Lambda Calculi through the Lens of Linear Logic — Delia Kesner

Lecture 4, Quantitative Types - *July 3, 2025*

Contents

1 Introduction

Different computation strategies require different techniques in order to prove the same property. We would like to do it homogeneously, i.e. we would like to prove each property once for a class of strategies. Call-by-value (CBV) and call-by-name (CBN) are an example of different strategies requiring different methods to prove same properties.

1.1 CBN and CBV

CBN and CBV are the two reduction strategies most important for λ-calculus. They were first studied by Plotkin ([1]), who has shown their equivalence.
Call-by-value first computes its argument, then substitutes it for the variable, what may cause duplication.
Call-by-value first substitutes (duplicates) function arguments and then it computes them in place they appear.

Examples:

$$\begin{aligned} &\text{CBV:} \quad (\lambda x.\mathrm{fst}(x,x))\ 3^2 \twoheadrightarrow \mathrm{fst}(3^2,3^2) \twoheadrightarrow 3^2 \to 9 \\ &\text{CBN:} \quad (\lambda x.\mathrm{fst}(x,x))\ 3^2 \twoheadrightarrow (\lambda x.\mathrm{fst}(x,x))\ 9 \twoheadrightarrow \mathrm{fst}(9,9) \twoheadrightarrow 9 \end{aligned}$$

Call-by-value strategy is better understood than call-by-name, but it is rarely used in actual programming languages. In practice, languages either use call-by-value or call-by-name that need more advanced machinery to be studied. In order to unify those two strategies and be able to reason about the CBV strategy using CBN's tools we define Bang calculus, that captures fundamental properties of both.

2 Subsuming Framework

The foundations of Bang-calculus come from Levy's call-by-push-value and Girard's Linear logic. These two frameworks focuses on two different aspects. Call-by-push-value focus on values versus computation, whilst linear logic focus on linear resources versus erasable/duplicable resources. However both subsume call-by-name and call-by-value.

2.1 Bang-Calculus

The first Bang-Calculus [2] was created by Ehrhard and it bridged the gap between call-by-push-value and linear logic, but it was incomplete, because terms denoting non-terminating programs

could be blocked. Hence, Ehrhard and Guerrieri [3] created a second Bang-Calculus that is complete but breaks confluence. The Bang-Calculus we will discuss, the (Distant) Bang-Calculus [4], created by Bucciarelli et. al. recovers both completeness and confluence.

Syntax Within the terms we have three special ones: value/bang ($!t$), deriliction (der t) and explicit substitution ($t[x\backslash u]$)

$$\textbf{Terms:} \quad t, u ::= x \mid tu \mid \lambda x.t \mid !t \mid \text{der } t \mid t[x\backslash u]$$

Operational semantics We have three operational semantics rules, β-reduction, substitution that only substitutes values (marked with a bang) and deriliction that turns a term/value into a computation.

Beta:	$(\lambda x.t)u \to t[x\backslash u]$
Substitution:	$t[x\backslash !u] \to t[x\backslash u]$
Value/Computation:	$\text{der}(!t) \to t$

2.2 Distant Bang-Calculus

Distant Bang-Calculus is inspired from linear logic, and, as such it avoid blocking situations. In this framework we use $t\ L$ (a list of substitutions) that abbreviates $t\ [x_1\backslash u_1]...[x_n\backslash u_n]$ ($n \geq 0$). Hence, the operational semantics are also altered.

Beta:	$(\lambda x.t)L\ u \to t[x\backslash u]\ L$
Substitution:	$t[x\backslash (!u)\ L] \to t[x\backslash u]\ L$
Value/Computation:	$\text{der}((!t)\ L) \to t\ L$

Note that this calculus is non-deterministic

3 Simple untyped properties

Given that we have t^N meaning that `NAME` maps to and t^V meaning `BANG` maps to `BANG` then we have the following general form:

$$t \text{ verifies property } P \text{ in } \texttt{NAME} \text{ iff } t^N \text{ verifies property} P \text{ in } \texttt{BANG}$$
$$t \text{ verifies property } P \text{ in } \texttt{VALUE} \text{ iff } t^V \text{ verifies property} P \text{ in } \texttt{BANG}$$

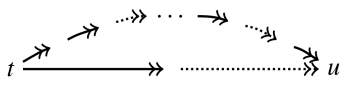
Encoding untyped CBN/CBV These following encodings are Girard's translations. However, there are several variation of the translation, each with different properties. We take note of them all and use different variants when applicable. Also since we can duplicate/erase abstraction, we add a bang to them. For example, if we have $(tu)^V$ then we have an abstraction. In Bang calculus an abstraction is a computation, hence we need a deriliction $(der(t^V)u^V)$.

$$-^N : \texttt{NAME} \mapsto \texttt{BANG} \qquad -^V \texttt{VALUE} \mapsto \texttt{BANG}$$
$$x^N \stackrel{\text{def}}{=} x \qquad x^V \stackrel{\text{def}}{=} !x$$
$$(\lambda x.t)^N \stackrel{\text{def}}{=} \lambda x.t^N \qquad (\lambda x.t)^V \stackrel{\text{def}}{=} !\lambda x.t^V$$
$$(tu)^N \stackrel{\text{def}}{=} t^N !u^N \qquad (tu)^V \stackrel{\text{def}}{=} der(t^V)u^V$$
$$(t[x\backslash u])^N \stackrel{\text{def}}{=} t^N[x\backslash !u^N] \qquad (t[x\backslash u])^V \stackrel{\text{def}}{=} t^V[x\backslash u^V]$$

3.1 Deriving properties

With Bang-calculus we can derive property P for `NAME` and `VALUE` from property P for `BANG`. P can be different properties such as: confluence, factorization, typability with intersection types, (qualitative/quantitative) termination, inhabitation, meaningfulness and genericity.

Take $P :=$ confluence, where confluence is defines as follows. A reduction relation enjoys confluence if for every term t, u, v such that $u \twoheadleftarrow t \twoheadrightarrow v$ there is a term s such that $u \twoheadrightarrow s \twoheadleftarrow$. From the Distant Bang-Calculus paper [4] we know that `BANG` is confluent and we also know that `NAME` and `VALUE` confluence can be derived from `BANG` confluence [5]. Another example is to derive factorization, which is defined as follows. A reduction relation enjoy factorization if every sequence can be rearranged so that (relevant) external steps are performed before (irrelevant) internal steps (see Fig below).



We will not derive it personally, but `BANG` admits factorization and `NAME` and `VALUE` factorization can be derived from `BANG` factorization [5].

4 Intersection types

Systems with intersection types allow us to express operational properties of languages. Different systems correspond to different reduction strategies. Gardner's system $\mathcal{H}$ that we have seen yesterday is an example. It corresponds to CBN reduction strategy. [6] We will refer to it as system $\mathcal{N}$. Another system (that we will call $\mathcal{V}$ here) modeling the CBV semantics ([7], [8]) will be discussed in detail in the next lecture. Here we will define system $\mathcal{B}$ covering semantics of BANG from [8].

All three models satisfy the following property:

Theorem 4.1. *(Qualitative termination) t is typable in $\mathcal{N}/\mathcal{V}/\mathcal{B}$ if and only if it terminates in NAME/VALUE/BANG*

4.1 Types for System $\mathcal{B}$

Type system $\mathcal{B}$ differs from the previous system in that we have explicit dereliction (der) and bang (!).

$$\begin{array}{rl} \text{(Types)} & \mathtt{A} ::= \iota \mid \mathtt{M} \mid \mathtt{M} \to \mathtt{A} \\ \text{(Multi-Types)} & \mathtt{M} ::= [\mathtt{A}_i]_{i\in I} \end{array}$$

$$\frac{}{x : [A] \vdash x : A}$$

$$\frac{\Gamma \vdash t : A}{\Gamma \setminus x \vdash \lambda x.t : \Gamma(x) \to A}$$

$$\frac{\Gamma \vdash t : M \to A \qquad \Delta \vdash u : M}{\Gamma \sqcup \Delta \vdash tu : A}$$

$$\frac{\Gamma \vdash t : A \qquad \Delta \vdash u : \Gamma(x)}{(\Gamma \setminus x) \sqcup \Delta \vdash t[x\backslash u] : A}$$

$$\frac{(\Gamma_i \vdash t : A_i)_{i\in I}}{\bigsqcup_{i\in I} \Gamma_i \vdash !t : [A_i]_{i\in I}}$$

$$\frac{\Gamma \vdash t : [A]}{\Gamma \vdash \text{der } t : A}$$

4.2 Termination

We can prove the following relation between the terms of `NAME` and the terms `BANG` and between the terms of `VALUE` and the terms `BANG`.

Theorem 4.2. *(Preservation [8])*

- $\triangleright_{\mathcal{N}}\Gamma \vdash t : A \iff \triangleright_{\mathcal{B}}\Gamma \vdash t^N : A$
- $\triangleright_{\mathcal{V}}\Gamma \vdash t : A \iff \triangleright_{\mathcal{B}}\Gamma \vdash t^V : A$

Corollary 4.3. *(Termination)*

- *t is NAME-terminating if and only if t^N is BANG-terminating*
- *t is VALUE-terminating if and only if t^V is BANG-terminating*

We can derive qualitative termination information (i.e. number of reduction steps and the size of the normal form) of NAME and VALUE from BANG. To this end, we decorate each type system with counters like we did in case of system $\mathcal{H}$ ([9]). We achieve systems $\mathcal{EN}$, $\mathcal{EV}$ and $\mathcal{EB}$ respectively with judgements of the form $\Gamma \vdash^{(L,S)} t : \sigma$, where L denotes the length of term's normalization procedure and S denotes the size of the result. For each system the following property holds.

Theorem 4.4. *(Quantitative termination) $\Pi \triangleright_{\mathcal{EN}/\mathcal{EV}/\mathcal{EB}} \Gamma \vdash^{(L,S)} t : A$ (i.e. t is (tightly) typable in $\mathcal{EN}/\mathcal{EV}/\mathcal{EB}$) if and only if t terminates in NAME/VALUE/BANG in L steps and its normal form is of size S*

We can also prove a refined version of the preservation theorem that relates the lengths of reduction paths and sizes of normal forms between all three calculi.

Theorem 4.5. *(Preservation [10])*

- $\triangleright_{\mathcal{EN}}\Gamma \vdash^{(L,S)} t : A \iff \triangleright_{\mathcal{B}}\Gamma \vdash^{(L,S)} t^N : A$
- $\triangleright_{\mathcal{EV}}\Gamma \vdash^{(L,S)} t : A \iff \triangleright_{\mathcal{B}}\Gamma \vdash^{(L',S)} t^V : A$

where each of L and L' is calculable from the derivation of the other

5 Inhabitation Property for Bang-Calculus

Definition of Inhabitation Problem (IP)

We define the inhabitation problem P for the Bang-calculus as:

> Given a context Γ and a type A, does there exist a term t such that $\Gamma \vdash t : A$?

This is denoted formally as:

$$P := \text{inhabitation}$$

The inhabitation problem connects the semantics of types with program synthesis: a solution to the problem yields a term that realizes the logical specification encoded by the type.

Key Properties of the BANG Inhabitation Algorithm

An algorithm exists to solve the IP for `BANG`, which satisfies the following critical properties:

- **Termination**: The algorithm always halts after a finite number of steps.
- **Soundness**: Every term it returns is indeed a valid inhabitant of the given type.
- **Completeness**: If a type is inhabited, the algorithm will find (at least one) such inhabitant.

Implementation: This algorithm has been implemented in OCaml by V. Arrial.

Preservation Theorem (Arrial–Guerrieri–Kesner, 2023, [11],[12])

A key meta-theoretical result is that:

> **The decidability of the IP for `BANG` implies the decidability of the IP for both call-by-name and call-by-value systems.**

Formally:

$$\text{Decidability of } \texttt{BANG} \Rightarrow \text{Decidability of } \texttt{NAME} \text{ and } \texttt{VALUE}$$

This reflects the idea that Bang-calculus subsumes both call-by-name and call-by-value as specific fragments. Thus, solving the inhabitation problem `BANG` provides a general method that also covers more familiar evaluation strategies.

Typing and Inhabitation: A Dual Perspective

The following table summarizes the relationship between typing and inhabitation problems in various type systems:

System	**Typing** ($\Gamma? \vdash t : A$)	**Inhabitation** ($\Gamma \vdash ? : A?$)
Simple Types	Decidable	Decidable
Idempotent Intersection Types	Undecidable	Undecidable
Non-Idempotent Types	Undecidable	`NAME`: Decidable, `VALUE`: Decidable

6 The Meaningfulness Property

Intuition

In a resource-sensitive setting like `BANG`, not all terms are well-behaved or "meaningful." Some terms are intuitively *clashes* – combinations that reduce to ill-formed or unsound configurations.

This motivates a semantic distinction:

- **Meaningful terms** are those which, when placed in an appropriate context, reduce to some observable value.
- **Meaningless terms** are those that fail to yield such observations, often due to incompatible structure or irreducibility.

Example of a clash:

$$(!t)\, u \quad \text{and} \quad t(\lambda x.u)$$

These forms are considered **meaningless** as they missalign duplication and function application in the linear calculus.

Formal Definition of Meaningfulness in Bang

We define meaningfulness using two key notions:

- **Testing contexts** $\mathcal{T}$: evaluation contexts that feed a term with arguments. Defined inductively:
$$\mathcal{T} ::= \langle\cdot\rangle \mid \mathcal{T}\, s \mid (\lambda x.\mathcal{T})\, s$$
- **Observable terms**: typically values, such as $!u$, that indicate computational content.

Definition 6.1. A term t is **meaningful** in `BANG` if there exists a testing context $\mathcal{T}$ and an observable term $!u$ such that:

$$\mathcal{T}\langle t\rangle \longrightarrow^* !u$$

Examples:

$$\lambda x.x \quad \text{is meaningful;} \qquad \Omega, \quad x\,\Omega, \quad x\,x \quad \text{are meaningless.}$$

Logical Characterization

Meaningfulness is sensitive to **contextual substitution roles**. For example, in $x\,x$:

- The rightmost x must act as a **value** (argument),
- The leftmost x must act as a **function**,

but a single variable x cannot simultaneously fulfill both roles in a linear setting — rendering the term meaningless.

This reflects incompatibility similar to that found in calculi with conflicting data interpretations (Bucciarelli–Kesner–Ronchi Della Rocca '15 [8]).

Meaningfulness in `NAME` and `VALUE`

Different calculi instantiate meaningfulness differently:

- `NAME`-meaningfulness = **solvability** (Wadsworth '71, [13]):

  $$\text{A term } t \text{ is solvable if } \exists \mathcal{T} \text{ such that } \mathcal{T}\langle t\rangle \to^* I$$

- `VALUE`-meaningfulness = **potential valuability** (Paolini–Ronchi Della Rocca '99):

  $$\text{A term } t \text{ is potentially valuable if } \exists \mathcal{T} \text{ such that } \mathcal{T}\langle t\rangle \to^* v \text{ for some value } v.$$

We write $P :=$ meaningfulness when discussing these properties generically.

Preservation Theorem

Theorem 6.1 (Arrial–Guerrieri–Kesner '24 [11][12])**.**

- *t is `NAME`-meaningful* $\iff$ *t^{N} is `BANG`-meaningful*

- t *is* `VALUE`*-meaningful* $\iff$ t^{V} *is* `BANG`*-meaningful*

This establishes that the Bang-calculus captures the meaningfulness properties of both evaluation strategies faithfully, via appropriate translations t^{N} and t^{V} into the Bang-calculus.

7 Conclusion and Further Work

The Bang-calculus presents itself as a robust and unifying framework, capable of subsuming both untyped and typed paradigms. In the untyped setting, it accounts for key meta-theoretical properties such as confluence, factorization, meaningfulness, and genericity. In the typed setting, it enables precise analysis of program behavior, including exact upper bounds on term length and size, as well as decidable algorithms for the inhabitation problem. These foundational strengths open the way to further inquiries, such as extending the framework to capture approximation theories relevant to both `NAME` and `VALUE` calculi, incorporating more expressive features like global state or non-determinism, or exploring alternative frameworks—such as those based on `NEED`—that might offer different yet compatible perspectives on subsumption and operational reasoning.

References

[1] G. Plotkin, "Call-by-name, call-by-value and the λ-calculus," *Theoretical Computer Science*, vol. 1, no. 2, pp. 125–159, 1975, ISSN: 0304-3975. DOI: https://doi.org/10.1016/0304-3975(75)90017-1. [Online]. Available: https://www.sciencedirect.com/science/article/pii/0304397575900171.

[2] T. Ehrhard, "Call-by-push-value from a linear logic point of view," in *Programming Languages and Systems*, P. Thiemann, Ed., Berlin, Heidelberg: Springer Berlin Heidelberg, 2016, pp. 202–228, ISBN: 978-3-662-49498-1.

[3] T. Ehrhard and G. Guerrieri, "The bang calculus: An untyped lambda-calculus generalizing call-by-name and call-by-value," in *Proceedings of the 18th International Symposium on Principles and Practice of Declarative Programming*, ser. PPDP '16, Edinburgh, United Kingdom: Association for Computing Machinery, 2016, pp. 174–187, ISBN: 9781450341486. DOI: 10.1145/2967973.2968608. [Online]. Available: https://doi.org/10.1145/2967973.2968608.

[4] A. Bucciarelli, D. Kesner, A. Ríos, and A. Viso, "The bang calculus revisited," *Information and Computation*, vol. 293, p. 105 047, 2023, ISSN: 0890-5401. DOI: https://doi.org/10.1016/j.ic.2023.105047. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0890540123000500.

[5] V. Arrial, G. Guerrieri, and D. Kesner, "Genericity through stratification," in *Proceedings of the 39th Annual ACM/IEEE Symposium on Logic in Computer Science*, ser. LICS '24, Tallinn, Estonia: Association for Computing Machinery, 2024, ISBN: 9798400706608. DOI: 10.1145/3661814.3662113. [Online]. Available: https://doi.org/10.1145/3661814.3662113.

[6] P. Gardner, "Discovering needed reductions using type theory," in *Theoretical Aspects of Computer Software*, M. Hagiya and J. C. Mitchell, Eds., Berlin, Heidelberg: Springer Berlin Heidelberg, 1994, pp. 555–574, ISBN: 978-3-540-48383-0.

[7] T. Ehrhard, "Collapsing non-idempotent intersection types," in *Computer Science Logic (CSL'12) - 26th International Workshop/21st Annual Conference of the EACSL*, P. Cégielski and A. Durand, Eds., ser. Leibniz International Proceedings in Informatics (LIPIcs), vol. 16, Dagstuhl, Germany: Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2012, pp. 259–273, ISBN: 978-3-939897-42-2. DOI: 10.4230/LIPIcs.CSL.2012.259. [Online]. Available: https://drops.dagstuhl.de/entities/document/10.4230/LIPIcs.CSL.2012.259.

[8] A. Bucciarelli, D. Kesner, A. Ríos, and A. Viso, "The bang calculus revisited," *Information and Computation*, vol. 293, p. 105 047, 2023, ISSN: 0890-5401. DOI: https://doi.org/10.1016/j.ic.2023.105047. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0890540123000500.

[9] B. Accattoli, S. Graham-Lengrand, and D. Kesner, "Tight typings and split bounds," *Proc. ACM Program. Lang.*, vol. 2, no. ICFP, Jul. 2018. DOI: 10.1145/3236789. [Online]. Available: https://doi.org/10.1145/3236789.

[10] D. Kesner and A. Viso, "Encoding tight typing in a unified framework," in *30th EACSL Annual Conference on Computer Science Logic, CSL 2022, February 14-19, 2022, Göttingen, Germany (Virtual Conference)*, F. Manea and A. Simpson, Eds., ser. LIPIcs, vol. 216, Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2022, 27:1–27:20. DOI: 10.4230/LIPICS.CSL.2022.27. [Online]. Available: https://doi.org/10.4230/LIPIcs.CSL.2022.27.

[11] V. Arrial, G. Guerrieri, and D. Kesner, *Genericity through stratification*, 2024. arXiv: 2401.12212 [cs.LO]. [Online]. Available: https://arxiv.org/abs/2401.12212.

[12] D. Kesner, V. Arrial, and G. Guerrieri, *Meaningfulness and genericity in a subsuming framework*, 2024. arXiv: 2404.06361 [cs.LO]. [Online]. Available: https://arxiv.org/abs/2404.06361.

[13] C. P. Wadsworth, *Semantics and pragmatics of the lambda-calculus*, Ph.D. Thesis, 1971.