Geometry of Interaction for an effectful concurrent λ-calculus

Yann Hamdaoui

IRIF, Universit Paris 7 - Diderot

August 28, 2016

Plan

GoI as an abstract machine

GoI has several advantages as an implementation mechanism :

- Fine-grained : (ME)LL proofs nets as "functional assembly language"
- Expressive : various computational paradigm
- Comes with a semantic
- Local computations : compositional & parallelizable
- Compact runtime and object code

Challenges

Evaluation strategy	Call-by-name
Effects & Recursion	Simply typed pure λ-calculus
Concurrency	Parallelism without interaction
Efficiency	Exponential executions

Geometry of Synchronization

Geometry of Synchronization [Dal Lago, Hasuo, Faggian, Valiron, Yoshimizu] adresses several points :

- Support both for call-by-value and call-by-name
- Parallel evaluation
- Recursion

Expressive power

What about effects ?

- Add ad-hoc features to automata
- Derive generic automata [Hoshino, Muroya, Hasuo]
- Monads [Tranquilli]

Goal

Design a parallel CBV GoI machine for a concurrent effectful λ-calculus.

The main ingredients we focus on are

- A new proof nets language (assembly) that handles naturally non-determinism
- A translation of the calculus in these proof nets
- A simulation theorem

Then a straightforward extension of the GoI/GoS provides the desired machine

Plan

Concurrent λ-calculus

We use Amadio's concurrent λ-calculus with regions

Terms

variables	$x, y, \ldots$		
regions	$r, s, \ldots$		
values	V	$::=$	$x \mid * \mid \lambda x.M$
terms	M	$::=$	$V \mid M\ M \mid \text{get}(r) \mid \text{set}(r, V) \mid M \parallel M$
stores	S	$::=$	$r \Leftarrow V \mid (S \parallel S)$
programs	P	$::=$	$M \mid S \mid (P \parallel P)$

Structural rules

$$P' \parallel P = P \parallel P'$$
$$(P \parallel P') \parallel P'' = P \parallel (P' \parallel P'')$$

Reduction of the calculus

Evaluation contexts

$$
\begin{array}{lcl}
E & ::= & [.] \mid E\ M \mid V\ E \\
C & ::= & [.] \mid (C \parallel P) \mid (P \parallel C)
\end{array}
$$

Reduction rules

$$
\begin{array}{lrcl}
(\beta_v) & C[E[(\lambda x.M)\ V] & \rightarrow & C[E[M[V/x]]] \\
(\mathtt{get}) & C[E[\mathrm{get}(r)]] \parallel r \Leftarrow V & \rightarrow & C[E[V]] \parallel r \Leftarrow V \\
(\mathtt{set}) & C[E[\mathrm{set}(r, V)]] & \rightarrow & C[E[*]] \parallel r \Leftarrow V
\end{array}
$$

Reduction of the calculus

$$(\lambda x.\mathrm{get}(r))\ * \parallel \mathrm{set}(r,1) \parallel \mathrm{set}(r,2)$$

$$\downarrow$$

$$(\lambda x.\mathrm{get}(r))\ * \parallel * \parallel * \parallel r \Leftarrow 1 \parallel r \Leftarrow 2$$

$$\swarrow \qquad \searrow$$

$$1 \parallel * \parallel * \parallel r \Leftarrow 1 \parallel r \Leftarrow 2 \qquad\qquad 2 \parallel * \parallel * \parallel r \Leftarrow 1 \parallel r \Leftarrow 2$$

Reduction of the calculus

$$\mathrm{set}(r,1);\mathrm{set}(r,2);\mathrm{get}(r) = \lambda x.(\lambda y.(\mathrm{get}(r))\ \mathrm{set}(r,2))\ \mathrm{set}(r,1)$$

$$\downarrow$$

$$\mathrm{set}(r,2);\mathrm{get}(r) \parallel r \Leftarrow 1$$

$$\downarrow$$

$$\mathrm{get}(r) \parallel r \Leftarrow 1 \parallel r \Leftarrow 2$$

$$\swarrow \qquad \searrow$$

$$1 \parallel r \Leftarrow 1 \parallel r \Leftarrow 2 \qquad\qquad 2 \parallel r \Leftarrow 1 \parallel r \Leftarrow 2$$

Type system

Types

effects	e, e'	$=$	$\{r_1, \ldots, r_n\}$
types	α	$::=$	$\mathbf{B} \mid A$
values types	A	$::=$	$\mathsf{Unit} \mid A \xrightarrow{e} \alpha \mid \mathsf{Reg}_r A$
variable contexts	Γ	$::=$	$x_1 : A_1, \ldots, x_n : A_n$
region contexts	R	$::=$	$r_1 : A_1, \ldots, r_n : A_n$

Judgement

$$R; \Gamma \vdash P : (\alpha, e)$$

Termination

Landin's trick gives a fixpoint

$$\begin{array}{rc} & \mathsf{set}(r, \lambda x.\mathsf{get}(r)\ x); \mathsf{get}(r) * \\ \to & \mathsf{get}(r) * \parallel r \Leftarrow \lambda x.\mathsf{get}(r)\ x \\ \to & (\lambda x.\mathsf{get}(r)\ x) * \parallel r \Leftarrow \lambda x.\mathsf{get}(r)\ x \\ \to & \mathsf{get}(r) * \parallel r \Leftarrow \lambda x.\mathsf{get}(r)\ x \\ \to & \ldots \end{array}$$

A natural constraint to recover termination is stratification

Stratification conditions

$$\frac{}{\emptyset \vdash} \qquad \frac{R \vdash A \qquad r \notin \mathrm{dom}(R)}{R, r : A \vdash}$$

$$\frac{R \vdash}{R \vdash \mathit{Unit}} \qquad \frac{R \vdash}{R \vdash \mathbf{B}}$$

$$\frac{R \vdash A \qquad R \vdash \alpha \qquad e \subseteq \mathrm{dom}(R)}{R \vdash A \xrightarrow{e} \alpha} \qquad \frac{R \vdash \qquad r : A \in R}{R \vdash \mathrm{Reg}_r A}$$

Stratification as conditions

For $R = r_1 : A_1, \ldots, r_n : A_n$, the following are equivalent

- R is stratified (well formed)
- $\mathrm{Eff}(\alpha) \subseteq \mathrm{dom}(R)$ and
  $(\forall i \ : 1 \leq i \leq n),\ \mathrm{Eff}(A_i) \subseteq \{r_{i-1}, \ldots, r_1\}$
- The following equation system is solvable
  - $\mathrm{Unit}^\bullet = !1$
  - $\mathbf{B}^\bullet = !1$
  - $(A \overset{\{s_1,\ldots,s_m\}}{\longrightarrow} \alpha)^\bullet =$
    $!((A^\bullet \otimes X_{s_1} \ldots \otimes X_{s_m}) \multimap (X_{s_1} \otimes \ldots \otimes X_{s_m} \otimes \alpha^\bullet))$
  - $X_{r_i} = {A_i}^\bullet$

Properties

Subject reduction

Termination

Progress

If $\Gamma \vdash M : (\alpha, e)$, then $M \rightarrow^* N_1 \parallel \ldots \parallel N_n$ where N_i is either a value or of the form $E[\text{get}(r)]$

Plan

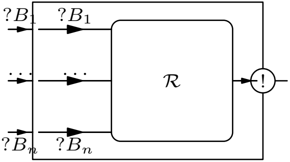
Differential linear logic

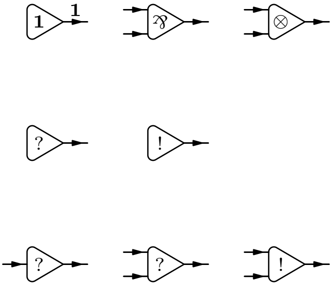
Differential linear logic [Ehrhard, Regnier] is an extension of linear logic with symmetric dual operators :

- codereliction : $A \to !A$
- coweakening : $1 \to !A$
- cocontraction : $!A \otimes !A \to !A$

We only need cocontraction and coweakening.
Reductions rules are naturally non-deterministic : we must consider **formal sums** of proofs.

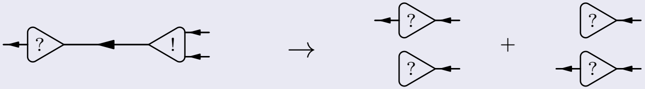
Differential interaction nets





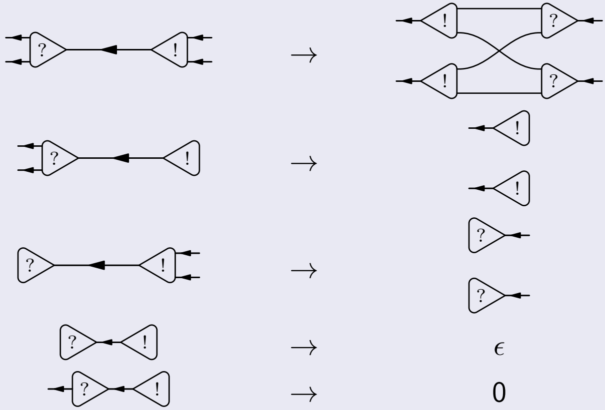
Reduction rules

Non-deterministic reduction



Reduction rules

Co-reductions



Plan

Using monads

Encoding for type and effect system :

$$\Gamma \vdash M : (\alpha, e) \text{ with } e = \{r_1, \ldots, r_n\}$$

$S = R_1 \times \ldots \times R_n$, then

$$\begin{aligned} T_e(\alpha) &= S \to S \times \alpha \\ T_e(A \xrightarrow{e} \alpha) &= A \to (S \to (S \times \alpha)) \cong A \times S \to S \times \alpha \end{aligned}$$

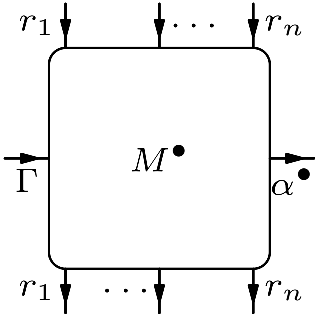
Stratification as conditions

For $R = r_1 : A_1, \ldots, r_n : A_n$, the following are equivalent

- R is stratified (well formed)
- $\mathrm{Eff}(\alpha) \subseteq \mathrm{dom}(R)$ and $(\forall i \ : 1 \leq i \leq n),\ \mathrm{Eff}(A_i) \subseteq \{r_{i-1}, \ldots, r_1\}$
- The following equation system is solvable
  - $\mathrm{Unit}^\bullet = !1$
  - $\mathbf{B}^\bullet = !1$
  - $(A \overset{\{s_1,\ldots,s_m\}}{\rightarrow} \alpha)^\bullet = !((A^\bullet \otimes X_{s_1} \ldots \otimes X_{s_m}) \multimap (X_{s_1} \otimes \ldots \otimes X_{s_m} \otimes \alpha^\bullet))$
  - $X_{r_i} = A_i^\bullet$

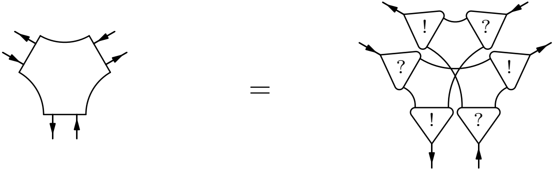
Idea

We translate a term $\Gamma \vdash M : (\alpha, e)$ to a net $M^\bullet$

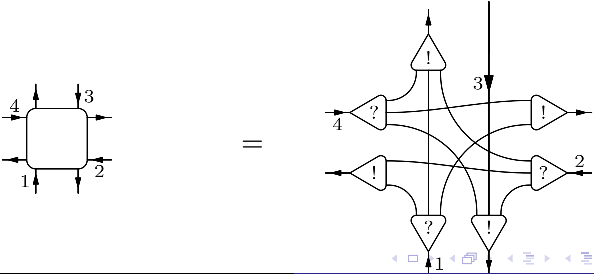


Building blocks

[Ehrhard, Laurent]

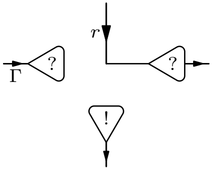


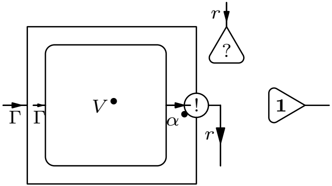
=



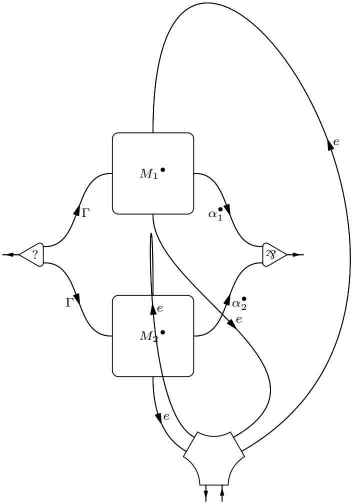
=

Translation





Translation



Cycles ?

Two kind of cycles :

First type

$$\lambda x.(\text{set}(r, x))\ \text{get}(r)$$

Fix : use custom communication area

Second type

$$\lambda x.(\text{set}(r, x))\ \text{get}(r) \parallel \lambda x.(\text{set}(r, x))\ \text{get}(r)$$

Seems to be "invisible" to GoI

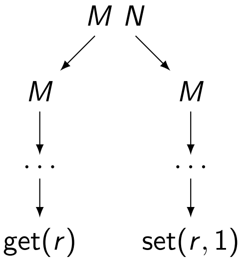
Simulation

Theorem

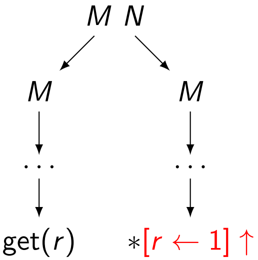
Let $\Gamma \vdash M : (\alpha, e)$ be a term

- If $M \to N$ by (β_v) or $(\mathtt{set})$ then $M^\bullet \to N^\bullet$
- If $M \to N_i$ by $(\mathtt{get})$ then $M^\bullet \to M_0^\bullet + \sum_i N_i^\bullet$

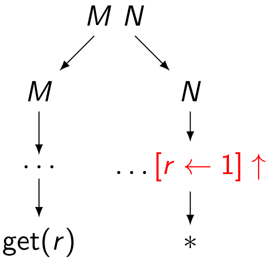
From shared memory to messages



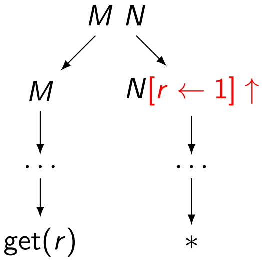
From shared memory to messages



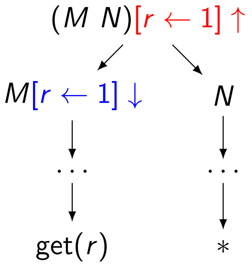
From shared memory to messages



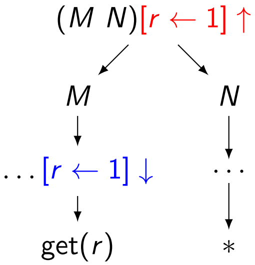
From shared memory to messages



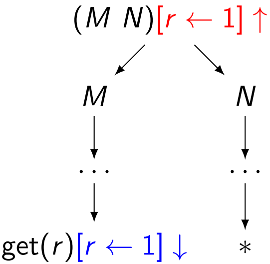
From shared memory to messages



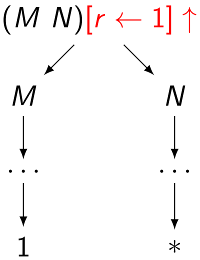
From shared memory to messages



From shared memory to messages



From shared memory to messages



Plan

What we've done

- Translation and simulation of a concurrent effectful λ-calculus in (a subset of) differential interaction nets
- Turned a global shared-memory model to a local message-passing model
- A [soon-to-be] parallel CBV token machine

Thank you !