\beta$-*normale de l'opposant* si elle ne vérifie pas forcément l'axiome (A789p) mais elle vérifie tous les autres. Les structures $\alpha\beta$-normales de l'opposant ou du joueur qui ne sont pas $\alpha\beta$-normales seront qualifiées d'*asymétriques*.

Le lemme suivant peut s'interpréter en affirmant que les axiomes (A789p) et (A789o) regroupent dans une seule condition les axiomes (A7), (A8) et (A9) des structures $\alpha\beta$-ordinaires.

LEMMA 6.3.8. *Toute structure $\alpha\beta$-ordinaire telle que la version unaire des opérations des joueurs est l'identité, $\overset{1}{\downarrow}\ x = \overset{1}{\uparrow}\ x = x$, est $\alpha\beta$-normale, si l'on considère la famille constituée de la seule équivalence $\approx$ engendrée par le préordre ($\{\approx_i\}_{i\in I} = \{\approx\}$) et si les heuristiques sont telles que :*

$$\forall\pi' \in \mathrm{Succ}(\pi).\quad h_1(\pi) \leq_D h_1(\pi') \ \wedge\ h_2(\pi) \geq_D h_2(\pi')$$

DÉMONSTRATION. *Il suffit de prouver les axiomes (A789p) et (A789o). Nous prouvons le premier, le second étant dual. Nous voulons donc prouver que :*

$$\frac{\lambda(\pi) = \text{Player} \qquad \mathrm{cut}(\pi,\alpha,\beta,x,\approx) \qquad x \bowtie y}{\alpha\uparrow(\beta\downarrow x)\ \approx\ \alpha\uparrow[\beta\downarrow(x\uparrow y)]}$$

c'est-à-dire :

$$\frac{\alpha\uparrow\beta \approx \alpha\uparrow[\beta\downarrow(x\uparrow h_1(\pi))] \qquad x \bowtie y}{\alpha\uparrow(\beta\downarrow x)\ \approx\ \alpha\uparrow[\beta\downarrow(x\uparrow y)]}$$

Par monotonie des opérations, l'heuristique pessimiste fournit une valeur plus petite que l'évaluation de n'importe quel élagage de l'arbre de la position. Donc $x \bowtie y$ implique $y \in D_\pi(t)$ pour un certain $t \in \mathrm{IT}(\mathcal{S},\pi)^\partial$, donc $(x\uparrow h_1(\pi)) \leq_D (x\uparrow y)$. Alors, d'une part :

$$\alpha\uparrow(\beta\downarrow x)\ \leq_D\ \alpha\uparrow[\beta\downarrow(x\uparrow y)]$$

par les axiomes (A7),(A8) et la monotonie de $\downarrow$. D'autre part :

$$\alpha\uparrow[\beta\downarrow(x\uparrow y)]\ \leq_D\ \alpha\uparrow\beta\ \approx\ \alpha\uparrow[\beta\downarrow(x\uparrow h_1(\pi))]\ \leq_D\ \alpha\uparrow[\beta\downarrow(x\uparrow y)]$$

par (A7),(A8) et la monotonie de $\uparrow$, par l'hypothèse $\mathrm{cut}(\pi,\alpha,\beta,x,\approx)$, et par la condition $(x\uparrow h_1(\pi)) \leq_D (x\uparrow y)$. Donc les trois valeurs sont équivalentes. Or, il y a deux seules possibilités. Si $x = \iota$ alors $\alpha\uparrow(\beta\downarrow x) = \alpha\uparrow\beta$ donc l'énoncé. En revanche, si $x \neq \iota$ alors $x \in D_\pi(t')$ pour un certain $t' \in \mathrm{IT}(\mathcal{S},\pi)^\partial$, donc $h_1(\pi) \leq_D x$. Alors, par l'axiome (A3) et la monotonie des opérations, nous obtenons :

$$\alpha\uparrow(\beta\downarrow x)\ =\ \alpha\uparrow(\beta\downarrow(x\uparrow x))\ \geq_D\ \alpha\uparrow[\beta\downarrow(x\uparrow h_1(\pi))]\ \approx\ \alpha\uparrow[\beta\downarrow(x\uparrow y)]$$

Puisque l'équivalence est engendrée par le préordre, $a \leq_D b$ et $b \leq_D a$ implique $a \approx b$. Donc $\alpha\uparrow(\beta\downarrow x)\ \approx\ \alpha\uparrow[\beta\downarrow(x\uparrow y)]$, c.q.f.d. □

6.3.4.1. *Lemme d'insertion.* Le lemme suivant énonce une propriété essentielle des structures $\alpha\beta$-normales, principale conséquence de la conjonction de la propriété distributive (A10) et de l'axiome (A3).

LEMMA 6.3.9. (INSERTION) *Soit* $\mathcal{D} = (D, \leq_D, \uparrow, \downarrow, \perp)$ *une structure de co-évaluation* $\alpha\beta$*-normale. Alors :*

$$\forall \alpha, \beta, x, i. \quad \begin{cases} \alpha \uparrow (\beta \downarrow x) & \approx_i & \alpha \uparrow [\beta \downarrow (\alpha \uparrow x)] \\ \beta \downarrow (\alpha \uparrow x) & \approx_i & \beta \downarrow [\alpha \uparrow (\beta \downarrow x)] \end{cases}$$

DÉMONSTRATION. *Nous prouvons la première équation, la seconde étant duale.*

$$\begin{array}{rcll} \alpha \uparrow [\beta \downarrow (\alpha \uparrow x)] & \approx_i & (\alpha \uparrow \beta) \downarrow [\alpha \uparrow (\alpha \uparrow x)] & \text{(A10)} \\ & = & (\alpha \uparrow \beta) \downarrow (\alpha \uparrow x) & \text{(A1), (A3)} \\ & \approx_i & \alpha \uparrow (\beta \downarrow x) & \text{(A10)} \end{array}$$

□

Ce lemme permet une interprétation assez intuitive de l'effet de la propriété distributive sur la correction des coupures. Dans tout noeud du joueur, nous pouvons imaginer que le joueur a un coup supplémentaire dont la valeur coïncide avec l'ensemble des valeurs inachevées des niveaux précédents du même joueur. Autrement dit, grâce à la propriété distributive et à l'axiome (A3), les coupures profondes peuvent être justifiées comme si elles étaient peu profondes, si l'on considère de toujours ajouter à la valeur inachevée du noeud les valeurs α ou β (selon le type de noeud) collectées aux niveaux supérieurs.

6.3.5. Théorème de correction. Le théorème de correction *relative* de la méthode Alpha-Bêta affirme que si un arbre *amputé* du jeu est évalué à la valeur v, alors il existe un préfixe *fini* t' de cet arbre dont la valeur bisémique $\ddot{v}(t') = [a, b]$ coïncide, modulo α, β et x, avec l'intervalle de précision maximale $[v, v]$. Le sens de l'égalité *modulo* les trois valeurs α, β et x, s'explique par les équations suivantes :

$$\begin{array}{lll} \alpha \uparrow (\beta \downarrow v) \approx \alpha \uparrow (\beta \downarrow (x \uparrow a)) \approx \alpha \uparrow (\beta \downarrow (x \uparrow b)) & & (\text{si } \lambda(t\,\varepsilon) = \text{Player}) \\ \beta \downarrow (\alpha \uparrow v) \approx \beta \downarrow (\alpha \uparrow (x \downarrow a)) \approx \beta \downarrow (\alpha \uparrow (x \downarrow b)) & & (\text{si } \lambda(t\,\varepsilon) = \text{Opponent}) \end{array}$$

NOTATION 6.3.10. La notation $\ddot{y}$ pour dénoter l'intervalle $[y, y]$ nous permettra de regrouper les équations précédentes sous une forme plus concise :

$$\begin{array}{ll} \ddot{\alpha} \mathrel{\ddot{\uparrow}} (\ddot{\beta} \mathrel{\ddot{\downarrow}} \ddot{v}) \approx \ddot{\alpha} \mathrel{\ddot{\uparrow}} (\ddot{\beta} \mathrel{\ddot{\downarrow}} (\ddot{x} \mathrel{\ddot{\uparrow}} [a, b])) & (\text{si } \lambda(t\,\varepsilon) = \text{Player}) \\ \ddot{\beta} \mathrel{\ddot{\downarrow}} (\ddot{\alpha} \mathrel{\ddot{\uparrow}} \ddot{v}) \approx \ddot{\beta} \mathrel{\ddot{\downarrow}} (\ddot{\alpha} \mathrel{\ddot{\uparrow}} (\ddot{x} \mathrel{\ddot{\downarrow}} [a, b])) & (\text{si } \lambda(t\,\varepsilon) = \text{Opponent}) \end{array}$$

THEOREM 6.3.11. (CORRECTION RELATIVE) *Soit* $\mathcal{S} = (\mathbb{P}, \lambda, \rightarrow, \mathbb{P}_I)$ *un jeu co-interprété dans une structure* $\alpha\beta$*-normale* $\mathcal{D} \in \mathsf{Cstruct}(\mathcal{S}, D, \uparrow, \downarrow, p, h_1, h_2)$*,* $\mathcal{D} = (D, \leq_D, \uparrow, \downarrow)$*, où* $\approx$ *est la famille congruente sur les valeurs. Soient* $\alpha, \beta, x \in D_\iota$ *et soit* $t \in \mathrm{IT}(\mathcal{S}, \pi)^\partial$*. Supposons que* t *et* x *soient dans le rapport suivant :*

$$\text{(Hp)} \quad (x = \iota) \quad \vee \quad (\exists t'' = t' \bowtie t \wedge x \in D_\pi(t'))$$

Sous cette hypothèse, si la méthode converge :

$$[\![\, t \,]\!]^{\alpha,\beta,x}_{\mathcal{D}_\approx} \Downarrow \nu$$

alors il existe un préfixe fini $t' \in \widehat{T}(\mathcal{S},\pi)^\partial$, $t' \leq t$, *non radical* ($t \neq \{(\varepsilon,\pi)\}$) *si* π *est non terminale, et tel que :*

- *si* $\lambda(\pi) = \text{Player}$ *:*
$$\exists i. \quad \ddot{\alpha} \mathrel{\ddot{\uparrow}} (\ddot{\beta} \mathrel{\ddot{\downarrow}} \ddot{v}) \;\approx_i\; \ddot{\alpha} \mathrel{\ddot{\uparrow}} (\ddot{\beta} \mathrel{\ddot{\downarrow}} (\ddot{x} \mathrel{\ddot{\uparrow}} \ddot{v}(t')))$$

- *si* $\lambda(\pi) = \text{Opponent}$ *:*
$$\exists i. \quad \ddot{\beta} \mathrel{\ddot{\downarrow}} (\ddot{\alpha} \mathrel{\ddot{\uparrow}} \ddot{v}) \;\approx_i\; \ddot{\beta} \mathrel{\ddot{\downarrow}} (\ddot{\alpha} \mathrel{\ddot{\uparrow}} (\ddot{x} \mathrel{\ddot{\downarrow}} \ddot{v}(t')))$$

DÉMONSTRATION. *Par induction sur la longueur de la preuve de* $[\![\, t \,]\!]^{\alpha,\beta,x}_{\mathcal{D}_\approx} \Downarrow \nu$. *Nous procédons par cas sur la dernière règle utilisée, en supposant toujours* $\lambda(\pi) = \text{Player}$ *(le cas des positions de l'opposant étant dual, par la symétrie des axiomes des structures* $\alpha\beta$*-normales).*

- **(base)**

$$\frac{t = \{(\varepsilon,\pi)\} \quad \pi \in \mathbb{P}_T}{[\![\, t \,]\!]^{\alpha,\beta,\iota}_{\mathcal{D}_\approx} \Downarrow p(\pi)}$$

Dans ce cas $x = \iota$. *L'arbre* t*, lui-même, est le préfixe recherché. Il s'agit, en effet, d'un arbre fini tel que :*

$$\begin{array}{rcll} \ddot{\alpha} \mathrel{\ddot{\uparrow}} (\ddot{\beta} \mathrel{\ddot{\downarrow}} \ddot{v}) & = & \ddot{\alpha} \mathrel{\ddot{\uparrow}} (\ddot{\beta} \mathrel{\ddot{\downarrow}} \ddot{x} \mathrel{\ddot{\uparrow}} [p(\pi), p(\pi)]) & \nu = p(\pi),\ x = \iota \\ & = & \ddot{\alpha} \mathrel{\ddot{\uparrow}} \{\ddot{\beta} \mathrel{\ddot{\downarrow}} [\ddot{x} \mathrel{\ddot{\uparrow}} \ddot{v}(t)]\} & (\ddot{v}(t) = [p(\pi), p(\pi)]) \end{array}$$

Les relations de la famille $\approx_i$ *étant des équivalences, elles vérifient la propriété réflexive, donc elles contiennent l'égalité.*

- **(more)**

Dans ce cas, l'arbre t *a été découpé en deux parties qui ont été évaluées séquentiellement :* $t = t_1 \bowtie t_2$.

$$\frac{\begin{array}{c} t = t_1 \bowtie t_2 \\ \lambda(t\,\varepsilon) = \text{Player} \end{array} \qquad \dfrac{\vdots}{[\![\, t_1 \,]\!]^{\alpha,\beta,x}_{\mathcal{D}_\approx} \Downarrow \nu_1} \qquad \dfrac{\vdots}{[\![\, t_2 \,]\!]^{\alpha,\beta,(x \uparrow \nu_1)}_{\mathcal{D}_\approx} \Downarrow \nu}}{[\![\, t \,]\!]^{\alpha,\beta,x}_{\mathcal{D}_\approx} \Downarrow \nu}$$

L'évaluation de t_1 *est effectuée sur une valeur inachevée* x. *Le couple* x, t_1 *vérifie l'hypothèse (Hp), puisqu'il existe* t_1'' *tel que* $t_1'' = t' \bowtie t_1$ *et* $x \in D_\pi(t')$. *Alors, par hypothèse d'induction sur* t_1*, il existe un préfixe fini* $t_1' \leq t_1$*, tel que :*

$$\exists j. \quad \ddot{\alpha} \mathrel{\ddot{\uparrow}} (\ddot{\beta} \mathrel{\ddot{\downarrow}} \ddot{v}_1) \;\approx_j\; \ddot{\alpha} \mathrel{\ddot{\uparrow}} (\ddot{\beta} \mathrel{\ddot{\downarrow}} (\ddot{x} \mathrel{\ddot{\uparrow}} \ddot{v}(t_1'))) \qquad (t_1.E)$$

Par le lemme de caractérisation, $\nu_1 \in D_\pi(t_1'')$. *Donc la valeur inachevée initiale* $x' = \nu_1$*, utilisée pour* t_2*, est telle qu'il existe* $t'' = t' \bowtie t = t' \bowtie (t_1 \bowtie t_2) = (t' \bowtie t_1) \bowtie t_2 = t_1'' \bowtie t_2$ *et* $x' \in D_\pi(t_1'')$. *Alors, par l'hypothèse d'induction sur* t_2 *il existe un préfixe fini* $t_2' \leq t_2$*, tel que :*

$$\exists i. \quad \ddot{\alpha} \mathrel{\ddot{\uparrow}} (\ddot{\beta} \mathrel{\ddot{\downarrow}} \ddot{v}) \;\approx_i\; \ddot{\alpha} \mathrel{\ddot{\uparrow}} (\ddot{\beta} \mathrel{\ddot{\downarrow}} (\ddot{v}_1 \mathrel{\ddot{\uparrow}} \ddot{v}(t_2'))) \qquad (t_2.E)$$

Puisque t'_1 *et* t'_2 *sont des préfixes non radicaux (*π *n'est pas terminale) respectivement de* t_1 *et* t_2*, l'arbre* $t' = t'_1 \cup t'_2$ *est un préfixe non radical de* t*, c'est-à-dire* $t' \in \widehat{T}(\mathcal{S},\pi)^{\partial}$*. La co-évaluation de ce préfixe* $\ddot{v}(t') = \ddot{v}(t'_1) \ddot{\uparrow} \ddot{v}(t'_2)$ *vérifie l'énoncé :*

$$
\begin{array}{lcll}
\ddot{\alpha} \ddot{\uparrow} (\ddot{\beta} \ddot{\downarrow} \ddot{v}) & \approx_i & \ddot{\alpha} \ddot{\uparrow} \{\ddot{\beta} \ddot{\downarrow} [\ddot{v}_1 \ddot{\uparrow} \ddot{v}(t'_2)]\} & (t_2.\mathrm{E}) \\
 & \approx_i & \ddot{\alpha} \ddot{\uparrow} (\ddot{\beta} \ddot{\downarrow} \ddot{v}_1) \ddot{\uparrow} [\ddot{\beta} \ddot{\downarrow} \ddot{v}(t'_2)] & (\mathrm{A10}) \\
 & \approx_i & \ddot{\alpha} \ddot{\uparrow} \{\ddot{\beta} \ddot{\downarrow} [\ddot{x} \ddot{\uparrow} \ddot{v}(t'_1)]\} \ddot{\uparrow} [\ddot{\beta} \ddot{\downarrow} \ddot{v}(t'_2)] & (\mathrm{A1}),\ (\mathrm{A6}),\ (t_1.\mathrm{E}) \\
 & \approx_i & \ddot{\alpha} \ddot{\uparrow} \{\ddot{\beta} \ddot{\downarrow} [\ddot{x} \ddot{\uparrow} \ddot{v}(t'_1) \ddot{\uparrow} \ddot{v}(t'_2)]\} & (\mathrm{A10}) \\
 & = & \ddot{\alpha} \ddot{\uparrow} \{\ddot{\beta} \ddot{\downarrow} [\ddot{x} \ddot{\uparrow} \ddot{v}(t')]\} & \text{def. de } t'
\end{array}
$$

– **(one)**

$$
\frac{\begin{array}{l}\lambda(t\,\varepsilon) = \text{Player} \\ \text{Fils}(t) = (\ell_1.t_1)\end{array} \qquad [\![\, t_1 \,]\!]^{(\alpha\uparrow x),\beta,\iota}_{\mathcal{D}\approx} \Downarrow \nu_1 \qquad \nu = x \uparrow \nu_1}{[\![\, t \,]\!]^{\alpha,\beta,x}_{\mathcal{D}\approx} \Downarrow \nu}
$$

Il y a deux cas, selon que la position du fils t_1 *soit du joueur ou de l'opposant.*

- $\lambda(t_1\,\varepsilon) = \text{Player}$
  Dans ce cas le joueur a deux coups consécutifs (le jeu n'est pas strictement alterné). La valeur inachevée de t_1 *est initialisée à* ι*. Nous pouvons donc appliquer l'hypothèse d'induction sur* t_1*, puisque la condition (Hp) du théorème est vérifiée. Alors, il existe un préfixe fini* $t'_1 \leq t_1$ *tel que :*

  $$\exists i.\quad (\ddot{\alpha} \ddot{\uparrow} \ddot{x}) \ddot{\uparrow} (\ddot{\beta} \ddot{\downarrow} \ddot{v}_1) \approx_i (\ddot{\alpha} \ddot{\uparrow} \ddot{x}) \ddot{\uparrow} [\ddot{\beta} \ddot{\downarrow} (\ddot{\iota} \ddot{\uparrow} \ddot{v}(t'_1))] \qquad (t_1.E)$$

  Alors, il existe un arbre $t' = \{(\varepsilon,\pi)\} \cup \ell_1.t'_1$*, qui est un préfixe fini non radical de* t*, qui vérifie l'énoncé :*

$$
\begin{array}{lcll}
\ddot{\alpha} \ddot{\uparrow} (\ddot{\beta} \ddot{\downarrow} \ddot{v}) & = & \ddot{\alpha} \ddot{\uparrow} \{\ddot{\beta} \ddot{\downarrow} (\ddot{x} \ddot{\uparrow} \ddot{v}_1)\} & \ddot{v} = \ddot{x} \ddot{\uparrow} \ddot{v}_1 \\
 & \approx_i & \ddot{\alpha} \ddot{\uparrow} \{\ddot{\beta} \ddot{\downarrow} [(\ddot{\alpha} \ddot{\uparrow} \ddot{x}) \ddot{\uparrow} \ddot{v}_1]\} & (\mathrm{Insertion}),\ (\mathrm{A1}) \\
 & \approx_i & \ddot{\alpha} \ddot{\uparrow} \{\ddot{\beta} \ddot{\downarrow} [(\ddot{\alpha} \ddot{\uparrow} \ddot{x}) \ddot{\uparrow} (\ddot{\beta} \ddot{\downarrow} \ddot{v}_1)]\} & (\mathrm{Insertion}) \\
 & \approx_i & \ddot{\alpha} \ddot{\uparrow} \{\ddot{\beta} \ddot{\downarrow} [(\ddot{\alpha} \ddot{\uparrow} \ddot{x}) \ddot{\uparrow} [\ddot{\beta} \ddot{\downarrow} (\ddot{\iota} \ddot{\uparrow} \ddot{v}(t'_1))]]\} & (t_1.\mathrm{E}),\ (\mathrm{A6}) \\
 & = & \ddot{\alpha} \ddot{\uparrow} \{\ddot{\beta} \ddot{\downarrow} [(\ddot{\alpha} \ddot{\uparrow} \ddot{x}) \ddot{\uparrow} [\ddot{\beta} \ddot{\downarrow} \ddot{v}(t'_1)]]\} & \iota \uparrow y = y \\
 & \approx_i & \ddot{\alpha} \ddot{\uparrow} \{\ddot{\beta} \ddot{\downarrow} [(\ddot{\alpha} \ddot{\uparrow} \ddot{x}) \ddot{\uparrow} \ddot{v}(t'_1)]\} & (\mathrm{Insertion}) \\
 & \approx_i & \ddot{\alpha} \ddot{\uparrow} \{\ddot{\beta} \ddot{\downarrow} [\ddot{x} \ddot{\uparrow} \ddot{v}(t'_1)]\} & (\mathrm{Insertion}) \\
 & = & \ddot{\alpha} \ddot{\uparrow} \{\ddot{\beta} \ddot{\downarrow} [\ddot{x} \ddot{\uparrow} \ddot{v}(t')]\} & \ddot{v}(t') = (\ \ddot{\uparrow} \ddot{v}(t'_1)) = \ddot{v}(t'_1)
\end{array}
$$

- $\lambda(t_1\,\varepsilon) = \text{Opponent}$
  Puisque la valeur inachevée de t_1 *est initialisée à* ι*, nous pouvons appliquer l'hypothèse d'induction sur* t_1*. Alors, il existe un préfixe fini* $t'_1 \leq t_1$ *tel que :*

  $$\exists i.\quad (\ddot{\beta} \ddot{\downarrow} \ddot{x}) \ddot{\downarrow} [\ddot{\alpha} \ddot{\uparrow} \ddot{v}_1] \approx_i (\ddot{\beta} \ddot{\downarrow} \ddot{x}) \ddot{\downarrow} [\ddot{\alpha} \ddot{\uparrow} (\ddot{\iota} \ddot{\downarrow} \ddot{v}(t'_1))] \qquad (t_1.E)$$

  Alors, il existe un arbre $t' = \{(\varepsilon,\pi)\} \cup \ell_1.t'_1$*, qui est un préfixe fini non radical de* t*, qui vérifie l'énoncé :*

$$
\begin{array}{lcll}
\ddot{\beta} \ddot{\downarrow} [\ddot{\alpha} \ddot{\uparrow} \ddot{v}] & = & \ddot{\beta} \ddot{\downarrow} [\ddot{\alpha} \ddot{\uparrow} (\ddot{x} \ddot{\downarrow} \ddot{v}_1)] & \ddot{v} = \ddot{x} \ddot{\downarrow} \ddot{v}_1 \\
 & \approx_i & \ddot{\beta} \ddot{\downarrow} [\ddot{\alpha} \ddot{\uparrow} ((\ddot{\beta} \ddot{\downarrow} \ddot{x}) \ddot{\downarrow} \ddot{v}_1)] & \text{Insertion, (A1)} \\
 & \approx_i & \ddot{\beta} \ddot{\downarrow} [\ddot{\alpha} \ddot{\uparrow} ((\ddot{\beta} \ddot{\downarrow} \ddot{x}) \ddot{\downarrow} (\ddot{\alpha} \ddot{\uparrow} \ddot{v}_1))] & \text{Insertion} \\
 & \approx_i & \ddot{\beta} \ddot{\downarrow} [\ddot{\alpha} \ddot{\uparrow} ((\ddot{\beta} \ddot{\downarrow} \ddot{x}) \ddot{\downarrow} [\ddot{\alpha} \ddot{\uparrow} (\ddot{\iota} \ddot{\downarrow} \ddot{v}(t_1')))])] & (t_1.\text{E}),\ (\text{A6}) \\
 & = & \ddot{\beta} \ddot{\downarrow} [\ddot{\alpha} \ddot{\uparrow} ((\ddot{\beta} \ddot{\downarrow} \ddot{x}) \ddot{\downarrow} [\ddot{\alpha} \ddot{\uparrow} \ddot{v}(t_1')])] & \iota \downarrow y = y \\
 & \approx_i & \ddot{\beta} \ddot{\downarrow} [\ddot{\alpha} \ddot{\uparrow} ((\ddot{\beta} \ddot{\downarrow} \ddot{x}) \ddot{\downarrow} \ddot{v}(t_1'))] & \text{Insertion} \\
 & \approx_i & \ddot{\beta} \ddot{\downarrow} [\ddot{\alpha} \ddot{\uparrow} (\ddot{x} \ddot{\downarrow} \ddot{v}(t_1'))] & \text{Insertion, (A1)} \\
 & = & \ddot{\beta} \ddot{\downarrow} [\ddot{\alpha} \ddot{\uparrow} (\ddot{x} \ddot{\downarrow} \ddot{v}(t'))] & \ddot{v}(t') = \ddot{v}(t_1')
\end{array}
$$

– **(cut)**

$$
\frac{\text{cut}(\pi, \alpha, \beta, x, \approx_i)}{[\![\, t \,]\!]^{\alpha,\beta,x}_{\mathcal{D}\approx} \Downarrow x}
$$

En supposant que le noeud est du joueur, cela revient à dire :

$$
\frac{\alpha \uparrow \beta \ \approx_i\ \alpha \uparrow [\beta \downarrow (x \uparrow h_1(\pi))]}{[\![\, t \,]\!]^{\alpha,\beta,x}_{\mathcal{D}\approx} \Downarrow x}
$$

Puisque tout arbre amputé du jeu a au moins un fils, l'arbre $t' = t/^1$ *est un préfixe non radical de* t. *Considérons la co-évaluation* $\ddot{v}(t') = [y_1, y_2]$ *de ce préfixe. Par l'hypothèse (Hp) il y a deux possibilités : soit* $x = \iota$, *soit il existe* t'' *tel que* $t'' = t_x \bowtie t$ *et* $x \in D_\pi(t_x)$. *Donc* $x \in E_\pi$. *D'autre part* $y_i \in D_\pi(t) \subseteq E_\pi$. *Alors, nous pouvons appliquer l'axiome (A789p) de correction de coupure, et vérifier l'énoncé :*

$$
\begin{array}{lcll}
\ddot{\alpha} \ddot{\uparrow} (\ddot{\beta} \ddot{\downarrow} \ddot{v}) & = & \ddot{\alpha} \ddot{\uparrow} (\ddot{\beta} \ddot{\downarrow} \ddot{x}) & \ddot{v} = \ddot{x} \\
 & \approx_i & \ddot{\alpha} \ddot{\uparrow} (\ddot{\beta} \ddot{\downarrow} (\ddot{x} \ddot{\uparrow} [y_1, y_2])) & \text{cut}(\pi, \alpha, \beta, x, \approx_i),\ x, y_1, y_2 \in E_\pi,\ (\text{A789p}) \\
 & = & \ddot{\alpha} \ddot{\uparrow} (\ddot{\beta} \ddot{\downarrow} [\ddot{x} \ddot{\uparrow} \ddot{v}(t')]) & \ddot{v}(t') = [y_1, y_2]
\end{array}
$$

□

Le théorème de correction relative appliqué spécialement aux arbres de jeu t_π, explique le rapport entre la méthode Alpha-Bêta et la sémantique des positions du jeu.

Corollary 6.3.12. (Correction Alpha-Bêta) *Soit* $\mathcal{S} = (\mathbb{P}, \lambda, \rightarrow, \mathbb{P}_I)$ *un jeu co-interprété dans une structure* $\alpha\beta$*-normale* $\mathcal{D} = (D, \leq_D, \uparrow, \downarrow)$, $\mathcal{D} \in \mathsf{Cstruct}(\mathcal{S}, D, \uparrow, \downarrow, p, h_1, h_2)$ *où la famille d'équivalences utilisées soit constituée de la seule l'égalité sur les valeurs. Soit* $\pi \in \mathbb{P}$ *une position du jeu. Alors :*

$$
[\![\, \pi \,]\!]^{\iota,\iota}_{\mathcal{D}} \Downarrow \nu \quad \Rightarrow \quad [\![\, \pi \,]\!]_{\mathcal{D}} = [\nu, \nu]
$$

Démonstration. *La condition* $[\![\, \pi \,]\!]^{\iota,\iota}_{\mathcal{D}} \Downarrow \nu$ *signifie* $[\![\, t_\pi \,]\!]^{\iota,\iota,\iota}_{\mathcal{D}} \Downarrow \nu$. *Par le théorème de correction relative nous avons qu'il existe un préfixe* t' *de l'arbre de jeu* t_π *tel que, si la position est du joueur, :*

$$
\ddot{\iota} \ddot{\uparrow} (\ddot{\iota} \ddot{\downarrow} \ddot{v}) = \ddot{\iota} \ddot{\uparrow} (\ddot{\iota} \ddot{\downarrow} (\ddot{\iota} \ddot{\uparrow} \ddot{v}(t')))
$$

tandis que si la position est de l'opposant :

$$
\ddot{\iota} \ddot{\downarrow} (\ddot{\iota} \ddot{\uparrow} \ddot{v}) = \ddot{\iota} \ddot{\downarrow} (\ddot{\iota} \ddot{\uparrow} (\ddot{\iota} \ddot{\downarrow} \ddot{v}(t')))
$$

Dans le deux cas l'équation se réduit (puisque $a = \iota \uparrow a = a \uparrow \iota = \iota \downarrow a = a \downarrow \iota$*) à la forme* $\ddot{\nu} = \ddot{v}(t')$*. Nous nous retrouvons donc dans les conditions d'application du corollaire 5.5.21, par lequel la sémantique pessimiste et la sémantique optimiste de la position* π *coïncident à la valeur* ν. □

De façon plus générale, la valeur calculée par l'algorithme avec la congruence engendrée par une interprétation abstraite du jeu, coincide avec la sémantique bisémique de la position dans la structure des valeurs abstraites. Le corollaire précédent peut donc être généralisé, l'égalité étant la congruence engendrée par l'interprétation abstraite particulière où le domaine abstrait coincide avec le concret (et l'homomorphisme est la fonction identité).

COROLLARY 6.3.13. (CORRECTION ALPHA-BÊTA MODULO ABSTRACTION) *Soit* $\mathcal{S} = (\mathbb{P}, \lambda, \rightarrow, \mathbb{P}_I)$ *un jeu co-interprété dans une structure* $\alpha\beta$*-normale* $\mathcal{D} = (D, \leq_D, \uparrow, \downarrow)$, $\mathcal{D} \in \mathsf{Cstruct}(\mathcal{S}, D, \uparrow, \downarrow, p, h_1, h_2)$. *Soit* $\mathcal{D}^{\mathrm{abs}}$ *et* φ *une interprétation abstraite du jeu et supposons d'utiliser l'algorithme avec la famille d'équivalences constituée d'une unique équivalence, celle engendrée par l'abstraction, i.e. :*

$$x \approx y \overset{\triangle}{\iff} \varphi(x) = \varphi(y)$$

Soit $\pi \in \mathbb{P}$ *une position du jeu. Alors :*

$$[\![\, \pi \,]\!]^{\iota,\iota}_{\mathcal{D},\approx} \Downarrow \nu \quad \Rightarrow \quad [\![\, \pi \,]\!]_{\mathcal{D}^{\mathrm{abs}}} = [[\nu]_\approx, [\nu]_\approx]$$

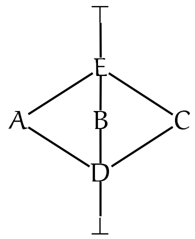
DÉMONSTRATION. *Par le théorème de correction relative nous avons qu'il existe un préfixe* t' *de l'arbre de jeu* t_π *tel que, , :*

$$\ddot{\iota} \ddot{\uparrow} (\ddot{\iota} \ddot{\downarrow} \ddot{\nu}) \approx \ddot{\iota} \ddot{\uparrow} (\ddot{\iota} \ddot{\downarrow} (\ddot{\iota} \ddot{\uparrow} \ddot{v}(t'))) \quad \textit{si la position est du joueur}$$

$$\ddot{\iota} \ddot{\downarrow} (\ddot{\iota} \ddot{\uparrow} \ddot{\nu}) \approx \ddot{\iota} \ddot{\downarrow} (\ddot{\iota} \ddot{\uparrow} (\ddot{\iota} \ddot{\downarrow} \ddot{v}(t'))) \quad \textit{si la position est de l'opposant}$$

Dans le deux cas la condition se réduit à la forme $\ddot{\nu} \approx \ddot{v}(t')$*, qui signifie* $\varphi(\nu) = \varphi(v_{p,h_1}(t')) = \varphi(v_{p,h_2}(t'))$*. Cela signifie,* φ *étant un homomorphisme,* $v^{\mathrm{abs}}_{p,h_1}(t') = v^{\mathrm{abs}}_{p,h_2}(t')$*. Puisque la structure abstraite est, elle aussi, une structure d'évaluation monotone, nous nous retrouvons dans les conditions d'application du corollaire 5.5.21, par lequel la sémantique pessimiste et la sémantique optimiste de la position* π *coïncident à la valeur* $\varphi(\nu) = [\nu]_\approx$. □

FIG. 6.4.1. Structure non distributive



6.3.6. Conclusions. Selon le théorème de correction Alpha-Bêta, si l'algorithme converge, la position analysée est une position *co-évaluable en* D. L'algorithme calcule donc la valeur des positions qui sont indépendantes de l'approche, pessimiste ou optimiste, choisie pour leur évaluation. Autrement dit, l'algorithme ne penche ni vers l'approche pessimiste, ni vers l'approche optimiste, mais il se limite à déterminer la valeur *exacte* de la position, si elle existe.

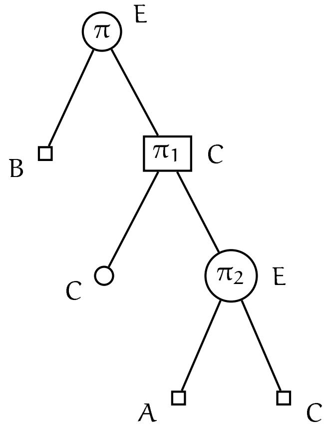
La définition de l'algorithme étant sous la forme des sémantiques opérationnelles structurées (SOS), la preuve de correction Alpha-Bêta se retrouve ici plus élégante et concise par rapport à celle présentée dans [**Di Cosmo & Loddo, 2000**]. Il est important de signaler aussi que la nouvelle preuve comprend le cas des jeux *non strictement alternés* (cas (one), sous-cas $\lambda(t_1\varepsilon) =$ Player), qui n'était pas envisagé par l'ancienne. De plus, la correction de l'algorithme est prouvée pour le cas général d'une interprétation abstraite du jeu. L'effet pratique de ce théorème est celui de permettre l'évaluation dans le domaine abstrait sans pour autant demander d'implementer les opérations abstraites $\downarrow^{abs}$ et $\uparrow^{abs}$ des joueurs. Le calcul pourra s'effectuer sur les valeurs concrètes et avec les opérations concrètes. Seul le test de coupure, vérifiant l'équivalence de deux valeurs, devra être implémenté. Par le théorème de correction, la classe d'équivalence du résultat fourni par l'algorithme sera le résultat correct, c'est-à-dire la sématique bisémique de la position, dans le domaine des valeurs abstraite.

En vue d'une plus grande généralité, le théorème est prouvé pour une famille de relation d'équivalences (dont l'union ne constitue pas forcément une relation d'équivalence).

6.4. Théorème Alpha-Bêta-Gamma-Delta

La correction d'Alpha-Bêta n'est plus garantie lorsque nous utilisons cette méthode avec une structure d'évaluation non distributive. Au cours de cette section nous allons donc définir deux nouveaux algorithmes, dont nous démontrerons la correction et qui seront, dans la section suivante, comparés à Alpha-Bêta. Puisque ces nouvelles méthodes ne seront pas appliquées dans les chapitres suivants, nous nous autoriserons un style de présentation plus intuitif et des preuves moins formelles par rapport à celles que nous avons déployées pour l'algorithme Alpha-Bêta. Dans cet esprit, un peu moins rigoureux, nous renoncerons également à la présentation des méthodes sous la forme des sémantiques opérationnelles structurées (SOS).

FIG. 6.4.2. Alpha-Bêta incorrect sur une structure non distributive



Nous pouvons construire un exemple de non correction de la méthode Alpha-Bêta sur une structure très simple. Considérons le treillis $(D, \leq_D)$, où, d'une part, $D = \{\bot, A, B, C, D, E, \top\}$ et, d'autre part, la relation d'ordre est représentée par la figure 6.4.1. Cette structure n'est pas distributive, par exemple :

$$A \uparrow (B \downarrow C) \;=\; A \uparrow D \;=\; A \;\neq\; E \;=\; E \downarrow E \;=\; (A \uparrow B) \downarrow (A \uparrow C)$$

Supposons alors d'utiliser l'algorithme Alpha-Bêta pour évaluer l'arbre de la figure 6.4.2. La valeur de la position π_2 est le sup entre A et C, donc E. Ainsi, la valeur de la position π_1 est l'inf entre E et C, donc C. La valeur qui remonte à la racine est donc E, résultat du sup des valeurs B et C. La méthode Alpha-Bêta ne calcule pas la même valeur. En supposant de visiter les noeuds de gauche à droite, la position π_1 est évaluée avec la valeur $\alpha = B$, et le noeud π_2 avec la fenêtre $\alpha = B$ et $\beta = C$. Ainsi, une fois déterminée la valeur A du premier fils de π_2, nous serons en mesure d'appliquer la règle de coupure, puisque $A \uparrow \alpha = A \uparrow B = E \geq_D C = \beta$. En conséquence, la valeur retournée par l'algorithme sera B (c'est-à-dire α) pour la position π_2, puis D (c'est-à-dire $C \downarrow B$) pour la position π_1, enfin B (c'est-à-dire $B \uparrow D$) pour la racine.

S'inspirant de la méthode Alpha-Bêta, nous pouvons définir un algorithme similaire qui est correcte pour tout type de structure, distributive ou non. Nous nous limiterons à l'étude des préordres de co-évaluation. Pour la construction d'un tel algorithme, nous remarquerons, en premier lieu, que la correction des coupures plus simples, qualifiées auparavant de *peu profondes*, est garantie par les axiomes (A6), (A7), (A8), (A9), et non par la propriété distributive (A10) des préordres de co-évaluation $\alpha\beta$-ordinaires, dont les axiomes sont regroupés dans la table 1. Ce type de coupure peut être donc utilisé assurément même lorsqu'on relâche les propriétés distributives. Sur la voie indiquée par la méthode Alpha-Bêta, le nouvel algorithme utilisera des paramètres résumant les évaluations déjà effectuées ailleurs dans l'arbre de jeu. Nous donnerons à ces paramètres, en nombre de quatre, les noms α, β, γ et δ, d'où le nom Alpha-Bêta-Gamma-Delta de la nouvelle méthode. Le relâchement de la propriété distributive nous conduit à la notion de structure $\alpha\beta\gamma\delta$-ordinaire.

DEFINITION 6.4.1. (STRUCTURE D'ÉVALUATION $\alpha\beta\gamma\delta$-ORDINAIRE) *Un préordre d'évaluation* $\mathcal{D} = (\mathsf{D}, \le_{\mathsf{D}}, \uparrow, \downarrow, \perp)$ *est* $\alpha\beta\gamma\delta$-ordinaire (*ou est une* structure $\alpha\beta\gamma\delta$-ordinaire) *s'il vérifie les conditions suivantes :*

(A6)	$\downarrow = \mathsf{ext}(\overset{2}{\downarrow})$	*et*	$\uparrow = \mathsf{ext}(\overset{2}{\uparrow})$	(*noyaux binaire*)
(A7)	$\overset{1}{\downarrow} x \le_{\mathsf{D}} x$	*et*	$\overset{1}{\uparrow} x \ge_{\mathsf{D}} x$	(*réductive, extensive*)
(A8)	$x \overset{2}{\downarrow} y \le_{\mathsf{D}} \overset{1}{\downarrow} x$	*et*	$x \overset{2}{\uparrow} y \ge_{\mathsf{D}} \overset{1}{\uparrow} x$	($\subseteq^*$*-monotonie*)
(A9)	$x \le_{\mathsf{D}} y \Rightarrow x \overset{2}{\downarrow} y = \overset{1}{\downarrow} x$	*et*	$x \le_{\mathsf{D}} y \Rightarrow x \overset{2}{\uparrow} y = \overset{1}{\uparrow} x$	*(simplification)*
(A11)	$x \overset{2}{\downarrow} (y \overset{2}{\uparrow} z) \ge_{\mathsf{D}} (x \overset{2}{\downarrow} y) \overset{2}{\uparrow} (x \overset{2}{\downarrow} z)$	*et*	$x \overset{2}{\uparrow} (y \overset{2}{\downarrow} z) \le_{\mathsf{D}} (x \overset{2}{\uparrow} y) \overset{2}{\downarrow} (x \overset{2}{\uparrow} z)$	(*distributives faibles*)

Lorsque le préordre $(\mathsf{D}, \le_{\mathsf{D}})$ *est un ordre partiel, nous disons qu'il s'agit d'un* ordre (partiel) $\alpha\beta\gamma\delta$-ordinaire. □

Avant de définir le nouvel algorithme et d'en démontrer la correction, nous pouvons étudier les conséquences des axiomes des structures $\alpha\beta\gamma\delta$-ordinaires pour caracteriser d'une autre manière la sous-classe des ordres (partiel) $\alpha\beta\gamma\delta$-ordinaire. Pour cela, nous suivrons la méthodologie expérimentée pour la caractérisation des ordres $\alpha\beta$-ordinaires de l'algorithme Alpha-Bêta.

6.4.1. Caractérisation des ordres $\alpha\beta\gamma\delta$-ordinaires. Toute structure $\alpha\beta$-ordinaire est trivialement $\alpha\beta\gamma\delta$-ordinaire, l'axiome (A11) étant impliqué par l'axiome (A10) des structures $\alpha\beta$-ordinaires. Les résultats déjà prouvés pour la caractérisation des ordres $\alpha\beta$-ordinaire peuvent être prouvés à nouveau, malgré le relâchement de la propriété distributive.

LEMMA 6.4.2. *Soit* $\mathcal{D} = (\mathsf{D}, \le_{\mathsf{D}}, \uparrow, \downarrow, \perp)$ *une structure d'évaluation* $\alpha\beta\gamma\delta$-*ordinaire. Alors :*

$$\begin{array}{llcl} (i) & z = \sup\{x, y\} \;\wedge\; \overset{1}{\uparrow} z = z & \Rightarrow & x \overset{2}{\uparrow} y = z \\ (ii) & w = \inf\{x, y\} \;\wedge\; \overset{1}{\downarrow} w = w & \Rightarrow & x \overset{2}{\downarrow} y = w \end{array}$$

DÉMONSTRATION. *Identique à celle du lemme 6.2.5, la preuve n'utilisant pas la propriété distributive.* □

PROPOSITION 6.4.3. *Dans toute ordre d'évaluation* $\alpha\beta\gamma\delta$-*ordinaire* $\mathcal{D} = (\mathsf{D}, \le_{\mathsf{D}}, \uparrow, \downarrow, \perp)$, *où* $\uparrow = \mathsf{ext}(\overset{2}{\uparrow})$ *et* $\downarrow = \mathsf{ext}(\overset{2}{\downarrow})$, *les versions unaires des deux opérations sont l'identité fonctionnelle :* $\overset{1}{\downarrow} = \overset{1}{\uparrow} = \lambda x.x$.

DÉMONSTRATION. *Tout comme dans la preuve de la proposition 6.2.6, nous pouvons prouver que les opérations unaires* $\overset{1}{\uparrow}$ *et* $\overset{1}{\downarrow}$ *sont chacune l'identité sur l'image de l'autre, c'est-à-dire* $\overset{1}{\downarrow}(\overset{1}{\uparrow} x) = \overset{1}{\uparrow} x$ *et* $\overset{1}{\uparrow}(\overset{1}{\downarrow} x) = \overset{1}{\downarrow} x$. *D'une part, nous avons :*

$$\begin{array}{rcll} \overset{1}{\uparrow} x & = & x \overset{2}{\uparrow} (x \overset{2}{\downarrow} x) & \text{(A9)}, (x \overset{2}{\downarrow} x) \le_{\mathsf{D}} x \\ & \le_{\mathsf{D}} & (x \overset{2}{\uparrow} x) \overset{2}{\downarrow} (x \overset{2}{\uparrow} x) & \text{(A11)} \\ & = & \overset{1}{\downarrow} (\overset{1}{\uparrow} x) & \textit{def. de } \overset{1}{\downarrow} \end{array}$$

D'autre part, la fonction $\overset{1}{\downarrow}$ *est réductive, donc* $\overset{1}{\downarrow}\ (\overset{1}{\uparrow}\ x) \leq_D \overset{1}{\uparrow}\ x$. *La conjonction des deux inégalités prouve donc l'équation* $\overset{1}{\downarrow}\ (\overset{1}{\uparrow}\ x) = \overset{1}{\uparrow}\ x$. *À partir de cette équation, la preuve continue de la même façon que dans la proposition 6.2.6.* □

PROPOSITION 6.4.4. *Soit* $\mathcal{D} = (D, \leq_D, \uparrow, \downarrow, \perp)$ *une structure d'évaluation. Alors les conditions suivantes sont équivalentes :*

(1) $\mathcal{D}$ *est un ordre partiel* $\alpha\beta\gamma\delta$*-ordinaire*

(2) $(D, \leq_D)$ *est un treillis où* $\sup = \uparrow$ *et* $\inf = \downarrow$

DÉMONSTRATION. *Les opérations* sup *et* inf *vérifient les axiomes de (A1) à (A9), plus l'axiome (A11) (qui dans la théorie de l'algèbre universelle est appelé théorème minimax), ce qui prouve l'implication (2)* $\Rightarrow$ *(1). D'autre part, supposons que* $\mathcal{D}$ *soit une structure* $\alpha\beta\gamma\delta$*-ordinaire et supposons par l'absurde que* $(D, \leq_D)$ *ne soit pas un treillis. En suivant la preuve de la proposition 6.2.7, nous obtiendrons l'existence d'un couple de points* (c, d)*, majorants d'un autre couple de points* (a, b)*, et tels que* $(c, d) \notin (\leq_D)$*. Tout comme dans cette preuve, nous pouvons alors démontrer l'équation* $c \overset{2}{\downarrow} d = c$*. En effet, d'une part nous avons l'inégalité suivante :*

$$\begin{array}{lcll}
c \overset{2}{\downarrow} d & = & (a \overset{2}{\uparrow} b) \overset{2}{\downarrow} d & c = a \overset{2}{\uparrow} b \\
 & = & (a \overset{2}{\uparrow} b) \overset{2}{\downarrow} (a \overset{2}{\uparrow} d) & a \leq_D d,\ (A9)\ ,\quad \overset{1}{\uparrow} = \lambda x.x \\
 & \geq_D & a \overset{2}{\uparrow} (b \overset{2}{\downarrow} d) & (A11) \\
 & = & a \overset{2}{\uparrow} b & b \leq_D d,\ (A9)\ ,\quad \overset{1}{\downarrow} = \lambda x.x \\
 & = & c &
\end{array}$$

D'autre part, les axiomes (A7) et (A8) impliquent la réciproque :

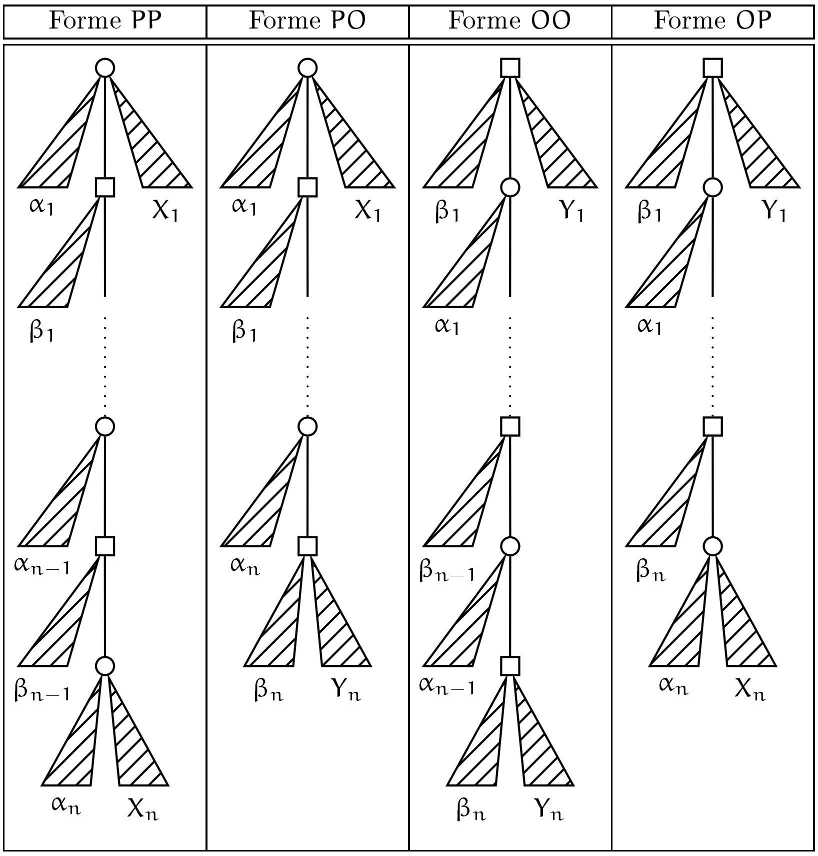
$$c \overset{2}{\downarrow} d \ \leq_D \ c$$

L'axiome (A8) est à nouveau violé, le résultat c *n'étant pas inférieur ou égal à l'argument* d *de l'opération* $\overset{2}{\downarrow}$. □

Ainsi, les ordres $\alpha\beta\gamma\delta$-ordinaires sont tout simplement les treillis (qui ne sont pas forcément distributifs).

6.4.2. La méthode Alpha-Bêta-Gamma-Delta. Considérons les quatre formes d'un arbre de jeu en cours d'évaluation représentées dans la figure 6.4.3. La première forme, que nous appellerons PP *de profondeur* n, où $n \geq 2$, débute et se termine par un noeud du joueur. Au premier niveau nous pouvons reconnaître trois groupes de fils, un premier groupe qui a déjà été évalué à la valeur α_1, un deuxième groupe constitué du fils en cours d'évaluation, et un dernier groupe dont la valeur X_1 est inconnue. Ensuite, à l'i-ème niveau intermédiaire, nous avons toujours deux groupes : un premier groupe qui a déjà été évalué (à la valeur α_i si le noeud est du joueur, ou à la valeur β_i si le noeud est de l'opposant), et un second groupe constitué d'un dernier fils en cours d'évaluation. Au dernier niveau nous avons deux groupes, un qui a été évalué à la valeur α_n et un second dont la valeur X_n est inconnue.

FIG. 6.4.3. Formes d'un arbre en évaluation



La question qui se pose est celle de déterminer des conditions permettant d'éviter l'évaluation de X_n.

Les autres formes, PO *de profondeur* n, où $n \geq 1$, qui débute par un noeud du joueur et se termine par un noeud de l'opposant, OP *de profondeur* n, où $n \geq 1$, qui a l'inverse débute par un noeud de l'opposant et se termine par un noeud du joueur, et OO *de profondeur* n, où $n \geq 2$, qui débute et se termine par un noeud de l'opposant, peuvent être interprétées de la même façon. Pour chacune de ces formes, nous allons établir les limites que la valeur sémantique de la racine (qu'elle soit pessimiste ou optimiste) ne pourra jamais franchir. Nous allons prouver, par exemple, que les formes PP *de profondeur* n sont telles que la valeur à la racine est toujours incluse entre les deux limites suivantes :

limite inférieure	limite supérieure
$\alpha_1 \uparrow \left\{ \mathop{\uparrow}_{i=2}^{n} \left[\alpha_i \downarrow \left(\mathop{\downarrow}_{j=1}^{i-1} \beta_j \right) \right] \right\} \uparrow X_1$	$\left\{ \mathop{\downarrow}_{i=1}^{n-1} \left[\beta_i \uparrow \left(\mathop{\uparrow}_{j=1}^{i} \alpha_j \right) \right] \right\} \uparrow X_1$

Les deux limites ne dépendent pas de la valeur inconnue X_n. Donc, lorsque l'équation suivante (que nous appellerons *équation de coupure*) sera vérifiée :

TAB. 4. Limites inférieures et supérieures pour les formes PP, PO, OO et OP

	limite inférieure	limite supérieure
(PP)	$\alpha_1 \uparrow \left\{ \mathop{\uparrow}\limits_{i=2}^{n} \left[\alpha_i \downarrow \left(\mathop{\downarrow}\limits_{j=1}^{i-1} \beta_j \right) \right] \right\} \uparrow X_1$	$\left\{ \mathop{\downarrow}\limits_{i=1}^{n-1} \left[\beta_i \uparrow \left(\mathop{\uparrow}\limits_{j=1}^{i} \alpha_j \right) \right] \right\} \uparrow X_1$
(PO)	$\alpha_1 \uparrow \left\{ \mathop{\uparrow}\limits_{i=2}^{n} \left[\alpha_i \downarrow \left(\mathop{\downarrow}\limits_{j=1}^{i-1} \beta_j \right) \right] \right\} \uparrow X_1$	$\left\{ \mathop{\downarrow}\limits_{i=1}^{n} \left[\beta_i \uparrow \left(\mathop{\uparrow}\limits_{j=1}^{i} \alpha_j \right) \right] \right\} \uparrow X_1$
(OO)	$\left\{ \mathop{\uparrow}\limits_{i=1}^{n-1} \left[\alpha_i \downarrow \left(\mathop{\downarrow}\limits_{j=1}^{i} \beta_j \right) \right] \right\} \downarrow Y_1$	$\beta_1 \downarrow \left\{ \mathop{\downarrow}\limits_{i=2}^{n} \left[\beta_i \uparrow \left(\mathop{\uparrow}\limits_{j=1}^{i-1} \alpha_j \right) \right] \right\} \downarrow Y_1$
(OP)	$\left\{ \mathop{\uparrow}\limits_{i=1}^{n} \left[\alpha_i \downarrow \left(\mathop{\downarrow}\limits_{j=1}^{i} \beta_j \right) \right] \right\} \downarrow Y_1$	$\beta_1 \downarrow \left\{ \mathop{\downarrow}\limits_{i=2}^{n} \left[\beta_i \uparrow \left(\mathop{\uparrow}\limits_{j=1}^{i-1} \alpha_j \right) \right] \right\} \downarrow Y_1$

$$\alpha_1 \uparrow \left\{ \mathop{\uparrow}\limits_{i=2}^{n} \left[\alpha_i \downarrow \left(\mathop{\downarrow}\limits_{j=1}^{i-1} \beta_j \right) \right] \right\} \quad = \quad \mathop{\downarrow}\limits_{i=1}^{n-1} \left[\beta_i \uparrow \left(\mathop{\uparrow}\limits_{j=1}^{i} \alpha_j \right) \right]$$

nous pourrons éviter de déterminer X_n, puisque cette valeur ne pourra, en aucun cas, conditionner la valeur de la racine. Les deux membres de l'équation peuvent être développés de la façon quelque peu plus mnémonique :

$$\text{gauche :} \quad \alpha_1 \uparrow (\alpha_2 \downarrow \beta_1) \uparrow (\alpha_3 \downarrow \beta_1 \downarrow \beta_2) \uparrow \cdots \uparrow (\alpha_n \downarrow \beta_1 \downarrow \cdots \downarrow \beta_{n-1})$$

$$\text{droit :} \quad (\beta_1 \uparrow \alpha_1) \downarrow (\beta_2 \uparrow \alpha_1 \uparrow \alpha_2) \downarrow \cdots \downarrow (\beta_{n-1} \uparrow \alpha_1 \uparrow \cdots \uparrow \alpha_{n-1})$$

La proposition suivante établie les limites inférieure et supérieure des valeurs à la racine pour chacune des formes PP, PO, OO et OP. Autrement dit, elle établie les équations de coupure pour les différentes formes de visite.

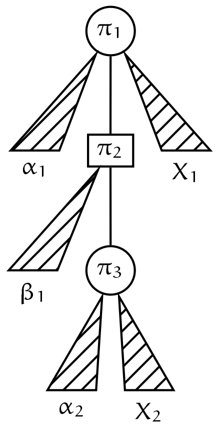
PROPOSITION 6.4.5. *Soit $\mathcal{S} = (\mathbb{P}, \lambda, \rightarrow, \mathbb{P}_I)$ un jeu strictement alterné, co-interprété dans un préordre de co-évaluation $\alpha\beta\gamma\delta$-ordinaire $\mathcal{D} \in \mathsf{Cstruct}(\mathcal{S}, D, \uparrow, \downarrow, p, h_1, h_2)$, $\mathcal{D} = (D, \leq_D, \uparrow, \downarrow)$. Alors, pour les quatre formes* PP, PO, OO *et* OP, *les limites inférieures et supérieures de la valeur à la racine de l'arbre, sont répertoriées dans la table 4.*

DÉMONSTRATION. *Nous allons montrer que, indépendemment de l'approche, pessimiste ou optimiste, la sémantique (monosémique) de la position à la racine est forcément contenue entre les deux valeurs spécifiées par la table. Les preuves sont entrelacées. Une preuve se fait par induction, sur la hauteur de l'arbre, pour le binôme* PP *et* OP, *l'autre preuve, que nous omettrons par dualité, se fait elle aussi par le même type d'induction, mais sur le second binôme, celui des formes* OO *et* PO.

Les cas de base sont la profondeur $n = 2$ pour la forme PP, *et la profondeur $n = 1$ pour la forme* OP. *Les limites sont alors :*

	limite inférieure	*limite supérieure*
$(n = 2)$ (PP)	$\alpha_1 \uparrow (\alpha_2 \downarrow \beta_1) \uparrow X_1$	$\beta_1 \uparrow \alpha_1 \uparrow X_1$
$(n = 1)$ (OP)	$\alpha_1 \downarrow \beta_1 \downarrow Y_1$	$\beta_1 \downarrow Y_1$

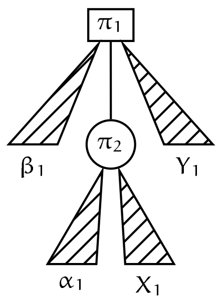
– *cas de base $(n = 2)$ pour la forme* PP
L'arbre a la forme suivante :



Par la remarque 5.5.18, nous avons que la valeur à la racine égale à $\llparenthesis \pi_1 \rrparenthesis = \alpha_1 \uparrow \llparenthesis \pi_2 \rrparenthesis \uparrow X_1$. Nous avons alors l'énoncé :

$$\begin{array}{llll} \beta_1 \uparrow \alpha_1 \uparrow X_1 & \geq_D & \alpha_1 \uparrow \llparenthesis \pi_2 \rrparenthesis \uparrow X_1 & \beta_1 \geq_D \llparenthesis \pi_2 \rrparenthesis \\ & = & \llparenthesis \pi_1 \rrparenthesis & \\ & = & \alpha_1 \uparrow (\beta_1 \downarrow \llparenthesis \pi_3 \rrparenthesis) \uparrow X_1 & \\ & \geq_D & \alpha_1 \uparrow (\beta_1 \downarrow \alpha_2) \uparrow X_1 & \llparenthesis \pi_3 \rrparenthesis \geq_D \alpha_2 \end{array}$$

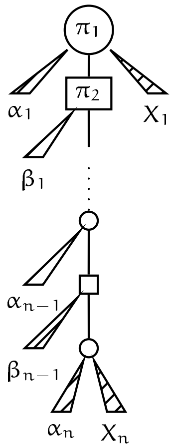
– *cas de base $(n = 1)$ pour la forme* OP
L'arbre a la forme suivante :



Toujours par la remarque 5.5.18, nous avons que la valeur à la racine égale à $\llparenthesis \pi_1 \rrparenthesis = \beta_1 \downarrow \llparenthesis \pi_2 \rrparenthesis \downarrow Y_1$. À nouveau, nous avons facilement l'énoncé :

$$
\begin{array}{llll}
\beta_1 \downarrow Y_1 & \geq_D & \beta_1 \downarrow \llparenthesis \pi_2 \rrparenthesis \downarrow Y_1 & \textit{monotonie de } \downarrow \\
 & = & \llparenthesis \pi_1 \rrparenthesis & \\
 & = & \beta_1 \downarrow (\alpha_1 \uparrow X_2) \downarrow Y_1 & \\
 & \geq_D & \beta_1 \downarrow \alpha_1 \downarrow Y_1 & \textit{monotonie de } \uparrow
\end{array}
$$

– *cas ($n > 2$) pour la forme* PP
L'arbre a la forme suivante :



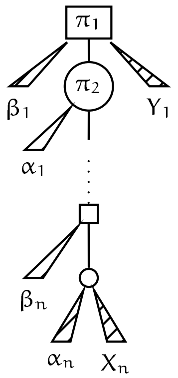
La branche de la position π_2 est une forme OP *d'une hauteur inférieure (et de profondeur $n-1$), sur laquelle nous pouvons appliquer l'hypothèse d'induction. Pour ce dernier, nous remarquerons simplement que la numérotation des groupes déjà évalués du joueur démarre avec α_2 (au lieu de α_1) et se termine avec α_n.*

$$
\begin{array}{lll}
\alpha_1 \uparrow \left\{ \overset{n}{\underset{i=2}{\uparrow}} \left[\alpha_i \downarrow \left(\overset{i-1}{\underset{j=1}{\downarrow}} \beta_j \right) \right] \right\} \uparrow X_1 & = & \alpha_1 \uparrow \left\{ \overset{n-1}{\underset{h=1}{\uparrow}} \left[\alpha_{h+1} \downarrow \left(\overset{h}{\underset{j=1}{\downarrow}} \beta_j \right) \right] \right\} \uparrow X_1 \qquad (1) \\
 & \leq_D & \alpha_1 \uparrow \llparenthesis \pi_2 \rrparenthesis \uparrow X_1 \qquad (2) \\
 & = & \llparenthesis \pi_1 \rrparenthesis \qquad (3) \\
 & = & \alpha_1 \uparrow \left(\beta_1 \downarrow \left\{ \overset{n-1}{\underset{i=2}{\downarrow}} \left[\beta_i \uparrow \left(\overset{i-1}{\underset{j=1}{\uparrow}} \alpha_{j+1} \right) \right] \right\} \right) \uparrow X_1 \qquad (4) \\
 & \leq_D & \left((\alpha_1 \uparrow \beta_1) \downarrow \left(\alpha_1 \uparrow \left\{ \overset{n-1}{\underset{i=2}{\downarrow}} \left[\beta_i \uparrow \left(\overset{i-1}{\underset{j=1}{\uparrow}} \alpha_{j+1} \right) \right] \right\} \right) \right) \uparrow X_1 \qquad (5) \\
 & \leq_D & \left((\alpha_1 \uparrow \beta_1) \downarrow \left\{ \overset{n-1}{\underset{i=2}{\downarrow}} \left[\alpha_1 \uparrow \beta_i \uparrow \left(\overset{i-1}{\underset{j=1}{\uparrow}} \alpha_{j+1} \right) \right] \right\} \right) \uparrow X_1 \qquad (6) \\
 & = & \left((\alpha_1 \uparrow \beta_1) \downarrow \left\{ \overset{n-1}{\underset{i=2}{\downarrow}} \left[\beta_i \uparrow \left(\overset{i}{\underset{h=1}{\uparrow}} \alpha_h \right) \right] \right\} \right) \uparrow X_1 \qquad (7) \\
 & = & \left\{ \overset{n-1}{\underset{i=1}{\downarrow}} \left[\beta_i \uparrow \left(\overset{i}{\underset{h=1}{\uparrow}} \alpha_h \right) \right] \right\} \uparrow X_1
\end{array}
$$

(1) *par la transformation* $h = i - 1$ *;* (2) *par hypothèse d'induction (limite inférieure) ;* (3) *par la remarque 5.5.18 ;* (4) *par*

hypothèse d'induction (limite supérieure) ; (5) *et* (6) *par la propriété distributive faible (axiome A11)* ; (7) *par la transformation* $h = j + 1$.

– *cas* ($n > 1$) *pour la forme* OP
L'arbre a la forme suivante :



Cette fois ci, la branche de la position π_2 *est une forme* PP *d'une hauteur inférieure (et de profondeur* n*), sur laquelle nous pouvons appliquer l'hypothèse d'induction. En ce qui concerne la branche* PP*, il faudra prêter attention, une fois de plus, au décalage des groupes de l'opposant déjà évalués : la numérotation démarre avec* β_2 *(au lieu de* β_1*) et se termine avec* β_n *(au lieu de* β_{n-1}*).*

$$
\begin{aligned}
\beta_1 \downarrow \left\{ \mathop{\downarrow}_{i=2}^{n} \left[\beta_i \uparrow \left(\mathop{\uparrow}_{j=1}^{i-1} \alpha_j \right) \right] \right\} \downarrow Y_1 &= \beta_1 \downarrow \left\{ \mathop{\downarrow}_{h=1}^{n-1} \left[\beta_{h+1} \uparrow \left(\mathop{\uparrow}_{j=1}^{h} \alpha_j \right) \right] \right\} \downarrow Y_1 && (1) \\
&\geq_D \beta_1 \downarrow \llparenthesis \pi_2 \rrparenthesis \downarrow Y_1 && (2) \\
&= \llparenthesis \pi_1 \rrparenthesis && (3) \\
&= \beta_1 \downarrow \left(\alpha_1 \uparrow \left\{ \mathop{\uparrow}_{i=2}^{n} \left[\alpha_i \downarrow \left(\mathop{\downarrow}_{j=1}^{i-1} \beta_{j+1} \right) \right] \right\} \right) \downarrow Y_1 && (4) \\
&\geq_D \left((\beta_1 \downarrow \alpha_1) \uparrow \left(\beta_1 \downarrow \left\{ \mathop{\uparrow}_{i=2}^{n} \left[\alpha_i \downarrow \left(\mathop{\downarrow}_{j=1}^{i-1} \beta_{j+1} \right) \right] \right\} \right) \right) \downarrow Y_1 && (5) \\
&\geq_D \left((\beta_1 \downarrow \alpha_1) \uparrow \left\{ \mathop{\uparrow}_{i=2}^{n} \left[\beta_1 \downarrow \alpha_i \downarrow \left(\mathop{\downarrow}_{j=1}^{i-1} \beta_{j+1} \right) \right] \right\} \right) \downarrow Y_1 && (6) \\
&= \left((\beta_1 \downarrow \alpha_1) \uparrow \left\{ \mathop{\uparrow}_{i=2}^{n} \left[\alpha_i \downarrow \left(\mathop{\downarrow}_{h=1}^{i} \beta_h \right) \right] \right\} \right) \downarrow Y_1 && (7) \\
&= \left\{ \mathop{\uparrow}_{i=1}^{n} \left[\alpha_i \downarrow \left(\mathop{\downarrow}_{h=1}^{i} \beta_h \right) \right] \right\} \downarrow Y_1
\end{aligned}
$$

(1) *par la transformation* $h = i - 1$; (2) *par hypothèse d'induction (limite supérieure)* ; (3) *par la remarque 5.5.18* ; (4) *par hypothèse d'induction (limite inférieure)* ; (5) *et* (6) *par la propriété distributive faible (axiome A11)* ; (7) *par la transformation* $h = j + 1$.

TAB. 5. Les équations de coupure

(PP)	$\alpha_1 \uparrow \left\{ \overset{n}{\underset{i=2}{\uparrow}} \left[\alpha_i \downarrow \left(\overset{i-1}{\underset{j=1}{\downarrow}} \beta_j \right) \right] \right\}$	$=$	$\overset{n-1}{\underset{i=1}{\downarrow}} \left[\beta_i \uparrow \left(\overset{i}{\underset{j=1}{\uparrow}} \alpha_j \right) \right]$
(PO)	$\alpha_1 \uparrow \left\{ \overset{n}{\underset{i=2}{\uparrow}} \left[\alpha_i \downarrow \left(\overset{i-1}{\underset{j=1}{\downarrow}} \beta_j \right) \right] \right\}$	$=$	$\overset{n}{\underset{i=1}{\downarrow}} \left[\beta_i \uparrow \left(\overset{i}{\underset{j=1}{\uparrow}} \alpha_j \right) \right]$
(OO)	$\overset{n-1}{\underset{i=1}{\uparrow}} \left[\alpha_i \downarrow \left(\overset{i}{\underset{j=1}{\downarrow}} \beta_j \right) \right]$	$=$	$\beta_1 \downarrow \left\{ \overset{n}{\underset{i=2}{\downarrow}} \left[\beta_i \uparrow \left(\overset{i-1}{\underset{j=1}{\uparrow}} \alpha_j \right) \right] \right\}$
(OP)	$\overset{n}{\underset{i=1}{\uparrow}} \left[\alpha_i \downarrow \left(\overset{i}{\underset{j=1}{\downarrow}} \beta_j \right) \right]$	$=$	$\beta_1 \downarrow \left\{ \overset{n}{\underset{i=2}{\downarrow}} \left[\beta_i \uparrow \left(\overset{i-1}{\underset{j=1}{\uparrow}} \alpha_j \right) \right] \right\}$

□

En ce qui concerne les limites pour les formes PP et PO, l'argument X_1 de $\uparrow$ est présent à la fois du coté de la limite inférieure et du coté de la limite supérieure. Nous pouvons donc l'éliminer des deux membres de l'équation de coupure correspondante. De façon duale, pour les formes PP et PO, c'est en éliminant la valeur Y_1 que nous simplifierons les équations de coupure correspondantes. L'ensemble des équations de coupures, illustré dans la table 5, justifie l'algorithme 7, où se trouve définie la fonction Alpha-Bêta-Gamma-Delta. Mis à part le premier paramètre, qui est toujours la position π en cours d'analyse, le sens des autres paramètres de cette fonction est le suivant :

- α est le résultat de la fonction du joueur $\uparrow$ appliquée à toutes les valeurs déjà calculées dans les noeuds antécédents du joueur :

$$\alpha = \begin{cases} \overset{n}{\underset{i=1}{\uparrow}} \alpha_i & \text{pour les formes PP, PO, OP} \\ \\ \overset{n-1}{\underset{i=1}{\uparrow}} \alpha_i & \text{pour la forme OO} \end{cases}$$

- β est le résultat de la fonction de l'opposant $\downarrow$ appliquée à toutes les valeurs déjà calculées dans les noeuds antécédents de l'opposant :

$$\beta = \begin{cases} \overset{n}{\underset{i=1}{\uparrow}} \beta_i & \text{pour les formes OO, PO, OP} \\ \\ \overset{n-1}{\underset{i=1}{\uparrow}} \beta_i & \text{pour la forme PP} \end{cases}$$

- γ et δ sont, à un niveau de profondeur n, la suite, respectivement, des membres gauches et des membres droits de $k \leq n$ équations :

Algorithm 7 *Alpha-Bêta-Gamma-Delta*

fonction Alpha-Bêta-Gamma-Delta(π : $\mathbb{P}$, α,β : D, γ,δ : D^*) : D

1. si $\pi \in \mathbb{P}_T$ alors retourner $p(\pi)$

2. sinon soit $\pi_1,\ldots,\pi_m$ l'ordre d'évaluation des m successeurs de π

3. si $\lambda(\pi) = \text{Player}$ alors
   - 3.1) $v \leftarrow h_1(\pi)$
   - 3.2) pour tout j de 1 à $|\gamma|$ faire $\gamma_j \leftarrow \gamma_j \uparrow (\beta \downarrow v)$
   - 3.3) pour i allant de 1 à $m-1$ et tant que ($\nexists j.\ \gamma_j = \delta_j$) faire
     - 3.3.1) $v \leftarrow v \uparrow$ Alpha-Bêta-Gamma-Delta($\pi_i, v, \top, v, \top$)
     - 3.3.2) pour tout j de 1 à $|\gamma|$ faire $\gamma_j \leftarrow \gamma_j \uparrow (\beta \downarrow v)$
   - 3.4) si ($\nexists j.\ \gamma_j = \delta_j$) faire $v \leftarrow v \uparrow$ Alpha-Bêta-Gamma-Delta($\pi_m, \alpha \uparrow v, \beta, v.\gamma, \top.\delta$)

4. si $\lambda(\pi) = \text{Opponent}$ alors
   - 4.1) $v \leftarrow h_2(\pi)$
   - 4.2) pour tout j de 1 à $|\delta|$ faire $\delta_j \leftarrow \delta_j \downarrow (\alpha \uparrow v)$
   - 4.3) pour i allant de 1 à $m-1$ et tant que ($\nexists j.\ \gamma_j = \delta_j$) faire
     - 4.3.1) $v \leftarrow v \downarrow$ Alpha-Bêta-Gamma-Delta($\pi_i, \bot, v, \bot, v$)
     - 4.3.2) pour tout j de 1 à $|\delta|$ faire $\delta_j \leftarrow \delta_j \downarrow (\alpha \uparrow v)$
   - 4.4) si ($\nexists j.\ \gamma_j = \delta_j$) faire $v \leftarrow v \downarrow$ Alpha-Bêta-Gamma-Delta($\pi_m, \alpha, \beta \downarrow v, \bot.\gamma, v.\delta$)

5. retourner v

$$\begin{cases} \gamma_1 & = & \delta_1 \\ & \vdots & \\ \gamma_k & = & \delta_k \end{cases}$$

plus fortes, d'un point de vue logique, que les k équations de coupure.

La méthode a une caractéristique spécifique : une fois terminée l'évaluation des autres coups, elle réserve un traitement particulier au *dernier* coup disponible. Supposons dorénavant que le jeu soit strictement alterné. Dans une position π du joueur, lorsque nous analysons un coup intermédiaire, nous débutons une visite qui prend la forme PO de profondeur 1. En revanche, lorsque nous analysons le dernier coup disponible, nous rallongeons la visite inaugurée au niveau, de l'opposant, précédent celui de la position π. Cette visite qui était de la forme OP devient alors, par effet de cette élongation, de la forme OO. Si nous regardons à un niveau encore supérieur (donc du joueur), une visite inaugurée à ce niveau et qui était de la forme PP de profondeur 1 au niveau de π, devient, une fois activée l'évaluation du dernier fils de π, une visite de la forme PO de profondeur 2. À tout moment, nous pouvons considérer d'avoir activé une multitude k de visites (au maximum autant que la profondeur n du noeud analysé), où $k = |\gamma| = |\delta|$, chacune ayant une des quatre formes canoniques, et chacune ayant une i-ème équation de coupure associée, dont le membre gauche est supérieur à γ_i et le membre droit est inférieur à δ_i. Autrement dit, la condition de coupure utilisée par l'algorithme est une condition plus forte que nécessaire.

Comme pour l'algorithme Alpha-Bêta, nous pouvons montrer la correction de l'algorithme vis-à-vis de la sémantique bisémique. Si l'algorithme se termine sur une position π, il s'agit à nouveau d'une position *évaluable en* D (cf. section 5.5.4).

THEOREM 6.4.6. (CORRECTION ALPHA-BÊTA-GAMMA-DELTA) *Soit* $\mathcal{S} = (\mathbb{P}, \lambda, \rightarrow, \mathbb{P}_I)$ *un jeu strictement alterné, co-interprété dans un préordre de co-évaluation* $\alpha\beta\gamma\delta$*-ordinaire* $\mathcal{D} \in \mathsf{Cstruct}(\mathcal{S}, \mathrm{D}, \uparrow, \downarrow, p, h_1, h_2)$, $\mathcal{D} = (\mathrm{D}, \leq_{\mathrm{D}}, \uparrow, \downarrow)$, *avec un élément minimum* $\perp$ *et un élément maximum* $\top$. *Soit* $\pi \in \mathbb{P}$ *une position du jeu. Alors :*

$$\texttt{Alpha-Bêta-Gamma-Delta}(\pi, \perp, \top, \varepsilon, \varepsilon) = \nu \qquad \Rightarrow \qquad [\![\, \pi \,]\!]_{\mathcal{D}} = [\nu, \nu]$$

DÉMONSTRATION. *Considérons, juste le temps de la preuve, de modifier l'algorithme en ajoutant le paramètre* p *représentant la suite de positions visitées. L'en-tête de la fonction sera :*

$\texttt{fonction Alpha-Bêta-Gamma-Delta}(\pi{:}\mathbb{P}, \alpha,\beta{:}\mathrm{D}, \gamma,\delta{:}\mathrm{D}^*, p{:}\mathbb{P}^*){:}\mathrm{D}$

et les appels récursifs seront modifiés de la façon suivante :

3.3.1)	$\texttt{Alpha-Bêta-Gamma-Delta}(\pi_i, v, \top, v, \top, \pi)$
3.4)	$\texttt{Alpha-Bêta-Gamma-Delta}(\pi_m, \alpha \uparrow v, \beta, v.\gamma, \top.\delta, \pi.p)$
4.3.1)	$\texttt{Alpha-Bêta-Gamma-Delta}(\pi_i, \perp, v, \perp, v, \pi)$
4.4)	$\texttt{Alpha-Bêta-Gamma-Delta}(\pi_m, \alpha, \beta \downarrow v, \perp.\gamma, v.\delta, \pi.p)$

Il est évident que les deux algorithmes, le nouveau et l'ancien, sont équivalents, le paramètre p *n'étant jamais utilisé durant le calcul. Ce dernier est, cependant, très utile pour formuler une propriété de l'exécution de la méthode à partir d'un appel avec les paramètres* $(\pi, \alpha, \beta, \gamma, \delta, p)$ *:*

$$\mathsf{P}(\pi, \alpha, \beta, \gamma, \delta, p) \stackrel{\triangle}{\Longleftrightarrow}$$

(1) *toute valeur intermédiaire de la variable* ν*, calculée à partir de* $(\pi, \alpha, \beta, \gamma, \delta, p)$*, est telle que :*

- *si* $\lambda(\pi) = \mathrm{Player}$*, alors pour tout* $i = 1..|p|$*, la condition* $\gamma_i \uparrow (\beta \downarrow \nu) = \delta_i$ *implique l'équation de coupure associée à la position* p_i *;*
- *si* $\lambda(\pi) = \mathrm{Opponent}$*, alors pour tout* $i = 1..|p|$*, la condition* $\gamma_i = \delta_i \downarrow (\alpha \uparrow \nu)$ *implique l'équation de coupure associée à la position* p_i *;*

(2) *la valeur* α *est le résultat de l'opération* $\uparrow$ *sur toutes les valeurs inachevées des positions du joueur, s'il en existe entre le noeud de la position* p_1 *et celui de la position* p_k*, où* $k = |p|$ *; s'il n'en existe pas,* α *est égale à* $\perp$ *;*

(3) *la valeur* β *est le résultat de l'opération* $\downarrow$ *sur toutes les valeurs inachevées des positions de l'opposant, s'il en existe entre le noeud de la position* p_1 *et celui de la position* p_k*, où* $k = |p|$ *; s'il n'en existe pas,* β *est égale à* $\top$ *;*

Nous allons prouver (a) que la propriété P *est vérifiée lors de l'appel initial, puis (b), que* P *est conservée dans tous les appels d'une exécution qui la vérifie.*

– *(a)* $P(\pi, \perp, \top, \varepsilon, \varepsilon, \varepsilon)$

  Puisque les suites $\gamma = \varepsilon$ *et* $\delta = \varepsilon$ *sont vides, la condition* (1) *est trivialement vrai. Les conditions* (2) *et* (3) *sont également vérifiées puisque* $|p| = 0$, $\alpha = \perp$ *et* $\beta = \top$.

– *(b)* $P(\pi, \alpha, \beta, \gamma, \delta, p) \Rightarrow P(\pi', \alpha', \beta', \gamma', \delta', p')$ *pour tout appel récursif :*

  $$\texttt{Alpha-Bêta-Gamma-Delta}(\pi', \alpha', \beta', \gamma', \delta', p')$$

  effectué durant l'exécution de :

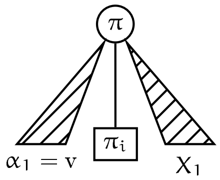
  $$\texttt{Alpha-Bêta-Gamma-Delta}(\pi, \alpha, \beta, \gamma, \delta, p).$$

  Supposons démarrer le processus d'évaluation sur une position du joueur (le cas de l'opposant est dual). Lorsque la position n'est pas terminale, il y a deux type d'appels récursifs, un durant l'évaluation des coups intermédiaires, l'autre durant l'évaluation du dernier coup.

  (coups intermédiaires) *Dans la boucle d'évaluation des coups intermédiaires (du premier* 1 *à l'avant-dernier* $m - 1$*), l'appel récursif est le suivant :*

  $$\texttt{Alpha-Bêta-Gamma-Delta}(\pi_i, \nu, \top, \nu, \top, \pi)$$

  et correspond précisément à l'inauguration d'une visite de la forme PO *(puisque* π *est du joueur et* π_i *de l'opposant) où la valeur* $\alpha_1 = \nu$ *est la valeur inachevée de* π *:*



  Pour une forme de ce genre, la proposition 6.4.5 prévoit que la valeur (pessimiste ou optimiste) de la position π*, si elle existe, soit comprise entre les limites suivantes :*

  $$\alpha_1 \uparrow X_1 \;\leq\; (\!|\pi|\!) \;\leq\; \alpha_1 \uparrow \beta_1 \uparrow X_1$$

  où $\beta_1 = \nu'$ *est une quelconque valeur inachevée, calculée au niveau de la position* π_i *avec les paramètres* $\alpha' = \nu$, $\beta' = \top$, $\gamma' = \nu$ *et* $\delta' = \top$. *L'équation de coupure associée à* π *dans l'évaluation de* π_i *est donc* $\nu = \nu \uparrow \nu'$*, c'est-à-dire (puisque* $\gamma' = \alpha' = \nu$ *et* $\delta' = \top$*) :*

  $$\gamma' = \delta' \downarrow (\alpha' \uparrow \nu')$$

donc l'appel vérifie, lui aussi, la condition (1). *De plus,* $\alpha' = \nu$ *est le résultat du* $\uparrow$ *sur toutes les valeurs inachevées du joueur à partir de la position* $p'_1 = \pi$. *Il n'y a pas de position de l'opposant dans la suite* p *et* $\beta' = \top$. *Autrement dit, les conditions* (2) *et* (3) *sont, elles aussi, vérifiées.*

(dernier coup) *Pour le dernier coup disponible, conduisant à la position* π', *l'appel récursif a lieu avec les valeurs* $\alpha' = \alpha \uparrow \nu$, $\beta' = \beta$, $\gamma' = \nu.\gamma$ *et* $\delta' = \top.\delta$. *L'appel récursif ajoute donc des éléments en tête sans modifier le reste des suites* γ *et* δ. *Nous devons prouver que les nouveaux paramètres sont tels que et* $\delta'_{i'} = \delta'_{i'} \downarrow (\alpha' \uparrow \nu')$ *est une équation qui implique l'équation de coupure associée à* $p'_{i'}$. *Nous allons donc traiter différemment le cas* $i' = 1$ *et le cas* $i' > 1$.

* $i' = 1$
  À nouveau, les éléments en tête des suite γ' *et* δ', *correspondent à l'inauguration d'une visite de la forme* PO *(puisque* π *est du joueur et* π_m *de l'opposant) où la valeur* $\alpha_1 = \nu$ *est la valeur inachevée de* π. *Pour l'indice* $i' = 1$, *les valeurs* $\gamma'_1 = \nu$ *et* $\delta'_1 = \top$ *sont les mêmes utilisées par les appels des coups intermédiaires. On peut alors prouver l'énoncé par les mêmes arguments utilisés auparavant.*

* $i' > 1$
  Nous avons que $i' = i + 1$ *où, par hypothèse, les variables* γ_i *et* β_i *représentent, à tout moment de l'évaluation de* π, *l'équation de coupure pour la position* p_i. *Nous ne discutons que le cas où la position* p_i *est du joueur. Le cas où elle est de l'opposant se prouve de façon duale. Si* p_i *est du joueur, l'équation* $\gamma_i = \delta_i$ *représente un relâchement de l'équation de coupure d'une forme* PP *avec* p_i *à la racine. Nous avons donc :*

$$\gamma_i \leq \alpha_1 \uparrow \left\{ \mathop{\uparrow}_{h=2}^{k} \left[\alpha_h \downarrow \left(\mathop{\downarrow}_{j=1}^{h-1} \beta_j \right) \right] \right\} \leq \mathop{\downarrow}_{h=1}^{k-1} \left[\beta_h \uparrow \left(\mathop{\uparrow}_{j=1}^{h} \alpha_j \right) \right] \leq \delta_i$$

  où $\alpha_1, .., \alpha_k$ *sont les valeurs inachevées des noeuds du joueurs entre* p_i *et* π *(*α_1 *au niveau de* p_i, $\alpha_k = \nu$ *au niveau de* π*), et* $\beta_1, .., \beta_{k-1}$ *sont les valeurs inachevées des noeuds de l'opposant entre les noeuds de* p_i *et de* π. *Au niveau de la nouvelle position* π', *de l'opposant, celle qui était une forme* PP *de profondeur* k *devient une forme* PO *de profondeur* k. *Si une valeur* ν' *est calculée à ce niveau, l'équation de coupure est, pour cette forme, la suivante :*

$$\alpha_1 \uparrow \left\{ \mathop{\uparrow}_{i=2}^{k} \left[\alpha_i \downarrow \left(\mathop{\downarrow}_{j=1}^{i-1} \beta_j \right) \right] \right\} = \mathop{\downarrow}_{i=1}^{k} \left[\beta_i \uparrow \left(\mathop{\uparrow}_{j=1}^{i} \alpha_j \right) \right]$$

  Il n'y a donc pas de différence avec le membre gauche précédent, qui reste alors supérieur à γ_i. *En revanche, on obtient le membre droit, en ajoutant au précédent l'argument*

$$\beta_k \uparrow \left(\mathop{\uparrow}_{j=1}^{k} \alpha_j \right)$$

  où $\beta_k = \nu'$ *est la valeur inachevée calculée au niveau de* π'. *Par ailleurs, le résultat de* $\mathop{\uparrow}_{j=1}^{k} \alpha_j$ *est forcément inférieur au paramètre* α' *reçu qui, lui, est le* $\uparrow$ de toutes *les valeurs inachevées* *du joueur. Donc, nous avons aussi que* $\delta'_{i'} = \delta_i \downarrow (\alpha' \uparrow \nu')$ *est supérieur au membre droit de l'équation de coupure de la forme* PO *associée à* $p'_{i'} = p_i$.

□

La preuve du théorème précédent nous suggère de corriger la méthode pour que les paramètres γ et δ représentent *exactement* le système des équations de coupures, et non pas un système d'équations relâchées. Il suffit, pour cela, de conserver la trace des valeurs $\underset{j=1..k}{\uparrow} \alpha_j$ et $\underset{j=1..k}{\uparrow} \beta_j$ pour toutes les visites activées, et non seulement pour la première inaugurée. Nous obtenons, par ces modifications, l'algorithme 8, où les paramètres α et β sont, cette fois ci, des suites : chaque α_i (resp. β_i) est le résultat de la fonction $\uparrow$ (resp. $\downarrow$) appliquée au valeurs inachevées du joueur (resp. de l'opposant) observées au cours de la visite inaugurée à la position p_i. Cette

Algorithm 8 *Alpha-Bêta-Gamma-Delta (2-ème version)*

fonction Alpha-Bêta-Gamma-Delta2(π : $\mathbb{P}$, α, β : D, γ, δ : D^*) : D

1. si $\pi \in \mathbb{P}_T$ alors retourner $p(\pi)$
2. sinon soit $\pi_1, \ldots, \pi_m$ l'ordre d'évaluation des m successeurs de π
3. si $\lambda(\pi) =$ Player alors
   - 3.1) v $\leftarrow h_1(\pi)$
   - 3.2) pour tout j de 1 à $|\gamma|$ faire $\gamma_j \leftarrow \gamma_j \uparrow (\beta \downarrow v)$
   - 3.3) pour i allant de 1 à $m-1$
     et tant que $(\nexists j.\ \gamma_j = \delta_j)$ faire
     3.3.1) v $\leftarrow$ v $\uparrow$ Alpha-Bêta-Gamma-Delta2(π_i,v,$\top$,v,$\top$)
     3.3.2) pour tout j de 1 à $|\gamma|$ faire
     $\gamma_j \leftarrow \gamma_j \uparrow (\beta_j \downarrow v)$
   - 3.4) si $(\nexists j.\ \gamma_j = \delta_j)$ faire
     v $\leftarrow$ v $\uparrow$ Alpha-Bêta-Gamma-Delta2(π_m,v.$(\alpha_1 \uparrow v) \ldots (\alpha_{|\alpha|} \uparrow v)$,$\top.\beta$,v.$\gamma$,$\top.\delta$)
4. si $\lambda(\pi) =$ Opponent alors
   - 4.1) v $\leftarrow h_2(\pi)$
   - 4.2) pour tout j de 1 à $|\delta|$ faire $\delta_j \leftarrow \delta_j \downarrow (\alpha \uparrow v)$
   - 4.3) pour i allant de 1 à $m-1$
     et tant que $(\nexists j.\ \gamma_j = \delta_j)$ faire
     4.3.1) v $\leftarrow$ v $\downarrow$ Alpha-Bêta-Gamma-Delta2(π_i,$\perp$,v,$\perp$,v)
     4.3.2) pour tout j de 1 à $|\delta|$ faire
     $\delta_j \leftarrow \delta_j \downarrow (\alpha_j \uparrow v)$
   - 4.4) si $(\nexists j.\ \gamma_j = \delta_j)$ faire
     v $\leftarrow$ v $\downarrow$ Alpha-Bêta-Gamma-Delta2(π_m,$\perp.\alpha$,v.$(\beta_1 \downarrow v) \ldots (\beta_{|\beta|} \downarrow v)$,$\perp.\gamma$,v.$\delta$)
5. retourner v

nouvelle version est, certes, plus efficace pour détecter les branches de l'arbre de jeu à élaguer, mais elle engage un surcroît d'actions dont le coût doit être évalué au cas par cas, selon le type de domaine d'évaluation utilisé. Dans la section suivante, nous allons illustrer le résultat de quelques tests d'évaluation d'arbres de jeu par les différentes méthodes de calcul. Il en ressortira que la seconde version de l'algorithme Alpha-Bêta-Gamma-Delta est meilleure que la première, mais d'une façon parfois négligeable.

6.5. Tests de correction et performance

L'ensemble des méthodes présentées dans ce chapitre est codé dans le programme simulation.ml, écrit dans le langage ocaml et présenté en appendice, section 4 (page 291). Par ce programme nous pouvons tester la correction et la performance (en termes de noeuds élagués), pour différents types de structures d'évaluation (distributive ou non), et pour différentes morphologies des arbres de jeu. Le programme contient, d'une part, l'implantation des évaluations suivantes : *exhaustive* (la fonction *val* de la déf. 5.2.2, page 133), *minimax*, *Alpha-Bêta*, *Alpha-Bêta-Gamma-Delta* (dans ses deux versions) et, d'autre part, l'implantation des méthodes de construction d'arbres aléatoires, en fonction de la profondeur et de la largeur (nombre minium et maximum de fils pour chaque noeud) désirées. Par ce programme, nous avons comparé les évaluations sur trois largeurs possibles :

(1) arbres binaires (toujours 2 fils pour chaque noeud, jusqu'à la profondeur maximale)
(2) arbres ternaires (toujours 3 fils pour chaque noeud, jusqu'à la profondeur maximale)
(3) arbres d'une largeur variable entre 0 et 5 (un nombre de fils compris entre 0 et 5 pour chaque noeud)

Et de façon orthogonale, nous avons choisi des profondeurs d'arbre allant d'un minimum de 3 à un maximum de 17. En combinant les possibles largeurs avec les possibles profondeurs, nous avons obtenus des arbres constitués d'un minimum d'une dizaine de noeuds (profondeur 3), jusqu'à un maximum d'environs 12000 noeuds (largeur $0-5$, profondeur 17).

En ce qui concerne les structures d'évaluation, nous en avons utilisé trois, une première *distributive*, celle des entiers entre 0 et 99 (avec min et max), une deuxième *non distributive*, du domaine $\mathsf{D} = \{\bot, \mathsf{A}, \mathsf{B}, \mathsf{C}, \mathsf{D}, \mathsf{E}, \top\}$ par lequel nous avons fourni l'exemple de non correction de Alpha-Bêta (cf. section 6.4, page 216), et une troisième *non distributive*, celle des intervalles $[a, b]$ construits sur des valeurs entières comprises entre 0 et 99 (avec les opérations $\uparrow =$ *union* et $\downarrow =$ *intersection*).

Ainsi, pour chaque structure d'évaluation, pour chaque largeur d'arbre et pour chaque profondeur d'arbre, nous avons testé le comportement des cinq différentes méthodes, par rapport à la correction de la valeur calculée (l'algorithme exhaustif faisant référence) et par rapport à la performance, i.e. le nombre de noeud visités. Les résultats de cette panoplie de tests sont affichés dans leur intégralité dans les tables 6 et 7, mais les diagrammes des tables 8, 9 et 10 permettent une plus agréable analyse des résultats.

Dans la table 6, on peut notamment constater que l'algorithme Alpha-Bêta (AB) est le seul à ne pas être toujours correct. Mais, d'une manière quelque peu surprenante, ces erreurs restent en nombre très restreint, comme si sa logique de

TAB. 6. Intégralité des tests de correction

Pourcentage des calculs corrects sur le nombre de tests

Modèle	Tests	Largeur	Profondeur	*corrects* minimax	*corrects* AB	*corrects* ABGD	*corrects* ABGD2
entiers 0–99	20000	3	3	100%	100%	100%	100%
entiers 0–99	20000	3	5	100%	100%	100%	100%
entiers 0–99	20000	3	7	100%	100%	100%	100%
entiers 0–99	20000	3	11	100%	100%	100%	100%
entiers 0–99	20000	3	15	100%	100%	100%	100%
entiers 0–99	1000	0–5	3	100%	100%	100%	100%
entiers 0–99	1000	0–5	5	100%	100%	100%	100%
entiers 0–99	1000	0–5	7	100%	100%	100%	100%
entiers 0–99	1000	0–5	11	100%	100%	100%	100%
entiers 0–99	1000	0–5	15	100%	100%	100%	100%
entiers 0–99	1000	0–5	17	100%	100%	100%	100%
simple non distrib.	20000	3	3	100%	99,8%	100%	100%
simple non distrib.	20000	3	5	100%	99,63%	100%	100%
simple non distrib.	20000	3	7	100%	99,65%	100%	100%
simple non distrib.	20000	3	11	100%	99,63%	100%	100%
simple non distrib.	20000	3	15	100%	99,67%	100%	100%
simple non distrib.	20000	2	3	100%	98,81%	100%	100%
simple non distrib.	20000	2	5	100%	98,16%	100%	100%
simple non distrib.	20000	2	7	100%	97,8%	100%	100%
simple non distrib.	20000	2	11	100%	97,38%	100%	100%
simple non distrib.	20000	2	15	100%	97,88%	100%	100%
simple non distrib.	1000	0–5	3	100%	99,3%	100%	100%
simple non distrib.	1000	0–5	5	100%	98,2%	100%	100%
simple non distrib.	1000	0–5	7	100%	97,1%	100%	100%
simple non distrib.	1000	0–5	11	100%	96,8%	100%	100%
simple non distrib.	1000	0–5	15	100%	94,3%	100%	100%
simple non distrib.	1000	0–5	17	100%	96,%	100%	100%
intervalles	20000	3	3	100%	99,74%	100%	100%
intervalles	20000	3	5	100%	99,35%	100%	100%
intervalles	20000	3	7	100%	99,47%	100%	100%
intervalles	20000	3	11	100%	99,39%	100%	100%
intervalles	20000	3	15	100%	99,44%	100%	100%
intervalles	20000	2	3	100%	95,41%	100%	100%
intervalles	20000	2	5	100%	91,32%	100%	100%
intervalles	20000	2	7	100%	90,37%	100%	100%
intervalles	20000	2	11	100%	90,08%	100%	100%
intervalles	20000	2	15	100%	90,37%	100%	100%
intervalles	1000	0–5	3	100%	98,5%	100%	100%
intervalles	1000	0–5	5	100%	94,4%	100%	100%
intervalles	1000	0–5	7	100%	91,7%	100%	100%
intervalles	1000	0–5	11	100%	88,%	100%	100%
intervalles	1000	0–5	15	100%	86,7%	100%	100%
intervalles	1000	0–5	17	100%	86,1%	100%	100%

coupure tenait, dans le cas des structures non distributives, plus à une forte probabilité de correction, qu'à une certitude. Cela explique l'effort non négligeable que nous avons produit pour déterminer un contre-exemple. La table 8 illustre, d'une façon plus évidente, que les fautes de Alpha-Bêta augmentent lorsqu'on utilise la structure, plus riche, des intervalles.

En faveur d'Alpha-Bêta, nous remarquerons, par la table 9, une très significative différence de performance sur la structures distributives des entiers, par rapport aux deux algorithmes Alpha-Bêta-Gamma-Delta (ABGD et ABGD2). Donc, nous ne pouvons pas espérer remplacer les deux théorèmes de correction par un seul

TAB. 7. Intégralité des tests de performance

Moyenne des noeuds visités sur l'ensemble des tests

Modèle	Tests	Largeur	Profondeur	visités minimax	visités AB	visités ABGD	visités ABGD2
entiers 0–99	20000	3	3	98,41%	69,9%	69,9%	69,9%
entiers 0–99	20000	3	5	97,29%	49,82%	50,98%	50,98%
entiers 0–99	20000	3	7	96,49%	37,65%	39,39%	39,39%
entiers 0–99	20000	3	11	94,82%	24,7%	26,23%	26,23%
entiers 0–99	20000	3	15	92,6%	18,24%	19,52%	19,52%
entiers 0–99	1000	0–5	3	98,38%	76,93%	76,93%	76,93%
entiers 0–99	1000	0–5	5	98,23%	59,48%	64,8%	64,8%
entiers 0–99	1000	0–5	7	97,7%	44,45%	53,89%	53,89%
entiers 0–99	1000	0–5	11	95,94%	22,96%	36,73%	36,73%
entiers 0–99	1000	0–5	15	95,1%	10,54%	23,71%	23,71%
entiers 0–99	1000	0–5	17	94,6%	7,01%	18,79%	18,79%
simple non distrib.	20000	3	3	79,7%	67,48%	68,34%	68,34%
simple non distrib.	20000	3	5	69,42%	48,41%	52,74%	52,74%
simple non distrib.	20000	3	7	61,3%	36,41%	42,13%	42,13%
simple non distrib.	20000	3	11	48,2%	23,71%	29,27%	29,27%
simple non distrib.	20000	3	15	39,27%	17,57%	22,15%	22,15%
simple non distrib.	20000	2	3	83,01%	74,44%	74,05%	74,05%
simple non distrib.	20000	2	5	72,91%	54,61%	55,86%	55,54%
simple non distrib.	20000	2	7	64,03%	41,51%	43,52%	42,84%
simple non distrib.	20000	2	11	50,33%	27,35%	29,36%	28,35%
simple non distrib.	20000	2	15	40,91%	20,15%	21,83%	20,97%
simple non distrib.	1000	0–5	3	82,46%	70,97%	70,64%	70,64%
simple non distrib.	1000	0–5	5	75,64%	55,83%	58,3%	58,29%
simple non distrib.	1000	0–5	7	69,4%	42,72%	47,43%	47,41%
simple non distrib.	1000	0–5	11	57,72%	22,34%	30,44%	30,35%
simple non distrib.	1000	0–5	15	46,66%	11,01%	19,15%	19,09%
simple non distrib.	1000	0–5	17	41,19%	7,17%	14,16%	14,11%
intervalles	20000	3	3	99,98%	82,63%	86,38%	86,38%
intervalles	20000	3	5	99,99%	60,94%	78,01%	78,01%
intervalles	20000	3	7	99,97%	45,68%	70,58%	70,58%
intervalles	20000	3	11	99,97%	29,66%	58,01%	58,01%
intervalles	20000	3	15	99,96%	22,14%	49,05%	49,05%
intervalles	20000	2	3	99,99%	93,38%	90,59%	90,59%
intervalles	20000	2	5	99,98%	72,96%	72,61%	71,83%
intervalles	20000	2	7	99,98%	56,04%	58,52%	56,74%
intervalles	20000	2	11	99,97%	36,88%	41,06%	37,92%
intervalles	20000	2	15	99,97%	27,17%	31,29%	28,17%
intervalles	1000	0–5	3	93,59%	76,61%	76,98%	76,98%
intervalles	1000	0–5	5	90,63%	64,7%	67,29%	67,26%
intervalles	1000	0–5	7	86,79%	52,66%	58,47%	58,41%
intervalles	1000	0–5	11	78,35%	28,54%	41,24%	41,09%
intervalles	1000	0–5	15	69,25%	15,05%	29,41%	29,25%
intervalles	1000	0–5	17	65,07%	9,62%	23,92%	23,77%

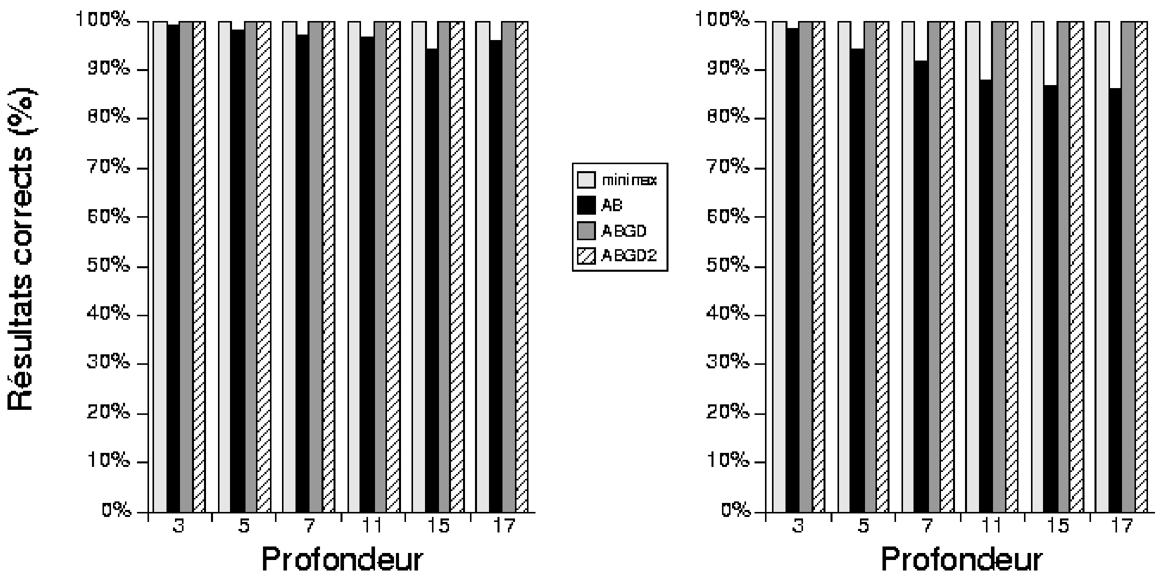
théorème, celui de la méthode Alpha-Bêta-Gamma-Delta, cette dernière ne coïncidant pas avec Alpha-Bêta sur les structures distributives. Comme on pouvait s'y attendre, le seuil entre Alpha-Bêta et les autres méthodes se creuse d'une façon importante lorsque les noeuds ont plusieurs fils. En faveur, cette fois ci, des méthodes Alpha-Bêta-Gamma-Delta, nous remarquerons que, avec Alpha-Bêta, les trois méthodes sont de loin supérieures à la simple méthode minimax, dont la capacité d'élagage est intéressante seulement lorsque la structure est simple, c'est-à-dire lorsque il y a plus de chances d'atteindre les valeurs maximum et minimum (cf. table 10).

TAB. 8. Test de correction sur des structures non distributives

Sur un échantillon de 1000 arbres de jeu
de largeur comprise entre 0 et 5

Structure simple non distributive

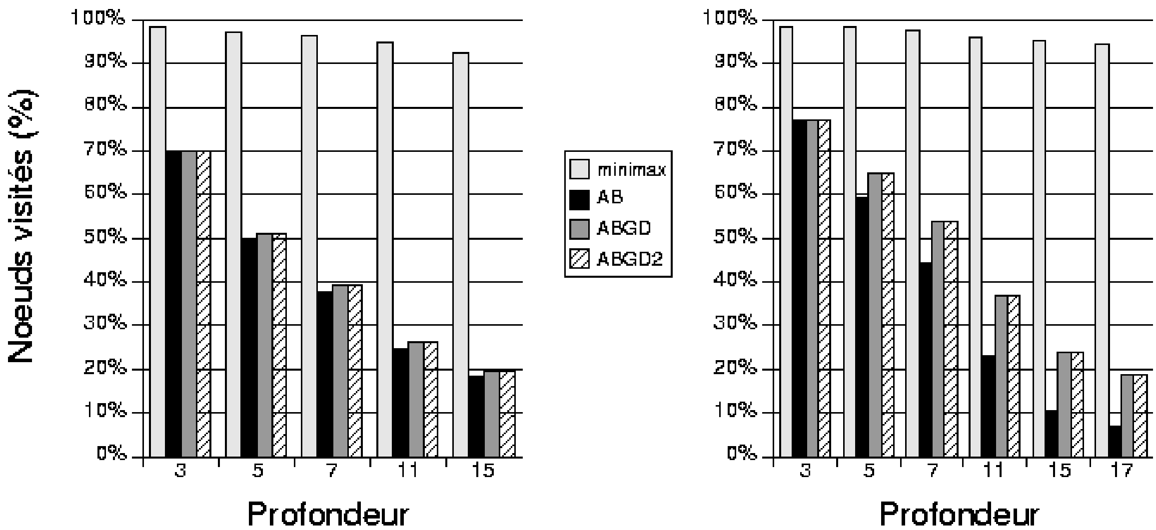
Structure des intervalles



TAB. 9. Test de performance sur la structure distributive des entiers entre 0 et 99

Échantillon de 20000 arbres de jeu
de largeur 3

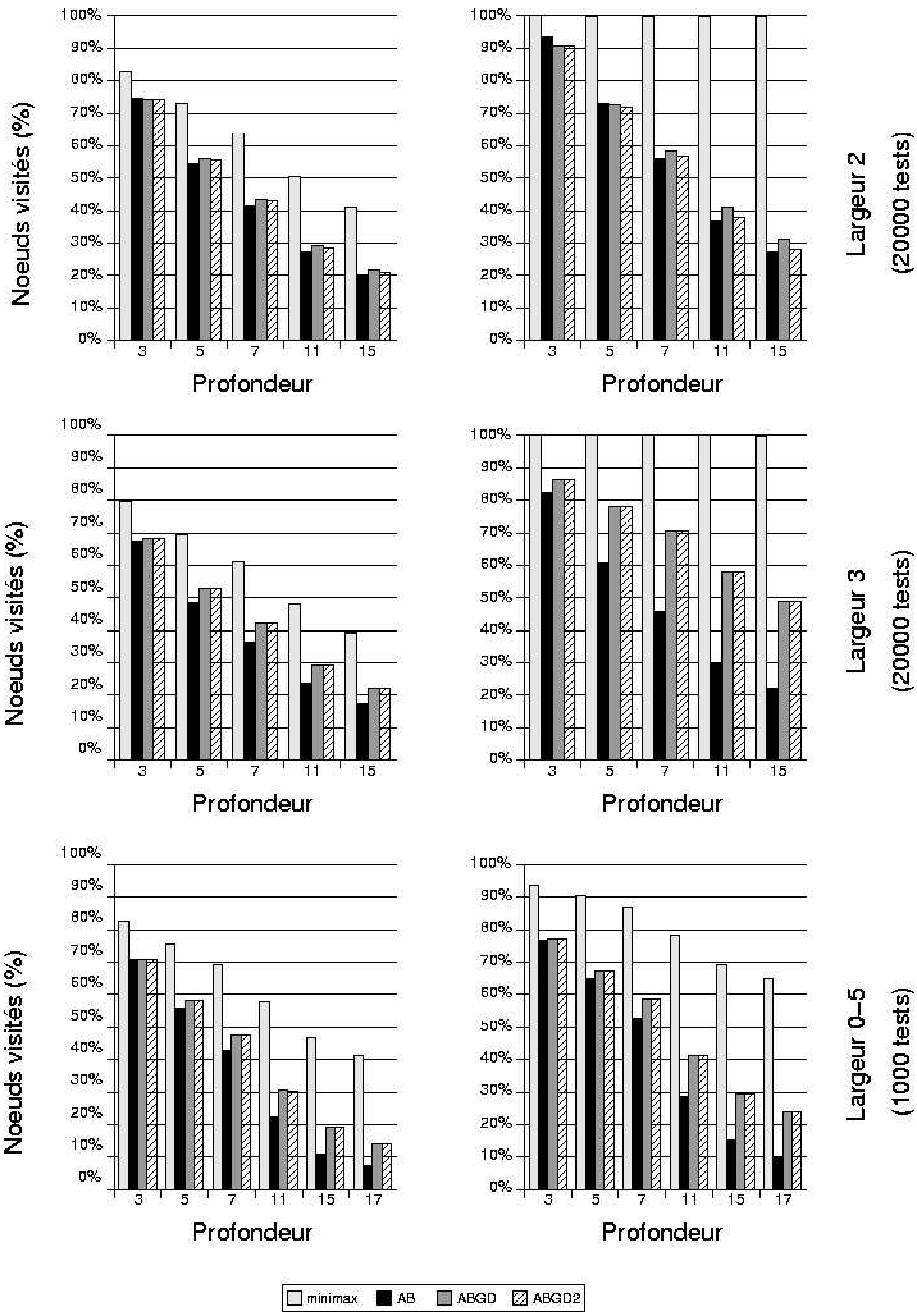
Échantillon de 1000 arbres de jeu
de largeur comprise entre 0 et 5



Enfin, en ce qui concerne le rapport entre les deux versions de l'algorithme Alpha-Bêta-Gamma-Delta, nous remarquerons, toujours par la table 10, que la deuxième version est, certes, plus performante, mais la différence devient sensible, comme on pouvait s'y attendre, seulement lorsque il s'agit d'arbres binaires. En effet, si la distinction se fait sur le traitement du dernier fils de chaque noeud, sur les arbres binaires cela représentera la moitié du calcul nécessaire.

TAB. 10. Test de performance sur des structure non distributives

Structure simple non distributive Structure des intervalles



Troisième partie

La Programmation Logique par les jeux combinatoires

CHAPITRE 7

Langages logiques du premier ordre

Ce chapitre a pour objectif de rappeler les notions fondamentales, utiles par la suite, de la théorie des langages logiques du premier ordre, à la base de la programmation logique pure (LP) et de celle avec contraintes (CLP). La présentation des résultats n'est certainement pas exhaustive pour un sujet de recherche si étendu, et les notations sont parfois légérement décalées par rapport à celles que l'on retrouve en littérature, dans un souci de rigueur et de cohérence avec les notations et les concepts élaborés au cours des chapitres précédents, notamment celui sur les algèbres typées.

7.1. Algèbres logiques

L'algèbre de Boole est la base sur laquelle est forgée la sémantique des langages logiques du premier ordre. Dans cette algèbre spécifique il n'y a pas de sortes, mais seulement des prédicats, représentant les opérations basiques sur les valeurs de vérité (de l'ensemble $2 = \{0, 1\}$).

DEFINITION 7.1.1. (ALGÈBRE DE BOOLE) *L'algèbre de Boole est l'algèbre sur la signature* $(\cdot^{\mathsf{A}} : \mathcal{B} \to \mathcal{B}^{\mathsf{A}} \oplus \mathsf{S} \to \mathsf{S}^{\mathsf{A}} \oplus \mathsf{F} \to \mathsf{F}^{\mathsf{A}}, \mathsf{T})$ *où* $2 \in \mathcal{B}$, $2^{\mathsf{A}} = \{0, 1\}$, $\mathsf{S} = \emptyset$, *et les opérations, appelées* opérateurs logiques *ou* booléens, *sont* $\mathsf{F} = \{\wedge, \vee, \Rightarrow, \Leftrightarrow, \neg, 0, 1\}$. *Les opérateurs logiques, qui sont toujours de type* $2 \times 2 \to 2$, *à l'exception de* $\neg$, *de type* $2 \to 2$ *et des constantes* 0 *et* 1 *(identiques, avec un abus de notation, aux valeurs de* 2^{A}*) de type* $\to 2$, *sont définis par les* tables de vérité *:*

(1) $0 \triangleq \{(\bullet, 0)\}$

(2) $1 \triangleq \{(\bullet, 1)\}$

(3) $\neg \triangleq \{(0, 1), (1, 0)\}$

(4) $\vee \triangleq \{(0,0,0),(0,1,1),(1,0,1),(1,1,1)\}$

(5) $\wedge \triangleq \{(0,0,0),(0,1,0),(1,0,0),(1,1,1)\}$

(6) $\Rightarrow \triangleq \{(0,0,1),(0,1,1),(1,0,0),(1,1,1)\}$

(7) $\Leftrightarrow \triangleq \{(0,0,1),(0,1,0),(1,0,0),(1,1,1)\}$

□

Les algèbres sémantiques sont des algèbres sur une signature *arborescente* particulière (cf. section 3.3.2). D'une part, comme toute algèbre de signature arborescente, elles contiennent des fonctions (interprétant les constructeurs de termes) restituant des valeurs sortées et, d'autre part, elles contiennent des prédicats logiques (interprétant les constructeurs de primitives) sur les valeurs sortées. Par la spécificité des opérations primitives, nous donnerons le nom d'*algèbre logique* à ce type d'algèbres et le nom de *signature logique* à leur signature.

Definition 7.1.2. (Algèbre logique, signature logique) *Une algèbre* $(\cdot^{A} : \mathcal{B} \to \mathcal{B}^{A} \oplus \mathsf{S} \to \mathsf{S}^{A} \oplus \mathsf{F} \to \mathsf{F}^{A}, \mathsf{T})$ *est une* algèbre logique *si les conditions suivantes sont vérifiées :*

(1) *la signature de l'algèbre est une signature arborescente*

l'ensemble des symbole est donc classifié en constructeurs de termes et constructeurs de primitives : $\mathsf{F} = \mathsf{F}_{\mathsf{S}} \oplus \mathsf{F}_{\mathcal{B}}$

(2) *les primitives de* $\mathsf{F}_{\mathcal{B}}$ *sont des* prédicats logiques

le domaine de base $2 \in \mathcal{B}$ *est le seul codomaine possible, avec les sortes, pour toute opération de l'algèbre :*

$$p \in \mathsf{F}_{\mathcal{B}} \quad \Rightarrow \quad \mathrm{cod}_{\mathsf{T}}(p) = 2$$

Les signatures logiques *sont les signatures des algèbres logiques. La famille des algèbres logiques sur une signature* Σ *sera notée* $\mathsf{LAlg}(\Sigma)$. □

En d'autres termes, une signature logique est une signature où les constructeurs de termes $f \in \mathsf{F}_{\mathsf{S}}$ et les constructeurs de primitives $p \in \mathsf{F}_{\mathcal{B}}$, ont associé un type qui appartient au langage des types clos générés par le symbole non terminal τ de la syntaxe suivante :

$$\begin{array}{lcl}
\tau & ::= & \forall \alpha{:}\chi.\,\tau \mid \delta \to \vartheta \\
\delta & ::= & 1 \mid \eta \mid \eta_1 \times \cdots \times \eta_n \mid \eta^* \\
\eta & ::= & \sigma \mid \alpha \\
\vartheta & ::= & 2 \mid \sigma \\
\chi & ::= & \mathsf{sort} \mid \{\sigma_1, \cdots, \sigma_n\}
\end{array}$$

où $\sigma \in \mathsf{S}$, $\alpha \in \mathcal{V}$, et $\mathcal{V}$ est un ensemble de variables sortées disjoint de $\mathcal{B}$, S et F.

TAB. 1. Variables libres d'une formule $\breve{\varphi} \in \breve{\Phi}(\Sigma, V)$

$$\mathsf{FV}(\breve{\varphi}) \triangleq \begin{cases} \mathrm{vars}(\pi) & \text{si} \quad \breve{\varphi} = \pi \\ \emptyset & \text{si} \quad \breve{\varphi} = \mathrm{false} \\ \mathsf{FV}(\breve{\varphi}') & \text{si} \quad \breve{\varphi} = \neg\breve{\varphi}' \\ \mathsf{FV}(\breve{\varphi}_1) \cup \mathsf{FV}(\breve{\varphi}_2) & \text{si} \quad \breve{\varphi} = \breve{\varphi}_1 \vee \breve{\varphi}_2 \quad \text{ou} \quad \breve{\varphi} = \breve{\varphi}_1 \wedge \breve{\varphi}_2 \\ \bigcup_{j \in I} \mathsf{FV}(\phi(j)) & \text{si} \quad \breve{\varphi} = \bigvee_{i \in I} \phi_{(i)} \quad \text{ou} \quad \breve{\varphi} = \bigwedge_{i \in I} \phi_{(i)} \\ \mathsf{FV}(\breve{\varphi}_1) \cup \mathsf{FV}(\breve{\varphi}_2) & \text{si} \quad \breve{\varphi} = \breve{\varphi}_1 \Rightarrow \breve{\varphi}_2 \quad \text{ou} \quad \breve{\varphi} = \breve{\varphi}_1 \Rightarrow \breve{\varphi}_2 \\ \mathsf{FV}(\breve{\varphi}) \setminus \{X\} & \text{si} \quad \breve{\varphi} = \forall X.\, \breve{\varphi}' \quad \text{ou} \quad \breve{\varphi} = \exists X.\, \breve{\varphi}' \end{cases}$$

EXAMPLE 7.1.3. Considérons l'algèbre logique constituée de l'unique sorte des entiers naturels $S = \{\mathbb{N}\}$ avec l'opération *somme* $+ : \mathbb{N} \times \mathbb{N} \to \mathbb{N}$, l'opération *produit* $* : \mathbb{N} \times \mathbb{N} \to \mathbb{N}$, la constante *zéro* $0 :\to \mathbb{N}$, la fonction *successeur* $\mathsf{s} : \mathbb{N} \to \mathbb{N}$ et les prédicats *pair* $\mathrm{even} : \mathbb{N} \to 2$, *impair* $\mathrm{odd} : \mathbb{N} \to 2$, *premier* $\mathrm{prime} : \mathbb{N} \to 2$, *multiple* $\mathrm{multiple} : \mathbb{N} \times \mathbb{N} \to 2$, *plus grand diviseur commun* $\mathrm{gcd} : \mathbb{N} \times \mathbb{N} \times \mathbb{N} \to 2$. L'ensemble des constructeurs de termes est alors $\mathsf{F}_S = \{+, *, 0, \mathsf{s}\}$, tandis que l'ensemble des constructeurs de primitives est $\mathsf{F}_{\mathcal{B}} = \{\mathrm{even}, \mathrm{odd}, \mathrm{premier}, \mathrm{multiple}, \mathrm{gcd}\}$. L'arbre $+(*(\mathsf{s}(0, \mathsf{s}(\mathsf{s}(0)))), 0)$ est un terme, l'arbre $\mathrm{even}(+(*(\mathsf{s}(0, \mathsf{s}(\mathsf{s}(0)))), 0))$ est une primitive. L'arbre $+(*, \mathrm{even})$ est un arbre de $\mathrm{T}(\mathsf{F})$ mais n'est ni un terme ni une primitive. $\square$

EXAMPLE 7.1.4. Supposons d'ajouter à l'algèbre de l'exemple précédent la sorte $\mathbb{Z}$ des entiers relatifs, les opérations de conversion $\mathrm{int\_of\_nat} : \mathbb{N} \to \mathbb{Z}$ et $\mathrm{abs} : \mathbb{Z} \to \mathbb{N}$, et l'égalité polymorphe $=: \forall\alpha : \mathsf{sort}.\, \alpha \times \alpha \to 2$. Dans cette hypothèse, l'arbre $\mathrm{int\_of\_nat}(+(*(\mathsf{s}(0, \mathsf{s}(\mathsf{s}(0)))), 0))$ est un terme, et l'arbre $=(0, \mathsf{s}(0))$ est une primitive. En revanche, par incompatibilité des types, l'arbre $\mathrm{abs}(0)$ n'est pas un terme, et l'arbre $=(0, \mathrm{int\_of\_nat}(0))$ n'est pas une primitive. $\square$

7.2. Formules logiques du premier ordre

Défini par une syntaxe formelle, tout langage des *formules logiques du premier ordre* (basé sur une signature logique Σ et une ensemble de variables sortées V) est équipé de la fonction d'interprétation sémantique dans les valeurs de vérité.

7.2.1. Syntaxe.

7.2.1. Syntaxe. Étant donné une signature logique $\Sigma = (\mathcal{B}, S, \mathsf{F}, \mathsf{T})$, et un ensemble $V = \biguplus_{s \in S} V_s$ de symboles de variables sortées disjoint de $\mathcal{B}$, S et F, le *langage des formules logiques du premier ordre* sur Σ et V, noté $\Phi(\Sigma, V)$, est le langage des formules closes (sans variables libres) du langage $\breve{\Phi}(\Sigma, V)$ généré par le symbole non terminal φ de la syntaxe suivante :

$$
\begin{array}{rcll}
\varphi & ::= & \pi & \text{(atome avec variables)} \\
& | & \mathsf{false} & \text{(le faux)} \\
& | & \neg\varphi & \text{(négation)} \\
& | & \varphi_1 \vee \varphi_2 & \text{(disjonction)} \\
& | & \varphi_1 \wedge \varphi_2 & \text{(conjonction)} \\
& | & \bigvee_{i\in I} \phi_{(i)} & \text{(disjonction arbitraire)} \\
& | & \bigwedge_{i\in I} \phi_{(i)} & \text{(conjonction arbitraire)} \\
& | & \varphi_1 \Rightarrow \varphi_2 & \text{(implication)} \\
& | & \varphi_1 \Leftrightarrow \varphi_2 & \text{(équivalence)} \\
& | & \forall X.\, \varphi & \text{(quantification universelle)} \\
& | & \exists X.\, \varphi & \text{(quantification existentielle)}
\end{array}
$$

où $X \in V$, $I \subseteq \mathbb{N}$, $\phi : \mathbb{N} \rightharpoonup \Phi(\Sigma, V)$, et $\pi \in T(\Sigma{+}V, 2)$ est une primitive de type 2 (un prédicat) de la signature logique augmentée par V. Dans les disjonctions et conjonctions arbitraires $\bigvee_{i\in I} \phi_{(i)}$ et $\bigwedge_{i\in I} \phi_{(i)}$, on supposera toujours $\mathrm{dom}(\phi) = I$. L'ensemble des variables libres d'une formule est défini de la façon habituelle dans la table 1. Par définition $\Phi(\Sigma, V) = \{\varphi \in \breve{\Phi}(\Sigma, V) \mid \mathsf{FV}(\varphi) = \emptyset\}$. Pour tout $\breve{\varphi} \in \breve{\Phi}(\Sigma, V)$, la clôture existentielle, notée $\exists\breve{\varphi}$, et la clôture universelle, notée $\forall\breve{\varphi}$, dénoterons respectivement les formules closes $\exists X_1..\exists X_n.\, \breve{\varphi}$ et $\forall X_1..\forall X_n.\, \breve{\varphi}$, où $\{X_1, .., X_n\} = \mathsf{FV}(\breve{\varphi})$. Les clôtures sont uniques modulo les permutations (qui ne changent pas le sens de la formule) des variables $X_1,..,X_n$.

7.2.2. Sémantique. La sémantique d'une formule logique est une valeur de vérité qui dépend de l'algèbre logique dans laquelle on interprète les primitives (les atomes) contenus dans la formule. Toute primitive p est un arbre ayant à la racine un symbole de prédicat et d'éventuels termes comme sous-arbres fils. Pour obtenir la sémantique des formules logiques il suffit d'étendre l'interprétation canonique $[\![.]\!]_A : \forall\alpha : \mathsf{S}.\, \alpha \to \alpha$ des termes de la signature Σ vers une algèbre quelconque $A \in \mathsf{Alg}(\Sigma)$ (cf. section 3.3.5), en s'appuyant, pour les formules composites, sur les opérateurs $\neg$, $\vee$, $\wedge$, $\Rightarrow$ et $\Leftrightarrow$ de l'algèbre de Boole.

Étant donné une algèbre A, pour donner un sens aux formules non closes, il est nécessaire de pouvoir interpréter les variables libres. La notion d *'environnement* (cf. définition 3.3.14), qui est une quelconque fonction $\xi : \forall\alpha : \mathsf{S}.\, V_\alpha \to \alpha^A$ affectant à toute variable sortée $X \in V_s$ une dénotation $\xi(X)$ dans le domaine algèbrique s^A, permet d'évaluer les formules contenant des variables libres. On notera $\xi[{}^a\!/\!_X]$ l'environnement ξ modifié par l'association de X à la valeur a :

$$\xi[{}^a\!/\!_X](Y) \triangleq \begin{cases} a & \text{si} \quad Y = X \\ \xi(Y) & \text{si} \quad Y \neq X \end{cases}$$

Comme pour les termes avec variables, tout couple (A, ξ), avec $\xi \in \mathrm{ENV}_A$, définit complétement l'interprétation sémantique des formules et sera appelé *interprétation implicite* des formules. L'ensemble des interprétations implicites des formules sur la signature Σ, sera noté $\mathsf{IAlg}(\Sigma, V)$:

$$\mathsf{IAlg}(\Sigma, V) \triangleq \{(A, \xi) \mid A \in \mathsf{LAlg}(\Sigma),\ \xi \in \mathrm{ENV}_A\}$$

Les équations de la table 2 définissent, de façon inductive, l'interprétation sémantique $[\![\varphi]\!]_{A,\xi}$ d'une formule quelconque, close ou non :

$$[\![.]\!] : \breve{\Phi}(\Sigma, V) \to \mathsf{IAlg}(\Sigma, V) \to 2$$

TAB. 2. Interprétation sémantique des formules logiques du premier ordre $\varphi \in \breve{\Phi}(\Sigma, V)$

$$
\begin{array}{rcl}
[\![\pi]\!]_{A,\xi} & = & p^A([\![t_1]\!]_{A,\xi},..,[\![t_n]\!]_{A,\xi}) \quad \text{où } \pi = p(t_1,...,t_n) \\
[\![\text{false}]\!]_{A,\xi} & = & 0 \\
[\![\neg\varphi]\!]_{A,\xi} & = & \neg[\![\varphi]\!]_{A,\xi} \\
[\![\varphi_1 \vee \varphi_2]\!]_{A,\xi} & = & [\![\varphi_1]\!]_{A,\xi} \vee [\![\varphi_2]\!]_{A,\xi} \\
[\![\varphi_1 \wedge \varphi_2]\!]_{A,\xi} & = & [\![\varphi_1]\!]_{A,\xi} \wedge [\![\varphi_2]\!]_{A,\xi} \\
[\![\bigvee_{i\in I} \varphi_i]\!]_{A,\xi} & = & \begin{cases} 1 & \text{si} \quad \text{il existe } i \in I \text{ tel que } [\![\varphi_i]\!]_{A,\xi} = 1 \\ 0 & \text{sinon} \end{cases} \\
[\![\bigwedge_{i\in I} \varphi_i]\!]_{A,\xi} & = & \begin{cases} 0 & \text{si} \quad \text{il existe } i \in I \text{ tel que } [\![\varphi_i]\!]_{A,\xi} = 0 \\ 1 & \text{sinon} \end{cases} \\
[\![\varphi_1 \Rightarrow \varphi_2]\!]_{A,\xi} & = & [\![\varphi_1]\!]_{A,\xi} \Rightarrow [\![\varphi_2]\!]_{A,\xi} \\
[\![\varphi_1 \Leftrightarrow \varphi_2]\!]_{A,\xi} & = & [\![\varphi_1]\!]_{A,\xi} \Leftrightarrow [\![\varphi_2]\!]_{A,\xi} \\
[\![\forall X.\, \varphi]\!]_{A,\xi} & = & \begin{cases} 1 & \text{si} \quad [\![\varphi]\!]_{A,\xi[a/X]} = 1 \;\text{ pour tout } a \in \sigma^A, \text{ où } X \in V_\sigma \\ 0 & \text{sinon} \end{cases} \\
[\![\exists X.\, \varphi]\!]_{A,\xi} & = & \begin{cases} 1 & \text{si} \quad [\![\varphi]\!]_{A,\xi[a/X]} = 1 \text{ pour au moins un } a \in \sigma^A, \text{ où } X \in V_\sigma \\ 0 & \text{sinon} \end{cases}
\end{array}
$$

Autrement dit, pour donner un sens vrai ou faux à une formule (close ou non), il faudra une algèbre logique A sur la même signature Σ et un environnement à valeurs dans A pour interpréter les variables libres. On prouve facilement (par induction, cf. [**Asperti & Ciabattoni, 1997**] par exemple) que la sémantique d'une formule close ne dépend pas de l'environnement utilisé pour son évaluation. Ainsi, la sémantique d'une formule close φ dans une algèbre logique A sera notée de la même façon que l'interprétation canonique des termes $[\![t]\!]_A$, par l'écriture $[\![\varphi]\!]_A$:

$$[\![.]\!] : \Phi(\Sigma, V) \to \mathsf{LAlg}(\Sigma) \to 2$$

7.2.2.1. *Classification des algèbres logiques.* Étant donné la signature logique Σ, augmentée des variables V, toute formule logique $\varphi \in \Phi(\Sigma, V)$ partage, par le biais de la fonction sémantique $[\![.]\!]$, les algèbres logiques $A \in \mathsf{LAlg}(\Sigma)$ en deux classes disjointes : celles qui rendent vraie la formule, $[\![\varphi]\!]_A = 1$ (noté le plus souvent $A \models \varphi$), et celle qui la rendent fausse, $[\![\varphi]\!]_A = 0$ (noté $A \not\models \varphi$). Les premières seront appelées *modèles* de φ. Le sous-ensemble de $\mathsf{LAlg}(\Sigma)$ constitué des modèles d'une formule φ sera noté $\mathsf{LAlg}(\Sigma)_\varphi$:

$$\mathsf{LAlg}(\Sigma)_\varphi \triangleq \{A \in \mathsf{LAlg}(\Sigma) \mid A \models \varphi\}$$

7.2.2.2. *Classification des formules logiques.* Réciproquement, toute algèbre logique $A \in \mathsf{LAlg}(\Sigma)$ partitionne l'ensemble des formules $\Phi(\Sigma, V)$ en deux ensembles disjoints : celui des formules fausses dans A, et celui des formules vraies dans A, que nous noterons $\Phi(\Sigma, V)_A$:

$$\Phi(\Sigma, V)_A = \{\varphi \in \Phi(\Sigma + V) \mid [\![\varphi]\!]_A = 1\}$$

Les formules sont aussi classifiées en fonction des modèles qu'elles possèdent. Nous disons qu'une formule logique $\varphi \in \Phi(\Sigma, V)$ est :

valide si elle est toujours vraie, indépendemment de l'algèbre logique qui l'interprète :

$$\mathsf{LAlg}(\Sigma)_\varphi = \mathsf{LAlg}(\Sigma)$$

On dit aussi que φ est une *tautologie*, noté $\models \varphi$.

inconsistante si elle n'est jamais vraie, pour aucune algèbre logique :

$$\mathsf{LAlg}(\Sigma)_\varphi = \emptyset$$

satisfiable (ou *consistante*) si elle est vraie pour au moins une algèbre logique :

$$\exists\, A \in \mathsf{LAlg}(\Sigma).\ A \models \varphi$$

Autrement dit, si $\mathsf{LAlg}(\Sigma)_\varphi \neq \emptyset$

contingente si elle est vrai pour au moins une algèbre logique et fausse pour au moins une autre :

$$\exists\, A_1, A_2 \in \mathsf{LAlg}(\Sigma).\ A_1 \models \varphi\ \wedge\ A_2 \not\models \varphi$$

Autrement dit, si $\emptyset \subset \mathsf{LAlg}(\Sigma)_\varphi \subset \mathsf{LAlg}(\Sigma)$

Les notions de validité et d'inconsistance sont duales. On peut facilement prouver que la négation d'une formule traduit une condition dans l'autre :

φ valide	ssi	$\neg\varphi$ inconsistente
φ inconsistente	ssi	$\neg\varphi$ valide

7.2.2.3. *Conséquences logiques.* Il existent des couples de formules φ_1 et φ_2 qui entretiennent un rapport de subordination : chaque fois que la première est vraie dans une algèbre, alors l'autre aussi est vraie dans la même algèbre. Plus formellement, la formule φ_2 est *conséquence logique* de φ_1 si :

$$\mathsf{LAlg}(\Sigma)_{\varphi_1} \subseteq \mathsf{LAlg}(\Sigma)_{\varphi_2}$$

La notion de conséquence logique peut s'exprimer par celle de validité, donc par celle d'inconsistance :

φ_2 conséquence logique de φ_1	ssi	$\varphi_1 \Rightarrow \varphi_2$ valide
	ssi	$\neg\varphi_1 \vee \varphi_2$ valide
	ssi	$\varphi_1 \wedge \neg\varphi_2$ inconsistente

7.3. Ensembles de formules logiques du premier ordre

Les notions de validité, inconsistence, consistance sont étendues aux ensembles de formules. Un ensemble de formules $\mathcal{F} \in \wp(\Phi(\Sigma, V))$ est :

- *valide* si toutes les formules $\varphi \in \mathcal{F}$ sont valides
- *satisfiable* s'il existe une algèbre A qui est un modèle pour toutes les formules de $\mathcal{F}$
- *inconsistente* si elle n'est pas satisfiable
- *contingente* si elle est satisfiable mais pas valide

On dit qu'une algèbre A est un modèle de $\mathcal{F}$, noté $A \models \mathcal{F}$, si A est un modèle de toutes les formules de $\mathcal{F}$. On dit qu'une formule φ est *conséquence logique de l'ensemble* $\mathcal{F}$ lorsque tout modèle de $\mathcal{F}$ est un modèle de φ. La condition de conséquence logique d'un ensemble est notée, avec un abus de notation, $\mathcal{F} \models \varphi$. L'ensemble des tous les modèles de $\mathcal{F}$ est noté $\mathsf{LAlg}(\Sigma)_{\mathcal{F}}$. À nouveau, on peut traduire le problème de déterminer si une formule est conséquence logique d'un ensemble $\mathcal{F}$, en terme de validité ou d'inconsistance (cf. par exemple [**Asperti & Ciabattoni, 1997**]) :

$$\varphi \text{ conséquence logique de } \mathcal{F} \qquad \text{ssi} \qquad \mathcal{F} \cup \{\neg\varphi\} \text{ inconsistente}$$

Déterminer si une formule est conséquence logique d'un ensemble de formules $\mathcal{F}$ revient au problème de décider l'inconsistance d'un autre ensemble, précisément l'ensemble initial augmenté de la négation de la formule. Puisqu'il existe une infinité d'algèbres logiques, nous ne pouvons pas établir de façon directe, en les essayant toutes, si un ensemble est inconsistent ou pas (tout comme nous ne pouvons pas établir directement s'il est valide ou pas). Le problème est de nature indécidable mais, toutefois, semi-décidable. Le théorème de Skolem nous achemine vers la solution, forcément partielle, du problème.

7.3.1. Clauses. Les clauses constituent un sous-langage des formules logiques du premier ordre. Il s'agit d'une disjonction d'atomes ou de négations d'atomes, éventuellement quantifiés universellement. Plus formellement, étant donné une signature logique $\Sigma = (\mathcal{B}, S, F, T)$ et un ensemble $V = \biguplus_{s \in S} V_s$ de symboles de variables sortées disjoint de $\mathcal{B}$, S et F, les clauses sont générées par le symbole non terminal ψ de la syntaxe suivante :

$$\begin{array}{rcll} \psi & ::= & \forall X.\, \psi & \text{(clause)} \\ & | & \breve{\psi} & \text{(clause ouverte)} \\ \breve{\psi} & ::= & \pi & \text{(littéral positif)} \\ & | & \neg\pi & \text{(littéral négatif)} \\ & | & \mathsf{false} & \text{(le faux)} \\ & | & \breve{\psi}_1 \vee \breve{\psi}_2 & \text{(disjonction)} \end{array}$$

où $\sigma \in S$, $X \in V$, et $\pi \in T(\Sigma{+}V, 2)$. Nous noterons $\Psi(\Sigma, V)$ l'ensemble des clauses sur la signature logique Σ et l'ensemble des variables V. Les clauses sont supposées, comme les formules logiques, closes, i.e. sans variables libres. En effet, on écrit le plus souvent les clauses sans les quantificateurs en les laissant implicites (puisqu'il n'y a qu'un seul type de quantificateur, l'universel, et il se trouvent tous au début de la formule).

7.3.1.1. *Théorème de Skolem.* Il existe une transformation appelée *skolémisation,* skolem : $\Phi(\Sigma, V) \to \Psi(\Sigma', V)$, telle que pour tout ensemble de formules du premier ordre $\mathcal{F}$, l'ensemble est inconsistent si et seulement si l'image de la transformation de skolem sur $\mathcal{F}$ est un ensemble (de clauses) inconsistent :

$$\mathcal{F} \text{ inconsistent} \quad \text{ssi} \quad \{\text{Skolem}(\varphi) \mid \varphi \in \mathcal{F}\} \text{ inconsistent}$$

Les symboles de la signature Σ' n'étant pas contenus dans Σ sont appelés *fonctions de Skolem.* Le problème de décider de l'inconsistance d'un ensemble de formules peut ainsi être limité aux ensembles de clauses.

7.3.1.2. *Théorème de Herbrand.* Il existe une classe d'algèbres logiques, précisément celle des algèbres des termes finis (cf. définition 3.3.4) sur la signature logique Σ donnée, qui contient forcément un modèle de tout ensemble de clauses, si ce dernièr n'est pas inconsistent. En littérature, les algèbres des termes $\mathsf{TAlg}(\Sigma)$ sont appelées *interprétations de Herbrand* lorsque la signature Σ est une signature logique. Par la suite, nous les appellerons *algèbres (des termes) de Herbrand* pour éviter toute ambiguïté avec la notion d'interprétation algébrique.

THEOREM 7.3.1. (HERBRAND) *Si un ensemble de clauses $\mathcal{F} \subseteq \Psi(\Sigma, V)$ est consistant alors il possède au moins une algèbre des termes de Herbrand parmi ses modèles* □

Un *modèle de Herbrand* sera, à la fois, une algèbre des termes de Herbrand et un modèle pour une formule ou un ensemble de formules. Le théorème de Herbrand permet d'affirmer qu'un ensemble de clauses $\mathcal{F}$ est inconsistent si et seulement si il n'a pas de modèle de Herbrand :

$$\mathsf{LAlg}(\Sigma)_{\mathcal{F}} = \emptyset \quad \text{ssi} \quad \mathsf{TAlg}(\Sigma)_{\mathcal{F}} = \emptyset$$

Et donc, par le théorème de Skolem, il permet d'affirmer qu'un ensemble de formules logiques du premier ordre est inconsistent si et seulement si sa skolémisation n'a pas de modèles de Herbrand :

$$\mathsf{LAlg}(\Sigma)_{\mathcal{F}} = \emptyset \quad \text{ssi} \quad \mathsf{TAlg}(\Sigma')_{\text{Skolem}(\mathcal{F})} = \emptyset$$

où Σ' est la signature comprenant les fonctions de Skolem.

7.3.2. Principe de résolution. La résolution est un principe de transformation des formules permettant de déduire à l'occasion (le problème étant de nature semi-décidable) l'inconsistance d'un ensemble de formules. La remarque fondamentale, très simple, est qu'un ensemble de formules est inconsistent si et seulement si on peut en déduire le faux :

$$\mathcal{F} \text{ inconsistent} \quad \text{ssi} \quad \mathcal{F} \models \text{false}$$

En effet, ($\mathcal{F} \models$ false) signifie ($\forall A \in \mathsf{LAlg}(\Sigma).\ A \models \mathcal{F} \Rightarrow A \models$ false). Mais comme ($A \models$ false) est impossible (la sémantique de false est toujours 0, cf. table 2), la condition ($A \models \mathcal{F}$) doit, elle aussi, être impossible. Donc, une façon de prouver qu'une formule φ est conséquence logique d'un ensemble $\mathcal{F}$ est celle de prouver que l'ensemble augmentée de la négation de la formule permet de déduire le faux :

$$\mathcal{F} \models \varphi \quad \text{ssi} \quad \mathcal{F} \cup \{\neg\varphi\} \models \text{false}$$

TAB. 3. Application d'une substitution à une clause ouverte $\breve{\psi} \in \breve{\Psi}(\Sigma, V)$

$$\breve{\psi}\,\theta \triangleq \begin{cases} \pi\,\theta & \text{si} \quad \breve{\psi} = \pi \\ \neg(\pi\,\theta) & \text{si} \quad \breve{\psi} = \neg\pi \\ \text{false} & \text{si} \quad \breve{\psi} = \text{false} \\ (\breve{\psi}_1\,\theta) \vee (\breve{\psi}_2\,\theta) & \text{si} \quad \breve{\varphi} = \breve{\varphi}_1 \vee \breve{\varphi}_2 \end{cases}$$

TAB. 4. Principe de résolution pour un ensemble de clauses $\mathcal{F}$

$$\frac{\begin{array}{ll} \psi, \psi' \in \mathcal{F} & \\ \left\{\begin{array}{l} \sigma \text{ renommage} \\ \text{t.q. vars}(\breve{\psi}) \cap \text{vars}(\breve{\psi}'\sigma) = \emptyset \end{array}\right. & \begin{array}{l} \theta \in \text{mgu}(\pi, \pi') \\ \\ \breve{\psi}'' = [(\breve{\psi} \setminus \{\pi\}) \cup ((\breve{\psi}'\sigma) \setminus \{\pi'\})]\,\theta \end{array} \\ \pi \in \breve{\psi}\,,\ \neg\pi' \in (\breve{\psi}'\,\sigma) & \end{array}}{\mathcal{F} \text{ logiquement équivalent à } \mathcal{F} \cup \{\psi''\}} \quad \begin{pmatrix} \text{Principe} \\ \text{de} \\ \text{résolution} \end{pmatrix}$$

$$\frac{\begin{array}{ll} \psi \in \mathcal{F}\,,\ \ l_1, l_2 \in \breve{\psi} & \\ \left\{\begin{array}{l} (l_1 = \pi \text{ et } l_2 = \pi') \\ \text{ou} \\ (l_1 = \neg\pi \text{ et } l_2 = \neg\pi') \end{array}\right. & \begin{array}{l} \theta \in \text{mgu}(\pi, \pi') \\ \\ \breve{\psi}' = (\breve{\psi} \setminus \{l_2\})\,\theta \end{array} \end{array}}{\psi' \text{ conséquence logique de } \psi} \quad \begin{pmatrix} \text{Règle} \\ \text{de} \\ \text{factorisation} \end{pmatrix}$$

7.3.2.1. *Clauses en forme normale.* Puisque tous les quantificateurs, éventuellement présents, d'une clause sont au début de la clause et sont tous du même genre (universels), il existe une correspondance stricte entre le langage des clauses et celui des clauses *ouvertes* ou *en forme normale*, noté $\breve{\Psi}(\Sigma, V)$, défini par le symbole $\breve{\psi}$ de la syntaxe des clause (cf. section 7.3.1). Étant donné une clause $\psi \in \Psi(\Sigma, V)$, nous noterons $\breve{\psi}$ sa version ouverte (complètement privée des quantificateurs), et réciproquement, étant donné une clause ouverte $\breve{\psi} \in \breve{\Psi}(\Sigma, V)$, nous noterons ψ sa version quantifiée universellement. On considère souvent les clauses ouvertes comme un ensemble de littéraux, sous-entendant leur disjonction. Le sens de la réunion étant dual par rapport aux ensembles de formules, l'ensemble vide (ou *clause vide*) aura, cette fois-ci, le sens du faux, c'est-à-dire le sens de la formule false.

7.3.2.2. *Application d'une substitution aux clauses ouvertes.* L'application d'une substitution, définie sur $T(\Sigma{+}V, 2)$ (cf. section 3.3.8), peut s'étendre à l'ensemble des clauses ouvertes $\breve{\Psi}(\Sigma, V)$ (qui contient $T(\Sigma{+}V, 2)$), comme illustré dans la table 3.

7.3.2.3. *Le principe de résolution et la règle de factorisation.* Illustrées dans la table 4, ces deux règles définissent implicitement des transformations correctes pour des ensembles de clauses. Par la règle (Principe de résolution), on peut obtenir un ensemble équivalent à celui de départ. On choisira deux formules de l'ensemble, on les ouvrira (on éliminera les quantificateurs) en faisant attention qu'elles ne contiennent pas de variables communes, on calculera (s'il existe) un unificateur principal de deux littéraux opposés (un positif et un négatif), et, enfin, on construira une nouvelle clause formée de l'union des littéraux des deux clauses unifiées (exceptés les deux littéraux unifiés). La nouvelle clause ψ'', que l'on peut ajouter à $\mathcal{F}$ pour obtenir un ensemble équivalent, est appelée clause *résolvante*. En revanche, pour la règle de factorisation, on peut construire une conséquence logique d'une formule en l'ouvrant et en détectant deux littéraux s'unifiant qui seront remplacés par le littéral unificateur. Or, supposons, dans la table 4, de remplacer la thèse ($\mathcal{F}$ logiquement équivalent à $\mathcal{F} \cup \{\psi''\}$) de la règle (Principe de résolution) par $\mathcal{F} \overset{r}{\to} \mathcal{F} \cup \{\psi''\}$, et la thèse ($\psi'$ conséquence logique de ψ) de la (Règle de factorisation) par $\mathcal{F} \overset{r}{\to} \mathcal{F} \cup \{\psi'\}$. On obtient alors la définition d'une relation de transition $\overset{r}{\to}$, pour les ensembles de clauses en forme normale, qui est *correcte* et *complète* par rapport aux preuves du faux (*réfutations*).

THEOREM 7.3.2. (CORRECTION ET COMPLÉTUDE DU PRINCIPE DE RÉSOLUTION) *Soit $\overset{r}{\to}^*$ la clôture réflexive et transitive de la relation de transition $\overset{r}{\to}$ du principe de résolution. Alors :*

$$\mathcal{F} \models \text{false} \qquad \textit{ssi} \qquad \mathcal{F} \overset{r}{\to}^* \mathcal{F}' \cup \{\text{false}\}$$

□

7.4. La programmation logique

Si en théorie le principe de résolution permet de résoudre le problème semi-décidable de déterminer si une formule φ est conséquence logique d'un ensemble de formules $\mathcal{F}$, en pratique il est difficilement utilisable à cause du non-déterminisme, souvent insoutenable, dû au nombre de clauses résolvantes générées.

7.4.1. Clauses de Horn. Supposons, en revanche, de construire des ensembles de *clauses de Horn*, sous-ensemble des clauses du premier ordre, définies par la syntaxe suivante :

$$\begin{array}{lllll}
\gamma & ::= & \forall X.\gamma \mid \breve{\gamma} & & \\
\breve{\gamma} & ::= & \pi \vee \neg\pi_1 \vee .. \vee \neg\pi_n & (n \geq 0) & \text{(clause de programme)} \\
& \mid & \neg\pi_1 \vee .. \vee \neg\pi_n & (n \geq 1) & \text{(clause but)} \\
& \mid & \text{false} & & \text{(clause vide)}
\end{array}$$

où $\sigma \in S$, $X \in V$, et $\pi \in T(\Sigma{+}V, 2)$. Le langage des clauses de Horn sera noté $\Gamma(\Sigma, V)$, celui des clauses de Horn ouvertes ou en forme normale (définies par le symbole non terminal $\breve{\gamma}$) sera noté $\breve{\Gamma}(\Sigma, V)$. Les clauses de programme $\pi \vee \neg\pi_1 \vee .. \vee \neg\pi_n$ sont notées le plus souvent $\pi : -\pi_1, .., \pi_n$, tandis que les clause but $\neg\pi_1 \vee .. \vee \neg\pi_n$ sont notées aussi $?\pi_1, .., \pi_n$.

TAB. 5. Résolution SLD pour un ensemble $\mathcal{P}$ de clauses de Horn de programme

$$\frac{\gamma' \in \mathcal{P} \qquad \left\{\begin{array}{l}\sigma \text{ renommage}\\ \text{t.q. } FV(\breve{\gamma}) \cap FV(\breve{\gamma}'\sigma) = \emptyset\end{array}\right. \qquad (\neg\pi) \in \breve{\gamma},\ \pi' \in (\breve{\gamma}'\sigma) \qquad \theta \in \mathrm{mgu}(\pi, \pi') \qquad \breve{\gamma}'' = [(\breve{\gamma} \setminus \{\pi\}) \cup ((\breve{\gamma}'\sigma) \setminus \{\pi'\})]\,\theta}{\gamma \overset{\theta}{\underset{\mathcal{P}}{\longmapsto}} \gamma''}$$

7.4.2. Résolution SLD. La *programmation logique* s'appuie sur une remarque très simple. Soit $\mathcal{P}$ un ensemble (appelé *programme*) de clauses Horn de programme et considérons la question de savoir s'il existe des valeurs pour laquelle une conjonction de prédicats :

$$\breve{\varphi} = \pi_1 \wedge .. \wedge \pi_n, \quad (\text{où } n \geq 1)$$

est conséquence logique du programme. Autrement dit, la question est celle de savoir si la clôture existentielle $\varphi = \exists(\pi_1 \wedge .. \wedge \pi_n)$ de la formule ouverte est une conséquence logique de $\mathcal{P}$. Nous savons alors que la condition $\mathcal{P} \models \varphi$ est équivalente à la condition $\mathcal{P} \cup \{\neg\varphi\} \models$ false. Or, d'une part, la négation de la formule φ est précisément une clause but $\neg\varphi = \forall(\neg\pi_1 \vee .. \vee \neg\pi_n)$, et d'autre part, lorsqu'on utilise le principe de résolution entre une clause de programme et une clause but, la clause résolvante est encore une clause but. Ainsi, ne modifiant jamais le programme, nous transformerons progressivement les buts par le principe de résolution. Si dans ce procédé, nous pourrons obtenir la clause vide (le faux), nous aurons prouvé que la conjonction initiale est conséquence logique du programme. Cette relation de transition entre buts, qui prend le nom de résolution SLD, notée $\overset{\theta}{\underset{\mathcal{P}}{\longmapsto}}$, où $\mathcal{P}$ est le programme et θ est l'unificateur principal calculé, est décrite formellement dans la table 5.

Si la correction de la résolution SLD ne pose aucun doute (étant une simple spécialisation du principe de résolution), la complétude (par rapport aux réfutations) est remise en question par l'hypothèse de ne jamais altérer le programme et par le fait de ne pas utiliser la règle de factorisation. Soit $\overset{\theta}{\underset{\mathcal{P}}{\longmapsto}}^{*}$ la relation définie par $\gamma \overset{\theta}{\underset{\mathcal{P}}{\longmapsto}}^{*} \gamma'$ si et seulement si il existent $\gamma_1,..,\gamma_n$ tels que $\gamma \overset{\theta_1}{\underset{\mathcal{P}}{\longmapsto}} \gamma_1 \cdots \gamma_{n-1} \overset{\theta_n}{\underset{\mathcal{P}}{\longmapsto}} \gamma_n$, $\gamma' = \gamma_n$ et $\theta = \theta_1 \cdot .. \cdot \theta_n$. Lorsque $\gamma \overset{\theta}{\underset{\mathcal{P}}{\longmapsto}}^{*}$false, nous disons que γ est résolue avec la *réponse calculée* θ.

THEOREM 7.4.1. (CORRECTION ET COMPLÉTUDE DE LA RÉSOLUTION SLD) *Toute réponse calculée θ transforme une conjonction d'atomes $\varphi = \exists\breve{\varphi}$ en une conséquence logique $\forall(\breve{\varphi}\,\theta)$ du programme. Vice-versa, pour toute instance $\varphi' = \forall(\breve{\varphi}\,\sigma)$ d'une conjonction d'atomes $\varphi = \exists\breve{\varphi}$ conséquence logique du programme, il existera une dérivation SLD calculant, à partir de $\neg\breve{\varphi}$, une réponse θ plus générale que σ :*

$$\mathcal{P} \models \forall(\breve{\varphi}\,\sigma) \quad \textit{où} \quad \breve{\varphi} = \pi_1 \wedge .. \wedge \pi_n \qquad \textit{ssi} \qquad ?\,\pi_1, .., \pi_n \overset{\theta}{\underset{\mathcal{P}}{\longmapsto}}^{*} \text{false} \quad \textit{où} \quad \theta \leq \sigma$$

□

7.4.3. Sémantique algébrique. Dans l'hypothèse d'un ensemble $\mathcal{P}$ de clauses de Horn de programme, nous avons aussi une propriété intéressante des modèles de Herbrand de $\mathcal{P}$. On peut définir un ordre de grandeur des algèbres logiques de $\mathsf{LAlg}(\Sigma)$ sur la base de la quantité de primitives $\pi \in \mathrm{T}(\Sigma, 2)$ qu'elles rendent vraies :

$$A_1 \leq A_2 \quad \text{ssi} \quad \mathrm{T}(\Sigma, 2)_{A_1} \subseteq \mathrm{T}(\Sigma, 2)_{A_2}$$

Cet ordre fournit une structure d'ordre partiel $(\mathsf{LAlg}(\Sigma), \leq)$ à la famille des algèbres logiques. Il se trouve alors que, étant donné un programme $\mathcal{P}$, la sous-famille des modèles de Herbrand de $\mathcal{P}$, équipée de la relation $\leq$, admet un élément minimum, appelé *plus petit modèle de Herbrand*. Ce modèle, que nous noterons $M_{\mathcal{P}}$, et qui constitue la *sémantique algébrique* du programme, a comme caractéristique principale celle d'interpréter à 1 précisément les primitives $\pi = p(t_1, .., t_n) \in \mathrm{T}(\Sigma, 2)$ qui sont conséquence logique du programme :

$$p^{M_{\mathcal{P}}}(t_1, .., t_n) = \begin{cases} 1 & \text{si} \quad \mathcal{P} \models p(t_1, .., t_n) \\ \\ 0 & \text{si} \quad \mathcal{P} \not\models p(t_1, .., t_n) \end{cases}$$

Cette propriété a une interprétation algébrique. Supposer $p^{M_{\mathcal{P}}}(t_1, .., t_n) = 1$ implique $\mathcal{P} \models p(t_1, .., t_n)$. Mais, par définition de conséquence logique, tout modèle de $\mathcal{P}$ est un modèle de $p(t_1, .., t_n)$. Donc, tout modèle A de $\mathcal{P}$ est un modèle de $p(t_1, .., t_n)$, autrement dit $p^A([\![t_1]\!]_A, .., [\![t_n]\!]_A) = 1$. Nous avons ainsi la condition de conservation des semi-prédicats :

$$p^{M_{\mathcal{P}}}(t_1, .., t_n) \quad \Rightarrow \quad p^A([\![t_1]\!]_A, .., [\![t_n]\!]_A)$$

Le plus petit modèle de Herbrand caracterise les primitives qui sont conséquences logiques du programme. Cela signifie, en termes algébriques, que pour tout autre modèle A du programme, l'interprétation canonique des termes est un homomorphisme de $M_{\mathcal{P}}$ vers A, considérant toutefois les prédicats comme des semi-prédicats. En théorie des catégories, cela se traduit en disant que le plus petit modèle de Herbrand est une algèbre *initiale* dans la catégorie des modèles du programme (où les morphismes sont les homomorphismes). De façon plus intuitive, le plus petit modèle de Herbrand représente ce qui doit être forcément vrai pour qu'une algèbre sur la signature Σ soit un modèle du programme.

7.5. La programmation logique avec contraintes

En programmation logique, la sémantique algébrique d'un programme de clause de Horn est le plus petit modèle de Herbrand, représentant l'ensemble des primitives (sans variables) conséquences logiques du programme. Aucune condition n'est fixée sur les prédicats logiques utilisés dans les clauses du programme. Ainsi, la recherche du modèle se fait dans la famille des algèbres des termes sur la signature logique Σ (constituée de constructeurs de termes et de constructeurs de prédicats), où la définition des prédicats est arbitraire (cf. définition 3.3.4).

En revanche, l'idée de la programmation logique avec contraintes est celle de définir, à priori, l'interprétation des termes et celle d'un certain nombre de prédicats, par la donnée d'une algèbre χ, appelée *algèbre des contraintes*, sur une signature logique Σ_χ. Les signature admissibles, sur lesquelles construire les programmes, seront celles n'ajoutant pas de nouveaux constructeurs de termes, mais seulement de nouveau constructeurs de prédicat à la signature Σ_χ. Les prédicats prédéfinis seront appelés χ-*primitives*, les nouveaux seront les *prédicats logiques* (définis par les clauses du programme). L'idée est donc celle de restreindre, à priori, la famille d'algèbres logiques dans laquelle rechercher les modèles d'un ensemble de clauses de Horn : seulement les algèbres contenant la sous-algèbre logique χ figée pourront être des modèles du programme.

7.5.1. Algèbre des contraintes. Une algèbre logique $\chi = ((.)^\chi : \mathcal{B} \to \mathcal{B}^\chi \oplus \mathsf{S} \to \mathsf{S}^\chi \oplus \mathsf{F} \to \mathsf{F}^\chi, \mathsf{T})$ est une *algèbre de contraintes* si, étant donné un ensemble $\mathsf{V} = \biguplus_{s \in \mathsf{S}} \mathsf{V}_s$ de variables sortées disjoint de $\mathcal{B}$, S et F, elle vérifie les propriétés suivantes :

(1) elle contient le prédicat d'égalité $= : \forall \alpha : \mathsf{sort}.\, \alpha \times \alpha \to 2$

(2) il est décidable si une clause but $\gamma \in \Gamma(\Sigma_\chi, \mathsf{V})$ est vraie ou fausse en χ, c'est-à-dire si $[\![\gamma]\!]_\chi = 0$ ou bien $[\![\gamma]\!]_\chi = 1$

Les termes et les primitives (prédicats) sur $\Sigma_\chi + \mathsf{V}$ sont appelés respectivement χ-*termes* et χ-*primitives*. Une χ-*contrainte* est donc une conjonction de χ-primitives (implicitement quantifiées de façon existentielle). L'algorithme décidant de la vérité d'une clause but $\gamma \in \Gamma(\Sigma_\chi, \mathsf{V})$ est le *solveur de contraintes*. En retournant la question ($c = \neg\gamma$) et la réponse ($r' = \neg r$), on peut considérer que cet algorithme décide si une χ-contrainte c est *satisfiable* en χ. Pour cette raison, la condition (2) s'exprime en disant que la satisfiabilité des χ-contraintes doit être *complète*.

Il est courant d'associer à l'algèbre des contraintes χ un ensemble de formules $\mathcal{T} \subseteq \Phi(\Sigma_\chi, \mathsf{V})$, dont χ est un modèle ($\chi \models \mathcal{T}$), définissant de façon équivalente la relation de satisfiabilité des χ-contraintes (cf. **[Jaffar & Maher, 1994]**) :

$$\chi \models \exists(c) \qquad \text{ssi} \qquad \mathcal{T} \models \exists(c)$$

Par la suite, nous exprimerons cette condition en affirmant simplement que c *est satisfiable dans* χ, indépendemment du fait que l'on puisse représenter l'algèbre χ par un ensemble de formules $\mathcal{T}$ caractérisant l'essentiel des ses propriétés.

7.5.2. Algèbres logiques sur χ. Nous noterons $\mathsf{LAlg}(\Sigma, \chi)$ la famille des algèbres logiques $A = ((.)^A : \mathcal{B} \to \mathcal{B}^A \oplus \mathsf{S} \to \mathsf{S}^A \oplus \mathsf{F} \to \mathsf{F}^A, \mathsf{T})$ de signature Σ contenant la sous-algèbre des contraintes χ, de signature $\Sigma_\chi = (\mathcal{B}, \mathsf{S}_\chi, \mathsf{F}_\chi, \mathsf{T}_\chi)$, et telles que tout constructeur ajouté est un prédicat :

$$\forall p \in \mathsf{F}.\;\; p \notin \mathsf{F}_\chi \;\Rightarrow\; \mathrm{cod}(\mathsf{T}(p)) = 2$$

On dira que la signature Σ est une *signature logique sur* χ. La famille $\mathsf{LAlg}(\Sigma, \chi)$ est celle où l'on recherche les modèles des programmes logiques.

TAB. 6. Résolution CSLD pour un ensemble $\mathcal{P}$ de clauses de Horn de programme

$$\frac{\gamma \in \mathcal{P} \qquad \left\{\begin{array}{l}\sigma \text{ renommage} \\ \text{t.q. } FV(?\,c; b) \cap FV(\breve{\gamma}\sigma) = \emptyset\end{array}\right. \qquad \breve{\gamma}\sigma = (\pi' : -\, c' \,;\, b') \qquad \pi \in b\,,\ \pi = p(t_1, .., t_n)\,,\ \pi' = p(t'_1, .., t'_n) \qquad \begin{array}{l} c'' = c \wedge c' \wedge (t_1 = t'_1) \wedge .. \wedge (t_n = t'_n) \\ \text{satisfiable dans } \chi \end{array} \qquad b'' = (b \setminus \{\pi\}) \cup b'}{?\,c; b \overset{\chi}{\underset{\mathcal{P}}{\longmapsto}} ?\,c''; b''}$$

7.5.2.1. *Notation pour les clauses.* Dans le cadre de la programmation logique par contraintes, il est courant de séparer, à l'intérieur des clauses, les χ-primitives des autres primitives, que nous qualifierons de *logiques*. Les deux genres de primitives sont regroupés dans les écritures habituelles :

$$\begin{array}{ll} \pi : -\, c \,;\, b & \text{(clause de programme)} \\ ?\, c \,;\, b & \text{(clause but)} \end{array}$$

où c est la conjonction des χ-primitives, donc une χ-contrainte (éventuellement vide), et b, le *corps* de la clause, est la conjonction des primitives logiques (lui aussi éventuellement vide). Comme dans la programmation logique pure, les clauses de programme $\pi : -\, c \,;\, b$ contribuent à définir les prédicats logiques de façon indirecte (par le rapport d'implication logique entretenu avec les autres prédicats). En particulier, dans la programmation logique avec contrainte, la tête de la clause $\pi = p(t_1, .., t_n)$, ne pourra être une χ-primitive (puisqu'elles sont déjà toutes définies), mais sera obligatoirement une primitive logique : $p \in S \setminus S_\chi$.

7.5.3. Résolution CSLD. La résolution CSLD, illustrée dans la table 6, est l'adaptation de la résolution SLD au cadre des langages de clauses de Horn de programme, où la partie contrainte est interprétée dans l'algèbre des contraintes χ. Si l'avancement de la résolution SLD dépend de la possibilité d'unifier un sous-but avec la tête d'une clause, dans la résolution CSLD il dépend du contrôle de satisfiabilité de l'ensemble des contraintes de la clause résolvante.

Soit $\overset{\chi}{\underset{\mathcal{P}}{\longmapsto}}{}^*$ la clôture réflexive et transitive de la relation $\overset{\chi}{\underset{\mathcal{P}}{\longmapsto}}$. Lorsque nous avons $\gamma \overset{\chi}{\underset{\mathcal{P}}{\longmapsto}}{}^* c\,;\text{false}$, nous disons que la clause but γ est résolue avec la *contrainte calculée* c (qui est satisfiable en χ). Le théorème de correction affirme que toute contrainte calculée c transforme la conjonction $\varphi = \exists\breve{\varphi}$ en une conséquence logique $\forall(\breve{\varphi} \wedge c)$ du programme. Viceversa, celui de complétude, affirme que pour toute instance $\forall(\breve{\varphi} \wedge c')$ d'une conjonction d'atomes $\varphi = \forall\breve{\varphi}$ conséquence logique du programme, il existera une dérivation SLD calculant, à partir du but $\neg\breve{\varphi}$, une réponse c plus générale que c'.

THEOREM 7.5.1. (CORRECTION ET COMPLÉTUDE DE LA RÉSOLUTION CSLD) *Soit* $\Sigma = (\mathcal{B}, S, F, T)$ *une signature logique sur* χ*, et* $V = \biguplus_{s \in S} V_s$ *un ensemble de variables sortées disjoint de* $\mathcal{B}$*,* S *et* F*. Soit* g *une clause but. Alors :*

(1) *(correction) toute réponse calculée est correcte, i.e. une réponse qui implique la conjonction* $\neg g$ *:*

$$\textit{si} \quad g \overset{\chi}{\underset{\mathcal{P}}{\longmapsto}}{}^{*} c\,;\emptyset \qquad \textit{alors} \qquad \chi \models \mathcal{P} \Rightarrow \forall(c \Rightarrow \neg g)$$

(2) *(complétude) toute réponse correcte est couverte par une disjonction arbitraire de réponses calculées :*

$$\textit{si} \quad \chi \models \mathcal{P} \Rightarrow \forall(c \Rightarrow \neg g) \qquad \textit{alors} \qquad \textit{il existe } d = \textstyle\bigwedge_{i\in I} c_i, \textit{ où}$$
$$g \overset{\chi}{\underset{\mathcal{P}}{\longmapsto}}{}^{*} c_i\,;\emptyset, \qquad \textit{tel que} \qquad \chi \models \forall(c \Rightarrow d)$$

□

En particulier, le théorème de correction représente, comme celui de la résolution SLD, un résultat fondamental et quelque peu surprenant. En effet, étant donné une formule existentielle $\varphi = \exists\breve{\varphi}$, où $\breve{\varphi} = \pi_1 \wedge .. \wedge \pi_n$, la condition :

$$\neg\breve{\varphi} \overset{\chi}{\underset{\mathcal{P}}{\longmapsto}}{}^{*} c\,;\emptyset$$

signifie non seulement que, dans le respect des contraintes c, la conjonction existentielle $\exists\breve{\varphi}$ est conséquence logique du programme, mais que sa version quantifié universellement $\forall\breve{\varphi}$, qui est une formule bien plus forte d'un point de vue logique, l'est aussi.

7.5.4. Sémantique algébrique. On dit qu'une algèbre des termes $A \in \mathsf{TAlg}(\Sigma)$ est une algèbre des termes sur χ, si Σ est une signature sur χ et si pour toute χ-primitive $p(t_1, .., t_n)$ nous avons :

$$p^A(t_1, .., t_n) = 1 \quad \text{ssi} \quad p^\chi([\![t_1]\!]_\chi, .., [\![t_n]\!]_\chi) = 1$$

L'ensemble des algèbres des termes sur χ sera noté $\mathsf{TAlg}(\Sigma, \chi)$. Comme pour la programmation logique pure, nous avons une sémantique des conséquences logiques des programmes avec contraintes. En effet, si nous nous limitons aux algèbres des termes sur χ, l'ensemble des modèles du programme $\mathcal{P}$ a un point minimum par rapport à l'ordre $\leq$ entre modèles :

$$M_{\mathcal{P}} = \min_{\leq} \mathsf{TAlg}(\Sigma, \chi)_{\mathcal{P}}$$

7.5.5. Compositionalité. Une propriété importante de la résolution CSLD est la *compositionalité* (ou *And-compositionalité*) des buts et des réponses. Tout but conjonctif $?\,d\,;\pi_1, .., \pi_n$ peut être décomposé en plusieurs buts simples $?\,d\,;\pi_i$ à résoudre individuellement. Les réponses du premier pourront toujours s'exprimer par la recomposition (la conjonction) des réponses individuelles.

THEOREM 7.5.2. (COMPOSITIONALITÉ DE LA RÉSOLUTION CSLD) *Soit* $\Sigma = (\mathcal{B}, \mathsf{S}, \mathsf{F}, \mathsf{T})$ *une signature logique sur* χ*, et* $\mathsf{V} = \biguplus_{s\in\mathsf{S}} \mathsf{V}_s$ *un ensemble de variables sortées disjoint de* $\mathcal{B}$*,* S *et* F*. Alors :*

$$?\,d\,;\pi_1,..,\pi_n \overset{\chi}{\underset{\mathcal{P}}{\longmapsto}}{}^{*}\ c\,;\emptyset \qquad \textit{ssi} \qquad \begin{cases} ?\,\mathrm{true}\,;\pi_1 \overset{\chi}{\underset{\mathcal{P}}{\longmapsto}}{}^{*}\ c_1\,;\emptyset \\ \vdots \\ ?\,\mathrm{true}\,;\pi_n \overset{\chi}{\underset{\mathcal{P}}{\longmapsto}}{}^{*}\ c_n\,;\emptyset \\ c = d \wedge c_1 \wedge .. \wedge c_n \end{cases}$$

où $n \geq 1$.

DÉMONSTRATION. *cf. par exemple* [**Gabbrielli et al., 1995**] *ou* [**Fages, 1996**]. □

CHAPITRE 8

Définition du jeu CLP(χ)

Dans ce chapitre nous verrons que la programmation logique, *pure* (LP) ou *par contraintes* (CLP), présentée dans le chapitre précédent, peut s'exprimer par un jeu combinatoire à deux joueurs. Nous définirons le jeu de façon formelle et nous fournirons la preuve que la sémantique habituelle d'un but dans un programme logique (ensemble des réponses ou des contraintes calculées) coïncide avec la sémantique *pessimiste* du jeu. De plus, nous montrerons que la structure utilisée pour une telle évaluation est une structure $\alpha\beta$-normale, ce qui autorise, dans la résolution d'un but, l'application de la méthode Alpha-Bêta.

8.1. Arène du jeu CLP(χ)

On peut donner une image du jeu CLP(χ) en affirmant que deux joueurs fictifs s'opposent sur un programme logique spécifique dont le rôle est comparable à celui d'un damier. Étant donné un but atomique, c'est-à-dire une position initiale, le *joueur* tentera de le prouver par une règle logique compatible (s'unifiant) du programme. Une fois choisi son coup, il passera la main à son adversaire, l'*opposant*, qui tentera, à son tour, de déterminer un sous-but de la règle ne pouvant pas être prouvé par son adversaire. D'une façon un peu plus précise, dans ce jeu les positions sont des quadruplets $< \mathcal{P}, g, c, \nu >$ où :

- $\mathcal{P}$ est le programme sur lequel se jouent les matchs
- g est soit un but sans contraintes $g = ?\emptyset; b$ (que nous noterons $g = ?b$), soit un atome $g = \pi$, où $\pi = p(t_1, .., t_n)$ et $p \notin F_\chi$. Dans le premier cas la position est de l'opposant, dans le second la position est du joueur
- c est une χ-contrainte, représentant le payoff courant du joueur (qui ne pourra qu'augmenter)

– h est la suite des ensembles (finis) de variables nouvelles introduites au cours du match (par le renommage des clauses de programme). Nous l'appellerons *histoire* des variables introduites.

Pour tout ensemble infini de variables V, nous supposerons de disposer d'une fonction *injective* $\xi_V : (\wp_F^V)^* \times \mathbb{N} \to \wp^V$, partitionnant l'ensemble des variables V en autant de parties, encore infinies, que de couples (h, i), où h est histoire et i un entier naturel.

Definition 8.1.1. (Syntaxe du jeu CLP(χ)) *Étant donné une algèbre de contraintes χ de signature $\Sigma_\chi = (\mathcal{B}, S_\chi, F_\chi, T_\chi)$, une signature logique $\Sigma = (\mathcal{B}, S, F, T)$ sur χ, et un ensemble infini de variables sortées $V = \biguplus_{s\in S} V_s$ disjoint de $\mathcal{B}$, S et F, la syntaxe du jeu CLP(χ) est le quadruplet $\mathcal{S} = (\mathbb{P}, \lambda, \rightarrow, \mathbb{P}_I)$ où :*

– *l'ensemble des positions du jeu est l'ensemble :*

$$\mathbb{P} \triangleq \text{Progs}(\chi,\Sigma,V) \times (\text{Bodies}(\chi,\Sigma,V) \oplus \text{Atoms}(\chi,\Sigma,V)) \times \text{Constr}(\chi,V) \times (\wp^V)^*$$

où :

$$\begin{array}{lcl}
\text{Progs}(\chi,\Sigma,V) & \triangleq & \text{PClauses}(\chi, \Sigma, V)^* \\
\text{PClauses}(\chi,\Sigma,V) & \triangleq & \{\gamma \in \Gamma(\Sigma,V) \mid \gamma = \forall(\pi :- \pi_1, .., \pi_n),\ n \geq 0,\ \pi = p(t_1, .., t_k),\ p \notin F_\chi\} \\
\text{Bodies}(\chi,\Sigma,V) & \triangleq & \{\breve{\gamma} \in \breve{\Gamma}(\Sigma,V) \mid \breve{\gamma} = (?\emptyset; \pi_1, .., \pi_n),\ n \geq 1\} \\
\text{Atoms}(\chi,\Sigma,V) & \triangleq & T(\Sigma + V, 2) \setminus T(\Sigma_\chi + V, 2)
\end{array}$$

– *la fonction qui classifie les positions des joueurs $\lambda : \mathbb{P} \to \{\text{Player}, \text{Opponent}\}$ est définie par :*

$$\lambda(\mathcal{P}, x, c, \nu) = \begin{cases} \text{Player} & si \quad x \in \text{Atoms}(\chi, \Sigma, V) \\ \\ \text{Opponent} & si \quad x \in \text{Bodies}(\chi, \Sigma, V) \end{cases}$$

– *la relation de transition $\rightarrow$: $(\mathbb{P} \times \mathbb{N}_+) \multimap \mathbb{P}$ représentant les mouvements est définie par les règles de la table 1*

– *l'ensemble des positions initiales du jeu est celui des positions de l'opposant où la suite des variables introduites est constituée du seul ensemble des variables libres du but implicite :*

$$\mathbb{P}_I \triangleq \{< \mathcal{P}, ?b, c, h > \ \mid h = FV(?b\,; c)\}$$

$\square$

Les programmes sont considérés comme des suites de clauses (de programme), au lieu de simples ensembles. L'écriture $\gamma = \mathcal{P}_i$, qui remplace l'ancienne $\gamma \in \mathcal{P}$, indique non seulement que γ est une clause du programme, mais qu'il s'agit de la i-ème (où $i \geq 1$). Cela permet de numéroter les clauses de programme et, par conséquent, les coups du joueur. Par la règle (Oppon. Moves), les mouvement possibles de l'opposant, dans une position $< \mathcal{P}, (?\pi_1, .., \pi_n), c, h >$, sont autant qu'il y a d'atomes π_i dans le corps du but. Avec ce mouvement, le jeu avance dans la nouvelle position du joueur $< \mathcal{P}, \pi_i, c, h >$, sans qu'aucune contrainte ni qu'aucun ensemble de variables s'ajoutent aux précédentes. D'une façon imagée, l'opposant

TAB. 1. Relation de transition du jeu CLP(χ)

$$\frac{}{< \mathcal{P}, (?\pi_1, .., \pi_n), c, v > \xrightarrow{i} < \mathcal{P}, \pi_i, c, v >} \qquad \binom{\text{Oppon.}}{\text{moves}}$$

$$\frac{\begin{array}{ll} \gamma = \mathcal{P}_i & \\ & \pi = p(t_1, .., t_n) \\ \left\{\begin{array}{l} \sigma \text{ renommage} \\ \text{t.q. } v = \mathsf{FV}(\breve{\gamma}\sigma) \subseteq \xi_V(h, i) \\ \text{et } \mathsf{FV}(?c;\pi) \cap \mathsf{FV}(v) = \emptyset \end{array}\right. & \begin{array}{l} \pi' = p(t'_1, .., t'_n) \\ \\ c'' = c \wedge c' \wedge (t_1 = t'_1) \wedge .. \wedge (t_n = t'_n) \\ \text{satisfiable dans } \chi \end{array} \\ \breve{\gamma}\sigma = (\pi' : - c' \, ; b) & \end{array}}{< \mathcal{P}, \pi, c, h > \xrightarrow{i} < \mathcal{P}, ?b, c'', h.v >} \qquad \binom{\text{Player}}{\text{moves}}$$

$$\frac{\exists i \geq 1. \; < \mathcal{P}, \pi, c, h > \xrightarrow{i} < \mathcal{P}, ?b, c', h' >}{< \mathcal{P}, \pi, c, h > \xrightarrow{0} < \mathcal{P}, ?\emptyset, \text{false}, h >} \qquad \binom{\text{Player}}{\text{suicide}}$$

dispose d'autant de possibilités que d'atomes pour démontrer que leur conjonction est fausse ; pour cela il n'introduit pas de nouvelles contraintes ni de nouvelles variables. En revanche, par la règle (Player Moves), le joueur dispose, dans une position $< \mathcal{P}, \pi, c, h >$, d'autant de coups que de clauses de programme *compatibles* avec le but π et la contrainte c (où le sens du mot compatible est celui de la résolution CSLD pour le but $?\,c;\,\pi$). Pour effectuer un coup il introduira des nouvelles contraintes qui s'ajouterons aux anciennes pour former un tout encore satisfiables en χ. Les nouvelles contraintes porterons sur des nouvelles variables qui s'ajouterons à la quatrième composante de la position, la *suite des variables introduites*. La règle (Player suicide), qui s'applique seulement lorsque le joueur a au moins un autre coup disponible (par la règle (Player move)), ne représente pas véritablement une seconde possibilité pour le joueur puisqu'elle conduit systématiquement le jeu dans une position où le score (false) est le plus mauvais possible pour le joueur. Il s'agit d'un coup fictif, qui n'ajoute aucune valeur intéressante au jeu, mais qui permettra de récupérer l'histoire de la position dans le mécanisme de rétro-propagation des valeurs. Avec ces règles du jeu, le joueur n'aura donc jamais qu'un seul coup. Il en aura aucun, ou bien au moins deux.

Dans ce jeu il est simple de caractériser l'ensemble des positions terminales. D'une part, les positions terminales de l'opposant, que nous noterons $< \mathcal{P}, (?\emptyset), c, h >$, sont les positions $< \mathcal{P}, (?\pi_1, .., \pi_n), c, h >$ où il n'y a pas d'atomes dans le corps du but ($n = 0$). D'autre part, les positions terminales du joueur $< \mathcal{P}, \pi, c, h >$ sont celles pour lesquelles il n'existe pas de clauses du programme compatibles.

8.2. Structure de co-évaluation pour CLP(χ)

Pour l'interprétation bisémique du jeu CLP(χ), nous devons définir une structure de co-évaluation monotone $\mathcal{D} = (D, \leq_D, \uparrow, \downarrow)$ (cf. définition 5.5.4). Autrement dit, nous devons :

(1) définir un domaine des valeurs du jeu D ;

(2) définir les fonctions des joueurs $\uparrow : D^* \to D$ et $\downarrow : D^* \to D$ constituant, avec D, une structure d'inteprétation élémentaire $(D, \uparrow, \downarrow)$ (cf. section 5.2.1) ;

(3) définir une fonction de payoff $p : \mathbb{P}_T \to D$ et deux heuristiques, la pessimiste et l'optimiste $h_1, h_2 : \mathbb{P}_{NT} \to D$;

(4) définir une relation $\leq : 1^{D \times D}$ sur les valeurs, telle que pour toute position $\pi \in \mathbb{P}$ du jeu la sous-structure $(D^{\pm}_{\pi}, \leq_{D^{\pm}_{\pi}})$ des valeurs pessimistes ou optimistes du jeu (i.e. les valeurs des ensembles $D^{\pm}_{\pi} = D^{p,h_1,h_2}_{\pi}$, cf. définition 5.5.2) soit un préordre ;

(5) vérifier que les fonctions $\uparrow$ et $\downarrow$ sont monotones dans toutes leurs composantes par rapport au valeurs pessimistes ou optimistes du jeu ;

(6) vérifier que les heuristiques sont correctes par rapport à la relation définie ;

(7) vérifier que les heuristiques sont dans le préordre extensionnel : $h_1(\pi) \leq_{D^{\pm}_{\pi}} h_2(\pi)$;

8.2.1. Le domaine des valeurs du jeu.

Le choix des valeurs pour le jeu CLP(χ) est largement inspiré par les travaux sur les *EX-équations* (introduites dans [**Marriot et al., 1994**]), et, plus en général, par la théorie de l'interprétation abstraite de programmes (cf. les travaux fondateurs [**Cousot & Cousot, 1977**] et [**Cousot & Cousot, 1979**], et les spécifiques au paradigme de la programmation logique [**Cousot & Cousot, 1992**]). En effet, nous introduirons une généralisation des EX-équations au cas d'une algèbre de contraintes, que nous appellerons *EX-réponses.*

8.2.1.1. *Réponses concrètes.* De la notion de réponse nous aurons une version *concrète* et une *abstraite* : dans la démarche habituelle de l'interprétation abstraite, la notion abstraite permettra de regrouper l'ensemble des réponses concrètes que l'on souhaite considérer équivalentes.

Definition 8.2.1. (Réponses, EX-réponses) *Le langage des* réponses, *noté* Ans(χ, V), *est généré par le symbole* ρ *de la syntaxe suivante :*

$$\begin{array}{lcll} \rho & ::= & \bigvee_{i \in I} c_i & \textit{(réponse)} \\ c & ::= & \pi_1 \wedge .. \wedge \pi_n \quad (n \geq 0) & \textit{(χ-contrainte)} \\ & | & \text{false} & \end{array}$$

où $X \in V$ *et les primitives* $\pi \in T(\Sigma_\chi + V, 2)$ *sont des* χ*-primitives. La formule* true *sera synonyme de la* χ*-contrainte* $\pi_1 \wedge .. \wedge \pi_n$ *avec* $n = 0$*. Les formules logiques de la forme* $\exists X_1 .. \exists X_n .\, \rho$ *seront appelées* EX-réponses. □

Les réponses sont donc des disjonctions arbitraires de χ-contraintes. Deux formules logiques φ_1 et φ_2, closes ou non, peuvent être comparées par l'implication logique dans l'algèbre des contraintes χ.

DEFINITION 8.2.2. (PRÉORDRE D'IMPLICATION LOGIQUE) *Étant donné une algèbre de contraintes* χ *de signature* $\Sigma_\chi = (\mathcal{B}, S_\chi, F_\chi, T_\chi)$, *une signature logique* $\Sigma = (\mathcal{B}, S, F, T)$ *sur* χ, *et un ensemble infini de variables sortées* $V = \biguplus_{s \in S} V_s$ *disjoint de* $\mathcal{B}$, S *et* F, *deux formules* $\varphi_1, \varphi_2 \in \breve{\Phi}(\Sigma, V)$, *sont dans l'ordre d'*implication (logique)*, noté* $\varphi_1 \subseteq \varphi_2$, *si l'algèbre des contraintes* χ *est un modèle de la clôture universelle de la formule logique* $\varphi_1 \Rightarrow \varphi_2$ *:*

$$\varphi_1 \subseteq \varphi_2 \qquad ssi \qquad \chi \models \forall(\varphi_1 \Rightarrow \varphi_2)$$

L'équivalence engendrée $(\subseteq \cap \supseteq)$ *est notée* $\equiv$. □

En particulier, la relation d'implication logique permet de comparer des réponses concrètes $\rho_1, \rho_2 \in \mathrm{Ans}(\chi, V)$. Intuitivement, deux réponses concrètes sont dans le préordre d'implication si les solutions dénotées par l'une sont contenues dans les solutions dénotées par l'autre, d'où le choix du symbole d'inclusion ensembliste $\subseteq$. La relation d'implication est un préordre propre, i.e. seules les propriétés réflexive et transitive sont vérifiées (la preuve est triviale), mais pas la propriété anti-symétrique des ordres partiels. Un contre-exemple est fourni par les formules (true) et $(X = X)$, qui sont équivalentes sans être égales.

8.2.1.2. *Réponses abstraites.* La relation d'équivalence engendrée (cf. section 2.1.2.3) par le préordre $\subseteq$ donne lieu à la notion de *réponse abstraite*.

DEFINITION 8.2.3. (RÉPONSES) *L'ensemble des* réponses abstraites*, noté* $\mathrm{Ans}(\chi, V)_\equiv$, *est l'ensemble des classes d'équivalences* $(\equiv\, = (\subseteq \cap \supseteq))$ *de la relation d'implication logique entre réponse :*

$$\mathrm{Ans}(\chi, V)_\equiv \triangleq \{[\rho]_\equiv \mid \rho \in \mathrm{Ans}(\chi, V)\}$$

□

Le préordre d'implication étendu aux classes d'équivalence, c'est-à-dire aux réponses abstraites, vérifie aussi la propriété anti-symétrique (cf. section 2.1.2.3). Donc la structure $(\mathrm{Ans}(\chi, V)_\equiv, \subseteq)$ est un ordre partiel. Pour alléger la notation, par la suite nous dénoterons la classe d'une réponse concrète ρ, par l'écriture $\rho_\equiv$ au lieu de $[\rho]_\equiv$.

8.2.1.3. *Valeurs du jeu.* Nous allons définir les valeurs du jeu comme des couples (ρ, h) formés, d'une part, par une réponses ρ, et, d'autre part, par une histoire h des variables introduites au cours du match. En divisant la composante histoire $h = p.s$ en deux parties, un préfixe p et un suffixe s, on sera en mesure de construire, par la clôture existentielle des variables du suffixe, une nouvelle formule, la EX-réponse :

$$\exists X_1..\exists X_n.\rho$$

où $\{X_1,..,X_n\} = \bigcup_j s_j$. La raison d'être de la composante histoire sera donc de permettre la comparaison des valeurs sur le plus grand préfixe commun de leurs histoires, en confrontant (par l'ordre d'implication) les EX-réponses obtenues par la clôture des variables des suffixes.

NOTATION 8.2.4. Par la suite, nous écrirons $\exists(h).\rho$ pour signifier $\exists X_1..\exists X_n.\rho$, où $\{X_1,..,X_n\} = \bigcup_j h_j$.

DEFINITION 8.2.5. (VALEURS DU JEU) *L'ensemble des valeurs du jeu est un ensemble de couples* (ρ, h) *où* ρ *est une réponse et* h *une histoire contenant les variables libres de la réponse :*

$$D \triangleq \{(\rho,h) \mid \rho \in \mathrm{Ans}(\chi,V),\ h \in (\wp^V)^*,\ FV(\rho) \subseteq \cup_j h_j\}$$

Deux valeurs du jeu $x = (\rho, h)$ *et* $y = (\rho', h')$ *sont* fortement équivalentes*, noté* $x \equiv y$*, si la composante histoire est identique (*$h = h'$*) et les réponses sont logiquement équivalentes (*$\rho \equiv \rho'$*) dans l'algèbre* χ*.* □

Les valeurs du jeu sont donc des réponses couplées avec des histoires contenant (considérant la suite comme l'union de ses éléments) les variables libres de la réponse. La notion d'équivalence $\equiv$ entre formules permet de faire abstraction des différences non significatives, d'un point de vue logique, entre valeurs du jeu. Par exemple, la notion d'équivalence forte capture les valeurs suivantes :

$$(\mathrm{true} \wedge X{=}1 \wedge Y{=}3, \{X,Y\}) \ \equiv\ (X{=}1 \wedge Y{=}3, \{X,Y\}) \ \equiv\ (Y{=}3 \wedge X{=}1, \{X,Y\})$$

8.2.2. Les fonctions des joueurs. Intuitivement, les fonctions des joueurs, de type $\uparrow: D^* \to D$ et $\downarrow: D^* \to D$, correspondent respectivement à la disjonction et à la conjonction logiques des contraintes. Autrement dit, le joueur collecte (fait le *ou* logique) de toutes les réponses provenant des coups à sa disposition, tandis que l'opposant vérifie la cohérence (fait le *et* logique) des valeurs des mouvements à sa disposition. Les deux fonctions élaborent la composante histoire de la même façon : elles restituent toujours, dans le résultat, la partie commune, c'est-à-dire le plus grand préfixe commun, des histoires des arguments. Cela permet de remonter, par le mécanisme de rétro-propagation des valeurs, seulement les variables relevantes pour les buts constituant les positions du jeu.

Les deux opérations sont définies comme l'extension d'un noyau binaire au suites de valeurs (cf. définition 6.2.2).

DEFINITION 8.2.6. (FONCTIONS DES JOUEURS DANS CLP(χ)) *Les fonctions du joueur* $\uparrow$ *et de l'opposant* $\downarrow$ *sont l'extension au suites* $\uparrow = \mathsf{ext}(\overset{2}{\uparrow})$*,* $\downarrow = \mathsf{ext}(\overset{2}{\downarrow})$*, des fonctions binaires définies de la façon suivante :*

$$(\rho, h) \overset{2}{\uparrow} (\rho', h') \triangleq (\rho \oslash \rho', p)$$

$$(\rho, h) \overset{2}{\downarrow} (\rho', h') \triangleq (\rho \otimes \rho', p)$$

où $\rho = c_1 \vee .. \vee c_n$, $\rho' = d_1 \vee .. \vee d_m$, *les opérations* $\oslash\!\!\!\vee$ *et* $\oslash\!\!\!\wedge$ *sont respectivement la disjonction et la conjonction de réponses :*

$$\rho \circledvee \rho' \triangleq c_1 \vee .. \vee c_n \vee d_1 \vee .. \vee d_m \qquad \rho \circledwedge \rho' \triangleq \bigvee_{\substack{i=1..n\\ j=1..m}} c_i \wedge d_j$$

p *est le plus grand préfixe commun des suites* $h = p.s$ *et* $h' = p.s'$ *:*

$$p = \inf_{\sqsubseteq}\{h, h'\}$$

et $\sqsubseteq$ *est la relation préfixe entre suites (cf. section 3.1.1).* □

Les opérations des joueurs ainsi définies sont mutuellement distributives.

LEMMA 8.2.7. (PROPRIÉTÉ DISTRIBUTIVE) *Les fonctions du joueur et de l'opposant sont distributives :*

$$x \overset{2}{\downarrow} (y \overset{2}{\uparrow} z) \equiv (x \overset{2}{\downarrow} y) \overset{2}{\uparrow} (x \overset{2}{\downarrow} z)$$
$$x \overset{2}{\uparrow} (y \overset{2}{\downarrow} z) \equiv (x \overset{2}{\uparrow} y) \overset{2}{\downarrow} (x \overset{2}{\uparrow} z)$$

DÉMONSTRATION. *Nous prouvons la première équation, la preuve de la seconde étant symétrique. Supposons* $x = (\rho, h)$, $y = (\rho', h')$ *et* $z = (\rho'', h'')$. *Le premier membre de l'équation est une valeur dont la composante histoire est égale à :*

$$p = \inf_{\sqsubseteq}\{h, \inf_{\sqsubseteq}\{h', h''\}\}$$

Mais la borne inférieure est toujours idempotente, commutative et associative, donc :

$$\begin{array}{rcl} p & = & \inf_{\sqsubseteq}\{h, h, \inf_{\sqsubseteq}\{h', h''\}, \inf_{\sqsubseteq}\{h', h''\}\} \\ & = & \inf_{\sqsubseteq}\{h, \inf_{\sqsubseteq}\{h', h''\}, h, \inf_{\sqsubseteq}\{h', h''\}\} \\ & = & \inf_{\sqsubseteq}\{\inf_{\sqsubseteq}\{h, h'\}, h'', \inf_{\sqsubseteq}\{h, h''\}, h'\} \end{array}$$

De plus, elle est réductive et simplifiable, donc :

$$p = \inf_{\sqsubseteq}\{\inf_{\sqsubseteq}\{h, h'\}, \inf_{\sqsubseteq}\{h, h''\}\}$$

qui est précisément la composante histoire du second membre. Autrement dit, les deux membres de l'équation ont la même histoire. Soit $h = p.s$, $h' = p.s'$ *et* $h'' = p.s''$. *Le premier membre est égal à* $(\rho \circledwedge (\rho' \circledvee \rho''), p)$, *tandis que le second est égal à* $((\rho \circledwedge \rho') \circledvee (\rho \circledwedge \rho''), p)$. *Les connecteurs logiques sont distributifs :*

$$\rho \circledwedge (\rho' \circledvee \rho'') \equiv (\rho \wedge \rho') \vee (\rho \wedge \rho'') \equiv (\rho \circledwedge \rho') \circledvee (\rho \circledwedge \rho'')$$

donc l'énoncé. □

L'utilité de cette propriété est double. D'une part elle fait partie des conditions d'application de la méthode Alpha-Bêta. D'autre part, elle permet, lorsqu'une heuristique est adéquate pour les stratégies, de caractériser la sémantique d'un jeu par l'ensemble de valeurs de ses stratégies (cf. section 5.6.3).

8.2.3. Fonction de payoff et heuristiques. Le coût, ou payoff, des positions terminales dépend du joueur qui avait la main et qui n'a pas eu de mouvements disponibles pour avancer dans le jeu. Si le joueur se retrouve dans une position où il ne peut plus avancer (aucune clause de programme permet de réduire le prédicat π de sa position $< \mathcal{P}, \pi, c, h >$), alors le payoff est pour lui le plus cher possible, c'est-à-dire la réponse false (avec l'histoire h). Vice-versa, si l'opposant est bloqué, c'est-à-dire il n'y a plus de prédicats dans le but de sa position $< \mathcal{P}, ?\emptyset, c, h >$, alors le payoff est la réponse (c, h). Il s'agit à nouveau du plus mauvais score possible dans la position actuelle, pour le joueur qui a la main, puisqu'il ne peut pas obliger son adversaire à un prix encore plus important par rapport à ce qui a déjà été cumulé au cours du match. Il devra donc se faire une raison et se "contenter" de la contrainte cumulée auparavant. Il s'agit donc d'une sorte de jeu où il ne faut surtout pas rester bloqué.

Definition 8.2.8. (Fonctions de payoff pour CLP(χ)) *La fonction de payoff* $p : \mathbb{P}_T \to D$ *pour le jeu CLP(χ) est la fonction suivante :*

$$p(x) \triangleq \begin{cases} (\text{false}, h) & si \quad x = < \mathcal{P}, \pi, c, h > \\ \\ (c, h) & si \quad x = < \mathcal{P}, ?\emptyset, c, h > \end{cases}$$

$\square$

Le caractère, pessimiste ou optimiste, des heuristiques dépend du point de vue choisi, qui est celui du joueur. Puisque le joueur cumule les contraintes par l'application des clauses du programme, il ne peut pas souhaiter mieux que de payer le prix représenté par la contrainte c d'une position $< \mathcal{P}, x, c, h >$. Il s'agit donc de son heuristique optimiste. D'autre part, s'il est vraiment pessimiste, il considérera que le chemin entrepris ne pourra conduire à aucune bonne solution pour lui. Son heuristique pessimiste est donc la réponse représentée par la contrainte false. Le point de vue de l'opposant est, bien entendu, opposé. Pour lui, la contrainte c représente ce qu'il y a de plus pessimiste (puisqu'il peut espérer de continuer le jeu pour obliger son adversaire à cumuler d'autres contraintes), tandis que la contrainte false représente ce qu'il y a de plus optimiste (il réussi dans son propos d'empêcher le joueur de prouver le but).

Definition 8.2.9. (Histoire d'une position) *La fonction* history : $\mathbb{P} \to D$ *est la projection de la composante histoire des positions du jeu :*

$$\text{history}(< \mathcal{P}, x, c, h >) \triangleq h$$

$\square$

Tout comme la fonction de payoff, les deux heuristiques renvoient toujours une formule équipée de l'histoire de la position qui leur est donnée en argument.

DEFINITION 8.2.10. (FONCTIONS HEURISTIQUES POUR CLP(χ)) *Les deux fonctions heuristiques pour le jeu CLP(χ), la pessimiste et l'optimiste,* $h_1, h_2 : \mathbb{P}_{NT} \to D$, *sont les fonctions suivantes :*

$$\begin{cases} h_1(< \mathcal{P}, x, c, h >) & \triangleq & (\text{false}, h) \\ \\ h_2(< \mathcal{P}, x, c, h >) & \triangleq & (c, h) \end{cases}$$

□

Un effet important de cette définition est que les valeurs pessimistes (obtenues par rétro-propagation avec p et h_1) ou optimistes (obtenues par rétro-propagation avec p et h_2) d'une position du jeu π partagent toutes la même histoire, précisément $\text{history}(\pi)$.

PROPOSITION 8.2.11. *Les valeurs pessimistes ou optimistes d'une position* π *du jeu CLP(χ) ont toutes la même composante histoire égale à la valeur* $\text{history}(\pi)$ *:*

$$\forall \pi \in \mathbb{P}.\ \ \forall (\rho, h) \in D^{\pm}_{\pi}.\ \ h = \text{history}(\pi)$$

DÉMONSTRATION. *L'ensemble* $D^{\pm}_{\pi}$ *est l'image de la rétro-propagation des valeurs (avec* h_1 *ou avec* h_2*) pour tous les préfixes finis de la position* π *:*

$$D^{\pm}_{\pi} = \{(\rho, h) \in D \mid (\rho, h) = v_{p,h_1}(t) \vee (\rho, h) = v_{p,h_2}(t)\ \ ,\ t \in \widehat{T}(\mathcal{S}, \pi)\}$$

Soit $(\rho, h) \in D^{\pm}_{\pi}$. *Alors il existe* $t \in \widehat{T}(\mathcal{S}, \pi)$ *tel que* $(\rho, h) = v_{p,h_1}(t)$ *ou tel que* $(\rho, h) = v_{p,h_2}(t)$. *Supposons le premier cas (l'autre étant symétrique). Nous prouvons l'énoncé par induction sur la profondeur (longueur maximale des adresses)* k *du préfixe.*

- $k = 0$

  Le préfixe t *est soit l'arbre du jeu complet, donc* π *est terminale ; dans ce cas* $v_{p,h_1}(t) = p(\pi) = (\rho, h)$, *donc* $h = \text{history}(\pi)$. *Soit il s'agit d'un préfixe radical à l'adresse* ε, *dans ce cas* $v_{p,h_1}(t) = h_1(\pi) = (\rho, h)$, *donc* $h = \text{history}(\pi)$.

- $k \Rightarrow k + 1$

  Si t *est un préfixe de longueur* $k + 1$, *tous se fils* t_i *sont des préfixes des arbres des positions* π_i *qui suivent* π *dans le jeu. Par hypothèse d'induction, les valeurs de chacune partagerons la même histoire* h_i. *Supposons* $\lambda(\pi) = \text{Player}$. *Par la règle (Player moves) du jeu, l'histoire* h_i *des positions* π_i *est celle de la position* π *augmentée par l'ensemble des variables nouvelles introduites pour utiliser la* i*-ème clause du programme permettant d'avancer dans le jeu. Donc* $h_i = h.\nu_i$. *Dans les règles du jeu nous avons assumé que la fonction* ξ_V *utilisée pour la génération des variables, est telle que tous les* ν_i *et toutes les parties de l'histoire qui s'ajouterons à partir des positions* π_i, *seront disjointes. Grâce à la règle (Player suicide), il y aura au moins un deuxième coup dont la composante histoire sera* h. *Alors, puisque* $v_{p,h_1}(t) = \overset{n}{\underset{i=1}{\Uparrow}} v_{p,h_1}(t_i)$, *la fonction du joueur restituera, dans la* *composante histoire, le plus grand préfixe commun, c'est-à-dire* $h = \text{history}(\pi)$. *Supposons en revanche* $\lambda(\pi) = \text{Opponent}$. *Par la règle (Oppon. moves) les positions* π_i *ont la même composante histoire que leur antécédente* π. *La valeur du préfixe sera alors* $v_{p,h_1}(t) = \downarrow_{i=1}^{n} v_{p,h_1}(t_i)$. *La fonction de l'opposant restitue elle aussi, comme celle du joueur, le plus grand préfixe commun des histoires* h_i *qui sont toutes égales à* $h = \text{history}(\pi)$.

□

8.2.4. La relation sur les valeurs. Le rôle de la composante histoire est celui de permettre la comparaison des valeurs. Lorsque deux valeurs (ρ_1, h_1) et (ρ_2, h_2) ont une partie d'histoire commune, c'est-à-dire lorsque le plus grand préfixe commun p est non vide ($h_1 = p.s$ et $h_2 = p.s'$), nous pouvons :

(1) construire deux EX-réponses par la clôture existentielle des variables réciproquement étrangères, i.e. :

$$\rho_1' = \exists X_1..\exists X_k.\,\rho_1 \qquad \text{et} \qquad \rho_2' = \exists Y_1..\exists Y_k.\,\rho_2$$

où $\{X_1, .., X_k\} = \bigcup_j s_j$, et $\{Y_1, .., Y_m\} = \bigcup_j s_j'$; ensuite :

(2) comparer les deux formules ainsi obtenues par l'ordre d'implication logique.

Cette méthode de comparaison des valeurs s'applique à n'importe quel couple de valeurs du domaine D. Plus en général, nous pouvons comparer deux formules sur une selection de variables relevantes H, en faisant, de façon préliminaire, la clôture existentielle de toutes les variables non présentes dans l'ensemble H, et ensuite en comparant par l'implication logique les deux formules ainsi obtenues.

Definition 8.2.12. (Préordre d'implication logique modulo H) *Étant donné une algèbre de contraintes* χ *de signature* $\Sigma_\chi = (\mathcal{B}, S_\chi, F_\chi, T_\chi)$, *une signature logique* $\Sigma = (\mathcal{B}, S, F, T)$ *sur* χ, *et un ensemble infini de variables sortées* $V = \biguplus_{s \in S} V_s$ *disjoint de* $\mathcal{B}$, S *et* F, *deux formules* $\varphi_1, \varphi_2 \in \breve{\Phi}(\Sigma, V)$, *sont dans l'ordre d'*implication (logique) modulo H, *noté* $\varphi_1 \subseteq_H \varphi_2$, *si l'algèbre des contraintes* χ *est un modèle de la clôture universelle de la formule logique* $\exists(s).\varphi_1 \Rightarrow \exists(r).\varphi_2$:

$$\varphi_1 \subseteq_H \varphi_2 \qquad \textit{ssi} \qquad \chi \models \forall(\exists(s).\varphi_1 \Rightarrow \exists(r).\varphi_2)$$

où $s = FV(\varphi_1)\backslash H$ *et* $r = FV(\varphi_2)\backslash H$ □

Deux valeurs du jeu $x = (\rho, h)$ et $y = (\rho', h')$ peuvent être comparées d'un point de vue logique, c'est-à-dire par leurs composantes logiques ρ et ρ', après avoir operé sur ces deux formules la clôture existentielle des variables mutuellement étrangères. Pour cela, on peut définir, en premier lieu, une relation $\leq_H$ contenant les couples de valeurs dont le plus grand préfixe commun p des histoires est tel que $H = \bigcup_i p_i$, et dont les composantes logiques sont dans le préordre d'implication logique $\subseteq_H$. Ensuite, on peut décreter que les deux valeurs sont comparables, c'est-à-dire $x \leq_D y$, si elles le sont sur leur plus grand préfixe commun, c'est-à-dire, s'il

TAB. 2. Relation $\leq_D$ entre valeurs du jeu CLP(χ)

$$\frac{\chi \models \forall(\varphi_1 \Rightarrow \varphi_2)}{\varphi_1 \subseteq \varphi_2}$$

$$\frac{\exists(s).\varphi_1 \subseteq \exists(r).\varphi_2 \qquad s = FV(\varphi_1)\backslash H \qquad r = FV(\varphi_2)\backslash H}{\varphi_1 \subseteq_H \varphi_2}$$

$$\frac{\rho \subseteq_H \rho' \qquad p = \inf_{\sqsubseteq}\{h, h'\} \qquad H = \bigcup_i p_i}{(\rho, h) \leq_H (\rho', h')}$$

$$\frac{\exists H.\ (\rho, h) \leq_H (\rho', h')}{(\rho, h) \leq_D (\rho', h')}$$

existe H tel que $x \leq_H y$. Les définitions des préordres d'implications, $\subseteq$ et $\subseteq_H$, et des relations de comparaison entre valeurs du jeu $\leq_H$ et $\leq_D$, sont résumées par la table 2.

La relation $\leq_D$ définie sur D vérifie la condition que la sous-structure $(D^{\pm}_{\pi}, \leq_{D^{\pm}_{\pi}})$ soit un préordre pour toute position $\pi \in \mathbb{P}$ du jeu. En effet, si la relation $\leq_D$est restreinte aux valeurs pessimistes ou optimistes de la position donnée, alors, par la proposition 8.2.11, elle est restreinte à des valeurs qui partagent la même histoire. Sur cette classe de valeurs, la relation $\leq_D$ coïncide avec la relation $\leq_H$, où $H = \bigcup_i h_i$ et $h = \text{history}(\pi)$, qui est un préordre.

8.2.5. Monotonie dans les composantes. La deuxième condition requise aux structures de co-évaluation monotones est que les fonctions $\uparrow$ et $\downarrow$ soient monotones dans toutes leurs composantes par rapport au valeurs pessimistes ou optimistes du jeu (définies par la donnée du payoff et des fonctions heuristiques).

L'ensemble des valeurs pessimiste et optimiste $D^{\pm}_{\pi}$ est la réunion des résultats de la rétro-propagation sur les préfixes finis du jeu, obtenus avec une quelconque des deux heuristiques. En terme des noyaux binaires (qui sont des opérations associative et commutative), la propriété souhaitée devient :

$$\forall x, y \in D^{\pm}_{\pi_i}.\ \forall z \in D^{\pm}_{\pi_j}. \quad x \leq_{D^{\pm}_{\pi_i}} y \Rightarrow \begin{cases} x \overset{2}{\uparrow} z \leq_{D^{\pm}_{\pi}} y \overset{2}{\uparrow} z \\ \\ x \overset{2}{\downarrow} z \leq_{D^{\pm}_{\pi}} y \overset{2}{\downarrow} z \end{cases}$$

où π_i et π_j sont de positions qui suivent π dans le jeu.

8.2.5.1. *Preuve de monotonie.* Pour montrer cette condition nous définissons deux nouvelle relations, sous-ensembles de la relation $\leq_{\mathrm{D}}$. La première, que nous noterons $\leq_1$, capture les couples de valeurs (x, y) où l'histoire de x est un préfixe de l'histoire de y. Vice-versa, la seconde, que nous noterons $\leq_2$, capture les couples (x, y) où l'histoire de y est un préfixe de l'histoire de x :

$$\begin{aligned} \leq_1 &= \{(x,y) \in \leq_{\mathrm{D}} \mid x = (\rho, h),\ y = (\rho, h.s)\} \\ \leq_2 &= \{(x,y) \in \leq_{\mathrm{D}} \mid x = (\rho, h.s),\ y = (\rho, h)\} \end{aligned}$$

Les deux relations $\leq_1$ et $\leq_2$ sont des relations de préordre sur D. Des deux propriétés à prouver, la réflexive et la transitive, la première est une conséquence triviale de la propriété réflexive du préordre $\subseteq_{\mathrm{H}}$. Le lemme suivant permet de prouver la propriété transitive.

LEMMA 8.2.13. *Si* $\rho \subseteq \rho'$ *alors* $\exists(s).\rho \subseteq \exists(s).\rho'$

DÉMONSTRATION. *L'énoncé se traduit en affirmant que la clôture universelle* $\forall(\rho \Rightarrow \rho')$*, est une formule plus forte, d'un point de vue logique, que la clôture universelle* $\forall(\exists(s).\rho \Rightarrow \exists(s).\rho')$*. Supposons :*

$$\chi \models \forall(\rho \Rightarrow \rho')$$

Dans une logique du premier ordre, tout modèle de $\forall x.(A \Rightarrow B)$ *est un modèle de* $\exists x.(A \Rightarrow B)$*, et tout modèle de cette dernière est un modèle de* $(\exists x.A) \Rightarrow (\exists x.B)$ *(cf. par exemple* **[Lassaigne & Rougemont, 1993]***). Autrement dit,* $\chi \models \forall(\exists(s).\rho \Rightarrow \exists(s).\rho')$*, qui signifie* $\exists(s).\rho \subseteq \exists(s).\rho'$. □

LEMMA 8.2.14. *Les relations de comparaison* $\leq_1$ *et* $\leq_2$ *sont des relations de préordre sur* D

DÉMONSTRATION. *Nous le prouvons pour* $\leq_1$*, le cas de* $\leq_2$ *étant symétrique. La relation est réflexive (puisque la relation* $\subseteq_{\mathrm{H}}$ *entre formules est réflexive). Pour prouver la propriété transitive, supposons d'avoir trois valeurs* $\nu = (\rho, p)$*,* $\nu' = (\rho', p.s')$ *et* $\nu'' = (\rho'', p.s'.s'')$ *telles que* $\nu \leq_1 \nu'$ *et* $\nu' \leq_1 \nu''$*. Donc* $\rho \subseteq \exists(s').\rho'$ *et* $\rho' \subseteq \exists(s'').\rho''$*. Par le lemme 8.2.13,* $\rho' \subseteq \exists(s'').\rho''$ *implique* $\exists(s').\rho' \subseteq \exists(s').\exists(s'').\rho''$*, donc* $\rho \subseteq \exists(s').\exists(s'').\rho''$ *par la propriété transitive de l'implication logique* $\subseteq$*, donc* $\nu \leq_1 \nu''$, c.q.f.d. □

Nous allons prouver, dans un premier temps, que l'opération du joueur est monotone par rapport à la relation $\leq_1$, tandis que l'opération de l'opposant est monotone par rapport à l'ordre $\leq_2$.

LEMMA 8.2.15. *La fonction* $\overset{2}{\uparrow}$ *est monotone par rapport à* $\leq_1$*, la fonction* $\overset{2}{\downarrow}$ *est monotone par rapport à* $\leq_2$ *:*

$$\forall x, y, z \in \mathrm{D}. \quad \begin{cases} x \leq_1 y & \Rightarrow \quad x \overset{2}{\uparrow} z \leq_1 y \overset{2}{\uparrow} z \\ \\ x \leq_2 y & \Rightarrow \quad x \overset{2}{\downarrow} z \leq_2 y \overset{2}{\downarrow} z \end{cases}$$

DÉMONSTRATION. *Nous prouvons la monotonie de* $\leq_1$. *Soit* $x \leq_1 y$, *où* $x = (\rho, h)$ *et* $y = (\rho', h')$, *où* $h' = h.s$. *Soit* $z = (\rho'', h'')$. *Soit* $p = \inf_{\sqsubseteq}(h, h'')$ *le préfixe sur lequel peuvent être comparées les valeurs* x *et* z, *et soit* $p' = \inf_{\sqsubseteq}(h.s, h'')$ *le préfixe sur lequel peuvent être comparées les valeurs* y *et* z. *Nous avons deux cas possibles, soit* p *est un préfixe stricte de la suite* h, *soit il est égal à* h.

- $p \sqsubset h$

  Dans ce cas x *et* y *se comparent à* z *sur le préfixe* p. *L'histoire de* x *est* $h = p.r$, *celle de* y *est* $h' = h.s = p.r.s$, *celle de* z *est* $h'' = p.r''$. *L'opération du joueur, appliquée au couple* (x, z) *et au couple* (y, z), *renvoie les résultats suivants :*

  $$\begin{array}{rcl} x \overset{2}{\uparrow} z &=& (\rho \oslash \rho'', p) \\ y \overset{2}{\uparrow} z &=& (\rho' \oslash \rho'', p) \end{array}$$

  renvoie un résultat tel que la composante histoire est p. *Soient* $\rho_1 = \rho \oslash \rho''$ *et* $\rho_2 = \rho' \oslash \rho''$. *Puisque les variables de* r'' *sont disjointes de celles de* r *et de* s, *nous avons les équivalences logiques :*

  $$\begin{array}{lcl} \exists(r).\exists(r'').\rho_1 &\equiv& (\exists(r).\rho) \vee (\exists(r'').\rho'') \\ \exists(r).\exists(s).\exists(r'').\rho_2 &\equiv& (\exists(r).\exists(s).\rho') \vee (\exists(r'').\rho'') \end{array}$$

  L'hypothèse $x \leq_1 y$ *implique* $\rho \subseteq \exists(s).\rho'$. *Donc par le lemme 8.2.13,* $\exists(r).\rho \subseteq \exists(r).\exists(s).\rho'$. *Donc* $\rho_1 \subseteq_H \rho_2$, *où* $H = \bigcup_i p$, *c'est-à-dire* $x \overset{2}{\uparrow} z \leq_1 y \overset{2}{\uparrow} z$.

- $p = h$

  Dans ce cas x *et* z *se comparent sur* $h = p$, *tandis que* y *et* z *se comparent sur une histoire* p' *éventuellement plus grande que* h *mais pas supérieure à* $h' = h.s$. *L'histoire de* x *est* $h = p$, *celle de* y *est* $h' = h.s = h.s'.r'$, *où* $s = s'.r'$ *et* $p' = h.s'$, *celle de* z *est* $h'' = p.t'' = h.s'.r''$. *L'opération du joueur, appliquée au couple* (x, z) *et au couple* (y, z), *renvoie les résultats suivants :*

  $$\begin{array}{rcl} x \overset{2}{\uparrow} z &=& (\rho \oslash \rho'', h) \\ y \overset{2}{\uparrow} z &=& (\rho' \oslash \rho'', h.s') \end{array}$$

  Soient ρ_1 *et* ρ_2 *définis par :*

  $$\begin{array}{rcl} \rho_1 &=& \exists(s').\exists(r'').\rho \vee \rho'' \\ \rho_2 &=& \exists(r').\exists(r'').\rho' \vee \rho'' \end{array}$$

  Nous devons prouver que $x \overset{2}{\uparrow} z \leq_1 y \overset{2}{\uparrow} z$, *donc* $\rho_1 \subseteq_H \rho_2$, *où* $H = \bigcup_i p$, *c'est-à-dire* $\rho_1 \subseteq \exists(s').\rho_2$. *Puisque les variables de* r'' *sont disjointes de celles de* r' *et de* s', *et puisque tout modèle de* $\exists x.(A \vee B)$ *est un modèle de* $(\exists x.A) \vee (\exists x.B)$ *et réciproquement, les deux formules sont équivalentes aux suivantes :*

  $$\begin{array}{lcl} \rho_1 &\equiv& \rho \vee (\exists(s').\exists(r'').\rho'') \\ \exists(s').\rho_2 &\equiv& (\exists(s').\exists(r').\rho') \vee (\exists(s').\exists(r'').\rho'') \end{array}$$

L'hypothèse $x \leq_1 y$ *implique* $\rho \subseteq \exists(s').\exists(r').\rho'$, *donc* $\rho_1 \subseteq \exists(s').\rho_2$, c.q.f.d.

La preuve de monotonie de $\overset{2}{\downarrow}$ *par rapport à* $\leq_2$ *est symétrique. Plus précisément, le cas* $p \sqsubset h$ *est identique : la preuve s'appuie seulement sur le fait qu'une clôture existentielle* $\exists(r).(\rho \vee \rho')$ *est équivalente* *à $\rho \vee (\exists(r).\rho')$ si ρ ne contient pas les variables libres en r. Cela indépendamment de l'opérateur logique $\vee$ou $\wedge$ utilisé pour composer les formules. En revanche, la preuve pour le cas $p = h$ s'appuie précisément sur le fait que tout modèle de $(\exists x.A) \vee (\exists x.B)$ est un modèle de $\exists x.(A \vee B)$. Pour la preuve de monotonie de $\overset{2}{\downarrow}$ par rapport à $\leq_2$, nous utiliserons l'implication duale, c'est-à-dire que tout modèle de $\exists x.(A \wedge B)$ est un modèle de $(\exists x.A) \wedge (\exists x.B)$.* □

La propriété de monotonie requise aux fonctions des joueurs par rapport à $\leq_{\text{D}}$ est une conséquence directe du lemme précédent et de la proposition 8.2.11.

COROLLARY 8.2.16. *Les fonctions des joueurs sont telles que, si* $\text{Succ}(\pi) = \pi_1..\pi_n$*, alors :*

$$\forall x, y \in D^{\pm}_{\pi_i}.\ \forall z \in D^{\pm}_{\pi_j}. \quad x \leq_{D^{\pm}_{\pi_i}} y \Rightarrow \begin{cases} x \overset{2}{\uparrow} z \leq_{D^{\pm}_{\pi}} y \overset{2}{\uparrow} z \\ \\ x \overset{2}{\downarrow} z \leq_{D^{\pm}_{\pi}} y \overset{2}{\downarrow} z \end{cases}$$

DÉMONSTRATION. *Par la proposition 8.2.11 nous savons que les valeurs d'une même position ont la même composante histoire. Étant donné $x, y \in D^{\pm}_{\pi_i}$, si l'on suppose $x \leq_{D^{\pm}_{\pi_i}} y$, puisque x et y ont la même histoire, il sont à la fois dans le préordre $\leq_1$ et dans le préordre $\leq_2$. Par le lemme précédent $x \leq_1 y$ implique $x \overset{2}{\uparrow} z \leq_1 y \overset{2}{\uparrow} z$ donc $x \overset{2}{\uparrow} z \leq_{D^{\pm}_{\pi}} y \overset{2}{\uparrow} z$ (puisque $\leq_{\text{D}}$ contient $\leq_1$). D'autre part, $x \leq_2 y$ implique $x \overset{2}{\downarrow} z \leq_2 y \overset{2}{\downarrow} z$ donc $x \overset{2}{\downarrow} z \leq_{D^{\pm}_{\pi}} y \overset{2}{\downarrow} z$ (puisque $\leq_{\text{D}}$ contient aussi $\leq_2$),* c.q.f.d. □

8.2.6. Correction des heuristiques. Nous devons apurer que les heuristiques définies vérifient la propriété de correction. On demande que pour toute position $\pi \in \mathbb{P}_{\text{NT}}$, où $\text{Succ}(\pi) = \pi_1..\pi_n$ $(n \geq 1)$, les conditions suivantes soient vérifiées :

$$\lambda(\pi) = \text{Player} \quad \Rightarrow \begin{cases} h_1(\pi) \leq_{D^{\pm}_{\pi}} \left(\overset{n}{\underset{i=1}{\uparrow}} p \oplus h_1(\pi_i) \right) \\ \\ h_2(\pi) \geq_{D^{\pm}_{\pi}} \left(\overset{n}{\underset{i=1}{\uparrow}} p \oplus h_2(\pi_i) \right) \end{cases}$$

$$\lambda(\pi) = \text{Opponent} \quad \Rightarrow \begin{cases} h_1(\pi) \leq_{D^{\pm}_{\pi}} \left(\overset{n}{\underset{i=1}{\downarrow}} p \oplus h_1(\pi_i) \right) \\ \\ h_2(\pi) \geq_{D^{\pm}_{\pi}} \left(\overset{n}{\underset{i=1}{\downarrow}} p \oplus h_2(\pi_i) \right) \end{cases}$$

LEMMA 8.2.17. *Les fonctions heuristiques du jeu CLP(χ) sont correctes.*

DÉMONSTRATION. *D'une part, la fonction pessimiste est telle que :*

$$h_1(\pi) = (\text{false}, \text{history}(\pi))$$

Tout autre valeur de cette position aura la même histoire, est sera forcément meilleure, d'un point de vue logique, que la formule false. *D'autre part, la fonction optimiste est telle que* $h_2(\pi) = (c, \text{history}(\pi))$. *Si la position est de l'opposant, toutes les positions suivantes auront la même composante contrainte et la même composante histoire :*

$$\forall \pi_i \in \text{Succ}(\pi). \quad p \oplus h_2(\pi_i) = (c, h) \quad \textit{où } h = \text{history}(\pi)$$

Le résultat de $\downarrow$ *sur toutes ces valeurs sera donc encore* (c, h), *ce qui implique :*

$$h_2(\pi) \leq_{D^{\pm}_{\pi}} \left(\overset{n}{\underset{i=1}{\downarrow}} p \oplus h_2(\pi_i) \right)$$

puisque $\leq_{D^{\pm}_{\pi}}$ *est un préordre, donc une relation réflexive. En revanche, si la position est du joueur, les positions suivantes auront des contraintes plus fortes (l'application d'une clause de programme ajoute des contraintes aux anciennes), et elles auront aussi une histoire plus grande (puisque de nouvelles variables auront étés introduites). Cependant, le résultat de l'application de la fonction* $\uparrow$ *sur toutes les valeurs* $p \oplus h_2(\pi_i)$ *est une valeur dont l'histoire est encore* $h = \text{history}(\pi)$. *Or, la partie logique, est la disjonction des parties logiques de* $p \oplus h_2(\pi_i) = x_i$, *c'est-à-dire la disjonction de formules* x_i *impliquant chacune la contrainte* c. *Puisque tout modèle de* $(x_1 \Rightarrow c \ \wedge .. \wedge \ x_n \Rightarrow c)$ *est un modèle de* $(x_1 \vee .. \vee x_n) \Rightarrow c$, *nous avons :*

$$h_2(\pi) \leq_{D^{\pm}_{\pi}} \left(\overset{n}{\underset{i=1}{\uparrow}} p \oplus h_2(\pi_i) \right)$$

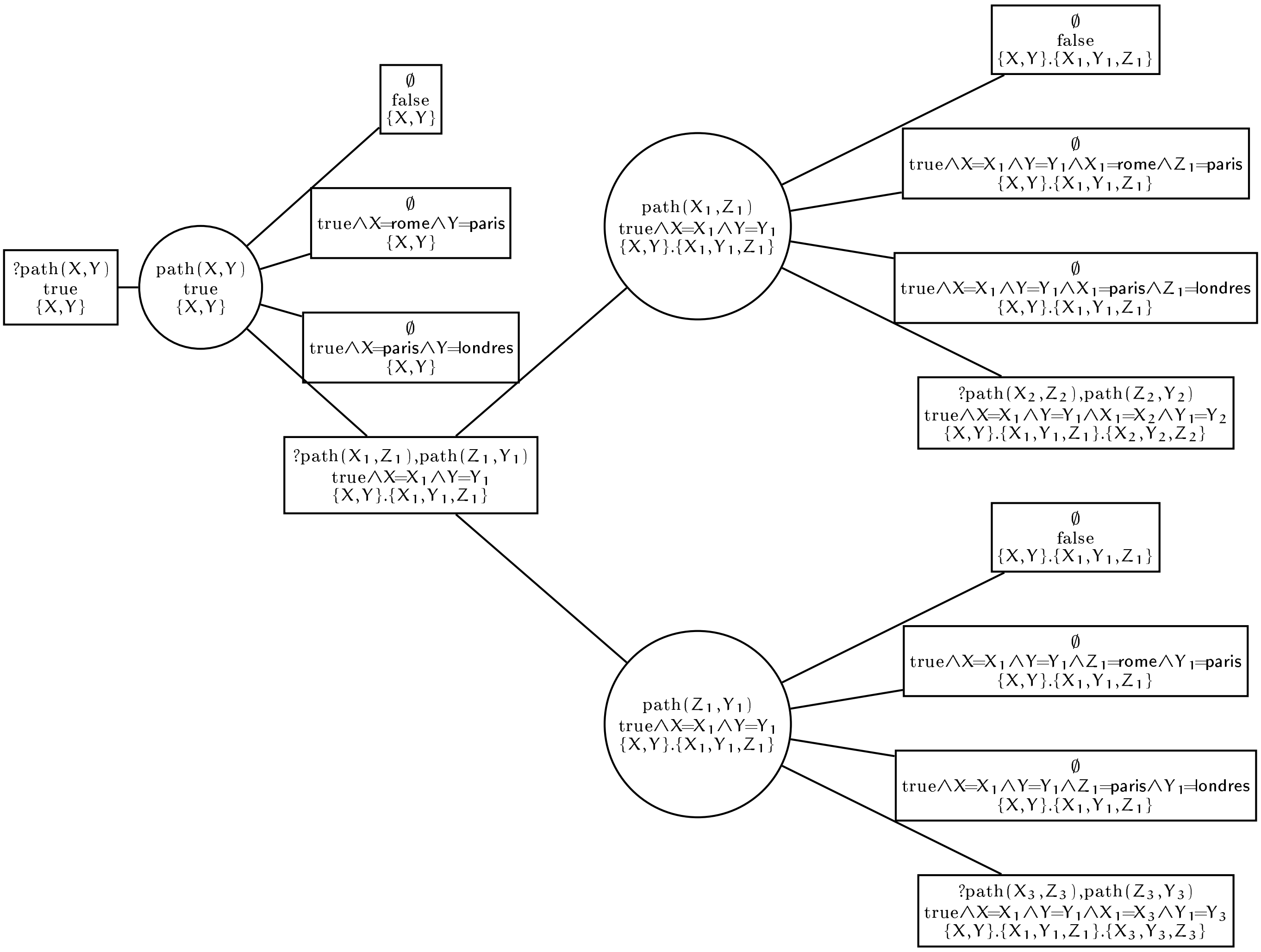
c.q.f.d. □

8.2.7. Heuristiques dans le préordre extensionnel. La dernière condition à vérifier est que les heuristiques soient dans le préordre extensionnel :

$$\forall \pi \in \mathbb{P}. \quad h_1(\pi) \leq_{D^{\pm}_{\pi}} h_2(\pi)$$

Or, $h_1(\pi) = (\text{false} \wedge c, \text{history}(\pi))$ tandis que $h_2(\pi) = (c, \text{history}(\pi))$, où c est la composante contrainte du quadruplet π. Donc les fonctions sont trivialement dans l'ordre extensionnel puisque false $\subseteq c$.

FIG. 8.2.1. Exemple programme path



8.2.8. Conclusion. Au cours de cette section, nous avons progressivement défini une algèbre $\mathcal{D} = (\mathsf{D}, \leq_{\mathsf{D}}, \uparrow, \downarrow)$, et nous avons prouvé qu'il s'agit bien d'une structure de co-évaluation :

$$\mathcal{D} \in \mathsf{Cstruct}(\mathcal{S}, \mathsf{D}, \uparrow, \downarrow, \mathrm{p}, \mathrm{h}_1, \mathrm{h}_2)$$

pour la fonction de payoff et les fonctions heuristiques définies pour le jeu CLP(χ). Ainsi, nous avons, pour le jeu CLP(χ), toutes les fonctions sémantiques définies en général pour les jeux évalués dans une structure de co-évaluation monotone. Considérons par exemple le programme logique pur (χ est l'algèbre des termes de Herband avec le prédicat d'égalité) :

```
path(rome,paris).
path(paris,londres).
path(X,Y) :- path(X,Z),path(Z,Y).
```

et supposons de vouloir résoudre le but `path(X,Y)`. En terme de jeu, cela signifie calculer la sémantique de la position :

$$\pi = (\mathcal{P}, \text{path}(\mathsf{X}, \mathsf{Y}), \text{true}, \{\mathsf{X}, \mathsf{Y}\})$$

où $\mathcal{P}$ est le programme contenant les trois règles ci-dessus. Par la suite, comme dans la figure, nous omettrons les premières composantes des quadruplets, constamment égales au programme. L'arbre de position π, qui est un arbre de jeu infini, est représenté dans la figure 8.2.1 jusqu'à la profondeur 5. Déjà à la profondeur 3, le préfixe $t_\pi/^3$ fournit des informations intéressantes sur la sémantique pessimiste de la position. En effet, à ce niveau de l'arbre, la retro-propagation min-max, avec l'heuristique pessimiste, renvoie une disjonction de quatres χ-contraintes :

$$(\text{false} \vee (\text{true} \wedge \mathsf{X} = \mathsf{rome} \wedge \mathsf{Y} = \mathsf{paris}) \vee (\text{true} \wedge \mathsf{X} = \mathsf{paris} \wedge \mathsf{Y} = \mathsf{londres}) \vee \text{false}, \{\mathsf{X}, \mathsf{Y}\})$$

La première contrainte (false), correspond au mouvement suicidaire du joueur (qui ne lui apporte jamais rien) ; la deuxième contrainte ($\text{true} \wedge \mathsf{X} = \mathsf{rome} \wedge \mathsf{Y} = \mathsf{paris}$) correspond à l'application de la première clause `path(rome,paris)` du programme ; la troisème contrainte ($\text{true} \wedge \mathsf{X} = \mathsf{paris} \wedge \mathsf{Y} = \mathsf{londres}$) correspond à l'application de la deuxième clause `path(paris,londre)` du programme ; enfin, la quatrième contrainte (false) correspond à la valeur pessimiste de la position atteinte après l'application de la troisième clause `path(X,Y) :- path(X,Z),path(Z,Y)`. La disjonction obtenue est donc équivalente à la valeur :

$$((\mathsf{X} = \mathsf{rome} \wedge \mathsf{Y} = \mathsf{paris}) \vee (\mathsf{X} = \mathsf{paris} \wedge \mathsf{Y} = \mathsf{londres}), \{\mathsf{X}, \mathsf{Y}\})$$

En revanche, au niveau 5, l'approximation pessimiste donne un résultat équivalent à la valeur :

$$((\mathsf{X} = \mathsf{rome} \wedge \mathsf{Y} = \mathsf{paris}) \vee (\mathsf{X} = \mathsf{paris} \wedge \mathsf{Y} = \mathsf{londres}) \vee (\mathsf{X} = \mathsf{rome} \wedge \mathsf{Y} = \mathsf{londres}), \{\mathsf{X}, \mathsf{Y}\})$$

qui représente (appliquée au but initial) l'ensemble des conséquences logiques du programme. Par ailleurs, on vérifie facilement que les approximations optimistes sont constamment égales à une valeur équivalente à $(\text{true}, \{\mathsf{X}, \mathsf{Y}\})$. Cela signifie que si l'on ajoute au programme $\mathcal{P}$ la formule logique $\forall \mathsf{X}, \mathsf{Y}. \text{path}(\mathsf{X}, \mathsf{Y})$, l'ensemble obtenu est un ensemble de formules (clauses) *consistent*. Dans la prochaine section nous allons prouver formellement l'adéquation de la sémantique pessimiste du jeu à la sémantique traditionnelle de CLP(χ).

8.3. Adéquation à la sémantique opérationnelle de CLP(χ)

Puisque la structure $\mathcal{D} = (\mathsf{D}, \leq_{\mathsf{D}}, \uparrow, \downarrow)$, définie pour le jeu CLP(χ) est une structure de co-évaluation $\mathcal{D} \in \mathsf{Cstruct}(\mathcal{S}, \mathsf{D}, \uparrow, \downarrow, \mathsf{p}, \mathsf{h}_1, \mathsf{h}_2)$, nous avons plusieurs fonctions sémantiques conséquentes pour toute position du jeu : la sémantique pessimiste $\llfloor \pi \rrfloor_{\mathcal{D}}$, l'optimiste $\llceil \pi \rrceil_{\mathcal{D}}$ et la sémantique bisémique $[\![\pi]\!]_{\mathcal{D}}$. Il est toutefois intéressant, pour le jeu CLP(χ), de caracteriser la sémantique (pessimiste) en fonction des stratégies du joueur.

8.3.1. Propriétés des stratégies du jeu CLP(χ). Étant donné une valeur du jeu $\nu = (\rho, h) \in D$, nous appellerons *contenu logique* la composante ρ de ν. Les stratégies du jeu CLP(χ) ont des propriétés spécifiques qui les rendent intéréssantes d'un point de vue logique. En premier lieu, puisqu'une stratégie est un élagage déterministe dans les noeuds du joueur et complet dans ceux de l'opposant, aucune adresse du joueur n'aura l'effet de produire une disjonction de formules. En d'autres termes, le contenu logique de toute stratégie (ou pré-stratégie, cf. définition 5.6.12) sera une conjonction de primitives, c'est-à-dire une χ-contrainte. En second lieu, nous pouvons remarquer que l'heuristique pessimiste est, dans sa composante logique (toujours égale à false), une sorte d'élément minimal. Alors, par la proposition 5.6.15, qui est applicable puisque les opérations $\downarrow$ et $\uparrow$ sont plurielles (i.e. $x = x \uparrow x = x \downarrow x$), nous avons que la valeur d'une pré-stratégie est égale à :

$$v_{p,h}(s) = \underset{\pi' \in \mathrm{Feuilles}(s)}{\downarrow} p \oplus h_1(\pi')$$

Donc, toute pré-stratégie propre, c'est-à-dire tout préfixe d'une stratégie contenant des feuilles étiquetées avec une position non terminale, aura un contenu logique égal à false. Intuitivement, il suffira d'appliquer l'heuristique pessimiste une seule fois et à une adresse quelconque pour que le contenu logique false remonte à la racine de la stratégie.

8.3.2. Stratégies gagnantes. Les stratégies gagnantes pour le jeu CLP(χ) sont les stratégies finies (donc des pré-stratégies) dont le contenu logique n'est pas équivalent à la contrainte false.

DEFINITION 8.3.1. (STRATÉGIES GAGNANTES POUR LE JEU CLP(χ)) *Une stratégie du joueur* $s \in \mathrm{IS}_p(\mathcal{S}, \pi)$ *pour la position* π *est une stratégie* gagnante *(du joueur) si :*

(1) *elle est finie, donc* $v_{p,h_1}(s) = v_{p,h_2}(s) = (\rho, h)$
(2) *elle ne représente pas le pire des scores :* $\rho \not\equiv \mathrm{false}$

Nous noterons le contenu logique ρ *de la stratégie gagnante par* $v_p(s)$*, et nous noterons l'ensemble de toutes les stratégies gagnantes de la position* π *par* $\mathrm{WS}(\pi)$. □

Il y a deux types de stratégies finies, celles ne contenant pas le mouvement suicidaire et celle le contenant. Le payoff de toute position atteinte par un coup suicidaire, est égal, par définition de $p(.)$ dans le jeu CLP(χ), à la contrainte false. Donc, aucune stratégie finie contenant un coup suicidaire ne peut être gagnante.

8.3.3. Goal split and fusion. Le théorème de compositionalité de la résolution CSLD (cf. théorème 7.5.2) affirme en particulier (pour $n = 1$) que la dérivation $?d\,;\pi \overset{x}{\underset{\mathcal{P}}{\longmapsto}}{}^{*} c\,;\emptyset$ est équivalente à la condition $?\mathrm{true}\,;\pi \overset{x}{\underset{\mathcal{P}}{\longmapsto}}{}^{*} c'\,;\emptyset$ où $d \wedge c' = c$. Cette remarque conduit à un résultat utile à la preuve d'adéquation de la sémantique des jeux de CLP(χ). En effet, on peut déduire un corollaire qui généralise au cadre CLP les résultats présentés dans [**Di Cosmo et al.,1998**] (lemmes *goal fusion* et *goal split*) pour l'interprétation en termes de jeu de la programmation logique pure (LP).

COROLLARY 8.3.2. (GOAL SPLIT AND FUSION) *Soit* $\Sigma = (\mathcal{B}, S, F, T)$ *une signature logique sur* χ, *et* $V = \biguplus_{s \in S} V_s$ *un ensemble de variables sortées disjoint de* $\mathcal{B}$, S *et* F. *Alors :*

$$?\,d\,;\pi_1,..,\pi_n \overset{\chi}{\underset{\mathcal{P}}{\longmapsto}}{}^{*}\, c\,;\emptyset \qquad ssi \qquad \begin{cases} ?\,d\,;\pi_1 \overset{\chi}{\underset{\mathcal{P}}{\longmapsto}}{}^{*}\, c_1\,;\emptyset \\ \vdots \\ ?\,d\,;\pi_n \overset{\chi}{\underset{\mathcal{P}}{\longmapsto}}{}^{*}\, c_n\,;\emptyset \\ c \equiv c_1 \wedge .. \wedge c_n \end{cases}$$

où $n \geq 1$.

DÉMONSTRATION. *(⇒) D'une part, si* $?\,d\,;\pi_1,..,\pi_n \overset{\chi}{\underset{\mathcal{P}}{\longmapsto}}{}^{*}\, c\,;\emptyset$, *par le théorème de compositionalité, pour tout* $i = 1..n$, $?\,\mathrm{true}\,;\pi_i \overset{\chi}{\underset{\mathcal{P}}{\longmapsto}}{}^{*}\, c'_i\,;\emptyset$ *et* $c = d \wedge c'_1 \wedge .. \wedge c'_n$. *Toujours par le même théorème (avec* $n = 1$*), la condition* $?\,\mathrm{true}\,;\pi_i \overset{\chi}{\underset{\mathcal{P}}{\longmapsto}}{}^{*}\, c'_i\,;\emptyset$ *implique* $?\,d\,;\pi_i \overset{\chi}{\underset{\mathcal{P}}{\longmapsto}}{}^{*}\, c_i\,;\emptyset$, *où* $c_i = d \wedge c'_i$. *Donc* $c = d \wedge c'_1 \wedge .. \wedge c'_n \equiv (d \wedge c'_1) \wedge .. \wedge (d \wedge c'_n) = c_1 \wedge .. \wedge c_n$. *(⇐) D'autre part, supposons que pour tout* $i = 1..n$, $?\,d\,;\pi_i \overset{\chi}{\underset{\mathcal{P}}{\longmapsto}}{}^{*}\, c_i\,;\emptyset$ *et* $c = c_1 \wedge .. \wedge c_n$. *Par le théorème de compositionalité, pour tout* $i = 1..n$ *nous avons* $?\,\mathrm{true}\,;\pi_i \overset{\chi}{\underset{\mathcal{P}}{\longmapsto}}{}^{*}\, c'_i\,;\emptyset$ *où* $c_i = d \wedge c'_i$. *Toujours par ce même théorème,* $?\,d\,;\pi_1,..,\pi_n \overset{\chi}{\underset{\mathcal{P}}{\longmapsto}}{}^{*}\, c\,;\emptyset$ *où* $c = d \wedge c'_1 \wedge .. \wedge c'_n \equiv (d \wedge c'_1) \wedge .. \wedge (d \wedge c'_n) = c_1 \wedge .. \wedge c_n$. □

8.3.4. Adéquation. Le théorème d'adéquation affirme que l'existence d'une réponse calculée c pour un but donné $?\,d;b$, c'est-à-dire l'existence d'une dérivation CLSD de la forme $?\,d;b \overset{\chi}{\underset{\mathcal{P}}{\longmapsto}} ?\,c;\emptyset$, se traduit, en terme de jeu, dans l'existence d'une stratégie gagnante dont la valeur est logiquement équivalente à c.

THEOREM 8.3.3. (ADÉQUATION) *L'ensemble des réponses calculées par la résolution CSLD correspond à l'ensemble des stratégies gagnantes dans le jeu CLP(*χ*) :*

$$?\,d;b \overset{\chi}{\underset{\mathcal{P}}{\longmapsto}}{}^{*} ?\,c;\emptyset \qquad ssi \qquad \exists s \in \mathrm{WS}(\pi).\ v_p(s) \equiv c$$

où $\pi = < \mathcal{P}, ?b, d, h >$ *et* $h = FV(?d;b)$.

DÉMONSTRATION. *La preuve de ce résultat, pour la programmation logique pure (LP), se trouve dans* **[Di Cosmo et al.,1998]**, *et s'appuie principalement sur le lemme de compositionalité des réponses calculées. Pour l'étendre au cas du schéma CLP, il suffit d'utiliser la version pour CLP(*χ*) du lemme de compositionalité (cf. corollaire 8.3.2).*

(⇒) Par induction sur la longueur k *de la dérivation CSLD*

– $k = 1$

Le corps du but est forcément atomique $b = A_1$, *et il existe une clause de programme (la* i*-ème), s'unifiant avec* A_1, *dont la contrainte* c' *est telle que* $c = d \wedge c'$. *Alors, dans la position* $\pi =< \mathcal{P}, ?A_1, d, h >$ *du jeu, l'opposant a un seul coup disponible, conduisant à une position du joueur* $\pi' =< \mathcal{P}, A_1, d, h >$, *où le joueur peut utiliser la même clause et atteindre la position* $\pi'' =< \mathcal{P}, \emptyset, c, h'' >$. *On peut donc contruire la stratégie* $s = \{(\varepsilon, \pi)\} \cdot_1 \{(\varepsilon, \pi')\} \cdot_i \{(\varepsilon, \pi'')\}$ *dont la valeur est précisément* c.

– $k \Rightarrow k + 1$

On suppose $? d; b \overset{\chi}{\underset{\mathcal{P}}{\longmapsto}}{}^{k+1} ? c; \emptyset$. *Si* $b = A_1, .., A_n$, *par le corollaire 8.3.2, nous avons :*

$$\forall i = 1..n. \quad ? d; b \overset{\chi}{\underset{\mathcal{P}}{\longmapsto}}{}^{*} ? c_i; \emptyset$$

où $c_1 \wedge .. \wedge c_n \equiv c$. *Les dérivations individuelles seront de longueur non supérieure à* $k + 1$. *Nous avons :*

$$? d; A_i \overset{\chi}{\underset{\mathcal{P}}{\longmapsto}} ? d_i; b_i \overset{\chi}{\underset{\mathcal{P}}{\longmapsto}}{}^{k} ? c_i; \emptyset$$

Dans la position $\pi =< \mathcal{P}, (?A_1, .., A_n), d, h >$, *pour n'importe quel coup* A_i *que l'opposant choisisse, dans la nouvelle position* $\pi'_i =< \mathcal{P}, A_i, d, h >$, *le joueur aura un coup (par un règle* ℓ_i*) conduisant à une position* $\pi''_i =< \mathcal{P}, b_i, d_i, h_i >$ *à partir de laquelle il aura, par hypothèse d'induction, une stratégie gagnante* s_i *dont la valeur sera* c_i. *Il s'agit précisément de la définition de stratégie gagnante pour la position* π. *En effet, on peut construire la stratégie :*

$$s = \{(\varepsilon, \pi)\} \ \cdot_1 \{(\varepsilon, \pi'_1)\} \cdot_{1.\ell_1} s_1 \ \ldots \ \cdot_n \{(\varepsilon, \pi'_n)\} \cdot_{n.\ell_n} s_n$$

dont la valeur est $c_1 \wedge .. \wedge c_n \equiv c$.

($\Leftarrow$) *Par induction sur la profondeur* k *de la stratégie gagnante* s.

– $k = 0$

Si $s = \{(\varepsilon, \pi)\}$ *alors la position* π *est de l'opposant, donc* $\pi =< \mathcal{P}, ?b, d, h >$, *terminale, donc* $b = \emptyset$, *et telle que* $d \not\equiv \mathsf{false}$ *et* $d = c$. *La relation* $\overset{\chi}{\underset{\mathcal{P}}{\longmapsto}}{}^{*}$ *est la clôture réflexive et transitive de* $\overset{\chi}{\underset{\mathcal{P}}{\longmapsto}}$, *donc l'énoncé* $? d; b \overset{\chi}{\underset{\mathcal{P}}{\longmapsto}}{}^{*} ? d; b = ? c; \emptyset$.

– $k \Rightarrow k + 1$

La stratégie s *a* $n \geq 1$ *fils situés aux adresses* $\ell_1, .., \ell_n$. *Chaque fils est, par définition, lui même une stratégie* s_i, *correspondant au coup* $A_i \in b$ *de l'opposant, dont la valeur est* c_i. *Alors, par hypothèse d'induction,* $? d; A_i \overset{\chi}{\underset{\mathcal{P}}{\longmapsto}}{}^{*} ? c_i; \emptyset$, *et par le corollaire 8.3.2, l'énoncé* $? d; b \overset{\chi}{\underset{\mathcal{P}}{\longmapsto}}{}^{*} ? c; \emptyset$, *où* $c_1 \wedge .. \wedge c_n \equiv c$.

$\square$

Puisque les opérations des joueurs sont associatives et distributives, par la proposition 5.6.13, nous pouvons caracteriser toute valeur d'un préfixe fini par un ensemble de stratégies gagnantes ayant, collectivement, la même valeur du préfixe. Donc, nous pouvons approximer la sémantique pessimiste d'une position par l'ensemble de ses stratégies.

PROPOSITION 8.3.4. *La sémantique pessimiste des positions coïncide, si elle existe, avec la disjonction de toutes les stratégies gagnantes du jeu :*

$$\llbracket \pi \rrbracket_{\mathcal{D}}^{\flat} = \sup_{(D^\flat_\pi, \leq_{D^\flat_\pi})} \{ (\bigvee_{s \in WS(\pi,k)} v_p(s)), h) \mid k \in \mathbb{N}\}$$

où $h = \text{history}(\pi)$ *et* $WS(\pi,k) = \{s \in WS(\pi) \mid s \subseteq t_\pi/^k\}$ *est l'ensemble des stratégies gagnantes de profondeur non supérieure à* k.

DÉMONSTRATION. *Par le corollaire 5.6.14, nous avons :*

$$\llbracket \pi \rrbracket_{\mathcal{D}}^{\flat} = \sup_{(D^\flat_\pi, \leq_{D^\flat_\pi})} \{ \uparrow_{s \in \widehat{S}(\mathcal{S},\pi)_k} v_{p,h_1}(s) \mid k \in \mathbb{N}\}$$

Le contenu logique de toute pré-stratégie est false *si l'heuristique pessimiste est appliquée à une quelconque adresse de l'arbre. On peut alors simplifier la formule en éliminant toutes les pré-stratégie n'étant pas des stratégies gagnantes :*

$$\llbracket \pi \rrbracket_{\mathcal{D}}^{\flat} = \sup_{(D^\flat_\pi, \leq_{D^\flat_\pi})} \{ \uparrow_{s \in WS(\pi,k)} v_{p,h_1}(s) \mid k \in \mathbb{N}\}$$

D'une part, la composante histoire coïncide forcément avec celle des préfixes, donc avec celle de la position : $h = \text{history}(\pi)$. *D'autre part, la valeur des stratégies gagnantes, représentant le contenu logique, est fournit par la fonction* v_p, *donc :*

$$\llbracket \pi \rrbracket_{\mathcal{D}}^{\flat} = \sup_{(D^\flat_\pi, \leq_{D^\flat_\pi})} \{(\bigvee_{s \in WS(\pi,k)} v_p(s)), h) \mid k \in \mathbb{N}\}$$

□

En général, la borne supérieure peut ne pas exister dans la structure des valeurs $(D_\pi, \leq_{D_\pi})$ d'une position donnée π. Considérons, par exemple, le programme suivant dans le schéma CLP($\mathbb{N}$) :

```
p(0).
p(X+1) :- p(X).
```

Pour le but p(X), l'ensemble des réponses s'exprime par une disjonction infinie :

$$(X = 0) \vee (X = 1) \vee (X = 2) \vee ... \vee (X = k) \vee ...$$

qui peut être représentée dans la complétion par idéaux $(D^\flat_\pi, \leq_{D^\flat_\pi})$ du préordre $(D_\pi, \leq_{D_\pi})$. Autrement dit, la sémantique pessimiste bémol de la position est la classe de co-finalité du dirigé :

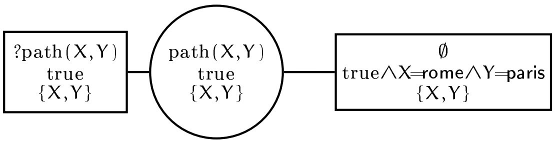
$$\{((X = 0) \vee ... \vee (X = k),\ h) \mid k \in \mathbb{N}\} \in \Delta(D_\pi, \leq_{D_\pi})$$

En revanche, la position $\pi = (\mathcal{P}, \text{path}(X,Y), \text{true}, \{X,Y\})$ analysée dans la figure 8.2.1 (page 270), est un exemple de position possédant une sémantique pessimiste, c'est-à-dire une position dont l'ensemble des approximants converge dans la structure $(D_\pi, \leq_{D_\pi})$. En effet, on peut vérifier que la sémantique pessimiste est égale à la disjonction des valeurs des trois stratégies gagnantes représentées dans la figure 8.3.1, c'est-à-dire :

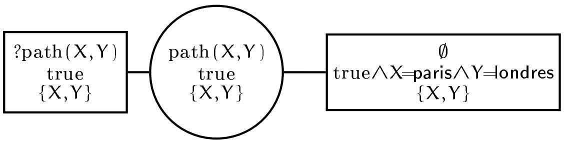
$$\llbracket \pi \rrbracket_{\mathcal{D}} = (\rho_1 \vee \rho_2 \vee \rho_3, \{X,Y\})$$

FIG. 8.3.1. Stratégies du programme *path*

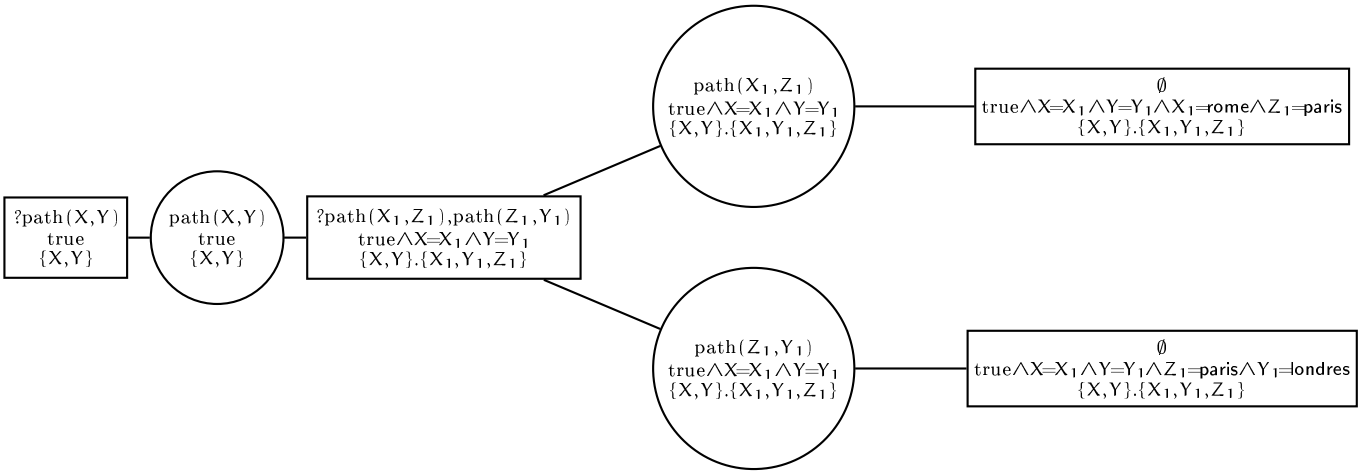
Stratégie gagnante s_1, contenu logique $\rho_1 \equiv (\mathsf{X}{=}\mathsf{rome} \wedge \mathsf{Y}{=}\mathsf{paris})$:



Stratégie gagnante s_2, contenu logique $\rho_2 \equiv (\mathsf{X}{=}\mathsf{paris} \wedge \mathsf{Y}{=}\mathsf{londres})$:



Stratégie gagnante s_3, contenu logique $\rho_3 \equiv (\mathsf{X}{=}\mathsf{rome} \wedge \mathsf{Y}{=}\mathsf{londres})$:



8.4. Application de l'algorithme Alpha-bêta pour CLP(χ)

Pour être en mesure d'appliquer la méthode Alpha-Bêta au jeu CLP(χ), il faut prouver que la structure de co-évaluation $\mathcal{D} \in \mathsf{Cstruct}(\mathcal{S}, \mathrm{D}, \uparrow, \downarrow, p, h_1, h_2)$, $\mathcal{D} = (\mathrm{D}, \leq_{\mathrm{D}}, \uparrow, \downarrow)$, définie pour ce jeu, vérifie tous les axiomes des structures $\alpha\beta$-normales, réunis dans la table 3 de la page 209.

8.4.1. Noyau binaire. Dans le jeu CLP(χ) les fonctions des joueurs sont générées par un noyau binaire, $\downarrow = \mathsf{ext}(\overset{2}{\downarrow})$ et $\uparrow = \mathsf{ext}(\overset{2}{\uparrow})$, et les deux opérations primitives sont associatives, commutatives et plurielles ($x \overset{2}{\downarrow} x = x \overset{2}{\uparrow} x = x$). Les axiomes (A1), (A2), (A3), (A4), (A5) sont donc vérifiés.

8.4.2. Congruence sur les valeurs du jeu. La relation d'équivalence forte $\equiv$ entre valeurs du jeu (cf. définition 8.2.2), est une notion trop stricte (ou trop concrète) qui rend impossibles les simplifications du calcul telles que les coupures de la méthode Alpha-Bêta. Elle doit être affaiblie en considérant que les variables générées dans le jeu CLP(χ), par le rennomage des clauses du programme, sont implicitement clôses de façon existentielle. Autrement dit, le sens (abstrait) que l'on attribue à une valeur (ρ, h), est celui de la formule ρ où toutes les variables de ρ n'apparaissant pas dans h sont implicitement clôses de façon existentielle. Par exemple, la valeur $(X = Y + 1, \{X\})$ s'interprète comme la formule $\exists Y.\, X = Y + 1$, qui, elle même, est quantifié implicitement, mais par une quantification universelle. Nous définissons alors le domaine des valeurs abstraites :

$$D_{\equiv}^{\text{EX}} \triangleq \{[\rho]_{\equiv}, h) \mid \rho \text{ est une EX–réponse, } h \in \wp^{(V^*)}\}$$

et la fonction d'abstraction des valeurs $\delta : D \to D_{\equiv}^{\text{EX}}$ transformant toute valeur (ρ, h) de D dans le couple $([\exists(\bar{h}).\rho]_{\equiv}, h)$, où $\bar{h}$ est l'ensemble des variables libres de ρ n'apparaissant pas dans h :

$$\delta(\rho, h) \triangleq ([\exists(\bar{h}).\rho]_{\equiv}, h) \qquad \text{où } \bar{h} = FV(\rho) \setminus \bigcup_i h_i$$

L'équivalence engendrée par l'abstraction est l'*équivalence faible* entre valeurs, notée $\approx$:

$$x \approx y \quad \stackrel{\triangle}{\Longleftrightarrow} \quad \delta(x) = \delta(y)$$

Pour l'application de la méthode Alpha-Bêta, nous utiliserons la famille d'équivalences constituée de la seule équivalence faible. Il s'agit d'une congruence pour les opérations des joueurs :

$$\begin{cases} \dfrac{x \approx x' \qquad x \uparrow y \approx z}{x' \uparrow y \approx z} \\[2ex] \dfrac{x \approx x' \qquad x \downarrow y \approx z}{x' \downarrow y \approx z} \end{cases}$$

En effet, si $x = (\rho, h) \approx (\rho', h') = y$, alors $h = h'$. Donc $x \uparrow z$, où $z = (\rho'', h'')$, aura forcément la même composante histoire de $y \uparrow z$, et elle sera plus courte (par effet de l'application de $\uparrow$) que l'histoire de x. Donc, si les deux formules ρ et ρ' sont équivalentes sur une histoire plus longue (qui provoque moins de quantifications existentielles), elles le seront sur une histoire h'' plus courte (qui provoque plus de quantifications existentielles). Donc, en ce qui concerne le contenu logique, la formule $\exists(\bar{h}''). \rho \circledvee \rho''$ sera équivalente à la formule $\exists(\bar{h}''). \rho' \circledvee \rho''$. La preuve de congruence pour $\downarrow$ est duale. L'axiome (A6) est donc vérifié.

8.4.3. Correction des coupures. Les axiomes de correction des coupures sont les suivants :

$$\frac{\lambda(\pi) = \text{Player} \qquad \text{cut}(\pi, \alpha, \beta, x, \approx) \qquad \alpha, \beta \in D \oplus \{\iota\} \qquad x, y \in E_\pi \oplus \{\iota\}}{\alpha \uparrow (\beta \downarrow x) \;\approx\; \alpha \uparrow [\beta \downarrow (x \uparrow y)]}$$

$$\frac{\lambda(\pi) = \text{Opponent} \qquad \text{cut}(\pi, \alpha, \beta, x, \approx) \qquad \alpha, \beta \in D \oplus \{\iota\} \qquad x, y \in E_\pi \oplus \{\iota\}}{\beta \downarrow (\alpha \uparrow x) \;\approx\; \beta \downarrow [\alpha \uparrow (x \downarrow y)]}$$

Nous prouvons le premier, la preuve du second étant duale. Le prédicat cut est défini de la façon suivante :

$$\text{cut}(\pi,\alpha,\beta,x,\approx) \triangleq \begin{cases} 1 \text{ si} & \alpha \uparrow \beta \approx \alpha \uparrow [\beta \downarrow (x \uparrow h_1(\pi))] \text{ et } \lambda(\pi) = \text{Player} \\ 1 \text{ si} & \beta \downarrow \alpha \approx \beta \downarrow [\alpha \uparrow (x \downarrow h_2(\pi))] \text{ et } \lambda(\pi) = \text{Opponent} \\ 0 \text{ sinon} \end{cases}$$

L'axiome de coupure peut donc être spécialisé en considérant que la position est du joueur :

$$\frac{\alpha \uparrow \beta \approx \alpha \uparrow [\beta \downarrow (x \uparrow h_1(\pi))] \qquad \alpha, \beta \in D \oplus \{\iota\} \qquad x, y \in E_\pi \oplus \{\iota\}}{\alpha \uparrow (\beta \downarrow x) \;\approx\; \alpha \uparrow [\beta \downarrow (x \uparrow y)]}$$

Supposons que les hypothèses de la règle soient vérifiées dans le jeu CLP(χ). La composante histoire de $\alpha \uparrow \beta$ est alors égale à celle de $\alpha \uparrow [\beta \downarrow (x \uparrow h_1(\pi))]$, donc l'histoire de x ne peut être plus courte que celle de $\alpha \uparrow \beta$. Donc l'histoire de $\alpha \uparrow (\beta \downarrow x)$ est celle de $\alpha \uparrow \beta$. De même, puisque $x, y \in E_\pi \oplus \{\iota\}$, soit $y = \iota$ et l'énoncé est trivialement vrai, soit l'histoire de y ne peut être plus courte que celle de $h_1(\pi)$. Donc $\alpha \uparrow [\beta \downarrow (x \uparrow y)]$ a, lui aussi, la même composante histoire de $\alpha \uparrow \beta$ et de $\alpha \uparrow (\beta \downarrow x)$, que nous appellerons h. Une fois fixée la composante histoire h, vérifier l'équivalence des valeurs signifie vérifier que les composantes logiques soient équivalentes, après la clôture existentielle des variables $\bar{h}$ n'aparaissant pas dans h. Il reste donc à prouver que $\exists(\bar{h}).\, \alpha_1 \vee (\beta_1 \wedge x_1)$ est logiquement équivalent à $\exists(\bar{h}).\, \alpha_1 \vee [\beta_1 \wedge (x_1 \vee y_1)]$, où par la notation ν_1 on indique la première projection, donc la composante logique ρ, d'un couple $\nu = (\rho, h)$. D'une part, nous avons :

$$\exists(\bar{h}).\, \alpha_1 \vee [\beta_1 \wedge (x_1 \vee y_1)] \;\subseteq\; \exists(\bar{h}).\, \alpha_1 \vee [\beta_1 \wedge \text{true}] \;\equiv\; \exists(\bar{h}).\, \alpha_1 \vee \beta_1$$

D'autre part, puisque l'heuristique est pessimiste et par l'hypothèse de l'axiome (A789p), nous avons :

$$\exists(\bar{h}).\, \alpha_1 \vee [\beta_1 \wedge (x_1 \vee y_1)] \;\supseteq\; \exists(\bar{h}).\, \alpha_1 \vee [\beta_1 \wedge (x_1 \vee (h_1(\pi))_1)] \;\equiv\; \exists(\bar{h}).\, \alpha_1 \vee \beta_1$$

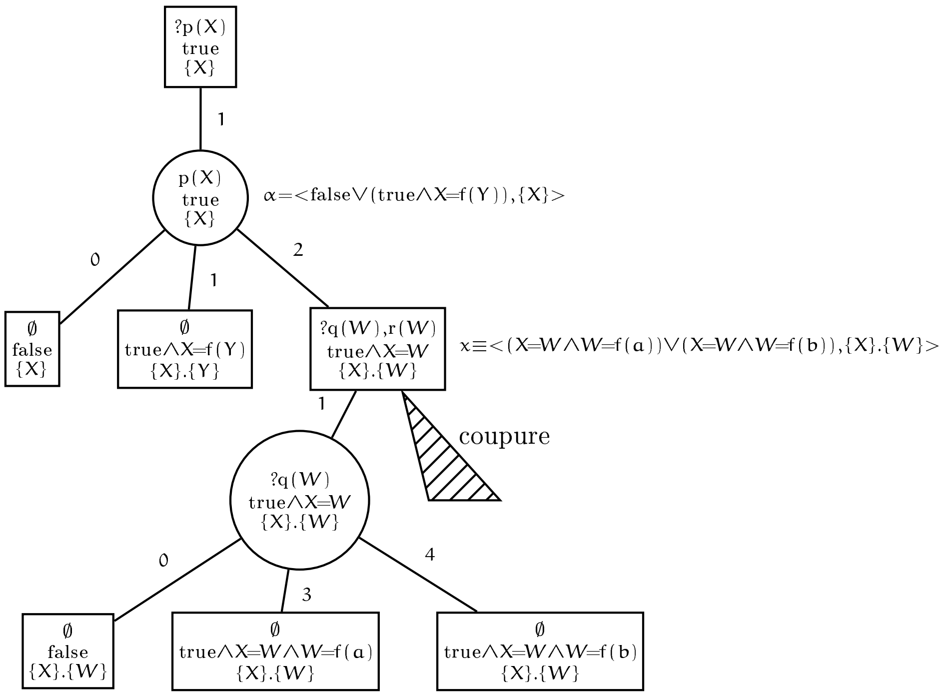
Les composantes logiques de $\alpha \uparrow [\beta \downarrow (x \uparrow y)]$, de $\alpha \uparrow (\beta \downarrow x)$ et de $\alpha \uparrow \beta$, sont donc équivalentes.

8.4.4. Propriété distributive. Nous avons prouvé précédemment que les opérations des joueurs sont distributives par rapport à l'équivalence forte des valeurs (cf. lemme 8.2.7), donc elles le sont forcément par rapport à toute autre relation, telle que l'équivalence faible $\approx$, contenant l'équivalence forte :

$$\begin{cases} x \downarrow (y \uparrow z) & \approx & (x \downarrow y) \uparrow (x \downarrow z) \\ x \uparrow (y \downarrow z) & \approx & (x \uparrow y) \downarrow (x \uparrow z) \end{cases}$$

8.4.5. Exemples. Nous allons montrer trois exemples de coupures effectuées par la méthode Alpha-Bêta appliquée à la résolution d'un but dans un programme logique. Les deux premiers seront des exemples de coupures peu profonde. Malgré leur simplicité, ils illustrerons la dualité des coupures, c'est-à-dire la possibilité, dans le jeu CLP(χ), de simplifier la résolution d'un but parfois dans les noeuds du joueur et parfois dans ceux de l'opposant.

FIG. 8.4.1. Exemple de coupure peu profonde chez l'opposant



8.4.5.1. *Coupures chez l'opposant.* Considérons le programme logique composé de cinq clauses :

```
1. p(f(Y)).
2. p(W) :- q(W),r(W).
3. q(f(a)).
4. q(f(b)).
5. r(Z) :-...
```

Le corps de la clause n°5 n'est pas important si l'on souhaite résoudre le but `p(X)`. En effet, supposons d'utiliser la règle n°1 pour obtenir une première réponse $X = f(Y)$. Ensuite nous utiliserons la règle n°2 en souhaitant ajouter de nouvelles réponses à celle déjà obtenue. Mais l'application de cette règle conduit à la résolution du sous-but `q(W)`, qui, lui, amène, par les règles n°3 et n°4, aux réponses $W = f(a)$ et $W = f(b)$. Ainsi, à ce stade de la résolution, nous pouvons prévoir que la règle n°2 ne portera rien qui pourra être mieux de la réponse $X = f(Y)$ déjà obtenue par la première règle du programme. On peut donc éviter de résoudre le sous-but `r(W)`. En termes de jeu, nous pouvons dire, dans un certain sens, que l'opposant s'aperçoit d'en avoir déjà trop fait en obligeant le joueur au payoff $(X = f(a)) \vee (X = f(b))$. Pour l'opposant il est inutile d'en vouloir plus, cette partie de l'arbre sera ignorée par son adversaire. En d'autres termes, nous allons voir qu'il s'agit d'une coupure (peu profonde) réalisé par l'algorithme Alpha-Bêta. On peut observer, dans la figure 8.4.1, qu'au troisième niveau de l'arbre, en réponse au coup n°2 du joueur, nous pouvons calculer une valeur inachevée, pour l'opposant, égale à :

$$x =< \text{false} \vee (X = W \wedge W = f(a)) \vee (X = W \wedge W = f(b)), \{X\}.\{W\} >$$

Auparavant, au deuxième niveau de l'arbre, nous avions calculé une valeur inachevée pour le joueur égale à :

$$\alpha = < \text{false} \vee (X = f(Y)), \{X\} >$$

Puisque $\beta = \iota$ est indéfini, la condition de coupure $\beta \downarrow \alpha \approx \beta \downarrow [\alpha \uparrow (x \downarrow h_2(\pi))]$ devient $\alpha \approx (\alpha \uparrow (x \downarrow h_2(\pi)))$. Or, les deux membres ont la composante histoire égale à $\{X\}$. Pour que la condition soit vérifié il faut alors que :

$$\exists Y.\ \text{false} \vee (X = f(Y)) \quad \equiv \quad \exists W.\ \text{false} \vee (X = W \wedge W = f(a)) \vee (X = W \wedge W = f(b))$$

ce qui est bien le cas, donc la coupure peut avoir lieu.

8.4.5.2. *Coupures chez le joueur.* Considérons le programme logique composé des clauses suivantes :

```
1. p(X,Y,Z) :- q(X,Y),r(X,Z).
2. q(a,b).
3. r(a,W).
4. r(X,Z) :-...
```

Dans ce cas, pour résoudre le but `p(X,Y,Z)`, nous utiliserons la règle n°1. Si nous essayons de résoudre le premier sous-but de cette règle, nous serons contraints à unifier la variable `X` au terme `a`, et la variable `Y` au terme `b`. Des trois variables initiales `X`,`Y` et `Z`, seulement la dernière demeure libre à ce stade. Or, en appliquant la règle n°3 pour résoudre le second sous-but de la n°1, nous obtiendrons tout ce que nous aurions pu souhaiter, c'est-à-dire une réponse qui laisse encore libre la variable `Z`. Il est donc inutile d'utiliser la règle n°4. En termes de jeu, c'est le joueur, cette fois, qui comprend l'inutilité de préciser la valeur inachevée de la position analysée, dans le contexte où cette position se trouve. L'arbre de la position $< ?p(X, Y, Z), \text{true}, \{X, Y, Z\} >$ est représenté dans la figure 8.4.2, où la contrainte true (de toute façon non significative) est souvent omise dans la composante formule des positions pour simplifier la représentation de l'arbre. À l'adresse 1.1 de cet arbre, l'opposant peut calculer une valeur inachevée x' égale à :

$$x' = < \text{false} \vee (\text{true} \wedge X = X_1 \wedge Y = Y_1 \wedge Z = Z_1 \wedge X_1 = a \wedge Y_1 = b), \{X, Y, Z\}.\{X_1, Y_1, Z_1\} >$$

qui devient, pour l'évaluation du fils à l'adresse 2, le paramètre β équivalent à :

$$\beta \equiv < (X = X_1 \wedge Y = Y_1 \wedge Z = Z_1 \wedge X_1 = a \wedge Y_1 = b), \{X, Y, Z\}.\{X_1, Y_1, Z_1\} >$$

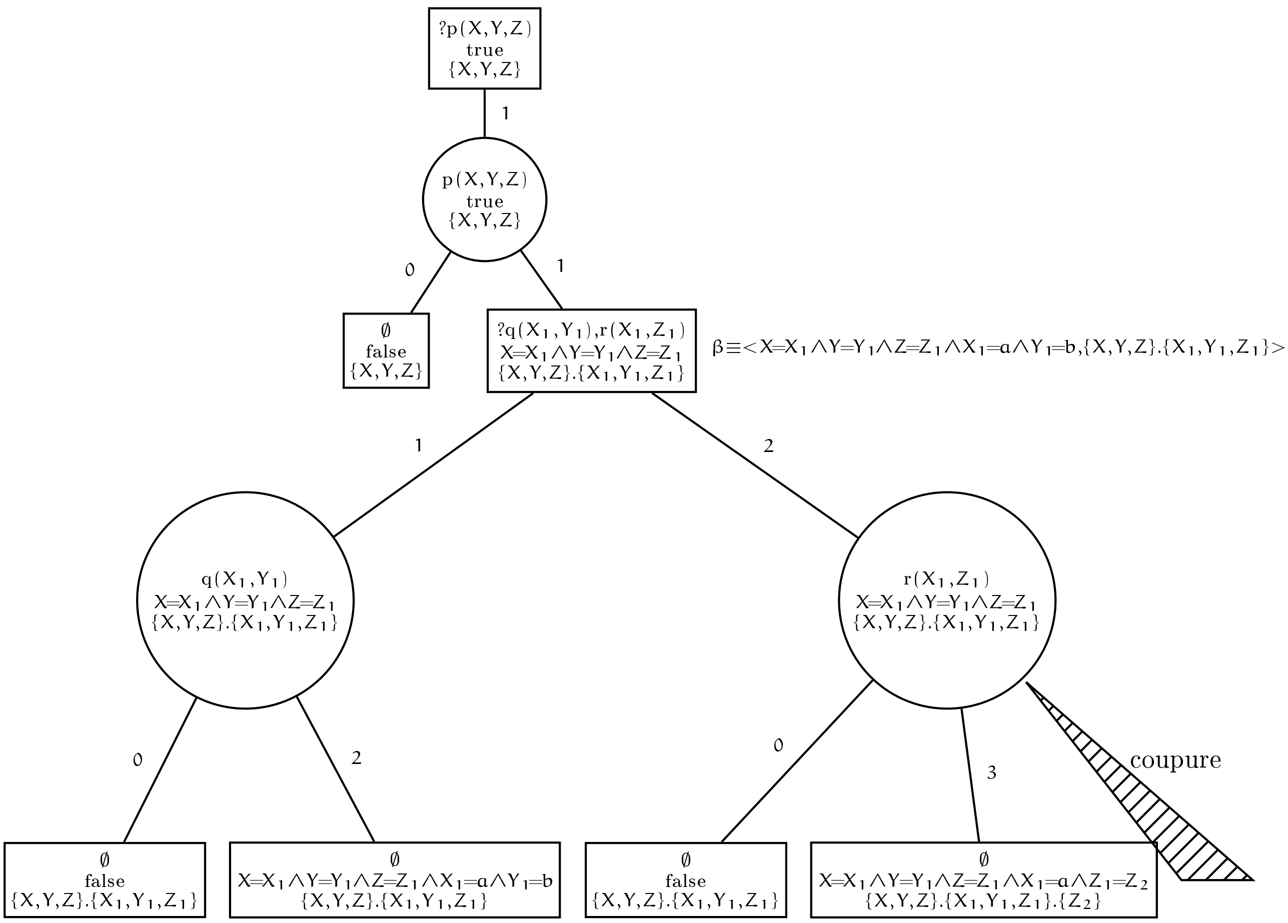
Or, après l'évaluation des coups 0 et 3, la valeur inachevée x du joueur à la position π de l'adresse 1.1.2 est égale à :

$$x = < \text{false} \vee (\text{true} \wedge X = X_1 \wedge Y = Y_1 \wedge Z = Z_1 \wedge X_1 = a \wedge Z_1 = Z_2), \{X, Y, Z\}.\{X_1, Y_1, Z_1\}.\{Z_2\} >$$

La condition de coupure est alors vérifiée, puisque $\alpha \uparrow \beta \approx \alpha \uparrow (\beta \downarrow (x \uparrow h_2(\pi)))$ signifie (étant donné que $\alpha = \iota$) $\beta \approx \beta \downarrow (x \uparrow h_2(\pi))$. Les deux membres ont la composante histoire égale à $\{X, Y, Z\}.\{X_1, Y_1, Z_1\}$, qui est l'histoire de β. Donc l'équivalence faible revient à la condition d'équivalence logique :

$$\begin{gathered}(X = X_1 \wedge Y = Y_1 \wedge Z = Z_1 \wedge X_1 = a \wedge Y_1 = b) \\ \equiv \\ \exists Z_2.\ X = X_1 \wedge Y = Y_1 \wedge Z = Z_1 \wedge X_1 = a \wedge Y_1 = b \wedge Z_1 = Z_2\end{gathered}$$

FIG. 8.4.2. Exemple coupure peu profonde chez le joueur



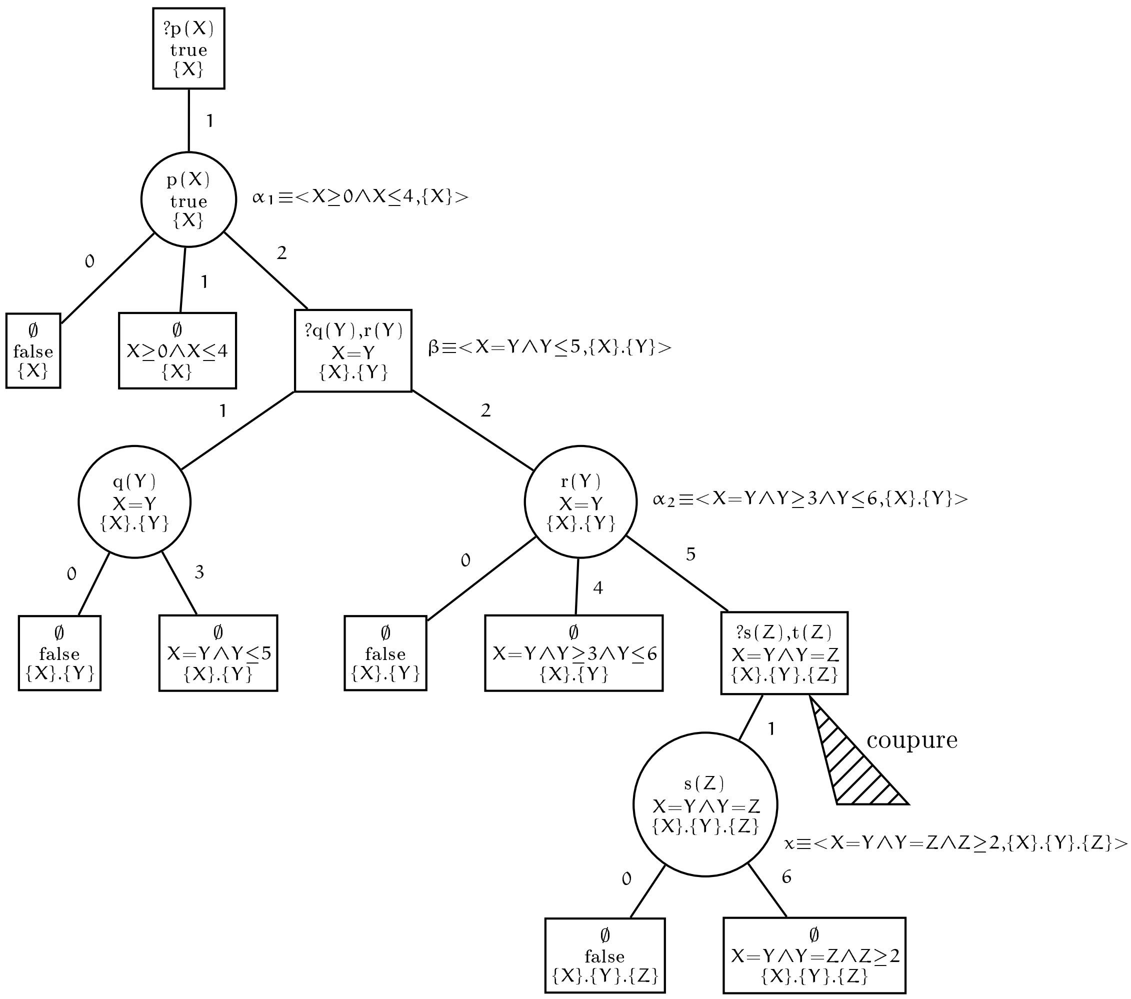
La coupure est donc autorisée.

8.4.5.3. *Coupures profondes.* Les coupures illustrées dans les exemples précédents, sont des coupures peu profondes, qui ne mettent pas en cause une propriété importante, la propriété *distributive*, de la conjonction et disjonction de formules logiques. Considérons le programme composé des clauses suivantes :

```
1. p(X) :- X>=0,X<=4.            5. r(X) :- s(X),t(X).
2. p(X) :- q(X),r(X).            6. s(X) :- X>=2.
3. q(X) :- X<=5.                 7. t(X) :- ...
4. r(X) :- X>=3,X<=6.
```

Pour le but `p(X)`, on s'aperçoit que la règle n°1 du programme donne un résultat, l'intervalle $[0,4]$, qui peut être partiellement amélioré par le résultat de l'application de la règle n°2. D'une part, la résolution du prédicat `q` contraint la variable `X` à l'intervalle $[-\infty,5]$. D'autre part, la résolution du prédicat `r`, amène à une première solution $[3,6]$ qui étend les solutions du but initial `p(X)` à l'intervalle $[0,5]$. Toute autre amélioration est impossible. En effet, la règle n°5 contraint la variable `X` à être supérieure à 2, tout en étant aussi inférieure à 5. Il est donc inutile de rechercher les solutions fournies par la règle n°7, puisque nous ne pourrons pas ajouter d'autres

FIG. 8.4.3. Exemple de coupure profonde



solutions outre celles, déjà obtenues, de l'intervalle $[0,5]$. Dans la figure 8.4.3 on peut suivre l'évolution du calcul conduisant à ce type de simplification. À l'adresse 1.2.2.5, la position de l'opposant est évaluée avec un paramètre α qui est le $\uparrow$ des valeurs inachevées des positions antécédentes du joueur, c'est-à-dire $\alpha = \alpha_1 \uparrow \alpha_2$, où α_1 est la valeur inachevée calculée à l'adresse 1 et α_2 est la valeur inachevée de l'adresse 1.2.2. Donc :

$$\alpha \ \equiv < (X \geq 0 \wedge X \leq 4) \vee (X = Y \wedge Y \geq 3 \wedge Y \leq 6), \{X\} >$$

Le paramètre β utilisé pour l'évaluation de la position π à l'adresse 1.2.2.5 est la valeur inachevée de la position 1.2, c'est-à-dire :

$$\beta \ \equiv < (X = Y \wedge Y \leq 5), \{X\}.\{Y\} >$$

Après l'évaluation du premier coup (le sous-but `s(Z)`), la valeur inachevée de π est :

$$x \ \equiv < (X = Y \wedge Y = Z \wedge Z \geq 2), \{X\}.\{Y\}.\{Z\} >$$

Ainsi, d'une part, nous avons :

$$\beta \downarrow \alpha \ \equiv < (X{=}Y \wedge Y{\leq}5 \wedge X{\geq}0 \wedge X{\leq}4) \vee (X{=}Y \wedge Y{\leq}5 \wedge Y{\geq}3), \{X\} >$$

et, d'autre part, nous avons :

$$\begin{aligned}\beta \downarrow (\alpha \uparrow (x \downarrow h_2(\pi))) \quad &\equiv \quad \beta \downarrow < (X{\geq}0 \wedge X{\leq}4) \vee (X{=}Y \wedge Y{\geq}3 \wedge Y{\leq}6) \\ &\qquad \vee (X{=}Y \wedge Y{=}Z \wedge Z{\geq}2), \{X\} > \\ &\equiv \quad < (X{\geq}0 \wedge X{\leq}4 \wedge X{=}Y) \vee (X{=}Y \wedge Y{\geq}3 \wedge Y{\leq}5) \\ &\qquad \vee (X{=}Y \wedge Y{=}Z \wedge Z{\geq}2 \wedge Y{\leq}5), \{X\} >\end{aligned}$$

Pour vérifier l'équivalence faible des deux membres il faut comparer les clôtures existentielles des variables n'apparaissant pas dans l'histoire $\{X\}$. Il faut donc comparer la formule de $\beta \downarrow \alpha$ close par rapport à Y :

$$\begin{gathered}\exists Y.\ (X{=}Y \wedge Y{\leq}5 \wedge X{\geq}0 \wedge X{\leq}4) \vee (X{=}Y \wedge Y{\leq}5 \wedge Y{\geq}3) \\ \equiv \\ \exists Y.\ (X{=}Y \wedge X{\geq}0 \wedge X{\leq}4) \vee (X{=}Y \wedge X{\leq}5 \wedge X{\geq}3) \\ \equiv \\ X{\geq}0 \wedge X{\leq}5\end{gathered}$$

avec la formule de $\beta \downarrow (\alpha \uparrow (x \downarrow h_2(\pi)))$ close par rapport à Y et Z :

$$\begin{gathered}\exists Y.\exists Z.\ (X{\geq}0{\wedge}X{\leq}4{\wedge}X{=}Y){\vee}(X{=}Y{\wedge}Y{\geq}3{\wedge}Y{\leq}5){\vee}(X{=}Y{\wedge}Y{=}Z{\wedge}Z{\geq}2{\wedge}Y{\leq}5) \\ \equiv \\ \exists Y.\exists Z.\ (X{\geq}0{\wedge}X{\leq}4{\wedge}X{=}Y){\vee}(X{=}Y{\wedge}X{\geq}3{\wedge}X{\leq}5){\vee}(X{=}Y{\wedge}Y{=}Z{\wedge}X{\geq}2{\wedge}X{\leq}5) \\ \equiv \\ (X{\geq}0 \wedge X{\leq}4) \vee (X{\geq}3 \wedge X{\leq}5) \vee (X{\geq}2 \wedge X{\leq}5) \\ \equiv \\ X{\geq}0 \wedge X{\leq}5\end{gathered}$$

Il s'agit bien de formules logiquement équivalentes (par la propriété distributive des connecteurs logiques), ce qui justifie la coupure.

8.4.6. Conclusions. L'interprétation en termes de jeu de la programmation logique permet donc d'introduire des simplifications du calcul inédites dans le problème de la résolution d'un but logique. Bien entendu, cette approche permet d'envisager des interpréteurs ou des compilateurs plus efficaces que les outils actuels dans la recherche de l'ensemble des réponses (plus générales) d'un but. Il faut cependant rendre effectif et efficace le test de coupure, c'est-à-dire le contrôle de l'équivalence logique de formules closes de façon existentielle. Il faudra donc demander au solveur de contraintes, ou à un algorithme auxiliaire, de savoir répondre à ce genre de question, et de savoir y répondre de façon efficace. En ce qui concerne l'efficacité, il faut toutefois remarquer que les coupures réalisées éliminent du champ de recherche aussi bien des parties finies que des parties infinies de l'arbre de jeu. Il se peut donc que le coût du test de coupure soit récompensé par l'élagage d'une importante branche de l'arbre de jeu et même d'une branche infinie. Il se peut, en particulier, que l'algorithme Alpha-Bêta termine en temps fini là où les interpréteurs traditionnels pourraient boucler à l'infini. Il s'agit bien d'un avantage indiscutable et flagrant de l'algorithme Alpha-Bêta vis-à-vis des moteurs classiques basés sur la résolution SLD ou CSLD.

Conclusions

Si la théorie des jeux a développé, à ses débuts, une vocation pour les sciences sociales et économiques, elle apparaît aujourd'hui comme un paradigme de concepts et de techniques très générales, dont le potentiel reste encore à exploiter en informatique. Dans cette thèse nous avons étudié une branche particulière, la théorie des jeux combinatoires (à deux joueurs), pour en tirer bénéfice dans le domaine, très actif, des sémantiques formelles des langages de programmation. Les concepts mis en oeuvre par cette théorie ne dépendent pas de la simplicité des structures algébriques utilisées traditionnellement, comme les booléens (*gagné*, *perdu*), éventuellement enrichis par la valeur auxiliaire *match nul*, ou comme les entiers naturels, modelisant, par exemple, un gain en argent, ou comme les entiers relatifs, modelant le coût positif ou négatif à l'issue du match.

La première contribution de cette thèse a donc été l'identification précise des propriétés fines du domaine d'évaluation nécessaires à l'application des notions traditionnelles, telles que la *valeur* d'une position, d'un jeu ou d'une stratégie. De façon surprenante, nous avons pu montrer que la structure d'ordre total des domaines habituellement utilisés, n'est pas essentielle. Au contraire, on peut relâcher de façon spectaculaire ce type d'hypothèse et demander que la relation utilisée soit un simple préordre, et qu'elle le soit localement pour toute position du jeu et non pas nécessairement par rapport à toutes les valeurs du jeu. De même, parmi toutes les caracteristiques spécifiques aux fonctions habituelles, le *max* et le *min*, nous avons identifié la *monotonie* comme l'unique propriété cruciale. Cela permet, à condition de respecter les peu nombreuses et faibles hypothèses, de donner une sémantique "des jeux", en utilisant une structure des valeurs arbitrairement complexe.

La généralisation opérée capture aussi le cas des jeux infinis, pour lesquels il n'y aura pas une seule façon d'interpréter l'issue du match, mais plutôt deux façons, et donc deux sémantiques, une *pessimiste*, et une *optimiste*, cernant, par un intervalle, le sens du jeu à l'infini. Cela donne une manière de caractériser par l'intermédiaire des jeux certains problèmes de l'informatique théorique, où la notion d'activité infinie est toujours présente et naturelle. Par exemple, la notion de bisimulation entre processus concurrents, définie comme le plus grand point fixe d'une opération sur les relations entre processus, peut être reformulée en termes de jeu. La réponse booléenne intéressante, *bisimilaires* ou *pas bisimilaires*, sera fournie par la sémantique optimiste du jeu. Cela ne tient pas au hasard, l'approche d'une définition de plus grand point fixe étant, dans un certain sens, celle de démarrer l'évaluation, dans l'optimisme le plus total, avec toutes les valeurs possibles, et de les éliminer ensuite, au fur et à mesure, par l'application itérée de l'opération concernée.

Pour mettre notre contribution à la disposition d'un public le plus large possible, nous avons délimité et regroupé les hypothèses nécessaires à l'existence d'une sémantique des jeux infinis. Par la simple vérification de ces conditions, l'utilisateur de notre cadre théorique aura pour garantie l'existence d'une sémantique pour les matchs infinis, et sera cependant affranchi des notions sous-jacentes, propres à la théorie des domaines et résumées dans le deuxième chapitre, telles que la notion de point compact, de complétion par idéaux ou de complétion par filtres.

Une fois définie la sémantique des jeux (finis ou infinis), on peut se poser la question de savoir si les algorithmes qui calculent les valeurs sémantiques peuvent être, eux aussi, adaptés et généralisés. Certains, comme *SSS**, semblent d'adaptation difficile (la notion de stratégie n'étant pas significative dans tous les jeux), mais d'autres, comme *solve* ou *Alpha-Bêta*, peuvent être généralisés à un cadre bien plus large que celui de la structure des entiers relatifs avec le max et le min. Dans cette thèse nous présentons une preuve de correction de l'algorithme Alpha-Bêta pour des structures répondant à un critère très général (structure $\alpha\beta$-normale), incluant, comme cas particulier, les treillis distributifs. L'algorithme est présenté de façon originale à la manière des sémantiques opérationnelles *naturelles* (*à grand pas* ou *à la Kahn*), ce qui facilite la rédaction d'une preuve formelle de correction. Nous avons aussi défini les bases théoriques pour l'application d'un variante de l'algorithme, appelée *Alpha-Bêta-Gamma-Delta*, qui s'applique à des structures où les fonctions des joueurs ne sont pas distributives. Cela paraît utile, entre autres, lorsqu'on désire calculer une approximation de la valeur d'un jeu par des opérations abstraites. En effet, dans un tel procédé d'abstraction, la conservation de la propriété distributive des opérations serait incertaine.

La programmation logique est l'application principale de toutes les notions introduites. Le domaine des valeurs est relativement complexe et avec peu de propriétés. Malgré cela, nous pouvons appliquer la théorie développée, d'une part pour donner une sémantique des jeux aux programmes logiques (d'un schéma CLP(χ) quelconque, donc y compris LP), et d'autre part pour calculer la sémantique, c'est-à-dire pour résoudre les buts logiques, au moyen de l'algorithme Alpha-Bêta. Nous présentons la preuve d'adéquation de la sémantique (pessimiste) des jeux par rapport à la sémantique traditionnelle de CLP, et la preuve que les hypothèses d'application de l'algorithme Alpha-Bêta polymorphe sont vérifiées. Si les simplifications du calcul réalisées par l'algorithme sont d'un genre nouveau en programmation logique, il s'agit en réalité des mêmes simplifications qui permettent aux ordinateurs d'être aussi compétitifs que les humains dans les jeux classiques tels que *Othello* ou les *échecs*.

Plusieurs sujets méritent d'être approfondis. Il serait par exemple intéressant de définir, pour l'algorithme Alpha-Bêta, une formulation *à petits pas* (*small steps*), dans le style des sémantiques opérationnelles structurées (SOS), afin de révéler de façon explicite le parallélisme potentiel dans l'exécution de l'algorithme. Concernant la variante Alpha-Bêta-Gamma-Delta, on peut bien entendu demander une formulation dans le style à grand ou à petit pas, et une preuve de correction aussi élégante que celle réservée à son précurseur. On peut aussi améliorer l'étude et la caractérisation des jeux infinis mais rationnels (avec un nombre fini de sous-arbres de jeu distincts), comme le sont, par exemple, le jeu de la bisimulation lorsque les processus sont à états finis, ou le jeu CLP(χ) pour certains programmes logiques. On peut envisager l'utilisation d'Alpha-Bêta, ou de la variante Alpha-Bêta-Gamma-Delta si nécessaire, pour des interprétations abstraites de programmes logiques. Enfin, par la dualité inhérente aux jeux (joueur contre opposant, pessimisme contre optimisme), nous avons le sentiment de pouvoir approcher de façon très intuitive la *négation*, un sujet complexe en programmation logique, par une sémantique adéquate des jeux combinatoires. Il s'agit là, bien entendu, d'une intuition plus que d'une certitude, mais la négation en programmation logique paraît être, très simplement, un jeu où les joueurs ont, dans un certain sens, le droit de retourner le damier et de continuer le jeu avec les couleurs de leur adversaire, faisant ainsi deux coup consécutifs. Le joueur qui tentait auparavant d'appliquer les règles du programme, se retrouve alors à sélectionner les sous-buts pour empêcher son adversaire de les démontrer, tandis que ce dernier tente désormais de résoudre les buts par l'application des clauses du programme.

Quatrième partie

Appendice

Code source ocaml

simulation.ml

Ce programme `ocaml` permet l'ensemble des tests de correction et de performance des différentes méthodes d'évaluation d'un arbre de jeu, traitées au cours du chapitre 6.

```
(*
  Simulation de l'évaluation d'arbres de jeu par les méthodes :

                    1) exhaustive
                    2) minimax
                    3) Alpha-Beta
                    4) Alpha-Beta-Gamma-Delta (dans ses deux versions)
 *)

(* ------------------------------------------------------------
    Définition des structures d'évaluation
   ------------------------------------------------------------*)

(* Définition du type des structures d'évaluation *)
type 'a structure = { sup  : 'a * 'a -> 'a ;
                      inf  : 'a * 'a -> 'a ;
                      ord  : 'a * 'a -> bool;
                      rnd  : unit -> 'a    ;
                      bot  : unit -> 'a    ;
                      top  : unit -> 'a    ;
                      echo : 'a -> unit    ;
                      name : string;           (* Nom de la structure *)
                      show : unit -> unit      (* Presentation de la structure *)
                    };;

(* Initialisation du générateur de nombres aléatoires *)
Random.init (int_of_float(Sys.time()));;

(* Premier modèle (mod1) : structure simple non distributive *)
type mod1_domain = Top | Bot | A | B | C | D | E;;

let mod1 = { sup = (fun c -> (match c with
                                (Top,_)  -> Top
                              | (_,Top)  -> Top
                              | (Bot,y)  -> y
                              | (x,Bot)  -> x
```

```
                           | (D,y) -> y
                           | (x,D) -> x
                           | (E,y) -> E
                           | (x,E) -> E
                           | (x,y)   -> if (x=y) then x else E
                           ));
  inf = (fun c -> (match c with
                             (Top,y) -> y
                           | (x,Top) -> x
                           | (Bot,_) -> Bot
                           | (_,Bot) -> Bot
                           | (D,y) -> D
                           | (x,D) -> D
                           | (E,y) -> y
                           | (x,E) -> x
                           | (x,y)   -> if (x=y) then x else D
                           ));
  ord = (fun c -> (match c with
                           | (_,Top) -> true
                           | (Bot,_) -> true
                           | (D,x)   -> x != Bot
                           | (x,E)   -> x != Top
                           | (_,_)   -> false
                           ));
  rnd = (fun (_ :unit) -> (match (Random.int 7) with
                           | 0 -> Top
                           | 1 -> Bot
                           | 2 -> A
                           | 3 -> B
                           | 4 -> C
                           | 5 -> D
                           | 6 -> E
                           ));
  bot = (fun (_ :unit) -> Bot);
  top = (fun (_ :unit) -> Top);
  echo = (fun x -> (match x with
                             Top -> print_string "Top"
                           | Bot -> print_string "Bot"
                           | A   -> print_string "A"
                           | B   -> print_string "B"
                           | C   -> print_string "C"
                           | D   -> print_string "D"
                           | E   -> print_string "E"
                           ))  ;
  name = "Simple modele non distributif";
  show = (fun (_ :unit) -> print_string ("\n\tTop\n\t|\n\tE\n"
                                ^"    /\r\t| \ \n  A\r\tB C\n    \ \r\t| / \n"
                                ^"\tD\n\t|\n\tBot\n\n") );
};;
```

```
(* Deuxième modèle (mod2) : structure distributive des entiers entre 0 e 99 *)
let mod2 = { sup = (fun (x,y) -> if x<=y then y else x );
             inf = (fun (x,y)  -> if x<=y then x else y );
             ord = (fun (x,y) -> (x<=y) )               ;
             rnd = (fun (_ :unit) -> (Random.int 100) )  ;
             bot = (fun (_ :unit) -> 0 )                ;
             top = (fun (_ :unit) -> 99 )               ;
             echo = print_int                           ;
             name = "Modèle distributif des entiers entre 0 et 99";
             show = (fun (_ :unit) -> print_string ("\nDomaine D=[0. .99] avec min et max\n\n") );
          };;


(* Troisième modèle (mod3) : structure non distributive des intervalles, avec l'union et l'intersection *)
type mod3_domain = Vide | Int of int * int;;

let mod3 = { sup = (fun c -> (match c with
                         | (Vide,y) -> y
                         | (x,Vide) -> x
                         | (Int(x,y),Int(x',y')) -> Int( mod2.inf(x,x'), mod2.sup(y,y') )));

             inf = (fun c -> (match c with
                         | (Vide,y) -> Vide
                         | (x,Vide) -> Vide
                         | (Int(x,y),Int(x',y')) -> let (u,v) = (mod2.sup(x,x'), mod2.inf(y,y')) in
                                                    if v<u then Vide else Int(u,v) ));

             ord = (fun c -> (match c with
                         | (Vide,_) -> true
                         | (Int(x,y),Int(x',y')) -> (x>=x') && (y<=y')
                         | (_,_)    -> false ));

             rnd = (fun (_ :unit) -> let x = (Random.int 100) in Int(x,x+(Random.int (100-x))))  ;

             bot = (fun (_ :unit) -> Vide);
             top = (fun (_ :unit) -> Int(0,99));
            echo = (fun i -> (match i with
                         | Vide     -> print_string "[]"
                         | Int(x,y) -> print_string ("["^string_of_int(x)^","^string_of_int(y)^"]") ));

            name = "Modèle non distributif des intervalles";
            show = (fun (_ :unit) ->
                          print_string (
                           "\nIntervalles sur [0. .99] avec\n sup = plus petit intervalle"^
                           " contenant l'union ensembliste\n inf = intersection ensembliste\n\n"
                          ));
          };;
```

```
(* Définition du type arbre (de jeu) *)

type 'a arbre = PN of ('a arbre list) (* Noeud non terminal du joueur (Player Node) *)
              | ON of ('a arbre list) (* Noeud non terminal de l'opposant (Opponent Node) *)
              | PL of 'a              (* Noeud terminal du joueur    (Player Leaf) *)
              | OL of 'a;;            (* Noeud non terminal de l'opposant (Opponent Node) *)


(* ————————————————————————
   Fabrication d'arbre de jeu aléatoires
   ———————————————————————— *)

(* Alias *)
type profondeur = int;;
type largeur  = int;;

(* Fabrique une liste de n éléments identiques α x *)
let rec mklist n x = if n=0 then [] else [x]@(mklist (n-1) x);;

(* (1) Fabrication d'arbres aleatoires strictement alternés de largeur entre 0 et 5 (alea) *)

let rec

  aleaP = fun (s :'a structure) (max_p :profondeur) (p :profondeur) ->
          if (max_p = p) then (PL(s.rnd()))
                         else
                          (* Au début au maximum 3 fils, puis au maximum 5 fils *)
                          let auplus = 1+(if ((max_p - p) < 3) then 5 else 3) in
                          (* Au début au minimum 2 fils, puis au minimum 0 fils *)
                          let aumoins = (if (p < 2) then 2 else 0) in
                          let n = (max aumoins (Random.int auplus))
                          in if n>0 then (PN(List.map (aleaO s max_p) (mklist n (p+1))))
                                    else (PL(s.rnd()))
 and

  aleaO = fun (s :'a structure) (max_p :profondeur) (p :profondeur) ->
          if (max_p = p) then (OL(s.rnd()))
                         else
                          (* Au début au maximum 3 fils, puis au maximum 5 fils *)
                          let auplus = 1+(if ((max_p - p) < 3) then 5 else 3) in
                          (* Au début au minimum 2 fils, puis au minimum 0 fils *)
                          let aumoins = (if (p < 2) then 2 else 0) in
                          let n = (max aumoins (Random.int auplus))
                          in if n>0 then (ON(List.map (aleaP s max_p) (mklist n (p+1))))
                                    else (OL(s.rnd()))
;;

(* fonction principale *)
let alea = fun (s :'a structure) (max_p :profondeur) ->
        Random.self_init();
        if Random.int 2 = 1 then (aleaP s max_p 0) else (aleaO s max_p 0);;
```

```
(* (2) Fabrication d'arbres ternaires (toujours 3 fils) aleatoires strictement alternés (alea3) *)
let rec

    alea3P_ = (fun (s :'a structure) (max_p :profondeur) (p :profondeur) ->
              if p=max_p then (PL(s.rnd()))
                      else (PN([OL(s.rnd()) ; (alea3O_ s max_p (p+1)) ; OL(s.rnd()) ] )))

 and

    alea3O_ = (fun (s :'a structure) (max_p :profondeur) (p :profondeur) ->
              if p=max_p then (OL(s.rnd()))
                      else (ON([PL(s.rnd()) ; (alea3P_ s max_p (p+1)) ; PL(s.rnd()) ] )))

;;
let alea3P = (fun s max_p -> alea3P_ s max_p 0)
and alea3O = (fun s max_p -> alea3O_ s max_p 0);;

(* fonction principale *)
let alea3 = fun s max_p -> Random.self_init();
  if Random.int 2 = 1 then (alea3P s max_p) else (alea3O s max_p)
;;

(* (3) Fabrication d'arbres binaires (toujours 2 fils) aleatoires strictement alternés (alea2) *)
let rec

    alea2P_ = (fun (s :'a structure) (max_p :profondeur) (p :profondeur) ->
              if p=max_p then (PL(s.rnd()))
                      else (PN([OL(s.rnd()) ; (alea2O_ s max_p (p+1)) ] )))

 and

    alea2O_ = (fun (s :'a structure) (max_p :profondeur) (p :profondeur) ->
              if p=max_p then (OL(s.rnd()))
                      else (ON([PL(s.rnd()) ; (alea2P_ s max_p (p+1)) ] )))

;;
let alea2P = (fun s max_p -> alea2P_ s max_p 0)
and alea2O = (fun s max_p -> alea2O_ s max_p 0);;

let alea2 = fun s max_p -> Random.self_init();
 if Random.int 2 = 1 then (alea2P s max_p) else (alea2O s max_p)
;;


(* Renvoie la fonction de tirage aléatoire des arbres en fonction de la largeur désirée *)
let tirage_arbre = fun (l :largeur) -> match l with
 | 2  -> alea2
 | 3  -> alea3
 | _  -> alea
;;
```

```
(* Fonction profondeur *)
let rec maxoflist = fun l -> (match l with [] -> 0 | x : :l -> let m = (maxoflist l) in if x>m then x else m);;

let rec prof = fun (t :'a arbre) -> match t with
           PN(l) -> 1 + maxoflist (List.map prof l)
         | ON(l) -> 1 + maxoflist (List.map prof l)
         | PL(_) -> 0
         | OL(_) -> 0
;;


(* ——————————————————————
    Fonction d'évaluation exhaustive
     (tous les noeuds sont visités)
  —————————————————————— *)


let rec sup_of_list = fun (s :'a structure) l ->
   (match l with [] -> s.bot() | x : :l -> let m = (sup_of_list s l) in (s.sup (x,m)))


let rec inf_of_list = fun (s :'a structure) l ->
   (match l with [] -> s.top() | x : :l -> let m = (inf_of_list s l) in (s.inf (x,m)))


let rec sum_of_list = fun l ->
   (match l with [] -> 0 | x : :l -> let m = (sum_of_list l) in (x+m))


(* La fonction calcule au même temps le nombre de noeuds dans l'arbre; elle renvoie
  donc un couple (v,n), où v est la valeur et n le nombre de noeuds visités.

  Pour les tests de performance, les autres fonctions d'évaluation calculerons,
  elles aussi, le nombre de noeuds visités *)


let rec exhaustive = fun (s :'a structure) (t :'a arbre) -> match t with

         | PN(l) -> (match List.split(List.map (exhaustive s) l) with
                   |  (lv,ln) -> ((sup_of_list s lv),(sum_of_list ln)+1))

         | ON(l) -> (match List.split(List.map (exhaustive s) l) with
                   |  (lv,ln) -> ((inf_of_list s lv),(sum_of_list ln)+1))

         | PL(x) -> (x,1)
         | OL(x) -> (x,1)
;;
```

```
(* ——————————————————————
        Fonction d'évaluation minimax

       (seul le test d'aboutissement aux
        aux éléments maximum et minimum
                est réalisé)
   —————————————————————— *)

let rec minimax = fun (s :'a structure) (t :'a arbre) -> match t with
              | PN([t']) -> (match minimax s t' with (v,n) -> (v,n+1) )
              | PN(t' : :l) -> (match minimax s t' with
                                (v1,n1) ->
                              if (s.ord (s.top(),v1)) then (v1,n1+1) else
                                 (match (minimax s (PN l)) with
                                    (v2,n2) -> (s.sup(v1,v2),n1+n2)))

              | ON([t']) -> (match minimax s t' with (v,n) -> (v,n+1) )
              | ON(t' : :l) -> (match minimax s t' with
                                (v1,n1) ->
                              if (s.ord(v1,s.bot())) then (v1,n1+1) else
                                 (match (minimax s (ON l)) with
                                    (v2,n2) -> (s.inf(v1,v2),n1+n2)))

              | PL(x)     -> (x,1)
              | OL(x)     -> (x,1)
; ;

(* ——————————————————————
      Fonction d'évaluation Alpha-Beta
   —————————————————————— *)

let rec alpha_beta_ = fun (s :'a structure) (a :'a) (b :'a) (t :'a arbre) -> match t with
              | PN([t']) -> (match alpha_beta_ s a b t' with (v,n) -> (v,n+1) )
              | PN(t' : :l) -> (match alpha_beta_ s a b t' with
                                (v1,n1) -> let v = s.sup(a,v1) in
                              if (s.ord (b,v)) then (v,n1+1) else
                                 (match (alpha_beta_ s v b (PN l)) with
                                    (v2,n2) -> (s.sup(v,v2),n1+n2)))

              | ON([t']) -> (match alpha_beta_ s a b t' with (v,n) -> (v,n+1) )
              | ON(t' : :l) -> (match alpha_beta_ s a b t' with
                                (v1,n1) -> let v = s.inf(b,v1) in
                              if (s.ord(v,a)) then (v,n1+1) else
                                 (match (alpha_beta_ s a v (ON l)) with
                                    (v2,n2) -> (s.inf(v,v2),n1+n2)))

              | PL(x)     -> (x,1)
              | OL(x)     -> (x,1)
; ;

let alpha_beta = fun (s :'a structure) (t :'a arbre) -> alpha_beta_ s (s.bot()) (s.top()) t ; ;
```

```
(* ---------------------------------------------------------------
    Fonction d'évaluation Alpha-beta-gamma-delta
   --------------------------------------------------------------- *)

let rec cut_condition = fun (s :'a structure) (g :'a list) (d :'a list) ->
    match (g,d) with
      | ([],[])            -> false
      | (x::g',x'::d') -> if s.ord(x',x) then true else cut_condition s g' d'

;;

(* Pour tous les éléments g_i de la suite g, faire g_i <- g_i MAX (b MIN v) *)
let gamma_update = fun (s :'a structure) (g :'a list) (b :'a) (v :'a)
  -> List.map (fun x -> s.sup(x,s.inf(b,v))) g
;;

(* Pour tous les éléments d_i de la suite d, faire d_i <- d_i MIN (a MAX v) *)
let delta_update = fun (s :'a structure) (d :'a list) (a :'a) (v :'a)
  -> List.map (fun x -> s.inf(x,s.sup(a,v))) d
;;

let start_value = fun (s :'a structure) (t :'a arbre)
  -> match t with
   | PL(_) -> s.bot()
   | PN(_) -> s.bot()
   | OL(_) -> s.top()
   | ON(_) -> s.top()
;;


let rec abgdV_ = fun (s :'a structure) (a :'a) (b :'a) (g :'a list) (d :'a list) (v :'a) (t :'a arbre)
  -> match t with

           (*- Noeud Player -*)

             (* Terminal *)
             | PL(x)     -> (x,1)

             (* Non Terminal (dernier fils) *)
             | PN([t'])  -> let a' = s.sup(a,v) in
                          (match abgdV_ s a' b (v::g) ((s.top())::d) (start_value s t') t'
                            with (v',n) -> (s.sup(v,v'),n+1) )

             (* Non Terminal (fils intermédiaire) *)
             | PN(t'::l) -> (match abgdV_ s v (s.top()) [v] [s.top()] (start_value s t') t' with
                          (v1,n1) -> let v'= s.sup(v,v1) in
                                     let g'= gamma_update s g b v' in
                                     let b'= List.hd d in
                                     if ((cut_condition s g' d) || (s.ord(b',v')) )
                                       then (v',n1+1)
                                       else (match (abgdV_ s a b g' d v' (PN l))
                                            with (v2,n2) -> (v2,n1+n2)) )
```

```
      (*- Noeud Opponent -*)

        (* Terminal *)
        | OL(x)     -> (x,1)

        (* Non Terminal (dernier fils) *)
        | ON([t'])  -> let b' = s.inf(b,v) in
                       (match abgdV_ s a b' ((s.bot()) :: g) (v :: d) (start_value s t') t'
                         with (v',n) -> (s.inf(v,v'),n+1) )

        (* Non Terminal (fils intermédiaire) *)
        | ON(t' :: l) -> (match abgdV_ s (s.bot()) v [s.bot()] [v] (start_value s t') t' with
                       (v1,n1) -> let v'= s.inf(v,v1) in
                                  let d'= delta_update s d a v' in
                                  let a'= List.hd g in
                                  if ((cut_condition s g d') || (s.ord(v',a')))
                                    then (v',n1+1)
                                    else (match (abgdV_ s a b g d' v' (ON l))
                                          with (v2,n2) -> (v2,n1+n2)) )
;;

let abgdV = fun (s :'a structure) (t :'a arbre) -> abgdV_ s (s.bot()) (s.top()) [s.bot()] [s.top()] (start_value s t) t;;


(* ------------------------------------------
     Fonction d'évaluation Alpha-beta-gamma-delta

                  Version n°2

          alpha et beta sont des suites (listes)
   ------------------------------------------ *)


(* Pour tous les éléments a_i de la suite a, faire a_i <- a_i MAX v *)
let alpha_update = fun (s :'a structure) (a :'a list) (v :'a)
  -> List.map (fun x -> s.sup(x,v)) a

(* Pour tous les éléments b_i de la suite b, faire b_i <- b_i MIN v *)
let beta_update = fun (s :'a structure) (b :'a list) (v :'a)
  -> List.map (fun x -> s.inf(x,v)) b
;;

(* Pour tous les éléments g_i de la suite g, faire g_i <- g_i MAX (b_i MIN v) *)
let gamma_update2 = fun (s :'a structure) (g :'a list) (b :'a list) (v :'a)
  -> List.map2 (fun x y -> s.sup(x,s.inf(y,v))) g b
;;

(* Pour tous les éléments d_i de la suite d, faire d_i <- d_i MIN (a_i MAX v) *)
let delta_update2 = fun (s :'a structure) (d :'a list) (a :'a list) (v :'a)
  -> List.map2 (fun x y -> s.inf(x,s.sup(y,v))) d a
;;
```

```
let rec abgdV2_ = fun (s :'a structure) (a :'a list) (b :'a list) (g :'a list) (d :'a list) (v :'a) (t :'a arbre)
  -> match t with

          (*- Noeud Player -*)

            (* Terminal *)
            | PL(x)     -> (x,1)

            (* Non Terminal (dernier fils) *)
            | PN([t'])  -> let a' = (alpha_update s a v) in
                         (match abgdV2_ s (v : :a') ((s.top()) : :b)
                                  (v : :g) ((s.top()) : :d)
                                  (start_value s t') t'
                          with (v',n) -> (s.sup(v,v'),n+1) )

            (* Non Terminal (fils intermédiaire) *)
            | PN(t' : :l) -> (match abgdV2_ s [v] [s.top()] [v] [s.top()] (start_value s t') t' with
                          (v1,n1) -> let v'= s.sup(v,v1) in
                                  let g'= gamma_update2 s g b v' in
                                  let b'= List.hd d in
                                  if ((cut_condition s g' d) || (s.ord(b',v')) )
                                    then (v',n1+1)
                                    else (match (abgdV2_ s a b g' d v' (PN l))
                                          with (v2,n2) -> (v2,n1+n2)) )

          (*— Noeud Opponent —*)

            (* Terminal *)
            | OL(x)     -> (x,1)

            (* Non Terminal (dernier fils) *)
            | ON([t'])  -> let b' = (beta_update s b v) in
                         (match abgdV2_ s ((s.bot()) : :a) (v : :b')
                                  ((s.bot()) : :g) (v : :d)
                                  (start_value s t') t'
                          with (v',n) -> (s.inf(v,v'),n+1) )

            (* Non Terminal (fils intermédiaire) *)
            | ON(t' : :l) -> (match abgdV2_ s [s.bot()] [v] [s.bot()] [v] (start_value s t') t' with
                          (v1,n1) -> let v'= s.inf(v,v1) in
                                  let d'= delta_update2 s d a v' in
                                  let a'= List.hd g in
                                  if ((cut_condition s g d') || (s.ord(v',a')))
                                    then (v',n1+1)
                                    else (match (abgdV2_ s a b g d' v' (ON l))
                                          with (v2,n2) -> (v2,n1+n2)) )
;;

let abgdV2 = fun (s :'a structure) (t :'a arbre) -> abgdV2_ s [s.bot()] [s.top()]
                                                       [s.bot()] [s.top()]
                                                        (start_value s t) t;;
```

```
(* ------------------------------------------------------------
         Mise en place des tests
   ------------------------------------------------------------ *)

(* fl est une liste de fonction, à comparer avec celle de référence f *)
let rec multi_test = fun f fl (s :'a structure) (p :profondeur) (l :largeur) (n :int) ->
  let compare = (fun f1 (s :'a structure) t f2 ->
  let ((x1,n1'),(x2,n2')) = ((f1 s t),(f2 s t)) in
  let n1=float_of_int(n1') and n2=float_of_int(n2') in
    ((if (x1=x2) then 1 else 0),(n2/.n1),(if (x1=x2) then 0 else 1),int_of_float(n2),int_of_float(n1))
    ) in
  let add = (fun (x,y,z,w,n) (x',y',z',w',n') -> (x+x',y+.y',z+z',w+w',n+n')) in
  let rec addlists = (fun l1 l2 -> match l1,l2 with
                              | ([t],[t'])      -> [add t t']
                              | (t : :l,t' : :l') -> (add t t') : :addlists l l' )
  in
  let nf = List.length fl in
  let subtot = ref (mklist nf (0,0.,0,0,0)) in
  let i = ref 1 in
  let alea = (tirage_arbre l) in
  while ( !i<=n) do
    let t = (alea s p) in
    let lc = List.map (compare f s t) fl in
    subtot := addlists ( !subtot) lc ;
    i := !i + 1 ;
  done ;
   !subtot
; ;

let affiche = (fun alg (x,y,z,w,tot) ->
  let x'= float_of_int(x) and z'= float_of_int(z) in
  let p = floor(10000. *. y/.x')/. 100. in
  (* moyenne des noeuds visités pour chaque arbre *)
  let noeuds = int_of_float((float_of_int(w)) /. (x'+.z')) in
  print_string (alg^"\r\t\t\t "^string_of_int(x)^" ("
  ^string_of_float(100. *. x'/.(x'+.z'))^"%)\r\t\t\t\t\t\t "
  ^string_of_int(noeuds)^" ("^string_of_float(p)^"%)\n")) ; ;

let table = fun (s,p,l,n) -> (* Structure , profondeur des arbres , largeur des arbres, nb de calculs *)
  print_string ("Profondeur des arbres de jeu : "^string_of_int(p)^"\n") ;
  print_string ("Largeur                      : "^(if l=0 then "arbitraire" else string_of_int(l))^"\n") ;
  print_string ("Nombre de tests              : "^string_of_int(n)^"\n\n") ;
  (match (multi_test exhaustive [exhaustive ; minimax ; alpha_beta ; abgdV ; abgdV2 ] s p l n) with
  [s1 ;s2 ;s3 ;s6 ;s7] ->
     print_string "Algorithme\r\t\t\tCalculs corrects\r\t\t\t\t\t\tNoeuds visites (moyennes)\n" ;
     print_string "---------\r\t\t\t----------------\r\t\t\t\t\t\t------------- \n" ;
     affiche "Exaustif" s1 ;
     affiche "Minimax" s2 ;
     affiche "Alpha-Beta" s3 ;
     affiche "Alpha-Beta-Gamma-Delta" s6 ;
     affiche "Alpha-Beta-Gamma-Delta2" s7 ;
     print_string "\n") ; ; ;
```

```
(* MAIN *)

let n = 20000 in (
  print_string ("+----------------------------------------------------------------+\n\n") ;
  print_string ("| STRUCTURE : "^mod2.name^"\r\t\t\t\t\t\t\t\t\t|\n\n") ;
  print_string ("+----------------------------------------------------------------+\n") ;
  mod2.show() ;
  List.iter table [(mod2,3,3,n) ;(mod2,5,3,n) ;(mod2,7,3,n) ;(mod2,11,3,n) ;(mod2,15,3,n)] ;
  print_string ("\n\t\t+------------------------------+\n") ;
  print_string ("\t\t|  LARGEUR Arbitraire    |\n") ;
  print_string ("\t\t+------------------------------+\n\n") ;
  (let n = n/5 in List.iter table [(mod2,3,0,n) ;(mod2,5,0,n) ;(mod2,7,0,n) ;(mod2,11,0,n) ;(mod2,15,0,n) ;(mod2,17,0,n)]) ;
  print_string ("+----------------------------------------------------------------+\n\n") ;
  print_string ("| STRUCTURE : "^mod1.name^"\r\t\t\t\t\t\t\t\t\t|\n\n") ;
  print_string ("+----------------------------------------------------------------+\n") ;
  mod1.show() ;
  print_string ("\n\t\t+-------------------------------+\n") ;
  print_string ("\t\t| LARGEUR 3 (Arbres ternaires) |\n");
  print_string ("\t\t+-------------------------------+\n\n") ;
  List.iter table [(mod1,3,3,n)] ;
  List.iter table [(mod1,5,3,n) ;(mod1,7,3,n) ;(mod1,11,3,n) ;(mod1,15,3,n)] ;
  print_string ("\n\t\t+------------------------------+\n") ;
  print_string ("\t\t| LARGEUR 2 (Arbres binaires) |\n") ;
  print_string ("\t\t+------------------------------+\n\n") ;
  List.iter table [(mod1,3,2,n)] ;
  List.iter table [(mod1,5,2,n) ;(mod1,7,2,n) ;(mod1,11,2,n) ;(mod1,15,2,n)] ;
  print_string ("\n\t\t+------------------------------+\n") ;
  print_string ("\t\t|  LARGEUR Arbitraire    |\n") ;
  print_string ("\t\t+------------------------------+\n\n") ;
  (let n = n/5 in List.iter table [(mod1,3,0,n) ;(mod1,5,0,n) ;(mod1,7,0,n) ;(mod1,11,0,n) ;(mod1,15,0,n) ;(mod1,17,0,n)]) ;
  print_string ("+----------------------------------------------------------------+\n\n") ;
  print_string ("| STRUCTURE : "^mod3.name^"\r\t\t\t\t\t\t\t\t\t|\n\n") ;
  print_string ("+----------------------------------------------------------------+\n") ;
  mod3.show() ;
  print_string ("\n\t\t+-------------------------------+\n") ;
  print_string ("\t\t| LARGEUR 3 (Arbres ternaires) |\n");
  print_string ("\t\t+-------------------------------+\n\n") ;
  List.iter table [(mod3,3,3,n)] ;
  List.iter table [(mod3,5,3,n) ;(mod3,7,3,n) ;(mod3,11,3,n) ;(mod3,15,3,n)] ;
  print_string ("\n\t\t+------------------------------+\n") ;
  print_string ("\t\t| LARGEUR 2 (Arbres binaires) |\n") ;
  print_string ("\t\t+------------------------------+\n\n") ;
  List.iter table [(mod3,3,2,n)] ;
  List.iter table [(mod3,5,2,n) ;(mod3,7,2,n) ;(mod3,11,2,n) ;(mod3,15,2,n)] ;
  print_string ("\n\t\t+------------------------------+\n") ;
  print_string ("\t\t|  LARGEUR Arbitraire    |\n") ;
  print_string ("\t\t+------------------------------+\n\n") ;
  (let n = n/5 in
    (List.iter table [(mod3,3,0,n) ;(mod3,5,0,n) ;(mod3,7,0,n) ;(mod3,11,0,n) ;(mod3,15,0,n) ;(mod3,17,0,n)]) ;

)
```

Cinquième partie

Bibliographie et Index des définitions

Bibliographie

[Abramsky & Jagad., 1994] S. Abramsky and R. Jagadeesan. *Games and full completeness for multiplicative linear logic.* The Journal of Symbolic Logic, 59(2), pp. 543-574. 1994.

[Abramsky & Jung, 1994] S. Abramsky and A. Jung. *Domain Theory.* In the Handbook of Logic in Computer Science. Vol. 3, pp. 1-168. Oxford University Press. 1994.

[Amadio & Curien, 1998] R. Amadio and P.-L. Curien. *Domains and Lambda-Calculi.* Cambridge Tracts in Theoretical Computer Science. Vol. 46. Cambridge University Press. 1998.

[Asperti & Ciabattoni, 1997] A. Asperti and A. Ciabattoni. *Logica a Informatica.* McGraw-Hill. Milano. 1997.

[Baillot et al., 1997] P. Baillot, V. Danos, T. Ehrhard and L. Régnier. *Believe it or not, AJM's games model is a model of classical linear logic.* Proceedings of the twelfth Symposium on Logic in Computer Science. Varsovie. 1997.

[Barendregt, 1984] H. Barendregt. *The Lambda Calculus, Its Syntax and Semantics.* Studies in Logic and the Foundations of Mathematics. Vol. 103. North-Holland Publishing Company. Amsterdam. 1984.

[Blass, 1992] A. Blass. *A game semantics for linear logic.* Annals of Pure and Applied Logic, 56(1-3), pp. 183-220. 1992.

[Bossi et al., 1994] A. Bossi, M. Gabbrielli, G. Levi and M. Martelli. *The s-semantics approach : Theory and applications.* Journal of Logic Programming, 19-20, pp. 149-197. 1994.

[Burris & Sankap., 1981] S. Burris and H.P. Sankappanavar. *A Course in Universal Algebra.* Springer Verlag. Berlin. 1981.

[Courcelle, 1983] B. Courcelle. *Fundamental properties of infinite trees.* Theoretical computer science 25 (1983), pp. 95-169. 1983.

[Cousot & Cousot, 1977] P. Cousot and R. Cousot. *Abstract Interpretation : A Unified Lattice Model for Static Analysis of Programs by Construction or Approximations of Fixpoints.* In Proceedings of the 4^{th} Annual ACM Symposium on Principles of Programming Languages. ACM Press. 1977.

[Cousot & Cousot, 1979] P. Cousot and R. Cousot. *Systematic design of program analysis frameworks.* In Proceedings of the 6^{th} Annual ACM Symposium on Principles of Programming Languages. ACM Press. 1979.

[Cousot & Cousot, 1992] P. Cousot and R. Cousot. *Abstract interpretation and application to logic programs.* Journal of Logic Programming. Vo.13, n.2-3, pp.103-179. 1992.

[Curien & Herbelin, 1998] P.-L. Curien and H. Herbelin. *Computing with Abstract Bohm Trees.* In Proceedings of the 3^{th} Fuji International Symposium on Functional and Logic Programming. Eds M. Sato & Y. Toyama, World Scientific, 20-3. 1998.

[Danos & Harmer, 2000] V. Danos and R. Harmer. *Probabilistic Game Semantics.* Proceedings of the 15^{th} Symposium on Logic in Computer Science, pp. 204-213. 2000.

[Davey & Priestley, 1990] B. A. Davey and H. A. Priestley. *Introduction to Lattices and Order.* Cambridge University Press. 1990.

[Di Cosmo & Loddo, 2000] R. Di Cosmo and J.-V. Loddo. *Playing Logic Programs with the Alpha-Beta Algorithm.* In proceedings of the 7^{th} International Conference on Logic for Programming and Automated Reasoning (LPAR'2000). St. Gilles, Ile de la Réunion, France. 2000.

[Di Cosmo et al.,1998] R. Di Cosmo, J.-V. Loddo and S. Nicolet. *A Game Semantics Foundation for Logic Programming.* In proceedings of the 6^{th} International Conference on Algebraic and Logic Programming (PLIP'98). LNCS, Springer-Verlag. Pisa, Italy. 1998

[Diderich & Gengler, 1995] C. G. Diderich and M. Gengler. *Minimax and Its Applications.* Kluwer Academic Publishers, The Netherlands. Chapter : *A Survey on Minimax Trees and Associated Algorithms*, pp. 25-54. 1995.

[Eder, 1985] E. Eder. *Properties of substitutions and unifications.* Journal of symbolic Computation, Vol. 1, pp. 31-46. 1985.

[Fages, 1996] F. Fages. *Programmation logique par contraintes.* Elipse, École Polytechnique. 1996.

[Fages, 1997] F. Fages. *Constructive negation by pruning.* Journal of Logic Programming, 32(2), pp. 85-118. 1997.

[Falaschi et al., 1993] M. Falaschi, G. Levi, M. Martelli, and C. Palamidessi. *A Model-Theoretic Reconstruction of the Operational Semantics of Logic Programs.* Information and Computation, 102(1), pp 86-113. 1993.

[Fuller et al., 1973] S. H. Fuller, J. G. Gaschnig, and J. J. Gillogly. *Analysis of the alpha-beta pruning algorithm.* Technical report. Department of Computer Science, Carnegie-Mellon University, Pittsburgh, Pennsylvania. 1973.

[Gabbrielli et al., 1995] M. Gabrielli, G. M. Dore, and G. Levi. *Observable semantics for constraint logic programs.* Journal of Logic and Computation, 5(2), pp. 133-171. 1995.

(1) (correction) toute réponse calculée est correcte, i.e. une réponse qui implique la conjonction $\neg g$:

[Gécseg & Steinby, 1997] F. Gécseg and M. Steinby. *Tree Languages.* In the Handbook of Formal Languages. Vol. 3. Springer-Verlag. 1997.

[Gericke, 1966] H. Gericke. *Lattice Theory.* Frederick Ungar Publishing Co. New York. 1966.

[Hart & Edwards, 1961] T. P. Hart and D. J. Edwards. *The Tree Prune (TP) algorithm.* Artificial Intelligence Project Memo 30. MIT Research Laboratory of Electronics and Computation Center, Cambridge, Massachusetts. 1961.

[Honsell & Sanella, 1999] F. Honsell and D. Sanella. *Pre-logical Relations.* Technical Report ES-LFCS-99-405. University of Edinburgh. 1999.

[Jaffar & Maher, 1994] J. Jaffar and M. J. Maher. *Constraint logic programming : A survey.* Journal of Logic Programming, 19/20, pp. 503-581. 1994.

[Joyal, 1995] A. Joyal. *Free Lattices, Communication and Money Games.* In Proceedings of the International Congress of Methodology and Philosophy of Sciences. Springer Verlag. Firenze, Italy. 1995.

[Karp, 1991] R. M. Karp. *An introduction to randomized algorithms.* Discrete Applied Mathematics 34, pp. 165-201. 1991.

[Karp & Rabin, 1987] R. M. Karp and M. O. Rabin. *Efficient randomized pattern-matching algorithms.* IBM Journal of Research and Development, Vol. 31, No. 2, pp. 249-260. 1987.

[Knuth & Moore, 1975] D. E. Knuth and R. W. Moore. *An analysis of alpha-beta pruning.* Artificial Intelligence, Vol. 6, pp. 293-326. 1975.

[Lamarche, 1995] F. Lamarche. *Games semantics for full propositional linear logic.* In Proceedings of 10^{th} Annual IEEE Symposium on Logic in Computer Science, IEEE Computer Society Press, pp. 464-473. San Diego, California. 1995.

[Lassaigne & Rougemont, 1993] R. Lassaigne and M. de Rougemont. *Logique et fondements de l'informatique.* Hermès. 1993

[Lipson, 1981] J. D. Lipson. *Elements of Algebra and Algebraic Computing.* AddisonWesley. 1981

[Loddo & Nicolet, 1998] J.-V. Loddo and S. Nicolet. *Théorie des jeux et langages de programmations.* Technical report. École normale supérieure (ENS), 45 Rue d'Ulm, Paris, France. May 1998.

[Loeckx et al., 2000] J. Loeckx, H.-D. Ehrich and M. Wolf. *Algebraic Specification of Abstract Data Types.* Handbook of Logic in Computer Science, Vol. 5, pp. 217-316. Oxford University Press. 2000.

[Lloyd, 1987] J.W. Lloyd. *Foundation of Logic Programming.* Springer Verlag. 1987.

[Mal'cev, 1973] A. I. Mal'cev. *Algebraic Systems.* Springer Verlag. 1973.

[Marriot et al., 1994] K Marriot, H Søndergaard, and N D Jones. *Denotational Abstract Interpretation of Logic Programs.* ACM Transactions on Programming Languages and Systems. 1994

[Mitchell, 1996] J. Mitchell. *Foundations for Programming Languages.* MIT Press. 1996.

[Motwani & Raghavan, 1995] R. Motwani, P. Raghavan. *Randomized Algorithms.* Cambrigde University Press. 1995.

[Owen, 1968] G. Owen. *Game Theory.* Saunders. Philadelphia. 1968.

[Palamidessi, 1990] C. Palamidessi. *Algebraic Properties of Idempotent Substitutions.* In Proceedings of the 17^{th} International Colloquium on Automata, Languages and Programming (ICALP). LNCS 443, pp. 386-399. Springer Verlag. Warwick, England. 1990

[Pearl, 1990] J. Pearl. *Heuristique. Stratégie de Recherche Intelligente pour la Résolution de Problème par Ordinateur.* Coll. Intelligence Artificielle. Cepaduès. 1990.

[Perrin & Pin, 2001] D. Perrin and J-E. Pin. *Infinite Words.* To publish ; available from `http ://www.liafa.jussieu.fr/~jep/Resumes/InfiniteWords.html`. February 2001.

[Plaat, 1996] A. Plaat. *Research Re : search & Re-search.* PhD Thesis. Tinbergen Institute and Department of Computer Science, Erasmus University Rotterdam. Thesis Publishers, Amsterdam, The Netherlands. June 1996.

[Plotkin, 1993] B. Plotkin. *Universal algebra, Algebraic logic and Databases.* Kluwer Publ.Co. 1993.

[Sedgewick, 1978] R. Sedgewick. *Implementing Quicksort Programs.* CACM 21(10), pp. 847-857. 1978

[Sterling & Shapiro, 1986] L. Sterling and E. Shapiro. *The Art of Prolog.* MIT Press. 1986.

[Stirling, 1997] C. Stirling. *Bisimulation, model checking and other games.* Notes for a Mathfit instructional meeting on games and computation, held in Edinburgh, Scotland. June 1997.

[Taylor, 1999] Paul Taylor. *Practical Foundations of Mathematics.* Cambridge University Press. 1999.

[Tucker & Zucker, 2000] J. V. Tucker and J. I. Zucker. *Computable Functions and Semicomputable Sets on Many-Sorted Algebras.* Handbook of Logic in Computer Science, Vol. 5, pp. 317-523, Oxford University Press. 2000.

[von Neumann, 1928] J. von Neumann. *Zur Theorie der Gesellschaftsspiele.* In Mathaematische Annalen 100, pp. 195-320. 1928.

[Welsh, 1983] D. J. A. Welsh. *Randomized algorithms.* Discrete Applied Mathematics 5, pp. 133-145. 1983.

[Zermelo, 1913] E. Zermelo. *Uber eine Anwendung der Mengellebre auf die Theorie des Schachspiels.* In Proceedings of the 5th International Congress of Mathematicians, Vol II, pp. 501-504. Cambridge University Press. 1913.

Index

V