Compilation: Syllabus

8 janvier 2021

1 Informations

Nom du cours : Compilation
Niveau : M1
Formation : Master d'informatique
Page du cours : https://lipn.univ-paris13.fr/~breuvart/compilation/
Responsable du cours : Flavien Breuvart
Chargés de TDs : Flavien Breuvart et Virgile Mogbil
Chargés de TPs : Flavien Breuvart et Virgile Mogbil
Email : breuvart@lipn.fr
Bureau : A209
Période du cours : Janvier-Avril 2021
Découpage du cours : 18h de CM, 6h de TP, 12h de TD et un projet **Notation :** $\texttt{max}\left(\texttt{Exam}, \frac{\texttt{Exam+Projet}}{2}\right)$

2 Prérequis

Automates : Une très bonne maîtrise de la théorie des automates est essentiel pour la première partie du cours. En particulier vous êtes sensé pouvoir déterminiser un automate aisément.

Grammaire : Les grammaires sont au centre de la première partie du cours. Il faut absolument savoir lire une grammaire.

Expressions régulière : Il faut connaître les expression régulière et sa grammaire, ainsi que trouver l'automate correspondant à une expression régulière. Dans l'idéal, il faut savoir coder la fonction qui transforme une expression régulière en automate.

Manipulation d'arbres : Vous êtes sensé savoir comprendre et coder des manipulations difficiles sur des arbres sans que cela demande le moindre effort.

Programmation (fonctionnel) : Il est attendu que vous sachiez coder correctement dans au moins un langage. Dans l'ordre de préférence : OCaml, C, Java, autre langage fonctionnel, autre langage. Les langages fonctionnels sont préférable car ils rendent la seconde partie du projet plus aisée (il s'agit d'une manipulation d'arbres...).

3 Objectifs

3.1 Les différents types de connaissances

Comprendre les principes de base de la compilation : Il s'agit de comprendre le sens de la compilation et de l'interprétation de code, ainsi que les différentes phases associées.

Savoir utiliser des générateurs de lexeurs et parseurs : À travers les TPs et le projet, vous serez amenés à utiliser un générateur de parseurs et un générateur de lexeurs. En avoir utilisé une fois est important : cela vous permettra de ne pas hésiter à utiliser cet outils lorsque vous pourrez en avoir besoin dans le futur.

Avoir programmé un compilateur de bout en bout : Le projet à beau être presque facultatif, il s'agit d'un apport réel pour le compréhension des langages de programmation.

Comprendre et maîtriser les algos de génération de lexeurs et de parseurs LR : La première partie du cours portera sur ces algorithmes. Bien que vous ayez peu de chance d'en avoir besoin plus tard, il s'agit d'un "cas d'école" d'algorithme pratique obtenu par dégradation d'une théorie.

Comprendre et utiliser le principe de machine abstraite et d'assembleur : La seconde partie du cours parlera de machine abstraite, on verra comment la sémantique d'un langage est liée à une machine simplifiée qui peut servir de langage intermédiaire pour la compilation.

Culture général sur la compilation : Tout au long de l'année, on décrira superficiellement différents aspects de la compilation afin de vous fournir une culture général sur le sujet.

Utilisation de git pour gérer un projet : Le projet donné a une progression bidimentionel (taille du fragment vs étapes de compilations), cette structure permetra d'explorer les avantage d'un gestionnaires de version, en l'occurence *git*.

3.2 Acquis nécessaires (objectif 7)

Les compétences suivantes sont requises, si vous ne les avez pas, vous n'aurez pas 7 à l'examen : [1]

- savoir déterminiser un automate et donner l'automate d'une expression régulière (cf. prérequis),
- savoir donner l'expression régulière (avec le formalisme de votre choix) à partir d'un énoncé simple en français,
- savoir lire une grammaire et comprendre les notions simples associés (ambiguïté, priorité),
- connaître et comprendre les grandes étapes de la compilation,
- savoir lire et écrire des programmes très simples (fragment 2 : arithmétique, variables, ifs) en assembleur,
- connaître et comprendre les structures compilés (environnement, objet, clôture, boucle...),

3.3 Acquis minimaux (objectif 12)

Les compétences suivantes devraient être suffisantes pour avoir 11-12 :

- comprendre la différences entre compilateur, interpréteur et machine virtuelle,
- savoir implémenter les algos de détermination d'un automate et de création d'un automate à partir d'une expression régulière,
- savoir donner l'expression régulière (avec le formalisme de votre choix) à partir d'un énoncé complexe en français,
- savoir écrire une grammaire à partir d'un énoncé simple en français,
- comprendre les notions plus complexes (associativités, LR_0),
- savoir utiliser un générateur de lexeur et de parseur (dans le langage de votre choix) à un niveau basique,
- pouvoir implémenter la traduction automatique de programmes très simples (bloc 1) en assembleur,
- savoir lire et écrire des programmes moyennement complexes (bloc 5 : boucles, fonctions...) en assembleur,
- savoir calculer un lexeur à partir d'expressions régulières simples,
- savoir calculer un parseur LR_0 et l'utiliser,
- savoir calculer les *firsts* et *follows*.

3.4 Acquis moyens (objectif 17)

Les compétences suivantes devraient être suffisantes pour avoir 16-17 (attention, cela descend très vite)

- savoir implémenter les algos de lexeur et LR_0,
- connaître et comprendre le *maximum munch*,
- savoir calculer un lexeur à partir d'expressions régulières complexes,

1. Une erreur d'inattention passera, mais de multiples erreurs fondamentales ne seront pas acceptés.

— savoir calculer un parseur SLR et l'utiliser,
— savoir écrire une grammaire à partir d'un énoncé complexe en français,
— pouvoir implémenter la traduction automatique de programmes moyennement complexes (bloc 2) en assembleur,
— savoir lire et écrire des programmes complexes (bloc 3 : objet, fonctionnel...) en assembleur,
— connaître les différents parseurs et les relations d'inclusions,
— connaître la notion de conservation de la sémantique,
— comprendre ce qu'est un *garbage colector*,
— comprendre la sémantique d'une nouvelle machine abstraite simple.

3.5 Acquis totaux (objectif 20)

Les compétences suivantes sont abordées dans le cours, elles ne là que pour avoir une excellente note (on peut avoir 20 sans toutes les maîtriser) : [2]

— savoir calculer un parseur LALR et LR_1,
— savoir implémenter l'algo LALR,
— connaître l'existence des algos non détaillés en cours (LL_1, LL_k, LL_*, *GLR*...),
— savoir modifier une grammaire pour la rendre non-ambiguë ou LR_0/SLR/LALR,
— pouvoir implémenter la traduction automatique de programmes complexes (bloc 3) en assembleur,
— comprendre la sémantique d'un langage ou d'une machine abstraite complexe (ex. : webassembly),
— compiler vers une nouvelle machine abstraite,
— comprendre les notions d'appels par nom/valeur,
— savoir faire de petites optimisations de compilation,
— avoir une idée de comment marche un *garbage colector*,
— connaître le noms de quelques procédure d'optimisation/vérification (typage, interprétation abstraite, *model checking*, recherche de patterns, inlinning...),
— savoir ce que sont les JITs.

Distanciel : Si nous restons en distanciel tout le semestre, le contenu du cours sera revu légèrement à la baisse. En particulier les algorithmes de parsing avancés seront plus optionels. A contrario, le projet, qui peut se traiter de manière plus autonome, sera mis en avant et ses composantes tomberont plus aisément à l'examen.

4 Déroulement du cours

Plan de cours : Voir document dédié.

2. Cette liste fait référence à un déroulement idéal du cours, avec deux séances bonus que l'on aura peut être (probablement) pas le temps de faire.

Projet : Voir document dédié.

CM : Une grande partie du cours prendra la forme d'exemples : Les algorithmes et théorèmes seront énoncés rapidement puis déroulés sur des exemples simples. Pour ce qui est de la formalisation, elle est faite dans le polycopié associé qui n'est pas un résumé optionnel du cours, mais un complément **nécessaire**. En particulier, certains exercices de TD ou même d'examen pourront utiliser la formalisation plus précise du polycopier.

CM distanciel : Les cours qui devront se faire en distanciel se feront *a priori* sur Zoom. S'il y a une forte demande pour une autre plateforme, nous pouront considérer d'en changer.

TP : Il n'y aura que 4 séances de TPs pour se familiariser avec le projet et le commencer. Le langage de programmation pour le TP sera au choix parmi une liste proposé (avec au moins Java, C, Ocaml et Haskell). Atention, le fait que l'on commence le projet en TP ne signifie pas que les TPs sont facultatifs, mais que ces composantes du projets peuvent tomber à l'examen. De plus, il s'agit bien de "commencer" le projet, il y a du travail à fournir chez vous entre et après les TPs.

TP distanciel : Les TPs qui devront se faire en distanciel se feront *a priori* en classe inversé : vous faites le TP et on utilise la séance pour répondre à des questions. La tennue exacte de ces séances est encore à définir (ce document sera alors is à jours).

TD : Les TDs vont prendre la forme d'application d'algorithmes sur des exemples donnés au tableau. Il n'y aura pas toujours de feuille d'exo ou de correction.

TD distanciel : Les TDs distanciels ont tendance à être plus magistraux, mais c'est généralement par un manque de participation, investissez vous et on poura en faire un vrais TD dynamique.

Matériel : Tout le matériel du cours, y compris des années passées, sont disponible sur la page web de l'enseignant. Certains sont obselet, certains autres ont des parties qui n'ont pas été mis à jours (si on vous parle de Pascal ou de Python, c'est probablement le cas).

5 Comportement

Retard : Pour les CMs, les retards sont tolérés, mais les retardataires doivent rester silencieux (pas de check !) et s'asseoir derrière. Je ne répondrais pas aux questions des retardataires. Pour les TP/TDs, cela dépendra du chargé de TP.

Absence : Mise à part pour les élèves ayant fait valoir une activité professionnelle en parallèle, le cours et son examination sont calibrés sur une présence assidue. Si vous faites le choix de sécher des CMs ou des TDs, assurez vous de bien rattraper...

Questions par mail : Je répond généralement assez rapidement aux questions par mail (si elles sont courtoises). [3] Si vous n'avez pas de réponses au bout de 3-4 jours, n'hésitez pas à la reposer, je l'ai peut être laissé filée par erreur (je peux avoir jusque 50-60 mails par jours...). Par contre, plus la date de l'examen approche, moins je suis enclin à répondre aux questions (simplement car j'en ai beaucoup plus...).

3. Autre précision : les mails sans ponctuation (vraisemblablement écris au dictaphonne) m'énèrvent au plus haut point, veuillez éviter.