

Type Systems for the Termination of Mobile Processes

Romain Demangeon

ENS Lyon

(Joint work with D.Hirschhoff, ENS Lyon and D.Sangiorgi, University of Bologna)

An overview of recent results

Starting point

- *Ensuring termination by typability* [06] (Deng, Sangiorgi)

Our contributions

- *On the complexity of termination inference for processes* [TGC'07] (Demangeon, Hirschhoff, Kobayashi, Sangiorgi).
- *Static and dynamic typing for the termination of mobile processes* [TCS'08] (Demangeon, Hirschhoff, Sangiorgi).
- *Termination in higher-order concurrent calculi* [current work] (Demangeon, Hirschhoff, Sangiorgi).

- 1 Termination in the π -calculus
- 2 Weight-based Type Systems
- 3 Allowing Forms of Recursion
- 4 Higher-order Mobile Calculi

- 1 Termination in the π -calculus
- 2 Weight-based Type Systems
- 3 Allowing Forms of Recursion
- 4 Higher-order Mobile Calculi

Termination for mobile processes

Termination

- Desirable in itself. Prerequisite for other properties (soundness, lock freedom, ...)
- Existence of multiple techniques for sequential programs.
- Termination of mobile systems is challenging. Structures evolve at run-time.

The Formalism

π -calculus: studying termination from different points of view inside a same framework:

- First-order: termination and name-passing. (System S1, S2, S3)
- Higher-order: termination and process-passing. (System S4)

Our framework: the π -calculus

- $!p(x).T \mid \bar{p}\langle a \rangle.U$
 - $!p(x).T$: server
 - $\bar{p}\langle a \rangle.U$: client $\bar{p}\langle a \rangle$ is the request
- Reduction: $!p(x).T \mid \bar{p}\langle a \rangle.U \rightarrow !p(x).T \mid T_{[a/x]} \mid U$
- Several requests: $!p(x).T \mid \bar{p}\langle a \rangle.U_1 \mid \bar{p}\langle b \rangle.U_2$
more generally: many servers, many requests

Our framework: the π -calculus

- $!p(x).T \mid \bar{p}\langle a \rangle.U$
 - $!p(x).T$: server
 - $\bar{p}\langle a \rangle.U$: client $\bar{p}\langle a \rangle$ is the request
- Reduction: $!p(x).T \mid \bar{p}\langle a \rangle.U \rightarrow !p(x).T \mid T_{[a/x]} \mid U$
- Several requests: $!p(x).T \mid \bar{p}\langle a \rangle.U_1 \mid \bar{p}\langle b \rangle.U_2$
more generally: many servers, many requests
- Replication: persistence, source of divergence
 $\bar{a}\langle b \rangle \mid !a(x).\bar{a}\langle x \rangle$ or even $\bar{a}\langle b \rangle \mid !a(x).(\bar{a}\langle x \rangle \mid \bar{a}\langle x \rangle)$

Our framework: the π -calculus

- $!p(x).T \mid \bar{p}\langle a \rangle.U$
 - $!p(x).T$: server
 - $\bar{p}\langle a \rangle.U$: client $\bar{p}\langle a \rangle$ is the request
- Reduction: $!p(x).T \mid \bar{p}\langle a \rangle.U \rightarrow !p(x).T \mid T_{[a/x]} \mid U$
- Several requests: $!p(x).T \mid \bar{p}\langle a \rangle.U_1 \mid \bar{p}\langle b \rangle.U_2$
more generally: many servers, many requests
- Replication: persistence, source of divergence
 $\bar{a}\langle b \rangle \mid !a(x).\bar{a}\langle x \rangle$ or even $\bar{a}\langle b \rangle \mid !a(x).(\bar{a}\langle x \rangle \mid \bar{a}\langle x \rangle)$

(we will sometimes use simply CCS processes: $\bar{a} \mid !a.\bar{a}$)

Our framework: the π -calculus

- $!p(x).T \mid \bar{p}\langle a \rangle.U$
 - $!p(x).T$: server
 - $\bar{p}\langle a \rangle.U$: client $\bar{p}\langle a \rangle$ is the request
- Reduction: $!p(x).T \mid \bar{p}\langle a \rangle.U \rightarrow !p(x).T \mid T_{[a/x]} \mid U$
- Several requests: $!p(x).T \mid \bar{p}\langle a \rangle.U_1 \mid \bar{p}\langle b \rangle.U_2$
more generally: many servers, many requests
- Replication: persistence, source of divergence
 $\bar{a}\langle b \rangle \mid !a(x).\bar{a}\langle x \rangle$ or even $\bar{a}\langle b \rangle \mid !a(x).(\bar{a}\langle x \rangle \mid \bar{a}\langle x \rangle)$

(we will sometimes use simply CCS processes: $\bar{a} \mid !a.\bar{a}$)

- 1 Termination in the π -calculus
- 2 Weight-based Type Systems**
- 3 Allowing Forms of Recursion
- 4 Higher-order Mobile Calculi

S1: the original type system

Principles

- Assigning *levels* (integers) to channels.
- Assigning levels to processes: maximum of names used in output (not appearing under a replication).

$$P_1 = !a.(\bar{b} \mid \bar{b} \mid \bar{c}) \mid !b.\bar{c} \mid \bar{a} \mid \bar{c}$$

- Typing $!a^n(\tilde{x}).P$ with $P : m$ when $n > m$.
- Firing a replicated input: trading an output for several smaller outputs.

S1: the original type system

Principles

- Assigning *levels* (integers) to channels.
- Assigning levels to processes: maximum of names used in output (not appearing under a replication).

$$P_1 = !a^3.(\bar{b}^2 \mid \bar{b}^2 \mid \bar{c}^1) \mid !b^2.\bar{c}^1 \mid \bar{a}^3 \mid \bar{c}^1$$

- Typing $!a^n(\tilde{x}).P$ with $P : m$ when $n > m$.
- Firing a replicated input: trading an output for several smaller outputs.

The System S1

Typing rule for replicated input

$$\frac{\Gamma, \tilde{x} : \tilde{T} \vdash P : m \quad m < n}{\Gamma \vdash !a^n(\tilde{x}).P : 0} \quad (\text{S1})$$

Giving level 0 to replicated process

- Guarded by a replication.
- $!a(x).(\bar{b} \mid !x(y).P)$: outputs in P do not count for the replication on a .

Soundness

- If P is S1-typable, P terminates
- (*Proof*) We show that a given well-founded measure decreases at each reduction step.

- $P_1 = !a^3.(\bar{b}^2 \mid \bar{b}^2 \mid \bar{c}^1) \mid !b^2.\bar{c}^1 \mid \bar{a}^3 \mid \bar{c}^1$
 $[1, 0, 1] \rightarrow [0, 2, 2] \rightarrow [0, 1, 3] \rightarrow [0, 0, 4]$.
- $!a^n.\bar{a}^n$ not S1-typable as it forces $n > n$.
- Similarly $!a^n.\bar{b}^m \mid !b^m.\bar{a}^n$ gives $n > m > n$.

Type inference

- Inference for S1 is polynomial.
- Graph where nodes are names and edges are constraints \Rightarrow topological sort gives a solution if there is one.

Soundness

- If P is S1-typable, P terminates
- (*Proof*) We show that a given well-founded measure decreases at each reduction step.

- $P_1 = !a^3.(\bar{b}^2 \mid \bar{b}^2 \mid \bar{c}^1) \mid !b^2.\bar{c}^1 \mid \bar{a}^3 \mid \bar{c}^1$
 $[1, 0, 1] \rightarrow [0, 2, 2] \rightarrow [0, 1, 3] \rightarrow [0, 0, 4]$.
- $!a^n.\bar{a}^n$ not S1-typable as it forces $n > n$.
- Similarly $!a^n.\bar{b}^m \mid !b^m.\bar{a}^n$ gives $n > m > n$.

Type inference

- Inference for S1 is polynomial.
- Graph where nodes are names and edges are constraints \Rightarrow topological sort gives a solution if there is one.

- 1 Termination in the π -calculus
- 2 Weight-based Type Systems
- 3 Allowing Forms of Recursion**
- 4 Higher-order Mobile Calculi

S2: a more expressive system

Principles

- No recursion is allowed in S1. $!a(x).P \Rightarrow \bar{a} \notin P$.
- Recursions:
 - sometimes desirable (modelling recursive functions)
 - sometimes innocuous like in $!a.b.\bar{a}$.
- S2: more expressive system.
 - Input sequences are taken as a whole. $!a_1(\tilde{x}_1) \dots a_k(\tilde{x}_k).P$
 - Comparing multisets of levels

$!a^n.b^m.\bar{a}^n \mid !a^n.\bar{b}^m \mid \bar{a}^n \mid \bar{b}^m$

- Ruled out by S1: $n > n$.
- Can be type-checked in S2 with $n = 2$ and $m = 1$. The first replication is typed as $\{1, 2\} >_{mul} \{2\}$.

S2: a more expressive system

Principles

- No recursion is allowed in S1. $!a(x).P \Rightarrow \bar{a} \notin P$.
- Recursions:
 - sometimes desirable (modelling recursive functions)
 - sometimes innocuous like in $!a.b.\bar{a}$.
- S2: more expressive system.
 - Input sequences are taken as a whole. $!a_1(\tilde{x}_1) \dots a_k(\tilde{x}_k).P$
 - Comparing multisets of levels

$!a^n.b^m.\bar{a}^n \mid !a^n.\bar{b}^m \mid \bar{a}^n \mid \bar{b}^m$

- Ruled out by S1: $n > n$.
- Can be type-checked in S2 with $n = 2$ and $m = 1$. The first replication is typed as $\{1, 2\} >_{mul} \{2\}$.

Typing rule for replicated inputs

$$\frac{\Gamma, \tilde{x}_1 : \tilde{T}_1, \dots, \tilde{x}_k : \tilde{T}_k \vdash P : M \quad \{n_1, \dots, n_k\} >_{mul} M}{\Gamma \vdash !a_1^{n_1}(\tilde{x}_1) \dots a_k^{n_k}(\tilde{x}_k).P : \emptyset} \quad (S2)$$

Complexity

- S1-typable \Rightarrow S2-typable.
- Inference for S2 is NP-complete.
- Polynomiality retrieved if we use algebraical operations on levels.

S3: A system using partial orders

Principles

- A system allowing non-decreasing replications.
- A well-founded partial order between names of the same level ensures termination.

Typing complex structures

$!p(a, b).a.(\bar{b} \mid \bar{p}\langle a, b \rangle)$ (a and b play the same role)

- Models the behavior of a list structure.
- Typed with a low level for p , a higher level for a, b and $a \succ b$.
- Type of p : first argument dominates the second one.
- S4: extension of S3 in which we can type tree structures
 $!p(a, b, c).a.(\bar{b} \mid \bar{c} \mid \bar{p}\langle a, b, c \rangle)$. \Rightarrow the weight increases.

- 1 Termination in the π -calculus
- 2 Weight-based Type Systems
- 3 Allowing Forms of Recursion
- 4 Higher-order Mobile Calculi**

HOPi as a framework for higher order mobility

HOPi

- Messages exchanged are processes.
- $a(X).P_1 \mid \bar{a}\langle Q \rangle.P_2 \rightarrow P_1[Q/X] \mid P_2.$
- Replication is no longer present ...

Another form of recursion

- ... however, divergences arise.
- $Q_0 = a(X).(\bar{a}\langle X \rangle \mid X)$ and $P_0 = Q_0 \mid \bar{a}\langle Q_0 \rangle.$ (P_0 diverges.)

A weight-based type system for higher order π

Typing rule for output

$$\frac{\Gamma \vdash P : k \quad \Gamma \vdash Q : m \quad k < n}{\Gamma \vdash \bar{a}^n \langle P \rangle . Q : \max(m, n)} \quad (\text{S4})$$

Principles

- Forbidding self-emissions: for $\bar{a} \langle P \rangle$ to be typable, $\bar{a} \notin P$.
- Use of levels: for $\bar{a}^n \langle P \rangle$ with $P : m$, we must have $n > m$.
- *Soundness*: typable processes terminate.

Examples

Ruling out P_0

$Q_0 = a(X).(\bar{a}\langle X \rangle \mid X)$ and $P_0 = Q_0 \mid \bar{a}\langle Q_0 \rangle$

- P_0 contains $\bar{a}^n\langle Q_0 \rangle$ and Q_0 contains an output on a^n .
- To type P_0 , we must have $n > n$.

A typable process

$a(X).(X \mid X) \mid \bar{a}\langle \bar{b}\langle \mathbf{0} \rangle \rangle \mid b(Y).Y$

Examples

Ruling out P_0

$Q_0 = a(X).(\bar{a}\langle X \rangle \mid X)$ and $P_0 = Q_0 \mid \bar{a}\langle Q_0 \rangle$

- P_0 contains $\bar{a}^n\langle Q_0 \rangle$ and Q_0 contains an output on a^n .
- To type P_0 , we must have $n > n$.

A typable process

$a^2(X).(X \mid X) \mid \bar{a}^2\langle \bar{b}^1\langle \mathbf{0} \rangle \rangle \mid b^1(Y).Y$

$[1, 0] \rightarrow [0, 2] \rightarrow [0, 1]$

Relation with S1

- S4 is to HOPi what S1 is to π .
- Polynomial inference: like in S1.
- Standard encoding of HOPi in π [SangiorgiWalker01]: a typable HOPi process is encoded into a S1-typable process.

Further extension

Exploring termination in richer higher-order calculi:

- A calculus where messages are functions of any order.
- A calculus where localized computation and passivation are allowed.

Conclusion

Future Works

- Develop extensions of the system for higher-order message passing.
- Analyse other approaches to termination (logical relations).