GETCO’2000

Occurrences Counting Analysis for the π-calculus

Jérôme Feret
École normale supérieure
http://www.di.ens.fr/~feret

August 21, 2000
Overview

1. Mobile systems
   (a) What is it?
   (b) $\pi$-calculus
   (c) Non-standard semantics

2. Approximate the collecting semantics
   (a) Abstract interpretation
   (b) Control flow analysis
   (c) Occurrences counting analysis
   (d) Examples

3. Trace-based analysis
   (a) Dynamic partitioning
   (b) Dead lock analysis
   (c) Example
Mobile systems
A pool of processes which interact via communications.

Communications allow to

- synchronize process computation;
- change structure of processes;
- create new communication links;
- create new processes.

**Topology of interaction may be unbounded!**
Example: a server
Objectives

We need a sound description of the multiset of the processes that occur inside computation sequences

- to prove that **physical resources are not exhausted**;

- to **refine control flow analysis** by detecting that some processes can never communicate, 
  by detecting **mutual exclusion**;

- to provide **a good criterion of partitioning**, 
  for **dead lock analysis**.

We propose **a polynomial solution**.
\[\pi\text{-calculus : syntax}\]

Let \textit{Channel} be an infinite set of channel names, and \textit{Label} an infinite set of labels,

\[
P ::= \text{action}.P \quad \text{(Action)} \\
| (P \mid P) \quad \text{(Parallel composition)} \\
| (P + P) \quad \text{(Non deterministic choice)} \\
| \emptyset \quad \text{(End of a process)}
\]

\[\begin{align*}
\text{action} ::= & \quad c!^i[x_1, \ldots, x_n] \quad \text{(Message)} \\
| & \quad c?^i[x_1, \ldots, x_n] \quad \text{(Input guard)} \\
| & \quad *c?^i[x_1, \ldots, x_n] \quad \text{(Replication guard)} \\
| & \quad (\nu x) \quad \text{(Channel creation)}
\end{align*}\]

where \(n \geq 0\),
\[
c, x_1, \ldots, x_n, x, \in \text{Channel} \text{ and } i \in \text{Label}.
\]

\(\nu\) and \(?\) are the only name binders. We denote by \(\mathcal{FN}(P)\) the set of free names in \(P\), and by \(\mathcal{BN}(P)\) the set of bound names in \(P\).
Transition semantics

A reduction relation and a congruence relation give the semantics of the π-calculus:

- the reduction relation specifies the result of process computations:

\[ c^? \{y\} Q \mid c^! \{x\} P \xrightarrow{i,j} Q[y\leftarrow x] \mid P \]
\[ \ast c^? \{y\} Q \mid c^! \{x\} P \xrightarrow{i,j} Q[y\leftarrow x] \mid \ast c^? \{y\} Q \mid P \]
\[ P + Q \xrightarrow{\epsilon} P \]
\[ P + Q \xrightarrow{\epsilon} Q \]

- the congruence relation reveals redexes:
  - names renaming (α-conversion),
  - structural modifications
    (Commutativity, associativity, and so on).
Example: syntax

\[ S := (\nu \text{ port}) \]

\[
(\text{Instance} | \text{port}!^5[ | \text{port}!^6[ | \text{port}!^7[])
\]

where

\[ \text{Instance} := *\text{port}^0[] (\nu \text{ in})(\nu \text{ out})(\nu \text{ query}) \]

\[
(\text{in}!^1[\text{query}] \\
| \text{in}^2[\text{response}].(\text{out}!^3[\text{response}] | \text{port}!^4[))
\]
Example: computation

\[
(\nu \text{ port})

(\text{Instance} \mid \text{port}!^5[] \mid \text{port}!^6[] \mid \text{port}!^7[])
\]

\[
(0,5) \rightarrow
\]

\[
(\nu \text{ port})(\nu \text{ in}_1)(\nu \text{ out}_1)(\nu \text{ query}_1)

(\text{Instance} \mid \text{port}!^6[] \mid \text{port}!^7[])

\mid \text{in}_1!^1[\text{query}_1]

\mid \text{in}_1?^2[\text{response}].(\text{out}_1!^3[\text{response}] \mid \text{port}!^4[])\]

\[
(2,1) \rightarrow
\]

\[
(\nu \text{ port})(\nu \text{ in}_1)(\nu \text{ out}_1)(\nu \text{ query}_1)

(\text{Instance} \mid \text{port}!^4[] \mid \text{port}!^6[] \mid \text{port}!^7[])

\mid \text{out}_1!^3[\text{query}_1])
\]
Non-standard semantics

A refined semantics in where

- recursive instances of processes are identified with unambiguous markers;
- channel names are enriched with the marker of the process which has declared them.

Example: non-standard configuration

\[ (\nu \text{ port})(\nu \text{ in}_1)(\nu \text{ out}_1)(\nu \text{ query}_1) \]
\[ \begin{cases} (0, \varepsilon, \{ \text{port} \mapsto (\text{port}, \varepsilon) \}) \\ (3, \text{id}, \{ \text{out} \mapsto (\text{out}, \text{id}) \}) \\ (4, \text{id}, \{ \text{response} \mapsto (\text{query}, \text{id}) \}) \\ (6, \varepsilon, \{ \text{port} \mapsto (\text{port}, \varepsilon) \}) \\ (7, \varepsilon, \{ \text{port} \mapsto (\text{port}, \varepsilon) \}) \end{cases} \]
Markers are binary trees:

- leaves are not labeled;
- nodes are labeled with a pair \((i, j) \in Label^2\).

They are recursively calculated when resources are fetched.

**Theorem:** Standard semantics and non-standard semantics are bisimilar.

The proof mainly relies on the consistence of marker allocation.
Abstraction
Abstract interpretation

$$(C, C_0, \rightarrow)$$ is a transition system,

$$S = \{C \mid \exists i \in C_0, \ i \rightarrow^* C\} = \operatorname{lfp} F$$

where $F : X \mapsto C_0 \cup \{C' \mid \exists C \in X, \ C \rightarrow C'\}$

- $$(\varnothing(C), \subseteq, \cup, \emptyset, \cap, C) \xrightarrow{\gamma} (D^#, \subseteq, \cup, \bot, \cap, \top)$$
- an abstract transition relation $\rightsquigarrow$ on $D^#$

Coherence hypothesis:
If $C \in \gamma(C^#)$ and $C \overset{\lambda}{\rightarrow} \overline{C}$, then there exists $\overline{C}^#$ such that $C^# \overset{\lambda}{\rightsquigarrow} \overline{C}^#$ and $\overline{C} \in \gamma(\overline{C}^#)$.

$$\begin{array}{c}
C \\
\gamma
\end{array} \xrightarrow{\gamma} \begin{array}{c}
C \\
\gamma
\end{array}$$

$$\begin{array}{c}
C^# \\
\overset{\lambda}{\rightsquigarrow}
\end{array} \xrightarrow{\lambda} \begin{array}{c}
\overline{C}^#
\end{array}$$

$$S \subseteq \bigcup_{n \in \mathbb{N}} \gamma(F^{n}(\bot))$$

where $F^{#}(C^#) = \alpha(C_0) \cup C^# \cup \bigcup \{\overline{C}^# \mid C^# \rightsquigarrow \overline{C}^#\}$
Control flow analysis

A description of the communication topology:

\[
\begin{align*}
(0, \varepsilon, \{ & \text{port} \mapsto (\text{port}, \varepsilon) \\
3, id, \{ & \text{out} \mapsto (\text{out}, id) \\
\text{response} & \mapsto (\text{query}, id) \\
4, id, \{ & \text{port} \mapsto (\text{port}, \varepsilon) \\
6, \varepsilon, \{ & \text{port} \mapsto (\text{port}, \varepsilon) \\
7, \varepsilon, \{ & \text{port} \mapsto (\text{port}, \varepsilon) \\
\end{align*}
\]

\[\Rightarrow \{(\text{port, port}), (\text{out, out}), (\text{response, query})\}\]

Abstract transition

\[x!^{i,j}[e]P \quad \rightsquigarrow \quad y?^{i,j}[r]Q(r)\]
Occurrences counting analysis

\[ \begin{align*}
(0, \varepsilon, \{ & \text{port } \mapsto (\text{port, } \varepsilon) \\
(3, id, \{ & \text{out } \mapsto (\text{out, } id) \\
& \text{response } \mapsto (\text{query, } id) \\
(4, id, \{ & \text{port } \mapsto (\text{port, } \varepsilon) \\
(6, \varepsilon, \{ & \text{port } \mapsto (\text{port, } \varepsilon) \\
(7, \varepsilon, \{ & \text{port } \mapsto (\text{port, } \varepsilon) \\
\end{align*} \]

Abstract transition

\[ C \overset{(i,j)}{\rightarrow} C' \]

\[ \overline{C}^\# \]
Abstract domains

We design a domain for representing numerical constraints between

- number of occurrences of processes \( \#(i) \);
- number of performed transitions \( \#(i,j) \).

We use the product of

- a non-relational domain:
  \[ \Rightarrow \] the interval lattice;
- a relational domain:
  \[ \Rightarrow \] the lattice of affine relationships.
In interval narrowing

An exact reduction is exponential.

We use:

- **Gauss reduction:**
  \[
  \begin{align*}
  x + y + z &= 1 \\
  x + y + t &= 2 \\
  \end{align*}
  \Rightarrow
  \begin{align*}
  x + y + z &= 1 \\
  t - z &= 1 \\
  \end{align*}
  \]

- **Interval propagation:**
  \[
  \begin{align*}
  x + y + z &= 3 \\
  x \in [0; \infty[ \\
  y \in [0; \infty[ \\
  z \in [0; \infty[ \\
  \end{align*}
  \Rightarrow
  \begin{align*}
  x + y + z &= 3 \\
  x \in [0; 3] \\
  y \in [0; \infty[ \\
  z \in [0; \infty[ \\
  \end{align*}
  \]

- **Redundancy introduction:**
  \[
  \begin{align*}
  x + y - z &= 3 \\
  x \in [1; 2[ \\
  \end{align*}
  \Rightarrow
  \begin{align*}
  x + y - z &= 3 \\
  y - z \in [1; 2] \\
  x \in [1; 2] \\
  \end{align*}
  \]

to get a polynomial approximated reduction.
Example: non-exhaustion of resources

\[ S := (\nu \text{ port}) \]
\[
(\text{Instance} \mid \text{port}!^5[] \mid \text{port}!^6[] \mid \text{port}!^7[])
\]

where

\[
\text{Instance} := *\text{port}^0[](\nu \text{ in})(\nu \text{ out})(\nu \text{ query})
\]
\[
(\text{in}!^1[\text{query}] \\
| \text{in}^2[\text{response}].(\text{out}!^3[\text{response}] \mid \text{port}!^4[]))
\]

\[
\begin{cases}
\#(0) = 1 \\
\#(3) \in [0; \infty[ \\
\#(i) \in [0; 3[, \forall i \in \{1; 2; 4\} \\
\#(i) \in [0; 1[, \forall i \in [5; 7[ \\
\#(1) + \#(4) + \#(5) + \#(6) + \#(7) = 3 \\
\#(1) = \#(2)
\end{cases}
\]
Example: exhaustion of resources

\[ S := (\nu \text{ port}) \]
\[ (\text{Instance} | \text{port}!^5[] | \text{port}!^6[] | \text{port}!^7[]) \]

where

\[ \text{Instance} := *\text{port}^?^0[](\nu \text{ in})(\nu \text{ out})(\nu \text{ query}) \]
\[ (\text{in}!^1[\text{query}] \]
\[ | \text{in}^?^2[\text{response}].\text{out}!^3[\text{response}] \]
\[ | \text{port}!^4[]) \]

\[
\begin{cases}
#(0) = 1 \\
#(i) \in \left[0; \infty\right[, \forall i \in \{1; 2; 3; 4\} \\
#i \in \left[0; 1\right[, \forall i \in \{5; 6; 7\} \\
#(1) + #(3) = \sum_{i \in \{4, 5, 6, 7\}} #(0, i)
\end{cases}
\]
Example: mutual exclusion

\[
A := \ast a?^1[x](x!^2[a] + c?^3[u]d!^4[u])
\]
\[
B := \ast b?^5[x](x!^6[b] + c!^7[e])
\]
\[
C := a!^8[b]
\]
\[
P := A \mid B \mid C
\]

\[\implies \text{We detect that processes 3 and 7 never communicate.}\]

since the following system:

\[
\begin{align*}
\#(2) + \#(3) + \#(6) + \#(7) + \#(8) &= 1 \\
\#(3) &\in [\llbracket 1; \infty \rrbracket [ \\
\#(7) &\in [\llbracket 1; \infty \rrbracket [ \\
\end{align*}
\]

has no solution in \( \mathbb{N}^+ \).
Trace-based analysis
We want to approximate the set of the configurations by which no infinite computation sequence can pass.

We propose to

1. abstract the trace semantics of a mobile system;

2. for each configuration,
   - approximate the set of the transitions that may occur inside a computation sequence which stems from this configuration;
   - detect and prove whether this set defines a well-founded relation.
Quotient
We finitely partition $\mathcal{D}^\#$

\[
\forall k \in [1; n], \ P_k \neq \bot
\]
\[
\forall k, l \in [1; n], \ k \neq l \implies P_k \cap P_l = \bot
\]
\[
\bigcup (P_k)_{k \in [1; n]} = \top
\]

$\mathcal{D}^\#$

by using our occurrences counting analysis.

We iteratively construct both

- a transition system over $(P_i)$,
- a representation function $f : [1; n] \rightarrow \mathcal{D}^\#$:

If $P_k \cap f(P_k) \overset{(i,j)}{\sim} \overline{C}^\#$ with $\overline{C}^\# \cap P_l \neq \bot$

then

\[
\begin{cases} 
 f(P_l) \leftarrow f(P_l) \cup (\overline{C}^\# \cap P_l) \\
 \text{the transition } P_k \overset{(i,j)}{\rightarrow} P_l \text{ is added}
\end{cases}
\]
Proof of termination

How to check that transition systems are well-founded?

Abstracting environments away,
transition rules look like chemical reactions.

\[ A|B \rightarrow A_1|A_2|...|B_1|B_2|... \] (communication)
\[ C|D \rightarrow C|C_1|C_2|...|D_1|D_2|... \] (resource fetching)

We decompose each transition in two half-transitions:

communication

\[
\begin{align*}
A & \rightarrow A_1|A_2|... \\
B & \rightarrow B_1|B_2|...
\end{align*}
\]

resource fetching

\[
\begin{align*}
C & \rightarrow C \\
D & \rightarrow C_1|C_2|...|D_1|D_2|...
\end{align*}
\]

Then we check if the following relation is well-founded:

communication

\[
\begin{align*}
A & > A_1, \ A > A_2, \ ... \\
B & > B_1, \ B > B_2, \ ...
\end{align*}
\]

resource fetching

\[
\begin{align*}
D & > C_1, \ D > C_2, \ ... \\
D & > D_1, \ D > D_2, \ ...
\end{align*}
\]
Example: a stack

\[ S := (\nu \text{ push})(\nu \text{ pop}) \\
\quad (((\ast \text{ push}^1[](\text{ pop}^2[] \mid \text{ push}^3[])) \\
\mid \ast \text{ pop}^4[] \\
\mid \ast \text{ push}^5[] \\
\mid \text{ push}^6[])) \]

\[
\begin{align*}
\pi(1) &= 1, \quad \pi(2) \in [0; +\infty[, \quad \pi(3) \in [0; 1], \\
\pi(4) &= 1, \quad \pi(5) = 1, \quad \pi(6) \in [0; 1], \\
\underline{\pi}(1, 6) &\in [0; 1], \quad \underline{\pi}(5, 3) \in [0; 1], \quad \underline{\pi}(5, 6) \in [0; 1], \\
\underline{\pi}(1, 3) &\in [0; \infty[, \quad \underline{\pi}(4, 2) \in [0; \infty[. \\
\end{align*}
\]

The analysis has proved that the computations of our system are bound to terminate as soon as a communication \((5, 3)\) or a communication \((5, 6)\) is performed.
Conclusion

- Our framework allows to infer a sound uniform description of mobile systems in the $\pi$-calculus.
- It has succeeded in proving:
  - non-exhaustion, in a polynomial time;
  - mutual exclusion, in a polynomial time;
  - some deadlocks, in an exponential time.

Future Works

- Refining our initial partitioning;
- investigating other approximated algorithms;
- designing a modular analysis;
- including administrative sites (mobile ambients).
  \[\rightarrow\] To analyze big programs.