Occurrence Counting Analysis for the π-calculus

Jérôme Feret

Laboratoire d'Informatique de l'École Normale Supérieure
ENS-LIENS, 45, rue d'Ulm, 75230 PARIS cédex 5, FRANCE

Abstract

We propose an abstract interpretation-based analysis for automatically proving non-trivial properties of mobile systems of processes. We focus on properties relying on the number of occurrences of processes during computation sequences, such as mutual exclusion and non-exhaustion of resources.

We design a non-standard semantics for the π-calculus in order to explicitly trace the origin of channels and to solve efficiently problems set by α-conversion and non-deterministic choices. We abstract this semantics into an approximate one. The use of a relational domain for counting the occurrences of processes allows us to prove quickly and efficiently properties such as mutual exclusion and non-exhaustion of resources. At last, dynamic partitioning allows us to detect some configurations by which no infinite computation sequences can pass.

1 Introduction

We are interested in automatically proving non-trivial properties of mobile systems of processes. We focus on properties relying on a good description of the multiset of processes that occur inside computation sequences, such as mutual exclusion and non-exhaustion of resources, for instance.

We propose an abstract interpretation-based analysis for the full π-calculus [19,18]. Since, the π-calculus is a communication-based formalism, no analysis can be done without a good approximation of the control-flow. Following Venet's methodology [23], we introduce a non-standard semantics to explicitly capture the origin of channels and to describe non-uniform distributions of processes. Our semantics considers the full π-calculus and is optimized to deal efficiently with non-deterministic choices (no useless thread is created).

We use the abstract interpretation framework [8,6,10] to derive an approximate semantics to analyze both the interaction between processes and the number of occurrences of these processes. We propose a well adapted

This is a preliminary version. The final version can be accessed at
URL: http://www.elsevier.nl/locate/entcs/volume39.html

domain to detect quickly mutual exclusion and non-exhaustion of resources. Our approach relies on the use of a relational domain the height of which is quadratic on the number of distinct counted processes. This relational domain helps in calculating properties of interest in a non-relational domain, by reduction. Complexity problems are solved by using approximated algorithms for calculating this reduction. Number of occurrences of processes defines a good criterion for partitioning: we use dynamic partitioning [3] to abstract precisely the trace semantics of mobile systems, which allows us to detect a set of configurations such that no infinite computation sequence can pass by a configuration in this set. Our methodology may very likely be adapted to other formalisms, such as the mobile ambients [4] for instance.

In Section 3, we define the standard semantics of the π-calculus. We define our non-standard semantics in Section 4. We design a generic abstract analysis in Section 5, and instantiate it in Section 6. We show how to use dynamic partitioning in Section 7.

2 Related work

Counting occurrences and analyzing the control flow of a system are deeply related. We cannot count number of occurrences of processes in a system without a good knowledge of its control flow, but counting occurrences helps in getting a more precise knowledge of the control flow by detecting mutual exclusion between processes and by providing a good criterion for partitioning. Our analysis can directly be combined with our previous analysis [14] to infer a non-uniform description of the interactions between the processes, but can also be adapted to many other flow analyses [2,23].

Only a very few analyses for counting occurrences of processes have been published. [15] proposes an exponential analysis for counting occurrences of processes inside ambients. [20] uses context-dependent counts for infering a more accurate description of the internal structure of processes at the expense of a higher time complexity (an exponential number of processes are distinguished). These analyses encounter the same problem: when a process occurs several times, they cannot decide whether this process is still occuring one or several times after being computed, so they have to consider the two possible cases, which leads to both a loss of precision and an exponential explosion. The use of both a relational domain for globally abstracting sets of multisets of processes and an approximated reduction allows us to solve this problem efficiently. We obtain a very accurate analysis which is polynomial on the number of distinguished processes.

3 π-calculus

The π-calculus [18,19] is used for describing mobile systems of processes which communicate channel names via channels. We consider a lazy synchronous

version of the polyadic π-calculus, inspired by the lazy asynchronous version introduced by Turner [21] and the chemical abstract machine [1] in which communication primitives are very simple, while ensuring the same expressive power. Let *Channel* be a countable set of channel names, the standard semantics of the π-calculus, given in Figure 1, relies on the use of both a reduction relation to define results of process computations, and a congruence relation to reveal redexes.

Example 3.1 We model by a mobile system $\mathcal{S}$ an *ftp* protocol for a server which cannot establish more than three simultaneous connections:

$$\mathcal{S} := (\nu\ \text{port})(\textbf{Allocate} \mid \text{port}![] \mid \text{port}![] \mid \text{port}![])$$

where

$$\begin{aligned}\textbf{Allocate} := {} & *\text{port}?[](\nu\ in)(\nu\ out)(\nu\ query)\\ & (in![query]\\ & \mid in?[response].(out![response] \mid \text{port}![]))\end{aligned}$$

Each message port![] occurring at top level symbolizes an available connection. When a customer requests an available connection, three channel names *query*, *in* and *out* are created. The customer sends its query, represented by the channel *query*, via the channel *in*. The server computes this query and sends it back via the channel *out*. A new message port![] is then spawn, which symbolizes that the connection is released. We shall notice that many computational aspects are abstracted away, since we present only an approximation of a realistic server. We finally propose a short computation for $\mathcal{S}$ as follows:

$$\begin{aligned}\mathcal{S} \quad = \quad & (\nu\ \text{port})\\ & \quad (\textbf{Allocate} \mid \text{port}![] \mid \text{port}![] \mid \text{port}![])\\ \rightarrow \quad & (\nu\ \text{port})(\nu\ in_1)(\nu\ out_1)(\nu\ query_1)\\ & \quad (\ \textbf{Allocate} \mid \text{port}![] \mid \text{port}![]\\ & \quad \mid in_1![query_1]\\ & \quad \mid in_1?[response].(out_1![response] \mid \text{port}![]))\\ \rightarrow \quad & (\nu\ \text{port})(\nu\ in_1)(\nu\ out_1)(\nu\ query_1)\\ & \quad (\ \textbf{Allocate} \mid \text{port}![] \mid \text{port}![] \mid \text{port}![]\\ & \quad \mid out_1![query_1])\end{aligned}$$

□

As illustrated in the above example, the configuration of a mobile system is at any stage congruent with a configuration of the form $(\nu\ c)(\mathrm{P}_1 \mid ... \mid \mathrm{P}_n)$, where c is a sequence of channel names, and P_1, ..., P_n are syntactic copies of sub-processes which have been substituted during communications. Nevertheless, standard semantics allows to trace neither the origin of those processes, nor the origin of the channels they have declared.

4 Non-standard semantics

A non-standard semantics [23] is a refined semantics, which explicitly specifies the link between channels and the instances of processes which have declared them. It identifies any instance of a process by an unambiguous marker

$$
\begin{array}{rcll}
P & ::= & \text{action}.P & \text{(Action)} \\
 & | & (P \mid P) & \text{(Parallel composition)} \\
 & | & (P + P) & \text{(Non-deterministic choice)} \\
 & | & \emptyset & \text{(End of a process)} \\
 & & & \\
\text{action} & ::= & c![x_1, ..., x_n] & \text{(Message)} \\
 & | & c?[x_1, ..., x_n] & \text{(Input guard)} \\
 & | & *c?[x_1, ..., x_n] & \text{(Replication guard)} \\
 & | & (\nu\ x) & \text{(Channel creation)}
\end{array}
$$

where c, x_1, ..., x_n, $x \in \mathit{Channel}$, $n \geqslant 0$. Input guard, replication guard and channel creation are the only name binders, i.e in $c?[x_1, ..., x_n]P$, $*d?[y_1, ..., y_p]Q$ and $(\nu\ x)R$, occurrences of x_1, ..., x_n in P, y_1, ..., y_p in Q and x in R are considered bound. Usual rules about scoping, substitution and α-conversion apply. We denote by $\mathcal{FN}(P)$ the set of free names of P, i.e names which are not under the scope of a binder and by $\mathcal{BN}(P)$ the set of bound names of P.

(a) Syntax

$$
\begin{array}{rcll}
(\nu\ x)P & \equiv & (\nu\ y)P[x \leftarrow y] \quad \text{if } y \notin \mathcal{FN}(P) & (\alpha\text{-conversion}) \\
P \mid Q & \equiv & Q \mid P & \text{(Commutativity)} \\
P \mid (Q \mid R) & \equiv & (P \mid Q) \mid R & \text{(Associativity)} \\
P \mid \emptyset & \equiv & P & \text{(End of a process)} \\
(\nu\ x)(\nu\ y)P & \equiv & (\nu\ y)(\nu\ x)P & \text{(Swapping)} \\
((\nu\ x)P) \mid Q & \equiv & (\nu\ x)(P \mid Q) \quad \text{if } x \notin \mathcal{FN}(Q) & \text{(Extrusion)}
\end{array}
$$

where x, $y \in \mathit{Channel}$

(b) Congruence relation

$$
\begin{array}{rl}
c![x_1, ..., x_n]P \mid c?[y_1, ..., y_n]Q \rightarrow & P \mid \widetilde{Q} \quad \text{(communication)} \\
c![x_1, ..., x_n]P \mid *c?[y_1, ..., y_n]Q \rightarrow & P \mid \widetilde{Q} \mid *c?[y_1, ..., y_n]Q \quad \text{(res. fetching)} \\
P + Q \rightarrow & P \quad \text{(left choice)} \\
P + Q \rightarrow & Q \quad \text{(right choice)}
\end{array}
$$

$$
\frac{P \rightarrow Q}{(\nu\ x)P \rightarrow (\nu\ x)Q} \qquad \frac{P' \equiv P \quad P \rightarrow Q \quad Q \equiv Q'}{P' \rightarrow Q'} \qquad \frac{P \rightarrow P'}{P \mid Q \rightarrow P' \mid Q}
$$

where c, x, x_1, ..., x_n, y_1, ..., $y_n \in \mathit{Channel}$
and $\widetilde{Q} = Q[y_1 \leftarrow x_1, ..., y_n \leftarrow x_n]$

(c) Reduction relation

Fig. 1. The chemical semantics

in order to distinguish each instance of a recursive process from all others. Then, the origin of channel names is easily traced by identifying each channel name with the marker of the process which has created it. In [14] we propose a non-standard semantics which considers the full π-calculus without non-deterministic choices. We redefine it in order to take them into account efficiently, without adding further reduction rules nor considering useless syntactic components. This allows for an easier analysis by making abstract domains smaller while giving a good intuition on the potential evolutions of the analyzed systems. For that purpose, we restrict the set of computations to those where non-deterministic choices are made as soon as possible. In such computations, no further communication nor resource fetching is performed while there are non-deterministic choices at the top level. This assumption does not change the subset of reachable standard configurations which contain no non-deterministic choice at top level.

Let Lbl be an infinite set of labels, we denote by $\mathcal{M}$ the set of all binary trees the leaves of which are not labelled (ε) and the nodes of which are labelled with a pair (i, j) where both i and j are in Lbl. The tree, having a node labelled a, a left sibling t_1 and a right sibling t_2 is denoted by $N(a, t_1, t_2)$. We use $\mathcal{M}$ as a set of markers and denote by Σ the set $Lbl \times Lbl$. Σ is used in labelling non-standard transitions. We consider a closed mobile system $\mathcal{S}$ in the π-calculus and assume without any loss of generality that two channel binders of $\mathcal{S}$ are never used on the same channel name. We locate syntactic components of $\mathcal{S}$ by marking each sign ? or ! occurring in $\mathcal{S}$ with distinct labels in Lbl. A non-standard configuration is a set of thread instances, where a thread instance is a triplet composed with a syntactic component, a marker and an environment. The syntactic component is a copy of a sub-process of $\mathcal{S}$, the marker is calculated at the creation of the thread and the environment specifies the semantic value of each free name in the syntactic component. Thread instances are created at the beginning of the system computation and during execution. In both cases, several threads are spawned, corresponding to a set of syntactic components, in accordance to which non-deterministic choices are made. Applying the function $Agent$, defined as follows, to either $\mathcal{S}$ for initial threads, or to the continuation of running processes, gives the set of all possibilities for the set of spawned syntactic components.

$$
\begin{aligned}
Agent(\emptyset) &= \{\{\}\} \\
Agent(x!^i[x_1, ..., x_n]P) &= \{\{x!^i[x_1, ..., x_n]P\}\} \\
Agent(y?^i[y_1, ..., y_n]P) &= \{\{y?^i[y_1, ..., y_n]P\}\} \\
Agent(*y?^i[y_1, ..., y_n]P) &= \{\{*y?^i[y_1, ..., y_n]P\}\} \\
Agent(P \mid Q) &= \{A \cup B \mid A \in Agent(P),\ B \in Agent(Q)\} \\
Agent(P + Q) &= Agent(P) \cup Agent(Q) \\
Agent((\nu\ x)P) &= Agent(P)
\end{aligned}
$$

The markers of initial threads are ε, while the markers of new threads are calculated recursively from the marker of the threads whose computation has

led to their creation:

- when an execution does not involve fetching a resource, the marker of the computed thread is just passed to the threads in its continuation;
- when a resource is fetched, the markers of the new threads created from the continuation of the resource are $N((i,j), id_*, id_!)$, where (i,id_*,E_*) is the fetched resource thread and $(j,id_!,E_!)$ the message sender thread.

Environments map each free channel name of syntactic components to a pair (a, b) where a is a bound channel name of $\mathcal{S}$, and b is a marker. Intuitively, a refers to the binder $(\nu\ a)$ which has been used in declaring the channel, and b is the marker of the thread which has declared it. While threads are running, environments are calculated in order to mimic the standard semantics.

We denote by $\mathcal{C}$ the set of all non-standard configurations. Our non-standard semantics is given in Figure 2. The function $\mathcal{C}_0$ gives the set of possible initial configurations, while the relation $\longrightarrow_2$ defines non-standard computation steps. Each non-standard communication steps are labelled with a pair $(i, j) \in \Sigma$, where i is the label of the message receiver and j is the label of the message sender. There is a *bisimulation* between standard and non-standard semantics, provided that we restrict the set of standard computations to those where all non-deterministic choices are always made before communications and resource fetching. The proof relies on the fact that non-standard computations cannot yield conflicts between the markers of the threads.

5 Abstraction

The set of all possible non-standard configurations a system may take during a finite computation sequence is given by its collecting semantics [7] and can be expressed as the least fixpoint of the following $\cup$-complete endomorphism $\mathbb{F}$ on the complete lattice $\wp(\Sigma^* \times \mathcal{C})$:

$$\mathbb{F}(X) = \{(\varepsilon, C) \mid C \in \mathcal{C}_0(\mathcal{S})\} \cup \{(u.\lambda, C) \mid \exists (u, C') \in X,\ C' \xrightarrow{\lambda}_2 C\}$$

We use abstract interpretation [8] to design an abstract domain in which a decidable description of $\mathit{Coll}(\mathcal{S})$ will be computed. Our abstract domain is the reduced product of two domains: the first one describes the control-flow of $\mathcal{S}$; the second one counts occurrences of its processes. Since the π-calculus is a communication-based formalism, any further analysis requires a good approximation of the communication topology. Several analyses [2,22,23,14] have already been proposed. For the sake of simplicity, we use a naive uniform analysis to abstract the control-flow of systems, but it could be enriched by using the non-uniform analysis proposed in [14]. We introduce the set $\mathcal{BN}(\mathcal{S})^2$ as the set of all possible interactions between agents of the system $\mathcal{S}$; intuitively the pair (x, y) denotes that the channel name x may be bound to a channel created by the binder $(\nu\ y)$. Our first abstract domain is then the complete

$$\mathcal{C}_0(\mathcal{S}) = \{\{(p, \varepsilon, E_p) \mid p \in Cont)\} \mid Cont \in Agent(\mathcal{S})\}$$
$$\text{where } E_p = \begin{cases} \mathcal{FN}(p) & \to \mathcal{BN}(\mathcal{S}) \times \mathcal{M} \\ x & \mapsto (x, \varepsilon) \end{cases}$$

(a) Set of initial non-standard configurations

If C is a non-standard configuration,
if there are λ, μ in C,
with λ=$(y?^i[y_1, ..., y_n]P, id_?, E_?)$ and μ=$(x!^j[x_1, ..., x_n]Q, id_!, E_!)$
such that $E_?(y) = E_!(x)$,
if $Cont_P$ is in $Agent(P)$ and $Cont_Q$ is in $Agent(Q)$,

then $C \xrightarrow{(i,j)}_2 C'$

where $C' = (C \setminus \{\lambda, \mu\}) \cup (f_?(Cont_P)) \cup (f_!(Cont_Q))$,

$$f_? : Ag \mapsto \left(Ag, id_?, \begin{cases} z \mapsto E_?(z) & \text{if } z \in \mathcal{FN}(Ag) \cap \mathcal{FN}(y?^i[y_1, ..., y_n]P) \\ y_k \mapsto E_!(x_k) & \text{if } y_k \in \mathcal{FN}(Ag) \\ z \mapsto (z, id_?) & \text{if } \begin{cases} z \in \mathcal{FN}(Ag) \cap \mathcal{BN}(y?^i[y_1, ..., y_n]P) \\ z \notin \{y_k | k \in [|1; n|]\} \end{cases} \end{cases} \right)$$

$$\text{and } f_! : Ag \mapsto \left(Ag, id_!, \begin{cases} z \mapsto E_!(z) & \text{if } z \in \mathcal{FN}(Ag) \cap \mathcal{FN}(x!^j[x_1, ..., x_n]Q) \\ z \mapsto (z, id_!) & \text{if } z \in \mathcal{FN}(Ag) \cap \mathcal{BN}(x!^j[x_1, ..., x_n]Q) \end{cases} \right)$$

(b) Non-standard communication

If C is a non-standard configuration,
if there are λ, μ in C,
with λ=$(*y?^i[y_1, ..., y_n]P, id_?, E_?)$ and $\mu = (x!^j[x_1, ..., x_n]Q, id_!, E_!)$
such that $E_?(y) = E_!(x)$,
if $Cont_P$ is in $Agent(P)$ and $Cont_Q$ is in $Agent(Q)$,

then $C \xrightarrow{(i,j)}_2 C'$

where $C' = (C \setminus \{\mu\}) \cup (f_?(Cont_P)) \cup (f_!(Cont_Q))$,
$id_* = N((i, j), id_?, id_!)$,

$$f_? : Ag \mapsto \left(Ag, id_*, \begin{cases} z \mapsto E_?(z) & \text{if } z \in \mathcal{FN}(Ag) \cap \mathcal{FN}(y?^i[y_1, ..., y_n]P) \\ y_k \mapsto E_!(x_k) & \text{if } y_k \in \mathcal{FN}(Ag) \\ z \mapsto (z, id_*) & \text{if } \begin{cases} z \in \mathcal{FN}(Ag) \cap \mathcal{BN}(y?^i[y_1, ..., y_n]P) \\ z \notin \{y_k | k \in [|1; n|]\} \end{cases} \end{cases} \right)$$

$$\text{and } f_! : Ag \mapsto \left(Ag, id_!, \begin{cases} z \mapsto E_!(z) & \text{if } z \in \mathcal{FN}(Ag) \cap \mathcal{FN}(x!^j[x_1, ..., x_n]Q) \\ z \mapsto (z, id_!) & \text{if } z \in \mathcal{FN}(Ag) \cap \mathcal{BN}(x!^j[x_1, ..., x_n]Q) \end{cases} \right)$$

(c) Non-standard resource fetching

Fig. 2. Non-standard semantics

lattice $\wp(\mathcal{BN}(\mathcal{S})^2)$ related to our concrete domain via a Galois connection $(\alpha_{\rm com},\gamma_{\rm com})$ defined as follows:

$$\alpha_{\rm com}(A) = \left\{ (x,y) \;\middle|\; \begin{array}{l} \exists (u,C) \in A,\ \exists (P,id_1,E) \in C, \exists id_2 \in \mathcal{M} \\ \text{such that } x \in \mathcal{FN}(P) \text{ and } E(x) = (y,id_2) \end{array} \right\}$$

$$\gamma_{\rm com}(A^\sharp) = \left\{ (u,C) \;\middle|\; \begin{array}{c} \forall (P,id_1,E) \in C,\ \forall x \in \mathcal{FN}(P),\ \forall y \in \mathcal{BN}(\mathcal{S}) \\ [\exists id_2,\ E(x) = (y,id_2)] \Longrightarrow (x,y) \in A^\sharp \end{array} \right\}$$

The second domain counts both the number of occurrences of processes and the number of performed transitions inside computations. We denote by Π the set of all sub-processes of $\mathcal{S}$. Let $\mathcal{V}$ be the set $\Pi + \Sigma$. We consider $\wp(\mathbb{N}^{\mathcal{V}})$, the complete lattice of the sets of natural number functions defined on $\mathcal{V}$. $\wp(\mathbb{N}^{\mathcal{V}})$ is related to our concrete domain via a Galois connection $(\alpha_{\mathbb{N}^{\mathcal{V}}}, \gamma_{\mathbb{N}^{\mathcal{V}}})$, defined as follows:

$$\alpha_{\mathbb{N}^{\mathcal{V}}}(A) = \left\{ \left\{ \begin{array}{ll} \mathcal{V} & \to \mathbb{N} \\ v \in \Pi & \mapsto \mathrm{Card}(\{(P,id,E) \in C \mid v = P\}) \\ \lambda \in \Sigma & \mapsto |u|_\lambda \end{array} \right. \;\middle|\; (u,C) \in A \right\}$$

$$\gamma_{\mathbb{N}^{\mathcal{V}}}(A^\sharp) = \left\{ (u,C) \;\middle|\; \begin{array}{l} \exists f \in A^\sharp, \forall \lambda \in \Sigma,\ |u|_\lambda = f(\lambda), \\ \forall v \in \Pi,\ \mathrm{Card}(\{(P,id,E) \in C \mid v = P\}) = f(v) \end{array} \right\}$$

The complete lattice, $(\mathcal{N}_\mathcal{V}, \sqsubseteq_{\mathcal{N}_\mathcal{V}}, \sqcup_{\mathcal{N}_\mathcal{V}}, \bot_{\mathcal{N}_\mathcal{V}}, \sqcap_{\mathcal{N}_\mathcal{V}}, \top_{\mathcal{N}_\mathcal{V}})$, left as a parameter of our abstraction, is related to $\wp(\mathbb{N}^{\mathcal{V}})$ by a Galois connection $(\alpha_{\mathcal{N}_\mathcal{V}}, \gamma_{\mathcal{N}_\mathcal{V}})$ which satisfies the condition: $\gamma_{\mathcal{N}_\mathcal{V}}(\bot_{\mathcal{N}_\mathcal{V}}) = \emptyset$. We require three abstract primitives which satisfy the following soundness hypotheses:

- *required* : $\wp(\Pi) \times \mathcal{N}_\mathcal{V} \to \mathcal{N}_\mathcal{V}$
$$\gamma_{\mathcal{N}_\mathcal{V}}(v^\sharp) \cap \{f \mid f(p) \geqslant 1,\ \forall p \in A\} \subseteq \gamma_{\mathcal{N}_\mathcal{V}}(\mathit{required}(A, v^\sharp))$$
- *trans* : $\mathcal{N}_\mathcal{V} \times \mathbb{N}^{\mathcal{V}} \to \mathcal{N}_\mathcal{V}$
$$\{x \mapsto f(x) + g(x) \mid g \in \gamma_{\mathcal{N}_\mathcal{V}}(v^\sharp)\} \subseteq \gamma_{\mathcal{N}_\mathcal{V}}(\mathit{trans}(v^\sharp, f))$$
- ρ : $\mathcal{N}_\mathcal{V} \to \mathcal{N}_\mathcal{V}$
$$\gamma_{\mathcal{N}_\mathcal{V}}(v^\sharp) \subseteq \gamma_{\mathcal{N}_\mathcal{V}}(\rho(v^\sharp))$$

Roughly speaking, ρ is a reduction, it maps each abstract value to another one which represents the same set but in which properties are easier to establish. Particularly, it will be used for proving that an abstract value represents the empty set. At each abstract computation step, *required* is used to extract from the abstract value the representation of configurations which simultaneously contain all the processes required by the computation step and *trans* is used to calculate the representation of the result of the computation step by taking into account newly spawned and destroyed processes.

We define our abstract domain $(\mathcal{C}^\sharp, \sqsubseteq, \sqcup, \bot, \sqcap, \top)$ as the complete lattice $(\wp(\mathcal{BN}(\mathcal{S})^2) \times \mathcal{N}_\mathcal{V})$, where $\sqsubseteq$, $\sqcup$, $\bot$, $\sqcap$ and $\top$ are defined pairwise. $\mathcal{C}^\sharp$ is related to $\wp(\Sigma^* \times \mathcal{C})$ by a Galois connection (α, γ), where $\alpha(A) = \alpha_{\rm com}(A), [\alpha_{\mathcal{N}_\mathcal{V}} \circ \alpha_{\mathbb{N}^{\mathcal{V}}}](A))$ and $\gamma(A,B) = \gamma_{\rm com}(A) \cap [\gamma_{\mathbb{N}^{\mathcal{V}}} \circ \gamma_{\mathcal{N}_\mathcal{V}}](B)$. Our abstract semantics is defined by a transition relation $\rightsquigarrow$ on our abstract domain $\mathcal{C}^\sharp$, given in Figure

3. Soundness hypotheses on abstract primitives ensure the soundness of our abstract transition:

Proposition 5.1 *If* $(u, C) \in \gamma(C^\sharp)$ *and* $C \xrightarrow{\lambda}_2 \overline{C}$*, then there exists* $\overline{C}^\sharp$ *such that* $C^\sharp \overset{\lambda}{\rightsquigarrow} \overline{C}^\sharp$ *and* $(u.\lambda, \overline{C}) \in \gamma(\overline{C}^\sharp)$.

As a consequence, the abstract counterpart $\mathbb{F}^\sharp$ of $\mathbb{F}$, defined by

$$\mathbb{F}^\sharp(C^\sharp) = (\alpha(\{(\varepsilon, C) \mid C \in \mathcal{C}_0(\mathcal{S})\})) \sqcup C^\sharp \sqcup \left(\bigsqcup\{\overline{C^\sharp} \mid \exists \lambda \in \Sigma,\ C^\sharp \overset{\lambda}{\rightsquigarrow} \overline{C^\sharp}\}\right)$$

satisfies the soundness condition $\mathbb{F}(C) \subseteq \gamma(\mathbb{F}^\sharp(\alpha(C)))$. Using Kleene's theorem, we obtain the soundness of our analysis:

Theorem 5.2 $lfp_\emptyset \mathbb{F} \subseteq \bigcup_{n \in \mathbb{N}} [\gamma \circ \mathbb{F}^{\sharp n}](\bot)$

Following [6,7], we compute a sound approximation of our abstract semantics by using a widening operator $\nabla : \mathcal{C}^\sharp \times \mathcal{C}^\sharp \to \mathcal{C}^\sharp$ which satisfies the following properties:

- $\forall C_1^\sharp,\ C_2^\sharp \in \mathcal{C}^\sharp,\ C_1^\sharp \sqcup C_2^\sharp \sqsubseteq C_1^\sharp \nabla C_2^\sharp$
- $(C_n^\sharp) \in \left(\mathcal{C}^\sharp\right)^{\mathbb{N}}$, the sequence (C_n^∇) defined as

$$\begin{cases} C_0^\nabla = C_0^\sharp \\ C_{n+1}^\nabla = C_n^\nabla \nabla C_{n+1}^\sharp \end{cases}$$

is ultimately stationary.

The abstract iteration of $\mathbb{F}^\sharp$ is then defined as follows:

$$\begin{cases} \mathbb{F}_0^\nabla = \bot \\ \mathbb{F}_{n+1}^\nabla = \begin{cases} \mathbb{F}_n^\nabla & \text{if } \mathbb{F}^\sharp(\mathbb{F}_n^\nabla) \sqsubseteq \mathbb{F}_n^\nabla \\ \mathbb{F}_n^\nabla \nabla \mathbb{F}^\sharp(\mathbb{F}_n^\nabla) & \text{else} \end{cases} \end{cases}$$

Theorem 5.3 ***Abstract iteration[10,11]*** *Abstract iteration* $(\mathbb{F}_n^\nabla)$ *is ultimately stationary and its limit* $\mathbb{F}^\nabla$ *satisfies* $Coll(\mathcal{S}) \subseteq \gamma(\rho(\mathbb{F}^\nabla))$.

6 Detecting exhaustion of resources and mutual exclusion

We only need to define an abstract domain to approximate set of tuples of natural numbers, in which abstract primitives can be precisely and efficiently implemented. We reject the use of usual numerical domains: we are unlikely to design a precise primitive *required* in non-relational domain, without using an exponential partitioning; we think that the domain of linear inequalities among a finite set of variables [12] is too expensive because we deal with too many variables. We propose the use of a product of two domains. The first domain is

Let $(c^\sharp, v^\sharp) \in \mathcal{C}^\sharp$, $u \in \mathcal{BN}(\mathcal{S})$, $y?^i[y_1, ..., y_n]P$ and $x!^j[x_1, ..., x_n]Q$ two subprocesses, $\text{Cont}_? \in \mathit{Agent}(P)$, $\text{Cont}_! \in \mathit{Agent}(Q)$, such that:

- $(y, u) \in c^\sharp$
- $(x, u) \in c^\sharp$
- $V \triangleq \rho(\mathit{required}(\{y?^i[y_1, ..., y_n]P; x!^j[x_1, ..., x_n]Q\}, v^\sharp)) \neq \bot_{\mathcal{N}_\mathcal{V}}$

then $(c^\sharp, v^\sharp) \overset{(i,j)}{\rightsquigarrow} (c'^\sharp, v'^\sharp)$, where

- $c'^\sharp = c^\sharp \cup \{(y_k, t) \mid k \in [|1; n|],\ t \in \mathcal{BN}(\mathcal{S}) \text{ and } (x_k, t) \in c^\sharp\}$
  $\cup \{(x, x) \mid \exists p \in \text{Cont}_?,\ x \in (\mathcal{BN}(P) \cap \mathcal{FN}(p)) \setminus \{y_k \mid k \in [|1; n|]\}\}$
  $\cup \{(x, x) \mid \exists q \in \text{Cont}_!,\ x \in \mathcal{BN}(Q) \cap \mathcal{FN}(q)\}$
- $v'^\sharp = \mathit{trans}(V, \delta)$

with

$$\forall x \in \Pi,\ \delta(x) = \begin{cases} -1 & \text{if } x \in \{y?^i[y_1, ..., y_n]P, x!^j[x_1, ..., x_n]Q\} \setminus (\mathit{cont}_? \cup \mathit{cont}_!) \\ +1 & \text{if } x \in (\mathit{cont}_? \cup \mathit{cont}_!) \setminus \{y?^i[y_1, ..., y_n]P, x!^j[x_1, ..., x_n]Q\} \\ 0 & \text{otherwise} \end{cases}$$

$$\forall \lambda \in \Sigma,\ \delta(\lambda) = \begin{cases} +1 & \text{if } \lambda = (i, j) \\ 0 & \text{otherwise} \end{cases}$$

(a) Abstract communication

Let $(c^\sharp, v^\sharp) \in \mathcal{C}^\sharp$, $u \in \mathcal{BN}(\mathcal{S})$, $*y?^i[y_1, ..., y_n]P$ and $x!^j[x_1, ..., x_n]Q$ two subprocesses, $\text{Cont}_* \in \mathit{Agent}(P)$, $\text{Cont}_! \in \mathit{Agent}(Q)$, such that:

- $(y, u) \in c^\sharp$
- $(x, u) \in c^\sharp$
- $V \triangleq \rho(\mathit{required}(\{*y?^i[y_1, ..., y_n]P; x!^j[x_1, ..., x_n]Q\}, v^\sharp)) \neq \bot_{\mathcal{N}_\mathcal{V}}$

then $(c^\sharp, v^\sharp) \overset{(i,j)}{\rightsquigarrow} (c'^\sharp, v'^\sharp)$, where

- $c'^\sharp = c^\sharp \cup \{(y_k, t) \mid k \in [|1; n|],\ t \in \mathcal{BN}(\mathcal{S}) \text{ and } (x_k, t) \in c^\sharp\}$
  $\cup \{(x, x) \mid \exists p \in \text{Cont}_*,\ x \in (\mathcal{BN}(P) \cap \mathcal{FN}(p)) \setminus \{y_k \mid k \in [|1; n|]\}\}$
  $\cup \{(x, x) \mid \exists q \in \text{Cont}_!,\ x \in \mathcal{BN}(Q) \cap \mathcal{FN}(q)\}$
- $v'^\sharp = \mathit{trans}(V, \delta)$

$$\text{with } \forall x \in \Pi,\ \delta(x) = \begin{cases} -1 & \text{if } x \in \{x!^j[x_1, ..., x_n]Q\} \setminus \mathit{cont}_* \\ +1 & \text{if } x \in (\mathit{cont}_* \cup \mathit{cont}_!) \setminus \{x!^j[x_1, ..., x_n]Q\} \\ 0 & \text{otherwise} \end{cases}$$

$$\text{and } \forall \lambda \in \Sigma,\ \delta(\lambda) = \begin{cases} +1 & \text{if } \lambda = (i, j) \\ 0 & \text{otherwise} \end{cases}$$

(b) Abstract resource fetching

Fig. 3. Abstract transition relation

based on the use of the interval lattice and is used for expressing properties of interest. This domain can represent all the information we need to express non-exhaustion of resources, but it cannot calculate them precisely without being refined. The second domain is based on the use of linear equalities between variables [17] and is used for expressing more complex properties, such as mutual exclusion for instance, which allows for more precise calculations in the first domain. The power of our analysis directly follows from an unexpensive algorithm, straightforwardly adapted from Linear Constraint Programming, to calculate an approximated reduction between these two domains.

The complete lattice $(\mathcal{I}_\mathcal{V}, \sqsubseteq_{\mathcal{I}_\mathcal{V}}, \sqcup_{\mathcal{I}_\mathcal{V}}, \bot_{\mathcal{I}_\mathcal{V}}, \sqcap_{\mathcal{I}_\mathcal{V}}, \top_{\mathcal{I}_\mathcal{V}})$ is the functional domain of the natural numbers intervals, where lattice operations are defined pointwise. A family $(\nabla^n_{\mathcal{I}_\mathcal{V}})$ of widening operators on $\mathcal{I}_\mathcal{V}$ is defined as follows:

$$[f\nabla^n_{\mathcal{I}_\mathcal{V}}g](x) = f(x)\nabla^n g(x)$$

$$\text{where } \begin{cases} [|a;b|] \ \nabla^n \ [|c;d|] = [|min\{a;c\};\infty|[\text{ if } d > max\{b;n\} \\ I \quad \nabla^n \quad J \quad = \quad I \cup J \qquad \text{otherwise} \end{cases}$$

The complete lattice $(\mathcal{K}_\mathcal{V}, \sqsubseteq_{\mathcal{K}_\mathcal{V}}, \cup_{\mathcal{K}_\mathcal{V}}, \top_{\mathcal{K}_\mathcal{V}}, \cap_{\mathcal{K}_\mathcal{V}}, \bot_{\mathcal{K}_\mathcal{V}})$ of linear equality systems between the finite set of variables $\mathcal{V}$ is described with its lattice operations in [17]. This domain uses Gauss reduction in order to normalize systems. Moreover, since there are no infinite increasing chain [17], we can choose $\cup_{\mathcal{K}_\mathcal{V}}$ as a widening operator. Our numerical domain is then the product $\mathcal{I}_\mathcal{V} \times \mathcal{K}_\mathcal{V}$. Generic primitives are expressed as follows:

- $\mathit{required}(P,(i,s)) = (i',s)$ where $\begin{cases} i'(x) = i(x) \cap [|1;+\infty|[& \forall x \in P \\ i'(x) = i(x) & \forall x \in \Pi \setminus P \end{cases}$
- $\mathit{trans}\left(\left(i, \left\{\sum_{v\in\mathcal{V}} a^k_v v = b_k, \ \forall k \in [|1;m|]\right\}\right), f\right) = (i',s')$

  where $\begin{cases} i'(x) & = \{k + f(x) | \forall k \in i(x)\} \cap [|0;+\infty|[\\ s' & = \left\{\sum_{v\in\mathcal{V}} a^k_v (v - f(v)) = b_k, \ \forall k \in [|1;m|]\right. \end{cases}$

We now present a reduction [9] ρ between $\mathcal{I}_\mathcal{V}$ and $\mathcal{K}_\mathcal{V}$. A reduction consists in taking into account linear constraints in order to narrow the domain of interval variables. For instance, the system of constraints $\{x + y = 12, \ x \in [|3;15|], \ y \in [|4;19|]$ can be reduced to the system $\{x+y = 12, \ x \in [|3;8|], \ y \in [|4;9|]$. Linear constraints are likely to be combined, via Gauss reduction, in order to give new linear constraints which will allow for further reductions. Therefore, generating the whole set of such combinations is likely to require an exponential time of execution.

We propose a two-step-polynomial algorithm for solving this problem. The first step aims at narrowing infinite intervals into finite ones. It uses Gauss reduction to obtain a positive representation of systems of linear equalities, that is to say an equivalent system of equations such that if a variable occurs

with a strictly negative coefficient in an equation, then this variable occurs with a negative coefficient in each equation. Positive representations contain only a few undefined forms, which allows to narrow infinite intervals into finite ones, with a worst-case in $\mathcal{O}(n^3)$. The second step is inspired by [5]: it consists in obtaining a triangular system of constraints of the form $a_1.x_1 + ... + a_n.x_n \in I$ where I is an interval. This system is then used for propagating unidirectionnally intervals from non-diagonal to diagonal variables. The result is a good reduction with a worst-case in $\mathcal{O}(n^4)$.

We now propose some examples of mobile systems analyzed with our prototype. For the sake of brevity, we have selected for each example a subset of significant constraints captured by our analysis; full results are available in [13]. In these constraints, the number of occurrences of a process $c?^i[x_1, ..., x_n]P$ or $c!^i[x_1, ..., x_n]P$ is denoted by $\sharp(i)$, while $\underline{\sharp}(i, j)$ denotes the number of times a communication reduction labelled with (i, j) is used.

Example 6.1 Our first example is the *ftp* protocol proposed in Example 3.1:

$$\mathcal{S} := (\nu\ \text{port})(\textbf{Allocate} \mid \text{port}!^5[] \mid \text{port}!^6[] \mid \text{port}!^7[])$$

where

$$\begin{aligned}\textbf{Allocate} := {} & *\text{port}?^0[](\nu\ \mathit{in})(\nu\ \mathit{out})(\nu\ \mathit{query}) \\ & (\mathit{in}!^1[\mathit{query}] \\ & \mid \mathit{in}?^2[\mathit{response}].(\mathit{out}!^3[\mathit{response}] \mid \text{port}!^4[]))\end{aligned}$$

$$\begin{cases} \sharp(0) = 1,\ \sharp(i) \in [|0; 3|],\ \forall i \in \{1; 2; 4\} \\ \sharp(3) \in [|0; \infty|[\\ \sharp(i) \in [|0; 1|],\ \forall i \in [|5; 6; 7|] \\ \sharp(1) + \sharp(4) + \sharp(5) + \sharp(6) + \sharp(7) = 3 \\ \sharp(3) = \underline{\pi}(2, 1) \end{cases}$$

Our analysis has proved that only three physical channels are required to simulate this protocol. □

Example 6.2 We now propose an example of mutual exclusion:

$$\mathcal{S} := \left(\begin{array}{l} (\nu\ \ \text{a})(\nu\ \ \text{b})(\nu\ \ \text{c})(\nu\ \ \text{d}) \\ \qquad (*\text{a}?^0[x](x!^1[\text{a}] + (\text{c}?^2[]\text{d}!^3[])) \\ \qquad |*\text{b}?^4[x](x!^5[\text{b}] + (\text{c}!^6[])) \\ \qquad |\text{a}!^7[\text{b}]) \end{array} \right)$$

$$\begin{cases} \sharp(i) \in [|0; 1|],\ \forall i \in \{1; 2; 5; 6; 7\} \\ \sharp(3) = 0 \\ \sharp(1) + \sharp(2) + \sharp(5) + \sharp(6) + \sharp(7) = 1 \end{cases}$$

Our analysis has proved that the sub-process $(\text{d}!^3[])$ is unreachable by detecting a mutual exclusion between the sub-processes $(\text{c}?^2[]\text{d}!^3[])$ and $(\text{c}!^6[])$. □

7 Detecting deadlocks

We use both our abstraction of the collecting semantics and dynamic partitioning tools [3] to solve the problem of detecting sets of non-standard configurations such that no infinite sequence of computations can pass by them. For that purpose, we analyze the trace semantics of the mobile system $\mathcal{S}$, which is defined as a transition system $(G,\ A \subseteq G \times G)$ where $G = \mathit{Coll}(\mathcal{S})$ is the set of states and $A = \{(u, C), \lambda, (u.\lambda, C') \mid (u, C) \in \mathit{Coll}(\mathcal{S}),\ C \xrightarrow{\lambda}_2 C'\}$ is the set of transitions. This is also the least fixpoint of an $\cup$-complete endomorphism $\mathbb{F}$ on the complete lattice $(\mathcal{T}, \sqsubset_{\mathcal{T}}, \cup_{\mathcal{T}}, \bot_{\mathcal{T}}, \cap_{\mathcal{T}}, \top_{\mathcal{T}})$ of transition systems on the alphabet Σ, the set of states of which is a subset of the set $(\Sigma^* \times \mathcal{C})$. Lattice operations are the usual set operations, while $\mathbb{F}((G, A))$ is the transition system (G', A') defined as follows:

$$\begin{cases} G' = \{(\varepsilon, C) \mid C \in \mathcal{C}_0(\mathcal{S})\} \cup \{(u.\lambda, C) \mid \exists (u, C') \in G,\ C' \xrightarrow{\lambda}_2 C\} \\ A' = \{((u, C), \lambda, (u.\lambda, C')) \mid (u, C) \in G,\ C \xrightarrow{\lambda}_2 C'\} \end{cases}$$

We abstract the trace semantics to compute a finite approximation in a finite time. We approximate infinite transition systems by finitely partitioning their set of states. An abstract transition system is then defined as a transition system on a finite set of states, and a function mapping each state to the set of concrete states it represents. Let $\mathbb{P}$ be a finite subset of the lattice $\wp(\mathbb{N}^{\mathcal{V}})$, such that $\emptyset \notin \mathbb{P}$, $\bigcup \mathbb{P} = \mathbb{N}^{\mathcal{V}}$ and $\forall a, a' \in \mathbb{P},\ a \neq a' \implies a \cap a' = \emptyset$. We introduce our abstract domain $\mathcal{T}^\sharp$ as the set of all pairs $((G^\sharp, A^\sharp), f)$ such that $(G^\sharp, A^\sharp)$ is a transition system on the alphabet Σ, where $G^\sharp$ is the quotient of $\mathbb{P}$ by an equivalence relation and f is a function mapping each element of $G^\sharp$ to an abstract element of $\mathcal{C}^\sharp$. $\mathcal{T}^\sharp$ is partially pre-ordered by the relation $\sqsubseteq_{\mathcal{T}^\sharp}$ where $((\mathbb{P}_{/\sim_1}, A_1^\sharp), f_1) \sqsubseteq_{\mathcal{T}^\sharp} ((\mathbb{P}_{/\sim_2}, A_2^\sharp), f_2)$ if and only:

$$\begin{cases} \forall q^\sharp, q'^\sharp \in \mathbb{P},\ q^\sharp \sim_1 q'^\sharp \implies q^\sharp \sim_2 q'^\sharp \\ \forall q^\sharp \in \mathbb{P},\ f_1([q^\sharp]_{\sim_1}) \sqsubseteq f_2([q^\sharp]_{\sim_2}) \\ \forall q^\sharp, q'^\sharp \in \mathbb{P},\ \forall \lambda \in \Sigma,\ ([q^\sharp]_{\sim_1}, \lambda, [q'^\sharp]_{\sim_1}) \in {A_1}^\sharp \implies ([q^\sharp]_{\sim_2}, \lambda, [q'^\sharp]_{\sim_2}) \in {A_2}^\sharp \end{cases}$$

Since there is no canonical choice for the equivalence relation, we are unlikely to define an abstraction function. So, we use a relaxed version of abstract interpretation [10], which only relies on the use of a concretization function

$\gamma_{\mathcal{T}}$ which relates the abstract domain to the concrete one, defined as follows:

$$\gamma_{\mathcal{T}}((\mathbb{P}_{/\sim}, A^\sharp), f) = (G, A)$$

$$\text{where } \begin{cases} G = \bigcup\limits_{q^\sharp \in \mathbb{P}} [(\gamma(f([q^\sharp]_\sim))) \cap (\gamma_{\mathbb{N}^\nu}(q^\sharp))] \\ A = \left\{ (q, \lambda, q') \;\middle|\; \exists q^\sharp,\ q'^\sharp \in \mathbb{P},\ \begin{cases} q \in (\gamma(f([q^\sharp]_\sim))) \cap (\gamma_{\mathbb{N}^\nu}(q^\sharp)) \\ q' \in (\gamma(f([q'^\sharp]_\sim))) \cap (\gamma_{\mathbb{N}^\nu}(q'^\sharp)) \\ ([q^\sharp]_\sim, \lambda, [q'^\sharp]_\sim) \in A^\sharp \\ q \xrightarrow{\lambda}_2 q' \end{cases} \right\} \end{cases}$$

We give in Figure 4 the definition of both an abstract counterpart $\mathbb{F}^\sharp$ of $\mathbb{F}$ and an accelerator of convergence $\mathbb{G}^\sharp$. Intuitively, $\mathbb{G}^\sharp$ merges the states of the abstract transition system, as soon as we are unable to prove that no infinite derivation can pass by a state they represent. $\mathbb{G}^\sharp$ uses a generic primitive relation, denoted by *finite* $\in \wp(\mathcal{T}^\sharp \times \mathbb{P})$, such that for all $t^\sharp = ((\mathbb{P}_{/\sim}, A^\sharp), f)$ in $\mathcal{T}^\sharp$, for all $q_0^\sharp$ in $\mathbb{P}$, if $(t^\sharp, q_0^\sharp)$ in *finite* then all derivations in transition system $\gamma_{\mathcal{T}}(t^\sharp)$ passing by a configuration c_0 in $\gamma(f([q_0^\sharp]_\sim)) \cap \gamma_{\mathbb{N}^\nu}(q_0{}^\sharp)$ are finite. $\mathbb{F}^\sharp$ and $\mathbb{G}^\sharp$ satisfy the soundness property: for all $t^\sharp$ in $\mathcal{T}^\sharp$, $[\mathbb{F} \circ \gamma_{\mathcal{T}}](t^\sharp) \sqsubseteq_{\mathcal{T}} [\gamma_{\mathcal{T}} \circ \mathbb{F}^\sharp \circ \mathbb{G}^\sharp](t^\sharp)$. Furthermore, the sequence $[\mathbb{F}^\sharp \circ \mathbb{G}^\sharp]^n((\mathbb{P}, \emptyset), \emptyset)$ is ultimately stationary and, thanks to Theorem 4.1.1.0.2 in [6], its limit $l^\sharp$ satisfies $lfp_{\perp_{\mathcal{T}}} \mathbb{F} \sqsubseteq \gamma_{\mathcal{T}}(l^\sharp)$.

Theorem 7.1 *Let $((\mathbb{P}_{/\sim}, A), f) = l^\sharp$. For all q in $\mathbb{P}$, for all C in $\gamma(f([q^\sharp]_\sim)) \cap (\gamma_{\mathbb{N}^\nu}(q^\sharp))$, there is no infinite computation sequence in the system $\mathcal{S}$ which passes by the state C if $(l^\sharp,\ q) \in$ finite.*

Our last task is to instantiate $\mathbb{P}$ and the primitive *finite*. A good choice for $\mathbb{P}$ consists in partitioning the set $\mathbb{N}^{\mathcal{V}}$ in accordance to the values of the variables of $\mathcal{V}$ which have a bounded behavior. For that purpose, we consider $v \subseteq V$ and a family (M_x) in $\mathbb{N}^v$, such that $\forall f \in \alpha_{\mathbb{N}^\nu}(Coll(\mathcal{S}))$, $f(x) \in [|0; M_x|]$. We take $\mathbb{P} = \mathbb{N}^{\mathcal{V}}{}_{/\sim}$ where $\sim$ is defined by: $g \sim h$ if and only if $(\forall x \in v,\ g(x) \in [|0; M_x|]$ *and* $g(x) = h(x))$ *or* $(\exists x_1, x_2 \in v, g(x_1) > 1$ *and* $h(x_2) > 1)$. Both v and (M_x) are given by the analysis presented in Section 5. The primitive *finite* is given by the following algorithm. Let $(t^\sharp, f) \in \mathcal{T}^\sharp$ and $q_0^\sharp \in \mathbb{P}$, we introduce *Available* $\subseteq \Sigma$ defined by $\lambda \in$ *Available* if and only if there is a derivation in the transition system $t^\sharp$ stemming from $q_0^\sharp$ and containing a transition labelled with λ. The following soundness proposition is valid:

Proposition 7.2 *$\forall \lambda \in \Sigma$, if there is a derivation in the transition system $\gamma_{\mathcal{T}}(t^\sharp, f)$ which stems from a configuration in $\gamma(f([q_0^\sharp]_\sim)) \cap (\gamma_{\mathbb{N}^\nu}(q_0^\sharp))$ and which contains a transition labelled with λ, then $\lambda \in$ Available.*

We denote by $\mapsto_2$ the subset of $\longrightarrow_2$ which only contains the computation steps labelled with an elements of *Available*. Following [16], we try to build a well-founded set in which we will interpret $(\mathcal{C}, \mapsto_2)$ via a morphism. Success in doing this will prove that any derivation in the system $\gamma_{\mathcal{T}}(t^\sharp, f)$ which passes by a configuration in $\gamma(f([q_0^\sharp]_\sim)) \cap (\gamma_{\mathbb{N}^\nu}(q_0^\sharp))$ is finite.

$$\mathbb{F}^\sharp_{\mathcal{P}}((\mathbb{P}_{/\sim}, A), f) = ((\mathbb{P}_{/\sim}, A'), f') \text{ where}$$

$$\begin{cases} f'(x) = f(x)\nabla h(x) \\ h(x) = \bigsqcup \left\{ (c^\sharp, v^\sharp \sqcap_{\mathcal{N}_\mathcal{V}} \alpha_{\mathcal{N}_\mathcal{V}}(u')) \;\middle|\; \begin{array}{l} u' \in x,\ v^\sharp \sqcap_{\mathcal{N}_\mathcal{V}} \alpha_{\mathcal{N}_\mathcal{V}}(u') \neq \bot_{\mathcal{N}_\mathcal{V}}, \\ \left\{ \begin{array}{l} (c^\sharp, v^\sharp) = \alpha(\{(\varepsilon, C) \mid C \in \mathcal{C}_0(\mathcal{S})\}) \\ \text{or } \exists u \in \mathbb{P},\ \exists \lambda \in \Sigma \text{ such that} \\ \quad f([u]_\sim) \sqcap (\top_{\text{com}}, \alpha_{\mathcal{N}_\mathcal{V}}(u)) \overset{\lambda}{\rightsquigarrow} (c^\sharp, v^\sharp) \end{array} \right. \end{array} \right\} \\ A' = A \cup \left\{ ([u]_\sim, \lambda, [u']_\sim) \;\middle|\; \begin{array}{l} \exists (c^\sharp, v^\sharp) \in \mathcal{C}^\sharp, \\ f([u]_\sim) \sqcap (\top_{\text{com}}, \alpha_{\mathcal{N}_\mathcal{V}}(u)) \overset{\lambda}{\rightsquigarrow} (c^\sharp, v^\sharp) \\ v^\sharp \sqcap_{\mathcal{N}_\mathcal{V}} \alpha_{\mathcal{N}_\mathcal{V}}(u') \neq \bot_{\mathcal{N}_\mathcal{V}} \end{array} \right\} \end{cases}$$

(a) counterpart function

$$\mathbb{G}^\sharp_{\mathcal{P}}((\mathbb{P}_{/\sim}, A), f) = ((\mathbb{P}_{/\sim'}, A'), f') \text{ where}$$

$$\begin{cases} a \sim' b \iff a \sim b \text{ or } \{(((\mathbb{P}_{/\sim}, A), f), a); (((\mathbb{P}_{/\sim}, A), f), b)\} \cap \mathit{finite} = \emptyset \\ f'([a]_{\sim'}) = \bigsqcup\limits_{x \in [a]_{\sim'}} f([x]_\sim) \\ A' = \{([a_0]_{\sim'}, \lambda, [a]_{\sim'}) \mid ([a_0]_\sim, \lambda, [a]_\sim) \in A\} \end{cases}$$

(b) quotient function

Fig. 4. Abstract trace semantics

We introduce the relation $\curvearrowright$ between the subprocesses of $\mathcal{S}$ such that $\forall p,\ q \in \Pi,\ p \curvearrowright q$ if and only if at least one of the following conditions is satisfied:

- p has the particular form $x!^j[]Q$, $q \in \bigcup \mathit{Agent}(Q)$, and there exists $i \in \mathit{Lbl}$ such that $(i, j) \in \mathit{Available}$;
- p has the particular form $x?^i[]P$, $q \in \bigcup \mathit{Agent}(P)$, and there exists $j \in \mathit{Lbl}$ such that $(i, j) \in \mathit{Available}$;
- p has the particular form $x!^j[]Q$, and there exists a process of the particular form $*y?^i[]P$ such that $q \in \bigcup \mathit{Agent}(P)$ and $(i, j) \in \mathit{Available}$.

We define the relation *finite* by: $((t^\sharp, f), q_0^\sharp) \in \mathit{finite} \iff (\Pi, \curvearrowright)$ is acyclic. Soundness hypotheses are satisfied since if $(\Pi, \curvearrowright)$ is acyclic, then $(\Pi, \curvearrowright^+)$ is a well founded order and the multiset extension [16] of $\curvearrowright^+$ gives a well founded order $\curvearrowright^+_{Mul}$ on $\mathbb{N}^\Pi$. The function $\Phi : (\mathcal{C}, \mapsto_2) \to (\mathbb{N}^\Pi, \curvearrowright^+_{Mul})$ defined by $\Phi(C)(p) = \mathit{Card}(\{(P, \mathit{id}, E) \in C \mid P = p\})$ is then a morphism.

Example 7.3 We propose to analyze the following mobile system:

$$\begin{aligned} \mathcal{S} \;:=\; & (\nu\ \text{push})(\nu\ \text{pop}) \\ & ((*\text{push}?^1[](\text{pop}!^2[] \mid \text{push}!^3[])) \mid *\,\text{pop}?^4[] \mid *\,\text{push}?^5[] \mid \text{push}!^6[]) \end{aligned}$$

This system describes the behavior of a stack. The height of the stack is symbolized by the number of processes $\text{pop}!^2[]$ that occurs at the top level.

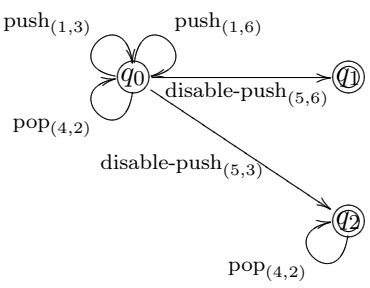
Push (resp. pop) operations are symbolized by communication with the resource $*\text{push}?^1[]$ (resp. $*\text{pop}?^4[]$). Finally, a communication with the resource $*\text{push}?^5[]$ symbolizes that further push operations are no longer allowed.

Occurrence counting analysis gives the following properties:

$$\begin{cases} \pi(1) = 1,\ \pi(2) \in [|0; +\infty|[,\ \pi(3) \in [|0; 1|], \\ \pi(4) = 1,\ \pi(5) = 1,\ \pi(6) \in [|0; 1|], \\ \underline{\pi}(1,6) \in [|0; 1|],\ \underline{\pi}(5,3) \in [|0; 1|],\ \underline{\pi}(5,6) \in [|0; 1|], \\ \underline{\pi}(1,3) \in [|0; \infty|[,\ \underline{\pi}(4,2) \in [|0; \infty|[. \end{cases}$$

We denote by $v \subseteq \mathcal{V}$ the set of variables $\{\text{push}!^3[]; \text{push}!^6[]; (1,6); (5,3); (5,6)\}$ and we take $\mathbb{P} = \mathbb{N}^{\mathcal{V}}/_{\sim}$ where $g \sim h$ if and only if $(\forall x \in v,\ g(x) \in [|0; 1|]$ *and* $g(x) = h(x))$ *or* $(\exists x_1, x_2 \in v,\ g(x_1) > 1$ *and* $h(x_2) > 1)$.

Our deadlock analysis then gives the abstract transition system $(t^\sharp, f)$ where $t^\sharp$ is given as follows:



where

$$q_0 = \left\{ f \;\middle|\; \begin{array}{l} f(\text{push}!^6[]) + f(\text{push}!^3[]) = 1 \\ f(\text{push}!^3[]) = f((1,6)) \in [|0; 1|] \\ f(x) = 0,\ \forall x \in \{(5,3); (5,6)\} \end{array} \right\}$$

$$q_1 = \left\{ f \;\middle|\; \begin{array}{l} f((5,6)) = 1 \\ f(x) = 0,\ \forall x \in v \setminus \{(5,6)\} \end{array} \right\}$$

$$q_2 = \left\{ f \;\middle|\; \begin{array}{l} f((1,6)) = f((5,3)) = 1 \\ f(x) = 0,\ \forall x \in v \setminus \{(1,6); (5,3)\} \end{array} \right\}$$

By applying our primitive relation *finite*, we prove that no infinite computation sequence can pass by a configuration in $\gamma_{\mathbb{N}^\nu}(q_1) \cup \gamma_{\mathbb{N}^\nu}(q_2)$, which means that executions of our system are bound to terminate as soon as a communication $(5,3)$ or a communication $(5,6)$ is performed. □

8 Conclusion

We have designed a powerful framework to prove properties on the potential behavior of a mobile system. Our analysis allows to detect, in polynomial time on the number of sub-processes of the mobile system, mutual exclusion and non-exhaustion of resources. Our analysis has succeeded in analyzing very quickly a few nested concrete systems, featuring unbounded communication topologies, with the expected level of accuracy. Deadlock detection is a much more difficult problem and our proposed approach is likely to require exponential time. However it gives interesting results on unbounded systems, which are out of reach of model checking methods. We are likely to refine our initial partitioning in order to get a quicker analysis.

This framework is likely to be enriched by using [14] in order to detect non-uniform confidentiality properties between processes, and to allow the analysis of mobile systems in hostile context. This will lead to a very powerfull modular

analysis for mobile systems. To scale up, for large nested mobile systems, the analysis must be more approximate to ensure short execution time.

Acknowledgement

We deeply thank anonymous referees for their significant comments on an early version. We wish also thank Patrick and Radhia Cousot, Arnaud Venet, Jorge Pinto and Antoine Miné, for their comments and discussions.

References

[1] G. Berry and G. Boudol. The chemical abstract machine. *Theoretical Computer Science*, 96:217–248, 1992.

[2] C. Bodei, P. Degano, F. Nielson, and H.R Nielson. Control flow analysis for the π-calculus. In *Proceedings of CONCUR'98*, Lecture Notes in Computer Science. Springer-Verlag, 1998.

[3] F. Bourdoncle. Abstract interpretation by dynamic partitioning. *Journal of Functional Programming*, 2(4), 1992.

[4] L. Cardelli and A. D. Gordon. Mobile ambients. In *Foundations of Software Science and Computational Structures*, volume 1378 of *Lecture Notes in Computer Science*, pages 140–155. Springer, 1998.

[5] C.K. Chiu and J.H.M. Lee. Interval linear constraint solving using the preconditioned interval gaus-seidel method. In *Proceedings of the Twelfth International Conference on Logic Programming*, Logic Programming, pages 17–32. The MIT Press, 1995.

[6] P. Cousot. *Méthodes itératives de construction et d'approximation de points fixes d'opérateurs monotones sur un treillis, analyse sémantique des programmes.* PhD thesis, Université Scientifique et Médicale de Grenoble, 1978.

[7] P. Cousot. Semantic foundations of program analysis. In S.S. Muchnick and N.D. Jones, editors, *Program Flow Analysis: Theory and Applications*, chapter 10, pages 303–342. Prentice-Hall, Inc., Englewood Cliffs, 1981.

[8] P. Cousot and R. Cousot. Abstract interpretation: a unified lattice model for static analysis of programs by construction or approximation of fixpoints. In *Conference Record of the Fourth ACM Symposium on Principles of Programming Languages*, pages 238–252, Los Angeles, California, U.S.A., 1977.

[9] P. Cousot and R. Cousot. Systematic design of program analysis frameworks. In *Proceedings of the Sixth Conference on Principles of Programming Languages POPL'79*. ACM Press, 1979.

[10] P. Cousot and R. Cousot. Abstract interpretation frameworks. *Journal of logic and computation*, 2(4):511–547, August 1992.

[11] P. Cousot and R. Cousot. Comparing the Galois connection and widening-narrowing approaches to abstract interpretation. In *Programming Language Implementation and Logic Programming, Proceedings of the Fourth International Symposium, PLILP'92*, volume 631 of *Lecture Notes in Computer Science*, pages 269–295. Springer-Verlag, 1992.

[12] P. Cousot and N. Halbwachs. Automatic discovery of linear restraints among variables of a program. In *Proceedings of the Fifth Conference on Principles of Programming Languages*. ACM Press, 1978.

[13] J. Feret. Conception de *π-sa* : un analyseur statique générique pour le π-calcul. Mémoire de dea, SPP, september 1999. Electronically available at `http://www.di.ens.fr/~feret/dea/dea.ps`.

[14] J. Feret. Confidentiality analysis of mobile systems. In *Seventh International Static Analysis Symposium (SAS'00)*, volume 1824 of *LNCS*. Springer-Verlag, 2000.

[15] R. R. Hansen, J. G. Jensen, F. Nielson, and H. R. Nielson. Abstract interpretation of mobile ambients. In *Proc. SAS'99*, number 1694 in Lecture Notes in Computer Science, pages 134–148. Springer-Verlag, 1999.

[16] Jean-Pierre Jouannaud. Rewrite proofs and computations. In Helmut Schwichtenberg, editor, *Proof and Computation*, volume 139 of *Computer and Systems Sciences*, pages 173–218. Springer-Verlag, 1995.

[17] M. Karr. Affine relationships among variables of a program. *Acta Informatica*, pages 133–151, 1976.

[18] R. Milner. The polyadic π-calculus: a tutorial. In *Proceedings of the International Summer School on Logic and Algebra of Specification*. Springer Verlag, 1991.

[19] R. Milner, J. Parrow, and D. Walker. A calculus of mobile processes. *Information and Computation*, 100:1–77, 1992.

[20] H. Riis Nielson and F. Nielson. Shape analysis for mobile ambients. In *Proc. POPL'00*, pages 142–154. ACM Press, 2000.

[21] D. N. Turner. *The Polymorphic Pi-Calculus: Theory and Implementation*. PhD thesis, Edinburgh University, 1995.

[22] A. Venet. Abstract interpretation of the π-calculus. In *Proc. of the Fifth LOMAPS Workshop on Analysis and Verification of High-Level Concurrent Languages*, volume 1192 of *Lecture Notes in Computer Science*, pages 51–75. Springer-Verlag, 1996.

[23] A. Venet. Automatic determination of communication topologies in mobile systems. In *Proceedings of the Fifth International Static Analysis Symposium SAS'98*, volume 1503 of *Lecture Notes in Computer Science*, pages 152–167. Springer-Verlag, 1998.