Termination in higher-order processes

L3 internship under the supervision of Daniel Hirschkoff

Simon Castellan

December, 1st 2011

Introduction

- a study of termination in higher order π-calculus (π-calculus where messages can be (parametrized) processes)
- talk overview
  - presentation of HOπ_ω : termination and non-termination in this calculus
  - description of $\lambda\pi$: HOπ_ω with the (simply typed) λ-calculs
  - Soft-$\lambda\pi$: a subcalculus on which types give information about the length of reductions

Contents

$\mathrm{HO}\pi_\omega$

- π-calculus where processes may exchange (parametrised processes)
- the grammar for this language is

$$
\begin{array}{rrll}
(\textit{names}) & a, b & & \\
(\textit{processes}) & P, Q ::= & 0 & \text{null} \\
& & \mid (P \parallel Q) & \text{parallel} \\
& & \mid a(x).\ P & \text{reception} \\
& & \mid \overline{a}\langle V \rangle & \text{message} \\
& & \mid (\nu a)P & \text{name restriction} \\
& & \mid V\ W & \text{functional application} \\
(\textit{values}) & V, W ::= & x & \text{variables} \neq \text{names} \\
& & \mid \lambda x.\ P & \text{function} \\
& & \mid \bigstar & \text{base value}
\end{array}
$$

- communication rule : $a(x).\ P \parallel \overline{a}\langle V \rangle \to P[V/x]$

Non termination in $HO\pi_\omega$

- reduction in $HO\pi_\omega$:
  - $a(x).\ P \mid\mid \overline{a}\langle V\rangle \rightarrow P[V/x]$
  - $(\lambda x.P)V \rightarrow P[V/x]$
- "concurrent auto-application" :
  - $\delta \equiv a(f).\ (f \bigstar \mid\mid \overline{a}\langle f\rangle)$
  - $\Omega \equiv \delta \mid\mid \overline{a}\langle \lambda x.\delta\rangle \rightarrow \Omega$

Non termination in $HO\pi_\omega$

- reduction in $HO\pi_\omega$:
  - $a(x).\ P \mid\mid \overline{a}\langle V\rangle \rightarrow P[V/x]$
  - $(\lambda x.P)V \rightarrow P[V/x]$
- "concurrent auto-application" :
  - $\delta \equiv a(f).\ (f \bigstar \mid\mid \overline{a}\langle f\rangle)$
  - $\Omega \equiv \delta \mid\mid \overline{a}\langle \lambda x.\delta\rangle \rightarrow \Omega$
- the similar process : $\delta' \equiv a(f).\ (f \bigstar \mid\mid b(x).\ (\overline{c}\langle x\rangle \mid\mid \overline{a}\langle f\rangle))$ waits for a message on b at each iteration : "*spawning*" (in [LMS10], Dal Lago, Martini and Sangiorgi)

Typing in HOπ_ω for termination

- two strategies developed in [LMS10] and [DHS10] to make a type system that guarantees termination
- in this talk, I use a strategy based on [DS06].

  the idea is to stratify channels : to each channel, we associate an integer representing the **level** of the channel

$$\frac{\Gamma \vdash t : k \qquad \Gamma \vdash t' : k'}{\Gamma \vdash t \parallel t' : \max(k, k')} \qquad \frac{\Gamma \vdash t : \tau \qquad \Gamma(a) = \sharp^k \tau}{\Gamma \vdash \overline{a}\langle t \rangle : k}$$

Typing in HOπ_ω for termination

- two strategies developed in [LMS10] and [DHS10] to make a type system that guarantees termination
- in this talk, I use a strategy based on [DS06].

  the idea is to stratify channels : to each channel, we associate an integer representing the **level** of the channel

$$\frac{\Gamma \vdash t : k \qquad \Gamma \vdash t' : k'}{\Gamma \vdash t \parallel t' : \max(k, k')} \qquad \frac{\Gamma \vdash t : \tau \qquad \Gamma(a) = \sharp^k \tau}{\Gamma \vdash \overline{a}\langle t \rangle : k}$$

- controlling inputs :

$$\frac{\Gamma, x : \tau \vdash P : n < k \qquad \Gamma(a) = \sharp^k(\tau)}{\Gamma \vdash a(x).P : 0}$$

Typing example

$$\frac{\Gamma \vdash t : k \qquad \Gamma \vdash t' : k'}{\Gamma \vdash t \parallel t' : \max(k, k')} \qquad \frac{\Gamma \vdash t : \tau \qquad \Gamma(a) = \sharp^k \tau}{\Gamma \vdash \overline{a}\langle t \rangle : k}$$

$$\frac{\Gamma, x : \tau \vdash P : n < k \qquad \Gamma(a) = \sharp^k(\tau)}{\Gamma \vdash a(x).P : 0}$$

- $a(x).\ \overline{a}\langle x \rangle$ is rejected
- $a(x).\ (\overline{b}\langle x \rangle \parallel \overline{c}\langle x \rangle)$ is accepted iff b and c are $< a$.
- $a(x).\ b(x).\overline{a}\langle x \rangle$ may be accepted (the output on a is hidden by the reception on a)

Typing example

$$\frac{\Gamma \vdash t : k \qquad \Gamma \vdash t' : k'}{\Gamma \vdash t \mid\mid t' : \max(k, k')} \qquad \frac{\Gamma \vdash t : \tau \qquad \Gamma(a) = \sharp^k \tau}{\Gamma \vdash \overline{a}\langle t \rangle : k}$$

$$\frac{\Gamma, x : \tau \vdash P : n < k \qquad \Gamma(a) = \sharp^k(\tau)}{\Gamma \vdash a(x).P : 0}$$

- $a(x).\ \overline{a}\langle x \rangle$ is rejected
- $a(x).\ (\overline{b}\langle x \rangle \mid\mid \overline{c}\langle x \rangle)$ is accepted iff b and c are $< a$.
- $a(x).\ b(x).\overline{a}\langle x \rangle$ may be accepted (the output on a is hidden by the reception on a)
- functions :
  - if $x : \tau \vdash P : n$, then $\vdash \lambda\ x.\ P : \tau \to^{n+1} \diamond$
  - application: the term $V\ W$ has level n iff V has type $\tau \to^n \diamond$

Termination proof

Theorem

Every well-typed term of $HO\pi_\omega$ is strongly normalizing.

Proof.

- attach to P, $m(P)$ the multiset of outputs ($\overline{a}^k\langle V\rangle$) at top-level in P and levels of applications ($(\lambda^n x.P)\ V$)
- $m(\overline{b}\langle x\rangle \mid\mid \overline{c}\langle x\rangle \mid\mid a(x).\ \overline{d}\langle x\rangle) = \{\mathrm{lvl}(b), \mathrm{lvl}(c)\}$
- show that $P \to P'$ implies $m(P) > m(P')$ for multiset ordering.
  - $P \equiv a(x).\ P_0 \mid\mid \overline{a}\langle V\rangle \to P' \equiv P_0[V/x]$
  - $P \equiv (\lambda x.\ P_0)V \to P' \equiv P_0[V/x]$

Contents

$\lambda\pi$: syntax and semantics

- $\lambda\pi$ is an extension in HOπ_ω where all functions are available.
- grammar where processes and functions coexist :

$$
\begin{array}{rrcrr}
t, u ::= 0 & \text{null} & | & (t \parallel u) & \text{parallel} \\
| \; a(x).\, t & \text{reception} & | & \overline{a}\langle t \rangle & \text{message} \\
| \; (\nu a)t & \text{name restriction} & | & x & \text{variable} \\
| \; \lambda x.\, t & \text{abstraction} & | & t\ u & \text{application}
\end{array}
$$

- channels are *not* first-class values (but π can be encoded)

$\lambda\pi$: syntax and semantics

- $\lambda\pi$ is an extension in HOπ_ω where all functions are available.
- grammar where processes and functions coexist :

$$
\begin{array}{rrcrr}
t, u ::= 0 & \text{null} & | & (t \parallel u) & \text{parallel} \\
| \; a(x).\, t & \text{reception} & | & \overline{a}\langle t \rangle & \text{message} \\
| \; (\nu a)t & \text{name restriction} & | & x & \text{variable} \\
| \; \lambda x.\, t & \text{abstraction} & | & t\ u & \text{application}
\end{array}
$$

- channels are *not* first-class values (but π can be encoded)
- reduction : same two primitve rules as in HOπ_ω.
- strategy : weak reduction for the λ-calculus (although we could reduce under abstraction and inputs), in particular reduction inside messages

A few examples

- contains spawning and classic idioms of HOπ_ω.
- *messages as localities*:
  - $P \to P'$ implies $\overline{a}\langle P \rangle \to \overline{a}\langle P' \rangle$
  - $\overline{a}\langle P \rangle$ (P not a normal form) can be read "P running at a"
  - *passivation*: $a(X).\overline{b}\langle X \rangle$ takes whatever process running at a and transfers it to b

A few examples

- contains spawning and classic idioms of HOπ_ω.
- *messages as localities*:
  - $P \to P'$ implies $\overline{a}\langle P\rangle \to \overline{a}\langle P'\rangle$
  - $\overline{a}\langle P\rangle$ (P not a normal form) can be read "P running at a"
  - *passivation*: $a(X).\overline{b}\langle X\rangle$ takes whatever process running at a and transfers it to b
- *channels as first-class values* :
  - given a, the couple $(\lambda x.\ \overline{a}\langle x\rangle), (\lambda\ f.a(x).\ f\ x)$ represents channel a

Typing in $\lambda\pi$

- type grammar :

$$\begin{array}{rrl} \sigma, \tau & ::= \bigstar & \text{base type} \\ & \sigma \to \tau & \text{function type} \\ & k & (\in \mathbb{N}) \text{ type of processes at level } k \\ \alpha & ::= \sharp^k(\tau) & \text{channel types carrying values of type } \tau \end{array}$$

- for the λ part : simple types
- for the concurrent part : tracking receptions' body for incorrect outputs (as before)
- type of a process $=$ maximum of the levels of the channels carrying top-level messages
- in particular, spawning is typable

Typing rules

$$\frac{\Gamma(x) = \tau}{\Gamma \vdash x : \tau} \qquad \frac{\Gamma, x : \tau \vdash t : \sigma}{\Gamma \vdash \lambda x.t : \tau \to \sigma} \qquad \frac{\Gamma \vdash t : \sigma \to \tau \qquad \Gamma \vdash u : \sigma}{\Gamma \vdash t\ u : \tau}$$

$$\frac{\Gamma \vdash t : \tau \qquad \Gamma(a) = \sharp^k \tau}{\Gamma \vdash \overline{a}\langle t \rangle : k} \qquad \frac{\Gamma(a) = \sharp^k \tau \qquad \Gamma, x : \tau \vdash t : p < k}{\Gamma \vdash a(x).t : 0}$$

$$\frac{\Gamma, a : \sharp^k \tau \vdash t : \tau}{\Gamma \vdash (\nu a)t : \tau} \qquad \frac{\Gamma \vdash t : k \qquad \Gamma \vdash t' : k'}{\Gamma \vdash t \mid\mid t' : \max(k, k')} \qquad \frac{}{\Gamma \vdash 0 : 0}$$

Typing rules

$$\frac{\Gamma(x) = \tau}{\Gamma \vdash x : \tau} \qquad \frac{\Gamma, x : \tau \vdash t : \sigma}{\Gamma \vdash \lambda x.t : \tau \to \sigma} \qquad \frac{\Gamma \vdash t : \sigma \to \tau \qquad \Gamma \vdash u : \sigma}{\Gamma \vdash t\ u : \tau}$$

$$\frac{\Gamma \vdash t : \tau \qquad \Gamma(a) = \sharp^k \tau}{\Gamma \vdash \overline{a}\langle t \rangle : k} \qquad \frac{\Gamma(a) = \sharp^k \tau \qquad \Gamma, x : \tau \vdash t : p < k}{\Gamma \vdash a(x).t : 0}$$

$$\frac{\Gamma, a : \sharp^k \tau \vdash t : \tau}{\Gamma \vdash (\nu a)t : \tau} \qquad \frac{\Gamma \vdash t : k \qquad \Gamma \vdash t' : k'}{\Gamma \vdash t \mid\mid t' : \max(k, k')} \qquad \frac{}{\Gamma \vdash 0 : 0}$$

Typing rules

$$\frac{\Gamma(x) = \tau}{\Gamma \vdash x : \tau} \qquad \frac{\Gamma, x : \tau \vdash t : \sigma}{\Gamma \vdash \lambda x.t : \tau \to \sigma} \qquad \frac{\Gamma \vdash t : \sigma \to \tau \qquad \Gamma \vdash u : \sigma}{\Gamma \vdash t\ u : \tau}$$

$$\frac{\Gamma \vdash t : \tau \qquad \Gamma(a) = \sharp^k \tau}{\Gamma \vdash \overline{a}\langle t \rangle : k} \qquad \frac{\Gamma(a) = \sharp^k \tau \qquad \Gamma, x : \tau \vdash t : p < k}{\Gamma \vdash a(x).t : 0}$$

$$\frac{\Gamma, a : \sharp^k \tau \vdash t : \tau}{\Gamma \vdash (\nu a)t : \tau} \qquad \frac{\Gamma \vdash t : k \qquad \Gamma \vdash t' : k'}{\Gamma \vdash t \parallel t' : \max(k, k')} \qquad \frac{}{\Gamma \vdash 0 : 0}$$

Subtyping

- there are two (related) problems with the previous rules :
  - $f\ (a(x).\ 0 \mid\mid \overline{a}\langle\bigstar\rangle) \to f\ 0$, does f have the type level$(a) \to ..$ or $0 \to ...$ (subject reduction) ?
  - if we have a process of level k why can't we pass it to a function expecting a process of level $k' > k$ (polymorphism) ?
- one way to resolve that problem is to introduce subtyping.
  - $k \sqsubseteq k'$ iff $k \leq k'$ (types of processes)
  - $\sigma \to \tau \sqsubseteq \sigma' \to \tau'$ iff $\sigma' \sqsubseteq \sigma$ and $\tau \sqsubseteq \tau'$
- then we add the subsumption rule

$$\frac{\Gamma \vdash t : \sigma \qquad \sigma \sqsubseteq \tau}{\Gamma \vdash t : \tau}$$

Termination in $\lambda\pi$

Theorem

Every well-typed term of $\lambda\pi$ is strongly normalizing.

Proof.

- proof by reducibility candidates.
- $[[k]] = \{t, \Gamma \vdash t : k \text{ and } t \text{ is strongly normalizing}\}$
- for the λ part, everything goes as usual
- for the concurrent part, we need a lemma :
  - if $P \mid\mid Q$ reduces to R with a communication between P and Q, then $w(R) < w(P||Q)$
  - same argument as in HOπ_ω.

□

Comparison with the λ-calculus with regions ([Ama09])

- the λ-calculus with regions is a calculus that abstracts references, channels into the the concept of regions
- two primitive operations
  - get(r) to read from a store (receive)
  - write(r, v) to write a value to a store (send)
- difference in operational semantics :
  - listen (in $\lambda\pi$) : blocking, erases the value
  - get (in [Ama09]) : non-blocking, leaves the value

Comparison with the λ-calculus with regions ([Ama09])

- the λ-calculus with regions is a calculus that abstracts references, channels into the the concept of regions
- two primitive operations
  - get(r) to read from a store (receive)
  - write(r, v) to write a value to a store (send)
- difference in operational semantics :
  - listen (in $\lambda\pi$) : blocking, erases the value
  - get (in [Ama09]) : non-blocking, leaves the value
- spawning example :
  - $t_1 \equiv (\lambda k.\ \lambda y.\mathrm{set}(a, k); k\ \star)(\mathrm{get}\ a)(\mathrm{get}\ b)$
  - then $t_1, (a \Leftarrow (\lambda\_.t_1))$ is ill-typed and loops
- difference between type systems
  - in $\lambda\pi$ we watch inputs
  - in [Ama09] we watch the type of the regions

Contents

Soft-$\lambda\pi$

- as in soft-lambda calculus, we add two syntaxic constructions $\lambda!x.\ t$ and $!t$
- goal : if $\vdash t : \tau$ then t does at most $f(t, \tau)$ reductions before reducing to a normal form
- for HOπ_ω, such a system has been built in [LMS10]
- we treat processes and λ-calculs separately
- but we could have extended [LMS10] to $\lambda\pi$ (no difficulty *a priori*)

Typing rules

$$\frac{\Gamma, x : \tau; \Delta \vdash t : \sigma}{\Gamma; \Delta \vdash \lambda !x.t : !\tau \to \sigma} \qquad \frac{\Gamma; \Delta, x : \tau \vdash t : \sigma}{\Gamma; \Delta \vdash \lambda x.t : \tau \to \sigma}$$

$$\frac{\Gamma; \Delta_1 \vdash t : \tau \to \sigma \qquad \Gamma; \Delta_2 \vdash u : \tau}{\Gamma; \Delta_1, \Delta_2 \vdash t\ u : \sigma} \qquad \frac{\Gamma \cup \Delta(x) = \tau}{\Gamma; \Delta \vdash x : \tau}$$

$$\frac{\emptyset; \Delta \vdash t : \tau}{\Gamma; !\Delta, \Delta' \vdash !t : !\tau} \qquad \frac{\Gamma; \Delta_1 \vdash t : e \qquad \Gamma; \Delta_2 \vdash u : e'}{\Gamma; \Delta_1, \Delta_2 \vdash t \parallel u : \max(e, e')}$$

$$\frac{\Gamma, a : \sharp^k(\tau); \Delta \vdash t : \sigma}{\Gamma; \Delta \vdash (\nu a)t : \sigma} \qquad \frac{\Gamma; \Delta \vdash t : \tau \qquad \Gamma(a) = \sharp^k(\tau)}{\Gamma; \Delta \vdash \overline{a}\langle t \rangle : k} \qquad \frac{}{\Gamma; \Delta \vdash 0 : 0}$$

$$\frac{\Gamma, x : \tau; \Delta \vdash t : e \qquad \Gamma(a) = \sharp^k(\tau) \qquad e < k}{\Gamma; \Delta \vdash a(x).t : 0} \qquad \frac{\Gamma \vdash t : \tau \qquad \tau \leq \sigma}{\Gamma \vdash t : \sigma}$$

A bound

Theorem

Every well-typed term t of Soft-$\lambda\pi$ normalizes in at most $O(d^{n+k})$ steps, where

- *d is the number of occurences of the variable (or name) that appears the most in t (duplicability factor)*
- *n is the number of channels used in t ;*
- *k is the box depth of t (maximum nesting of bangs appearing in t's typing derivation)*

- to ease reasoning we consider a derivation of t where channels are assigned different levels
- use of two measure : one for the concurrent aspects and one for the sequential aspects
- the "concurrent" bound is reached :
  - $p_n \equiv (a_0(x).\overline{a}_1\langle x \parallel x\rangle) \parallel \ldots \parallel (a_{n-1}(x).\overline{a}_n\langle x \parallel x\rangle)$
  - $p_n \| \overline{a}_1\langle P\rangle \| a_n(x).x$ reduces to $\underbrace{P \parallel \ldots \parallel P}_{2^n \text{times}}$

Conclusion

To go further :

- see if the proof can be extended to more complex type systems for λ (eg. System F)
- some ideas for type inference in $\lambda\pi$
- improve the bound (perhaps d^{n+k} where is n the highest level of names bound in t)

π-calculus encoding into $\lambda\pi$

- two features needed : first-class channels and replication
- first-class channels : ok (cf. example)
- replication : through spawning,

$$[[!a(x).P]] = (\nu b)S \,||\overline{b}\langle \lambda x.\, S\rangle, S = a(f).\, b(x).\, ([[P]] \,||\, \overline{a}\langle f\rangle \,||\, f\ ())$$

$\lambda\pi$ versus the rest of the world

$\lambda\pi$ versus :

- [LMS10] :
  - in $\lambda\pi$: $a(k).(k||\overline{a}\langle k\rangle)$ – different levels for x
  - in [LMS10] : $a(x).\overline{a}\langle x\rangle$ – reception body too large
- [DHS10] :
  - in $\lambda\pi$: $\overline{a}\langle\overline{a}\langle x\rangle\rangle$ – nesting output
  - in [DHS10] : $a(x).\overline{a}\langle x\rangle$ – reception body too large


Roberto M. Amadio.
On stratified regions.
CoRR, abs/0904.2076, 2009.


R. Demangeon, D. Hirschkoff, and D. Sangiorgi.
Termination in higher-order concurrent calculi.
Journal of Logic and Algebraic Programming, 2010.


Y. Deng and D. Sangiorgi.
Ensuring termination by typability.
Information and Computation, 204(7):1045–1082, 2006.


Ugo Dal Lago, Simone Martini, and Davide Sangiorgi.
Light logics and higher-order processes.
In *EXPRESS'10*, pages 46–60, 2010.