Abstract Interpretation-Based Static Analysis of Mobile Ambients*

Jérôme Feret

jerome.feret@ens.fr
Département d'Informatique de l'École Normale Supérieure
ENS-DI, 45, rue d'Ulm, 75230 PARIS cedex 5, FRANCE
http://www.di.ens.fr/~feret

Abstract. We use Abstract Interpretation to automatically prove safety properties of mobile ambients with name communications. We introduce a non-standard semantics in order to distinguish different recursive instances of agents. This allows us to specify explicitly both the link between agents and the ambient names they have declared, and the link between agents and the ambients they have activated.
Then we derive from this non-standard semantics an abstract semantics which focuses on interactions between agents. This abstract semantics describes non uniformly which agents can be launched in which ambients and which ambient names can be communicated to which agents. Such a description is required to prove security properties such as non-interference or confinement for instance.

1 Introduction

The development of large scale communicating distributed systems imposes the design of both good models for mobile computation and well-fitted methods for analyzing properties of mobile systems. Mobility has quite a broad meaning. In the π-calculus [16], mobility is implicitly described by name passing: agents communicate channel names. This dynamically changes the communication topology between agents. In mobile ambients [4], mobility is explicit: ambients are bounded places and agents give them the capability to move inside other ambients taking their content with them. The connections between these two models are not well known yet. Anyway, we know that the asynchronous π-calculus can be encoded into the ambient calculus [2, p:5]. Security properties of mobile systems are usually described either by simulation relations (such as non-interference [15]) or by some constrains over their control flow (such as confinement or level of secrecy [12]). Nevertheless, existing analyses often describe a set of configurations which lead to a leak of security, then prove that such configurations can never occur. This second step requires both control flow and system shape approximation.

* This work was supported by the RTD project IST-1999-20527 "DAEDALUS" of the European FP5 programme.

In previous works, we have proposed a control flow analysis [9], and an occurrence counting analysis [10] for π-calculus specified mobile systems. We propose to extend our framework with explicit mobility. In this paper we restrict our study to the analysis of the control flow of a mobile ambient, just considering name communications (instead of capability path communications). Our analysis consists in tagging each agent by an unambiguous marker which encodes the history of the replications that have led to its creation. Then, we label all the objects (ambients and ambient names) with the marker of the agent which has created them. We abstract for each agent both the set of the ambients it can be spawned in, and the set of the ambient names which can be communicated to it. We capture algebraic properties on the involved markers too.

We claim that distinguishing the objects created by the recursive instances of an agent is crucial when analyzing mobile systems. The main difference between a mobile system and a system written in CCS is that recursive instances of agents can interfere via the objects they have themselves declared. Nevertheless, this aspect of mobility is ignored by most of the analyses proposed in literature, which either identify recursive instances of agents [17, 11, 18] or prevent the recursive declaration of ambient names [3]. Ambient groups [1] use dependent types to prevent names of a fresh group from ever being received outside the initial scope of this group. In addition, our analysis also handles the algebraic properties of the markers. This allows us to describe the interaction between recursive instances of agents, whereas [1] can only prove that ambient names are confined inside the recursive instance of the agent which has created it.

The semantics for mobile ambients is given in Sect. 2. The non-standard semantics is introduced in Sect. 3. The abstract interpretation framework is reminded in Sect. 4. Eventually we design a generic abstraction of the interactions between agents in Sect. 5 and give three examples of analyses in Sect. 6.

Acknowledgments. We deeply thank anonymous referees for their significant comments on an early version of this paper. We wish also to thank Patrick and Radhia Cousot, Arnaud Venet, Antoine Miné and Francesco Logozzo, for their comments and discussions.

2 Mobile Ambients

Mobile ambients [4] are a model of mobile computation. It describes a set of *agents* which are distributed throughout hierarchically organized domains called *ambients*. Agents interact inside ambients which makes the ambients move, taking with them their content. We consider a lazy version of mobile ambients in that replications are performed only when necessary. For the sake of simplicity, we restrict ourselves to name communications: just names and not capability paths can be communicated. Let $\mathcal{N}$ be a countable set of ambient names and *Lbl* a countable set of labels. We give in Fig. 1 the standard semantics of the mobile ambients. We locate each syntactic component of the system by placing distinct labels of *Lbl*.

$$
\begin{array}{llllll}
n \in \mathcal{N} & \text{(ambient name)} & M ::= & in^l\ n.P & \text{(can enter } n) \\
l \in Lbl & \text{(label)} & \mid & out^l\ n.P & \text{(can exit } n) \\
 & & \mid & open^l\ n.P & \text{(can open } n) \\
 & & \mid & !open^l\ n.P & \text{(can duplicate itself before opening } n)
\end{array}
$$

$$
\begin{array}{lll}
P, Q ::= & (\nu\ n)P & \text{(restriction)} \\
\mid & \mathbf{0} & \text{(inactivity)} \\
\mid & P \mid Q & \text{(composition)} \\
\mid & n^l[P] & \text{(ambient)} \\
\mid & M & \text{(capability action)} \\
\mid & io & \text{(input/output action)}
\end{array}
\qquad
\begin{array}{lll}
io ::= & (n)^l.P & \text{(input action)} \\
\mid & !(n)^l.P & \text{(input action with replication)} \\
\mid & \langle n \rangle^l & \text{(async output action)}
\end{array}
$$

Input action and restriction are the only name binders, in $(n)^l.P$, $!(n)^l.P$ and $(\nu\ n)P$, occurrences n in P are bound. Usual rules about scopes, substitution and α-conversion apply. We denote by $\mathcal{FN}(P)$ (resp. $\mathcal{BN}(P)$) the set of the free (resp. bound) names of P.

(a) Syntax.

$$
\begin{array}{rcll}
P & \equiv & Q & \text{if } P \sim_\alpha Q \quad (\alpha\text{-conversion}) \\
P \mid Q & \equiv & Q \mid P & \text{(Commutativity)} \\
(P \mid Q) \mid R & \equiv & P \mid (Q \mid R) & \text{(Associativity)} \\
P \mid \mathbf{0} & \equiv & P & \text{(Zero par)} \\
(\nu\ n)\mathbf{0} & \equiv & \mathbf{0} & \text{(Zero Res)} \\
(\nu\ n)(\nu\ m)P & \equiv & (\nu\ m)(\nu\ n)P & \text{(Swapping)} \\
(\nu\ n)(P \mid Q) & \equiv & P \mid ((\nu\ n)Q) & \text{if } n \notin \mathcal{FN}(P) \quad \text{(Extrusion Par)} \\
(\nu\ n)(m^l[P]) & \equiv & m^l[(\nu\ n)P] & \text{if } n \neq m \quad \text{(Extrusion Amb)}
\end{array}
$$

(b) Congruence relation.

$$
\begin{array}{rcl}
n^i[in^k\ m.P \mid Q] \mid m^j[R] & \xrightarrow{in(i,j,k)} & m^j[n^i[P \mid Q] \mid R] \\
m^i[n^j[out^k\ m.P \mid Q] \mid R] & \xrightarrow{out(i,j,k)} & n^j[P \mid Q] \mid m^i[R] \\
open^i\ n.P \mid n^j[Q] & \xrightarrow{open(i,j)} & P \mid Q \\
!open^i\ n.P \mid n^j[Q] & \xrightarrow{open(i,j)} & P \mid Q \mid !open^i\ n.P \\
(n)^i.P \mid \langle m \rangle^j & \xrightarrow{com(i,j)} & P[n \leftarrow m] \\
!(n)^i.P \mid \langle m \rangle^j & \xrightarrow{com(i,j)} & P[n \leftarrow m] \mid !(n)^i.P
\end{array}
$$

$$
\frac{P \xrightarrow{\lambda} Q}{n^i[P] \xrightarrow{\lambda} n^i[Q]}
\qquad
\frac{P \xrightarrow{\lambda} Q}{(\nu\ n)P \xrightarrow{\lambda} (\nu\ n)Q}
\qquad
\frac{P \xrightarrow{\lambda} Q}{P \mid R \xrightarrow{\lambda} Q \mid R}
$$

$$
\frac{P' \equiv P,\ P \xrightarrow{\lambda} Q,\ Q \equiv Q'}{P' \xrightarrow{\lambda} Q'}
$$

(c) Reduction relation.

Fig. 1. Standard semantics

Example 1. We model a system $\mathcal{S}$ which describes a *client-server* protocol. To make things clearer, public (or global) names are written in roman, all the other names are written in *italic* and we abstract away many computational aspects. A resource creates recursively an unbounded number of clients. Each client is represented by a packet $p[]$ which contains an ambient named "request". This ambient contains the client's query $\langle q \rangle$. At first, each packet enters the "server" ambient and then activates a pilot ambient "duplicate" which communicates the packet name to the server. This communication creates a recursive instance of an ambient named "instance" which will process the packet. The "instance" ambient enters the packet, reads the request and sends it back inside an ambient named "answer". At last, the packet exits the "server" ambient.

$\mathcal{S}$ is defined as follows:

$$
\begin{array}{l}
\nu\mathbf{Pub} := (\nu\ \mathrm{request})(\nu\ \mathrm{make})(\nu\ \mathrm{server})(\nu\ \mathrm{duplicate})(\nu\ \mathrm{instance})(\nu\ \mathrm{answer}),\\
\mathbf{C}_1 := \mathrm{request}^{13}[\langle q\rangle^{14}],\ \mathbf{C}_2 := \mathit{open}^{15}\mathrm{instance},\\
\mathbf{C}_3 := \mathit{in}^{16}\mathrm{server}.\mathrm{duplicate}^{17}[\mathit{out}^{18}p.\langle p\rangle^{19}],\\
\mathbf{C} := (\nu\ q)(\nu\ p)p^{12}[\mathbf{C}_1 \mid \mathbf{C}_2 \mid \mathbf{C}_3] \mid \langle \mathrm{make}\rangle^{20},\\
\mathbf{I}_1 := \mathrm{answer}^{8}[\langle \mathit{rep}\rangle^{9}],\ \mathbf{I}_2 := \mathit{out}^{10}\mathrm{server},\ \mathbf{I} := \mathit{in}^{5}k.\mathit{open}^{6}\mathrm{request}.(\mathit{rep})^{7}(\mathbf{I}_1 \mid \mathbf{I}_2),\\
\mathbf{S}_1 := !\mathit{open}^{2}\mathrm{duplicate},\ \mathbf{S}_2 := !(k)^{3}.\mathrm{instance}^{4}[\mathbf{I}],\ \mathbf{S} := \mathrm{server}^{1}[\mathbf{S}_1 \mid \mathbf{S}_2],\\
\mathcal{S} := (\nu\mathbf{Pub})(\mathbf{S} \mid !(x)^{11}.C \mid \langle \mathrm{make}\rangle^{21}).
\end{array}
$$

The following computation sequence describes the behaviour of the system:

$$
\begin{array}{l}
\quad (\nu\mathbf{Pub})(\mathbf{S} \mid !(\underline{x})^{11}.C \mid \langle \underline{\mathrm{make}}\rangle^{21})\\
\longrightarrow (\nu\mathbf{Pub})\left(\begin{array}{l} !(x)^{11}.C \mid \langle \mathrm{make}\rangle^{20} \mid \underline{\mathrm{server}}^{1}[\mathbf{S}_1 \mid \mathbf{S}_2] \mid \\ (\nu\ q_1)(\nu\ p_1)p_1{}^{12}\left[\begin{array}{l}\mathrm{request}^{13}[\langle q_1\rangle^{14}] \mid \mathbf{C}_2 \mid \\ \mathit{in}^{16}\underline{\mathrm{server}}.\mathrm{duplicate}^{17}[\mathit{out}^{18}p_1.\langle p_1\rangle^{19}]\end{array}\right]\end{array}\right)\\
\longrightarrow (\nu\mathbf{Pub})(\nu\ q_1)(\nu\ p_1)\\
\quad \left(!(x)^{11}.C \mid \langle \mathrm{make}\rangle^{20} \mid \mathrm{server}^{1}\left[\mathbf{S}_1 \mid \mathbf{S}_2 \mid \underline{p_1}{}^{12}\left[\begin{array}{l}\mathrm{request}^{13}[\langle q_1\rangle^{14}] \mid \mathbf{C}_2 \mid \\ \mathrm{duplicate}^{17}[\mathit{out}^{18}\underline{p_1}.\langle p_1\rangle^{19}]\end{array}\right]\right]\right)\\
\longrightarrow (\nu\mathbf{Pub})(\nu\ q_1)(\nu\ p_1)\\
\quad \left(!(x)^{11}.C \mid \langle \mathrm{make}\rangle^{20} \mid \mathrm{server}^{1}\left[\begin{array}{l}!\mathit{open}^{2}\underline{\mathrm{duplicate}} \mid \mathbf{S}_2 \mid \underline{\mathrm{duplicate}}^{17}[\langle p_1\rangle^{19}] \mid \\ p_1{}^{12}[\mathrm{request}^{13}[\langle q_1\rangle^{14}] \mid \mathbf{C}_2]\end{array}\right]\right)\\
\longrightarrow (\nu\mathbf{Pub})(\nu\ q_1)(\nu\ p_1)\\
\quad \left(!(x)^{11}.C \mid \langle \mathrm{make}\rangle^{20} \mid \mathrm{server}^{1}\left[\begin{array}{l}\mathbf{S}_1 \mid !(\underline{k})^{3}.\mathrm{instance}^{4}[\mathbf{I}] \mid \langle \underline{p_1}\rangle^{19} \mid \\ p_1{}^{12}[\mathrm{request}^{13}[\langle q_1\rangle^{14}] \mid \mathbf{C}_2]\end{array}\right]\right)\\
\longrightarrow (\nu\mathbf{Pub})(\nu\ q_1)(\nu\ p_1)\\
\quad \left(!(x)^{11}.C \mid \langle \mathrm{make}\rangle^{20} \mid \mathrm{server}^{1}\left[\begin{array}{l}\mathbf{S}_1 \mid \mathbf{S}_2 \mid \underline{p_1}{}^{12}[\mathrm{request}^{13}[\langle q_1\rangle^{14}] \mid \mathbf{C}_2] \mid \\ \mathrm{instance}^{4}[\mathit{in}^{5}\underline{p_1}.\mathit{open}^{6}\mathrm{request}.(\mathit{rep})^{7}(\mathrm{I}_1 \mid \mathrm{I}_2)]\end{array}\right]\right)\\
\longrightarrow (\nu\mathbf{Pub})(\nu\ q_1)(\nu\ p_1)\\
\quad \left(!(x)^{11}.C \mid \langle \mathrm{make}\rangle^{20} \mid \mathrm{server}^{1}\left[\begin{array}{l}\mathbf{S}_1 \mid \mathbf{S}_2 \mid \\ p_1{}^{12}\left[\begin{array}{l}\mathrm{request}^{13}[\langle q_1\rangle^{14}] \mid \mathit{open}^{15}\underline{\mathrm{instance}} \mid \\ \underline{\mathrm{instance}}^{4}[\mathit{open}^{6}\mathrm{request}.(\mathit{rep})^{7}(\mathrm{I}_1 \mid \mathrm{I}_2)]\end{array}\right]\end{array}\right]\right)\\
\longrightarrow (\nu\mathbf{Pub})(\nu\ q_1)(\nu\ p_1)\\
\quad \left(!(x)^{11}.C \mid \langle \mathrm{make}\rangle^{20} \mid \mathrm{server}^{1}\left[\mathbf{S}_1 \mid \mathbf{S}_2 \mid p_1{}^{12}\left[\begin{array}{l}\underline{\mathrm{request}}^{13}[\langle \underline{q_1}\rangle^{14}] \mid \\ \mathit{open}^{6}\underline{\mathrm{request}}.(\underline{\mathit{rep}})^{7}(\mathrm{I}_1 \mid \mathrm{I}_2)\end{array}\right]\right]\right)
\end{array}
$$

$$\begin{array}{l}
\longrightarrow^{*}(\nu\mathbf{Pub})(\nu\ q_1)(\nu\ p_1)\\
\quad(!(x)^{11}.C \mid \langle\text{make}\rangle^{20}\mid \underline{\text{server}}^{1}[\mathbf{S}_1 \mid \mathbf{S}_2 \mid {p_1}^{12}[\text{answer}^{8}[\langle q_1\rangle^{9}] \mid out^{10}\underline{\text{server}}]])\\
\longrightarrow\ (\nu\mathbf{Pub})(\nu\ q_1)(\nu\ p_1)\\
\quad(!(x)^{11}.C \mid \langle\text{make}\rangle^{20}\mid \text{server}^{1}[\mathbf{S}_1 \mid \mathbf{S}_2] \mid {p_1}^{12}[\text{answer}^{8}[\langle q_1\rangle^{9}]])\\
\longrightarrow^{*}(\nu\mathbf{Pub})(\nu\ q_1)(\nu\ p_1)(\nu\ q_2)(\nu\ p_2)(!(x)^{11}.C \mid \langle\text{make}\rangle^{20}\mid \text{server}^{1}[\mathbf{S}_1 \mid \mathbf{S}_2]\mid\\
\qquad\qquad\qquad {p_1}^{12}[\text{answer}^{8}[\langle q_1\rangle^{9}]] \mid {p_2}^{12}[\text{answer}^{8}[\langle q_2\rangle^{9}]]) \qquad \square
\end{array}$$

3 Non-standard Semantics

The non-standard semantics is a refined one with explicit substitution. It restores the link between the recursive instances of agents and the objects they have created (i.e. the names they have declared and the ambients they have activated). Following $D\pi$ [13] style, we describe a mobile system with a set of agents tagged with a location marker. Furthermore, the embedding structure of the ambients imposes a description of the hierarchical tree of the administrative domains (or ambients). This is given by a set of activated ambients[1] (seen as locations) tagged with location markers specifying the surrounding ambient. We assume that a system is run inside a top level ambient which has no location. The link between agents and the ambient names they have declared is made explicit by tagging each agent by an unambiguous history marker allocated at its creation. Then, each new ambient name is tagged with the history marker of the agent which has declared it. Thus, we restore the link between agents and the ambients they have activated by tagging each activated ambient with the history marker of the agent which has activated it.

Let $\mathcal{S}$ be a closed mobile system in the ambient calculus. We assume without any loss of generality that two name binders ($\nu\ n$ or (n)) are never used to bind the same ambient name. *History markers* are binary trees the node of which are labeled with elements of Lbl^2 and the leaves of which are not labeled. The tree having a node labeled λ, a left sibling t_1 and a right one t_2 is denoted by $N(\lambda, t_1, t_2)$. We denote by Id the set of the history markers. *Ambient names* are described by a pair (n, id) where n specifies which action $(\nu\ n)$ has created it while id is the history marker of the agents which has declared this name. *Activated ambients* are identified by a pair (i, id) where i is the label of the ambient constructor which has activated the ambient while id is the history marker of its activator[2]. The top level ambient is represented by the pair $(\text{top}, \varepsilon)$ (we assume that $\text{top} \in Lbl$ has not been used for labeling $\mathcal{S}$ yet). *Location markers* are pairs (i, id), too. A location marker refers to the ambient where a process is spawned.

A *non-standard configuration* [20, 9] is a set of thread instances, where a thread instance is a tuple composed by a syntactic component, a history marker, a location marker and an environment. The syntactic component is either a syntactic copy of an agent of $\mathcal{S}$ or an activated ambient denoted by $n^i[\bullet]$. The history marker is unambiguously allocated at the thread creation. The location

[1] Also called privileged ambients in [4].
[2] An ambient cannot be identified by its ambient name because two distinct activated ambients can have the same name [4, p:12].

marker indicates where the thread is run. The environment specifies the origin of the free syntactic ambient names of the syntactic component.

Example 2. We give here the non-standard configuration reached after completing two sessions of our server[3]:

$$\begin{cases}(\mathrm{server}^1[\bullet], \varepsilon, (\mathrm{top}, \varepsilon), \emptyset)\\ (\mathrm{answer}^8[\bullet], id'_0, (12, id_0), \emptyset)\\ (\mathrm{answer}^8[\bullet], id'_1, (12, id_1), \emptyset)\\ (p^{12}[\bullet], id_0, (\mathrm{top}, \varepsilon), [p \mapsto (p, id_0)])\\ (p^{12}[\bullet], id_1, (\mathrm{top}, \varepsilon), [p \mapsto (p, id_1)])\\ (!(x)^{11}.\mathbf{C}, \varepsilon, (\mathrm{top}, \varepsilon), \emptyset)\\ (\langle \mathrm{make}\rangle^{20}, id_1, (\mathrm{top}, \varepsilon), \emptyset)\\ (\mathbf{S}_1, \varepsilon, (1, \varepsilon), \emptyset)\\ (\mathbf{S}_2, \varepsilon, (1, \varepsilon), \emptyset)\\ (\langle rep\rangle^9, id'_0, (8, id'_0), [rep \mapsto (q, id_0)])\\ (\langle rep\rangle^9, id'_1, (8, id'_1), [rep \mapsto (q, id_1)])\end{cases} \quad \text{where:} \quad \begin{cases} id_0 = N((11,21), \varepsilon, \varepsilon)\\ id_1 = N((11,20), \varepsilon, id_0)\\ id'_0 = N((3,19), \varepsilon, id_0)\\ id'_1 = N((3,19), \varepsilon, id_1)\end{cases}$$

The top five instances represent the hierachic structure of nested ambients, the others describe the agent distribution. Location markers allow in reconstructing the following ambient:

$$(\nu\, \overline{n})\ {}^{(\mathrm{top},\varepsilon)}\left[\begin{array}{l} !(x)^{(11,\varepsilon)}.\mathbf{C} \mid \langle \mathrm{make}\rangle^{(20,id_1)} \mid \mathrm{server}^{(1,\varepsilon)}[\mathbf{S}_1 \mid \mathbf{S}_2] \mid \\ (p, id_0)^{(12,id_0)}[\mathrm{answer}^{(8,id'_0)}[\langle (q, id_0)\rangle^{(9,id'_0)}]] \mid \\ (p, id_1)^{(12,id_1)}[\mathrm{answer}^{(8,id'_1)}[\langle (q, id_1)\rangle^{(9,id'_1)}]] \end{array}\right]$$

in where ambients, ambient names and agents are stamped with their own makers. Thanks to name markers, we avoid conflict between ambient names. So we can extrude their declaration inside the top level ambient. In this way, the shortcut $(\nu\overline{n})$ denotes the declaration of all the ambient names of the configuration. It appears explicitly that, in each packet, both the name of the packet and that contained in the "answer" ambient embedded in the packet have been declared by the same recursive instance of the resource $!(x).\mathbf{C}$. This means that the answer of a query is sent to the good client. □

The non-standard semantics is given in Fig. 3 by both an initial non-standard configuration and a reduction relation. Their definitions use the extraction function β defined in Fig. 2. Given a continuation P, an history marker id, a location marker loc and an environment E, $\beta(P, id, loc, E)$ gives the set of all the thread instances that must be spawned to simulate the computation of the process $E(P)$ identified with the marker id, in the ambient denoted by loc. It especially deals with new ambient name declaration and new ambient activation.

[3] We do not figure the origin of public names.

$$
\begin{aligned}
\beta(n^i[P], id, loc, E) &= \beta(P, id, (i, id), E) \cup \{(n^i[\bullet], id, loc, [n \mapsto E(n)])\} \\
\beta(P \mid Q, id, loc, E) &= \beta(P, id, loc, E) \cup \beta(Q, id, loc, E) \\
\beta((\nu\ n)P, id, loc, E) &= \beta(P, id, loc, (E[n \mapsto (n, id)])) \\
\beta(M, id, loc, E) &= \{(M, id, loc, E_{|\mathcal{FN}(M)})\} \\
\beta(io, id, loc, E) &= \{(io, id, loc, E_{|\mathcal{FN}(io)})\} \\
\beta(\mathbf{0}, id, loc, E) &= \emptyset
\end{aligned}
$$

Fig. 2. Extraction function.

We informally describe the non-standard semantics. For the sake of the brevity, we only detail the non-standard *in* migration rule. *in* migration rule involves two distinct ambients λ, μ and an agent ψ. They are respectively denoted by three configurations $(n^i[\bullet], id_1, loc_1, E_1)$, $(m^j[\bullet], id_2, loc_2, E_2)$ and $(in^k o.P, id_3, loc_3, E_3)$. The *in* migration rule is enabled if and only if the two ambients are located in the same ambient (this gives the constrain $loc_1 = loc_2$), the agent is located in the first ambient (this gives $loc_3 = (i, id_1)$) and the agent capability can interact with the name of the second ambient (this is encoded by the constrain $E_2(m) = E_3(o)$). The result of such a migration is that the first ambient moves inside the second one (its location is just replaced by (j, id_2)). All it content is taken with it (this does change neither their location markers, nor their environments), but the agent ψ is executed and its continuation is spawned inside the first ambient (ψ is replaced by $\beta(P, id_3, loc_3, E_{3|\mathcal{FN}(P)})$). The *out* migration is simulated in the same way. The ambient dissolution is a bit much complex since all the locations of the dissolved ambient content are changed. We shall notice that each time a resource is fetched a new history marker is deterministically allocated: it is given by $N((i, j), id_i, id_j)$ where i is the label of the resource, id_i is the history marker of the resource, j is the label of the thread which enforces the resource fetching and id_j is the history marker of this thread. We do not need a congruence relation because our set-based representation of configurations makes structural congruence rules useless and the use of history markers avoids conflicts between ambient names.

Standard and non-standard semantics are strongly bisimilar. The proof relies on that non-standard computations cannot yield conflicts between history markers. Moreover, in accordance to the following proposition, we can simplify the shape of the history markers without losing the consistency of our semantics.

$$
\phi_1 : \begin{cases} Id & \to (Lbl^2)^\star \\ N(a, b, c) & \mapsto a.\phi_1(c) \\ \varepsilon & \mapsto \varepsilon \end{cases}
\qquad
\phi_2 : \begin{cases} Id & \to Lbl^\star \\ N((i, j), b, c) & \mapsto j.\phi_2(c) \\ \varepsilon & \mapsto \varepsilon \end{cases}
$$

Proposition 1. *Let ϕ be ϕ_1 or ϕ_2 and $C_0 \longrightarrow \ldots \longrightarrow C_n$ be a non-standard computation sequence, where $C_0 = C_0(\mathcal{S})$. For all $i, j \in [|0, n|]$, $(p, id, loc, E) \in C_i$ and $(p', id', loc', E') \in C_j$, such that $\phi(id) = \phi(id')$ then $id = id'$.*

Such simplifications allow us to reduce the cost of our analysis, but also lead to a loss of accuracy, since they merge information related to distinct computation sequences of the system.

$$C_0(\mathcal{S}) = \beta(\mathcal{S}, \varepsilon, (\text{top}, \varepsilon), \emptyset).$$

(a) Initial configuration.

If C is a non-standard configuration,
if there are λ, μ, ψ in C, $(\lambda \neq \mu)$
with $\lambda = \left(n^i[\bullet], id_1, loc_1, E_1\right)$, $\mu = \left(m^j[\bullet], id_2, loc_2, E_2\right)$ and $\psi = \left(in^k o.P, id_3, loc_3, E_3\right)$,
such that $loc_1 = loc_2$, $loc_3 = (i, id_1)$ and $E_2(m) = E_3(o)$
then $C \overset{in(i,j,k)}{\longrightarrow} (C \setminus \{\lambda, \psi\}) \cup \left(n^i[\bullet], id_1, (j, id_2), E_1\right) \cup \beta\left(P, id_3, loc_3, E_{3|\mathcal{FN}(P)}\right)$.

If C is a non-standard configuration,
if there are λ, μ, ψ in C,
with $\lambda = \left(m^i[\bullet], id_1, loc_1, E_1\right)$, $\mu = \left(n^j[\bullet], id_2, loc_2, E_2\right)$ and $\psi = \left(out^k o.P, id_3, loc_3, E_3\right)$,
such that $loc_2 = (i, id_1)$, $loc_3 = (j, id_2)$ and $E_1(m) = E_3(o)$
then $C \overset{out(i,j,k)}{\longrightarrow} (C \setminus \{\mu, \psi\}) \cup \left(n^j[\bullet], id_2, loc_1, E_2\right) \cup \beta\left(P, id_3, loc_3, E_{3|\mathcal{FN}(P)}\right)$.

If C is a non-standard configuration,
if there are λ, μ in C, with $\lambda = \left(open^i m.P, id_1, loc_1, E_1\right)$ and $\mu = \left(n^j[\bullet], id_2, loc_2, E_2\right)$,
such that $loc_1 = loc_2$ and $E_1(m) = E_2(n)$.
then $C \overset{open(i,j)}{\longrightarrow} (C \setminus (\{\lambda, \mu\} \cup A)) \cup \beta\left(P, id_1, loc_1, E_{1|\mathcal{FN}(P)}\right) \cup A'$
where $\begin{cases} A = \{(a, id, loc, E) \in C \mid loc = (j, id_2)\} \\ A' = \{(a, id, loc_2, E) \mid (a, id, (j, id_2), E) \in C\}. \end{cases}$

If C is a non-standard configuration,
if there are λ, μ in C, with $\lambda = \left(!open^i m.P, id_1, loc_1, E_1\right)$ and $\mu = \left(n^j[\bullet], id_2, loc_2, E_2\right)$,
such that $loc_1 = loc_2$ and $E_1(m) = E_2(n)$,
then $C \overset{open(i,j)}{\longrightarrow} (C \setminus (\{\mu\} \cup A)) \cup \beta\left(P, N((i,j), id_1, id_2), loc_1, E_{1|\mathcal{FN}(P)}\right) \cup A'$
where $\begin{cases} A = \{(a, id, loc, E) \in C \mid loc = (j, id_2)\} \\ A' = \{(a, id, loc_2, E) \mid (a, id, (j, id_2)), E) \in C\}. \end{cases}$

(b) Move rules.

If C is a non-standard configuration,
if there are λ, μ in C, with $\lambda = ((n)^i.P, id_1, loc_1, E_1)$ and $\mu = (\langle m\rangle^j, id_2, loc_2.E_2)$,
such that $loc_1 = loc_2$,
then $C \overset{com(i,j)}{\longrightarrow} (C \setminus \{\lambda, \mu\}) \cup \beta(P, id_1, loc_1, E_1[n \mapsto E_2(m)]_{|\mathcal{FN}(P)})$.

If C is a non-standard configuration,
if there are λ, μ in C, with $\lambda = (!(n)^i.P, id_1, loc_1, E_1)$ and $\mu = (\langle m\rangle^j, id_2, loc_2.E_2)$,
such that $loc_1 = loc_2$,
then $C \overset{com(i,j)}{\longrightarrow} (C \setminus \{\mu\}) \cup \beta(P, N((i,j), id_1, id_2), loc_1, E_1[n \mapsto E_2(m)]_{|\mathcal{FN}(P)})$.

(c) Communication rules.

Fig. 3. Non-standard semantics.

4 Abstract Interpretation Framework

We denote by $\mathcal{C}$ the set of all possible non-standard configurations and by Σ the set of transition labels. We are actually interested in the set $\mathit{Coll}(\mathcal{S})$ of all the configurations a system may take during a finite sequence of computation steps. This is given by its collecting semantics [5] and can be expressed as the least fix point of a $\cup$-complete endomorphism $\mathbb{F}$ on the complete lattice $\wp(\Sigma^\star \times \mathcal{C})$ defined as follows:

$$\mathbb{F}(X) = \{(\varepsilon, \beta(\mathcal{S}))\} \cup \left\{(u.\lambda, C') \,\middle|\, \exists C \in \mathcal{C},\ (u, C) \in X \text{ and } C \xrightarrow{\lambda} C' \right\}$$

This least fix-point is usually not decidable, so we use the Abstract Interpretation framework [6] to compute a sound – but not necessary complete approximation of it. More precisely, we use the relaxed version of Abstract Interpretation [7], in where, among others, the abstract domain is not supposed to be complete under lowest upper bound; furthermore, no abstraction function is required.

Definition 1. *An abstraction is a tuple* $(\mathcal{C}^\sharp, \sqsubseteq^\sharp, \bigsqcup^\sharp, \bot^\sharp, \gamma, C_0^\sharp, \rightsquigarrow, \nabla)$ *such that*

1. $(\mathcal{C}^\sharp, \sqsubseteq^\sharp)$ *is a pre-order;*
2. $\bigsqcup^\sharp : \wp_{\mathit{finite}}(\mathcal{C}^\sharp) \rightarrow \mathcal{C}^\sharp$ *such that* $\forall A^\sharp \in \wp_{\mathit{finite}}(\mathcal{C}^\sharp),\ \forall a^\sharp \in A^\sharp,\ a^\sharp \sqsubseteq^\sharp \bigsqcup^\sharp(A^\sharp)$*;*
3. $\bot^\sharp \in \mathcal{C}^\sharp$ *satisfies* $\forall a^\sharp \in \mathcal{C}^\sharp,\ \bot^\sharp \sqsubseteq^\sharp a^\sharp$*;*
4. $\gamma : \mathcal{C}^\sharp \rightarrow \wp(\Sigma^\star \times \mathcal{C})$ *is a monotonic map which satisfies* $\gamma(\bot^\sharp) = \emptyset$*;*
5. $C_0^\sharp \in \mathcal{C}^\sharp$ *is such that* $(\varepsilon, C_0(\mathcal{S})) \in \gamma(C_0^\sharp)$*;*
6. $\rightsquigarrow \in \wp(\mathcal{C}^\sharp \times \Sigma \times \mathcal{C}^\sharp)$ *is an abstract deterministic labeled transition relation over* $\mathcal{C}^\sharp$ *such that :* $\forall C^\sharp \in \mathcal{C}^\sharp,\ \forall (u, C) \in \gamma(C^\sharp),\ \forall \lambda \in \Sigma,\ \forall \overline{C} \in \mathcal{C}$,
$$C \xrightarrow{\lambda} \overline{C} \Longrightarrow \exists \overline{C}^\sharp \in \mathcal{C}^\sharp,\ (C^\sharp \overset{\lambda}{\rightsquigarrow} \overline{C}^\sharp \text{ and } (u.\lambda, \overline{C}) \in \gamma(\overline{C}^\sharp));$$
7. $\nabla : \mathcal{C}^\sharp \times \mathcal{C}^\sharp \rightarrow \mathcal{C}^\sharp$ *is a widening operator which satisfies:*
   - $\forall C_1^\sharp,\ C_2^\sharp \in \mathcal{C}^\sharp,\ C_1^\sharp \sqsubseteq^\sharp C_1^\sharp \nabla C_2^\sharp$ *and* $C_2^\sharp \sqsubseteq C_1^\sharp \nabla C_2^\sharp$,
   - $\forall (C_n^\sharp)_{n \in \mathbb{N}} \in \left(\mathcal{C}^\sharp\right)^{\mathbb{N}}$*, the sequence* $(C_n^\nabla)_{n \in \mathbb{N}}$ *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.

$\mathcal{C}^\sharp$ is an abstract domain. It captures the properties we are interested in, and abstracts away many other properties. The pre-order $\sqsubseteq^\sharp$ describes the amount of information which is known about the properties we approximate. We use only a pre-order to allow some concrete properties to be described by several unrelated abstract elements. $\bigsqcup^\sharp$ is used to gather the information described by several abstract elements, for the sake of generality, it does not necessarily compute the lowest upper bound of a finite set of abstract elements which may not even exist. $\bot^\sharp$ describes the empty set, it provides the basis for our abstract iteration. γ is a concretization function which maps each abstract property to the set of the concrete elements which satisfy this property. $C_0^\sharp$ is an abstract element which describes the properties satisfied by the initial configuration of the system. $\rightsquigarrow$ is used for mimicking the concrete transition system in the abstract domain and ∇ is used to ensure the convergence of the analysis.

In accordance with Def. 1.6, the abstract counterpart $\mathbb{F}^\sharp$ to $\mathbb{F}$, defined as:

$$\mathbb{F}^\sharp(X^\sharp) = \bigsqcup^\sharp \left\{ \overline{C}^\sharp \mid \exists \lambda \in \Sigma,\ C^\sharp \in X^\sharp,\ C^\sharp \overset{\lambda}{\rightsquigarrow} \overline{C}^\sharp \right\} \cup \{C_0^\sharp; X^\sharp\}$$

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

Theorem 1. $\mathit{lfp}_\emptyset \mathbb{F} \subseteq \bigcup\limits_{n\in\mathbb{N}} \left[\gamma \circ {\mathbb{F}^\sharp}^n\right](\bot^\sharp)$.

Following [5], we compute a sound and decidable approximation of our abstract semantics by using the widening operator ∇:

Theorem 2. *The abstract iteration [7, 8] of $\mathbb{F}^\sharp$ defined as follows:*

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

is ultimately stationary and its limit $\mathbb{F}^\nabla$ satisfies $\mathit{Coll}(\mathcal{S}) \subseteq \gamma(\mathbb{F}^\nabla)$.

Remark 1. We claim that this framework is highly extensible: given two abstractions $(\mathcal{C}_1^\sharp, \sqsubseteq_1^\sharp, \bigsqcup_1^\sharp, \bot_1^\sharp, \gamma_1, C_{0_1}^\sharp, \rightsquigarrow_1, \nabla_1)$ and $(\mathcal{C}_2^\sharp, \sqsubseteq_2^\sharp, \bigsqcup_2^\sharp, \bot_2^\sharp, \gamma_2, C_{0_2}^\sharp, \rightsquigarrow_2, \nabla_2)$ and a reduction operator[4] $\rho : \mathcal{C}_1^\sharp \times \mathcal{C}_2^\sharp$ which satisfies:

$$\forall a^\sharp \in \mathcal{C}_1^\sharp \times \mathcal{C}_2^\sharp,\ \gamma_1(a_1^\sharp) \cap \gamma_2(a_2^\sharp) \subseteq \gamma_1(b_1^\sharp) \cap \gamma_2(b_2^\sharp),\ \textit{denoting } \rho(a^\sharp) \textit{ by } (b_1^\sharp, b_2^\sharp).$$

The following tuple $(\mathcal{C}^\sharp, \sqsubseteq^\sharp, \bigsqcup^\sharp, \bot^\sharp, \gamma, C_0^\sharp, \rightsquigarrow, \nabla)$ where

- $\mathcal{C}^\sharp = \mathcal{C}_1^\sharp \times \mathcal{C}_2^\sharp$;
- $\sqsubseteq^\sharp$, $\bigsqcup^\sharp$, $\bot^\sharp$ and ∇ are defined pair-wise;
- $\gamma : \begin{cases} \mathcal{C}^\sharp \to \wp(\Sigma^\star \times \mathcal{C}) \\ (a_1^\sharp, a_2^\sharp) \mapsto \gamma_1(a_1^\sharp) \cap \gamma_2(a_2^\sharp); \end{cases}$
- $C_0^\sharp(\mathcal{S}) = \rho(C_{0_1}^\sharp, C_{0_2}^\sharp)$;
- $\rightsquigarrow$ is defined by:
  $a \rightsquigarrow d$ if and only if denoting $\rho(a)$ by $(b_1^\sharp, b_2^\sharp)$, there exists $c_1^\sharp \in \mathcal{C}_1^\sharp$, $c_2^\sharp \in \mathcal{C}_2^\sharp$ such that $b_1^\sharp \rightsquigarrow_1 c_1^\sharp$, $b_2^\sharp \rightsquigarrow_2 c_2^\sharp$, $d = \rho(c_1^\sharp, c_2^\sharp)$.

is also an abstraction.

[4] ρ allow simplifying the properties obtained in the two abstractions

5 Control Flow Analysis

We propose to describe all the potential interactions between all the agents of a given mobile ambient. For that purpose we will compute for each thread an approximation of both the set of the ambients it can be immediately located in and the set of the ambient names which can be communicated to this thread. We want a non-uniform description of this. This means that we will compare the history marker of each thread with the marker of its location and with the markers of the ambient names it is communicated to.

The main difficulty is to synthesize such comparisons throughout computation steps. We use the history marker of each thread as a pivot to synthesize the comparison between the other markers (location markers and markers of the ambient names) of this thread. Furthermore we use synchronization conditions on ambient names and on locations of the agents in establishing a comparison between the history markers of all the involved threads. Our main strategy is easy: we first gather all the information we have about the pairs of markers (this means we will abstract sets of tuples of markers). Then synchronization conditions give equality relationships between tuple components. If equality relationships are satisfiable, the abstract computation step is enabled and we compute, for each new thread, the comparison between its history marker and its other markers.

For each $n \in \mathbb{N}$, we introduce an abstract pre-order $(Id^\sharp_n, \sqsubseteq_n)$ to represent sets of n-uples of history markers. Thus, each $Id^\sharp_n$ is related to $\wp(Id^n)$ by a monotonic concretization function γ_n. We introduce a few abstract primitives to handle these domains: a representation of the empty set $\bot^\sharp_n$, a representation of the initial identifier $\varepsilon^\sharp$, an abstract union $\bigsqcup_n$, an associative abstract concatenation $\bullet^\sharp$ to gather the abstraction of tuple sets, an abstract join *assert* to enforce synchronization conditions, an abstract projection Π and an abstract push operator *push* which is used to calculate the abstraction of the set of the new markers when fetching a resource. These primitives shall satisfy the following properties:

- $\gamma_n(\bot^\sharp_n) = \emptyset$;
- $\varepsilon \in \gamma_1(\varepsilon^\sharp)$;
- $\forall A \in \wp_{finite}(Id^\sharp_n),\ \bigsqcup_n(A) \in Id^\sharp_n$ and $\forall a^\sharp \in A,\ a^\sharp \sqsubseteq_n \bigsqcup_n(A)$;
- $\forall a \in Id^\sharp_n, b \in Id^\sharp_m,\ (a \bullet^\sharp b) \in Id^\sharp_{n+m}$ and
  $\left\{ (id_i)_{i\in[|1;n+m|]} \,\middle|\, \begin{array}{l} (id_i)_{i\in[|1;n|]} \in \gamma_n(a^\sharp), \\ (id_{i+n})_{i\in[|1;m|]} \in \gamma_m(b^\sharp) \end{array} \right\} \subseteq \gamma_{n+m}(a \bullet^\sharp b)$;
- $\forall a^\sharp \in Id^\sharp_n,\ \forall A \in \wp([|1;n|]^2),\ assert(A, a^\sharp) \in Id^\sharp_n$ and
  $\{(id_i)_{i\in[|1;n|]} \mid (id_i) \in \gamma_n(a^\sharp),\ \forall (k,l) \in A,\ id_k = id_l\} \in \gamma_n(assert(A, a^\sharp))$;
- $\forall a^\sharp \in Id^\sharp_n, \forall p \in \mathbb{N},\ \forall (s_k)_{k\in[|1;p|]} \in [|1;n|]^{[|1;p|]}$ such that (s_k) is a one to one sequence, $\Pi_{(s_k)}(a^\sharp) \in Id^\sharp_p$ and
  $\{(id_{s_k})_{k\in[|1;p|]} \mid (id_i)_{i\in[|1;n|]} \in \gamma_n(a^\sharp)\} \subseteq \gamma_p(\Pi_{(s_k)}(a^\sharp))$;
- $\forall a^\sharp \in Id^\sharp_3,\ push_{(i,j)}(a^\sharp) \in Id^\sharp_2$ and
  $\{(N((i,j), id_1, id_2), id_3) \mid (id_1, id_2, id_3) \in \gamma_3(a^\sharp)\} \subseteq \gamma_2(push_{(i,j)}(a^\sharp))$.

Moreover we define the operator $dpush \in \mathcal{F}(Id^\sharp_1, Id^\sharp_2)$ by:

$$\forall a \in Id^\sharp_1,\ dpush(a) = assert(\{(1,2)\}, a \bullet^\sharp a).$$

dpush satisfies the following property:

$$\forall a \in Id_1^\sharp,\ \{(id, id) \mid id \in \gamma_1(a)\} \subseteq \gamma_2(dpush(a)).$$

We denote by $\mathcal{P}$ the set of all the syntactic components of $\mathcal{S}$. We describe the set of ambients in which threads can be launched by associating with each pair $(P, i) \in (\mathcal{P} \times Lbl)$ a description of the set of the marker pairs (id_P, id_i) such that an instance of P may be stamped with both the history marker id_P and the location marker (i, id_i). In addition, we describe the set of the ambient names which can be communicated to a thread by relating each triplet $(P, m, n) \in (\mathcal{P} \times \mathcal{BN}(\mathcal{S}) \times \mathcal{BN}(\mathcal{S}))$ to a description of the set of the marker pairs (id_P, id_n) such that an instance of the P may be stamped with the marker id_P whereas the syntactic name m of P is bound to the ambient name (n, id_n). In this way, our abstraction $(\mathcal{C}^\sharp, \sqsubseteq^\sharp, \bigsqcup^\sharp, \bot^\sharp, \gamma, C_0^\sharp, \rightsquigarrow, \nabla)$ is defined as follows:

- $\mathcal{C}^\sharp = \mathcal{F}(\mathcal{P} \times Lbl, Id_2^\sharp) \times \mathcal{F}(\mathcal{P} \times \mathcal{BN}(\mathcal{S}) \times \mathcal{BN}(\mathcal{S}), Id_2^\sharp)$
- $\sqsubseteq^\sharp, \bigsqcup^\sharp, \nabla$ are defined component-wise then pair-wise.
- $\bot^\sharp$ is given by the pair of functions which relate any elements to $\bot_2^\sharp$;
- $\gamma((f, g))$ is given by the set of the configurations C which satisfy:
  $$(P, id, (i, id_i), E) \in C \Longrightarrow \begin{cases} (id, id_i) \in \gamma_2(f(P, i)) \text{ and} \\ E(m) = (n, id_n) \Longrightarrow (id, id_n) \in \gamma_2(g(P, m, n)). \end{cases}$$
- $C_0^\sharp$ and $\rightsquigarrow$ are given in Figs. 5-7.

Abstract transition rules just mimic the non-standard ones, therefore they are quite complicated because non-standard transition rules handle several synchronizations between markers. Their definition uses an abstract extraction function $\beta^\sharp$ defined in Fig. 4.

- $\beta^\sharp(\mathrm{n}^{\mathrm{i}}[P], id^\sharp, loc^\sharp, E^\sharp) = (a[(\mathrm{n}^{\mathrm{i}}[\bullet], j) \mapsto loc^\sharp(j)], b[(\mathrm{n}^{\mathrm{i}}[\bullet], \mathrm{n}, m) \mapsto E^\sharp(\mathrm{n}, m)])$
  where $(a, b) = \beta^\sharp(P, id^\sharp, [\mathrm{i} \mapsto dpush(id^\sharp)], E^\sharp)$;
- $\beta^\sharp(P \mid Q, id^\sharp, loc^\sharp, E^\sharp) = \bigsqcup^\sharp\{\beta^\sharp(P, id^\sharp, loc^\sharp, E^\sharp); \beta^\sharp(Q, id^\sharp, loc^\sharp, E^\sharp)\}$;
- $\beta^\sharp((\nu\ \mathrm{n})P, id^\sharp, loc^\sharp, E^\sharp) = \beta^\sharp(P, id^\sharp, loc^\sharp, E^\sharp[(\mathrm{n}, \mathrm{n}) \mapsto dpush(id^\sharp)])$;
- $\beta^\sharp(M, id^\sharp, loc^\sharp, E^\sharp) = (a, b)$
  where $\begin{cases} a = [(M, i) \mapsto loc^\sharp(i)] \\ b = [(M, m, n) \mapsto E^\sharp(m, n)\ \ if\ m \in \mathcal{FN}(M)]; \end{cases}$
- $\beta^\sharp(io, id^\sharp, loc^\sharp, E^\sharp) = (a, b)$
  where $\begin{cases} a = [(io, i) \mapsto loc^\sharp(i)] \\ b = [(io, m, o) \mapsto E^\sharp(m, o)\ \ if\ m \in \mathcal{FN}(io)]; \end{cases}$
- $\beta^\sharp(\mathbf{0}, id^\sharp, loc^\sharp, E^\sharp) = \bot^\sharp$.

Fig. 4. Abstract extraction function

$\beta^\sharp$ is an abstract counterpart to β. It calculates all the interactions obtained by spawning a continuation in an abstract location: given $P \in \mathcal{P}$, $id^\sharp \in Id_1^\sharp$,

$$C_0^\sharp = \beta^\sharp(\mathcal{S}, \varepsilon^\sharp, [\text{top} \mapsto dpush(\varepsilon^\sharp)], \emptyset)$$

Fig. 5. Abstract initial configuration

Let $(f,g) \in \mathcal{C}^\sharp$, if there are $\lambda = n^i[\bullet]$, $\mu = m^j[\bullet]$, $\psi = in^k o.P$,
if $\bigsqcup_{10}\{A(l_\lambda, n_m) \mid l_\lambda \in Lbl,\ n_m \in \mathcal{BN}(\mathcal{S})\} \neq \bot_{10}^\sharp$,
then $(f,g) \overset{in(i,j,k)}{\rightsquigarrow} \bigsqcup^\sharp\{(f',g); \beta^\sharp(P, id^\sharp, loc^\sharp, E^\sharp)\}$ where

- $A(l_\lambda, n_m) = assert(sync, f(\lambda, l_\lambda) \bullet^\sharp f(\mu, l_\lambda) \bullet^\sharp f(\psi, i) \bullet^\sharp g(\mu, m, n_m) \bullet^\sharp g(\psi, o, n_m))$,
- $sync = \{(2,4);(1,6);(8,10);(3,7);(5,9)\}$,
- $f' = f[(\lambda, j) \mapsto \bigsqcup_2(\{f(\lambda,j)\} \cup \{\Pi_{(1,3)}(A(l_\lambda, n_m)) \mid l_\lambda \in Lbl,\ n_m \in \mathcal{BN}(\mathcal{S})\})]$,
- $id^\sharp = \bigsqcup_1\{\Pi_{(5)}(A(l_\lambda, n_m)) \mid l_\lambda \in Lbl, n_m \in \mathcal{BN}(\mathcal{S})\}$,
- $loc^\sharp = [i \mapsto \bigsqcup_2\{\Pi_{(5,6)}(A(l_\lambda, n_m)) \mid l_\lambda \in Lbl, n_m \in \mathcal{BN}(\mathcal{S})\}]$,
- $E^\sharp = [(q,r) \mapsto \Pi_{(1,2)}(assert(\{(1,3)\}, (g(\phi,q,r) \bullet^\sharp id^\sharp))), \forall (q,r) \in \mathcal{BN}(\mathcal{S})^2]$.

Let $(f,g) \in \mathcal{C}^\sharp$, if there are $\lambda = m^i[\bullet]$, $\mu = n^j[\bullet]$, $\psi = out^k o.P$,
if $\bigsqcup_{10}\{A(l_\lambda, n_m) \mid l_\lambda \in Lbl,\ n_m \in \mathcal{BN}(\mathcal{S})\} \neq \bot_{10}^\sharp$,
then $(f,g) \overset{out(i,j,k)}{\rightsquigarrow} \bigsqcup^\sharp\{(f',g); \beta^\sharp(P, id^\sharp, loc^\sharp, E^\sharp)\}$ where

- $A(l_\lambda, n_m) = assert(sync, f(\lambda, l_\lambda) \bullet^\sharp f(\mu, i) \bullet^\sharp f(\psi, j) \bullet^\sharp g(\lambda, m, n_m) \bullet^\sharp g(\psi, o, n_m))$,
- $sync = \{(1,4);(1,7);(3,6);(5,9);(8,10)\}$,
- $f' = f[(\mu, l_\lambda) \mapsto \bigsqcup_2(\{f(\mu,l_\lambda)\} \cup \{\Pi_{(3,2)}(A(l_\lambda, n_m)) \mid n_m \in \mathcal{BN}(\mathcal{S})\})]$,
- $id^\sharp = \bigsqcup_1(\{\Pi_{(5)}(A(l_\lambda, n_m)) \mid l_\lambda \in Lbl, n_m \in \mathcal{BN}(\mathcal{S})\})$,
- $loc^\sharp = [i \mapsto \bigsqcup_2(\{\Pi_{(5,6)}(A(l_\lambda, n_m)) \mid l_\lambda \in Lbl, n_m \in \mathcal{BN}(\mathcal{S})\})]$,
- $E^\sharp = [(q,r) \mapsto \Pi_{(1,2)}(assert(\{(1,3)\}, (g(\phi,q,r) \bullet^\sharp id^\sharp))), \forall (q,r) \in \mathcal{BN}(\mathcal{S})^2]$.

Let $(f,g) \in \mathcal{C}^\sharp$, if there are $\lambda = open^i m.P$, $\mu = n^j[\bullet]$,
if $\bigsqcup_{8}\{A(l_\lambda, n_m) \mid l_\lambda \in Lbl,\ n_m \in \mathcal{BN}(\mathcal{S})\} \neq \bot_{8}^\sharp$,
then $(f,g) \overset{open(i,j)}{\rightsquigarrow} \bigsqcup^\sharp\{(f',g); \beta^\sharp(P, id^\sharp, loc^\sharp, E^\sharp)\}$ where

- $A(l_\lambda, n_m) = assert(sync, f(\lambda, l_\lambda) \bullet^\sharp f(\mu, l_\lambda) \bullet^\sharp g(\lambda, m, n_m) \bullet^\sharp g(\mu, o, n_m))$,
- $sync = \{(1,5);(2,4);(3,7);(6,8)\}$,
- $f' = f[(\psi, l_\lambda) \mapsto \bigsqcup_2(\{f(\psi,l_\lambda)\} \cup \{B(\psi, l_\lambda, n_m) \mid n_m \in \mathcal{BN}(\mathcal{S})\}), \forall \psi \in \mathcal{P}]$,
  where $B(\psi, l_\lambda, n_m) = \Pi_{(9,2)}(assert(\{(3,10)\}, A(l_\lambda, n_m) \bullet^\sharp f(\psi, j)))$,
- $id^\sharp = \bigsqcup_1\{\Pi_{(1)}(A(l_\lambda, n_m)) \mid l_\lambda \in Lbl,\ n_m \in \mathcal{BN}(\mathcal{S})\}$,
- $loc^\sharp = [l_\lambda \mapsto \bigsqcup_2\{\Pi_{(1,2)}(A(l_\lambda, n_m)) \mid\ n_m \in \mathcal{BN}(\mathcal{S})\},\ \forall l_\lambda \in Lbl]$,
- $E^\sharp = [(q,r) \mapsto \Pi_{(1,2)}(assert(\{(1,3)\}, (g(\lambda,q,r) \bullet^\sharp id^\sharp))), \forall (q,r) \in \mathcal{BN}(\mathcal{S})^2]$.

Let $(f,g) \in \mathcal{C}^\sharp$, if there are $\lambda = !open^i m.P$, $\mu = n^j[\bullet]$,
if $\bigsqcup_{8}\{A(l_\lambda, n_m) \mid l_\lambda \in Lbl,\ n_m \in \mathcal{BN}(\mathcal{S})\} \neq \bot_{8}^\sharp$,
then $(f,g) \overset{open(i,j)}{\rightsquigarrow} \bigsqcup^\sharp\{(f',g); \beta^\sharp(P, id^\sharp, loc^\sharp, E^\sharp)\}$ where

- $A(l_\lambda, n_m) = assert(sync, f(\lambda, l_\lambda) \bullet^\sharp f(\mu, l_\lambda) \bullet^\sharp g(\lambda, m, n_m) \bullet^\sharp g(\mu, o, n_m))$,
- $sync = \{(1,5);(2,4);(3,7);(6,8)\}$,
- $f' = f[(\psi, l_\lambda) \mapsto \bigsqcup_2(\{f(\psi,l_\lambda)\} \cup \{I(\psi, l_\lambda, n_m) \mid n_m \in \mathcal{BN}(\mathcal{S})\}), \forall \psi \in \mathcal{P}]$
  where $I(\psi, l_\lambda, n_m) = \Pi_{(9,2)}(assert(\{(3,10)\}, A(l_\lambda, n_m) \bullet^\sharp f(\psi, j)))$,
- $id^\sharp = \bigsqcup_1\{\Pi_{(1)}(loc^\sharp(l_\lambda)) \mid \forall l_\lambda \in Lbl\}$,
- $loc^\sharp = [l_\lambda \mapsto push_{(i,j)}(\bigsqcup_3\{\Pi_{(1,3,2)}(A(l_\lambda, n_m)) \mid n_m \in \mathcal{BN}(\mathcal{S})\}),\ \forall l_\lambda \in Lbl]$,
- $E^\sharp = [(q,r) \mapsto push_{(i,j)} \bigsqcup_3\{I(q,r,l_\lambda, n_m) \mid l_\lambda \in Lbl,\ n_m \in Lbl\}, \forall (q,r) \in \mathcal{BN}(\mathcal{S})^2]$
  where $I(q,r,l_\lambda,n_m) = \Pi_{(1,3,10)}(assert(\{(1,9)\}, A(l_\lambda, n_m) \bullet^\sharp g(\lambda, q, r)))$.

Fig. 6. Abstract move rules.

Let $(f,g) \in \mathcal{C}^\sharp$, if there are $\lambda = (n)^i.P$, $\mu = \langle m\rangle^j$ two sub-processes of $\mathcal{S}$, if $\bigsqcup_4\{A(l_\lambda) \mid l_\lambda \in Lbl\} \neq \bot_4^\sharp$,
then $(f,g) \overset{com(i,j)}{\rightsquigarrow} \bigsqcup^\sharp\{(f,g); \beta^\sharp(P, id^\sharp, loc^\sharp, E^\sharp)\}$ where

- $A(l_\lambda) = assert(\{(2,4)\}, f(\lambda, l_\lambda) \bullet^\sharp f(\mu, l_\lambda))$
- $id^\sharp = \bigsqcup_1(\{\Pi_{(1)}(A(l_\lambda)) \mid \forall l_\lambda \in Lbl\})$,
- $loc^\sharp = [l_\lambda \mapsto \Pi_{(1,2)}(A(l_\lambda)),\ \forall l_\lambda \in Lbl]$,
- $E^\sharp = [(q,r) \mapsto \bigsqcup_2\{I(q,r,l_\lambda) \mid l_\lambda \in Lbl\},\ \forall (q,r) \in \mathcal{BN}(\mathcal{S})^2]$
  where $I(q,r,l_\lambda) = \begin{cases} \Pi_{(1,6)}(assert(\{(3,5)\}, A(l_\lambda) \bullet^\sharp g(\mu, m, r))) & \textit{if } q = n \\ \Pi_{(1,6)}(assert(\{(1,5)\}, A(l_\lambda) \bullet^\sharp g(\lambda, q, r))) & \textit{otherwise} \end{cases}$

Let $(f,g) \in \mathcal{C}^\sharp$, if there are $\lambda = !(n)^i.P$, $\mu = \langle m\rangle^j$ two sub-processes of $\mathcal{S}$, if $\bigsqcup_4\{A(l_\lambda) \mid l_\lambda \in Lbl\} \neq \bot_4^\sharp$,
then $(f,g) \overset{com(i,j)}{\rightsquigarrow} \bigsqcup^\sharp\{(f,g); \beta^\sharp(P, id^\sharp, loc^\sharp, E^\sharp)\}$ where

- $A(l_\lambda) = assert(\{(2,4)\}, f(\lambda, l_\lambda) \bullet^\sharp f(\mu, l_\lambda))$,
- $id^\sharp = \bigsqcup_1\{\Pi_{(1)}(loc^\sharp(l_\lambda)) \mid l_\lambda \in Lbl\}$,
- $loc^\sharp = [l_\lambda \mapsto push_{(i,j)}(\Pi_{(1,3,2)}(A(l_\lambda))),\ \forall l_\lambda \in Lbl]$,
- $E^\sharp = [(q,r) \mapsto \bigsqcup_2\{I(q,r,l_\lambda) \mid l_\lambda \in Lbl\}\ \forall (q,r) \in \mathcal{BN}(\mathcal{S})^2]$ where
  $I(q,r,l_\lambda) = \begin{cases} push_{(i,j)}(\Pi_{(1,3,6)}(assert(\{(3,5)\}, A(l_\lambda) \bullet^\sharp g(\mu, m, r)))) & \textit{if } q = n \\ push_{(i,j)}(\Pi_{(1,3,6)}(assert(\{(1,5)\}, A(l_\lambda) \bullet^\sharp g(\lambda, q, r)))) & \textit{otherwise} \end{cases}$

Fig. 7. Abstract communication rules.

$loc^\sharp \in \mathcal{F}(Lbl, Id_2^\sharp)$ and $E^\sharp \in \mathcal{F}(\mathcal{BN}(\mathcal{S}) \times \mathcal{BN}(\mathcal{S}), Id_2^\sharp)$, $\beta^\sharp(P, id^\sharp, loc^\sharp, E^\sharp)$ gives a pair $(a,b) \in \mathcal{C}^\sharp$ which describes all the interactions obtained by spawning a syntactic component P identified by a marker described by $id^\sharp$, in a location described by $loc^\sharp$ and with an environment described by $E^\sharp$, as expressed by the following proposition:

Proposition 2. *$\beta(P, id, (i, id_i), E) \in \gamma(\beta^\sharp(P, id^\sharp, loc^\sharp, E^\sharp))$, $\forall i \in Lbl, id, id_i \in Id$ with $id \in \gamma_1(id^\sharp), (id, id_i) \in \gamma_2(loc^\sharp(i))$ and $\forall E \in \mathcal{F}(\mathcal{FN}(P), (\mathcal{BN}(\mathcal{S}) \times Id))$ such that $\forall m, n \in \mathcal{BN}(\mathcal{S})$, $\forall id_n \in Id$, $[E(m) = (n, id_n) \Rightarrow (id, id_n) \in \gamma_2(E^\sharp(m,n))]$.*

We now give some intuition about the abstract transition rules. For the sake of the brevity, we focus on the *in* migration abstract rule. The three syntactic components λ, μ and ψ denote the three threads involved in the non-standard *in* migration rule. We check for each pair (l_λ, n_m) whether there can be a configuration containing instances of λ, μ and ψ, such that both instances of λ and μ are surrounded by an instance of an ambient labeled with l_λ, ψ is located in the instance of λ and both the ambient name of the instance μ and the name the capability of ψ work on are linked to an ambient name created by an instance of the action $(\nu\ n_m)$. We then compute $A(l_\lambda, n_m)$ which is a description of the relation between the involved markers: we first gather the descriptions of all the involved marker pair abstractions: the first three marker pair abstractions describe the location of λ, μ and ψ while the two last marker pair abstractions describe the linkage of the syntactic ambient names m in λ and o in ψ; we then take into account synchronization conditions between the components of these abstract tuples: the third and the seventh (resp. the fifth and the ninth) components shall be equal since they both denote the thread marker associated to μ (resp. ψ); the synchronization between the second and the fourth components denotes that λ and μ must be located in the same instance of the ambient labeled l_λ; the one between the first and the sixth denotes that ψ must be located in the good instance of the ambient λ and the one between the eight and the tenth enforces the equality between the ambient name of μ and the name the capability of ψ works on. We then extract from $A(l_\lambda, n_m)$ an approximation of the interactions which may be created by performing the *in* migration rule on this redex: the instance of λ can move inside the one of μ, keeping the same history marker (given by the first component of the abstract tuples), but its location marker is then the history marker of the instance of μ (given by the third component). We are left to spawn the continuation of ψ, all its markers remain unchanged. The other abstract rules follow the same schema.

6 Abstract Domain

Various domains can be used to instantiate the family of parametric domains $(Id_n)_{n\in\mathbb{N}}$, depending on the expected complexity and accuracy. We propose three particular instantiations. The first one abstracts away the information about markers. The result is an uniform control flow analysis. The second one only keeps the equality relationships among markers, and gives an analysis which presents strong connections with group creation [1]. The third one allows for the algebraic comparison of markers which is, to the best of our knowledge, out of the range of analyses presented in literature.

6.1 Uniform Control Flow Analysis

An uniform analysis can be obtained by instantiating all the elements of the family $(Id_n^\sharp)_{n\in\mathbb{N}}$ with the lattice $(\{\bot,\top\},\sqsubseteq)$. $\{\bot,\top\}$ is related to $\wp(Id^n)$ by the following concretization function γ_n defined by $\gamma_n(\bot)=\emptyset$ and $\gamma_n(\top)=Id^n$.

The abstract primitives are then defined as follows:

- $\varepsilon^\sharp = \top$;
- $\forall A \in \wp(\{\bot;\top\}),\ \bigsqcup_n(A) = \begin{cases}\bot & \textit{if } \top \notin A\\ \top & \textit{otherwise};\end{cases}$
- $\forall a, b \in \{\bot,\top\},\ (a \bullet^\sharp b) = \begin{cases}\bot & \textit{if } a=\bot \textit{ or } b=\bot\\ \top & \textit{if } a=\top \textit{ and } b=\top;\end{cases}$
- $\forall a \in \{\bot,\top\},\ \textit{assert}(A,a) = a,\ \Pi_X(a) = a \textit{ and } \textit{push}_{(i,j)}(a) = a.$

The resulting analysis is always at least as precise as [17], but takes into account unreachable code.

6.2 Confinement

We now focus on the equality relationships between markers. This allows us to analyze whether an ambient name can only be communicated to the recursive instance which has created it, and whether a thread is always surrounded by an ambient activated by the recursive instance which have spawned it.

We define $Id_n^\sharp$ as the lifted set $\mathcal{G}_n \cup \bot_n$ of all the non-oriented graph having vertices in $[|1;n|]$. The transitive closure of a graph $(\mathcal{G},\curvearrowright)$ is denoted by $(\mathcal{G},\curvearrowright^*)$. The pre-order, the concretization function and the abstract primitives are defined on $\mathcal{G}_n$ as follows, and they can be easily lifted to $\mathcal{G}_n \cup \bot_n$:

- $\forall \curvearrowright_1, \curvearrowright_2 \in \wp([|1;n|]^2),\ ([|1;n|],\curvearrowright_1) \sqsubseteq_n ([|1;n|],\curvearrowright_2) \iff \curvearrowright_2 \subseteq \curvearrowright_1$;
- $\forall([|1;n|],\curvearrowright) \in \mathcal{G}_n,\ \gamma_n(([|1;n|],\curvearrowright)) = \{(id_i)_{i\in[|1;n|]} \mid k \curvearrowright l \implies id_k = id_l\}$;
- $\varepsilon^\sharp = (\{1\},\emptyset)$;
- $\forall A \in \wp(\mathcal{G}_n),\ \bigsqcup_n(A) = ([|1;n|];\curvearrowright_\cup)$ where $i \curvearrowright_\cup j \overset{\Delta}{\iff} \forall([|1;n|],\curvearrowright) \in A, i \curvearrowright^* j$;
- $\forall a = ([|1;n|],\curvearrowright_a) \in \mathcal{G}_n, b = ([|1;m|],\curvearrowright_b) \in \mathcal{G}_m,\ (a \bullet^\sharp b) = ([|1;n+m|],\curvearrowright_\bullet)$ where $i \curvearrowright_\bullet j \overset{\Delta}{\iff} \begin{cases} i \curvearrowright_a j \ \mathit{if}\ i,j \in [|1;n|] \\ (i-n) \curvearrowright_b (j-n) \ \mathit{if}\ i,j \in [|n+1;n+m|]; \end{cases}$
- $\forall([|1;n|],\curvearrowright) \in \mathcal{G}_n, A \in \wp([|1;n|]^2),\ \mathit{assert}(A,([|1;n|],\curvearrowright)) = ([|1;n|], A \cup \curvearrowright)$;
- $\forall a = ([|1;n|],\curvearrowright_a) \in \mathcal{G}_n,\ \Pi_{(s_k)_{k\in[|1;p|]}}(a) = ([|1;p|],\curvearrowright_\Pi)$ where $i \curvearrowright_\Pi j \overset{\Delta}{\iff} i,j \in [|1;p|],\ s_i \curvearrowright_a^* s_j$;
- $\forall a \in \mathcal{G}_3,\ \mathit{push}_{(i,j)}(a) = ([|1;2|],\emptyset)$.

As in [1] this analysis can only prove that an ambient name is confined inside the scope of the recursive instance which has declared it. It is unable to prove that a name which first exits this scope can then only be sent back to the recursive instance which had created it.

6.3 Non-Uniform Analysis with Algebraic Comparisons

We now abstract algebraic comparisons between markers. Following Prop. 1, we only abstract the right comb of each tree. We then use the reduced-product of two abstractions. Our first abstraction consists in abstracting component-wise the shape of the history markers associated to threads, their locations, and their ambient names. We use a regular description of sets of words: we introduce Id_n^{Reg} as the set of all the n-uples of regular automata over the alphabet Lbl^2. Id_n^{Reg} is related to $\wp(Id^n)$ by the following concretization function:

$$\gamma_n((A_i)_{i\in[|1;n|]}) = \{(id_i)_{i\in[|1;n|]} \mid \forall i \in [|1;n|],\ \phi_1(id_i) \in \mathcal{L}(A_i)\}$$

- $\varepsilon^\sharp$ is an automaton which only recognizes the word ε;
- $\bigsqcup_n$ applies component-wise the classical finite union of regular automata;
- $\bullet^\sharp$ (resp. Π) is the classical concatenation (resp. projection) of tuples;
- $\mathit{assert}(\{(a,b)\} \cup A, Q)_i = \begin{cases} (\mathit{assert}(A,Q))_i & \text{if } i \notin \{a;b\} \\ (\mathit{assert}(A,Q))_a \cap (\mathit{assert}(A,Q))_b & \text{otherwise} \end{cases}$

- since there can be infinite increasing sequences of regular languages, we need a widening operator ∇. It is sufficient to construct a widening operator for regular automata and to apply it component-wise. Given $\delta \in \mathbb{N}^*$, a convenient choice for $A_1 \nabla A_2$ consists in quotienting the set of the states of the automaton $A_1 \cup A_2$ by the relation $\sim_\delta$ that identifies the states of an automaton which have the same δ-depth residues[5]. The higher δ is, the more accurate and expensive the analysis is.

Our second abstraction captures non-uniform comparisons between the number of occurrences of each pattern inside sets of marker pairs. For each $n \in \mathbb{N}$, we introduce the set $\mathcal{V}_n$ of distinct variables $\{x_i^\lambda \mid i \in [|1;n|],\ \lambda \in Lbl^2\}$. The abstract domain $\wp(\mathbb{N}^{\mathcal{V}_n})$ is related to $\wp(Id^n)$ by the monotonic map γ_n:

$$\gamma_n(A) = \{(id_i)_{i \in [|1;n|]} \mid \exists (n_t)_{t \in \mathcal{V}_n} \in A,\ \forall x_i^\lambda \in \mathcal{V}_n,\ n_{x_i^\lambda} = |\phi_1(id_i)|_\lambda\}.$$

$\wp(Id^n)$ is then related to the complete lattice of the affine equality systems on the set of variables $\mathcal{V}_n$, denoted by Id_n^{rel}. This domain is described with its lattice operations in [14]. We describe the remaining abstract primitives as follows:

- $\varepsilon^\sharp$ is given by the system $\{x_1^\lambda = 0,\ \forall \lambda \in Lbl^2$;
- given $K \in Id_n^{rel}$ and $K' \in Id_m^{rel}$, we obtain the abstract concatenation of K and K', by renaming each variable x_i^λ to x_{i+n}^λ in K', and gathering all the constrains of the two systems;
- $assert(\{i_1 = j_1; ...; i_p = j_p\}, K)$ corresponds to inserting all the constrains of the form $x_{i_k}^\lambda = x_{j_k}^\lambda$, $\forall k \in [|1;p|]$, $\lambda \in Lbl^2$ in K;
- $\Pi_{(i_1,\ldots,i_p)}(K)$ corresponds to collect all the constrains involving just the variables $\{x_{i_k}^\lambda\}$ and then renaming each variable $x_{i_k}^\lambda$ into the variable x_k^λ;
- $push_{(i,j)}(K)$ is obtained by replacing in each constrains each occurrence of the variable $x_2^{(i,j)}$ by the expression $x_2^{(i,j)} - 1$ and then applying the abstract projection $\Pi_{(2,3)}$.

Example 3. We run the third analysis on the system of Example 1, we denote the result by (f,g). We succeed in proving that an ambient name created by the binder $(\nu\ q)$ can only be communicated either to the agent $\langle q\rangle^{14}$ in a "request" ambient surrounded by a p ambient, or to the agent $\langle rep\rangle^9$ in an "answer" ambient surrounded by a p ambient. In the second case, we also capture these properties:

$$\begin{cases} \gamma_2(g(\langle rep\rangle^9, rep, q)) = \{(3.19).(11,20)^n.(11,21), (11,20)^n(11,21)\} \\ \gamma_2(f(\langle rep\rangle^9, 8)) = \{(3.19).(11,20)^n.(11,21), (3.19).(11,20)^n.(11,21)\} \\ \gamma_2(f(\text{answer}^8[\bullet], 12)) = \{(3.19).(11,20)^n.(11,21), (11,20)^n.(11,21)\} \\ \gamma_2(g(p^{12}[\bullet], p, p) = \{(11,20)^n.(11,21), (11,20)^n.(11,21)\}, \end{cases}$$

this proves that the name communicated inside the "answer" ambient and the name of the packet which surrounds this "answer" ambient have been both declared by the same recursive instance of the client resource. $\square$

[5] the δ-depth residue set of a state q in a labeled transition system $(\mathcal{Q}, \Sigma, \rightarrow)$ is defined as $\{u \in \Sigma^* \mid |u| \leqslant \delta \ and\ \exists q' \in \mathcal{Q}\ q \xrightarrow{u} q'\}$

Remark 2. Our confinement analysis is not an abstraction of our non-uniform analysis, because two distinct markers may be recognized by the same automaton while containing the same occurrence number of each pattern (i.e having the same Parikh vector [19]). The equality of the Parikh vector implies the equality of markers if they are recognized by an automaton only composed of an acyclic path between an initial and a final state and without embedded cycle, and such that the set of the Parikh vectors of the cycles of this automaton are linearly independent. Nevertheless, we may use the reduced product of both our confinement analysis and our non-uniform control flow analysis to solve this problem.

6.4 About the complexity of our analyses

We shortly describe the time complexity of our analyses. In the following table, the first line denotes the redex detection and information propagation, the second line denotes the cost of performing an abstract operation and the third one denotes the maximum iteration number.

	0-CFA	confinement	ω-CFA ($\delta = 1$)
scan	$t.n^2.i_p$	$t.n^2.i_p$	$t.n^2.i_p$
domain complexity	1	1	σ^3
abstract iteration height	i	i	$i.\sigma^2$
time-complexity	$i.t.n^2.i_p$	$i.t.n^2.i_p$	$i.t.n^2.\sigma^5.i_p$

where N is the system length; t is the number of the distinct transition labels which occur during the analysis: t is cubic in the worst case, but is only quasi-linear in practice; n is the sum of the number of name binders and the number of ambient activators: it is linear in N; i is the number of interactions between the agents of the system: in practice i is quasi-quadratic in N, but is cubic in the worst case; i_p is a bound to the number of the interactions with Q, for any fixed process Q: i_p is quadratic in worst case in N, but is quasi-linear in practice; σ is the number of pattern occurring in markers: it is either linear or quadratic in N, depending on the choice for the history markers (Cf. Prop. 1).

Both effective transitions and effective interactions are detected during our iteration. This allows us to speed up the analysis, the cost of which only depends on the number of both effective transition kinds and effective interactions.

7 Conclusion and perspectives

We have described a parametric framework for automatically inferring a description of the interferences between recursive instances of the agents of a mobile ambient in a polynomial time. Our framework also applies when extending the model with mobility control [15] or higher order communication. As in the π-calculus [9], we would like to extend this framework to analyze the behaviour of an open system executed in a hostile context.

This framework is highly extensible and is very likely to be enriched by an occurrence counting analysis. Analyses in literature [11, 18] are not polynomial, but we are working on a polynomial one inspired by [10]. Our long-range forecast is to use the low-level properties we compute to synthesize high-level properties which may be expressed in a modal logic as suggested in [3].

References

1. L. Cardelli, G. Ghelli, and A. D. Gordon. Ambient groups and mobility types. In *Proc. TCS'00*, LNCS, pages 333–347. Springer, 2000.
2. L. Cardelli and A. D. Gordon. Types for mobile ambients. In *Proc. POPL'99*, pages 79–92. ACM, 1999.
3. L. Cardelli and A. D. Gordon. Anytime, anywhere: Modal logics for mobile ambients. In *Proc. POPL'00*, pages 365–377. ACM Press, 2000.
4. Luca Cardelli and A. D. Gordon. Mobile ambients. *Theoretical Computer Science*, 240(1):177–213, June 2000.
5. 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.
6. P. Cousot and R. Cousot. Abstract interpretation: a unified lattice model for static analysis of programs by construction or approximation of fixpoints. In *Proc. POPL'77*, pages 238–252, Los Angeles, California, U.S.A., 1977.
7. P. Cousot and R. Cousot. Abstract interpretation frameworks. *Journal of logic and computation*, 2(4):511–547, 1992.
8. P. Cousot and R. Cousot. Comparing the Galois connection and widening--narrowing approaches to abstract interpretation. In *Proc. PLILP'92*, LNCS, pages 269–295. Springer, 1992.
9. J. Feret. Confidentiality analysis of mobile systems. In *Proc. SAS'00*, LNCS, pages 135–154. Springer, 2000.
10. J. Feret. Occurrence counting analysis for the pi-calculus. *Electronic Notes in Theoretical Computer Science*, 39(2), 2001.
11. R. R. Hansen, J. G. Jensen, F. Nielson, and H. Riis Nielson. Abstract interpretation of mobile ambients. In *Proc. SAS'99*, LNCS, pages 134–148. Springer, 1999.
12. M. Hennessy and J. Riely. Resource access control in systems of mobile agents. In *Proc. HLCL'98*, ENTCS. Elsevier, 1998.
13. M. Hennessy and J. Riely. A typed language for distributed mobile processes. In *Proc. POPL'98*. ACM Press, 1998.
14. M. Karr. Affine relationships among variables of a program. *Acta Informatica*, pages 133–151, 1976.
15. F. Levi and D. Sangiorgi. Controlling interference in ambients. In *Proc. POPL'00*, pages 352–364. ACM Press, 2000.
16. R. Milner, J. Parrow, and D. Walker. A calculus of mobile processes. *Information and Computation*, pages 1–77, 1992.
17. F. Nielson, H. Riis Nielson, R. R. Hansen, and J. G. Jensen. Validating firewalls in mobile ambients. In *Proc. CONCUR'99*, LNCS, pages 463–477. Springer, 1999.
18. H. Riis Nielson and F. Nielson. Shape analysis for mobile ambients. In *Proc. POPL'00*, pages 142–154. ACM Press, 2000.
19. R. J. Parikh. On context-free languages. *Journal of the ACM*, 13:570–581, 1966.
20. A. Venet. Automatic determination of communication topologies in mobile systems. In *Proc. SAS'98*, LNCS, pages 152–167. Springer, 1998.