Termination in higher-order processes
L3 internship under the supervision of Daniel Hirschkoff

Simon Castellan

December, 1\textsuperscript{st} 2011
a study of termination in higher order $\pi$-calculus ($\pi$-calculus where messages can be (parametrized) processes)

talk overview
  - presentation of $\text{HO}\pi_\omega$ : termination and non-termination in this calculus
  - description of $\lambda\pi$ : $\text{HO}\pi_\omega$ with the (simply typed) $\lambda$-calculs
  - $\text{Soft-}\lambda\pi$ : a subcalculus on which types give information about the length of reductions
Contents

\( \text{HO} \pi_\omega \): Higher order \( \pi \)-calculus

\( \lambda\pi \): higher order \( \pi \)-calculs with full \( \lambda \)-calculus

Soft-\( \lambda\pi \): toward a bound on the length of reductions
\( \text{HO} \pi \omega \)

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

\[
\begin{align*}
\text{(names)} & \quad a, b \\
\text{(processes)} & \quad P, Q ::= 0 \quad \text{null} \\
& \quad | (P || Q) \quad \text{parallel} \\
& \quad | a(x). P \quad \text{reception} \\
& \quad | \bar{a}\langle V \rangle \quad \text{message} \\
& \quad | (\nu a)P \quad \text{name restriction} \\
& \quad | V \ W \quad \text{functional application} \\
\text{(values)} & \quad V, W ::= x \quad \text{variables } \neq \text{ names} \\
& \quad | \lambda x. P \quad \text{function} \\
& \quad | \star \quad \text{base value}
\end{align*}
\]

- communication rule : \( a(x). P || \bar{a}\langle V \rangle \rightarrow P[V/x] \)
Non termination in $\textit{HO}_\pi\omega$

- reduction in $\textit{HO}_\pi\omega$:
  - $a(x). \ P \parallel \bar{a}\langle V \rangle \rightarrow P[V/x]$
  - $(\lambda x. P)V \rightarrow P[V/x]$
- “concurrent auto-application”:
  - $\delta \equiv a(f). (f \star \parallel \bar{a}\langle f \rangle)$
  - $\Omega \equiv \delta \parallel \bar{a}\langle \lambda x. \delta \rangle \rightarrow \Omega$
Non termination in $\mathcal{HO}\pi\omega$

- reduction in $\mathcal{HO}\pi\omega$:
  - $a(x).\ P \parallel \bar{a}\langle V \rangle \rightarrow P[V/x]$
  - $(\lambda x. P)V \rightarrow P[V/x]$
- “concurrent auto-application”:
  - $\delta \equiv a(f).\ (f \star \parallel \bar{a}\langle f \rangle)$
  - $\Omega \equiv \delta \parallel \bar{a}\langle \lambda x.\delta \rangle \rightarrow \Omega$
- the similar process: $\delta' \equiv a(f).\ (f \star \parallel b(x).\ (\bar{c}\langle x \rangle \parallel \bar{a}\langle f \rangle))$
  waits for a message on $b$ at each iteration: “spawning” (in [LMS10], Dal Lago, Martini and Sangiorgi)
Typing in $\text{HO}^{\pi_\omega}$ 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

\[
\Gamma \vdash t : k \quad \Gamma \vdash t' : k' \quad \Gamma \vdash t : \tau \quad \Gamma(a) = \#^k\tau \\
\Gamma \vdash t \parallel t' : \max(k, k') \quad \Gamma \vdash a(t) : k
\]
Typing in $\text{HO}^{\pi_\omega}$ 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

\[
\begin{align*}
\Gamma \vdash t : k & \quad \Gamma \vdash t' : k' \\
\Gamma \vdash t \parallel t' : \max(k, k') & \\
\Gamma \vdash t : \tau & \quad \Gamma(a) = \#^k \tau \\
\Gamma \vdash a(t) : k
\end{align*}
\]

- controlling inputs:

\[
\begin{align*}
\Gamma, x : \tau \vdash P : n < k & \quad \Gamma(a) = \#^k(\tau) \\
\Gamma \vdash a(x).P : 0
\end{align*}
\]
Typing example

\[
\begin{align*}
\Gamma \vdash t : k & \quad \Gamma \vdash t' : k' \\
\Gamma \vdash t \parallel t' : \text{max}(k, k') & \\
\Gamma \vdash t : \tau & \quad \Gamma(a) = \#^k \tau \\
\Gamma \vdash \bar{a} \langle t \rangle : k
\end{align*}
\]

\[
\begin{align*}
\Gamma, x : \tau & \vdash P : n < k & \Gamma(a) = \#^k (\tau) \\
\Gamma & \vdash a(x). P : 0
\end{align*}
\]

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

\[
\begin{align*}
\Gamma \vdash t : k \quad \Gamma \vdash t' : k' & \quad \Gamma \vdash t : \tau \quad \Gamma(a) = \#^k \tau \\
\Gamma \vdash t \parallel t' : \max(k, k') & \quad \Gamma \vdash a\langle t \rangle : k \\
\Gamma, x : \tau \vdash P : n < k \quad \Gamma(a) = \#^k(\tau) & \quad \Gamma \vdash a(x).P : 0
\end{align*}
\]

- \(a(x). \overline{a}\langle x \rangle\) 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 \(a\)).
- functions :
  - if \(x : \tau \vdash P : n\), then \(\vdash \lambda x. P : \tau \rightarrow^{n+1}\)
  - application: the term \(V W\) has level \(n\) iff \(V\) has type \(\tau \rightarrow^n\)
Termination proof

**Theorem**
Every well-typed term of $HO^{\pi_\omega}$ is strongly normalizing.

**Proof.**

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

$\text{HO}\pi_\omega$ : Higher order $\pi$-calculus

$\lambda\pi$ : higher order $\pi$-calculs with full $\lambda$-calculus

Soft-$\lambda\pi$ : toward a bound on the length of reductions
\(\lambda\pi\): syntax and semantics

- \(\lambda\pi\) is an extension in HO\(\pi_\omega\) where all functions are available.
- grammar where processes and functions coexist:

\[
\begin{align*}
t, u & ::= 0 & \text{null} & (t \parallel u) & \text{parallel} \\
& | a(x). t & \text{reception} & \bar{a}\langle t \rangle & \text{message} \\
& | (\nu a)t & \text{name restriction} & x & \text{variable} \\
& | \lambda x. t & \text{abstraction} & t u & \text{application}
\end{align*}
\]

- channels are not first-class values (but \(\pi\) can be encoded)
$\lambda\pi$ : syntax and semantics

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

\[
t, u ::= 0 \quad \text{null} \quad | \quad (t || u) \quad \text{parallel} \\
| \quad a(x). t \quad \text{reception} \quad | \quad \overline{a}(t) \quad \text{message} \\
| \quad (\nu a)t \quad \text{name restriction} \quad | \quad x \quad \text{variable} \\
| \quad \lambda x. t \quad \text{abstraction} \quad | \quad t u \quad \text{application}
\]

- channels are not first-class values (but $\pi$ can be encoded)
- reduction : same two primitive rules as in HO$\pi_\omega$.
- strategy : weak reduction for the $\lambda$-calculus (although we could reduce under abstraction and inputs), in particular reduction inside messages
A few examples

- contains spawning and classic idioms of $\text{HO}\pi_\omega$.
- *messages as localities:*
  - $P \to P'$ implies $\bar{a}\langle P \rangle \to \bar{a}\langle P' \rangle$
  - $\bar{a}\langle P \rangle$ ($P$ not a normal form) can be read “$P$ running at $a$”
  - *passivation:* $a(X).\bar{b}\langle X \rangle$ takes whatever process running at $a$ and transfers it to $b$
A few examples

- contains spawning and classic idioms of HO$\pi_\omega$.
- messages as localities:
  - $P \rightarrow P'$ implies $\bar{a}\langle P \rangle \rightarrow \bar{a}\langle P' \rangle$
  - $\bar{a}\langle P \rangle$ ($P$ not a normal form) can be read “$P$ running at $a$”
  - passivation: $a(X).\bar{b}\langle X \rangle$ takes whatever process running at $a$ and transfers it to $b$
- channels as first-class values:
  - given $a$, the couple $(\lambda x. \bar{a}\langle x \rangle), (\lambda f.a(x). f x)$ represents channel $a$
Typing in $\lambda\pi$

- type grammar:

$$\sigma, \tau ::= \star$$

- base type

$$\sigma \rightarrow \tau$$

- function type

$$k \quad (\in \mathbb{N}) \text{ type of processes at level } k$$

$$\alpha ::= \#^k(\tau) \quad \text{channel types carrying values of type } \tau$$

- for the $\lambda$ 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

\[
\begin{align*}
\Gamma(x) &= \tau, & \Gamma, x : \tau &\vdash t : \sigma, & \Gamma \vdash t : \sigma \to \tau, & \Gamma \vdash u : \sigma \\
\Gamma \vdash x : \tau, & \Gamma \vdash \lambda x. t : \tau \to \sigma, & \Gamma \vdash t u : \tau \\
\Gamma \vdash t : \tau, & \Gamma(a) = \#^k \tau, & \Gamma \vdash a \langle t \rangle : k, & \Gamma \vdash a(x). t : 0 \\
\Gamma, a : \#^k \tau &\vdash t : \tau, & \Gamma \vdash t : k, & \Gamma \vdash t' : k', & \Gamma \vdash 0 : 0 \\
\Gamma \vdash (\nu a) t : \tau, & \Gamma \vdash t \parallel t' : \max(k, k'), & \\
\end{align*}
\]
Typing rules

\[
\begin{align*}
\Gamma(x) &= \tau & \Gamma, x : \tau \vdash t : \sigma & \Rightarrow & \Gamma \vdash \lambda x. t : \tau \to \sigma \\
\Gamma \vdash x : \tau & & \Gamma \vdash t : \sigma \to \tau & \Rightarrow & \Gamma \vdash u : \sigma \\
\Gamma \vdash t : \tau & & \text{if } \Gamma(a) = \#^k \tau & \Rightarrow & \Gamma \vdash a(t) : k \\
& & \text{if } \Gamma(a) = \#^k \tau, x : \tau \vdash t : p < k & \Rightarrow & \Gamma \vdash a(x). t : 0 \\
\Gamma, a : \#^k \tau \vdash t : \tau & & \Gamma \vdash t : k & \Rightarrow & \Gamma \vdash (\nu a) t : \tau \\
\Gamma \vdash t : k & & \Gamma \vdash t' : k' & \Rightarrow & \Gamma \vdash t || t' : \max(k, k') \\
\Gamma \vdash 0 : 0
\end{align*}
\]
Typing rules

\[ \Gamma(x) = \tau \quad \frac{\Gamma, x : \tau \vdash t : \sigma}{\Gamma \vdash \lambda x. t : \tau \rightarrow \sigma} \quad \frac{\Gamma \vdash t : \sigma \rightarrow \tau \quad \Gamma \vdash u : \sigma}{\Gamma \vdash t \ u : \tau} \]

\[ \frac{\Gamma \vdash t : \tau \quad \Gamma(a) = \#^k \tau}{\Gamma \vdash \bar{a}\langle t \rangle : k} \quad \frac{\Gamma(a) = \#^k \tau \quad \Gamma, x : \tau \vdash t : p < k}{\Gamma \vdash a(x). t : 0} \]

\[ \frac{\Gamma, a : \#^k \tau \vdash t : \tau}{\Gamma \vdash (\nu a)t : \tau} \quad \frac{\Gamma \vdash t : k \quad \Gamma \vdash t' : k'}{\Gamma \vdash t \parallel t' : \max(k, k')} \quad \Gamma \vdash 0 : 0 \]
Subtyping

- there are two (related) problems with the previous rules:
  - \( f \ (a(x). \ 0 \ || \ \bar{a}(\star)) \rightarrow f \ 0 \), does \( f \) have the type \( a \rightarrow \ldots \) or \( 0 \rightarrow \ldots \) (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)
  - \( \sigma \rightarrow \tau \sqsubseteq \sigma' \rightarrow \tau' \) iff \( \sigma' \sqsubseteq \sigma \) and \( \tau \sqsubseteq \tau' \)

- then we add the subsumption rule

\[
\Gamma \vdash t : \sigma \quad \sigma \sqsubseteq \tau \\
\hline
\Gamma \vdash t : \tau
\]
**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 $\lambda$ part, everything goes as usual
- for the concurrent part, we need a lemma:
  - if $P \parallel Q$ reduces to $R$ with a communication between $P$ and $Q$, then $w(R) < w(P \parallel Q)$
  - same argument as in $\text{HO}\pi\omega$.  

□
Comparison with the $\lambda$-calculus with regions ([Ama09])

- the $\lambda$-calculus with regions is a calculus that abstracts references, channels into 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 $\lambda$-calculus with regions ([Ama09])

- The $\lambda$-calculus with regions is a calculus that abstracts references, channels into 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:
  - $t_1 \equiv (\lambda k. \lambda y. \text{set}(a, k); k \star) (\text{get } a)(\text{get } b)$
  - 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

$\text{HO}_{\pi_\omega}$ : Higher order $\pi$-calculus

$\lambda\pi$ : higher order $\pi$-calculs with full $\lambda$-calculus

Soft-$\lambda\pi$ : toward a bound on the length of reductions
as in soft-lambda calculus, we add two syntaxic constructions \( \lambda !x. \ t \) and \( !t \)

good : if \( \vdash t : \tau \) then \( t \) does at most \( f(t, \tau) \) reductions before reducing to a normal form

for HO\(\pi \omega\), such a system has been built in [LMS10]

we treat processes and \( \lambda \)-calculs separately

but we could have extended [LMS10] to \( \lambda \pi \) (no difficulty \( a \ priori \))
Typing rules

\[
\begin{align*}
\Gamma, x : \tau ; \Delta & \vdash t : \sigma \\
\Gamma ; \Delta & \vdash \lambda x . t : !\tau \to \sigma \\
\Gamma ; \Delta, x : \tau & \vdash t : \sigma \\
\Gamma ; \Delta & \vdash \lambda x . t : \tau \to \sigma \\
\Gamma ; \Delta_1 & \vdash t : \tau \to \sigma \\
\Gamma ; \Delta_2 & \vdash u : \tau \\
\Gamma ; \Delta_1, \Delta_2 & \vdash t \ u : \sigma \\
\Gamma & \cup \Delta(x) = \tau \\
\Gamma ; \Delta & \vdash x : \tau \\
\emptyset ; \Delta & \vdash t : \tau \\
\Gamma ; !\Delta, \Delta' & \vdash !t : !\tau \\
\Gamma ; \Delta_1 & \vdash t : e \\
\Gamma ; \Delta_2 & \vdash u : e' \\
\Gamma ; \Delta_1, \Delta_2 & \vdash t \parallel u : \max(e, e') \\
\Gamma, a : \#^k(\tau) ; \Delta & \vdash t : \sigma \\
\Gamma ; \Delta & \vdash (\nu a)t : \sigma \\
\Gamma ; \Delta & \vdash t : \tau \\
\Gamma(a) & = \#^k(\tau) \\
\Gamma ; \Delta & \vdash a(t) : k \\
\Gamma ; \Delta & \vdash 0 : 0 \\
\Gamma, x : \tau ; \Delta & \vdash t : e \\
\Gamma(a) & = \#^k(\tau) \\
e & < k \\
\Gamma ; \Delta & \vdash a(x) . t : 0 \\
\Gamma & \vdash t : \tau \\
\tau & \leq \sigma \\
\Gamma & \vdash t : \sigma
\end{align*}
\]
A bound

Theorem

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

- \( d \) is the number of occurrences 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:
  - \( p_n \equiv (a_0(x).\overline{a}_1\langle x \parallel x \rangle) \parallel \ldots \parallel (a_{n-1}(x).\overline{a}_n\langle x \parallel x \rangle) \)
  - \( p_n \parallel \overline{a}_1\langle P \rangle \parallel a_n(x).x \) reduces to \( P \parallel \ldots \parallel P \) \( 2^n \) times
Conclusion

To go further:

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

two features needed: first-class channels and replication

first-class channels: ok (cf. example)

replication: through spawning,

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

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.

Ugo Dal Lago, Simone Martini, and Davide Sangiorgi.
Light logics and higher-order processes.