Call-by-value non-determinism in a linear logic type discipline

Alejandro Díaz-Caro*
Université Paris-Ouest & INRIA

Giulio Manzonetto
LIPN, Université Paris 13

Michele Pagani
LIPN, Université Paris 13

Symposium on Logical Foundations of Computer Science
San Diego, California, U.S.A., January 6-8, 2013

* Supported by the DIGITEO project 2011-070D "ALAL"

Intersection types discipline [Coppo-Dezani'78]

$$M : \alpha \cap \beta$$

M **enjoys both properties** α **and** β

With this idea in mind intersection is idempotent $\alpha \cap \alpha = \alpha$.

Intersection types discipline [Coppo-Dezani'78]

$$M : \alpha \cap \beta$$

M enjoys both properties α and β

With this idea in mind intersection is idempotent $\alpha \cap \alpha = \alpha$.

Used to capture various notions of termination:
Head, Weak and Strong normalisation [Coppo-Dezani'78, Sallé'80]

Intersection types discipline [Coppo-Dezani'78]

$$M : \alpha \cap \beta$$

M enjoys both properties α and β

With this idea in mind intersection is idempotent $\alpha \cap \alpha = \alpha$.

Used to capture various notions of termination:
Head, Weak and Strong normalisation [Coppo-Dezani'78, Sallé'80]

Resource-aware intersection types [De Carvalho'07]

Let us change point of view:

$$M : \alpha \cap \beta$$

M will be called once as data of type α and once as data of type β

Hence $\alpha \cap \alpha \neq \alpha \quad \Longrightarrow \quad$ Multisets

Used to capture quantitative properties of programs, e.g.:
CBN λ-calculus: number of linear head-reduction steps [De Carvalho'07]
CBV λ-calculus: number of weak head-reduction steps [Ehrhard'12]

Intersection types discipline [Coppo-Dezani'78]

$$M : \alpha \cap \beta$$

M enjoys both properties α and β

With this idea in mind intersection is idempotent $\alpha \cap \alpha = \alpha$.

Used to capture various notions of termination:
Head, Weak and Strong normalisation [Coppo-Dezani'78, Sallé'80]

Resource-aware intersection types [De Carvalho'07]

Let us change point of view:

$$M : \alpha \cap \beta$$

M will be called once as data of type α and once as data of type β

Hence $\alpha \cap \alpha \neq \alpha \quad \Longrightarrow \quad$ Multisets

Used to capture quantitative properties of programs, e.g.:
CBN λ-calculus: number of linear head-reduction steps [De Carvalho'07]
CBV λ-calculus: number of weak head-reduction steps [Ehrhard'12]

Our goal: extend Ehrhard's system with non-determinism

May/Must-convergent non-determinism

Consider the CBV λ-calculus extended with...

Non-deterministic choice

$M + N$ The machine choses either M or N

Parallel composition

$M \parallel N$ The machine interleaves reductions in M and in N

May/Must-convergent non-determinism

Consider the CBV λ-calculus extended with...

Non-deterministic choice

$M + N$ The machine choses either M or N

- The non-deterministic choice $M + N$ is *may*-convergent:
  it converges if either M or N converges

Parallel composition

$M \parallel N$ The machine interleaves reductions in M and in N

- The parallel composition $M \parallel N$ is *must*-convergent:
  it converges if both M and N do

$\Lambda_{+\parallel}$: Its syntax and operational semantics

Grammar of $\Lambda_{+\parallel}$ terms

Terms:	$M, N, P, Q ::= \quad V \mid MN \mid M + N \mid M \parallel N$
Values:	$V ::= \quad x \mid \lambda x.M$

Reduction semantics

β_v*-reduction*	+*-reductions*	$\parallel$*-reductions*
$(\lambda x.M)V \to M[V/x]$	$M + N \to M$	$(M \parallel N)P \to MP \parallel NP$
	$M + N \to N$	$V(M \parallel N) \to VM \parallel VN$

+ Contextual rules selecting the *head* redex...

The reduction is *lazy* (it does not reduce under λ-abstractions)

$\Lambda_{+\|}$: Its syntax and operational semantics

Grammar of $\Lambda_{+\|}$ terms

Terms: $M, N, P, Q ::= V \mid MN \mid M + N \mid M \| N$

Values: $V ::= x \mid \lambda x.M$

Reduction semantics

β_v-reduction	$+$-reductions	$\|$-reductions
$(\lambda x.M)V \to M[V/x]$	$M + N \to M$	$(M \| N)P \to MP \| NP$
	$M + N \to N$	$V(M \| N) \to VM \| VN$

$+$ Contextual rules selecting the *head* redex...

The reduction is *lazy* (it does not reduce under λ-abstractions)

Convergence

$$M \text{ converges} \quad \Leftrightarrow \quad M \to^* V_1 \| \cdots \| V_n$$

Examples and remarks

Application is bilinear

$$(M + M')(N + N') \quad \overset{op}{\equiv} \quad MN + MN' + M'N + M'N'$$

. . . but λ-abstraction is not

$$\lambda x.(M + N) \quad \overset{op}{\not\equiv} \quad \lambda x.M + \lambda x.N$$

Examples and remarks

Application is bilinear

$$(M + M')(N + N') \quad \overset{op}{\equiv} \quad MN + MN' + M'N + M'N'$$

... but λ-abstraction is not

$$\lambda x.(M + N) \quad \overset{op}{\not\equiv} \quad \lambda x.M + \lambda x.N$$

Example of parallel composition and non-deterministic choice

$$(\lambda x.(x \parallel x))(V + V') \text{ converges to either } V \parallel V \text{ or } V' \parallel V'$$

$$(\lambda x.(x + x))(V \parallel V') \text{ converges to } V \parallel V' \text{ only}$$

Linear logic based type system

Translation: Intuitionistic Logic $\mapsto$ Polarized fragment of LL

$$\iota^v = \iota, \qquad (\alpha \to \beta)^v = \alpha^c \multimap \beta^{\parallel}, \qquad \alpha^c = !\alpha^v, \qquad \alpha^{\parallel} = ?\alpha^c$$

Based on [Ehrhard'12], based on second Girard's translation.

Intuitions from the relational semantics of LL

- The type for computations $(\cdot)^c$ is a multiset $[\alpha_1^v, \ldots, \alpha_n^v]$ of value types (representing n calls to a single value of type α_i^v),
- The type of parallel compositions $(\cdot)^{\parallel}$ is another multiset $[\alpha_1^c, \ldots, \alpha_n^c]$ of types of each term in the composition,
- The type for values $(\cdot)^v$ are either basic types or functional types,
- A functional type in this system is a pair $(\alpha^c, [\alpha_1^c, \ldots, \alpha_n^c])$.

Linear logic based type system

Translation: Intuitionistic Logic $\mapsto$ Polarized fragment of LL

$$\iota^v = \iota, \qquad (\alpha \to \beta)^v = \alpha^c \multimap \beta^{\|}, \qquad \alpha^c = !\alpha^v, \qquad \alpha^{\|} = ?\alpha^c$$

Based on `[Ehrhard'12]`, based on second Girard's translation.

Intuitions from the relational semantics of LL

- The type for computations $(\cdot)^c$ is a multiset $[\alpha_1^v, \ldots, \alpha_n^v]$ of value types (representing n calls to a single value of type α_i^v),
- The type of parallel compositions $(\cdot)^{\|}$ is another multiset $[\alpha_1^c, \ldots, \alpha_n^c]$ of types of each term in the composition,
- The type for values $(\cdot)^v$ are either basic types or functional types,
- A functional type in this system is a pair $(\alpha^c, [\alpha_1^c, \ldots, \alpha_n^c])$.

Notation

First multiset layer	$\longrightarrow$	$\otimes$
Second multiset layer	$\longrightarrow$	$\parr$
Functional type $(\alpha^c, [\alpha_1^c, \ldots, \alpha_n^c])$	$\longrightarrow$	$\alpha^c \multimap \alpha_1^c \parr \cdots \parr \alpha_n^c$
Empty computational multiset	$\longrightarrow$	$\mathbf{1}$

Linear logic based type system (cont.)

Grammar of Types:

parallel-types:	$\alpha, \beta ::= \alpha \parr \beta \mid \tau$
computational-types:	$\tau, \rho ::= \mathbf{1} \mid \tau \otimes \rho \mid \tau \multimap \alpha$

$\otimes$	tensor product	associative and commutative
$\parr$	par	associative and commutative
$\mathbf{1}$	neutral element of $\otimes$	

Linear logic based type system (cont.)

Grammar of Types:

parallel-types:	$\alpha, \beta ::=$	$\alpha \invamp \beta \mid \tau$
computational-types:	$\tau, \rho ::=$	$\mathbf{1} \mid \tau \otimes \rho \mid \tau \multimap \alpha$

$$\left.\begin{array}{ll} \otimes & \text{tensor product} \\ \invamp & \text{par} \end{array}\right\} \text{associative and commutative}$$

$$\begin{array}{ll} \mathbf{1} & \text{neutral element of } \otimes \end{array}$$

Type environments:

$\Gamma = x_1 : \tau_1, \ldots, x_n : \tau_n$ represents the map

$$\Gamma(y) = \begin{cases} \tau_i & \text{if } y = x_i, \\ \mathbf{1} & \text{otherwise.} \end{cases}$$

Tensor is extended to environments pointwise $(\Gamma \otimes \Delta)(x) = \Gamma(x) \otimes \Delta(x)$.

Linear logic based type system (cont.)

Type inference rules

$$\frac{\Delta \vdash M : \alpha}{\Delta \vdash M + N : \alpha}\,+_\ell \qquad \frac{\Delta \vdash N : \alpha}{\Delta \vdash M + N : \alpha}\,+_r$$

$+$ is may-convergent, so it is enough that one term is typable

Linear logic based type system (cont.)

Type inference rules

$$\frac{\Delta \vdash M : \alpha}{\Delta \vdash M + N : \alpha} +_\ell \qquad \frac{\Delta \vdash N : \alpha}{\Delta \vdash M + N : \alpha} +_r$$

$+$ is may-convergent, so it is enough that one term is typable

$$\frac{\Delta \vdash M : \alpha_1 \qquad \Gamma \vdash N : \alpha_2}{\Delta \otimes \Gamma \vdash M \parallel N : \alpha_1 \parr \alpha_2} \parallel_I$$

$\parallel$ is must-convergent, so both components must be typable

Linear logic based type system (cont.)

Type inference rules

$$\frac{\Delta \vdash M : \alpha}{\Delta \vdash M + N : \alpha}\,+_\ell \qquad \frac{\Delta \vdash N : \alpha}{\Delta \vdash M + N : \alpha}\,+_r$$

$+$ is may-convergent, so it is enough that one term is typable

$$\frac{\Delta \vdash M : \alpha_1 \qquad \Gamma \vdash N : \alpha_2}{\Delta \otimes \Gamma \vdash M \parallel N : \alpha_1 \parr \alpha_2}\,\parallel_I$$

$\parallel$ is must-convergent, so both components must be typable

$$\frac{\Delta \vdash M : \bigparr_{i=1}^{k} \bigotimes_{j=1}^{n_i} (\tau_{ij} \multimap \alpha_{ij}) \qquad \Gamma_i \vdash N : \bigparr_{j=1}^{n_i} \tau_{ij} \quad 1 \leq i \leq k}{\Delta \otimes \bigotimes_{i=1}^{k} \Gamma_i \vdash MN : \bigparr_{i=1}^{k} \bigparr_{j=1}^{n_i} \alpha_{ij}}\,\multimap_E \qquad \begin{array}{c} k \geq 1 \\ n_i \geq 1 \end{array}$$

It reflects the distribution of the parallel operator over the application

Linear logic based type system (cont.)

Type inference rules

$$\frac{\Delta \vdash M : \alpha}{\Delta \vdash M + N : \alpha} +_\ell \qquad \frac{\Delta \vdash N : \alpha}{\Delta \vdash M + N : \alpha} +_r$$

$+$ is may-convergent, so it is enough that one term is typable

$$\frac{\Delta \vdash M : \alpha_1 \qquad \Gamma \vdash N : \alpha_2}{\Delta \otimes \Gamma \vdash M \parallel N : \alpha_1 \parr \alpha_2} \parallel_I$$

$\parallel$ is must-convergent, so both components must be typable

$$\frac{\Delta \vdash M : \bigparr_{i=1}^{k} \bigotimes_{j=1}^{n_i} (\tau_{ij} \multimap \alpha_{ij}) \qquad \Gamma_i \vdash N : \bigparr_{j=1}^{n_i} \tau_{ij} \quad 1 \leq i \leq k}{\Delta \otimes \bigotimes_{i=1}^{k} \Gamma_i \vdash MN : \bigparr_{i=1}^{k} \bigparr_{j=1}^{n_i} \alpha_{ij}} \multimap_E \qquad \begin{array}{l} k \geq 1 \\ n_i \geq 1 \end{array}$$

It reflects the distribution of the parallel operator over the application

$$\frac{}{x : \tau \vdash x : \tau}\; ax \qquad \frac{\Delta_i, x : \tau_i \vdash M : \alpha_i \qquad 1 \leq i \leq n}{\bigotimes_{i=1}^{n} \Delta_i \vdash \lambda x.M : \bigotimes_{i=1}^{n} (\tau_i \multimap \alpha_i)} \multimap_I \qquad n \geq 0$$

The axiom and the intersection type for values respectively

Examples

$$\Delta = x : (\tau_1 \multimap \alpha_1) \otimes (\tau_2 \multimap \alpha_2) \qquad \Gamma = y : \tau_1, y' : \tau_2$$

$$\dfrac{\Delta \;\vdash\; x : (\tau_1 \multimap \alpha_1) \otimes (\tau_2 \multimap \alpha_2) \qquad \Gamma \;\vdash\; y \parallel y' : \tau_1 \parr \tau_2}{\Delta \otimes \Gamma \;\vdash\; x(y \parallel y') : \alpha_1 \parr \alpha_2} \multimap_E$$

$$x(y \parallel y') \to xy \parallel xy'$$

Examples

$$\Delta = x : (\tau_1 \multimap \alpha_1) \otimes (\tau_2 \multimap \alpha_2) \qquad \Gamma = y : \tau_1, y' : \tau_2$$

$$\frac{\Delta \ \vdash \ x : (\tau_1 \multimap \alpha_1) \otimes (\tau_2 \multimap \alpha_2) \qquad \Gamma \ \vdash \ y \parallel y' : \tau_1 \parr \tau_2}{\Delta \otimes \Gamma \ \vdash \ x(y \parallel y') : \alpha_1 \parr \alpha_2} \multimap_E$$

$$x(y \parallel y') \to xy \parallel xy'$$

$$\Delta' = x' : (\tau_1 \multimap \alpha_3) \otimes (\tau_2 \multimap \alpha_4)$$

$$\frac{\begin{array}{c}\Delta \otimes \Delta' \vdash x \parallel x' : ((\tau_1 \multimap \alpha_1) \otimes (\tau_2 \multimap \alpha_2)) \parr ((\tau_1 \multimap \alpha_3) \otimes (\tau_2 \multimap \alpha_4)) \\ \Gamma \ \vdash \ y \parallel y' : \tau_1 \parr \tau_2 \qquad \Gamma \ \vdash \ y \parallel y' : \tau_1 \parr \tau_2\end{array}}{\Delta \otimes \Delta' \otimes \Gamma \otimes \Gamma \vdash (x \parallel x')(y \parallel y') : \alpha_1 \parr \alpha_2 \parr \alpha_3 \parr \alpha_4} \multimap_E$$

$$(x \parallel x')(y \parallel y') \to^* xy \parallel xy' \parallel x'y \parallel x'y'$$

Measuring derivation trees

$$\pi = \frac{}{S}\, ax \qquad |\pi| = 0$$

$$\pi = \frac{\pi_1 \quad \cdots \quad \pi_n}{S} \multimap_I \qquad |\pi| = \sum_{i=1}^{n} |\pi_i|$$

$$\pi = \frac{\pi_1 \qquad \pi_2}{S} \parallel_I \qquad |\pi| = |\pi_1| + |\pi_2|$$

$$\pi = \frac{\pi_0 \quad \pi_1 \ldots \pi_k}{S} \multimap_E \quad n_i \geq 1 \qquad |\pi| = \sum_{i=0}^{k} |\pi_i| + (\sum_{i=1}^{k} 2n_i) - 1$$

$$\pi = \frac{\pi'}{S} +_\ell \quad \text{or} \quad \pi = \frac{\pi'}{S} +_r \qquad |\pi| = |\pi'| + 1$$

Only $\multimap_E$, $+_\ell$ and $+_r$ type redexes $\left[\begin{array}{l}\beta_v \text{ and } \parallel \text{ redexes are typed by } \multimap_E \\ + \text{ redexes by } +_\ell \text{ and } +_r\end{array}\right]$

Each $+_\ell$ and $+_r$ counts for 1 because a $+$-red. does not create new rules in the derivation typing the contractum

$\multimap_E$ counts the number of "active" connectives in the principal premise

Measuring derivation trees (cont.)

$$\frac{\Delta \vdash M : \bigparr_{i=1}^{k} \bigotimes_{j=1}^{n_i} (\tau_{ij} \multimap \alpha_{ij}) \qquad \Gamma_i \vdash N : \bigparr_{j=1}^{n_i} \tau_{ij} \quad 1 \leq i \leq k}{\Delta \otimes \bigotimes_{i=1}^{k} \Gamma_i \vdash MN : \bigparr_{i=1}^{k} \bigparr_{j=1}^{n_i} \alpha_{ij}} \multimap_E$$

$$\underbrace{\sum_{i=1}^{k} n_i}_{\multimap\text{'s}} + \underbrace{\sum_{i=1}^{k} (n_i - 1)}_{\otimes\text{'s}} + \underbrace{(k-1)}_{\parr\text{'s}} = (\sum_{i=1}^{k} 2n_i) - 1$$

The $\|$-reduction creates two new $\multimap_E$ rules in the derivation typing the contractum

The measure decreases because the sum of their weights is less than the weight of the eliminated rule

Properties of the type system

Our system enjoys a quantitative version of standard properties.

Subject reduction

Let $\pi = \Delta \vdash M : \alpha$

- **If** $M \to N$ without $+$-red. **then** $\exists \pi' = \Delta \vdash N : \alpha$
- **If** $M \to N_1$ and $M \to N_2$ with $+$-red.
  then $\exists \pi' = \Delta \vdash N_1 : \alpha$ or $\pi' = \Delta \vdash N_2 : \alpha$

In both cases, $|\pi'| = |\pi| - 1$

Properties of the type system

Our system enjoys a quantitative version of standard properties.

Subject reduction

Let $\pi = \Delta \vdash M : \alpha$

- **If** $M \to N$ without $+$-red. **then** $\exists \pi' = \Delta \vdash N : \alpha$
- **If** $M \to N_1$ and $M \to N_2$ with $+$-red.
  then $\exists \pi' = \Delta \vdash N_1 : \alpha$ or $\pi' = \Delta \vdash N_2 : \alpha$

In both cases, $|\pi'| = |\pi| - 1$

Subject expansion

If $M \to N$ and $\pi = \Delta \vdash N : \alpha$
then $\exists \pi' = \Delta \vdash M : \alpha$ s.t. $|\pi'| = |\pi| + 1$

Properties of the type system

Our system enjoys a quantitative version of standard properties.

Subject reduction

Let $\pi = \Delta \vdash M : \alpha$

- **If** $M \to N$ without $+$-red. **then** $\exists \pi' = \Delta \vdash N : \alpha$
- **If** $M \to N_1$ and $M \to N_2$ with $+$-red.
  then $\exists \pi' = \Delta \vdash N_1 : \alpha$ or $\pi' = \Delta \vdash N_2 : \alpha$

In both cases, $|\pi'| = |\pi| - 1$

Subject expansion

If $M \to N$ and $\pi = \Delta \vdash N : \alpha$
then $\exists \pi' = \Delta \vdash M : \alpha$ s.t. $|\pi'| = |\pi| + 1$

Characterization of convergence

Let M closed. M typable $\Leftrightarrow$ M converges

Can we say anything more quantitative?

Combinatorial proof of normalization

Measure

Let M be a closed term. If π is a derivation of

$$\vdash M : \alpha,$$

then $|\pi|$ gives a bound on the number of steps M converges.

More precisely...

Exact bound

Let M be a closed term. If π is a derivation of

$$\vdash M : \mathbf{1} \parr \cdots \parr \mathbf{1},$$

then M reaches its normal form in exactly $|\pi|$ steps

Properties of the underlying relational model

Let M, N and $\vec{P}$ be closed terms.

Definitions

- A closed term M is interpreted by $\quad [\![M]\!] = \{\alpha \mid\ \vdash M : \alpha\}$
- $M \sqsubseteq N \quad \text{iff} \quad \forall \vec{P} \quad \left[M\vec{P} \text{ converges} \quad \Rightarrow \quad N\vec{P} \text{ converges} \right]$

As a corollary of the Convergence Theorem we get:

Adequacy

$$[\![M]\!] \subseteq [\![N]\!] \quad \text{implies} \quad M \sqsubseteq N$$

Lack of full abstraction

Lack of full abstraction

$$M \sqsubseteq N \qquad \text{does not imply} \qquad [\![M]\!] \subseteq [\![N]\!]$$

CBV λ-calculus admits the creation of an ogre

$$\mathbf{Y}^\star = \Delta^\star\Delta^\star \text{ where } \Delta^\star = \lambda xy.xx.$$

Remark: The ogre $\mathbf{Y}^\star$ is a top of $\sqsubseteq$:

$$\mathbf{Y}^\star V\vec{V}' \to (\lambda y.\mathbf{Y}^\star)V\vec{V}' \to \mathbf{Y}^\star\vec{V}' \to \cdots \to \mathbf{Y}^\star.$$

All types of $\mathbf{Y}^\star$ have shape $\alpha = \bigotimes_{i=0}^{n}(\mathbf{1} \multimap \alpha_i)$.

Lack of full abstraction

Lack of full abstraction

$M \sqsubseteq N$ does not imply $[\![M]\!] \subseteq [\![N]\!]$

CBV λ-calculus admits the creation of an ogre

$$\mathbf{Y}^\star = \Delta^\star\Delta^\star \text{ where } \Delta^\star = \lambda xy.xx.$$

Remark: The ogre $\mathbf{Y}^\star$ is a top of $\sqsubseteq$:

$$\mathbf{Y}^\star V \vec{V}' \to (\lambda y.\mathbf{Y}^\star) V \vec{V}' \to \mathbf{Y}^\star \vec{V}' \to \cdots \to \mathbf{Y}^\star.$$

All types of $\mathbf{Y}^\star$ have shape $\alpha = \bigotimes_{i=0}^{n}(\mathbf{1} \multimap \alpha_i)$.

Counterexample (independent from + and $\parallel$)

Let $\mathbf{I} = \lambda x.x$, then

$$\mathbf{I} \sqsubseteq \mathbf{Y}^\star, \text{ while } [\![\mathbf{I}]\!] \not\subseteq [\![\mathbf{Y}^\star]\!]$$

since $(\mathbf{1} \multimap \mathbf{1}) \multimap (\mathbf{1} \multimap \mathbf{1}) \in [\![\mathbf{I}]\!] - [\![\mathbf{Y}^\star]\!]$

Summarising

- We introduced a call-by-value non-deterministic λ-calculus with a type system ensuring convergence
- The type system gives a bound of the length of the lazy cbv reduction sequences (exact when the typing is minimal)
- We show an adequate (but not fully abstract) model capturing the type system