The Bang Calculus and the Two Girard’s Translations

Giulio Guerrieri
Dipartimento di Informatica – Scienza e Ingegneria (DISI), Universitá di Bologna, Bologna, Italy
giulio.guerrieri@unibo.it

Giulio Manzonetto
LIPN, UMR 7030, Université Paris 13, Sorbonne Paris Cité, F-93430, Villetaneuse, France
giulio.manzonetto@lipn.univ-paris13.fr

We study the two Girard’s translations of intuitionistic implication into linear logic by exploiting the bang calculus, a paradigmatic functional language with an explicit box-operator that allows both the call-by-name and call-by-value λ-calculi to be encoded in. We investigate how the bang calculus subsumes both call-by-name and call-by-value λ-calculi from a syntactic and a semantic viewpoint.

1 Introduction

The λ-calculus is a simple framework formalizing many features of functional programming languages. For instance, the λ-calculus can be endowed with two different evaluation mechanisms, call-by-name (CbN) and call-by-value (CbV), which have quite different properties. A CbN discipline re-evaluates an argument each time it is used. On the contrary, a CbV discipline evaluates an argument just once and recalls its value each time it is used. CbN and CbV λ-calculi are usually defined by means of operational rules giving rise to two different rewriting systems on the same set of λ-terms: in CbN there is no restriction on firing a β-redex, whereas in CbV a β-redex can be fired only when the argument is a value, i.e. a variable or an abstraction. The standard categorical setting for describing denotational models of the λ-calculus, cartesian closed categories, provides models which are adequate for CbN, but typically not for CbV. For CbV, the introduction of an additional computational monad (in the sense of Moggi [22, 23]) is necessary. While CbN λ-calculus [4] has a rich and refined semantic and syntactic theory featuring advanced concepts such as separability, solvability, Böhm trees, classification of λ-theories, full-abstraction, etc., this is not the case for CbV λ-calculus [25], in the sense that concerning the CbV counterpart of these theoretical notions there are only partial and not satisfactory results (or they do not exist at all!).

Quoting from [17], “the existence of two separate paradigms is troubling” for at least two reasons:

• it makes each language appear arbitrary (whereas a unified language might be more canonical);
• each time we create a new style of semantics, e.g. Scott semantics, operational semantics, game semantics, continuation semantics, etc., we always need to do it twice — once for each paradigm.

Girard’s Linear Logic (LL, [15]) provides a unifying setting where this discrepancy could be solved since both CbN and CbV λ-calculi can be faithfully translated, via two different translations, into LL proof-nets. Following [17], we can claim that, via these translations, LL proof-nets “subsume” the CbN and CbV paradigms, in the sense that both operational and denotational semantics for those paradigms can be seen as arising, via these translations, from similar semantics for LL.

Indeed, LL can be understood as a refinement of intuitionistic logic (and hence λ-calculus) in which resource management is made explicit thanks to the introduction of a new pair of dual connectives: the exponentials “!” and “?” . In proof-nets, the standard syntax for LL proofs, boxes (introducing the
modality “!”) mark the sub-proofs available at will: during cut-elimination, such boxes can be erased (by weakening rules), can be duplicated (by contraction rules), can be opened (by dereliction rules) or can enter other boxes. The categorical counterpart of this refinement is well known: it is the notion of a cartesian +-autonomous category, equipped with a comonad endowed with a strong monoidal structure. Every instance of such a kind of structure yields a denotational model of LL.

In his seminal article \[15\], Girard proposes a standard translation of intuitionistic logic (and hence simply typed \(\lambda\)-calculus) in multiplicative-exponential LL proof-nets whose semantic counterpart is well known: the Kleisli category of the exponential comonad “!” is cartesian closed thanks to the strong (by weakening rules), can be duplicated (by contraction rules), can be opened (by dereliction rules) or structural rules being well known: if the translation \((\cdot)^N\) maps the intuitionistic implication \(A \Rightarrow B\) to the LL formula \(!A^N \rightarrow B^N\). In \[15\] Girard proposes also another translation \((\cdot)^V\) that he calls “boring”: it maps the intuitionistic implication \(A \Rightarrow B\) to the LL formula \(!A^V \rightarrow !B^V\). Since the untyped \(\lambda\)-calculus can be seen as simply typed with only one ground type \(o\) satisfying the recursive identity \(o = o \Rightarrow o\), the two Girard’s translations \((\cdot)^N\) and \((\cdot)^V\) decompose this identity into \(o = !o \rightarrow o\) and \(o = !(o \rightarrow o)\) (or equivalently, \(o = !o \rightarrow !o)\), respectively. At the \(\lambda\)-term level, these two translations differ only by the way they use logical exponential rules (\(i.e.\) box and dereliction), whereas they use multiplicative and structural (\(i.e.\) contraction and weakening) ingredients in the same way. Because of this difference, the translation \((\cdot)^N\) en codes the CbN \(\lambda\)-calculus into LL proof-nets (in the sense that CbN evaluation \(\rightarrow_\beta\) is simulated by cut-elimination via \((\cdot)^N\)), while \((\cdot)^V\) encodes the CbV \(\lambda\)-calculus of LL proof-nets (CbV evaluation \(\rightarrow_\beta^v\) is simulated by cut-elimination via \((\cdot)^V\)). Indeed, since in CbN \(\lambda\)-calculus there is no restriction on firing a \(\beta\)-redex (its argument can be freely copied or erased), the translation \((\cdot)^N\) puts the argument of every application into a box (see \[7, 27, 16\]); on the other hand, the translation \((\cdot)^V\) puts only values into boxes (see \[2\]) since in CbV \(\lambda\)-calculus values are the only duplicable and discardable \(\lambda\)-terms. Thus, as deeply studied in \[20\], the two Girard’s logical translations explain the two different evaluation mechanisms, bringing them into the scope of the Curry-Howard isomorphism.

The syntax of multiplicative-exponential LL proof-nets is extremely expressive and powerful, but it is too general and sophisticated for the computational purpose of representing purely functional programs. For instance, simulation of \(\beta\)-reduction on LL proof-nets passes through intermediate states/proof-nets that cannot be expressed as \(\lambda\)-terms, since LL proof-nets have many spurious cuts with axioms that have no counterpart on \(\lambda\)-terms. More in general, LL proof-nets are manipulated in their graphical form, and while this is a handy formalism for intuitions, it is far from practical for formal reasoning.

From the analysis of Girard’s translations it seems worthwhile to extend the syntax of the \(\lambda\)-calculus to internalize the insights coming from LL in a \(\lambda\)-like syntax. The idea is to enrich the \(\lambda\)-calculus with explicit boxes marking the “values” of the calculus, \(i.e.\) the terms that can be freely duplicated and discarded: such a linear \(\lambda\)-calculus subsumes both CbN and CbV \(\lambda\)-calculi, via suitable translations. This, of course, has been done quite early in the history of LL by defining various linear \(\lambda\)-calculi, such as \[19, 11, 5, 28, 20\].

All these calculi require a clear distinction between linear and non-linear variables, structural rules being freely (and implicitly) available for the latter and forbidden for the former. This distinction complicates the formalism and is actually useless as far as we are interested in subsuming \(\lambda\)-cali

Inspired by Ehrhard \[13\], in \[14\] it has been introduced an intermediate formalism enjoying at the same time the conceptual simplicity of \(\lambda\)-calculus (without any distinction between linear and non-linear variables) and the operational expressiveness of LL proof-nets: the bang calculus. It is a variant of the \(\lambda\)-calculus which is “linear” in the sense that the exponential rules of LL (box and dereliction) are part of the syntax, so as to subsume CbN and CbV \(\lambda\)-calculi via two translations \((\cdot)^p\) and \((\cdot)^v\), respectively.

\[1\] Actually the full symmetry of such a category is not really essential as far as the \(\lambda\)-calculus is concerned, it is however quite natural from the LL viewpoint: LL restores the classical involutivity of negation in a constructive setting.
from the set \( \Lambda \) of \( \lambda \)-terms to the set \( !\Lambda \) of terms of the bang calculus (see §2). These two translations are deeply related to Girard’s encodings \((\cdot)^{N}\) and \((\cdot)^{V}\) of CbN and CbV \( \lambda \)-calculi into LL proof-nets. Indeed, Girard’s translations \((\cdot)^{N}\) and \((\cdot)^{V}\) decompose in such a way that the following diagrams commute:

\[
\Lambda \xrightarrow{(\cdot)^{N}} LL \quad \Lambda \xrightarrow{(\cdot)^{V}} LL
\]

where \((\cdot)^{V}\) is a natural translation of the bang calculus into multiplicative-exponential LL proof-nets. Thus, the bang calculus internalizes the two Girard’s translations in a \( \lambda \)-like calculus instead of LL proof-nets. It subsumes both CbN and CbV \( \lambda \)-calculi in the same rewriting system and denotational model, so that it may be a general setting to compare CbN and CbV. The bang calculus can be seen as a metalanguage where the choice of CbN or CbV evaluation depends on the way the term is built up. If we consider the syntax of the \( \lambda \)-calculus as a programming language, issues like CbN versus CbV evaluations affect the way the \( \lambda \)-calculus is translated in this metalanguage, but does not affect the metalanguage itself.

It turns out that this bang calculus was already known in the literature: it is an untyped version of the implicative fragment of Paul Levy’s Call-By-Push-Value calculus \([17, 18]\). Interestingly, his work was not motivated by an investigation of the two Girard’s translations. This link is not casual, since it holds even when the bang calculus is extended to a PCF-like system, as shown by Ehrhard \([13]\).

The aim of our paper is to further investigate the way the bang calculus subsumes CbN and CbV \( \lambda \)-calculi, refining and extending some results already obtained in \([14]\).

1. From a syntactic viewpoint, we show in §3 that the bang calculus subsumes in the same rewriting system both CbN and CbV \( \lambda \)-calculi, in the sense that the translations \((\cdot)^{N}\) and \((\cdot)^{V}\) from the \( \lambda \)-calculus to the bang calculus are sound and complete with respect to \( \beta \)-reduction and \( \beta_{v} \)-reduction, respectively (in \([14]\) only soundness was proven, and in a less elegant way). In other words, the diagrams

\[
\begin{align*}
\Lambda \ni t & \xrightarrow{\beta} s \in \Lambda \\
\Lambda \ni t & \xrightarrow{\beta_{v}} s \in \Lambda
\end{align*}
\]

\[
\begin{align*}
!\Lambda \ni t^{n} & \xrightarrow{b} s^{n} \in !\Lambda \\
!\Lambda \ni t^{n} & \xrightarrow{b} s^{n} \in !\Lambda
\end{align*}
\]

commute in the two ways: starting from the \( \beta \)-reduction step \( \rightarrow_{\beta} \) for the CbN \( \lambda \)-calculus (on the left) or the \( \beta_{v} \)-reduction step \( \rightarrow_{\beta_{v}} \) for the CbV \( \lambda \)-calculus (on the right), and starting from the \( b \)-reduction step \( \rightarrow_{b} \) of the bang calculus\(^{2}\).

2. From a semantic viewpoint, we show in §4 that the every LL-based model \( \mathcal{W} \) of the bang calculus (as categorically defined in \([14]\)) provides a model for both CbN and CbV \( \lambda \)-calculi (in \([14]\) this was done only for the special case of relational semantics). Moreover, given a \( \lambda \)-term \( t \), we investigate the relation between its interpretations \( [t]^{n} \) in CbN (resp. \( [t]^{v} \) in CbV) and the interpretation \( [\cdot]^{n} \) of its translation \( \tau^{n} \) (resp. \( \tau^{v} \)) into the bang calculus. We prove that the diagram below on the left (for CbN) commutes, whereas we give a counterexample (in the relational semantics) to the commutation of the diagram below on the right (for CbV). We conjecture that there still exists a relationship in CbV between \( [t]^{v} \) and \( [\tau]^{v} \), but it should be more sophisticated than in CbN.

\[
\begin{align*}
\Lambda \ni t & \xrightarrow{(\cdot)^{n}} [t]^{n} = [\tau]^{n} \in \mathcal{W} \\
\Lambda \ni t & \xrightarrow{(\cdot)^{v}} [t]^{v} = [\tau]^{v} \in \mathcal{W}
\end{align*}
\]

\[
\begin{align*}
t^{n} & \in !\Lambda \\
\tau^{v} & \in !\Lambda
\end{align*}
\]

In order to achieve these results in a clearer and simpler way, we have slightly modified (see §2) the syntax and operational semantics of the bang calculus with respect to its original formulation in \([14]\).

\(^{2}\)Actually, for the CbV \( \lambda \)-calculus the diagram is slightly more complex, as we will see in §3 but the essence does not change.
We define the syntax and operational semantics of the bang calculus such that the argument is a box, i.e. $S \mapsto \rightarrow \ell$ might be a box. The root-step $r$ is only terms that can be erased and duplicated. When the argument of a constructor $\text{der}$ is a box and $\rightarrow \ell \not \rightarrow \rightarrow$, does not reduce under $!$ (but both reduce under contexts). Note that contexts are contexts but the converse fails: $L \rightarrow \rightarrow r \not \rightarrow \rightarrow r$. From confluence it follows that: $\rightarrow \rightarrow$ is reflexive-transitive and symmetric closure of $\rightarrow$. Let $g$ be a ground context, i.e. $\forall x T \in \Lambda$. The set of free variables of a term $T$, denoted by $\text{fv}(T)$, is defined as expected, $\lambda$ being the only binding constructor. All terms are considered up to $\alpha$-conversion. Given $T, S \in \Lambda$ and a variable $x$, $T\{S/x\}$ denotes the term obtained by the capture-avoiding substitution of $S$ (and not $S'$) for each free occurrence of $x$ in $T$: so, $T'\{S/x\} = (T\{S/x\})' \in \Lambda$. Contexts $\mathcal{C}$ and ground contexts $G$ (both with exactly one hole $\langle \rangle$) are defined in Fig. 1. All ground contexts are contexts but the converse fails: $\langle \rangle$ is a non-ground context. We write $\mathcal{C}\{T\}$ for the term obtained by the capture-avoiding substitution of the term $T$ for the hole $\langle \rangle$ in the context $\mathcal{C}$.

Reductions in the bang calculus are defined in Fig. 1 as follows: given a root-step rule $\rightarrow r \in \Lambda \times \Lambda$, we define the $r$-reduction $\rightarrow r$ (resp. $r_g$-reduction $\rightarrow r_g$) as the closure of $\rightarrow r$ under contexts (resp. ground contexts). Note that $\rightarrow r \subseteq \rightarrow r$ as $\Lambda \subseteq \Lambda_g$: the only difference between $\rightarrow r$ and $\rightarrow r_g$ is that the latter does not reduce under $!$ (but both reduce under $\lambda$). The root-steps used in the bang calculus are $\rightarrow \ell$ and $\rightarrow d$ and $\rightarrow b := \rightarrow \ell \cup \rightarrow d$. From the definitions in Fig. 1 it follows that $\rightarrow b = \rightarrow \ell \cup \rightarrow d$ and $\rightarrow b_g = \rightarrow \ell_g \cup \rightarrow d_g$.

Intuitively, the basic idea behind the root-steps $\rightarrow \ell$ and $\rightarrow d$ is that the box-constructor $!$ marks the only terms that can be erased and duplicated. When the argument of a constructor $\text{der}$ is a box $T'$, the root-step $\rightarrow d$ opens the box, i.e. accesses its content $T$, destroying its status of availability at will (but $T'$, in turn, might be a box). The root-step $\rightarrow \ell$ says that a $\beta$-like redex $\langle \lambda x T\rangle S$ can be fired only when its argument is a box, i.e. $S \in \Lambda$: if so, the content $R$ of the box $S$ replaces any free occurrence of $x$ in $T$.

Example 1. Let $\Delta := \lambda x (x)^2$ and $\Delta' := \lambda x (\text{der}(x')^2)^2$. Then, $\Delta \rightarrow d_g \Delta$ and $\langle \Delta \rangle \Delta' \rightarrow d_g \langle \Delta \rangle \Delta'$ and $\langle \text{der}(\Delta') \rangle \Delta' \rightarrow d_g \langle \text{der}(\Delta') \rangle \Delta' \rightarrow d_g \ldots$ and $\langle \text{der}(\Delta') \rangle \Delta' \rightarrow d_g \langle \Delta \rangle \Delta'$ and $\langle \text{der}(\Delta') \rangle \Delta' \rightarrow d_g \ldots$ Note that $\langle \Delta \rangle \Delta'$ is $b_g$-normal but not $b$-normalizable.

The bang (resp. ground bang) calculus is the set $\Lambda$ endowed with the reduction $\rightarrow b$ (resp. $\rightarrow b_g$).
Quasi-strong confluence of $b_g$-reduction and confluence of $b$-reduction. To prove the confluence of $\rightarrow_{b}$ (Prop. 4), first we show that $\rightarrow_{\ell}$ is confluent (Lemma 34). The latter is proved by a standard adaptation of Tait–Martin-Löf technique — as improved by Takahashi [29] — based on parallel reduction. For this purpose, we introduce parallel $\ell$-reduction, denoted by $\Rightarrow_{\ell}$, a binary relation on $!\Lambda$ defined by the rules in Fig. 2. Intuitively, $\Rightarrow_{\ell}$ reduces simultaneously a number of $\ell$-redexes existing in a term. It is immediate to check that $\Rightarrow_{\ell}$ is reflexive and $\Rightarrow_{\ell} \subseteq \Rightarrow_{\ell} \subseteq \Rightarrow_{\ell}^*$, whence $\Rightarrow_{\ell}^* = \Rightarrow_{\ell}^*$.

For any term $T$, we denote by $T^*$ the term obtained by reducing all $\ell$-redexes in $T$ simultaneously. Formally, $T^*$ is defined by induction on $T \in !\Lambda$ as follows:

$$x^* := x \quad (\lambda x T)^* := \lambda x T^* \quad (T')^* := (T^*)^! \quad \text{der}(T)^* := \text{der}(T^*)$$

$$((T)S)^* := (T^*)S^* \text{ if } T \neq \lambda x R \text{ or } S \notin !\Lambda! \quad ((\lambda x T)S)^* := T^*\{S^*/x\}$$

**Lemma 2** (Development). Let $T, S \in !\Lambda$. If $T \Rightarrow_{\ell} S$, then $S \Rightarrow_{\ell} T^*$.

**Proof p. 17**

Lemma 2 is the key ingredient to prove the confluence of $\rightarrow_{\ell}$ (Lemma 34 below).

The next lemma lists a series of good rewriting properties of $\ell^*, \ell_g^*$, $d^*$ and $d_g^*$-reductions that will be used to prove quasi-strong confluence of $\rightarrow_{b_g}$ and confluence of $\rightarrow_{b}$ (Prop. 4 below).

**Lemma 3** (Basic properties of reductions).

1. $\rightarrow_{\ell_g}$ is quasi-strongly confluent: $\ell_g \cdot \cdot \cdot \rightarrow_{\ell_g} \subseteq (\rightarrow_{\ell_g} \cdot \ell_g \cdot \cdot \cdot)$.
2. $\rightarrow_{d_g}$ and $\rightarrow_{d}$ are quasi-strongly confluent (separately).
3. $\rightarrow_{d_g}$ and $\rightarrow_{\ell_g}$ strongly commute: $d_g \cdot \cdot \cdot \rightarrow_{\ell_g} \subseteq (\rightarrow_{\ell_g} \cdot d_g \cdot \cdot \cdot)$.

$\rightarrow_{d}$ quasi-strongly commutes over $\rightarrow_{\ell}$: $d \cdot \cdot \cdot \rightarrow_{\ell} \subseteq \rightarrow_{\ell} \cdot d \cdot \cdot \cdot$.

$\rightarrow_{d}$ and $\rightarrow_{\ell}$ commute: $\ell \cdot \cdot \cdot \rightarrow_{\ell} \subseteq \rightarrow_{\ell} \cdot \ell \cdot \cdot \cdot$.

4. $\rightarrow_{\ell}$ is confluent: $\ell \cdot \cdot \cdot \rightarrow_{\ell}^* \subseteq \rightarrow_{\ell}^* \cdot \ell \cdot \cdot \cdot$.

**Proposition 4** (Quasi-strong confluence of $\rightarrow_{b_g}$ and confluence of $\rightarrow_{b}$).

1. The reduction $\rightarrow_{b_g}$ is quasi-strongly confluent, i.e. $b_g \cdot \cdot \cdot \rightarrow_{b_g} \subseteq (\rightarrow_{b_g} \cdot b_g \cdot \cdot \cdot) \cup =$.
2. The reduction $\rightarrow_{b}$ is confluent: $b \cdot \cdot \cdot \rightarrow_{b} \subseteq \rightarrow_{b}^* \cdot b \cdot \cdot \cdot$.

3 The bang calculus with respect to CbN and CbV $\lambda$-calculi, syntactically

One of the interests of the bang calculus is that it is a general framework where both call-by-name (CbN i.e. ordinary, [4]) and Plotkin’s call-by-value (CbV, [25]) $\lambda$-calculi can be embedded. Syntax and reduction rules of CbN and CbV $\lambda$-calculi are in Fig. 3. $\beta$ reduction $\rightarrow_{\beta}$ (resp. $\beta^v$-reduction $\rightarrow_{\beta^v}$) is the reduction for the CbN (resp. CbV) $\lambda$-calculus. CbN and CbV $\lambda$-calculi share the same term syntax (the set $\Lambda$ of $\lambda$-terms of CbV and CbN $\lambda$-calculus can be seen as a subset of $!\Lambda$), whereas $\rightarrow_{b}$ is just the restriction of $\rightarrow_{\beta}$ allowing to fire a $\beta$-redex $(\lambda x T)S$ only when $S$ is a $\lambda$-value, i.e. variable or abstraction. Ground $\beta$-(resp. $\beta^v$-)reduction $\rightarrow_{\beta_g}$ (resp. $\rightarrow_{\beta_g^v}$) is an interesting restriction of $\beta$-(resp. $\beta^v$-)reduction:

3Here, with CbN or CbV $\lambda$-calculus we refer to the whole calculus and its general reduction rules, not only to CbN or CbV (deterministic) evaluation strategy in the $\lambda$-calculus.
\[
\begin{align*}
\text{\(\lambda\)-terms:} & \quad t,s,r \ ::= \ v \mid ts & \text{(set: } \Lambda) \\
\text{\(\lambda\)-values:} & \quad v \ ::= x \mid \lambda x t & \text{(set: } \Lambda_v) \\
\text{\(\lambda\)-contexts:} & \quad C \ ::= \ \langle \emptyset \rangle \mid \lambda x C \mid C t \mid tC & \text{(set: } \Lambda_C) \\
\text{CbN ground \(\lambda\)-contexts:} & \quad N \ ::= \ \langle \emptyset \rangle \mid \lambda x N \mid N t & \text{(set: } \Lambda_R) \\
\text{CbV ground \(\lambda\)-contexts:} & \quad V \ ::= \ \langle \emptyset \rangle \mid \forall t \mid V & \text{(set: } \Lambda_V) \\
\text{Root-steps:} & \quad (\lambda x t)s \ \rightarrow \beta \ t \{s/x\} \ (\text{CbN}) & \quad (\lambda x t)v \ \rightarrow \beta \ t \{v/x\} \ (\text{CbV}) \\
\text{\(r\)-reduction:} & \quad t \ \rightarrow_r \ s & \iff \ \exists C \in \Lambda_C, \ \exists t',s' \in \Lambda : t = C[t'], s = C[s'], t' \ \rightarrow_r \ s' \\
\beta_g\text{-reduction:} & \quad t \ \rightarrow_{\beta_g} \ s & \iff \ \exists N \in \Lambda_R, \ \exists t',s' \in \Lambda : t = N[t'], s = N[s'], t' \ \rightarrow_{\beta} \ s' \\
\beta_v\text{-reduction:} & \quad t \ \rightarrow_{\beta_v} \ s & \iff \ \exists V \in \Lambda_V, \ \exists t',s' \in \Lambda : t = V[t'], s = V[s'], t' \ \rightarrow_{\beta_v} \ s'
\end{align*}
\]

Figure 3: The CbN and CbV \(\lambda\)-calculi: their syntax and reduction rules, where \(r \in \{\beta, \beta_v\}\).

- \(\rightarrow_{\beta_g}\) contains head \(\beta\)-reduction and weak head \(\beta\)-reduction, two well-known evaluation strategies for CbN \(\lambda\)-calculus (both reduce the \(\beta\)-redex in head position, the latter does not reduce under \(\lambda\)’s);

- \(\rightarrow_{\beta_v}\) contains (weak) head \(\beta_v\)-reduction (aka left reduction in [25, p. 136]), the well-known evaluation strategy for CbV \(\lambda\)-calculus firing the leftmost-outermost \(\beta_v\)-redex (if any) not under \(\lambda\)’s.

\textbf{CbN and CbV translations into the bang calculus.} \textit{CbN and CbV translations} are two functions 
\((\cdot)^n : \Lambda \rightarrow !\Lambda\) and 
\((\cdot)^v : \Lambda \rightarrow !\Lambda\), respectively, translating \(\lambda\)-terms into terms of the bang calculus:

\[
\begin{align*}
(x^n)^0 & \ := x & (\lambda x t)^n & \ := \lambda x t^n & (ts)^n & \ := \langle t^n \rangle s^{n!} \\
(x^v)^! & \ := x^! & (\lambda x t)^v & \ := (\lambda x t^v)^! & (ts)^v & \ := \langle \text{der} \ t^v \rangle s^v.
\end{align*}
\]

\textit{Example 5.} Let \(\Omega := (\lambda.xxx)\lambda.xxx\), the typical diverging \(\lambda\)-term for CbN and CbV \(\lambda\)-calculi: one has 
\(\Omega^n = (\langle \Delta \rangle \Delta^! \ \Delta^! \Delta)\), and \(\Omega^v = (\langle \text{der}(\Delta^!) \rangle \Delta^! \ \Delta^! \Delta)\), which are not \(b_{g}\)- nor \(b\)-normalizable (\(\Delta\) and \(\Delta^!\) are defined in Ex.1).

For any \(\lambda\)-term \(t\), \(t^n\) and \(t^v\) are just different decorations of \(t\) by means of the monadic operators \(!\) and \(\text{der}\) (the latter does not occur in \(t^n\)). Note that the translation \((\cdot)^n\) puts the argument of any application into a box: in CbN \(\lambda\)-calculus any \(\lambda\)-term is duplicable or discardable. On the other hand, only \(\lambda\)-values \((i.e.\) abstractions and variables\)) are translated by \((\cdot)^v\) into boxes, as they are the only \(\lambda\)-terms duplicable or discardable in CbV \(\lambda\)-calculus.

\textit{Lemma 6 (Substitution).} Let \(t, s\) be \(\lambda\)-terms and \(x\) be a variable.

1. One has that \(t^n \{s^n/x\} = (t \{s/x\})^n\).

2. If \(s\) is such that \(s^v = R^!\) for some \(R \in !\Lambda\), then \(t^v \{R/x\} = (t \{s/x\})^v\).

\textit{Proof.} The proofs of both points are by induction on the \(\lambda\)-term \(t\).

1. \textit{Variable:} If \(t\) is a variable then there are two subcases. If \(t := x\) then \(t^n = x\), so \(t^n \{s^n/x\} = s^n = (t \{s/x\})^n\). Otherwise \(t := y \neq x\) and then \(t^n = y\), hence \(t^n \{r/x\} = y = (t \{s/x\})^n\).

2. \textit{Abstraction:} If \(t := \lambda y t\) then \(t^n = \lambda y t^n\) and we can supposes without loss of generality that \(y \notin \text{fv}(s) \cup \{x\}\). By i.h., \(r^n \{s^n/x\} = (r \{s/x\})^n\) and hence \(t^n \{s^n/x\} = \lambda y (r^n \{s^n/x\}) = \lambda y (r \{s/x\})^n = (t \{s/x\})^n\).
• **Application:** If \( t := RQ \), then \( t^n = (r^n)q^n \). By i.h., \( r^n(s^n/x) = (r(s/x))^n \) and \( q^n(s^n/x) = (q(s/x))^n \). So, \( t^n(s^n/x) = (r^n(s^n/x))(q^n(s^n/x))^1 = ((r(s/x))^n)((q(s/x))^n)^1 = (t(s/x))^n \).

2. **Variable:** If \( t \) is a variable then there are two subcases. If \( t := x \) then \( t^\nu = x^\nu \), so \( t^\nu(R/x) = R^\nu = s^\nu = (t(s/x))^\nu \). Otherwise \( t := y \neq x \) and then \( t^\nu = y^\nu \), hence \( t^\nu(R/x) = y^\nu = (t(s/x))^\nu \).

• **Application:** If \( t := pq \) then \( t^\nu = (\text{der}p^\nu)q^\nu \). By i.h., \( p^\nu(R/x) = (p(s/x))^\nu \) and \( q^\nu(R/x) = (q(s/x))^\nu \). So, \( t^\nu(R/x) = (\text{der}(p^\nu(R/x)))q^\nu(R/x) = (\text{der}(p(s/x))^\nu)(q(s/x))^\nu = (t(s/x))^\nu \).

• **Abstraction:** If \( t := (\lambda y q)^\nu \) and we can suppose without loss of generality that \( y \notin \text{fv}(s) \cup \{x\} \). By i.h., \( q^\nu(R/x) = (q(s/x))^\nu \), and thus \( t^\nu(R/x) = (\lambda y (q^\nu(R/x))^1 = (\lambda y (q(s/x))^\nu)^1 = (t(s/x))^\nu \).

Note that the hypothesis about \( s \) in Lemma 6 is fulfilled if and only if \( s \) is a \( \lambda \)-value.

**Remark 7** (CbV translation is \( \ell \)-normal). It is immediate to prove by induction on \( t \in \Lambda \) that \( t^\nu \) is \( \ell \)-normal, so if \( t^\nu \to_d S_0 \to \gamma S \) then the only \( \ell \)-redex in \( S_0 \) has been created by the step \( t^\nu \to_d S_0 \) and is absent in \( t^\nu \).

### Simulating CbN and CbV reductions into the bang calculus

We can now show that the CbN translation \((\cdot)^n\) (resp. CbV translation \((\cdot)^\nu\)) from the CbN (resp. CbV) \( \lambda \)-calculus into the bang calculus is *sound* and *complete*: said differently, the target of the CbN (resp. CbV) translation into the bang calculus is a conservative extension of the CbN (resp. CbV) \( \lambda \)-calculus.

**Theorem 8** (Simulation of CbN and CbV \( \lambda \)-calculi). \( t \) be a \( \lambda \)-term.

1. **Conservative extension of CbN \( \lambda \)-calculus:**
   - **Soundness:** If \( t \to^\beta t' \) then \( t^n \to^\ell t'^n \) (and \( t^n \to_b t'^n \));
   - **Completeness:** Conversely, if \( t^n \to^\ell S \) then \( t^n \to^\beta S = t'^n \) for some \( \lambda \)-term \( t' \).

2. **Conservative extension of ground CbN \( \lambda \)-calculus:**
   - **Soundness:** If \( t \to^\beta t' \) then \( t^n \to^\ell t'^n \) (and \( t^n \to_b t'^n \));
   - **Completeness:** Conversely, if \( t^n \to_b S \) then \( t^n \to^\ell S = t'^n \) and \( t \to^\beta t' \) for some \( \lambda \)-term \( t' \).

3. **Conservative extension of CbV \( \lambda \)-calculus:**
   - **Soundness:** If \( t \to^\beta t' \) then \( t^\nu \to^\ell t'^\nu \) (and hence \( t^\nu \to_b t'^\nu \));
   - **Completeness:** Conversely, if \( t^\nu \to^\ell S \) then \( S = t'^\nu \) and \( t \to^\beta t' \) for some \( \lambda \)-term \( t' \).

4. **Conservative extension of ground CbV \( \lambda \)-calculus:**
   - **Soundness:** If \( t \to^\beta t' \) then \( t^\nu \to_b t'^\nu \) (and hence \( t^\nu \to_b t'^\nu \));
   - **Completeness:** Conversely, if \( t^\nu \to_b S \) then \( S = t'^\nu \) and \( t \to^\beta t' \) for some \( \lambda \)-term \( t' \).

**Proof.** 1. **Soundness:** We prove by induction on the \( \lambda \)-term \( t \) that if \( t \to^\beta t' \) then \( t^n \to^\ell t'^n \). By definition of \( t \to^\beta t' \), there are the following cases:

   - **Root-step, i.e.,** \( t := (\lambda x r)s \Rightarrow t' \): by Lemma 6, \( t^n = (\lambda x r^n)s^n \to^\ell t'^n(s^n/x) = t'^n \).
   - **Abstraction, i.e.,** \( t := \lambda x r \Rightarrow \lambda x r' \): by i.h., \( r^n \to^\ell r'^n \), thus \( t^n = \lambda x r^n \to^\ell \lambda x r'^n = t'^n \).
   - **Application left, i.e.,** \( t := rs \Rightarrow r's = t' \): by i.h., \( r^n \to^\ell r'^n \), since \( r^n \) is \( d \)-normal. We prove by induction on \( \lambda \)-term \( t \) that if \( t^n \to^\ell S \) then \( S = t'^n \) and \( t \to^\beta t' \) for some \( \lambda \)-term \( t' \).
   - **Application right, i.e.,** \( t := sr \Rightarrow sr' = t' \): analogous to the previous case.

   **Completeness:** First, observe that \( t^n \to_b s \) entails \( t^n \to^\ell s \) since \( \text{der} \) does not occur in \( t^n \), hence \( t^n \) is \( d \)-normal. We prove by induction on \( \lambda \)-term \( t \) that if \( t^n \to^\ell S \) then \( S = t'^n \) and \( t \to^\beta t' \) for some \( \lambda \)-term \( t' \). According to the definition of \( t^n \to^\ell S \), there are the following cases:
• **Root-step**, i.e. \( t^n := (\lambda x r^n) q^n \mapsto_\ell r^n \{ q^n / x \} =: S \): by Lemma 4 \[ S = (r\{q/x\})^n, \] so \( t = (\lambda x r) q \mapsto_\beta r \{q/x\} =: t' \) where \( t'^n = S \).

• **Abstraction**, i.e. \( t^n := \lambda x r^n \mapsto_\ell \lambda x S' =: S \) with \( r^n \mapsto_\ell S' \): by i.h., there is a \( \lambda \)-term \( r' \) such that \( r'^n = S' \) and \( r \mapsto_\beta r' \), thus \( t = \lambda x r \mapsto_\beta \lambda x r' =: t' \) where \( t'^n = \lambda x r'^n = S \).

• **Application left**, i.e. \( t^n := \langle r^n \rangle d^n \mapsto_\ell \langle S' \rangle q^n =: S \) with \( r^n \mapsto_\ell S' \): analogously to above.

• **Application right**, i.e. \( t^n := \langle q^n \rangle r^n \mapsto_\ell \langle q^n \rangle S^n =: S \) with \( r^n \mapsto_\ell S' \): by i.h., there is a \( \lambda \)-term \( r' \) such that \( r'^n = S' \) and \( r \mapsto_\beta r' \), so \( t = qr \mapsto_\beta qr' =: t' \) with \( t'^n = \langle q^n \rangle r'^n = S \).

2. Since \( \mapsto_\beta \) is simulated by \( \mapsto_\ell \) and vice-versa (see the root-cases above), Thm. 3 is proved analogously to the proof of Thm. 4. \( \mapsto_\ell \) replaces \( \mapsto_\ell \), and \( \mapsto_\beta \) replaces \( \mapsto_\beta \), with the difference that, by definition, \( \mapsto_\beta \) and \( \mapsto_\ell \) do not give rise to the case **Application right**.

3. **Soundness**: We prove by induction on the \( \lambda \)-term \( t \) that if \( t \mapsto_\beta t' \) then \( t'^{\ell} \mapsto_\ell t'^{\ell} \). According to the definition of \( T \mapsto_\beta T' \), there are the following cases:

• **Root-step**, i.e. \( t := (\lambda x r) v \mapsto_\beta r \{ v / x \} =: t' \) where \( v \) is a \( \lambda \)-value, i.e. a variable or an abstraction: then \( v'^n = S' \) for some \( S' \in !\Lambda \), hence \( t'^n = (\langle \lambda x r' \rangle)^n \mapsto_\ell r^n \{ S/x \} = t'^n \) by Lemma 4 [recall that \( \mapsto_\ell \subseteq \mapsto_d \)].

• **Abstraction**, i.e. \( t := \lambda x r \mapsto_\beta \lambda x r' =: t' \) with \( r \mapsto_\beta r' \): by i.h., \( r'^n \mapsto_\ell r'^n \), therefore \( t'^n = (\langle \lambda x r' \rangle)^n \mapsto_\ell (\lambda x r'^n)^n = r'^n \).

• **Application left**, i.e. \( t := rs \mapsto_\beta r' s =: t' \) with \( r \mapsto_\beta r' \): by i.h. \( r'^{\ell} = r'^n \mapsto_\ell r'^n \); therefore \( t'^n = (\langle r'^n \rangle s^n)^n \) such that \( \langle r'^n \rangle s^n = T'^n \).

• **Application right**, i.e. \( t := sr \mapsto_\beta s' r' =: t' \) with \( r \mapsto_\beta r' \): analogous to the previous case.

**Completeness**: We prove by induction on \( S_0 \in !\Lambda \) that if \( t'^{\ell} \mapsto_d S_0 \mapsto_\ell S \) then \( S = t'^{\ell} \) and \( t \mapsto_\beta t' \) for some \( \lambda \)-term \( t' \). According to the definition of \( S_0 \mapsto_\ell S \), there are the following cases:

• **Root-step**, i.e. \( S_0 := \langle \lambda x R \rangle Q \mapsto_\ell R \{ Q / x \} =: S \). According to Rmk. 7, necessarily \( t'^n = \langle \langle \lambda x R \rangle \rangle Q'^n \mapsto_\ell r^n \{ Q/x \} =: S \) such that \( r'^n = R \) and \( v'^n = Q' \). Notice that \( t'^n \mapsto_\ell S_0 \). Let \( t' := r_0 \{ v/x \} \): then, \( t \mapsto_\beta t' \) and \( t'^n = r_0^n \{ Q/x \} = S \) according to Lemma 4 [recall that \( \mapsto_\ell \subseteq \mapsto_d \)].

• **Abstraction**, i.e. \( S_0 := \lambda x R_0 \mapsto_\ell \lambda x R' =: S \) with \( R_0 \mapsto_\ell R' \). This case is impossible because, according to Rmk. 7, necessarily \( t'^n = \lambda x R' \) for some \( \ell \)-normal \( R' \in !\Lambda \) such that \( R \mapsto_d R_0 \), but there is no \( \lambda \)-term \( t \) such that \( t'^n \) is an abstraction.

• **Derelection**, i.e. \( S_0 := \text{der} R_0 \mapsto_\ell \text{der} R' =: S \) with \( R_0 \mapsto_\ell R' \). This case is impossible because, according to Rmk. 7, necessarily \( t'^n = \text{der} R' \) for some \( \ell \)-normal \( R' \in !\Lambda \) such that \( R \mapsto_d R_0 \), but there is no \( \lambda \)-term \( t \) such that \( t'^n \) is a dereliction.

• **Application left**, i.e. \( S_0 := \langle R_0 \rangle R_1 \mapsto_\ell \langle R_1 \rangle R' =: S \) with \( R_0 \mapsto_\ell R' \). By Rmk. 7, necessarily \( t'^n = \langle \langle \text{der} P_1 \rangle R_1 \) for some \( \ell \)-normal \( P_1 \in !\Lambda \) such that \( \text{der} P \mapsto_d R_0 \), and thus \( R_0 = \text{der} P_0 \) where \( P \mapsto_d P_0 \) (indeed \( P = R_0 \) is impossible because \( P \) is \( \ell \)-normal), and hence \( R' = \text{der} P' \) with \( P_0 \mapsto_\ell P' \). So, \( t = qq_1 \) for some \( \lambda \)-terms \( q \) and \( q_1 \) such that \( q'^n = P \) and \( q^n_1 = R_1 \). Notice that \( \text{der} P = q'^n \mapsto_\ell R_1 \mapsto_\ell P' \). By i.h., \( P' = q'^n \) and \( q \mapsto_\beta q'^n \) for some \( \lambda \)-term \( q' \). Let \( t' := q q_1' \): then, \( t \mapsto_\beta t' \) and \( t'^n = \langle \langle \text{der} P_1 \rangle \rangle q_1'^n \).

• **Application right**, i.e. \( S_0 := \langle R_1 \rangle R_0 \mapsto_\ell \langle R_1 \rangle R' =: S \) with \( R_0 \mapsto_\ell R' \). By Rmk. 7, necessarily \( t'^n = \langle \langle \text{der} P_1 \rangle R \) for some \( \ell \)-normal \( R, P_1 \in !\Lambda \) such that \( \text{der} P_1 = R_1 \) and \( R \mapsto_d R_0 \). So, \( t = q_1 Q \) for some \( \lambda \)-terms \( q_1 \) and \( q' \) such that \( q_1'^n = P_1 \) and \( q'^n = R \). By i.h., \( R' = q'^n \) and \( q \mapsto_\beta q'^n \) for some \( \lambda \)-term \( q' \). Let \( t' := q_1 q' \): then, \( t \mapsto_\beta t' \) and \( t'^n = \langle \langle q_1 q' \rangle \rangle q'^n \).
target of CbN translation into !Λ: \( T,S ::= x \mid (T)S \mid \lambda x T \) (set: \( !\Lambda^n \))

**target of CbV translation into !Λ:** \( M,N ::= U \mid (derM)N \mid (U)M \) (set: \( !\Lambda^\nu \)) \( U ::= x \mid \lambda x M \) (set: \( !\Lambda^\nu \)).

Figure 4: Targets of CbN and CbV translations into the bang calculus.

- **Box, i.e.** \( S_0 := R_0 \rightarrow_\ell R'' := S \) with \( R_0 \rightarrow_\ell R' \). According to Rmk.7, necessarily \( t^\nu = R \) for some \( \ell \)-normal \( R \in !\Lambda \) such that \( R \rightarrow_\ell R_0 \). So, \( t = \lambda x q' \) (since \( t^\nu \) is a box and \( x^\nu \) is \( d \)-normal) for some \( \ell \)-term \( q \) such that \( R = \lambda x q^\nu \), and hence there are \( R_0, P' \in !\Lambda \) such that \( R_0 = \lambda x P_0 \) and \( R' = \lambda x R'' \) with \( q' \rightarrow_\ell P_0 \rightarrow_\ell P' \). By i.h., \( P' = q'^\nu \) and \( q \rightarrow_\beta q' \) for some \( \ell \)-term \( q' \). Let \( t' ::= \lambda x q' \); then, \( t = \lambda x q \rightarrow_\beta t' \) and \( t'^\nu = (\lambda x q'^\nu) = S \).

4. Since \( \rightarrow_\ell \) is simulated by \( \rightarrow_\ell \rightarrow_\ell \) and vice-versa (see the root-cases above), Thm.8.4 is proved analogously to the proof of Thm.8.3 (replace \( \rightarrow_\ell \) with \( \rightarrow_\ell \), and \( \rightarrow d \) with \( \rightarrow_{d_\ell} \), as well as \( \rightarrow_\beta \), with \( \rightarrow_{d_\ell} \)), with the difference that \( \rightarrow_{d_\ell} \) does not give rise to the case Abstraction (in the soundness proof) and Box (in the completeness proof).

So, the bang calculus can simulate \( \beta \)- and \( \beta^\nu \)-reductions via \( (\cdot)^n \) and \( (\cdot)^\nu \) and, conversely, \( \ell \)-reductions in the targets of \( (\cdot)^n \) and \( (\cdot)^\nu \) correspond to \( \beta \)- and \( \beta^\nu \)-reductions. Also, these simulations are:

- **modular, in the sense that** ground \( \beta \)-reduction (including head \( \beta \)-reduction and weak head \( \beta \)-reduction) is simulated by ground \( \ell \)-reduction, and vice-versa (Thm.8.2): ground \( \beta^\nu \)-reduction (including head \( \beta^\nu \)-reduction) is simulated by ground \( d \) and \( \ell \)-reductions, and vice-versa (Thm.8.3).

- **quantitative sensitive, meaning that one step of (ground) \( \beta \)-reduction corresponds exactly, via \( (\cdot)^n \), to one step of (ground) \( \ell \)-reduction, and vice-versa; one step of (ground) \( \beta^\nu \)-reduction corresponds exactly, via \( (\cdot)^\nu \), to one step of (ground) \( \ell \)-reduction, and vice-versa.

According to the definition of CbN translation \((\cdot)^n\), the target of \((\cdot)^n\) into the bang calculus can be characterized syntactically: it is the subset \(!\Lambda^n\) of \(!\Lambda\) defined in Fig.4. This means that \( t^\nu \in !\Lambda^n \) for any \( t \in \Lambda \) and conversely, for any \( T \in !\Lambda^n \), \( T^n = t \) for some \( t \in \Lambda \). Note that in \(!\Lambda^n\) the constructor \( der \) does not occur and hence reductions \( \rightarrow_\ell \) and \( \rightarrow_{d_\ell} \) coincide with \( \rightarrow_b \) and \( \rightarrow_{d_b} \), respectively, in \(!\Lambda^n\).

So, Thm.8.12 says that \(!\Lambda^n\) endowed with the reduction \( \rightarrow_\ell \) (resp. \( \rightarrow_{d_\ell} \)) which coincides with \( \rightarrow_b \) (resp. \( \rightarrow_{d_b} \)) in \(!\Lambda^n\) — is isomorphic to CbN (resp. ground CbN) \( \lambda \)-calculus. In particular:

**Corollary 9** (Preservations with respect to CbN \( \lambda \)-calculus). Let \( t,s \in \Lambda \).

1. **CbN equational theory:** \( t \simeq_\beta s \iff t^n \simeq_\ell s^n \iff t^n \simeq_b s^n \).

2. **CbN normal forms:** \( t \) is (ground) \( \beta \)-normal iff \( t^n \) is (ground) \( \ell \)-normal iff \( t^n \) is (ground) \( b \)-normal.

**Proof.**

1. If \( t \simeq_\beta s \) then \( t \rightarrow_b r^* \leftarrow_s \) for some \( r \in \Lambda \), as \( \rightarrow_b \) is confluent. By Thm.8.11 (soundness), \( t^n \rightarrow_\ell r^n^* \leftarrow_s \) so, \( t^n \simeq_\ell s^n \) and \( r^n \simeq_\ell s^n \) since \( \rightarrow_\ell \subseteq \rightarrow_b \).

Conversely, if \( t^n \simeq_b s^n \) then \( t^n \rightarrow_{d_b} R^*_b \leftarrow_{s^n} \) for some \( R \in !\Lambda \), since \( \rightarrow_b \) is confluent (Prop.4.12). By Thm.8.11 (completeness), \( t \rightarrow_{d_b} q^* \leftarrow_s \) for some \( \lambda \)-term \( q \) such that \( q^n = R \), and therefore \( t \simeq_\beta s \).

2. Immediate consequence of Thm.8.112.

The correspondence between CbV \( \lambda \)-calculus and bang calculus is slightly more delicate: CbV translation \((\cdot)^\nu\) gives a sound and complete embedding of \( \rightarrow_{d \rightarrow_\ell} \) into \( \rightarrow_d \rightarrow_\ell \) (and similarly for their ground variants), but it is not complete with respect to generic \( \rightarrow_b \). Indeed, Ex.1\(^7\) and Ex.5\(^8\) have shown that
\((\lambda x x x) \mapsto \Delta \rightarrow \Delta', \) where \(\Delta'\) is b-normal and there is no \(\lambda\)-term \(t\) such that \(t' = \Delta\). Note that \(\lambda x x x\) is \(\beta\)-normal but \((\lambda x x x) \mapsto \Delta'\) is not b-normal: in CbV the analogous of Cor. 11.2 does not hold for \((\cdot)\mapsto\).

Actually, an analogous of Cor. 9.1 for CbV holds: CbV translation preserves \(\beta\)-equivalence in a sound and complete way with respect to \(b\)-equivalence (see Cor. 12 below). The proof requires a fine analysis of CbV translation \((\cdot)\mapsto\). First, we define two subsets \(!\Lambda^v\) and \(!\Lambda^v\) of \!\(\Lambda\), see Fig. 4.

**Remark 10** (Image of CbV translation). If \(t \in \Lambda\) then \(t' \in !\Lambda^v\) (proof by induction on \(t \in \Lambda\)). In particular, for any \(v \in \Lambda_n, v' = U^\dagger\) for some \(U \in !\Lambda_n^v\). Note that \(\Delta \in !\Lambda^v\) but there is no \(\lambda\)-term \(t\) such that \(t' = \Delta\).

We have just shown that \((\cdot)\mapsto\) is not surjective in \!\(\Lambda^v\). Anyway, it can be shown (Lemma 11.3) that \(!\Lambda^v\) is the set of terms in \!\(\Lambda\) reachable by \(b\)-reduction from CbV translations of \(\lambda\)-terms (i.e. for any \(t \in \Lambda\), if \(t' \xrightarrow{b} S\) then \(S \in !\Lambda^v\)) and \(b\)-equivalence in \!\(\Lambda^v\) preserves \(\beta\)-equivalence (Cor. 12). To prove that, we define a forgetful translation \((\cdot)^\dagger: !\Lambda^v \cup !\Lambda_n^v \rightarrow \Lambda\) transforming terms \(M \in !\Lambda^v\) and \(U \in !\Lambda_n^v\) into \(\lambda\)-terms:

\[
(\langle U \rangle)^\dagger := U^\dagger; \quad (\langle \text{der}(M)N \rangle)^\dagger := M^\dagger N^\dagger; \quad (\langle \text{U} \rangle)^\dagger := U^\dagger; \quad x^\dagger := x; \quad (\lambda x M)^\dagger := \lambda x M^\dagger.
\]

**Lemma 11** (Properties of the forgetful translation \((\cdot)^\dagger\)).

1. \((\cdot)^\dagger\) is a right-inverse of \((\cdot)\mapsto\): For every \(t \in \Lambda\), one has \(t^\dagger \mapsto t\).
2. Substitution: \(M\{U/x\} \in !\Lambda^v\) with \((M\{U/x\})^\dagger = M^\dagger\{U^\dagger/x\}\), for any \(M \in !\Lambda^v\) and \(U \in !\Lambda_n^v\).
3. \(b\)-reduction vs. \(\beta\)-reduction: For any \(M \in !\Lambda^v\) and \(T \in !\Lambda\), if \(M \xrightarrow{b} T\) then \(T \in !\Lambda^v\) and \(M^\dagger \xrightarrow{b^\dagger} T^\dagger\).

**Rmk. 10** and Lemma 11.3 mean that \(!\Lambda^v\) is the set of terms in \!\(\Lambda\) reachable by \(b\)-reduction from CbV translations of \(\lambda\)-terms (i.e. for any \(t \in \Lambda\), if \(t' \xrightarrow{\beta^\dagger} S\) then \(S \in !\Lambda^v\)). We can conclude:

**Corollary 12** (Preservation of CbV equational theory). Let \(t, s \in \Lambda\). One has \(t \simeq_{\beta^\dagger} s\) iff \(t^\dagger \simeq_{b^\dagger} s^\dagger\).

**Proof.** If \(t \simeq_{\beta^\dagger} s\) then \(t \xrightarrow{\beta^\dagger} b^\dagger r_{b^\dagger} b^\dagger s\) for some \(r \in \Lambda\), as \(\rightarrow_{b^\dagger}\) is confluent; by Thm. 3.3, \(t^\dagger \xrightarrow{\beta^\dagger} b^\dagger r^\dagger b^\dagger s^\dagger\), since \(\rightarrow_{\beta} \subseteq \rightarrow_{b^\dagger}\) and \(\rightarrow_{d} \subseteq \rightarrow_{b} \); therefore, \(t^\dagger \simeq_{b^\dagger} s^\dagger\). Conversely, if \(t^\dagger \simeq_{b^\dagger} s^\dagger\) then \(t^\dagger \xrightarrow{\beta\dagger} b^\dagger R_{\beta^\dagger} R_{b^\dagger} s^\dagger\) for some \(R \in !\Lambda\), since \(\rightarrow_{b} \) is confluent (Prop. 3.2); by Rmk. 10, \(t^\dagger, s^\dagger \in !\Lambda^v\); so, \(R \in !\Lambda^v\) and \(T^\dagger \xrightarrow{b^\dagger} R^\dagger R^\dagger b^\dagger s^\dagger\) by Lemma 11.3; hence \(t = t^\dagger \simeq_{b^\dagger} s^\dagger = s\) by Lemma 11.1.

So, Cor. 12 says that CbV translation \((\cdot)\mapsto\) — even if it is a sound but not complete embedding of \(\beta\)-reduction into \(b\)-reduction — is a sound and complete embedding of \(\beta\)-equivalence into \(b\)-equivalence.

**4 The bang calculus with respect to CbN and CbV \(\lambda\)-calculi, semantically**

The denotational models of the bang calculus we are interested in this paper are those induced by a denotational model of LL. We recall the basic definitions and notations, see [21][13][14] for more details.
Linear logic based denotational semantics of bang calculus. A denotational model of LL is given by:

- A *-autonomous category, namely a symmetric monoidal closed category \( (\mathcal{L}, \otimes, 1, \lambda, \rho, \alpha, \sigma) \) with a dualizing object \( \bot \). We use \( X \to Y \) for the linear exponential object, \( ev \in \mathcal{L}'((X \to Y) \otimes X, Y) \) for the evaluation morphism and \( \text{cur} \) for the linear currying map \( \mathcal{L}'(Z \otimes X, Y) \to \mathcal{L}'(Z, X \to Y) \). We use \( X^\perp \) for the object \( X \to \bot \) of \( \mathcal{L} \) (the linear negation of \( X \)). This operation \( (-)^\perp \) is a functor \( \mathcal{L}^{\text{op}} \to \mathcal{L} \). The category \( \mathcal{L} \) is cartesian with terminal object \( \top \), product \&, projections \( \text{pr}_i \). By *-autonomy, \( \mathcal{L} \) is cocartesian with initial object 0, coproduct \( \oplus \) and injections \( \iota_j \).

- A functor \( !: \mathcal{L} \to \mathcal{L} \) which:
  - a comonad with counit \( \text{der}_X \in \mathcal{L}(!X, X) \) (dereliction) and comultiplication \( \text{dig}_X \in \mathcal{L}(!X, !!X) \) (digging), and
  - a strong symmetric monoidal functor—-with Seely isos \( m^0 \in \mathcal{L}(1, !\top) \) and \( m^2_X \in \mathcal{L}(!X \otimes !Y, !(X \& Y)) \)—from the symmetric monoidal category \( (\mathcal{L}, \& , 1) \) to the symmetric monoidal category \( (\mathcal{L}, \otimes, 1) \), satisfying an additional coherence condition with respect to \( \text{dig} \).

In order that \( \mathcal{L} \) is also a denotational model of the bang calculus we need a further assumption:

We assume that the unique morphism in \( \mathcal{L}(0, \top) \) is an iso (to simplify, assume just \( 0 = \top \)).

(1)

From (1) it follows that for any two objects \( X \) and \( Y \) there is a morphism \( 0_{X,Y} := \iota \in \mathcal{L}(X, Y) \) where \( \iota \) is the unique morphism \( X \to \top \) and \( \iota \) is the unique morphism \( 0 \to Y \). It turns out that this specific zero morphism satisfies the identities \( f 0_{X,Y} = 0_{X,Z} = 0_{Y,Z} g \) for all \( f \in \mathcal{L}(Y, Z) \) and \( g \in \mathcal{L}(X, Y) \). Assumption (1) is satisfied by many models of LL, like relational model [6], finiteness spaces [10], Scott model [12], (hyper-)coherence [15,19] and probabilistic coherence spaces [8], all models based on Indexed LL [6].

A model of the bang calculus is any object \( \mathcal{U} \) of \( \mathcal{L} \) satisfying the identity \( \mathcal{U} \cong !\mathcal{U} \& (!\mathcal{U} \to \mathcal{U}) \) (we assume this iso to be an equality). Note that this entails both \( !\mathcal{U} \subset \mathcal{U} \) and \( !\mathcal{U} \to \mathcal{U} \subset \mathcal{U} \).

Given a term \( T \) and a repetition-free list of variables \( \bar{x} = (x_1, \ldots, x_n) \) which contains all the free variables of \( T \), we can define a morphism \( [T]_{\bar{x}} \in \mathcal{L}((!\mathcal{U})^\otimes, \mathcal{U}) \), the denotational semantics (or interpretation) of \( T \) (where \( (!\mathcal{U})^\otimes := \bigotimes_{i=1}^k (!\mathcal{U}) \)). The definition is by induction on \( T \in !\Lambda \):

- \( [x_i]_{\bar{x}} := w^\otimes_{\mathcal{U}} \otimes \text{der}_Y \otimes w^{\otimes n-i}_{!\mathcal{U}} \) where \( w_{\mathcal{U}} \in \mathcal{L}(!\mathcal{U}, 1) \) is the weakening and we keep implicit the monoidality isos \( 1 \otimes \mathcal{U} \cong \mathcal{U} \),
- \( [\lambda y S]_{\bar{x}} := \langle 0_{(!\mathcal{U})^\otimes, !\mathcal{U}, \text{cur} ([S]_{\bar{x},Y}^Y) \rangle \), where we assume without loss of generality \( y \notin \{x_1, \ldots, x_n\} \),
- \( [S]_{\bar{x}} := \langle [S]_{\bar{x}} \rangle_{\bar{x}} \otimes \text{pr}_1 [R]_{\bar{x}} \) where \( c \in \mathcal{L}((!\mathcal{U})^\otimes, (!\mathcal{U})^\otimes \otimes (!\mathcal{U})^\otimes) \) is the contraction,
- \( [S]_{\bar{x}} := \langle [S]_{\bar{x}} \rangle_{\bar{x}} \otimes 0_{(!\mathcal{U})^\otimes, !\mathcal{U} \to \mathcal{U}} \), for \( ([S]_{\bar{x}})_{\bar{x}} \) is the coalgebra structure map of \( (!\mathcal{U})^\otimes \) (see [13]),
- \( [\text{der} S]_{\bar{x}} := \text{der}_Y \otimes \text{pr}_1 [S]_{\bar{x}} \).

Theorem 13 (Invariance. [14]). Let \( T, S \in !\Lambda \) and \( \bar{x} \) be a repetition-free list of variables which contains all free variables of \( T \) and \( S \). If \( T \approx_b S \) then \( [T]_{\bar{x}} = [S]_{\bar{x}} \).

The proof of Thm.13 uses crucially the fact that \([R]_{\bar{x}} \) is a coalgebra morphism, see [13].

The general notion of denotational model for the bang calculus presented here and obtained from any denotational model \( \mathcal{L} \) of LL satisfying the assumption (1) above is a particular case of Moggi’s semantics of computations based on monads [22,23], if one keeps in mind that the functor “!” defines a strong monad on the Kleisli category \( \mathcal{L}_! \) of \( \mathcal{L} \).
Call-by-name. A model of the CBN λ-calculus is a reflexive object in a cartesian closed category. The category \( \mathcal{L} \) being *-autonomous, its Kleisli \( \mathcal{L} \) over the comonad \((!, \text{dig}, \text{der})\) is cartesian closed. The category \( \mathcal{L}_1 \) (whose objects are the same as \( \mathcal{L} \) and morphisms are given by \( \mathcal{L}_1(A, B) = \mathcal{L}((A, B)) \)) has composition \( f \circ g \) defined as \( f! g \) and identities \( A \) given by \( \text{der}_A \). In \( \mathcal{L}_1 \), products \( A \times B \) are preserved, with projections \( \pi_i := \text{pr}_i \circ \text{der}_{(A \times B)} \) \( (i \in \{1, 2\}) \); the exponential object \( A \Rightarrow B \) is \(!A \rightarrow B \) (this is the semantic counterpart of Girard’s Cbn translation) and has an evaluation morphism \( \text{Ev} = \text{ev}(\text{der}_{A \rightarrow B} \otimes \text{id}_A)(m^2)^{-1} \in \mathcal{L}(!((!A \rightarrow B) \& !A), B) \). This defines an exponentiation since for all \( f \in \mathcal{L}(!((C \& A), B)) \) there is a unique morphism \( \Lambda(f) = \text{cur}(f^2) \in \mathcal{L}(!((C, A) \rightarrow B) \& !A) \) satisfying \( \text{Ev} \circ \langle \Lambda(f), A \rangle = f \).

The identity \( \mathcal{U} = !\mathcal{U} \& (|\mathcal{U} \rightarrow \mathcal{U} \& !\mathcal{U} \rightarrow \mathcal{U}) \) entails \( |\mathcal{U} \rightarrow \mathcal{U} \& !\mathcal{U} \rightarrow \mathcal{U} \) in \( \mathcal{L} \) via \( \text{lam} := \langle 0_{|\mathcal{U} \rightarrow \mathcal{U} \& !\mathcal{U} \rightarrow \mathcal{U}}, \text{id}_{|\mathcal{U} \rightarrow \mathcal{U}} \rangle \in \mathcal{L}(!|\mathcal{U} \rightarrow \mathcal{U} \& !\mathcal{U} \rightarrow \mathcal{U} \& |\mathcal{U} \rightarrow \mathcal{U}), |\mathcal{U} \rightarrow \mathcal{U} \& !\mathcal{U} \rightarrow \mathcal{U} \rangle \) and \( \text{app} := \text{pr}_2 \in \mathcal{L}(|\mathcal{U} \rightarrow \mathcal{U}), |\mathcal{U} \rightarrow \mathcal{U} \& !\mathcal{U} \rightarrow \mathcal{U} \) \), since \( \text{app} \circ \text{lam} = \text{id}_{|\mathcal{U} \rightarrow \mathcal{U}} \). So, \( \mathcal{U} \) is a reflexive object (i.e. \( \mathcal{U} \rightarrow \mathcal{U} \& !\mathcal{U} \) in \( \mathcal{L} \) via \( \text{app}_n := \text{der}_{|\mathcal{U} \rightarrow \mathcal{U}} \circ \text{app} \in \mathcal{L}(|\mathcal{U}, |\mathcal{U} \rightarrow \mathcal{U} \& !\mathcal{U} \rightarrow \mathcal{U}) \) and \( \text{lam}_n := \text{der}_{|\mathcal{U} \rightarrow \mathcal{U}} \circ \text{lam} \in \mathcal{L}(|\mathcal{U} \rightarrow \mathcal{U} \& !\mathcal{U} \rightarrow \mathcal{U}, |\mathcal{U} \rightarrow \mathcal{U}) \), since \( \text{app}_n \circ \text{lam}_n = !\mathcal{U} \rightarrow \mathcal{U} \). Therefore, the interpretation of a \( \lambda \)-term \( t \) can be defined as usual as a morphism \( |t|^n \in \mathcal{L}(|\mathcal{U}^k, |\mathcal{U} \& !\mathcal{U} \), with \( x = (x_1, \ldots, x_k) \) such that \( \text{fv}(t) \subseteq \{x_1, \ldots, x_k\} \):

\[
|t|^n |\mathcal{U}_k \defeq \pi^n, \\
|\lambda x t|^n |\mathcal{U}_k \defeq \text{lam}_n \circ \Lambda(|t|^n |\mathcal{U}_k), \\
|ts|^n |\mathcal{U}_k \defeq \text{Ev} \circ \langle \text{app}_n \circ |t|^n |\mathcal{U}_k, |s|^n |\mathcal{U}_k \rangle.
\]

Summing up, the object \( \mathcal{U} \) provides both a model of the bang calculus and a model of the CBN λ-calculus. The relation between the two: the semantics \( |t|^n \) in the Cbn model of the \( \lambda \)-calculus of a \( \lambda \)-term \( t \) decomposes into the semantics \( |t|^n \) in the model of the bang calculus of the Cbn translation \( t^n \) of \( t \).

Theorem 14 (Factorization of any Cbn semantics). For every \( \lambda \)-term \( t \), \( t^n = t^n |\mathcal{U} \) (up to Seely’s isos).

Proof. Below we use \( \cong \) to transform a morphism \( f \in \mathcal{L}(|\mathcal{U}^k \& !\mathcal{U} \& |\mathcal{U} \rightarrow \mathcal{U}), \mathcal{U} \) into a morphism \( g \in \mathcal{L}(|\mathcal{U}^k \& !\mathcal{U} \& |\mathcal{U} \rightarrow \mathcal{U}), \mathcal{U} \) using Seely’s isos (where \( K := \sum_{i=1}^k |\mathcal{U}_i \)) We proceed by induction on \( t \).

- If \( t := x_i \) then \( t^n = x_i \), so we have \( |t|^n |\mathcal{U} = |x_i|_x^n \).
- If \( t := \lambda y s \) then \( t^n = \lambda y s^n \).

\[
[\lambda y s^n]_x = \langle 0_{|\mathcal{U}^{\leq 2}, !\mathcal{U}, \text{cur}([s^n]_x^{1+y})} \rangle
\]

\[
= \text{der}_{|\mathcal{U}} !\langle 0_{|\mathcal{U}^{\leq 2}, !\mathcal{U}, \text{cur}([s^n]_x^{1+y})} \rangle \text{dig}_{|\mathcal{U}}
\]

\[
= \text{der}_{|\mathcal{U}} !\langle 0_{|\mathcal{U} \& !\mathcal{U} \& !\mathcal{U}, \text{id}_{|\mathcal{U} \& !\mathcal{U}} \rangle \text{cur}([s^n]_x^{1+y}) \text{dig}_{|\mathcal{U}}
\]

\[
= \langle \text{der}_{|\mathcal{U}} !\text{lam} \rangle \circ \Lambda([s^n]_x^{1+y}) = [\lambda y s]_x^{1+y}
\]

- If \( t := sr \) then \( t^n = (s^n)(r^n) \).

\[
[(s^n)(r^n)]_x = \text{Ev}((|\text{pr}_2 s^n]_x \otimes (|\text{pr}_1 (r^n)]_x)) c
\]

\[
= \text{Ev}((|\text{pr}_2 s^n]_x \otimes (|0_{|s^n} \& !\mathcal{U}, !\mathcal{U} \& !\mathcal{U})]) c
\]

\[
= \text{Ev}((|\text{pr}_2 s^n]_x \otimes (|f]_x)) c
\]

\[
= \text{Ev}(\langle \text{app} |s^n]_x \otimes (!|f]_x \text{dig}_{|\mathcal{U}} \rangle) c
\]

\[
= \text{Ev}(\text{der}_{|\mathcal{U} \& !\mathcal{U}} !\langle \text{app} |s^n]_x \rangle \text{dig}_{|\mathcal{U}} \otimes (!|f]_x \text{dig}_{|\mathcal{U}} \rangle) c
\]

\[
= \text{Ev}(\text{der}_{|\mathcal{U} \& !\mathcal{U}} \otimes \text{id}_{|\mathcal{U}} \rangle !\langle \text{app} |s^n]_x \rangle \text{dig}_{|\mathcal{U}} \otimes !|f]_x \text{dig}_{|\mathcal{U}} \rangle c
\]

\[
= \text{Ev} \circ \langle \text{der}_{|\mathcal{U} \& !\mathcal{U}} \rangle !\langle \text{app} |s^n]_x \rangle \text{dig}_{|\mathcal{U}} , !|f]_x \rangle
\]

\[
= \text{Ev} \circ \langle !\langle \text{der}_{|\mathcal{U} \& !\mathcal{U}} \rangle \text{app} \rangle \circ |s^n]_x \otimes |f]_x \rangle = |s^n]_x \otimes |f]_x \rangle
\]
Thm. [14] is a powerful result: it says not only that every LL based model of the bang calculus is also a model of the CbN $\lambda$-calculus, but also that the CbN semantics of any $\lambda$-term in such a model always naturally factorizes into the CbN translation of the $\lambda$-term and its semantics in the bang calculus.

**Call-by-value.** Following [26][11], models of the CbV $\lambda$-calculus can be defined using Girard’s “boring” CbV translation of the intuitionistic implication into LL. It is enough to find an object $X$ in $\mathcal{L}$ satisfying 

$$
!X \rightsquigarrow !X < X \text{ (or equivalently, } !(X \rightarrow X) < X) \text{.}
$$

This is the case for our object $\mathcal{U}$ (the model of the bang calculus) since $!\mathcal{U} \rightarrow \mathcal{U} < \mathcal{U}$ and $!\mathcal{U} < \mathcal{U}$ entail $!\mathcal{U} \rightarrow \mathcal{U} < \mathcal{U}$ (by the variance of $!X \rightarrow X$) via the morphisms $\lambda v = (0_{!X \rightarrow \mathcal{U}}, \lambda v (ev(0_{!X \rightarrow \mathcal{U}})) : !\mathcal{U} \rightarrow !\mathcal{U}$ and $\lambda v \in \mathcal{L}(\mathcal{U}, !\mathcal{U} \rightarrow \mathcal{U})$. As in [11], we can then define the interpretation of a $\lambda$-term $t$ as a morphism $|t|_x^y \in \mathcal{L}((!\mathcal{U})^{>k}, !\mathcal{U})$, with $x = (x_1, \ldots, x_k)$ such that $\text{fv}(t) \subseteq \{x_1, \ldots, x_k\}$:

$$
|x_i|_x^y = w_{!\mathcal{U}}^{x_{i-1}} \otimes id_{!\mathcal{U}} \otimes w_{!\mathcal{U}}^{x_k}, \quad |\lambda y \, t|_x^y = (\text{lam}_v (ev(|x|_x^y)))^t, \quad |t|_x^y = ev((\text{app}_v |t|_x^y) \otimes (|s|_x^y)).
$$

We now have two possible ways of interpreting the CbV $\lambda$-calculus in our model $\mathcal{U}$: either by translating a $\lambda$-term $t$ into $t^\lambda \in !\Lambda$ and then compute $|t^\lambda|$, or by computing directly $|t^\lambda|$. It is natural to wonder whether the two interpretations $|t^\lambda|$ and $|t|_x^y$ are related, and in what way. In [14] the authors conjectured that, at least in the case of a particular relational model $\mathcal{V}$ satisfying $\mathcal{V} = !\mathcal{U} \cup (!\mathcal{U} \times !\mathcal{U}) = !\mathcal{U} \& (!\mathcal{U} \rightarrow \mathcal{U})$, the two interpretations coincide. We show that the situation is actually more complicated than expected.

The relational model $\mathcal{U}$ introduced in [14] admits the following concrete description as a type system.

The set $\mathcal{U}$ of types and the set $!\mathcal{U}$ of finite multisets over $\mathcal{U}$ are defined by mutual induction as follows:

$$
\text{(set: } \mathcal{U} \text{)} \quad a, b, c := a | a \rightarrow a \quad \text{(set: } !\mathcal{U} \text{)} \quad a, b, c := [a_1, \ldots, a_k] \quad \text{for any } k \geq 0.
$$

**Environments** $\Gamma$ are functions from variables to $!\mathcal{U}$ whose support $\text{supp}(\Gamma) = \{x | \Gamma(x) \neq []\}$ is finite. We write $x_1 : a_1, \ldots, x_n : a_n$ for the environment $\Gamma$ satisfying $\Gamma(x_i) = a_i$ and $\Gamma(y) = []$ for $y \notin \bar{x}$. The multiset union $a + b$ is extended to environments pointwise, namely $(\Gamma + \Delta)(x) = \Gamma(x) + \Delta(x)$.

On the one hand, the relational model $\mathcal{U}$ for the CbV $\lambda$-calculus interprets a $\lambda$-term $t$ using $|t|_x^y$, which gives $|t|_x^y = \{(\Gamma, \beta) | \Gamma \vdash t : \beta\}$ where $\vdash$ is the type system below (note that if $\Gamma \vdash t : \beta$ then $\beta \in !\mathcal{U}$):

$$
\Gamma \vdash x : a \quad \frac{\Gamma \vdash t : [a \rightarrow b]}{\Gamma + \Delta \vdash t : b} \quad \frac{\Delta \vdash s : a}{\text{app}} \quad \frac{\Delta = \sum_{k=1}^K \Gamma_i}{\text{lam}}
$$

On the other hand, the relational model $\mathcal{V}$ for the bang calculus interprets a term $T \in !\Lambda$ using $[\cdot]_x$, which gives $[T]_x = \{(\Gamma, \beta) | \Gamma \vdash t : \beta\}$ where $\vdash$ is the type system defined as follows:

$$
\Gamma \vdash x : [\alpha] \quad \frac{\alpha = x \vdash x : [\alpha]}{\text{ax}} \quad \frac{\Gamma \vdash t : [a \rightarrow \beta]}{\Gamma + \Delta \vdash t : \beta} \quad \frac{\Delta = \sum_{k=1}^K \Gamma_i}{\text{der}} \quad \frac{\Gamma \vdash [\alpha]}{\Gamma \vdash \lambda x : [a \rightarrow \beta]}
$$

In $\mathcal{V}$ (seen as the relational model for the bang calculus) what is the interpretation $[t^\lambda]_x$ of the CbV translation $t^\lambda$ of a $\lambda$-term $t$? Easy calculations show that in the type system $\vdash$, the rules below — the ones needed to interpret terms of the form $t^\lambda$ for some $\lambda$-term $t$ — can be derived:

$$
\frac{\Gamma \vdash t^\lambda : [a \rightarrow \beta]}{\Delta \vdash s^\lambda : a} \quad \frac{\Delta = \sum_{k=1}^K \Gamma_i}{\text{app}} \quad \frac{\Gamma \vdash [\alpha]}{\Gamma \vdash \lambda y : [a \rightarrow \beta]}
$$

This approach is compatible with other notions of model such as Moggi’s one [22][23], since the functor $\text{""}1\text{""}$ defines a strong monad on the Kleisli category $\mathcal{L}$. The reflexive object $!X \rightarrow !X$ in $\mathcal{L}$ (or equivalently, $!(X \rightarrow X)$ in the CbV version in $\mathcal{L}$) of the reflexive object $X \Rightarrow X < X$ in a cartesian closed category, in accordance with Girard’s CbV decomposition of the arrow.
Intuitively, the type system $\vdash_v$ is obtained from the restriction of the type system $\vdash$ to the image of $(\cdot)^v$ by substituting arbitrary types $\beta$ with multisets $b$ of types. So, given a $\lambda$-term $t$, the two interpretations $[t]_X^v$ and $[t^v]_X$ can be different: for $\alpha \in \mathcal{U} \setminus !\mathcal{U}$ (e.g. take $\alpha = [\!] \to [\!]$), one has $[(a \to \alpha + a) \to \alpha] \in ((\lambda x(x))^v) \setminus (\lambda x(x))^v$.

**Proposition 15** (Relational semantics for CbV). In the relational model $\mathcal{U}$, $[t]_X^v \subseteq [t^v]_X$ for any $\lambda$-term $t$, with $\bar{x} = (x_1, \ldots, x_k)$ such that $\text{fv}(t) \subseteq \{x_1, \ldots, x_k\}$. There exists a closed $\lambda$-term $s$ such that $[s]_X^v \neq [s^v]_X$.

**Proof.** We have just shown that $[s]_X^v \neq [s^v]_X$ for $s = \lambda x.x$. To prove that $[t]_X^v \subseteq [t^v]_X$, it is enough to show, by induction on $t \in \Lambda$, that $x_1 : a_1, \ldots, x_n : a_n \vdash_v t^v : \beta$ is derivable whenever $x_1 : a_1, \ldots, x_k : a_k \vdash t : \beta$.

If $t$ is a variable, then $t = x_i$ for some $1 \leq i \leq k$, and $t^v = x_i^v$. All the derivations for $t$ in the type system $\vdash_v$ are of the form $x : a \vdash_v x : a$ for any $a \in !\mathcal{U}$, and in the type system $\vdash$, according to (3), $x : a \vdash x : a$ is derivable.

If $t = sr$, then $t^v = \langle \text{der } s^v \rangle r^v$ and all the type derivations for $t$ in the type system $\vdash_v$ are of the form

$$\Gamma \vdash_v s : [a \to b] \quad \Delta \vdash_r r : a \quad \text{app} \quad \Gamma + \Delta \vdash_v sr : b$$

for any $a, b \in !\mathcal{U}$.

By i.h., $\Gamma \vdash s^v : [a \to b]$ and $\Delta \vdash a^v : a$ are derivable in the type system $\vdash_v$, hence the following derivation is derivable in the type system $\vdash$, according to (3) since $!\mathcal{U} \subseteq \mathcal{U}$

$$\Gamma \vdash s^v : [a \to b] \quad \Delta \vdash a^v : a \quad \text{app} \quad \Gamma + \Delta \vdash \langle \text{der } s^v \rangle r^v : b$$

If $t = \lambda ys$, then $t^v = (\lambda y s^v)^v$ and all the type derivations for $t$ in the type system $\vdash_v$ are of the form

$$\frac{\sum_{i=1}^k \Gamma_i \vdash \lambda y s^v : [a_1 \to b_1, \ldots, a_k \to b_k]}{(\Gamma_i, y : a_i \vdash s^v : b_i)_{1 \leq i \leq k} \quad k \geq 0 \quad \text{lam}}$$

for any $a_1, b_1, \ldots, a_k, b_k \in !\mathcal{U}$.

By i.h., $\Gamma_i, y : a_i \vdash_v s^v : b_i$ is derivable in the type system $\vdash_v$ for all $1 \leq i \leq k$, hence the following derivation is derivable in the type system $\vdash$, according to (3) since $!\mathcal{U} \subseteq \mathcal{U}$

$$\frac{\sum_{i=1}^k \Gamma_i \vdash (\lambda y s^v)^v : [a_1 \to b_1, \ldots, a_k \to b_k]}{\frac{\sum_{i=1}^k \Gamma_i \vdash (\lambda y s^v)^v : [a_1 \to b_1, \ldots, a_k \to b_k]}{(\Gamma_i, y : a_i \vdash s^v : b_i)_{1 \leq i \leq k} \quad k \geq 0 \quad \text{lam}}}$$

The example above of $[s]_X^v \neq [s^v]_X$ shows also that in general neither $[t^v]_X = \langle [t]_0^v, 0 \rangle_X$ nor $\text{pr}_1 [t^v]_X = [t^v]_X$ hold in relational semantics. We conjecture that, for any $\lambda$-term $t$, $[t]_X^v$ can be obtained from $[t]_X^v$ by iterating the application of $\text{pr}_1$ to $[\cdot]$ along the structure of $t$, but how to express this formally and categorically for a generic model $\mathcal{U}$ of the bang calculus? Usually in these situations one defines a logical relation between the two interpretations, but this is complicated by the fact that we are in the untyped setting so there is no type hierarchy to base our induction. We plan to investigate whether the (syntactic) logical relations introduced by Pitts in [24] can give an inspiration to define semantic logical relations in the untyped setting. Another source of inspiration might be the study of other concrete LL based models of the CbV $\lambda$-calculus, such as Scott domains and coherent semantics [26, 11].

---

5Relational semantics interprets terms in the object $\mathcal{U}$ — defined in [2], where $(a \to \alpha)$ denotes the ordered pair $(a, \alpha)$ — of the category $\text{Ref}$ of sets and relations. The cartesian product $\&$ is the disjoint union, with the empty set as terminal and initial object $\top = 0$, so that the zero morphism $0_{X Y}$ for any objects $X$ and $Y$ is the empty relation and the projection $\text{pr}_1$ is the obvious selection. Therefore, in relational semantics, $[t]_X^v = [t]_X^v$ and $\text{pr}_1 [t^v]_X = [t^v]_X$ for any $\lambda$-term $t$. 
References

A Technical appendix: omitted proofs

The enumeration of propositions, theorems, lemmas already stated in the body of the article is unchanged.

A.1 Omitted proofs and remarks of Section 2

Let $n \in \mathbb{N}$ and $T, S_1, \ldots, S_n, x_1, \ldots, x_n \in \lambda \forall \text{ar}$. The substitution of $S_1, \ldots, S_n$ for $x_1, \ldots, x_n$ in $T$ is the term $T\{S_1/x_1, \ldots, S_n/x_n\}$ defined by induction on $T \in \lambda \forall$ as follows:

$$
\begin{align*}
&x_1\{S_1/x_1, \ldots, S_n/x_n\} := S_i \text{ for all } 1 \leq i \leq n \\
y\{S_1/x_1, \ldots, S_n/x_n\} := y \text{ with } y \notin \{x_1, \ldots, x_n\} \\
&(\lambda y)T\{S_1/x_1, \ldots, S_n/x_n\} := \lambda y T\{S_1/x_1, \ldots, S_n/x_n\} \text{ with } y \notin \bigcup \text{fv}(S_i) \cup \{x_1\} \\
&(\langle T \rangle R)\{S_1/x_1, \ldots, S_n/x_n\} := (T\{S_1/x_1, \ldots, S_n/x_n\}) R\{S_1/x_1, \ldots, S_n/x_n\} \\
&(T^1)\{S_1/x_1, \ldots, S_n/x_n\} := (T\{S_1/x_1, \ldots, S_n/x_n\})^1 \\
&(\text{der} T)\{S_1/x_1, \ldots, S_n/x_n\} := \text{der}(T\{S_1/x_1, \ldots, S_n/x_n\})
\end{align*}
$$

Lemma 16 (Composition of substitution). Let $T, S, R \in \lambda \forall$ and $x, y \in \forall \text{ar}$. If $y \notin \text{fv}(R) \cup \{x\}$ then $T\{y\} R/x = T R/x S\{R/x\}/y$.

Proof. By straightforward induction on $T \in \lambda \forall$. \qed

Remark 17 (Reflexive-transitive closure of parallel $\ell$-reduction). It is immediate to check that $\Rightarrow_\ell$ is reflexive and $\Rightarrow_\ell \subset \Rightarrow_* \subset \Rightarrow_\ell$, whence $\Rightarrow_* \equiv \Rightarrow_\ell$.

Lemma 18 (Substitution Lemma for parallel $\ell$-reduction). Let $T, S, R, Q \in \lambda \forall$ and $x \in \forall \text{ar}$. If $T \Rightarrow_\ell S$ and $R \Rightarrow_\ell Q$, then $T R/x \Rightarrow_\ell S Q/x$.

Proof. By induction on the derivation of $T \Rightarrow_\ell S$. Let us consider its last rule $r$.

- If $r = \text{var}$, then there are two sub-cases:
  - either $T = S = y \neq x$ and then $T R/x = y = S Q/x$, therefore $T R/x \Rightarrow_\ell S Q/x$ by applying the rule $\text{var}$;
  - or $T = x = S$ and then $T R/x = R \Rightarrow_\ell Q = S Q/x$ by hypothesis.

- If $r = \text{!}$, then $T = T'^1 \Rightarrow_\ell S'^1 = S$ with $T' \Rightarrow_\ell S'$. By induction hypothesis, $T' R/x \Rightarrow_\ell S' R/x$ and hence $S = Q^1 \Rightarrow_\ell (R^*)^1 = T^*$ (apply the rule $\text{!}$).
  
  The cases where $r = \lambda$ and $r = \text{der}$ are analogous to the previous one.

- If $r = \ell$, then $T = (\lambda y T_1) T_2 \Rightarrow_\ell S_1 S_2/y = S$ with $T_1 \Rightarrow_\ell S_1$ and $T_2 \Rightarrow_\ell S_2$ as conclusions of sub-derivations. We can suppose without loss of generality that $y \notin \text{fv}(R) \cup \{x\}$. By induction hypothesis, $T_1 R/x \Rightarrow_\ell S_1 R/x$ and $T_2 R/x \Rightarrow_\ell S_2 R/x$, so $T R/x = (\lambda y T_1 R/x) (T_2 R/x) \Rightarrow_\ell S_1 R/x S_2 R/x/y = S_1 S_2 R/x = S R/x$ by applying the rule $\ell$ and Lemma 16.

- Finally, if $r = \odot$, then there are two sub-cases:
  - either $T = \langle R \odot Q \rangle$ and $S = \langle R' \odot Q' \rangle$ with $R \Rightarrow_\ell R'$, $Q \Rightarrow_\ell Q'$, and $R \neq \lambda x P$ or $Q \neq \lambda \forall x$; by induction hypothesis, $R' \Rightarrow_\ell R''$ and $Q' \Rightarrow_\ell Q''$, therefore $S = \langle R'' \odot Q'' \rangle = \langle R' \odot Q' \rangle \odot = T^*$ (apply the rule $\odot$);
  - or $T = \lambda x R \Rightarrow_\ell T'$ and $S = \lambda y Q S'$ with $R \Rightarrow_\ell Q$ and $T' \Rightarrow_\ell S'$; by induction hypothesis, $Q \Rightarrow_\ell R^*$ and $S' \Rightarrow_\ell T'^*$, hence $S = \langle \lambda x Q \rangle S' \Rightarrow_\ell R^* T'^*/x \Rightarrow_\ell T^*$ by applying the rule $\ell$. \qed
For any term $T$, we denote by $T^*$ the term obtained by reducing all $\ell$-redexes in $T$ simultaneously. Formally, $T^*$ is defined by induction on $T \in !\Lambda$ as follows:

$$
\begin{align*}
  x^* := x & \quad (\lambda x T)^* := \lambda x T^* & \quad (T^1)^* := (T^*)^1 & \quad (\text{der} T)^* := \text{der}(T^*) \\
  ((T)S)^* := (T^*)S^* & \quad \text{if } T \neq \lambda x R \text{ or } S \notin !\Lambda & \quad (\langle \lambda x T \rangle S)^* := T^*\{S/x\}
\end{align*}
$$

**Lemma 2** (Development). Let $T, S \in !\Lambda$. If $T \Rightarrow_\ell S$, then $S \Rightarrow_\ell T^*$.

*Proof.* By induction on the derivation of $T \Rightarrow_\ell S$. Let us consider its last rule $r$.

If $r = \text{var}$, then $T = x = S$ and $T^* = x$, thus $S \Rightarrow_\ell T^*$ (apply the var rule).

If $r = \lambda$, then $T = R \Rightarrow_\ell Q = S$ with $R \Rightarrow_\ell Q$ as premise. By induction hypothesis, $Q \Rightarrow_\ell R^*$ and hence $S = Q^* \Rightarrow_\ell (R^*)^* = T^*$ (apply the rule $\lambda$).

The cases where $r = \lambda$ and $r = \text{der}$ are analogous to the previous one.

If $r = \ell$, then $T = (\langle \lambda x R \rangle T_1^1 \Rightarrow_\ell Q \{S_1/x\}) = S$ with $R \Rightarrow_\ell Q$ and $T_1 \Rightarrow_\ell S_1$ as conclusions of subderivations. By induction hypothesis, $Q \Rightarrow_\ell R^*$ and $S_1 \Rightarrow_\ell T_1^*$, hence $S = Q \{S_1/x\} \Rightarrow_\ell R^* \{T_1^*/x\} = T^*$, according to Lemma [18].

Finally, if $r = \otimes$, then there are two subcases:

- either $T = (\langle R \rangle Q \Rightarrow_\ell (R')Q') = S$ with $R \Rightarrow_\ell R'$ and $Q \Rightarrow_\ell Q'$ as premises, and $R \neq \lambda x P$ or $Q \notin !\Lambda$; by i.h., $R' \Rightarrow_\ell R^*$ and $Q' \Rightarrow_\ell Q^*$, therefore $S = (R')Q' \Rightarrow_\ell (R')Q^* = T^*$ (apply the rule $\otimes$);
- or $T = (\langle \lambda x R \rangle T_1^1 \Rightarrow_\ell (\lambda x Q) S_1^1) = S$ with $R \Rightarrow_\ell Q$ and $T_1 \Rightarrow_\ell S_1$ as conclusions of subderivations; by i.h., $Q \Rightarrow_\ell R^*$ and $S_1 \Rightarrow_\ell T_1^*$, so $S = (\lambda x Q) S_1^1 \Rightarrow_\ell R^* \{T_1^*/x\} = T^*$ (apply the rule $\ell$). $\square$

**Lemma 2** is the key ingredient to prove the confluence of $\Rightarrow_\ell$ (Lemma [14]), indeed it entails that $\Rightarrow_\ell$, and hence $\Rightarrow_\ell^*$ by Rmk. [17], are strongly confluent.

**Lemma 19** (Substitution Lemma for reductions). Let $T, T', R \in !\Lambda, x \in \text{fv}(r)$.

1. For any $r \in \{\ell, d, b, \ell_g, d_g, b_g\}$, if $T \rightarrow_\ell T'$ then $T\{R/x\} \rightarrow_\ell T'\{R/x\}$.

2. For any $r \in \{\ell, d, b\}$, if $T \rightarrow_\ell T'$ then $R\{T/x\} \rightarrow_\ell R\{T'/x\}$.

*Proof.*

1. By induction on $T \in !\Lambda$

First, let us consider the cases where $r \in \{\ell_g, d_g, b_g\}$. $T$ cannot be a box because boxes are $r$-normal. Therefore, there are only the following cases:

- **Root-step**, i.e. $T \rightarrow_\ell T'$ where $r \in \{\ell, d, b\}$. Since $\rightarrow_\ell = \rightarrow_d \cup \rightarrow_\ell$, there are only two subcases to consider:
  - $T := \text{der}(T'' \rightarrow_d T')$, so $T\{R/x\} = \text{der}(T'\{R/x\}) \rightarrow_\ell T'\{R/x\}$.
  - $T := (\lambda y Q) S' \rightarrow_\ell Q\{S/y\} = T'$ and we can suppose without loss of generality that $y \notin \text{fv}(R) \cup \{x\}$. Note that $S'\{R/x\} = (S\{R/x\})^1 \in !\Lambda$, so $T\{R/x\} = (\lambda y Q\{R/x\})(S\{R/x\})^1 \rightarrow_\ell Q\{R/x\}\{S\{R/x\}/y\} = Q\{S/y\}\{R/x\} = T'\{R/x\}$.

- **Dereliction**, i.e. $T := \text{der} S \rightarrow_\ell \text{der} S' := T'$ with $S \rightarrow_\ell S'$: by induction hypothesis, $S\{R/x\} \rightarrow_\ell S'\{R/x\}$ and hence $T\{R/x\} = \text{der}(S\{R/x\}) \rightarrow_\ell \text{der}(S'\{R/x\}) = T'\{R/x\}$.

- **Abstraction**, i.e. $T := \lambda y S \rightarrow_\ell \lambda y S' := T'$ with $S \rightarrow_\ell S'$: we can suppose without loss of generality that $y \notin \text{fv}(R) \cup \{x\}$. By i.h., $S\{R/x\} \rightarrow_\ell S'\{R/x\}$ and hence $T\{R/x\} = \lambda y (S\{R/x\}) \rightarrow_\ell (\lambda y S\{R/x\}) \rightarrow_\ell (\lambda y S'\{R/x\}) = T'\{R/x\}$.

- **Application Right**, i.e. $T := \langle Q \rangle S \rightarrow_\ell \langle Q \rangle S' := T'$ with $S \rightarrow_\ell S'$: By i.h., $S\{R/x\} \rightarrow_\ell S'\{R/x\}$ and hence $T\{R/x\} = \langle Q\{R/x\}\rangle (S\{R/x\}) \rightarrow_\ell \langle Q\{R/x\}\rangle (S'\{R/x\}) = T'\{R/x\}$.
• Application Left, i.e. \( T := \langle S \rangle Q \rightarrow r \langle S' \rangle Q =: T' \) with \( S \rightarrow r, S' \): analogous to the previous case. Now, let us consider the case where \( r \in \{ \ell, !, b \} \). The proof is perfectly analogous to the one above, the only novelty is a new case:

• Box, i.e. \( T := S' \rightarrow \rho S' := T' \) with \( S \rightarrow r, S' \): by induction hypothesis, \( S' \{ R/x \} \rightarrow r, S' \{ R/x \} \) and hence \( T \{ R/x \} = (S \{ R/x \})' \rightarrow (S' \{ R/x \})' = T' \{ R/x \} \).

2. Concerning \( r = \ell \), from \( T \rightarrow_{\ell} T' \) it follows that \( T \Rightarrow_{\ell} T' \) by Remark 17. According to Lemma 18 (since \( R \Rightarrow_{\ell} R \) by reflexivity of \( \Rightarrow_{\ell} \)), \( R \{ T/x \} \Rightarrow_{\ell} R \{ T'/x \} \) and hence \( R \{ T/x \} \rightarrow_{\ell} R \{ T'/x \} \) by Remark 17 again.

Concerning \( r = d \), the proof is by straightforward induction on \( R \in A \).

Concerning \( r = b \), we have just proved that \( R \{ T'/x \} \rightarrow_{d}^* R \{ T'/x \} \) and \( R \{ T/x \} \rightarrow_{d}^* R \{ T'/x \} \), therefore we are done because \( \rightarrow_{b} = \rightarrow_{\ell} \cup \rightarrow_{d} \).

The next lemma lists a series of properties of \( \ell\cdot, \ell_{g}\cdot, d\cdot \) and \( d_{g}\cdot \)reductions that will be used to prove strong confluence of \( \rightarrow_{b_{g}} \) and confluence of \( \rightarrow_{b} \) (Prop. 4). See p. 5

Lemma 3 (Basic properties of reductions).

1. \( \rightarrow_{\ell_{g}} \) is strongly confluent (i.e. \( \ell_{g} \leftarrow \cdot \rightarrow_{\ell_{g}} \subseteq (\rightarrow_{\ell_{g}} \cdot \ell_{g} \leftarrow) \cup =) \).

2. \( \rightarrow_{d_{g}} \) and \( \rightarrow_{d} \) are strongly confluent (separately).

3. \( \rightarrow_{d_{g}} \) and \( \rightarrow_{\ell_{g}} \) strongly commute (i.e. \( \ell_{g} \leftarrow \cdot \rightarrow_{\ell_{g}} \subseteq \ell_{g} \cdot d_{g} \leftarrow) \).

\( \rightarrow_{d} \) quasi-strongly commutes over \( \ell \) (i.e. \( d \leftarrow \cdot \rightarrow_{\ell} \subseteq \ell \cdot d_{g} \leftarrow) \).

4. \( \rightarrow_{\ell} \) is confluent.

Proof. 1. Let \( T, T_{1}, T_{2} \in A \) be such that \( T_{1} \ell_{g} \leftarrow T \rightarrow_{\ell_{g}} T_{2} \) and \( T_{1} \neq T_{2} \): we prove by induction on \( T \in A \) that \( T_{1} \rightarrow_{\ell_{g}} T' \ell_{g} \leftarrow T_{2} \) for some \( T' \in A \). Since \( \rightarrow_{\ell_{g}} \) does not reduce under \( ! \), there are only the following cases:

• Root-step for \( T \rightarrow_{\ell_{g}} T_{1} \) and non-root-step for \( T \rightarrow_{\ell_{g}} T_{2} \), i.e. \( T := \langle \lambda x S \rangle R' \rightarrow_{\ell_{g}} S \{ R/x \} =: T_{1} \) and \( T \neq T_{2} \). Note that \( R' \) is \( \ell_{g} \)-normal since \( \rightarrow_{\ell_{g}} \) does not reduce under \( ! \), hence the fact that \( T \rightarrow_{\ell_{g}} T_{2} \) via a non-root-step implies that \( T_{2} := \langle \lambda x S \rangle R \) and \( S \rightarrow_{\ell_{g}} Q \). By Lemma 19.1 \( T_{1} \rightarrow_{\ell_{g}} Q \{ R/x \} \ell_{g} \leftarrow T_{2} \).

• Dereffiction for both \( T \rightarrow_{\ell_{g}} T_{1} \) and \( T \rightarrow_{\ell_{g}} T_{2} \), i.e. \( T \leftarrow_{\ell_{g}} T \rightarrow_{\ell_{g}} T \) with \( T := \lambda S_{1} \ell_{g} \leftarrow T \rightarrow_{\ell_{g}} S_{1} \ell_{g} \leftarrow S \rightarrow_{\ell_{g}} S \). By induction hypothesis, there exists \( S' \in A \) such that \( S_{1} \rightarrow_{\ell_{g}} S' \ell_{g} \leftarrow S_{2} \), whence \( T_{1} \rightarrow_{\ell_{g}} S' \ell_{g} \leftarrow T_{2} \).

• Abstraction for both \( T \rightarrow_{\ell_{g}} T_{1} \) and \( T \rightarrow_{\ell_{g}} T_{2} \), i.e. \( T \leftarrow_{\ell_{g}} \lambda S_{1} \ell_{g} \leftarrow T \rightarrow_{\ell_{g}} \lambda S_{2} \leftarrow T \). By induction hypothesis, there exists \( S' \in A \) such that \( S_{1} \rightarrow_{\ell_{g}} S' \ell_{g} \leftarrow S_{2} ; \) analogous to the previous case.

• Application left for both \( T \rightarrow_{\ell_{g}} T_{1} \) and \( T \rightarrow_{\ell_{g}} T_{2} \), i.e. \( T \leftarrow_{\ell_{g}} \langle S \rangle R \ell_{g} \leftarrow T \rightarrow_{\ell_{g}} \langle S \rangle R \edge =: T_{2} \). By induction hypothesis, there exists \( S' \in A \) such that \( S_{1} \rightarrow_{\ell_{g}} S' \ell_{g} \leftarrow S_{2} \), whence \( T_{1} \rightarrow_{\ell_{g}} \langle S' \rangle R \ell_{g} \leftarrow T_{2} \).

• Application right for both \( T \rightarrow_{\ell_{g}} T_{1} \) and \( T \rightarrow_{\ell_{g}} T_{2} \), i.e. \( T \leftarrow_{\ell_{g}} \langle R \rangle S_{1} \ell_{g} \leftarrow T \rightarrow_{\ell_{g}} \langle R \rangle S \edge =: T_{2} \). By induction hypothesis, there exists \( S' \in A \) such that \( S_{1} \rightarrow_{\ell_{g}} S' \ell_{g} \leftarrow S_{2} ; \) analogous to the previous case.

• Application left for \( T \rightarrow_{\ell_{g}} T_{1} \) and Application Right \( T \rightarrow_{\ell_{g}} T_{2} \), i.e. \( T := \langle S \rangle R \edge \) and \( T_{1} := \langle S' \rangle R_{\ell_{g}} \leftarrow T \rightarrow_{\ell_{g}} \langle S \rangle R' \edge =: T_{2} \) with \( S \rightarrow_{\ell_{g}} S' \) and \( R \rightarrow_{\ell_{g}} R' \). Then, \( T_{1} \rightarrow_{\ell_{g}} \langle S' \rangle R' \ell_{g} \leftarrow T_{2} \).

Box Application Left
2. We prove that $\rightarrow_d$ is strongly confluent; the proof of strong confluence of $\rightarrow_{d^*}$ is analogous (and without the cases box and root-step). Let $T, T_1, T_2 \in !\Lambda$ be such that $T_1 \leftarrow_d T \rightarrow_d T_2$ and $T_1 \not\equiv T_2$; we prove by induction on $T \in !\Lambda$ that $T_1 \rightarrow_d T' \leftarrow_d T_2$ for some $T' \in !\Lambda$. Cases:

- **Root-step for** $T \rightarrow_d T_1$ and **non-Root-step for** $T \rightarrow_d T_2$, i.e. $T := \text{der}(T_1') \rightarrow_d T_1$ and $T \rightarrow_d \text{der}(S') := T_2$ with $T_1 \rightarrow_d S$. Then, $T_1 \rightarrow_d S \leftarrow_d T_2$.
- **Dereliction for both** $T \rightarrow_d T_1$ and $T \rightarrow_d T_2$, i.e. $T := \text{der} S$ and $T_1 := \text{der} S_1 \leftarrow_d T \rightarrow_d \text{der} S_2 := T_2$ with $S_1 \leftarrow_d S \rightarrow_d S_2$. By induction hypothesis, there exists $S' \in !\Lambda$ such that $S_1 \rightarrow_d S' \leftarrow_d S_2$, whence $T_1 \rightarrow_d \text{der} S' \leftarrow_d T_2$.
- **Abstraction for both** $T \rightarrow_d T_1$ and $T \rightarrow_d T_2$, i.e. $T := \lambda x S$ and $T_1 := \lambda x S_1 \leftarrow_d T \rightarrow_d \lambda x S_2 := T_2$ with $S_1 \leftarrow_d S \rightarrow_d S_2$: analogous to the previous case.
- **Box for both** $T \rightarrow_d T_1$ and $T \rightarrow_d T_2$, i.e. $T_1 := S_1 \leftarrow_d T \rightarrow_d S_2 := T_2$ with $T := S'$ and $S_1 \leftarrow_d S \rightarrow_d S_2$: analogous to the previous case.
- **Application Left for both** $T \rightarrow_d T_1$ and $T \rightarrow_d T_2$, i.e. $T := \langle S \rangle R$ and $T_1 := \langle S_1 \rangle R \leftarrow_d T \rightarrow_d \langle S \rangle R := T_2$ with $S_1 \leftarrow_d S \rightarrow_d S_2$. By induction hypothesis, there exists $S' \in !\Lambda$ such that $S_1 \rightarrow_d S' \leftarrow_d S$, whence $T_1 \rightarrow_d \langle S' \rangle R \leftarrow_d T_2$.
- **Application Right for both** $T \rightarrow_d T_1$ and $T \rightarrow_d T_2$, i.e. $T := \langle S \rangle R$ and $T_1 := \langle S \rangle S_1 \leftarrow_d T \rightarrow_d \langle S \rangle S := T_2$ with $S \rightarrow_d S'$ and $R \rightarrow_d R'$. Then, $T_1 \rightarrow_d \langle S' \rangle R' \leftarrow_d T_2$.

3. First we show that $\rightarrow_{e*}$ and $\rightarrow_{d*}$ strongly commute. Let $T, T_1, T_2 \in !\Lambda$ be such that $T_1 \rightarrow_{d*} T \rightarrow_{e*} T_2$; we prove by induction on $T \in !\Lambda$ that there exists $T' \in !\Lambda$ such that $T_1 \rightarrow_{e*} T' \rightarrow_{d*} T_2$. Since $\rightarrow_{e*}$ and $\rightarrow_{d*}$ do not reduce under $\rightarrow_\ell$, the only interesting case is the following:

- **Root-step for** $T \rightarrow_{e*} T_2$ and **non-Root-step for** $T \rightarrow_{d*} T_1$, i.e. $T := \langle \lambda x S \rangle R' \rightarrow_{e*} \langle \lambda x S \rangle R' = T_2$ and $T \rightarrow_{d*} \langle \lambda x S' \rangle R' := T_1$ with $S \rightarrow_{d*} S'$ (recall that $R'$ is $d^*$-normal, so $T \rightarrow_{d*} T_1$ implies $S \rightarrow_{d*} S'$ by necessity). By Lemma 19.1, $T_1 \rightarrow_{e*} S' \langle R/x \rangle_d \leftarrow_d T_2$.

The other cases are analogous to the ones in the proof of Lemma 3.1 not involving root-steps, paying attention that now the induction hypothesis is different.

Now we show that $\rightarrow_d$ quasi-strongly commutes over $\rightarrow_\ell$, i.e. $d \leftarrow_\ell \rightarrow_\ell \emptyset \subseteq \leftarrow_{e*} \rightarrow_\ell$. This implies that $\rightarrow_\ell$ and $\rightarrow_d$ commute. Let $T, T_1, T_2 \in !\Lambda$ be such that $T_1 \rightarrow_d T \rightarrow_\ell T_2$: we prove by induction on $T \in !\Lambda$ that $T_1 \rightarrow_\ell T' \rightarrow_\ell T_2$ for some $T' \in !\Lambda$. The only interesting cases are the following:

- **Root-step for** $T \rightarrow_d T_1$ and **non-Root-step for** $T \rightarrow_\ell T_2$, i.e. $T := \text{der}(T_1') \rightarrow_d T_1$ and $T \rightarrow_\ell \text{der}(T''_1) := T_2$ with $T_1 \rightarrow_\ell T''$. Then, $T_1 \rightarrow_\ell T' \leftarrow_\ell T_2$.
- **Root-step for** $T \rightarrow_\ell T_2$ and **non-Root-step for** $T \rightarrow_d T_1$. There are two subcases:
  - $T := \langle \lambda x S \rangle R \rightarrow_d \langle \lambda x S' \rangle R := T_1$ and $T \rightarrow_\ell S \langle R/x \rangle := T_2$ with $S \rightarrow_d S'$. Then, $T_1 \rightarrow_\ell S_1 \langle R/x \rangle \leftarrow_d T_2$ by Lemma 19.1.
  - $T := \langle \lambda x S \rangle R \rightarrow_d \langle \lambda x S \rangle R := T_1$ and $T \rightarrow_\ell S \langle R/x \rangle := T_2$ with $S \rightarrow_d R'$. Then, $T_1 \rightarrow_\ell S_1 \langle R/x \rangle \leftarrow_d T_2$ according to Lemma 19.2.

The other cases are analogous to the ones in the proof of Lemma 3.2 not involving root-steps, paying attention that now the inductive hypothesis is different.

4. Lemma 2 entails that $\rightarrow_\ell$ is strongly confluent: indeed, if $S \leftarrow_\ell T$ then $S \rightarrow_\ell T^* \leftarrow_\ell R$. Therefore, $\rightarrow_\ell$ is confluent, that is $\rightarrow_\ell$ is strongly confluent. But $\rightarrow_\ell = \rightarrow_\ell^*$ according to Rmk. 17. So, $\rightarrow_\ell$ is confluent.
Proposition 4 (Strong confluence of $\rightarrow_{b^*}$ and confluence of $\rightarrow_b$).

1. The reduction $\rightarrow_{b^*}$ is strongly confluent, i.e. $b^* \leftarrow \cdots \rightarrow b^* \subseteq (\rightarrow_{b^*} \cdot b^* \leftarrow) \cup \cdot$.

2. The reduction $\rightarrow_b$ is confluent: $b \leftarrow \cdots \rightarrow b \subseteq (b^* \cdot b^* \leftarrow) \cup \cdot$.

Proof.

1. Since $\rightarrow_{d^*}$ and $\rightarrow_{d^*}$ are strongly confluent (Lemmas 3 and 2) and strongly commute (Lemma 3), it follows immediately that $\rightarrow_{d^*} \cup \rightarrow_{d^*}$ is strongly confluent, where by definition $\rightarrow_{b^*} = \rightarrow_{d^*} \cup \rightarrow_{d^*}$.

2. Since $\rightarrow_d$ and $\rightarrow_e$ are confluent (Lemmas 3 and 3) and commute (Lemma 3), $\rightarrow_d \cup \rightarrow_e$ is confluent by Hindley-Rosenthal Lemma 4, Lemma 3.5], where by definition $\rightarrow_b = \rightarrow_d \cup \rightarrow_e$. 

A.2 Omitted proofs and remarks of Section 3

Lemma 11 (Properties of the forgetful translation $(\cdot)^*$).

1. $(\cdot)^*$ is a right-inverse of $(\cdot)^+$: For every $t \in \Lambda$, one has $t^{x^*} = t$.

2. $(\cdot)^*$ preserves substitution: $M \{ U / x \} \in !\Lambda^v$ with $(M \{ U / x \})^+ = M^+ \{ U^+ / x \}$, and $U^+ \{ U / x \} \in !\Lambda^v$ with $(U^+ \{ U / x \})^+ = U^+ \{ U^+ / x \}$, for any $M \in !\Lambda^v$ and $U, U^+ \in !\Lambda^v$.

3. $(\cdot)^*$ maps b-reduction into $\beta^v$-reduction: For any $M \in !\Lambda^v, U \in !\Lambda^v$ and $T, S \in !\Lambda$, if $M \rightarrow_b T$ then $T \in !\Lambda^v$ and $M^+ \rightarrow^+ T^+$; if $U \rightarrow_b S$ then $S \in !\Lambda^v$ and $U^+ \rightarrow^+ S^+$.

Proof. Statements of Lemmas 1, 2, 3 proved here are slightly stronger than in p. 10 to get the right i.h.

1. Proof by induction on $t \in \Lambda$. Cases:

   - Variable, i.e. $t := x$, then $t^{x^*} = x^*$ and hence $t^{x^*} = x = t$.
   - Abstraction, i.e. $t := \lambda x s$ for some $s \in \Lambda$, then $t^{x^*} = (\lambda_x x^{s^*})^+$. By i.h., $s^{x^*} = s$ and hence $i^{x^*} = \lambda_x x^{s^*} = \lambda_x s = t$.
   - Application, i.e. $t := sr$ with $s, r \in \Lambda$, then $t^{x^*} = (s x^{s^*})^+$. By i.h., $s^{x^*} = s$ and $R^{x^*} = R$, so $i^{x^*} = s^{x^*} = s = r = t$.

2. Proof by mutual induction on $M \in !\Lambda^v$ and $U \in !\Lambda^v$. Cases:

   - Variable, i.e. $U^+ \in \Lambda$. There are two subcases:
     - either $y = x$ and then $U^+ \{ U / x \} = U \in !\Lambda^v$ and $U^{x^*} = x$, hence $(U^+ \{ U / x \})^+ = U^+ = U^+ \{ U^+ / x \}$.
     - or $y \neq x$ and then $U^+ \{ U / x \} = y \in !\Lambda^v$ and $U^{x^*} = y$, thus $(U^+ \{ U / x \})^+ = y = U^+ \{ U^+ / x \}$.
   - Abstraction, i.e. $U^+ := \lambda_y N$. We can suppose without loss of generality that $y \notin \text{fv}(U)\cup\{x\}$, hence $U^+ \{ U / x \} = \lambda_y N \{ U / x \} \in !\Lambda^v$ (since $N \{ U / x \} \in !\Lambda^v$ by i.h.) and $U^{x^*} = \lambda_y N$. By i.h., $(N \{ U / x \})^+ = N^+ \{ U^+ / x \}$. So, $(U^+ \{ U / x \})^+ = \lambda_y N \{ U \} \in !\Lambda^v$ (since $U^+ \{ U / x \} \in !\Lambda^v$ by i.h.) and $U^+ = U^+ \{ U^+ / x \}$. By i.h., $(U^+ \{ U / x \})^+ = \lambda_y N \{ U^+ / x \} = U^+ \{ U^+ / x \}$.
   - Box, i.e. $M := (U^+) \cdot N$. Then, $M \{ U / x \} = (U^+ \{ U / x \})^+ \in !\Lambda^v$ (since $U^+ \{ U / x \} \in !\Lambda^v$ by i.h.) and $M^+ = U^{x^*} \{ U^+ / x \}$. By i.h., $(U^+ \{ U / x \})^+ = U^{x^*} \{ U^+ / x \}$. Therefore, $(M \{ U / x \})^+ = (U^+ \{ U / x \})^+ = U^+ \{ U^+ / x \} \cdot M^+ \{ U^+ / x \}$.
   - Application without top-level dereliction, i.e. $M := (U^+) \cdot N$. So, $M \{ U / x \} \in !\Lambda^v$ (as $U^+ \{ U / x \} \in !\Lambda^v$ and $N \{ U / x \} \in !\Lambda^v$ by i.h.) and $M^+ = U^{x^*} \{ U^+ / x \}$ and $(N \{ U / x \})^+ = N^+ \{ U^+ / x \}$. Therefore, $(M \{ U / x \})^+ = (U^+ \{ U / x \})^+ \cdot (N \{ U / x \})^+ = U^+ \{ U^+ / x \} \cdot N^+ \{ U^+ / x \} \cdot M^+ \{ U^+ / x \}$.
• Application with top-level dereliction, i.e. \( M := \langle \text{der} N \rangle L \). Then, \( M^\dagger = N^\dagger L^\dagger \) and \( M\{U/x\} = \langle \text{der} N\{U/x\} \rangle L\{U/x\} \in !\Lambda^v \) (since \( N\{U/x\}, L\{U/x\} \in !\Lambda^v \)). By induction hypothesis, \( (N\{U/x\})^\dagger = N^\dagger \{U^\dagger/x\} \) and \( (L\{U/x\})^\dagger = L^\dagger \{U^\dagger/x\} \). Therefore,
\[
(M\{U/x\})^\dagger = (N\{U/x\})^\dagger (L\{U/x\})^\dagger = N^\dagger \{U^\dagger/x\} L^\dagger \{U^\dagger/x\} = M^\dagger \{U^\dagger/x\}.
\]

3. Proof by mutual induction on \( M \in !\Lambda^v \) and \( U \in !\Lambda^v \). Since there is neither \( M = \text{der} R \) nor \( U = \text{der} R \), there are only the following cases:

• Root-step, i.e. \( M \eisarrow b T \). As \( \eisarrow b = \eisarrow d \cup \eisarrow i \), there are two cases to consider. The case \( M \eisarrow d T \) is impossible since there is no \( M = \text{der} R \in !\Lambda^v \). It remains only one case for the root-step: \( M := \langle \lambda x N \rangle U^\dagger \eisarrow i N\{U/x\} =: T \). Notice that \( U^\dagger \) is \( \lambda \)-value, i.e. a variable or an abstraction. Then, by Lemma 11.2, \( T \in !\Lambda^v \) and \( M^\dagger = (\lambda x N^\dagger)U^\dagger \eisarrow b N^\dagger \{U^\dagger/x\} = (N\{U/x\})^\dagger = T^\dagger \).

• Box, i.e. \( M := U^\dagger \eisarrow b S^\dagger =: T \) with \( U \eisarrow b S \). By i.h., \( S \in !\Lambda^v \) and \( U^\dagger \eisarrow b S^\dagger \). Hence, \( M^\dagger = U^\dagger \eisarrow b S^\dagger = T^\dagger \).

• Application left, i.e. \( M := \langle R \rangle N \eisarrow b \langle R' \rangle N =: T \) with \( R \eisarrow b R' \). There are three subcases:
  - either \( R := \text{der} (U^\dagger) \eisarrow d U :=: R', \) and then \( T = \langle U \rangle N \in !\Lambda^v \) with \( M^\dagger = (\text{der} (U^\dagger))^\dagger N^\dagger = U^\dagger N^\dagger = T^\dagger \) (in particular, \( M^\dagger \eisarrow b S^\dagger, T^\dagger \));
  - or \( R := \text{der} L \eisarrow b \text{der} S :=: R' \) with \( L \eisarrow b S \), and then \( S \in !\Lambda^v \) with \( L^\dagger \eisarrow b S^\dagger \) by i.h., so \( T = \langle \text{der} S \rangle R \in !\Lambda^v \) and \( M^\dagger = L^\dagger N^\dagger \eisarrow b S^\dagger N^\dagger = T^\dagger \);
  - or \( R := U \eisarrow b S :=: R' \) with \( U \eisarrow b S \), and then \( S \in !\Lambda^v \) and \( U^\dagger \eisarrow b S^\dagger \) by i.h., thus \( T = \langle S \rangle N \in !\Lambda^v \) and \( M^\dagger = U^\dagger N^\dagger \eisarrow b S^\dagger N^\dagger = T^\dagger \).

• Application right, i.e. \( M := \langle R \rangle N \eisarrow b \langle R' \rangle S =: T \) where \( N \eisarrow b S \) and either \( R := U \) or \( R := \text{der} L \). By i.h., \( S \in !\Lambda^v \) and \( N^\dagger \eisarrow b S^\dagger \). Therefore, either \( T = \langle U \rangle S \in !\Lambda^v \) and \( M^\dagger = U^\dagger N^\dagger \eisarrow b U^\dagger S^\dagger = T^\dagger \), or \( T = \langle \text{der} L \rangle S \in !\Lambda^v \) and \( M^\dagger = L^\dagger N^\dagger \eisarrow b L^\dagger S^\dagger = T^\dagger \).

• Abstraction, i.e. \( U := \lambda x N \eisarrow b \lambda x R =: S \) with \( N \eisarrow b R \). By i.h., \( R \in !\Lambda^v \) and \( N^\dagger \eisarrow b R^\dagger \). Hence, \( S \in !\Lambda^v \) and \( U^\dagger = \lambda x N^\dagger \eisarrow b \lambda x R^\dagger = S^\dagger \). \( \square \)