Types of Fireballs
The noble art of non-idempotent intersection types in Call-by-Value

Giulio Guerrieri\textsuperscript{1} \hspace{1cm} Beniamino Accattoli\textsuperscript{2}

\textsuperscript{1}DISI, Università di Bologna (Italy) \hspace{1cm} \textsuperscript{2}INRIA, Saclay (France)

Mathematical Foundations Seminars, University of Bath
Bath (United Kingdom), 20\textsuperscript{th} November 2018
Outline

1. Introduction

2. The rise and the fall of the fireball calculus $\lambda_{\text{fire}}$

3. The split fireball calculus $\lambda_{\text{fire}}^{\text{split}}$ and multi types

4. Tight type derivations and exact bounds
Outline

1 Introduction

2 The rise and the fall of the fireball calculus $\lambda_{\text{fire}}$

3 The split fireball calculus $\lambda_{\text{fire}}^{\text{split}}$ and multi types

4 Tight type derivations and exact bounds
What is a good λ-calculus?

There are many variants of λ-calculus:

- **Strong** or **weak** (evaluation can fire under λ’s or not);
- Term may be **open** or only **closed**;
- Evaluation strategies: call-by-name (CbN), call-by-value (CbV), call-by-need;
- With or without explicit substitutions, with or without linear substitutions;
- …

What is a good λ-calculus? Some (non-exhaustive) criteria:

1. Rewriting;
2. Logic;
3. Implementation;
4. Cost model;
5. Denotations;

Meta-principle: The more principles are connected, the better.
What is a good λ-calculus?

There are many variants of λ-calculus:

- **Strong** or **weak** (evaluation can fire under λ’s or not);
- Term may be **open** or only **closed**;
- Evaluation strategies: call-by-name (CbN), call-by-value (CbV), call-by-need;
- With or without explicit substitutions, with or without linear substitutions;
- ...

What is a good λ-calculus? Some (non-exhaustive) criteria:

1. **Rewriting**;
2. **Logic**;
3. **Implementation**;
4. **Cost model**;
5. **Denotations**;
6. **Equality**.

**Meta-principle:** The more principles are connected, the better.
Protagonist 1: Open Call-by-Value

The setting we will study via non-idempotent intersection types: Open CbV, i.e.
- evaluation is weak (does not reduce under \(\lambda\)'s),
- terms are possibly open.

\(\Rightarrow\) intermediate setting between Strong CbV (which evaluates under \(\lambda\)'s) and Closed CbV (terms are closed and evaluation is weak).

**Motivation:** Closed CbV is enough for modeling programming languages, not proof-assistants. Strong CbV is a very tricky setting.

In the literature, there are many equivalent presentations of Open CbV. The fireball calculus \(\lambda_{\text{fire}}\) is one of them, with many good properties...
- elegant rewrite theory, clear logical understanding;
- simple implementation in abstract machines, with a reasonable cost model;

...many inventors with different motivations: Ronchi Della Rocca & Paolini (1999, 2004), Grégoire & Leroy (2002), Accattoli & Sacerdoti Coen (2014);
...and a big problem: there is no adequate denotational model for it!
Protagonist 1: Open Call-by-Value

The setting we will study via non-idempotent intersection types: Open CbV, i.e.

- evaluation is **weak** (does not reduce under $\lambda$’s),
- terms are possibly **open**.

$\Rightarrow$ intermediate setting between Strong CbV (which evaluates under $\lambda$’s) and Closed CbV (terms are closed and evaluation is weak).

**Motivation:** Closed CbV is enough for modeling programming languages, not proof-assistants. Strong CbV is a very tricky setting.

In the literature, there are many equivalent presentations of Open CbV. The **fireball calculus** $\lambda_{fire}$ is one of them, with many good properties...

- elegant rewrite theory, clear logical understanding;
- simple implementation in abstract machines, with a reasonable cost model;

...many inventors with different motivations: Ronchi Della Rocca & Paolini (1999, 2004), Grégoire & Leroy (2002), Accattoli & Sacerdoti Coen (2014);

...and a **big problem:** there is no adequate denotational model for it!
Protagonist 2: Non-idempotent intersection types

Non-idempotent intersections types (aka multi-types) have many features:

- **Qualitative**: they characterize normalization for (some strategy of) $\lambda$-calculus;
- **Quantitative**: they provide bounds on the execution time (i.e. the number of $\beta$-steps) to reach the normal form;
- **Denotational semantics**: they can be seen as a syntactic presentation of relational semantics, a denotational model of $\lambda$-calculus;
- **Linear logic interpretation**: they are deeply linked to linear logic (LL).

De Carvalho’s System R (2009) shows all these features for CbN $\lambda$-calculus.

Our goal: to show these features in a CbV setting (the fireball calculus).

Rmk (for LL friends only): The multi-type system used in the $\lambda$-calculus depends on the evaluation mechanism, according the two Girard’s translations.

$$ (A \rightarrow B)^n = !A^n \rightarrow B^n $$

$$ (A \rightarrow B)^v = !A^v \rightarrow !B^v $$
Protagonist 2: Non-idempotent intersection types

Non-idempotent intersections types (aka multi-types) have many features:

- **qualitative**: they characterize normalization for (some strategy of) $\lambda$-calculus;
- **quantitative**: they provide bounds on the execution time (i.e. the number of $\beta$-steps) to reach the normal form;
- **denotational semantics**: they can be seen as a syntactic presentation of relational semantics, a denotational model of $\lambda$-calculus;
- **linear logic interpretation**: they are deeply linked to linear logic (LL).

De Carvalho’s System R (2009) shows all these features for CbN $\lambda$-calculus.

**Our goal**: to show these features in a CbV setting (the fireball calculus).

**Rmk** (for LL friends only): The multi-type system used in the $\lambda$-calculus depends on the evaluation mechanism, according the two Girard’s translations.

\[
\begin{align*}
\text{CbN} & \quad (A \to B)^n = !A^n \multimap B^n \\
\text{CbV ("boring")} & \quad (A \to B)^v = !A^v \multimap !B^v
\end{align*}
\]
Protagonist 2: Non-idempotent intersection types

Non-idempotent intersections types (aka multi-types) have many features:

- **qualitative**: they characterize normalization for (some strategy of) $\lambda$-calculus;
- **quantitative**: they provide bounds on the execution time (i.e. the number of $\beta$-steps) to reach the normal form;
- **denotational semantics**: they can be seen as a syntactic presentation of relational semantics, a denotational model of $\lambda$-calculus;
- **linear logic interpretation**: they are deeply linked to linear logic (LL).

De Carvalho’s System R (2009) shows all these features for CbN $\lambda$-calculus.

**Our goal**: to show these features in a CbV setting (the fireball calculus).

**Rmk** (for LL friends only): The multi-type system used in the $\lambda$-calculus depends on the evaluation mechanism, according the two Girard’s translations.

\[
\begin{align*}
\text{CbN} & \quad (A \to B)^n = !A^n \circ B^n \\
\text{CbV ("boring")} & \quad (A \to B)^\nu = !A^\nu \circ !B^\nu
\end{align*}
\]
Outline

1. Introduction

2. The rise and the fall of the fireball calculus $\lambda_{\text{fire}}$

3. The split fireball calculus $\lambda_{\text{split}}^{\text{fire}}$ and multi types

4. Tight type derivations and exact bounds
Plotkin’s CbV $\lambda$-calculus (1975)

Terms $t, u, s ::= x | \lambda x.t | tu$

Values $v, v' ::= x | \lambda x.t$

Left evaluation contexts $C ::= \langle \cdot \rangle | vC | Ct$

Rule at top level

$\lambda x.t)v \rightarrow_{\beta_v} t\{v/x\} \quad C\langle t\rangle \rightarrow_{\beta_v} C\langle u\rangle \text{ if } t \rightarrow_{\beta_v} u$

Plotkin’s CbV is well-behaved when evaluation is weak and terms are only closed.

- **Harmony:** A closed term $t$ is $\beta_v$-normal iff $t$ is a value (i.e. an abstraction).

Consider the case of weak evaluation and possibly open terms (Open CbV).

\[
t := (\lambda z.\delta)(xx)\delta \quad \text{where } \delta := \lambda y.yy
\]

$t$ is $\beta_v$-normal but it "morally" shouldn’t! It should behave like the diverging $\delta\delta$.

- In any denotational model for CbV, $t$ has the same semantics as $\delta\delta$;
- for any sensitive notion of observational equivalence, $t$ and $\delta\delta$ are equivalent.
The fireball calculus $\lambda_{fire}$ (1999, 2002, 2014)

Restore the good properties of Plotkin’s CbV for Open CbV.

| Terms     | $t, u, s, r$                             | $::= x | \lambda x.t | tu$ |
|------------|-----------------------------------------|----------------------------------|
| Values     | $v, v', v''$                            | $::= x | \lambda x.t$            |
| Fireballs  | $f, f', f''$                            | $::= v | i$                           |
| Inert terms| $i, i', i''$                            | $::= xf_1 \ldots f_n \quad n > 0$ |

Right evaluation contexts

$C ::= \langle \cdot \rangle | tC | Cf$

Rule at top level

$$(\lambda x.t)v \mapsto_{\beta_v} t\{v/x\}$$

$(\lambda x.t)i \mapsto_{\beta_i} t\{i/x\}$

Contextual closure

$C\langle t \rangle \mapsto_{\beta_v} C\langle u \rangle \quad \text{if} \quad t \mapsto_{\beta_v} u$

$C\langle t \rangle \mapsto_{\beta_i} C\langle u \rangle \quad \text{if} \quad t \mapsto_{\beta_i} u$

Reduction

$$\rightarrow_{\beta_f} := \rightarrow_{\beta_v} \cup \rightarrow_{\beta_i}$$

Example: $(\lambda z.\delta)(xx)\delta \rightarrow_{\beta_i} \delta\delta \rightarrow_{\beta_v} \delta\delta \rightarrow_{\beta_v} \ldots$ (where $\delta := \lambda y.yy$)

---

Proposition (Distinctive properties of $\lambda_{fire}$)

1. Open harmony: $t$ is $\beta_f$-normal if and only if $t$ is a fireball.
2. Conservative open extension: $t \rightarrow_{\beta_f} u$ if and only if $t \rightarrow_{\beta_v} u$, when $t$ is closed.
3. Evaluation and inert substitutions commute: $t \rightarrow_{\beta_f} u$ iff $t\{i/x\} \rightarrow_{\beta_f} u\{i/x\}$. 

G. Guerrieri, B. Accattoli

Types of Fireballs

Bath 2018/11/20 9 / 23
The fireball calculus $\lambda_{\text{fire}}$ (1999, 2002, 2014)

Restore the good properties of Plotkin’s CbV for Open CbV.

Terms $t, u, s, r ::= x \mid \lambda x. t \mid tu$

Values $v, v', v'' ::= x \mid \lambda x. t$

Fireballs $f, f', f'' ::= v \mid i$

Inert terms $i, i', i'' ::= xf_1 \ldots f_n \quad n > 0$

Right evaluation contexts $C ::= \langle \cdot \rangle \mid tC \mid Cf$

Rule at top level

$$(\lambda x. t)v \mapsto_{\beta_v} t\{v/x\}$$

$(\lambda x. t)i \mapsto_{\beta_i} t\{i/x\}$

Contextual closure

$C\langle t \rangle \mapsto_{\beta_v} C\langle u \rangle$ if $t \mapsto_{\beta_v} u$

$C\langle t \rangle \mapsto_{\beta_i} C\langle u \rangle$ if $t \mapsto_{\beta_i} u$

Reduction

$\rightarrow_{\beta_f} ::= \rightarrow_{\beta_v} \cup \rightarrow_{\beta_i}$

Example: $(\lambda z. \delta)(xx)\delta \rightarrow_{\beta_i} \delta\delta \rightarrow_{\beta_v} \delta\delta \rightarrow_{\beta_v} \ldots \quad \text{(where } \delta ::= \lambda y. yy)$

Proposition (Distinctive properties of $\lambda_{\text{fire}}$)

1. **Open harmony:** $t$ is $\beta_f$-normal if and only if $t$ is a fireball.

2. **Conservative open extension:** $t \rightarrow_{\beta_f} u$ if and only if $t \rightarrow_{\beta_v} u$, when $t$ is closed.

3. **Evaluation and inert substitutions commute:** $t \rightarrow_{\beta_f} u$ iff $t\{i/x\} \rightarrow_{\beta_f} u\{i/x\}$. 
Non-idempotent intersection types for CbV (Ehrhard, 2012)

Multi types and linear types are defined by mutual induction:

*linear types* $L, L' ::= P \rightarrow Q$

*multi types* $P, Q ::= [L_1, \ldots, L_n] \ (n \geq 0)$

- there are no base types: their role is played by the empty multiset $[] \ (n = 0)$;
- A multi type $[L_1, \ldots, L_n]$ has to be intended as a conjunction $L_1 \land \cdots \land L_n$;
- this conjunction $\land$ is commutative and associative, non-idempotent;
- $t : [L_1, L_2, L_2]$ intuitively means that $t$ can be used once as data of type $L_1$ and twice as data of type $L_2$.

\[
\frac{}{x : P \vdash x : P} \quad \text{ax} \quad \frac{\Gamma \vdash t : [P \rightarrow Q] \quad \Delta \vdash u : P}{\Gamma \uplus \Delta \vdash tu : Q} \quad \ominus
\]

\[
\frac{\Gamma_1, x : P_1 \vdash t : Q_1 \quad \ldots \quad \Gamma_n, x : P_n \vdash t : Q_n}{\Gamma_1 \uplus \cdots \uplus \Gamma_n \vdash \lambda x.t : [P_1 \rightarrow Q_1, \ldots, P_n \rightarrow Q_n]} \quad \lambda
\]

where $\Gamma \uplus \Delta$ is the pointwise multiset sum between type contexts.

This nothing but the CbV counterpart of De Carvalho’s System R for CbN.
Non-idempotent intersection types for CbV (Ehrhard, 2012)

Multi types and linear types are defined by mutual induction:

- **linear types** $L, L' ::=$ $P \rightarrow Q$
- **multi types** $P, Q ::=$ $[L_1, \ldots, L_n]$ ($n \geq 0$)

- There are no base types: their role is played by the empty multiset $[]$ ($n = 0$);
- A multi type $[L_1, \ldots, L_n]$ has to be intended as a conjunction $L_1 \land \cdots \land L_n$;
- This conjunction $\land$ is commutative and associative, non-idempotent;
- $t : [L_1, L_2, L_2]$ intuitively means that $t$ can be used *once* as data of type $L_1$ and *twice* as data of type $L_2$.

\[
\begin{align*}
  \Gamma, x : P & \vdash x : P \quad \text{ax} \\
  \Gamma \vdash t : [P \rightarrow Q] & \quad \Delta \vdash u : P \\
 \hline \\
  \Gamma \uplus \Delta & \vdash tu : Q
\end{align*}
\]

\[
\begin{align*}
  \Gamma_1, x : P_1 & \vdash t : Q_1 \quad n \in \mathbb{N} \\
  \Gamma_n, x : P_n & \vdash t : Q_n \\
 \hline \\
  \Gamma_1 \uplus \cdots \uplus \Gamma_n & \vdash \lambda x.t : [P_1 \rightarrow Q_1, \ldots, P_n \rightarrow Q_n]
\end{align*}
\]

where $\Gamma \uplus \Delta$ is the pointwise multiset sum between type contexts.

This nothing but the CbV counterpart of De Carvalho’s System R for CbN.
A concrete model for Plotkin’s CbV: relational semantics

The semantics of a term is a the set of its types, together with their type contexts.

If $x_1, \ldots, x_n$ are pairwise distinct variables, and $\text{fv}(t) \subseteq \{x_1, \ldots, x_n\}$, we say that $\vec{x} = (x_1, \ldots, x_n)$ is suitable for $t$. Given $\vec{x}$ suitable for $t$, its semantics is:

$$\llbracket t \rrbracket_{\vec{x}} := \{((P_1, \ldots, P_n), Q) \mid \exists \pi \triangleright x_1 : P_1, \ldots, x_n : P_n \vdash t : Q \}.$$  

Notation: $\pi \triangleright \Gamma \vdash t : P$ means "$\pi$ is a type derivation with conclusion $\Gamma \vdash t : P$".

Theorem (invariance under $\rightarrow_{\beta_v}$, Ehrhard 2012)

Let $\vec{x}$ be a suitable list of variables for $t$ and $u$. If $t \rightarrow_{\beta_v} u$ then $\llbracket t \rrbracket_{\vec{x}} = \llbracket u \rrbracket_{\vec{x}}$.

Terminology: A denotational model for a $\lambda$-calculus is adequate if it characterizes all and only the normalisable terms.

$\Rightarrow$ In relational semantics: $[t] = \emptyset$ if and only if $t$ is not normalisable.

$\Rightarrow$ CbV relational semantics is not adequate for Plotkin’s CbV:

$$[\lambda z. \delta(x x) \delta]_x = \emptyset \quad \text{but} \quad (\lambda z. \delta)(x x) \delta \text{ is } \beta_v\text{-normal}.$$
A concrete model for Plotkin’s CbV: relational semantics

The semantics of a term is a the set of its types, together with their type contexts.

If \( x_1, \ldots, x_n \) are pairwise distinct variables, and \( \text{fv}(t) \subseteq \{x_1, \ldots, x_n\} \), we say that \( \vec{x} = (x_1, \ldots, x_n) \) is suitable for \( t \). Given \( \vec{x} \) suitable for \( t \), its semantics is:

\[
\lbrack t \rbrack_{\vec{x}} := \{((P_1, \ldots, P_n), Q) \mid \exists \pi \triangleright x_1 : P_1, \ldots, x_n : P_n \vdash t : Q\}.
\]

Notation: \( \pi \triangleright \Gamma \vdash t : P \) means "\( \pi \) is a type derivation with conclusion \( \Gamma \vdash t : P \)."

Theorem (invariance under \( \rightarrow_{\beta_v} \), Ehrhard 2012)

Let \( \vec{x} \) be a suitable list of variables for \( t \) and \( u \). If \( t \rightarrow_{\beta_v} u \) then \( \lbrack t \rbrack_{\vec{x}} = \lbrack u \rbrack_{\vec{x}} \).

Terminology: A denotational model for a \( \lambda \)-calculus is adequate if it characterizes all and only the normalisable terms.

\( \rightsquigarrow \) In relational semantics: \( \lbrack t \rbrack = \emptyset \) if and only if \( t \) is not normalisable.

\( \rightsquigarrow \) CbV relational semantics is not adequate for Plotkin’s CbV:

\[
\lbrack (\lambda z. \delta)(xx)\delta \rbrack_x = \emptyset \quad \text{but} \quad (\lambda z. \delta)(xx)\delta \ \text{is} \ \beta_v\text{-normal}.
\]
But relational semantics is not a denotational model for $\lambda_{\text{fire}}$!

Relational semantics is not invariant under $\rightarrow_{\beta_f}$!

$$(\lambda z. y)(x x) \rightarrow_{\beta_f} y \quad \text{or} \quad (\lambda z. z z)(x x) \rightarrow_{\beta_f} (x x)(x x)$$

$$\llbracket (\lambda z. y)(x x) \rrbracket_{x, y} \neq \llbracket y \rrbracket_{x} \quad \text{or} \quad \llbracket (\lambda z. z)(x x) \rrbracket_{x, y} \neq \llbracket (x x)(x x) \rrbracket_{x}$$

All counterexamples are due to $\rightarrow_{\beta_i}$, when an inert term is erased or duplicated.

- Relational semantics is not adequate for Plotkin’s CbV;
- Relat. semantics is not a denotational model for $\lambda_{\text{fire}}$ (but it is "adequate").

**Goal:** Forcing relational semantics to be an adequate denotational model for $\lambda_{\text{fire}}$.

**Rmk:** Any denotational model for the CbN $\lambda$-calculus is invariant under $\rightarrow_{\beta_f}$ (since $\rightarrow_{\beta_f} \subseteq \rightarrow_{\beta}$), but there is no hope that it could be adequate for $\lambda_{\text{fire}}$:

in CbN $\llbracket (\lambda x. y)(\delta \delta) \rrbracket_{y} = \llbracket y \rrbracket_{y}$ but $y$ is $\beta_f$-normal, $(\lambda x. y)(\delta \delta)$ is $\beta_f$-diverging.

Methodological principle (what I learned in my PhD)

If there is a mismatch between syntax and semantics, the problem is in the syntax.

$\Rightarrow$ Change the syntax!
But relational semantics is not a denotational model for $\lambda_{\text{fire}}$!

Relational semantics is **not invariant** under $\rightarrow_{\beta_f}$!

\[
(\lambda z. y)(x x) \rightarrow_{\beta_f} y \quad \text{or} \quad (\lambda z. z z)(x x) \rightarrow_{\beta_f} (x x)(x x)
\]

\[
[ (\lambda z. y)(x x) ]_{x, y} \neq [ y ]_x \quad \text{or} \quad [ (\lambda z. z)(x x) ]_{x, y} \neq [ (x x)(x x) ]_x
\]

All counterexamples are due to $\rightarrow_{\beta_i}$, when an inert term is erased or duplicated.

- Relational semantics is **not adequate** for Plotkin’s CbV;
- Relat. semantics is **not a denotational model** for $\lambda_{\text{fire}}$ (but it is "adequate").

**Goal:** Forcing relational semantics to be an adequate denotational model for $\lambda_{\text{fire}}$.

**Rmk:** Any denotational model for the CbN $\lambda$-calculus is invariant under $\rightarrow_{\beta_f}$ (since $\rightarrow_{\beta_f} \subseteq \rightarrow_{\beta}$), but there is no hope that it could be adequate for $\lambda_{\text{fire}}$:

- in CbN $[ (\lambda x. y)(\delta \delta) ]_y = [ y ]_y$ but $y$ is $\beta_f$-normal, $(\lambda x. y)(\delta \delta)$ is $\beta_f$-diverging.

Methodological principle (what I learned in my PhD)

*If there is a mismatch between syntax and semantics, the problem is in the syntax.*

$\Rightarrow$ Change the syntax!
But relational semantics is not a denotational model for $\lambda_{fire}$!

Relational semantics is not invariant under $\rightarrow_{\beta_f}$!

\[
(\lambda z. y)(xx) \rightarrow_{\beta_f} y \quad \text{or} \quad (\lambda z. zz)(xx) \rightarrow_{\beta_f} (xx)(xx)
\]

\[
[(\lambda z. y)(xx)]_{x,y} \neq [y]_x \quad \text{or} \quad [(\lambda z. z)(xx)]_{x,y} \neq [(xx)(xx)]_x
\]

All counterexamples are due to $\rightarrow_{\beta_i}$, when an inert term is erased or duplicated.

- Relational semantics is not adequate for Plotkin’s CbV;
- Relat. semantics is not a denotational model for $\lambda_{fire}$ (but it is "adequate").

**Goal:** Forcing relational semantics to be an adequate denotational model for $\lambda_{fire}$.

**Rmk:** Any denotational model for the CbN $\lambda$-calculus is invariant under $\rightarrow_{\beta_f}$ (since $\rightarrow_{\beta_f} \subseteq \rightarrow_{\beta}$), but there is no hope that it could be adequate for $\lambda_{fire}$:

- in CbN $[(\lambda x. y)(\delta\delta)]_y = [y]_y$ but $y$ is $\beta_f$-normal, $(\lambda x. y)(\delta\delta)$ is $\beta_f$-diverging.

**Methodological principle (what I learned in my PhD)**

If there is a mismatch between syntax and semantics, the problem is in the syntax. 
\[\mapsto\text{ Change the syntax!}\]
Outline

1 Introduction

2 The rise and the fall of the fireball calculus $\lambda_{fire}$

3 The split fireball calculus $\lambda_{split}^{fire}$ and multi types

4 Tight type derivations and exact bounds
The split fireball calculus $\lambda_{\text{split}}_{\text{fire}}$

Goal: A different presentation of $\lambda_{\text{fire}}$ that has an adequate denotational model.

Terms, Values, Fireballs, Inert terms, Right ev. contexts as for the fireball calculus $\lambda_{\text{fire}}$

Environments

$E ::= \epsilon \mid [i/x]:E$

Programs

$p ::= (t, E)$

Rules

$(C\langle(\lambda x.t)v\rangle, E) \rightarrow_{\beta_v} (C\langle t\{v/x}\rangle, E)$

$(C\langle(\lambda x.t)i\rangle, E) \rightarrow_{\beta_i} (C\langle t\rangle, [i/x]:E)$

Reduction

$\rightarrow_{\beta_f} ::= \rightarrow_{\beta_v} \cup \rightarrow_{\beta_i}$

Proposition (open harmony for $\lambda_{\text{split}}_{\text{fire}}$)

A program $p$ is normal iff $p = (f, E)$.

Unfolding (from $\lambda_{\text{split}}_{\text{fire}}$ to $\lambda_{\text{fire}}$): $(t, \epsilon)\downarrow ::= t \quad (t, E@[i/x])\downarrow ::= (t, E)\downarrow\{i/x\}$.

Proposition (strong bisimulation)

1. **Split to plain**: if $p \rightarrow_{\beta_f} q$ then $p\downarrow \rightarrow_{\beta_f} q\downarrow$.

2. **Plain to split**: if $p\downarrow \rightarrow_{\beta_f} u$ then there exists $q$ such that $p \rightarrow_{\beta_f} q$ and $q\downarrow = u$. 
The split fireball calculus $\lambda_{\text{split}}_{\text{fire}}$

**Goal:** A different presentation of $\lambda_{\text{fire}}$ that has an adequate denotational model.

Terms, Values, Fireballs, Inert terms, Right ev. contexts

as for the fireball calculus $\lambda_{\text{fire}}$

**Environments**

$E ::= \epsilon \mid [i/x]:E$

**Programs**

$p ::= (t, E)$

**Rules**

$(C\langle(\lambda x.t)v\rangle, E) \rightarrow_{\beta_v} (C\langle t\{v/x}\rangle, E)$

$(C\langle(\lambda x.t)i\rangle, E) \rightarrow_{\beta_i} (C\langle t\rangle, [i/x]:E)$

**Reduction**

$\rightarrow_{\beta_f} ::= \rightarrow_{\beta_v} \cup \rightarrow_{\beta_i}$

---

**Proposition (open harmony for $\lambda_{\text{split}}_{\text{fire}}$)**

A program $p$ is normal iff $p = (f, E)$.

**Unfolding (from $\lambda_{\text{split}}_{\text{fire}}$ to $\lambda_{\text{fire}}$):**

$(t, \epsilon)_\downarrow ::= t \quad (t, E@[i/x])_\downarrow ::= (t, E)_\downarrow\{i/x\}.$

**Proposition (strong bisimulation)**

1. **Split to plain:** if $p \rightarrow_{\beta_f} q$ then $p_\downarrow \rightarrow_{\beta_f} q_\downarrow$.
2. **Plain to split:** if $p_\downarrow \rightarrow_{\beta_f} u$ then there exists $q$ such that $p \rightarrow_{\beta_f} q$ and $q_\downarrow = u$. 
The multi type system for $\lambda^{\text{split}}_{\text{fire}}$ and sizes

\[
\frac{x : P \vdash x : P}{x : P \vdash x : P} \quad \text{ax}
\]

\[
\frac{\Gamma \vdash t : [P \rightarrow Q] \quad \Delta \vdash u : P}{\Gamma \cup \Delta \vdash tu : Q} \quad \oplus
\]

\[
\frac{\Gamma_1, x : P_1 \vdash t : Q_1 \quad n \in \mathbb{N} \quad \Gamma_n, x : P_n \vdash t : Q_n}{\Gamma_1 \uplus \cdots \uplus \Gamma_n \vdash \lambda x. t : [P_1 \rightarrow Q_1, \ldots, P_n \rightarrow Q_n]} \quad \lambda
\]

\[
\frac{\Gamma \vdash t : P}{\Gamma \vdash (t, \epsilon) : P} \quad \text{es}_{\epsilon}
\]

\[
\frac{\Gamma, x : P \vdash (t, E) : Q \quad \Delta \vdash i : P}{\Gamma \cup \Delta \vdash (t, E \oplus[i/x]) : Q} \quad \text{es}_{\oplus}
\]

1. **Size** $|\pi|$ of a type derivation $\pi$: the number of its $\oplus$ rules.
2. **Size** $|t|$ of a term $t$: the number of its applications not in the scope of $\lambda$'s.
3. **Size** $|p|$ of a program $(t, E)$: $|t|$ plus the sizes of the inert terms in $E$.
4. **Length** $|e|$ of an evaluation $e$: $t \rightarrow^* \beta_f u$: the number of $\beta_f$-steps in $e$. 
Examples and intuitions about multi types for CbV

Example: Let $I := \lambda x.x$

$$
\pi_{II} = \frac{x : [] \vdash x : []}{\vdash I : [[] \rightarrow []]} \frac{\vdash I : []}{\vdash II : []} \frac{\vdash I : []}{\vdash (II, \epsilon) : []} \frac{\vdash (I, \epsilon) : []}{\vdash (I, \epsilon) : []}
$$

Note that $(II, \epsilon) \rightarrow_{\beta_f} (I, \epsilon)$ (with 1 $\beta_f$-step) and $|\pi_{II}| = 1 = |\pi_I| + 1$.

Idea: $[]$ is the type of terms that can be erased.

- In multi types for CbN, every term, even a diverging one, is typable with $[]$.
  - $\rightsquigarrow$ in CbN every term can be erased, even the diverging ones;
  - $\rightsquigarrow$ in CbN adequacy is formulated with respect to non-empty types.

- In multi-types for CbV, terms have to be evaluated before being erased.
  - $\rightsquigarrow$ in CbV, normalisable terms and erasable terms coincide;
  - $\rightsquigarrow$ in CbV, a term is typable if and only if it is typable with $[]$. 
Examples and intuitions about multi types for CbV

Example: Let \( I := \lambda x.x \)

\[
\begin{align*}
\pi_{II} &= \frac{\frac{x : [] \vdash x : []}{\vdash I : [[]] \rightarrow [ ]} \text{ ax}}{\vdash II : [ ]} \\
\pi_I &= \frac{\frac{\vdash I : []}{\vdash (I, \epsilon) : [ ]} \text{ es} \epsilon}
\end{align*}
\]

Note that \((II, \epsilon) \rightarrow_{\beta_f} (I, \epsilon)\) (with 1 \(\beta_f\)-step) and \(|\pi_{II}| = 1 = |\pi_I| + 1\).

Idea: \([\ ]\) is the type of terms that can be erased.

- In multi types for CbN, every term, even a diverging one, is typable with \([\ ]\).
- In CbN every term can be erased, even the diverging ones;
- In CbN adequacy is formulated with respect to non-empty types.

- In multi-types for CbV, terms have to be evaluated before being erased.
- In CbV, normalisable terms and erasable terms coincide;
- In CbV, a term is typable if and only if it is typable with \([\ ]\).
Examples and intuitions about multi types for CbV

Example: Let $I := \lambda x.x$

\[
\pi_{\|} = \frac{x : [] \vdash x : []}{\vdash \| : [] \rightarrow []} \quad \lambda \quad \frac{\vdash I : []}{\vdash \| : []} \quad \otimes \quad \frac{\vdash (\|, \epsilon) : []}{\vdash (I, \epsilon) : []}
\]

\[
\pi_{I} = \frac{\vdash I : []}{\vdash (I, \epsilon) : []}
\]

Note that $(\|, \epsilon) \rightarrow_{\beta_f} (I, \epsilon)$ (with 1 $\beta_f$-step) and $|\pi_{\|}| = 1 = |\pi_{I}| + 1$.

Idea: [] is the type of terms that can be erased.

- In multi types for CbN, every term, even a diverging one, is typable with [].
- In CbN every term can be erased, even the diverging ones;
- In CbN adequacy is formulated with respect to non-empty types.

- In multi-types for CbV, terms have to be evaluated before being erased.
- In CbV, normalisable terms and erasable terms coincide;
- In CbV, a term is typable if and only if it is typable with [].
Examples and intuitions about multi types for CbV

Example: Let $I := \lambda x.x$

\[
\begin{align*}
\pi_{II} &= \frac{x : [] \vdash x : []}{\vdash \text{I} : []} \quad \text{ax} \\
\pi_{II} &= \frac{\vdash 
\text{I} : [[] \to [[]] \quad \lambda}{\vdash \text{II} : []} \quad \forall \quad \vdash (\text{II}, \epsilon) : []} \quad \text{es}_\epsilon \\
\pi_I &= \frac{\vdash \text{I} : []}{\vdash (\text{I}, \epsilon) : []} \quad \text{es}_\epsilon
\end{align*}
\]

Note that $(II, \epsilon) \to_{\beta_f} (I, \epsilon) \text{ (with 1 } \beta_f\text{-step)}$ and $|\pi_{II}| = 1 = |\pi_I| + 1$.

Idea: $[]$ is the type of terms that can be erased.

- In multi types for CbN, every term, even a diverging one, is typable with $[]$.
  - $\leadsto$ in CbN every term can be erased, even the diverging ones;
  - $\leadsto$ in CbN adequacy is formulated with respect to non-empty types.

- In multi-types for CbV, terms have to be evaluated before being erased.
  - $\leadsto$ in CbV, normalisable terms and erasable terms coincide;
  - $\leadsto$ in CbV, a term is typable if and only if it is typable with $[]$. 
Properties of multi types for CbV: correctness

**Proposition (type derivations bound the size of normal forms)**
Let $\pi \triangleright \Gamma \vdash p : P$ be a type derivation for a normal program $p$. Then $|p| \leq |\pi|$.

**Proposition (quantitative subject reduction)**
Let $p$ and $q$ be programs and $\pi \triangleright \Gamma \vdash p : P$ be a type derivation. If $p \rightarrow_{\beta_r} q$ then there exists a type derivation $\sigma \triangleright \Gamma \vdash q : P$ such that $|\pi| = |\sigma| + 1$.

From propositions above, it follows that typability implies termination:

**Theorem (correctness)**
Let $\pi \triangleright \Gamma \vdash p : P$ be a type derivation. Then there is a normalising evaluation $e : p \rightarrow^{*}_{\beta_r} q$ such that $|e| + |q| \leq |\pi|$.
Properties of multi types for CbV: correctness

Proposition (type derivations bound the size of normal forms)
Let $\pi \triangleright \Gamma \vdash p : P$ be a type derivation for a normal program $p$. Then $|p| \leq |\pi|$.

Proposition (quantitative subject reduction)
Let $p$ and $q$ be programs and $\pi \triangleright \Gamma \vdash p : P$ be a type derivation. If $p \rightarrow_{\beta_i} q$ then there exists a type derivation $\sigma \triangleright \Gamma \vdash q : P$ such that $|\pi| = |\sigma| + 1$.

From propositions above, it follows that typability implies termination:

Theorem (correctness)
Let $\pi \triangleright \Gamma \vdash p : P$ be a type derivation. Then there is a normalising evaluation $e : p \rightarrow_{\beta_i}^* q$ such that $|e| + |q| \leq |\pi|$.
Properties of multi types for CbV: completeness

Proposition (normal forms are typable)

1. Normal program: For any normal program, there exists a type derivation \( \pi \triangleright \Gamma \vdash p : P \) for some type context \( \Gamma \) and some multi type \( P \).

2. Inert term: For any multi type \( Q \) and any inert term \( i \), there exists a type derivation \( \sigma \triangleright \Delta \vdash i : Q \) for some type context \( \Delta \).

Proposition (quantitative subject expansion)

Let \( p \) and \( q \) be programs and \( \sigma \triangleright \Gamma \vdash q : P \) be a type derivation. If \( p \xrightarrow{\beta_f} q \) then there exists a type derivation \( \pi \triangleright \Gamma \vdash p : P \) such that \( |\pi| = |\sigma| + 1 \).

From propositions above, it follows that termination implies typability:

Theorem (completeness)

Let \( e : p \xrightarrow{\ast} q \) be a normalising evaluation. Then there is a type derivation \( \pi \triangleright \Gamma \vdash p : P \), and it satisfies \( |e| + |q| \leq |\pi| \).
Properties of multi types for CbV: completeness

**Proposition (normal forms are typable)**

1. **Normal program**: For any normal program, there exists a type derivation \( \pi \triangleright \Gamma \vdash p : P \) for some type context \( \Gamma \) and some multi type \( P \).

2. **Inert term**: For any multi type \( Q \) and any inert term \( i \), there exists a type derivation \( \sigma \triangleright \Delta \vdash i : Q \) for some type context \( \Delta \).

**Proposition (quantitative subject expansion)**

Let \( p \) and \( q \) be programs and \( \sigma \triangleright \Gamma \vdash q : P \) be a type derivation. If \( p \rightarrow_{\beta_f} q \) then there exists a type derivation \( \pi \triangleright \Gamma \vdash p : P \) such that \( |\pi| = |\sigma| + 1 \).

From propositions above, it follows that termination implies typability:

**Theorem (completeness)**

Let \( e : p \rightarrow^{*}_{\beta_f} q \) be a normalising evaluation. Then there is a type derivation \( \pi \triangleright \Gamma \vdash p : P \), and it satisfies \( |e| + |q| \leq |\pi| \).
Relational semantics is an adequate model for $\lambda_{\text{split fire}}$

**Corollary (invariance)**

Let $\vec{x}$ be a suitable list of variables for $p$ and $q$. If $p \rightarrow_{\beta_f} q$ then $[p]_{\vec{x}} = [q]_{\vec{x}}$.

**Corollary (adequacy)**

Let $\vec{x}$ be a suitable list of variables for $p$. The following are equivalent:

1. **Termination**: $p$ is $\beta_f$-normalizable;
2. **Typability**: there is a type derivation $\pi \triangleright \Gamma \vdash p : P$ for some $\Gamma$ and $P$;
3. **Non-empty denotation**: $[p]_{\vec{x}} \neq \emptyset$.

**Rmk:** $\lambda_{\text{split fire}}$ and $\lambda_{\text{fire}}$ and Plotkin's CbV are the same if restricted to closed terms.

$\Rightarrow$ Adequacy theorem above shows also that relational semantics is adequate for closed Plotkin's CbV.
Relational semantics is an adequate model for $\lambda_{\text{split}}$.

**Corollary (invariance)**

Let $\vec{x}$ be a suitable list of variables for $p$ and $q$. If $p \rightarrow_{\beta_f} q$ then $[p]_{\vec{x}} = [q]_{\vec{x}}$.

**Corollary (adequacy)**

Let $\vec{x}$ be a suitable list of variables for $p$. The following are equivalent:

1. **Termination:** $p$ is $\beta_f$-normalizable;
2. **Typability:** there is a type derivation $\pi \triangleright \Gamma \vdash p : P$ for some $\Gamma$ and $P$;
3. **Non-empty denotation:** $[p]_{\vec{x}} \neq \emptyset$.

**Rmk:** $\lambda_{\text{split}}$ and $\lambda_{\text{fire}}$ and Plotkin’s CbV are the same if restricted to closed terms.

$\Rightarrow$ Adequacy theorem above shows also that relational semantics is adequate for closed Plotkin’s CbV.
Outline

1. Introduction

2. The rise and the fall of the fireball calculus $\lambda_{\text{fire}}$

3. The split fireball calculus $\lambda_{\text{fire}}^{\text{split}}$ and multi types

4. Tight type derivations and exact bounds
Tight type derivations

In CbN, multi-types can provide exact bounds for the evaluation length and for the size of normal forms.

Can we extract this quantitative information in Open CbV as well? Yes!

We define two subsets of linear types and multi types:

- inert linear types: \( L^i ::= [\_] \rightarrow P^i \)
- inert multi types: \( P^i ::= [L^i_1, \ldots, L^i_n] \) \((n \geq 0)\)

A type context \( \Gamma \) is inert if it assigns only inert multi types to variables.
A type derivation \( \pi \triangleright \Gamma \vdash p : P \) is

- inert if \( \Gamma \) is a inert type context;
- tight if \( \pi \) is inert and \( P = [\_] \).

Intuition: Any \( \beta_f \)-normalisable (i.e. any typable) program is typable with type \( [\_] \). More precisely, with a tight type derivation.
Tight type derivations

In CbN, multi-types can provide exact bounds for the evaluation length and for the size of normal forms.

Can we extract this quantitative information in Open CbV as well? Yes!

We define two subsets of linear types and multi types:

\[
L^i ::= [] \rightarrow P^i \quad \text{inert linear types} \\
P^i ::= [L_1^i, \ldots, L_n^i] \quad (n \geq 0) \quad \text{inert multi types}
\]

A type context \( \Gamma \) is inert if it assigns only inert multi types to variables. A type derivation \( \pi \triangleright \Gamma \vdash p : P \) is

- inert if \( \Gamma \) is a inert type context;
- tight if \( \pi \) is inert and \( P = [] \).

Intuition: Any \( \beta_f \)-normalisable (i.e. any typable) program is typable with type \([\]\). More precisely, with a tight type derivation.
Exact bounds for Open CbV

Tight type derivations have a nice property with respect to normal forms:

**Lemma (tight type derivations are minimal)**

If $\pi \triangleright \Gamma \vdash p : []$ is a *tight* type derivation and $p$ is $\beta_f$-normal, then $|p| = |\pi|$ and $|\pi|$ is minimal among the type derivations of $p$.

We can refine correctness and completeness with *exact* quantitative information:

**Theorem (tight correctness)**

If $\pi \triangleright \Gamma \vdash p : []$ is a *tight* type derivation, then there is a normalising evaluation $e : p \rightarrow^{*}_{\beta_f} q$ such that $|e| + |q| = |\pi|$. And if $q$ is a value, then $|e| = |\pi|$.

**Theorem (tight completeness)**

If $e : p \rightarrow^{*}_{\beta_f} q$ is a normalizing evaluation, then there is a *tight* type derivation $\pi \triangleright \Gamma \vdash p : []$, and it satisfies $|e| + |q| = |\pi|$. And if $q$ is a value, then $|e| = |\pi|$.
Exact bounds for Open CbV

Tight type derivations have a nice property with respect to normal forms:

**Lemma (tight type derivations are minimal)**

If \( \pi \triangleright \Gamma \vdash p : [] \) is a tight type derivation and \( p \) is \( \beta_f \)-normal, then \( |p| = |\pi| \) and \( |\pi| \) is minimal among the type derivations of \( p \).

We can refine correctness and completeness with exact quantitative information:

**Theorem (tight correctness)**

If \( \pi \triangleright \Gamma \vdash p : [] \) is a tight type derivation, then there is a normalising evaluation \( e : p \rightarrow^{*}_{\beta_f} q \) such that \( |e| + |q| = |\pi| \). And if \( q \) is a value, then \( |e| = |\pi| \).

**Theorem (tight completeness)**

If \( e : p \rightarrow^{*}_{\beta_f} q \) is a normalizing evaluation, then there is a tight type derivation \( \pi \triangleright \Gamma \vdash p : [] \), and it satisfies \( |e| + |q| = |\pi| \). And if \( q \) is a value, then \( |e| = |\pi| \).
What are we counting?

\( \lambda_{\text{split}} \) mimics LL proof-nets behavior via "boring" translation \( A \Rightarrow B \xrightarrow{()} !A^v \dashv !B^v. \)

\[
(C\langle(\lambda x.t)v\rangle, E) \rightarrow_{\beta_v} (C\langle t\{v/x\}\rangle, E) \quad (C\langle(\lambda x.t)i\rangle, E) \rightarrow_{\beta_i} (C\langle t\rangle, [i/x]: E)
\]

- a \( \beta_v \)-step corresponds to a multiplicative followed by an exponential step;
- a \( \beta_i \)-step corresponds to a multiplicative step.

\( \Rightarrow \) The number of \( \beta_f \)-steps is the number of multiplicative steps in LL proof-nets.

\( \Rightarrow \) Tight derivations count the number of multiplicative steps to reach a normal form of a LL proof-net.

Rmk: The number of \( \beta_f \)-steps is a reasonable cost model (\( \lambda_{\text{fire}}^{\text{split}} \) can be implemented on a RAM with an overhead that is linear in the number of \( \beta_f \)-step).

Summing up: At least in the CbV fragment of LL:

- the multiplicative step is the computationally meaningful cut-elimination step;
- the exponential step just allows evaluation to go on.
What are we counting?

\( \text{split}_\text{fire} \) mimics LL proof-nets behavior via "boring" translation \( A \Rightarrow B \xrightarrow{\cdot} !A \xrightarrow{\circ} !B. \)

\[
(C\langle(\lambda x.t)v\rangle, E) \rightarrow_{\beta_v} (C\langle t\{v/x\}\rangle, E) \quad (C\langle(\lambda x.t)i\rangle, E) \rightarrow_{\beta_i} (C\langle t\rangle, [i/x]:E)
\]

- a \( \beta_v \)-step corresponds to a multiplicative followed by an exponential step;
- a \( \beta_i \)-step corresponds to a multiplicative step.

\( \Rightarrow \) The number of \( \beta_f \)-steps is the number of multiplicative steps in LL proof-nets.
\( \Rightarrow \) Tight derivations count the number of multiplicative steps to reach a normal form of a LL proof-net.

**Rmk:** The number of \( \beta_f \)-steps is a reasonable cost model (\( \text{split}_\text{fire} \) can be implemented on a RAM with an overhead that is linear in the number of \( \beta_f \)-step).

**Summing up:** At least in the CbV fragment of LL:
- the multiplicative step is the computationally meaningful cut-elimination step;
- the exponential step just allows evaluation to go on.