Abstract Interpretation of Mobile Systems [1]

Jérôme Feret

Département d'Informatique de l'École Normale Supérieure,
75230 PARIS Cedex 5, FRANCE

Abstract

We propose an Abstract Interpretation-based context-free analysis for mobile systems written in the π-calculus. Our analysis automatically captures a sound – but not complete – description of the potential behavior of a mobile system interacting with an unknown context. It focuses on both the control flow and the occurrence number of agents during computation sequences.

Control flow analysis detects all the possible interactions between the agents of a system, but also the potential interactions with the context. In order to deal with dynamic creation of both names and agents which is an inherent feature of mobility, our analysis distinguishes between recursive instances of the same agent. This way, we are able to prove the integrity of an *ftp*-server used by an unbounded number of clients. Occurrence counting analysis just consists in abstracting the occurrence number of instances of agents. Our abstraction is relational; this makes us able to detect both quickly and precisely mutual exclusion and non-exhaustion of resources.

Key words: mobility, π-calculus, static analysis, abstract interpretation, control flow analysis, occurrence counting analysis, worst case analysis.

1 Introduction

The development of large scale communicating distributed systems demands the design of methods for analyzing mobile systems. In such systems, agent distribution dynamically changes during computation, making their analysis a very difficult task. Furthermore, the size of these systems, such as the Internet for instance, is large enough to prevent a single person from knowing the whole system. This is why we are interested in validating properties on a

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

mobile system which belongs to a bigger one, called its context, without having precise knowledge of this context. We propose a fully automatic Abstract Interpretation-based analysis for detecting and proving some properties of such mobile systems.

In this study, we focus on mobile systems described in the π-calculus [22, 23] which is a formalism well suited for understanding the problems related to mobility. The π-calculus describes systems of agents which may interact via the communication of channel names through channels. When receiving a channel name, an agent gets some control over this name: it can communicate with other agents sharing this name or even communicate this name to other agents. The expressive power of the π-calculus also follows from a replication mechanism, which allows the spawning of number of instances of a same agent. Thus, each instance of each agent opens its own channels and can then communicate their names to other agents. The semantics of the π-calculus is usually described up to a congruence relation. This relation allows agents to interact by solving conflict between channel names and making the agents move in the syntactic description of the system. Nevertheless, because of this congruence relation, it would be very difficult to derive an abstraction of the usual semantics of the π-calculus. To solve this problem, we introduce a non-standard semantics. This non-standard semantics is fully operational (i. e. it requires no congruence relation). It uses a fresh name allocation scheme, in order to provide fresh names to opened channels, and describes configurations as sets of agents, so that the congruence relation is no longer required. Moreover, this fresh name allocation scheme allows us to encode some interesting properties in the semantics, such as the fact that two channels have been opened by the same instance of an agent, or by two successive instances of an agent.

Having chosen the appropriate semantics, we use the Abstract Interpretation framework to design decidable analyses of the π-calculus. This framework is highly generic: it can be applied to various analyses, provided some abstract primitives are given. Moreover, it is extensible: it allows us to build the (approximated reduced) product of several analyses expressed in this framework. We then use our framework to address two orthogonal issues: the control flow and the occurrence counting. Control flow analysis consists in detecting the set of agent instances that can receive the names of the channels opened by a given instance of an agent. This analysis is context-free: it will detect which channel names can be communicated to the context. It is also non-uniform in the sense that it distinguishes between recursive instances of agents. It can prove, for instance, that a name can be communicated to only one other instance of an agent, and not to the other ones. In the case of the *ftp*-server, it detects that the server can return a query only to the correct client even in the case where an unbounded number of clients are created. Occurrence counting analysis consists in abstracting the occurrence number of instances of agents during computation sequences. It is especially useful to detect mu-

tual exclusion. It also helps in discovering a bound to the number of agents during computation sequences, so that we can verify that some part of the systems will not exceed physical limits imposed by the implementation of the system. In the case of the *ftp*-server, we can automatically infer the maximum number of simultaneous client sessions. Our approach relies on the use of a reduced product between a non-relational and a relational domain. Complexity problems are solved by using approximated algorithms for calculating a reduction between these two domains.

As a result we get a polynomial analysis which has been implemented in OCAML. The corresponding prototype can be used on the web at:

`http://www.di.ens.fr/~feret/prototypes/prototypes.html.en`

Some examples and a short tutorial are also provided.

Related works are discussed in Sect. 2. The standard semantics of the π-calculus is given in Sect. 3. It is first refined in Sect. 4 and extended to open systems in Sect. 5. A generic abstract analysis is designed in Sect. 6. It is instantiated in both Sect. 7 and Sect. 8 to get, respectively, a control flow and an occurrence counting analysis. Complexity results and analysis times are given in Sect. 9.

2 Related work

Flow analysis. Control flow analyses focus on the explicit flow of information. Nielson *et al.* use abstract interpretation in [1–3] to infer a uniform description of the interactions between agents and apply Seidl's solver to get a cubic implementation of their analysis in [25]. Hennessy and Riely have designed a type-based analysis with the same expressive power in [19]. These analyses use explicit information flow to detect whether some security constraints specified using a security level cannot be violated. Nevertheless, these analyzes are uniform (or *mono-variant*). They cannot distinguish between distinct instances of the same agent. For example, it is impossible to give distinct security levels to distinct instances of the same agent. System specification could be rewritten so that several instances of a given agent are syntacticly distinguished from the others. Therefore, this requires a human intervention to guess which replication have to be syntactically unfolded and how many instances have to be distinguished. Our analysis requires no human analysis, and can find interesting properties even if an arbitrary number of instances have to be distinguished. Moreover, it is not obvious in many cases that there exists a syntactic rewriting of the system so that several different recursive instances can be distinguished on purely syntactic ground and where the security policy

is checkable using purely uniform analyses.

Cardelli *et al.* use group creation in [4] to assign dependent security levels to channel names. A group can be associated with each recursive instance of an agent, and can then be used to prevent names of channels from exiting the scope of the instance which has opened these channels. Nevertheless, our analysis is much more expressive: we can infer algebraic comparisons between agent instances, which allows us to express the fact that an instance of an agent can only communicate with the next instance of it or with the previous one. As a consequence, we can prove that the name of a channel is sent back to the instance which has previously opened this channel, even if this name is not confined into the scope of this recursive instance.

Venet has already proposed a non-uniform analysis in [30, 31]. This analysis infers a sound non-uniform description of the topology of communications between the agents of *friendly systems* [22], in which replication guards cannot be nested and systems are closed. The main contributions with respect to Venet's work are:

(1) the extension of the non-uniform analysis to agents with nested replication guards;
(2) the extension to open systems acting in a possibly unknown context;
(3) the occurrence counting analysis.

Occurrence counting analysis. Only very few analyses for counting occurrences of agents have been published. Nevertheless, this problem is very close to the problem of approximating the behavior of a Petri net, and of occurrence counting in mobile ambients. In [18], Nielson *et al.* propose an exponential analysis for counting occurrences of agents inside ambients. In [26], they use context-dependent counts for inferring a more accurate description of the internal structure of agents, at the expense of a higher time complexity (an exponential number of agents are distinguished). These analyses rely on the use of a non-relational domain to abstract the content of an ambient. Then, they use disjunctive completion, and abstract the set of all the potential contents of a syntactic ambient as the power set of this abstract domain. These two analyses encounter the same problem: in the case that several instances of the same agent may coexist, when one instance of this agent performs a computation step, these analyses cannot decide whether only one or several instances remain after this computation step, so they have to consider the two possible cases, which leads to both a loss of precision and an exponential explosion in complexity. The use of an approximated reduced product between a relational domain and a non-relational domain to globally abstract sets of multi-sets of agents allows us to solve this problem efficiently. Thus we obtain a very accurate analysis which is polynomial in the number of syntactic agents (i.e. polynomial in the size of the initial system configuration).

Behavioral types. In [20], Kobayashi and Igarashi use CCS processes as types for mobile systems and check some behavioral invariants expressed in modal logic. Nevertheless, describing causality between actions leads to an explosion of the size of the types. Another problem is that their type system cannot express properties that deal with the dynamic creation of channel names. Rajamani and Rehof have extended this type system in [28], so that it handles dynamic name creation. But type checking is undecidable in general. So, they will have to propose an approximation in future work.

Modular analysis. Context-free semantics is an important issue in static analysis. It allows the analysis of only a part of a system, without much knowledge of its context. It can be used to abstract the behavior of an instance of an agent, and detect which names may escape the scope of this part. This can be used to detect dead code, for instance. Rajamani and Rehof propose a modular analysis in [28]. Having abstracted the behavior of two modules, they can calculate an approximation of the parallel composition of them. But this analysis is very restrictive because module types must satisfy some *assume-guarantee* properties.

Security analysis. In the Dolev-Yao's model [13], a system is usually proved not to be vulnerable to a given class of attackers. Such classes are then described by a set of rules which explain how an attacker can interact with a system. In our approach, the class of the attacker is actually any system expressed in the π-calculus. The benefit is that our context-free analysis is complete with respect to the model, so that we can use our results to analyze a module. The main drawback of our approach is that we cannot express other attacks. For instance, we could imagine that an attacker could guess some sensitive information, by observing the time of execution of some agents.

3 π-calculus

We introduce in this section the π-calculus and a standard semantics for it. The π-calculus is a formalism used to describe mobile systems. It describes a system as a set of agents which exchange information over channels. These communications enable agent synchronization, but also dynamic modification of the system topology: agents can open new channels, they can also pass control over some channels to other agents, and they can even dynamically create other agents. Here, we consider a lazy version of the synchronous polyadic π-calculus [22] with internal choice operator. In the polyadic π-calculus, agents can communicate tuples of channel names. We use the lazy version of replication introduced in [29, Chap. 7]: agent creations are performed only when they are required by a communication. This is not a limitation as full replication can be encoded with lazy replication (Cf. [29, page:102]).

3.1 Syntax

Let $\mathcal{N}$ be a countable set of channel names and $\mathcal{L}$ a countable set of labels. The syntax of agents is described in Fig. 1(a). Syntactic components are identified by distinct labels in $\mathcal{L}$. Input guard, replication guard and name restriction act as name binders, i.e in the agents $c?^j[x_1, ..., x_n]P$, $*d?^j[y_1, ..., y_p]Q$ and $(\nu\ x)R$, the occurrences of x_1, ..., x_n in P, y_1, ..., y_p in Q and x in R are bound. We also assume that no name occurs twice in a whole system, as an argument of an input guard, a replication guard or a name restriction. Usual rules about scope, substitution and α-conversion apply. We denote by $fn(P)$ the set of free names in P, i.e names that are not under the scope of a binder, and by $bn(P)$ the set of bound names in P.

3.2 Semantics

We now informally introduce the semantics of the π-calculus. The agent aP first computes the action a before launching the continuation P. The agent $(\nu\ x)P$ opens a new channel, named x, the agent P can use this channel for communicating, it can also send the name x to the other agents. In $(P \mid Q)$, P and Q are two concurrent agents which may behave independently, or interact by communicating. The formula $(P \oplus Q)$ denotes an internal choice between two agents. Either P or Q is run, while the other fades away; the choice between P and Q does not depend on the other agents. The agent $\mathbf{0}$ does nothing. The agent $c!^i[x_1, ..., x_n]P$ sends a message via the channel named c, this message is the tuple of channel names $(x_1, ..., x_n)$. The agent $c?^i[y_1, ..., y_n]P$ waits for a message on the channel named c, and binds the channel names y_1,...,y_n to the received channel names. The agent $*c?^i[y_1, ..., y_n]P$ is a *resource*: it replicates itself just before receiving a message: a new instance of P is launched with y_1,...,y_n bound to the received channel names while $*c?^i[y_1, ..., y_n]P$ waits for the next message.

The operational semantics is given by both a congruence relation in Fig. 1(b) and a reduction relation in Fig. 1(c). The congruence relation allows agents to interact, while the reduction relation describes agent computations. Some rules in the congruence relation make agents move inside the syntactic tree: they assert the associativity and commutativity of the parallel composition. Some others extend the scope of names to the agents they are communicated to: α-conversion solves conflicts between names, swapping selects the name the scope of which we wish to extend, and extrusion extends its scope to another agent. The reduction relation describes agents' communications. A communication is allowed when there are two concurrent agents, such that the first one sends a message on a channel, while the second one waits for a message on

$$
\begin{array}{rcll}
P & ::= & aP & \text{(action)} \\
 & | & (\nu\ x)P & \text{(name restriction)} \\
 & | & (P \mid P) & \text{(parallel composition)} \\
 & | & (P \oplus P) & \text{(internal choice)} \\
 & | & \mathbf{0} & \text{(nil)} \\
a & ::= & c!^j[x_1, ..., x_n] & \text{(output guard)} \\
 & | & c?^i[x_1, ..., x_n] & \text{(input guard)} \\
 & | & *c?^i[x_1, ..., x_n] & \text{(replication guard)}
\end{array}
$$

where $c, x_1, ..., x_n, x \in \mathcal{N},\ i, j \in \mathcal{L}$ *and* $n \geqslant 0$.

(a) Syntax

$$
\begin{array}{rcll}
(\nu\ x)P & \equiv & (\nu\ y)P[x \leftarrow y] & \text{if } y \notin fn(P) \quad (\alpha\text{-conversion}) \\
P \mid Q & \equiv & Q \mid P & \text{(commutativity)} \\
P \mid (Q \mid R) & \equiv & (P \mid Q) \mid R & \text{(associativity)} \\
P \mid \mathbf{0} & \equiv & P & \text{(end of an agent)} \\
(\nu\ x)\mathbf{0} & \equiv & \mathbf{0} & \text{(garbage collecting)} \\
(\nu\ x)(\nu\ y)P & \equiv & (\nu\ y)(\nu\ x)P & \text{(swapping)} \\
((\nu\ x)P) \mid Q & \equiv & (\nu\ x)(P \mid Q) & \text{if } x \notin fn(Q) \quad \text{(extrusion)}
\end{array}
$$

where $x,\ y \in \mathcal{N}$.

(b) Congruence relation

$$
\begin{array}{rl}
c!^j[x_1, ..., x_n]P \mid c?^i[y_1, ..., y_n]Q \xrightarrow{(i,j)} P \mid \widetilde{Q} & \text{(comm.)} \\
c!^j[x_1, ..., x_n]P \mid *c?^i[y_1, ..., y_n]Q \xrightarrow{(i,j)} P \mid \widetilde{Q} \mid *c?^i[y_1, ..., y_n]Q & \text{(replication)} \\
P \oplus Q \xrightarrow{\oplus} P & \text{(left choice)} \\
P \oplus Q \xrightarrow{\oplus} Q & \text{(right choice)}
\end{array}
$$

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

where $c,\ x,\ x_1,\ ...,\ x_n,\ y_1,\ ...,\ y_n \in \mathcal{N},\ i, j \in \mathcal{L},\ \lambda \in \{\oplus\} \cup (\mathcal{L} \times \mathcal{L})$,
and $\widetilde{Q} = Q[y_1 \leftarrow x_1, ..., y_n \leftarrow x_n]$.

(c) Reduction relation

Fig. 1. Standard operational semantics

the same channel (we also request that both messages have the same arity). The results of such a communication are obtained by applying the substitution of the λ-calculus in the continuation of the message receiver. When the receiver is a resource, it is just syntactically replicated before performing the communication; this way the resource is still available after the communication. We have labelled each choice reduction step with the symbol $\oplus$ and each communication reduction step with the labels of both agents involved in the communication: this will allow us to relate the state of a system to the history of the computation steps that have led to this state.

3.3 Examples

We now propose some examples to illustrate both this semantics and the kind of properties we are interested in. We will find that the semantics we have considered is not precise enough to handle the properties we are interested in.

Example 1 *An* ftp*-server can be described by the system given in Fig. 2. The first resource repeatedly creates a new client which sends a query to a server. This query is composed of a query* ***request****, and an address* ***address****. The client sends its query again in the case that it receives a failure report denoted by the agent* ***address![]****. The second resource describes the server. When this one receives a query, it replicates itself. Then, either it uses an available port and computes the query or it reports a failure to the client by spawning the agent* ***address![]****. Available ports are denoted by agents* ***port![]****. Data processing just consists in a communication between two agents of the server, through the channel the name of which is* ***deal****: most computational features are abstracted away. After this communication, the port is released, while the answer is sent back to the client. An instance of the agent* ***email***$![$***rep***$]$ *is left as a trace of the session.*

Our analysis will prove both the integrity and the non-exhaustion of this system: it will discover that each time an agent ***email![rep]*** *is spawned, the names* ***email*** *and* ***rep*** *are respectively bound to the names of two channels opened by the restrictions* (ν ***address***) *and* (ν ***request***) *of the same instance of a resource, and so the server returns its computed answer to the correct client; it also captures the fact that no more than three instances of the syntactic agent* ***deal***$![$***data***$]$ *can occur simultaneously, which means that no more than three simultaneous sessions can be active in the same time.*

Example 2 *We propose in Fig. 3 a mobile system which creates a ring of processes, with a token passed around this ring. The names of the channels opened by name restrictions* (ν **left0**) *and* (ν ***right***) *denote the processes of the ring. The first part of the system describes the ring creation. The first pro-*

$$
\begin{array}{l}
((\nu\ \mathrm{make})(\nu\ \mathrm{server})(\nu\ \mathrm{port}) \\
((*\mathrm{make}?^{1}[](\nu\ \mathit{address})(\nu\ \mathit{request}) \\
\qquad\qquad (\\
\qquad\qquad (*\mathit{address}?^{2}[]\mathrm{server}!^{3}[\mathit{address}, \mathit{request}]) \\
\qquad\qquad | \\
\qquad\qquad \mathit{address}!^{4}[] \\
\qquad\qquad | \\
\qquad\qquad \mathrm{make}!^{5}[] \\
\qquad\qquad)) \\
| \\
(*\mathrm{server}?^{6}[\mathit{email}, \mathit{data}](\nu\ \mathit{deal}) \\
\qquad\qquad (\\
\qquad\qquad \mathrm{port}?^{7}[](\mathit{deal}!^{8}[\mathit{data}] \mid \mathit{deal}?^{9}[\mathit{rep}](\mathit{email}!^{10}[\mathit{rep}] \mid \mathrm{port}!^{11}[])) \\
\qquad\qquad \oplus \\
\qquad\qquad \mathit{email}!^{12}[] \\
\qquad\qquad)) \\
| \mathrm{port}!^{13}[] \\
| \mathrm{port}!^{14}[] \\
| \mathrm{port}!^{15}[] \\
| \mathrm{make}!^{16}[]) \\
)
\end{array}
$$

Fig. 2. An *ftp*-server

$$
\begin{array}{l}
((\nu\ \mathrm{make})(\nu\ \mathrm{mon})(\nu\ \mathrm{left0}) \\
(\\
\ ((*\mathrm{make}?^{1}[\mathit{left}](\nu\ \mathit{right})(\mathrm{mon}!^{2}[\mathit{left}, \mathit{right}] \mid \mathrm{make}!^{3}[\mathit{right}])) \\
\ | (*\mathrm{make}?^{4}[\mathit{left}](\mathrm{mon}!^{5}[\mathit{left}, \mathrm{left0}])) \\
\ | \mathrm{make}!^{6}[\mathrm{left0}]) \\
| \\
\ ((*\mathrm{mon}?^{7}[\mathit{prev}, \mathit{next}] \\
\qquad\qquad (*\mathit{prev}?^{8}[](\nu\ \mathit{crit})(\mathit{crit}?[]^{9}\mathit{next}!^{10}[] \\
\qquad\qquad\qquad\qquad\qquad\qquad \mid \mathit{crit}!^{11}[]))) \\
\ | \mathrm{left0}!^{12}[])) \\
)
\end{array}
$$

Fig. 3. A ring of processes

cess is created by the restriction (ν **left0**). *An agent* **mon**![***v1***; ***v2***] *denotes a connection between two processes. Then, each time the first resource is replicated, a new process is created and linked to the previous process, which has been passed as an argument of the replication. The second resource replication closes the ring by linking the last created process to the first created process. The second part of the system describes the execution of the processes: an additional resource spawns a resource for each process of the ring. Then a token is put into the ring of processes: the token is denoted by syntactic copies of the agents* ***next***![] *and* **left0**![]. *The name of this agent describes the token location. When the token is available, the corresponding process can replicate its resource, and as a result the process enters its critical section. The critical section is exited when the two agents* ***crit***![] *and* ***crit***?[] *have interacted; the token is then passed to the next process.*

Our analysis can prove both the integrity and the non-exhaustion of this system: it discovers that each time an agent ***mon![left;right]*** *is spawned, either the name* ***left*** *is linked to the channel opened by the restriction* (ν **left0**), *or both names* ***left*** *and* ***right*** *are linked to two channels opened by instances of the restriction* (ν ***right***), *but the channel linked to the name* ***left*** *has been opened by the previous instance of it, which means that a process of the ring can only be connected to either the first one or to the next one; it captures the fact that only one simultaneous instance of the syntactic agent* ***crit***![] *can exist. That is to say, that only one process of the ring can enter its critical section at a given time.*

Remark 3 *The standard semantics is not well suited to express and capture integrity properties, because the link between agent instances and the names of the channel they have opened is not encoded explicitly. For instance, if we think about the example of the* ftp*-server and if we cleverly choose the names of opened channels by indexing them with the instance number of the client resource, we obtain after two sessions of the server a system of the following form*[2] *:*

$$(\nu\ \overline{c})(\nu\ address_1)(\nu\ request_1)(\nu\ address_2)(\nu\ request_2)$$
$$(\mathcal{S}' \mid address_1!^{10}[request_1] \mid address_2!^{10}[request_2]).$$

It appears explicitly that request answers are returned at good addresses. However, we could have chosen the names differently and obtained the following α*-equivalent configuration:*

$$(\nu\ \overline{c})(\nu\ address_2)(\nu\ request_1)(\nu\ address_1)(\nu\ request_2)$$
$$(\mathcal{S}' \mid address_2!^{10}[request_1] \mid address_1!^{10}[request_2])$$

[2] This term only shows explicitly the information we are interested in. The variable $\mathcal{S}'$ denotes the rest of the system and the notation ($\nu\ \overline{c}$) denotes a sequence of restrictions for all implicit names.

in which this property is lost.

The link between the recursive instances of an agent and the names of the channels they have opened could be easily hard-coded: it would be enough to open a new channel named **p** *for each recursive instance of an agent, and then encoding the relation that this instance has opened a given channel name* **n** *by spawning an agent* $\mathbf{has\_opened}!^{\mathbf{i}}[\mathbf{p};\mathbf{n}]$*, where* **has_opened** *is the name of a channel opened at the beginning of the system computation. Nevertheless, it would be very difficult to abstract this relation. All the more so since we are also interested in more complex properties, such as whether two channels have to be opened by two successive instances of an agent. Moreover, we do not know statically which complex properties are required to prove easier ones.*

The purpose of the next section is to design a semantics in which channel name origin is carefully traced and can easily be abstracted.

4 Refining the semantics

The non-standard semantics is a refined semantics which aims at explicitly specifying the links between the channels and the instances of agents which have opened them. Any instance of an agent is identified unambiguously by a marker in order to distinguish that instance from all others. Each time a channel is opened, the name of this channel is tagged with the marker of the agent instance which has opened this channel, so that the origin of channel names is easily traced. Venet, in [31], has presented such a non-standard semantics, but it applies only to a small part of the π-calculus, called *the friendly systems* [22]. In particular, it requires replication guards not to be nested, and the system to be closed. We propose here a new non-standard semantics in order to relax those restrictions.

This section will be organized as follows: we first describe our marker allocation scheme, then we propose a naive fully operational semantics for describing the behavior of closed mobile systems, and we finally improve this semantics in order to reduce the number of computation steps. We deal with open systems in the next section.

4.1 Fresh name allocation

As explained before, α-conversion prevents expressing the link between recursive instances and the names of the channels they have opened. To avoid the use of α-conversion, we propose a name allocation scheme which ensures the

freshness of allocated names. Such a scheme has already been proposed by De Bruijn in [11]. Nevertheless, our requirement is quite different. De Bruijn's naming scheme allows α-conversion to be avoided, in order to simplify some manual proofs. We also expect our scheme to allow us to express some integrity properties. For instance, we would like to express in our semantics the fact that two names are denoting channels which have been opened by the same instance of a given agent. Furthermore, as we want to make static analyses, we want to capture invariants on allocated markers. For that purpose, we want the scheme not to depend on the interleaving order. To solve that problem, we propose to tag each instance of agent by a marker which encodes the history of the replications which have led to its creation. Each name will then be tagged with the marker of the agent which has opened the channel denoted by this name.

We denote by $\mathcal{M}$ the set of all binary trees the leaves of which are all labelled with ε and the nodes of which are labelled with pairs (i, j) where both i and j are in $\mathcal{L}$. The tree having a node labelled with a, a left sibling t_1 and a right one t_2 is denoted by $N(a, t_1, t_2)$. Markers are binary trees in $\mathcal{M}$. Initial agent instances are tagged with the marker ε, while the marker of each new agent instance is calculated recursively from the marker of the agent instances the computation of which has lead to its creation:

- when a computation step does not involve replicating a resource, the marker of the computed agent is just passed to its continuation;
- when a resource is replicated, a new marker is deterministically allocated to the spawned instance: it is given by $N((i, j), id_i, id_j)$ where i is the label of the resource, id_i is the marker of the resource, j is the label of the agent instance which replicates the resource and id_j is the marker of this agent instance.

Marker allocation consistency is expressed by the following proposition:

Proposition 4 *During each computation sequence, two distinct instances of the same agent are always tagged with distinct markers.*

PROOF. The proof of Prop. 4 can be made by induction on the length of the computation sequence. It relies on the fact that each tagged agent instance contains explicitly both the label and the markers of an agent instance which has necessarily been consumed to spawn this instance. □

Moreover, in accordance with the following proposition, we can simplify the shape of the markers without losing marker allocation consistency:

Proposition 5 *Let ϕ_1 and ϕ_2 be the two following functions:*

$$\phi_1 : \begin{cases} \mathcal{M} & \to & (\mathcal{L}^2)^* \\ N(a,b,c) & \mapsto & \phi_1(c).a \\ \varepsilon & \mapsto & \varepsilon \end{cases} \qquad \phi_2 : \begin{cases} \mathcal{M} & \to & \mathcal{L}^* \\ N((i,j),b,c) & \mapsto & \phi_2(c).j \\ \varepsilon & \mapsto & \varepsilon. \end{cases}$$

Marker allocation remains consistent when replacing each marker by its image by ϕ_1 or ϕ_2.

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.

Example 6 *Coming back to the example of the* ftp*-server, with this allocation scheme, the first instance of the client resource will be tagged with the marker $id_1 = N((1,16),\varepsilon,\varepsilon)$, while the second instance will be tagged with the marker $id_2 = N((1,5),\varepsilon,N((1,16),\varepsilon,\varepsilon))$. So that the configuration reached after two sessions of the server will be of the following form:*

$$(\nu\ \overline{\mathbf{c}})(\nu\ address_{id_1})(\nu\ request_{id_1})(\nu\ address_{id_2})(\nu\ request_{id_2})$$
$$(\mathcal{S}' \mid address_{id_1}!^{10}[request_{id_1}] \mid address_{id_2}!^{10}[request_{id_2}]),$$

where the names are indexed by the marker of the threads which have opened the channels they denote. We did not indicate the marker of agent instances which depends on the number of attempts required to establish the connection with the server. It appears explicitly that the names $\boldsymbol{address_{id_i}}$ and $\boldsymbol{request_{id_i}}$ communicated to an instance of the agent labelled **10** *denote two channels opened by the same recursive instance of an agent.*

4.2 Naive semantics

We now propose a fully operational semantics of the π-calculus, in which the channel names are allocated in accordance to the previously proposed fresh name allocation scheme. Furthermore, we get rid of the congruence relation by orienting it, and simulating it by additional operational rules.

4.2.1 Definition

Let us first consider the case of a closed mobile system $\mathcal{S}$ in the π-calculus. The subset of $\mathcal{L}$ used in labeling $\mathcal{S}$ is denoted by $\mathcal{L}_{\text{used}}$. A *non-standard configuration* is a set of thread instances, where a *thread instance* is a triplet composed of a syntactic component, a marker and an environment. The *syntactic component* is a copy of a sub-term of $\mathcal{S}$, the *marker* is calculated at

the creation of the thread and the *environment* specifies the semantic value of each free name of the syntactic component: it maps each free name of the syntactic component to a pair (x, id), where x is a bound name of $\mathcal{S}$ and id is a marker. Intuitively, (x, id) refers to the name of the channel opened by the instance of the restriction $(\nu\ x)$ tagged with the marker id. While threads are running, environments are calculated in order to mimic the standard semantics. The translation of a labelled system $\mathcal{S}$ into a set of initial threads and non-standard computation rules are given in Fig. 4.

Roughly speaking, the initial configuration contains only one thread: the system itself, tagged with the marker of the initial thread, ε. Since the system is closed, its environment is empty. Structural rules mimic and orient the congruence relation. A thread the syntactic component of which is composed of two concurrent agents can be replaced by the two corresponding threads. Name restriction consists in opening a channel denoted by a fresh name, and binding the corresponding variable to this name in the environment of the continuation. The fresh name is obtained by tagging the name used in the restriction by the marker of the agent which has opened the corresponding channel. A thread the syntactic component of which is the empty agent can be removed. Choice rules mimic choice reduction rules. A thread the syntactic component of which is a choice between two agents can be replaced by a thread corresponding to one of these agents. Communication rules mimic communication reduction rules. The synchronization condition is checked in the environment of the communicating threads. Name passing is described by explicit substitution in environments. In the case that a resource is replicated, a fresh marker is inferred in accordance with our marker allocation scheme.

Example 7 *We consider the following system:*

$$(\nu\ a)(*a?^1[]((\nu\ b)(b!^2[b]0 \mid a!^3[]0)) \mid a!^4[]0),$$

and propose a computation sequence for it in the naive non-standard semantics in Fig. 5. The initial state C_0 is just a single thread the syntactic component of which is the system, the marker of which is ε and the environment of which is empty. The first computation step $C_0 \xrightarrow{\varepsilon}_{\mathrm{n}} C_1$ consists in opening a new channel, named (a, ε), since it is opened using the restriction $(\nu\ a)$ of a thread the marker of which is ε. The second computation step $C_1 \xrightarrow{\varepsilon}_{\mathrm{n}} C_2$ consists in decomposing the thread into two concurrent threads; thus the marker of the single thread is just passed to both threads. The third computation step $C_2 \xrightarrow{(1,4)}_{\mathrm{n}} C_3$ is a communication between the two threads. Since the first one is a resource, it is still available after the communication. The thread which has sent the message is consumed and new threads, corresponding to the continuation of the communicating threads are spawned. The continuation of the resource is tagged with a new marker $N((1,4),\varepsilon,\varepsilon)$ obtained from both the labels and the markers of both communicating threads, while the marker

$$\mathcal{C}_0^{\mathrm{n}}(\mathcal{S}) = (\mathcal{S}, \varepsilon, \emptyset)$$

(a) Initial configuration

$$\begin{aligned}
C \cup \{(P \mid Q, id, E)\} &\xrightarrow{\varepsilon}_{\mathrm{n}} C \cup \{(P, id, E_{|fn(P)}); (Q, id, E_{|fn(Q)})\} \\
C \cup \{((\nu\, x)P, id, E)\} &\xrightarrow{\varepsilon}_{\mathrm{n}} C \cup \{(P, id, E[x \to (x, id)]_{|fn(P)})\} \\
C \cup \{(\mathbf{0}, id, E)\} &\xrightarrow{\varepsilon}_{\mathrm{n}} C
\end{aligned}$$

(b) Structural rules

$$\begin{aligned}
C \cup \{P \oplus Q, id, E\} &\xrightarrow{\oplus}_{\mathrm{n}} C \cup \{(P, id, E_{|fn(P)})\} \\
C \cup \{P \oplus Q, id, E\} &\xrightarrow{\oplus}_{\mathrm{n}} C \cup \{(Q, id, E_{|fn(Q)})\}
\end{aligned}$$

(c) Choice rules

$$\frac{E_?(y) = E_!(x)}{C \cup \left\{ \begin{array}{l} (y?^i[\overline{y}]P, id_?, E_?); \\ (x!^j[\overline{x}]Q, id_!, E_!) \end{array} \right\} \xrightarrow{(i,j)}_{\mathrm{n}} C \cup \left\{ \begin{array}{l} (P, id_?, E_?[\overline{y} \to E_!(\overline{x})]_{|fn(P)}); \\ (Q, id_!, E_{!|fn(Q)}) \end{array} \right\}}$$

$$\frac{E_?(y) = E_!(x),\ id_* = N((i,j), id_?, id_!)}{C \cup \left\{ \begin{array}{l} (*y?^i[\overline{y}]P, id_?, E_?); \\ (x!^j[\overline{x}]Q, id_!, E_!) \end{array} \right\} \xrightarrow{(i,j)}_{\mathrm{n}} C \cup \left\{ \begin{array}{l} (*y?^i[\overline{y}]P, id_?, E_?); \\ (P, id_*, E_?[\overline{y} \to E_!(\overline{x})]_{|fn(P)}); \\ (Q, id_!, E_{!|fn(Q)}) \end{array} \right\}}$$

(d) Communication rules

Fig. 4. Naive semantics

$$C_0 \xrightarrow{\varepsilon}_{\mathrm{n}} C_1 \xrightarrow{\varepsilon}_{\mathrm{n}} C_2 \xrightarrow{(1,4)}_{\mathrm{n}} C_3 \xrightarrow{\varepsilon}_{\mathrm{n}} C_4 \xrightarrow{\varepsilon}_{\mathrm{n}} C_5 \xrightarrow{\varepsilon}_{\mathrm{n}} C_6 \xrightarrow{(1,3)}_{\mathrm{n}} C_7 \longrightarrow^{*}_{\mathrm{n}} C_8$$

where

$$C_0 = \{((\nu\ \mathrm{a})((*\mathrm{a}?^1[](\nu\ b)(b!^2[b]0 \mid \mathrm{a}!^3[]0)) \mid \mathrm{a}!^4[]0), \varepsilon, \emptyset)\}$$

$$C_1 = \{((*\mathrm{a}?^1[](\nu\ b)(b!^2[b]0 \mid \mathrm{a}!^3[]0)) \mid \mathrm{a}!^4[]0, \varepsilon, [\mathrm{a} \mapsto (\mathrm{a}, \varepsilon)])\}$$

$$C_2 = \left\{ \begin{array}{l} (*\mathrm{a}?^1[](\nu\ b)(b!^2[b]0 \mid \mathrm{a}!^3[]0), \varepsilon, [\mathrm{a} \mapsto (\mathrm{a}, \varepsilon)]); \\ (\mathrm{a}!^4[]0, \varepsilon, [\mathrm{a} \mapsto (\mathrm{a}, \varepsilon)]) \end{array} \right\}$$

$$C_3 = \left\{ \begin{array}{l} (*\mathrm{a}?^1[](\nu\ b)(b!^2[b]0 \mid \mathrm{a}!^3[]0), \varepsilon, [\mathrm{a} \mapsto (\mathrm{a}, \varepsilon)]); \\ ((\nu\ b)(b!^2[b]0 \mid \mathrm{a}!^3[]0), N((1,4), \varepsilon, \varepsilon), [\mathrm{a} \mapsto (\mathrm{a}, \varepsilon)]); \\ (0, \varepsilon, \emptyset) \end{array} \right\}$$

$$C_4 = \left\{ \begin{array}{l} (*\mathrm{a}?^1[](\nu\ b)(b!^2[b]0 \mid \mathrm{a}!^3[]0), \varepsilon, [\mathrm{a} \mapsto (\mathrm{a}, \varepsilon)]); \\ ((\nu\ b)(b!^2[b]0 \mid \mathrm{a}!^3[]0), N((1,4), \varepsilon, \varepsilon), [\mathrm{a} \mapsto (\mathrm{a}, \varepsilon)]) \end{array} \right\}$$

$$C_5 = \left\{ \begin{array}{l} (*\mathrm{a}?^1[](\nu\ b)(b!^2[b]0 \mid \mathrm{a}!^3[]0), \varepsilon, [\mathrm{a} \mapsto (\mathrm{a}, \varepsilon)]); \\ ((b!^2[b]0 \mid \mathrm{a}!^3[]0), N((1,4), \varepsilon, \varepsilon), [\mathrm{a} \mapsto (\mathrm{a}, \varepsilon), b \mapsto (b, N((1,4), \varepsilon, \varepsilon))]) \end{array} \right\}$$

$$C_6 = \left\{ \begin{array}{l} (*\mathrm{a}?^1[](\nu\ b)(b!^2[b]0 \mid \mathrm{a}!^3[]0), \varepsilon, [\mathrm{a} \mapsto (\mathrm{a}, \varepsilon)]); \\ (b!^2[b]0, N((1,4), \varepsilon, \varepsilon), [b \mapsto (b, N((1,4), \varepsilon, \varepsilon))]); \\ (\mathrm{a}!^3[]0, N((1,4), \varepsilon, \varepsilon), [\mathrm{a} \mapsto (\mathrm{a}, \varepsilon)]) \end{array} \right\}$$

$$C_7 = \left\{ \begin{array}{l} (*\mathrm{a}?^1[](\nu\ b)(b!^2[b]0 \mid \mathrm{a}!^3[]0), \varepsilon, [\mathrm{a} \mapsto (\mathrm{a}, \varepsilon)]); \\ (b!^2[b]0, N((1,4), \varepsilon, \varepsilon), [b \mapsto (b, N((1,4), \varepsilon, \varepsilon))]); \\ ((\nu\ b)(b!^2[b]0 \mid \mathrm{a}!^3[]0), N((1,3), \varepsilon, N((1,4), \varepsilon, \varepsilon)), [\mathrm{a} \mapsto (\mathrm{a}, \varepsilon)]); \\ (0, \varepsilon, \emptyset) \end{array} \right\}$$

$$C_8 = \left\{ \begin{array}{l} (*\mathrm{a}?^1[](\nu\ b)(b!^2[b]0 \mid \mathrm{a}!^3[]0), \varepsilon, [\mathrm{a} \mapsto (\mathrm{a}, \varepsilon)]); \\ (b!^2[b]0, N((1,4), \varepsilon, \varepsilon), [b \mapsto (b, N((1,4), \varepsilon, \varepsilon))]); \\ (b!^2[b]0, N((1,3), \varepsilon, N((1,4), \varepsilon, \varepsilon)), [b \mapsto (b, N((1,3), \varepsilon, N((1,4), \varepsilon, \varepsilon)))]); \\ (\mathrm{a}!^3[]0, N((1,3), \varepsilon, N((1,4), \varepsilon, \varepsilon)), [\mathrm{a} \mapsto (\mathrm{a}, \varepsilon)]) \end{array} \right\}$$

Fig. 5. A computation sequence in the naive semantics

of the thread which has sent the message is just passed to its continuation. The next computation step, $C_3 \xrightarrow{\varepsilon}_{\mathrm{n}} C_4$, is a garbage collection: it consists in removing the thread corresponding to the empty agent. The fifth computation step $C_4 \xrightarrow{\varepsilon}_{\mathrm{n}} C_5$ opens a channel. Its name is given by $(b, N((1,4),\varepsilon,\varepsilon))$ since it is opened by the restriction $(\nu\ b)$ of a thread the marker of which is $N((1,4),\varepsilon,\varepsilon)$. Then in the computation step $C_5 \xrightarrow{\varepsilon}_{\mathrm{n}} C_6$ a single thread is cut into two concurrent threads, which allows us to go on with the recursion: the computation step $C_6 \xrightarrow{(1,3)}_{\mathrm{n}} C_7$ allows the system to spawn another instance of the resource with the fresh marker $N((1,3),\varepsilon,N((1,4),\varepsilon,\varepsilon))$. There is no confusion between recursive instances and the names of the channels they have opened: we can notice that in each thread corresponding to the agent labelled $b!^2[b]$, the marker of the thread and the marker of the name communicated to the variable b are the same. □

4.2.2 Correspondence

The correspondence between the standard and the naive non-standard semantics is established by a translation function Π. We define the translation $\Pi(C)$ of a non-standard configuration C as follows:

$$\Pi(C) = (\nu\ c_1)...(\nu\ c_k)(\overline{E}(t_1)| \ ... \ |\overline{E}(t_l))$$

where $\{c_i \mid i \in [\![1;k]\!]\} = \{E(x) \mid (P, id, E) \in C,\ x \in fn(P)\}$, $C = \{t_i \mid i \in [\![1;l]\!]\}$ and $\overline{E}$ is the function which substitutes each free name of an agent by its image in the environment. The system $\Pi(C)$ is well defined thanks to associativity, commutativity, and swapping rules. We have also assumed [3] that $\mathcal{N} \times \mathcal{M}$ was a subset of $\mathcal{N}$.

The standard and the non-standard semantics are in *weak bisimulation*, as expressed by the following theorem:

Theorem 8 *We have $\mathcal{S} = \Pi(\mathcal{C}_0^{\mathrm{n}}(\mathcal{S}))$, and for any non-standard configurations C and for any word $u \in (\mathcal{L}^2 \cup \{\varepsilon; \oplus\})^*$ such that $\mathcal{C}_0^{\mathrm{n}}(\mathcal{S}) \xrightarrow{u}_{\mathrm{n}}^* C$, we have:*

(1) $C \xrightarrow{\varepsilon}_{\mathrm{n}} C' \implies \Pi(C) \equiv \Pi(C')$;

(2) $\forall \lambda \in \mathcal{L}^2 \cup \{\oplus\},\ C \xrightarrow{\lambda}_{\mathrm{n}} C' \implies \Pi(C) \xrightarrow{\lambda} \Pi(C')$;

(3) $\forall \lambda \in \mathcal{L}^2 \cup \{\oplus\},\ \Pi(C) \xrightarrow{\lambda} P \implies \exists D,\ \exists E,\ \begin{cases} C \xrightarrow{\varepsilon}_{\mathrm{n}}^* D \xrightarrow{\lambda}_{\mathrm{n}} E \\ \Pi(E) \equiv P. \end{cases}$

The proof of Thm. 8 is shown in appendix A.

[3] We consider in fact that a tagged name is a name, since there exists a bijection between the set of the tagged names and the set of the names.

We now propose to reduce the number of reduction steps in order to make analysis design easier, and also to obtain a more efficient analysis. We will first consider a semantics in where all structural steps are automatically performed. Then we will introduce another semantics in where choice steps are also dealt in the same way.

4.3.1 Strongly bisimilar semantics

The binary relation $\stackrel{\varepsilon}{\longrightarrow}_{\mathrm{n}}$ is nœtherian and locally confluent. So, it is confluent [12], and we can define its limit $\Downarrow$ as follows:

$$a \Downarrow b \textit{ if and only if } a \stackrel{\varepsilon}{\longrightarrow}_{\mathrm{n}}^{*} b \textit{ and } \nexists c, b \stackrel{\varepsilon}{\longrightarrow}_{\mathrm{n}} c.$$

We now express explicitly this limit by designing an extraction function β^{i} which performs all structural rules in parallel. The definition of β^{i} is given in Fig. 6.

$$\begin{aligned}
\beta^{\mathrm{i}}((\nu\ n)P, id, E) &= \beta^{\mathrm{i}}\left(P, id, (E[n \mapsto (n, id)])\right) \\
\beta^{\mathrm{i}}(\mathbf{0}, id, E) &= \emptyset \\
\beta^{\mathrm{i}}(P \mid Q, id, E) &= \beta^{\mathrm{i}}(P, id, E) \cup \beta^{\mathrm{i}}(Q, id, E) \\
\beta^{\mathrm{i}}(P \ \oplus\ Q, id, E) &= \{(P \ \oplus\ Q, id, E_{|fn(P \oplus Q)})\} \\
\beta^{\mathrm{i}}(aP, id, E) &= \{(aP, id, E_{|fn(aP)})\}
\end{aligned}$$

Fig. 6. Extraction function

Proposition 9 *For any non-standard configuration C, we have $C \Downarrow \bigcup_{t \in C} \beta^{\mathrm{i}}(t)$.*

The proof of Prop. 9 is shown in the appendix B.

We define the intermediate semantics as the transition system $(\{\mathcal{C}_0^{\mathrm{i}}(\mathcal{S})\}, \longrightarrow_{\mathrm{i}})$, where the set of the initial states $\{\mathcal{C}_0^{\mathrm{i}}(\mathcal{S})\}$ and the computation rule $\longrightarrow_{\mathrm{i}}$ are given as follows:

- $\mathcal{C}_0^{\mathrm{i}}(\mathcal{S}) = \beta^{\mathrm{i}}(\mathcal{C}_0^{\mathrm{n}}(\mathcal{S}))$,
- $\forall \lambda \in (\mathcal{L}^2 \cup \{\oplus\}), a \stackrel{\lambda}{\longrightarrow}_{\mathrm{i}} b$ *if and only if* $\exists c,\ a \stackrel{\lambda}{\longrightarrow}_{\mathrm{n}} c$ *and* $c \Downarrow b$.

Example 10 *We come back to the previously given system:*

$$(\nu\ a)(*a?^1[]((\nu\ b)(b!^2[b]0 \mid a!^3[]0)) \mid a!^4[]0),$$

$$D_0 \xrightarrow{(1,4)}_{\mathrm{i}} D_1 \xrightarrow{(1,3)}_{\mathrm{i}} D_2$$

where

$$D_0 = \left\{ \begin{array}{l} (*\mathrm{a}?^1[](\nu\ b)(b!^2[b]0 \mid \mathrm{a}!^3[]0), \varepsilon, [\mathrm{a} \mapsto (\mathrm{a}, \varepsilon)]); \\ (\mathrm{a}!^4[]0, \varepsilon, [\mathrm{a} \mapsto (\mathrm{a}, \varepsilon)]) \end{array} \right\}$$

$$D_1 = \left\{ \begin{array}{l} (*\mathrm{a}?^1[](\nu\ b)(b!^2[b]0 \mid \mathrm{a}!^3[]0), \varepsilon, [\mathrm{a} \mapsto (\mathrm{a}, \varepsilon)]); \\ (b!^2[b]0, N((1,4), \varepsilon, \varepsilon), [b \mapsto (b, N((1,4), \varepsilon, \varepsilon))]) \\ (\mathrm{a}!^3[]0, N((1,4), \varepsilon, \varepsilon), [\mathrm{a} \mapsto (\mathrm{a}, \varepsilon)]) \end{array} \right\}$$

$$D_2 = \left\{ \begin{array}{l} (*\mathrm{a}?^1[](\nu\ b)(b!^2[b]0 \mid \mathrm{a}!^3[]0), \varepsilon, [\mathrm{a} \mapsto (\mathrm{a}, \varepsilon)]); \\ (b!^2[b]0, N((1,4), \varepsilon, \varepsilon), [b \mapsto (b, N((1,4), \varepsilon, \varepsilon))]) \\ (b!^2[b]0, N((1,3), \varepsilon, N((1,4), \varepsilon, \varepsilon)), [b \mapsto (b, N((1,3), \varepsilon, N((1,4), \varepsilon, \varepsilon)))]) \\ (\mathrm{a}!^3[]0, N((1,3), \varepsilon, N((1,4), \varepsilon, \varepsilon)), [\mathrm{a} \mapsto (\mathrm{a}, \varepsilon)]) \end{array} \right\}$$

Fig. 7. A computation sequence in the intermediate semantics

and give a computation sequence for it in Fig. 7. It appears explicitly that structural computation steps are not described anymore. They are all performed implicitly at the beginning of the system, and after each communication or choice computation step: this way, the initial state D_0 is equal to the state C_2 while the whole computation sequence $C_0 \xrightarrow{\varepsilon}_{\mathrm{n}} C_1 \xrightarrow{\varepsilon}_{\mathrm{n}} C_2$ becomes implicit, then the single computation step $D_1 \xrightarrow{(1,4)}_{\mathrm{i}} D_2$ encodes the computation sequence $C_2 \xrightarrow{(1,4)}_{\mathrm{n}} C_3 \xrightarrow{\varepsilon}_{\mathrm{n}} C_4 \xrightarrow{\varepsilon}_{\mathrm{n}} C_5 \xrightarrow{\varepsilon}_{\mathrm{n}} C_6$, and the single computation step $D_2 \xrightarrow{(1,3)}_{\mathrm{i}} D_3$ encodes the computation sequence $C_6 \xrightarrow{(1,3)}_{\mathrm{n}} C_7 \longrightarrow^*_{\mathrm{n}} C_8$.

The standard and the intermediate semantics are in *strong bisimilation* (up to $\equiv$), as expressed by the following theorem:

Theorem 11 *We have $\mathcal{S} \equiv \Pi(\mathcal{C}^{\mathrm{i}}_0(\mathcal{S}))$, and for all non-standard configurations C and for all word $u \in (\mathcal{L}^2 \cup \{\varepsilon; \oplus\})^*$ such that $\mathcal{C}^{\mathrm{i}}_0(\mathcal{S}) \xrightarrow{u}{}^*_{\mathrm{i}} C$, we have:*

(1) $\forall \lambda \in \mathcal{L}^2 \cup \{\oplus\},\ C \xrightarrow{\lambda}_{\mathrm{i}} C' \implies \Pi(C) \xrightarrow{\lambda} \Pi(C')$;

(2) $\forall \lambda \in \mathcal{L}^2 \cup \{\oplus\},\ \Pi(C) \xrightarrow{\lambda} P \implies \exists D,\ \left\{ \begin{array}{l} C \xrightarrow{\lambda}_{\mathrm{i}} D \\ \Pi(D) \equiv P. \end{array} \right.$

The proof of Thm. 11 is shown in appendix B.

4.3.2 Efficient semantics

We propose to focus on communication rules and also to factor choice rules. To do this, we have to restrict the set of the traces: to make things easier, we propose to consider only the traces in which communications are delayed until no choices can be made. [4]

We denote by $\dashrightarrow$ the binary relation $\xrightarrow{\varepsilon}_{\mathrm{n}} \cup \xrightarrow{\oplus}_{\mathrm{n}}$. The reduction relation $\dashrightarrow$ is nœtherian, but not locally confluent. We define the relation $\Longrightarrow$ as follows:

$$a \Longrightarrow b \text{ if and only if } a \dashrightarrow^{*} b \text{ and } \nexists c,\ b \dashrightarrow c.$$

Since $\dashrightarrow$ is not confluent, $\Longrightarrow$ is not deterministic. We now express explicitly its action by designing an extraction function β which computes the set of all the successors of a given configuration. The definition of β is given in Fig. 8(a).

Proposition 12 *For any non standard configuration C, we have:*

$$\{b \mid C \Longrightarrow b\} = \left\{ \bigcup Cont_t \mid \forall t \in C,\ Cont_t \in \beta(t) \right\}.$$

The proof of Prop. 12 is shown in appendix C.

Intuitively, β gives the set of all the choices when spawning a continuation. To spawn a continuation for a thread the syntactic component of which is a choice $(P \oplus Q)$, we either spawn a continuation choice for P or a continuation choice for Q. Spawning a continuation for a thread the syntactic component of which is a parallel composition $(P \mid Q)$, consists in choosing a continuation for P, choosing a continuation for Q, and spawning concurrently these two continuations.

We now define the efficient semantics as the transition system $(\mathcal{C}_0^{\mathrm{e}}(\mathcal{S}), \longrightarrow_{\mathrm{e}})$:

- $\mathcal{C}_0^{\mathrm{e}}(\mathcal{S}) = \beta(\mathcal{S}, \varepsilon, \emptyset)$
- $\forall \lambda \in \mathcal{L}^2, a \xrightarrow{\lambda}_{\mathrm{e}} b$ *if and only if* $\exists c,\ a \xrightarrow{\lambda}_{\mathrm{n}} c$ *and* $c \Longrightarrow b$.

An explicit definition of $(\mathcal{C}_0^{\mathrm{e}}(\mathcal{S}), \longrightarrow_{\mathrm{e}})$ is given in Fig. 8.

The following theorem establishes the correspondence between the standard and the efficient semantics:

Theorem 13 *For any initial non-standard configuration $C_0 \in \mathcal{C}_0^{\mathrm{e}}(\mathcal{S})$, there*

[4] We have also considered in [14, Sect. 3.3] restricting the set of the traces to those for which choices are made only when necessary, which comes down considering external choices instead of internal ones.

$$
\begin{aligned}
\beta((\nu\ x)P, id, E) &= \beta\left(P, id, (E[x \mapsto (x, id)])\right) \\
\beta(\mathbf{0}, id, E) &= \{\emptyset\} \\
\beta(P \ \oplus \ Q, id, E) &= \beta(P, id, E) \cup \beta(Q, id, E) \\
\beta(P \mid Q, id, E) &= \{A \cup B \mid A \in \beta(P, id, E),\ B \in \beta(Q, id, E)\} \\
\beta(aP, id, E) &= \{\{(aP, id, E_{|fn(aP)})\}\}
\end{aligned}
$$

(a) Extraction function

$$
\mathcal{C}_0^{\mathrm{e}}(\mathcal{S}) = \beta(\mathcal{S}, \varepsilon, \emptyset)
$$

(b) Initial configurations

$$
\frac{\begin{cases} E_?(y) = E_!(x), \\ Ct_P \in \beta(P, id_?, E_?[y_i \mapsto E_!(x_i)]), \\ Ct_Q \in \beta(Q, id_!, E_!) \end{cases}}{C \cup \left\{ \begin{array}{l} (y?^i[\overline{y}]P, id_?, E_?), \\ (x!^j[\overline{x}]Q, id_!, E_!) \end{array} \right\} \xrightarrow{(i,j)}_{\mathrm{e}} (C \cup Ct_P \cup Ct_Q)}
$$

$$
\frac{\begin{cases} E_*(y) = E_!(x), \\ Ct_P \in \beta(P, N((i,j), id_*, id_!), E_*[y_i \mapsto E_!(x_i)]), \\ Ct_Q \in \beta(Q, id_!, E_!) \end{cases}}{C \cup \left\{ \begin{array}{l} (*y?^i[\overline{y}]P, id_*, E_*), \\ (x!^j[\overline{x}]Q, id_!, E_!) \end{array} \right\} \xrightarrow{(i,j)}_{\mathrm{e}} \Big(C \cup \{(*y?^i[\overline{y}]P, id_*, E_*)\} \cup Ct_P \cup Ct_Q\Big)}
$$

(c) Communication rules

Fig. 8. Efficient semantics

exists $k \in \mathbb{N}$ *such that* $\mathcal{S} \xrightarrow{\oplus^k}{}^* \Pi(C_0)$ *and for all non-standard configurations* C *and for all word* $u \in (\mathcal{L}^2)^*$ *such that* $C_0 \xrightarrow{u}{}^*_{\mathrm{e}} C$*, we have:*

(1) $\forall \lambda \in \mathcal{L}^2,\ C \xrightarrow{\lambda}_{\mathrm{e}} C' \implies \exists k \in \mathbb{N},\ \exists P,\ \Pi(C) \xrightarrow{\lambda} P \xrightarrow{\oplus^k}{}^* \Pi(C')$*;*

(2) $\forall \lambda \in \mathcal{L}^2,\ \Pi(C) \xrightarrow{\lambda} P \implies \exists D,\ \begin{cases} C \xrightarrow{\lambda}_{\mathrm{e}} D \\ \text{and} \begin{cases} \exists k > 0,\ P \xrightarrow{\oplus^k}{}^+ \Pi(D) \\ \text{or } P \equiv \Pi(D). \end{cases} \end{cases}$

The proof of Thm. 13 is shown in appendix C.

Remark 14 *There is no bisimulation between the standard and the efficient semantics. We have restricted both the set of traces to those where choices are always performed before communications, and the set of states to those where no choice appears at the top level. Therefore, these restrictions do not change the properties we want to observe on mobile systems.*

5 Context-free semantics

5.1 Construction

We now extend our non-standard semantics to open systems. An open system $\mathcal{S}$ is a part of a bigger closed system, the rest of which is called its context. The context is a set of agents, concurrently running with $\mathcal{S}$. We represent this context by the set of channel names it shares with the system $\mathcal{S}$, we call such names the *unsafe names*, and approximate its behavior as if it was an intruder who was able to compose any possible agent working on these channel names. An interaction between the system $\mathcal{S}$ and its context may only consist in a communication between an agent $p_{\mathcal{S}}$ of the first and an agent p_{cont} of the second, via a channel the name of which is unsafe. This communication is called *spying* when p_{cont} is the receiver, and *spoiling* when p_{cont} is the message sender. When spying, the context listens to obtain new channel names which become unsafe. When spoiling, the context may pass any name to $\mathcal{S}$, either an unsafe name denoting a channel opened by a binder of the system $\mathcal{S}$ or a name denoting a channel opened by the context itself; as a consequence, we have to introduce an infinite set of unsafe names for the channels that the context may have opened. Eventually, spoiling may lead to the replication of a resource, which requires the allocation of an unambiguous marker, otherwise the consistency of the semantics would not be preserved.

Since α-conversion allows us to choose the names of the new channels opened by the context, we may assume that those channels have been declared by

recursive instances of a single agent. By choosing $cont_?, cont_! \in \mathcal{L} \setminus \mathcal{L}_{\text{used}}$ and $ext \in \mathcal{N} \setminus bn(\mathcal{S})$, such channels will be seen as if they had been created by the restriction $(\nu\ ext)$ of a recursive instance of an agent the marker of which is t_n, where t_n is recursively defined as follows:

$$\begin{cases} t_0 = N((cont_?, \text{cont}_!), \varepsilon, \varepsilon) \\ t_{n+1} = N((cont_?,\ cont_!), \varepsilon, t_n). \end{cases}$$

Thus the set of the names of the channel opened by the context can be chosen as the set $\{(ext, t_n) \mid n \in \mathbb{N}\}$, which we denote by en. We also assume that all spoiling messages are recursive instances of a single agent the first action of which is labelled with $0 \in \mathcal{L} \setminus (\mathcal{L}_{\text{used}} \cup \{cont_?; cont_!\})$. The coherence of our semantics mainly relies on the fact that during a computation sequence, there cannot be two different instances of a single agent with the same marker. We guarantee this property by associating to each spoiling message a fresh marker in the set $\{t_n \mid n \in \mathbb{N}\}$.

A non-standard configuration is now a triple (C, U, F), where C is a set of threads, U is a set of channel names, and F is a set of markers. The set C contains the running threads. The set U contains all the name (a, id) such that the channel opened by the restriction $(\nu\ a)$ of the agent instance tagged with the marker id is unsafe. The set F contains fresh markers which have not been used as markers for spoiling message. At the beginning of the system, free names have to be chosen among the set of initial unsafe names. We make no assumptions about the past of the system, so that distinct free names can be bound to the same unsafe name. This is especially useful when analyzing an instance of a resource without any knowledge of the relations between channel names that have been communicated to it.

The transition relation $\looparrowright$ takes into account the computations inside the mobile system $\mathcal{S}$, as well as the computations involving the system $\mathcal{S}$ and its context. Initial non-standard configurations and computation rules are given in Fig. 9.

5.2 Coherence

We propose to establish a relation between the non-standard semantics of closed and open systems. Let $\mathcal{S}_{\text{I}}(x_1, ..., x_n)$ be an open system the set of the free names of which are exactly the set $\{x_i \mid i \in [\![1; n]\!]\}$. We want to construct a projection function Π_τ, such that:

- any non-standard computation sequence τ of a closed system of the form $(\nu\ c_1)...(\nu\ c_k)(\mathcal{S}_{\text{I}}(c_{i_1}, ..., c_{i_n}) \mid \mathcal{S}_{\text{c}}(c_{j_1}, ...c_{j_l}))$, is mapped to a non-standard

$$\mathcal{C}_0^o(\mathcal{S}) = \left\{ (Ct, en, \{t_n \mid n \in \mathbb{N}\}) \;\middle|\; \begin{array}{l} Ct \in \beta(\mathcal{S}, \varepsilon, E), \\ E \in (fn(\mathcal{S}) \to en) \end{array} \right\}$$

(a) Initial configurations

$$\frac{C \xrightarrow{\lambda}_{\mathrm{e}} C'}{(C, U, F) \overset{\lambda}{\hookrightarrow} (C', U, F)}$$

(b) Internal interactions

$$\frac{t = (x!^j[x_1, ...x_n]P, id, E),\ E(x) \in U,\ Ct \in \beta(P, id, E)}{(C \cup \{t\}, U, F) \overset{(0,j)}{\hookrightarrow} (C \cup Ct, U \cup \{E(x_k) \mid k \in [\![1; n]\!]\}, F)}$$

(c) Spied communication

$$\frac{\begin{cases} t = (y?^i[y_1, ...y_n]P, id, E), \\ E(y) \in U,\ c_1, ..., c_n \in U, \\ Ct \in \beta(P, id, E[y_k \mapsto c_k]) \end{cases}}{(C \cup \{t\}, U, F) \overset{(i,0)}{\hookrightarrow} (C \cup Ct, U, F)}$$

(d) Spoiled communication

$$\frac{\begin{cases} t = (*y?^i[y_1, ...y_n]P, id, E), \\ E(y), c_1, ..., c_n \in U, \\ id_! \in F, \\ id_* = N((i, 0), id, id_!), \\ Ct \in \beta(P, id_*, E[y_k \mapsto c_k]) \end{cases}}{(C \cup \{t\}, U, F) \overset{(i,0)}{\hookrightarrow} (C \cup \{t\} \cup Ct, U, F \setminus \{id_!\})}$$

(e) Spoiled resource replication

Fig. 9. Context-free non-standard semantics

computation sequence of the open system $\mathcal{S}_I$;

- reciprocally for any non-standard computation sequence τ' of the open system $\mathcal{S}_\mathrm{I}$, there exists a computation sequence τ of a closed system of the form $(\nu\ c_1)...(\nu\ c_k)\ (\mathcal{S}_\mathrm{I}(c_{i_1},...,c_{i_n}) \mid \mathcal{S}_\mathrm{c}(c_{j_1},...c_{j_l}))$, such that $\tau' = \Pi_\tau(\tau)$.

Let $\mathcal{L}_\mathrm{I} \subseteq \mathcal{L}$ be the subset of the labels occurring in $\mathcal{S}_\mathrm{I}$ and $\mathcal{N}_\mathrm{I} \subseteq \mathcal{N}$ be the subset of the names used in name restrictions of $\mathcal{S}_\mathrm{I}$. The function *lab* maps each syntactic component beginning with an action to the label of this action. We introduce two one-to-one functions in order to interpret names and agents created by the context of $\mathcal{S}_\mathrm{I}$: let $\Phi_\mathcal{M}$ be a one-to-one map from the set $(\mathcal{L} \times \mathcal{M})$ into the set $\{t_n \mid n \in \mathbb{N}\}$, and $\Phi_\mathcal{N}$ be a one-to-one function from the set $(\mathcal{N} \times \mathcal{M})$ into the set *en*. We now define the projection $\Pi_\tau(\mathcal{S}_\mathrm{I}, \Phi_\mathcal{M}, \Phi_\mathcal{N})$ which transforms each computation sequence of a closed mobile system $\mathcal{S} = (\nu\ \overline{c})(\mathcal{S}_\mathrm{I}(c_{i_1},...,c_{i_n}) \mid \mathcal{S}_\mathrm{c}(c_{j_1},...c_{j_l}))$ into a computation sequence of the part $\mathcal{S}_\mathrm{I}$ of the system $\mathcal{S}$. We assume without any loss of generality that $\mathit{fn}(\mathcal{S}_\mathrm{I}) = \{c_{i_k} \mid k \in [\![1; n]\!]\}$, and that no name occurs twice as an argument of a name binder.

We first project each syntactic component label, by replacing each label which does not occur in $\mathcal{S}_\mathrm{I}$ by the unique label of the context, that is to say the 0 label.

Definition 15 *The projection $\Pi_\mathrm{l}(l)$ of a syntactic component label $l \in \mathcal{L}$ is defined as follows:*

$$\Pi_\mathrm{l}(l) = \begin{cases} l & \textit{if } l \in \mathcal{L}_\mathrm{I} \\ 0 & \textit{otherwise.} \end{cases}$$

We then apply the syntactic component label projection pair-wise on transition labels.

Definition 16 *The projection $\Pi_\lambda(i,j)$ of transition label $(i,j) \in \mathcal{L}^2$ is defined as follows:*

$$\Pi_\lambda(i,j) = (\Pi_\mathrm{l}(i), \Pi_\mathrm{l}(j)).$$

Next, we project the instance markers of the syntactic components of $\mathcal{S}_\mathrm{I}$. In such a marker, only the right sibling can be the marker of a syntactic component of the context, as we know that the replicated resource necessarily belongs to $\mathcal{S}_\mathrm{I}$. When a resource is replicated by a message of the context, we replace its syntactic component, and calculate a coherent marker according to $\Phi_\mathcal{M}$.

Definition 17 *Marker projection is defined as follows:*

$$\Pi_{\mathcal{M}}^{\Phi_{\mathcal{M}}} : \begin{cases} N((i,j),t_1,t_2) \mapsto N(\Pi_\lambda(i,j), \Pi_{\mathcal{M}}^{\Phi_{\mathcal{M}}}(t_1), \Pi_{\mathcal{M}}^{\Phi_{\mathcal{M}}}(t_2)) & \text{if } j \in \mathcal{L}_{\mathrm{I}} \\ N((i,j),t_1,t_2) \mapsto N(\Pi_\lambda(i,j), \Pi_{\mathcal{M}}^{\Phi_{\mathcal{M}}}(t_1), \Phi_{\mathcal{M}}(j,t_2))) & \text{if } j \notin \mathcal{L}_{\mathrm{I}} \\ \varepsilon \mapsto \varepsilon. \end{cases}$$

We now project channel names. For a channel having been opened by a name restriction of $\mathcal{S}_{\mathrm{I}}$, we just project the marker. For those having been opened by the context, we replace the name restriction by the unique restriction (ν *ext*) of the context, and calculate the coherent marker according to $\Phi_{\mathcal{N}}$.

Definition 18 *Channel name projection is defined as follows:*

$$\Pi_{\mathcal{N}}^{\Phi_{\mathcal{M}},\Phi_{\mathcal{N}}}(x, id_x) = \begin{cases} (x, \Pi_{\mathcal{M}}^{\Phi_{\mathcal{M}}}(id_x)) & \text{if } x \in \mathcal{N}_{\mathrm{I}} \\ \Phi_{\mathcal{N}}(x, id_x) & \text{otherwise.} \end{cases}$$

We can easily project an instance of a syntactic component of $\mathcal{S}_{\mathrm{I}}$ by projecting its marker, and each channel inside its environment.

Definition 19 *Thread projection is defined as follows:*

$$\Pi_{\mathrm{t}}^{\Phi_{\mathcal{M}},\Phi_{\mathcal{N}}}(P, id, E) = (P, \Pi_{\mathcal{M}}^{\Phi_{\mathcal{M}}}(id), [x \mapsto \Pi_{\mathcal{N}}^{\Phi_{\mathcal{M}},\Phi_{\mathcal{N}}}(E(x))]).$$

Then, we can project a configuration by projecting all the threads the syntactic component of which is a sub-term of $\mathcal{S}_{\mathrm{I}}$, and removing the other threads:

Definition 20 *Configuration projection is defined as follows:*

$$\Pi_{\mathrm{C}}^{\Phi_{\mathcal{M}},\Phi_{\mathcal{N}}}(C) = \{\Pi_{\mathrm{t}}^{\Phi_{\mathcal{M}},\Phi_{\mathcal{N}}}(P, id, E) \mid (P, id, E) \in C,\ lab(P) \in \mathcal{L}_{\mathrm{I}}\}.$$

We can now define the projection of a computation sequence. At each computation step, we obtain the set of the threads by projecting all the instances of syntactic components of $\mathcal{S}_{\mathrm{I}}$, and throwing away instances of syntactic components of the context. Unfortunately, the set of unsafe names, and the set of fresh markers, cannot be constructed without any knowledge of the previous computation steps, so we construct them incrementally: at each computation step, we insert spied names into the set of unsafe names, and remove the used markers from the set of fresh markers. We also ignore all computation steps only involving the context.

Definition 21 *Computation sequence projection is then defined as follows: Let* $\tau = C_0 \xrightarrow{\lambda_1}_{\mathrm{e}} \ldots \xrightarrow{\lambda_n}_{\mathrm{e}} C_n$ *be a non-standard computation sequence, with*

$C_0 \in \mathcal{C}_0^{\mathrm{e}}(\mathcal{S})$. We define the projection of τ, $\Pi_\tau(\mathcal{S}_{\mathrm{I}}, \Phi_{\mathcal{M}}, \Phi_{\mathcal{N}})(\tau)$ as the non-standard computation sequence:

$$(A_0, U_0, F_0) \overset{\Pi_\lambda(\lambda_{a_1})}{\twoheadrightarrow} \ ... \ \overset{\Pi_\lambda(\lambda_{a_p})}{\twoheadrightarrow} \ (A_p, U_p, F_p)$$

of the open system $\mathcal{S}_{\mathrm{I}}$, where

- *$a_1, ..., a_p$ is the strictly ascending sequence of the elements of the set $\{i \in [\![1; n]\!] \mid \lambda_i \in \mathcal{L}^2 \setminus (\mathcal{L} \setminus \mathcal{L}_{\mathrm{I}})^2\}$;*
- *the initial configuration (A_0, U_0, F_0) is the following triple:*

$$(\Pi_{\mathrm{C}}^{\Phi_{\mathcal{M}}, \Phi_{\mathcal{N}}}(C_0), en, \{t_n \mid n \in \mathbb{N}\});$$

- *for $k \in [\![1; p]\!]$, the configuration (A_k, U_k, F_k) is then defined as follows:*
  - $A_k = \Pi_{\mathrm{C}}^{\Phi_{\mathcal{M}}, \Phi_{\mathcal{N}}}(C_{a_k})$,
  - $U_k = \begin{cases} U_{k-1} & \text{if } fst(\lambda_{a_k}) \in \mathcal{L}_{\mathrm{I}}, \\ U_{k-1} \cup \{\Pi_{\mathcal{N}}^{\Phi_{\mathcal{M}}, \Phi_{\mathcal{N}}}(E(x_r)) \mid r \in [\![1; n]\!]\} & \text{otherwise,} \end{cases}$
    where, in the last case, $t = (x!^j[x_1, ..., x_n]P, id, E)$ denotes the unique thread in $C_{a_{k-1}} \setminus C_{a_k}$ which matches this notation;
  - $F_k = \begin{cases} F_{k-1} \text{ if } snd(\lambda_{a_k}) \in \mathcal{L}_{\mathrm{I}} \text{ or if } \xrightarrow{\lambda_{a_k}}_{\mathrm{e}} \text{ is not a resource replication,} \\ F_{k-1} \setminus \{\Phi_{\mathcal{M}}(snd(\lambda_{a_k}), id) \mid (P, id, E) \in C_{a_k - 1} \setminus C_{a_k}\} \text{ otherwise.} \end{cases}$

We establish the relation between the non-standard semantics of closed and open systems, as follows:

Theorem 22 (Soundness) *Let $\tau = C_0...C_n$ be a non-standard computation sequence of the following closed system:*

$$(\nu\ c_1)...(\nu\ c_k)(\mathcal{S}_{\mathrm{I}}(c_{i_1}, ..., c_{i_n}) \mid \mathcal{S}_{\mathrm{c}}(c_{j_1}, ...c_{j_l})),$$

with $C_0 \in \mathcal{C}_0^{\mathrm{e}}(\mathcal{S})$. Then $\Pi_\tau(\mathcal{S}_{\mathrm{I}}, \Phi_{\mathcal{M}}, \Phi_{\mathcal{N}})(\tau) = (A_0, U_0, F_0)...(A_p, U_p, F_p)$ is a non-standard computation sequence of the open system $\mathcal{S}_{\mathrm{I}}$ and $(A_0, U_0, F_0) \in \mathcal{C}_0^{\mathrm{o}}(\mathcal{S})$.

Theorem 23 (Completeness) *Let τ' be the non-standard computation sequence of an open system $\mathcal{S}_{\mathrm{I}}$, that we denote by:*

$$(C_0, U_0, F_0) \overset{(i_1, j_1)}{\twoheadrightarrow} \ ... \ \overset{(i_n, j_n)}{\twoheadrightarrow} \ (C_n, U_n, F_n),$$

where $(C_0, U_0, F_0) \in \mathcal{C}_0^{\mathrm{o}}(\mathcal{S}_{\mathrm{I}})$.

Then there exists:

- *a closed system $\mathcal{S}_* = (\nu\ \overline{c})(\mathcal{S}_{\mathrm{I}}(c_{i_1}, ..., c_{i_n}) \mid \mathcal{S}_{\mathrm{c}}(c_{j_1}, ...c_{j_l}))$,*

- *two one-to-one functions* $\Phi_{\mathcal{N}}$ *and* $\Phi_{\mathcal{M}}$,
- *a non-standard computation sequence* τ *of the system* $\mathcal{S}_*$,

such that $\Pi_\tau(\mathcal{S}_{\mathrm{I}}, \Phi_{\mathcal{M}}, \Phi_{\mathcal{N}})(\tau) = \tau'$.

Soundness is ensured by construction. Completeness relies on the existence of a most general context which can be used in simulating any context. It is given in Fig. 10. It uses a global channel, named **unsafe** via which unsafe names are sent an arbitrary amount of times. It is made off four kinds of resources. The resource **new** opens a new unsafe channel; the resource **repli** is used to replicates the information that a channel name is unsafe, so that a context may use each unsafe name an arbitrary number of time; the resource $\mathbf{spy}_k$ collects an unsafe channel c, and receive through this channel a message of arity k; the resource $\mathbf{spoil}_k$ collects an unsafe channel c, and k unsafe names, and sends the k names through the channel c. Resources $\mathbf{spy}_0$ and $\mathbf{spoil}_0$ only enforce some synchronization with the system. In Fig. 10, n denotes the greatest arity of the messages occurring in the system part we analyze. Thus the closed system can be chosen as the following one:

$$\mathcal{S}_* = (\nu\ \mathrm{unsafe})(\nu\ x_1)...(\nu\ x_p)$$
$$(\mathrm{unsafe}![x_1]\ |...|\ \mathrm{unsafe}![x_p]\ |\ \mathcal{S}_{\mathrm{I}}(x_{i_1}, ..., x_{i_n})\ |\ \mathcal{S}_{\mathrm{c}}(\mathrm{unsafe})).$$

The non-standard computation sequence is obtained by mimicking spied and spoiled computation steps in $\mathcal{S}_{\mathrm{c}}$. The full proof of Thms. 22 and 23 is shown in appendix D.

$\mathcal{S}_{\mathrm{c}} = (\nu\ \mathrm{new})$
(**new** | **repli**
| $\mathbf{spy}_0$ |...| $\mathbf{spy}_n$
| $\mathbf{spoil}_0$ | ... | $\mathbf{spoil}_n$
| new![]
)

where
- **new** := $*\mathrm{new}?[]((\nu\ \mathit{channel})(\mathrm{unsafe}![\mathit{channel}]\ |\ \mathrm{new}![]))$
- **repli** := $*\mathrm{unsafe}?[x](\mathrm{unsafe}![x]\ |\ \mathrm{unsafe}![x])$
- $\mathbf{spy}_i$:= $*\mathrm{unsafe}?[c]c?[y_1, ..., y_i](\mathrm{unsafe}![y_1]\ |\ ...\ |\ \mathrm{unsafe}![y_i])$
- $\mathbf{spoil}_i$:= $*\mathrm{unsafe}?[c]\mathrm{unsafe}?[x_1]...\mathrm{unsafe}?[x_i]c![x_1, ..., x_i]$

Fig. 10. The most general context

---

6 Abstract Interpretation

Abstract Interpretation is a theory of the approximation of semantics. It formalizes the idea that the semantics can be more or less precise according to the considered level of observation. In static analysis, abstract interpretation is used to derive a decidable semantics from a concrete one. Because of the upper-approximation, the result is not complete: this means that not all the properties of the program are discovered, nevertheless, the result is sound: this means that all the captured properties are correct.

In this section, we introduce a generic abstraction to approximate the behavior of a mobile system. It could apply indeed to any transition system. This abstraction does not yet depend on the abstracted properties: they are left as a parameter of our analysis. Hence, our framework is highly generic, and we can make a reduced product between several analyses. We will use this framework to derive an analysis of the control flow in Sect. 7, and an analysis of the occurrence number of the agents in Sect. 8.

6.1 Generic abstraction

We denote by $\mathcal{C}$ the set of the non-standard configurations and by Σ the set of the transition labels. We are actually interested in the set $\mathcal{C}(\mathcal{S})$ of all the configurations that a system may take during a finite sequence of computation steps. This is given by its collecting semantics, which is defined in [6]. It can be expressed as the least fixpoint of a $\cup$-complete endomorphism $\mathbb{F}$ on the complete lattice $\wp(\Sigma^* \times \mathcal{C})$ defined as follows:

$$\mathbb{F}(X) = (\{\varepsilon\} \times \mathcal{C}_0^{\mathrm{o}}(\mathcal{S})) \cup \left\{ (u.\lambda, C') \;\middle|\; \exists C \in \mathcal{C},\ (u, C) \in X \text{ and } C \overset{\lambda}{\nrightarrow} C' \right\}.$$

This least fixpoint is usually not decidable, so we use the Abstract Interpretation framework [7] to compute a sound – but not necessarily complete – approximation of it. More precisely, we use the relaxed version of Abstract Interpretation [8], in which the abstract domain is not supposed to be complete under least upper bound; furthermore, no abstraction function is required.

Definition 24 *An abstraction is a tuple* $(\mathcal{C}^\sharp, \sqsubseteq^\sharp, \bigsqcup^\sharp, \bot^\sharp, \gamma, C_0^\sharp, \rightsquigarrow, \nabla)$ *which satisfies:*

(1) $(\mathcal{C}^\sharp, \sqsubseteq^\sharp)$ *is a pre-order;*
(2) $\bigsqcup^\sharp : \wp_{finite}(\mathcal{C}^\sharp) \to \mathcal{C}^\sharp$ *is such that* $\forall A^\sharp \in \wp_{finite}(\mathcal{C}^\sharp),\ \forall a^\sharp \in A^\sharp,\ a^\sharp \sqsubseteq^\sharp \bigsqcup^\sharp(A^\sharp)$*;*
(3) $\bot^\sharp \in \mathcal{C}^\sharp$ *satisfies* $\forall a^\sharp \in \mathcal{C}^\sharp,\ \bot^\sharp \sqsubseteq^\sharp a^\sharp$*;*

(4) $\gamma : \mathcal{C}^\sharp \to \wp(\Sigma^* \times \mathcal{C})$ *is a monotonic map;*
(5) $C_0^\sharp \in \mathcal{C}^\sharp$ *is such that* $\{\varepsilon\} \times \mathcal{C}_0^o(\mathcal{S}) \subseteq \gamma(C_0^\sharp)$*;*
(6) $\rightsquigarrow \in \wp(\mathcal{C}^\sharp \times \Sigma \times \mathcal{C}^\sharp)$ *is an abstract deterministic labelled transition relation over* $\mathcal{C}^\sharp$ *such that :* $\forall C^\sharp \in \mathcal{C}^\sharp,\ \forall (u, C) \in \gamma(C^\sharp),\ \forall \lambda \in \Sigma,\ \forall \overline{C} \in \mathcal{C}$*,*

$$C \overset{\lambda}{\hookrightarrow} \overline{C} \Longrightarrow \exists \overline{C}^\sharp \in \mathcal{C}^\sharp,\ (C^\sharp \overset{\lambda}{\rightsquigarrow} \overline{C}^\sharp \text{ and } (u.\lambda, \overline{C}) \in \gamma(\overline{C}^\sharp));$$

(7) $\nabla : \mathcal{C}^\sharp \times \mathcal{C}^\sharp \to \mathcal{C}^\sharp$ *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.

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

Given an abstraction $(\mathcal{C}^\sharp, \sqsubseteq^\sharp, \bigsqcup^\sharp, \bot^\sharp, \gamma, C_0^\sharp, \rightsquigarrow, \nabla)$, the abstract counterpart $\mathbb{F}^\sharp$ to $\mathbb{F}$ defined as:

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

satisfies the soundness condition $\forall C^\sharp \in \mathcal{C}^\sharp,\ \mathbb{F} \circ \gamma(C^\sharp) \subseteq \gamma \circ \mathbb{F}^\sharp(C^\sharp)$. Since $\mathbb{F}$ is monotonic, we have $\forall n \in \mathbb{N}, \forall C^\sharp \in \mathcal{C}^\sharp,\ \mathbb{F}^n \circ \gamma(C^\sharp) \subseteq \gamma \circ \mathbb{F}^{\sharp n}(C^\sharp)$. On the other hand, since $\mathbb{F}$ is a $\cup$-complete endomorphism, we have $lfp_\emptyset\ \mathbb{F} = \bigcup\limits_{n\in\mathbb{N}} \mathbb{F}^n(\emptyset)$. As a consequence, we obtain the soundness of our analysis:

Theorem 25 $lfp_\emptyset\ \mathbb{F} \subseteq \bigcup\limits_{n\in\mathbb{N}} \left[\gamma \circ \mathbb{F}^{\sharp n}\right] (\bot^\sharp)$.

Following [6], we compute a sound and decidable approximation of our abstract semantics by using the widening operator ∇:

Theorem 26 *The abstract iteration [8, 9] of $\mathbb{F}^\sharp$ defined as follows:*

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

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

6.2 Abstraction algebra

Our framework is highly extensible. We now give some operations over abstractions to compose them.

Proposition 27 (Product) *Let $(\mathcal{C}_1^\sharp, \sqsubseteq_1^\sharp, \bigsqcup_1^\sharp, \bot_1^\sharp, \gamma_1, C_{0_1}^\sharp, \rightsquigarrow_1, \nabla_1)$ and $(\mathcal{C}_2^\sharp, \sqsubseteq_2^\sharp, \bigsqcup_2^\sharp, \bot_2^\sharp, \gamma_2, C_{0_2}^\sharp, \rightsquigarrow_2, \nabla_2)$ be two abstractions.*

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;*
- $\gamma \ : \begin{cases} \mathcal{C}^\sharp \to \wp(\Sigma^* \times \mathcal{C}) \\ (a_1^\sharp, a_2^\sharp) \mapsto \gamma_1(a_1^\sharp) \cap \gamma_2(a_2^\sharp); \end{cases}$
- $C_0^\sharp = (C_{0_1}^\sharp, C_{0_2}^\sharp)$;
- $\rightsquigarrow$ *is defined by:*
  $(a_1, a_2) \overset{\lambda}{\rightsquigarrow} (b_1, b_2)$ *if and only if* $a_1 \overset{\lambda}{\rightsquigarrow}_1 b_1$ *and* $a_2 \overset{\lambda}{\rightsquigarrow}_2 b_2$

is also an abstraction.

Proposition 28 (Reduction) *Let $(\mathcal{C}^\sharp, \sqsubseteq^\sharp, \bigsqcup^\sharp, \bot^\sharp, \gamma, C_0^\sharp, \rightsquigarrow, \nabla)$ be an abstraction, and ρ be a reduction operator*[5] *$\rho : \mathcal{C}^\sharp \to \mathcal{C}^\sharp$ which satisfies:*

$$\forall a^\sharp \in \mathcal{C}^\sharp,\ \gamma(a^\sharp) \subseteq \gamma(\rho(a^\sharp)).$$

The following tuple $(\mathcal{C}^\sharp, \sqsubseteq^\sharp, \bigsqcup^\sharp, \bot^\sharp, \gamma, C_{0_\rho}^\sharp, \rightsquigarrow_\rho, \nabla)$ where

- $C_{0_\rho}^\sharp = \rho(C_0^\sharp)$;
- $\rightsquigarrow_\rho$ *is defined by:*
  $a \rightsquigarrow_\rho c$ *if and only if there exists $b \in \mathcal{C}^\sharp$, such that $\rho(a) \rightsquigarrow b$ and $c = \rho(b)$*

[5] ρ simplifies the properties obtained in the abstract domain.

is also an abstraction.

Furthermore, ρ can also be used to simplify the final result of the abstract iteration.

The proof of Prop. 27 and Prop. 28 is shown in appendix E.

7 Control flow analysis

We now use this framework to derive an analysis of the control flow. This analysis requires an abstract domain for describing sets of marker tuples. In the first subsection, we introduce the generic control flow analysis, independently of the chosen abstract domain. In the next subsection, we propose three different choices of abstract domains, so that we get three different analyses.

7.1 Generic control flow analysis

We propose to compute a description of both the potential interactions between the agents of a mobile system, and the potential interactions between its agents and a hostile context. For that purpose, we will compute for each syntactic name restriction an approximation of the set of the syntactic agents that may receive the name of a channel opened by an instance of this name restriction. As we want a non-uniform description, we will also compute a comparison between the markers of the agents which have opened channels and the ones of the agents to which the names of those channels have been communicated. Due to the approximation, some of the discovered interactions may be ineffective. But the analyzer detects all the interactions. So, if the analyzer does not detect an interaction between two components, there cannot be any interaction between them.

The main difficulty is to synthesize comparisons between markers throughout computation steps. We use the marker of the agent instance to which channel names are communicated as a pivot to synthesize the comparison between the markers of the agent instances which have opened these channels. Furthermore, we use synchronization conditions on the channel names via which the communication is performed to establish a comparison between the markers of all the involved agent instances. Our main strategy is easy: we first gather all the information we have about the marker pairs (this means we will abstract sets of marker tuples). Then, synchronization conditions give equality relations between some tuple components. If these equality relations are satisfiable, the abstract computation step is enabled and we compute, for each

new agent instance, the comparison between the marker of the new instance and the markers of the instances which had opened the channels.

We introduce for each $n \in \mathbb{N}$ an abstract pre-order $(Id_n^\sharp, \sqsubseteq_n)$ to represent sets of n-tuples of thread markers. Thus, each $Id_n^\sharp$ is related to $\wp(\mathcal{M}^n)$ by a monotonic concretization function γ_n. We introduce some abstract primitives to handle these domains: a representation of the empty set $\bot_n^\sharp$, a representation of the initial marker $\varepsilon^\sharp$, an abstraction of the set of the context markers $t^\sharp$, an abstract union $\bigsqcup_n$, a widening operator ∇_n, an associative abstract concatenation $\bullet$ that is a correct abstraction of tuple concatenation, an abstract join *sync* 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 replicating a resource. These primitives shall satisfy the following properties:

(1) $\gamma_n(\bot_n^\sharp) = \emptyset$;
(2) $\varepsilon \in \gamma_1(\varepsilon^\sharp)$;
(3) $\{t_n \mid n \in \mathbb{N}\} \subseteq \gamma_1(t^\sharp)$;
(4) $\forall A \in \wp_{\text{finite}}(Id_n^\sharp),\ \bigsqcup_n(A) \in Id_n^\sharp$ and $\forall a^\sharp \in A,\ a^\sharp \sqsubseteq_n \bigsqcup_n(A)$;
(5) $\nabla_n\ :\ (Id_n^\sharp)^2 \to Id_n^\sharp$ is a widening operator;
(6) $\forall a \in Id_n^\sharp, b \in Id_m^\sharp,\ (a \bullet b) \in Id_{n+m}^\sharp$ and
$$\left\{ (id_i)_{i\in[\![1;n+m]\!]} \;\middle|\; \begin{array}{l} (id_i)_{i\in[\![1;n]\!]} \in \gamma_n(a^\sharp), \\ (id_{i+n})_{i\in[\![1;m]\!]} \in \gamma_m(b^\sharp) \end{array} \right\} \subseteq \gamma_{n+m}(a \bullet b);$$
(7) $\forall a^\sharp \in Id_n^\sharp,\ \forall A \in \wp([\![1;n]\!]^2),\ sync(A, a^\sharp) \in Id_n^\sharp$ and
$\{(id_i)_{i\in[\![1;n]\!]} \mid (id_i) \in \gamma_n(a^\sharp),\ \forall (k,l) \in A,\ id_k = id_l\} \subseteq \gamma_n(sync(A, a^\sharp))$;
(8) $\forall a^\sharp \in Id_n^\sharp, \forall p \in \mathbb{N},\ \forall (s_k)_{k\in[\![1;p]\!]} \in [\![1;n]\!]^{[\![1;p]\!]}$ one-to-one sequence,
$\underset{(s_k)}{\Pi}(a^\sharp) \in Id_p^\sharp$ and $\{(id_{s_k})_{k\in[\![1;p]\!]} \mid (id_i)_{i\in[\![1;n]\!]} \in \gamma_n(a^\sharp)\} \subseteq \gamma_p(\underset{(s_k)}{\Pi}(a^\sharp))$;
(9) $\forall a^\sharp \in Id_3^\sharp,\ \underset{(i,j)}{push}(a^\sharp) \in Id_2^\sharp$ and
$\{(N((i,j), id_1, id_2), id_3) \mid (id_1, id_2, id_3) \in \gamma_3(a^\sharp)\} \subseteq \gamma_2(\underset{(i,j)}{push}(a^\sharp))$.

Moreover we define the operator $dpush \in (Id_1^\sharp \to Id_2^\sharp)$ by:

$$\forall a \in Id_1^\sharp,\ dpush(a) = sync(\{(1,2)\}, a \bullet a).$$

The operator *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 syntactic components of $\mathcal{S}$. We introduce $\nu n(\mathcal{S})$ as the set of the syntactic names used in name restrictions, including the name *ext*. We then denote by I the set $\{(p, x, y) \mid p \in \mathcal{P},\ x \in fn(p),\ y \in \nu n(\mathcal{S})\}$. I is the set of the possible interactions between agents of $\mathcal{S}$. In the case that

$y \neq ext$, the interaction (p, x, y) denotes the fact that the variable x of an instance of the agent p may be linked to the name of a channel opened by the restriction $(\nu\ y)$; the interaction (p, x, ext) denotes the fact that the variable x of an instance of the agent p may be linked to the name of a channel opened by the context.

In this way, our abstraction $(\mathcal{C}^\sharp, \sqsubseteq^\sharp, \bigsqcup^\sharp, \bot^\sharp, \gamma, C_0^\sharp, \rightsquigarrow_{\text{cfa}}, \nabla)$ is defined as follows:

- $\mathcal{C}^\sharp = ((I \to id_2^\sharp) \times (\nu n(\mathcal{S}) \to id_1^\sharp))$;
- $\sqsubseteq^\sharp, \bigsqcup^\sharp, \nabla$ are defined component-wise and pair-wise respectively from $(\sqsubseteq_2, \sqsubseteq_1)$, $(\bigsqcup_2, \bigsqcup_1)$ and (∇_2, ∇_1);
- $\bot^\sharp$ is given by the pair of functions which relate any elements respectively to $\bot_2^\sharp$ and $\bot_1^\sharp$;
- the concretization $\gamma(f, g)$ is the set $(\Sigma^* \times A)$ where A is the set of configurations (C, U, F) which satisfy:

(1) $(P, id_P, E) \in C, x \in fn(P), E(x) = (y, id_x) \Longrightarrow (id_P, id_x) \in \gamma_2(f(P, x, y))$,
(2) $(x, id) \in U \Longrightarrow id \in \gamma_1(g(x))$;

- $C_0^\sharp$ and $\rightsquigarrow_{\text{cfa}}$ are given in Figs. 12-14.

An abstract configuration is given by two functions. The first one maps each interaction (P, x, y) to the description of the set of marker pairs (id_P, id_x) such that, in the instance of P tagged with the marker id_P, the variable x may be linked to the name of the channel opened by the instance of the name restriction $(\nu\ y)$ tagged with the marker id_x. The second one maps each name x to a description of the set of the markers id such that a channel opened by the instance of the name restriction $(\nu\ x)$ tagged with the marker id may be communicated to the context. Abstract transition rules just mimic the non-standard ones. Their definition uses an abstract extraction function $\beta^\sharp$ defined in Fig. 11.

$$
\begin{aligned}
\beta^\sharp((\nu\ n)P, id^\sharp, E^\sharp) &= \beta^\sharp(P, id^\sharp, E^\sharp[(n, n) \mapsto dpush(id^\sharp)]) \\
\beta^\sharp(\mathbf{0}, id^\sharp, E^\sharp) &= \bot^\sharp \\
\beta^\sharp(P \ \oplus\ Q, id^\sharp, E^\sharp) &= \sqcup^\sharp\{\beta^\sharp(P, id^\sharp, E^\sharp); \beta^\sharp(Q, id^\sharp, E^\sharp)\} \\
\beta^\sharp(P \mid Q, id^\sharp, E^\sharp) &= \sqcup^\sharp\{\beta^\sharp(P, id^\sharp, E^\sharp); \beta^\sharp(Q, id^\sharp, E^\sharp)\} \\
\beta^\sharp(aP, id^\sharp, E^\sharp) &= ([(aP, m, n) \mapsto E^\sharp(m, n), \forall m \in fn(aP), n \in \nu n(S)], \emptyset)
\end{aligned}
$$

Fig. 11. Abstract semantics: abstract extraction

The function $\beta^\sharp$ is an abstract counterpart to β. It calculates all the interactions obtained by spawning a continuation having a description of both its marker and its environment: given $P \in \mathcal{P}$, $id^\sharp \in Id_1^\sharp$ and $E^\sharp \in ((bn(\mathcal{S}) \times \nu n(\mathcal{S})) \to Id_2^\sharp)$, the element $\beta^\sharp(P, id^\sharp, E^\sharp)$ belongs to the abstract domain $\mathcal{C}^\sharp$ and satisfies the following soundness property:

$$C_0^\sharp = \sqcup^\sharp\{\beta^\sharp(\mathcal{S}, \epsilon^\sharp, [(n, ext) \mapsto \epsilon^\sharp \bullet t^\sharp, \forall n \in fn(\mathcal{S})]); \{\emptyset, [ext \mapsto t^\sharp]\}\}$$

Fig. 12. Abstract semantics: initial configurations

Let $(f^\sharp_{\text{can}}, f^\sharp_{\text{esc}})$ be an abstract configuration,
if there are $\lambda = y?^i[y_1, ..., y_n]P$, $\mu = x!^j[x_1, ..., x_n]Q$,
if $\bigsqcup_4\{A(u) \mid u \in \nu n(\mathcal{S})\} \neq \bot^\sharp_4$,
then $(f^\sharp_{\text{can}}, f^\sharp_{\text{esc}}) \rightsquigarrow_{\text{cfa}} \sqcup^\sharp \{(f^\sharp_{\text{can}}, f^\sharp_{\text{esc}}); \beta^\sharp(P, id^\sharp_P, E^\sharp_P); \beta^\sharp(Q, id^\sharp_Q, E^\sharp_Q)\}$ where

- $A(u) = sync(\{2, 4\}, f^\sharp_{\text{can}}(\lambda, y, u) \bullet f^\sharp_{\text{can}}(\mu, x, u))$,
- $id^\sharp_P = \bigsqcup_1\{\prod_{(1)}(A(u)) \mid u \in \nu n(\mathcal{S})\}$,
- $id^\sharp_Q = \bigsqcup_1\{\prod_{(3)}(A(u)) \mid u \in \nu n(\mathcal{S})\}$,
- $E^\sharp_P = [(m, n) \mapsto \bigsqcup_2\{I_P(m, n, u) \mid u \in \nu n(\mathcal{S})\}, \forall m \in fn(P), n \in \nu n(\mathcal{S})]$
  where $I_P(m, n, u) = \begin{cases} \prod_{(1,6)} sync(\{3, 5\}, A(u) \bullet f^\sharp_{\text{can}}(\mu, x_i, n)) & \text{if } m = y_i \\ \prod_{(1,6)} sync(\{1, 5\}, A(u) \bullet f^\sharp_{\text{can}}(\lambda, m, n)) & \text{otherwise,} \end{cases}$
- $E^\sharp_Q = [(m, n) \mapsto \bigsqcup_2\{I_Q(m, n, u) \mid u \in \nu n(\mathcal{S})\}, \forall m \in fn(Q), n \in \nu n(\mathcal{S})]$
  where $I_Q(m, n, u) = \prod_{(3,6)} sync(\{3, 5\}, A(u) \bullet f^\sharp_{\text{can}}(\mu, m, n))$.

(a) Abstract communication

Let $(f^\sharp_{\text{can}}, f^\sharp_{\text{esc}})$ be an abstract configuration,
if there are $\lambda = *y?^i[y_1, ..., y_n]P$, $\mu = x!^j[x_1, ..., x_n]Q$,
if $\bigsqcup_4\{A(u) \mid u \in \nu n(\mathcal{S})\} \neq \bot^\sharp_4$,
then $(f^\sharp_{\text{can}}, f^\sharp_{\text{esc}}) \rightsquigarrow_{\text{cfa}} \sqcup^\sharp \{(f^\sharp_{\text{can}}, f^\sharp_{\text{esc}}); \beta^\sharp(P, id^\sharp_P, E^\sharp_P); \beta^\sharp(Q, id^\sharp_Q, E^\sharp_Q)\}$ where

- $A(u) = sync(\{2, 4\}, f^\sharp_{\text{can}}(\lambda, y, u) \bullet f^\sharp_{\text{can}}(\mu, x, u))$,
- $id^\sharp_P = \bigsqcup_1\{\prod_{\{1\}}(push_{(i,j)}(\prod_{(1,3,4)}(A(u)))) \mid u \in \nu n(\mathcal{S})\}$,
- $id^\sharp_Q = \bigsqcup_1\{\prod_{\{3\}}(A(u)) \mid u \in \nu n(\mathcal{S})\}$,
- $E^\sharp_P = [(m, n) \mapsto \bigsqcup_2\{I_P(m, n, u) \mid u \in \nu n(\mathcal{S})\}, \forall m \in fn(P), n \in \nu n(\mathcal{S})]$
  where
  $I_P(m, n, u) = \begin{cases} push_{(i,j)}(\prod_{(1,3,6)} sync(\{3, 5\}, A(u) \bullet f^\sharp_{\text{can}}(\mu, x_i, n))) & \text{if } m = y_i \\ push_{(i,j)}(\prod_{(1,3,6)} sync(\{1, 5\}, A(u) \bullet f^\sharp_{\text{can}}(\lambda, m, n))) & \text{otherwise,} \end{cases}$
- $E^\sharp_Q = [(m, n) \mapsto \bigsqcup_2\{I_Q(m, n, u) \mid u \in \nu n(\mathcal{S})\}, \forall m \in fn(Q), n \in \nu n(\mathcal{S})]$
  where $I_Q(m, n, u) = \prod_{(3,6)} sync(\{3, 5\}, A(u) \bullet f^\sharp_{\text{can}}(\mu, m, n))$.

(b) Abstract resource replication

Fig. 13. Abstract semantics: internal interactions

Let $(f^\sharp_{\text{can}}, f^\sharp_{\text{esc}})$ be an abstract configuration,
if there is $\mu = x!^j[x_1, ..., x_n]Q$,
if $\bigsqcup_3\{A(u) \mid u \in \nu n(\mathcal{S})\} \neq \bot^\sharp_3$,
then $(f^\sharp_{\text{can}}, f^\sharp_{\text{esc}}) \rightsquigarrow_{\text{cfa}} \sqcup^\sharp \{(f^\sharp_{\text{can}}, f^\sharp_{\text{esc}}); \beta^\sharp(Q, id^\sharp_Q, E^\sharp_Q); (\emptyset, Escape^\sharp)\}$ where

- $A(u) = sync(\{(2,3)\}, f^\sharp_{\text{can}}(\mu, x, u) \bullet f^\sharp_{\text{esc}}(u))$,
- $id^\sharp_Q = \bigsqcup_1\{\prod_{(1)}(A(u)) \mid u \in \nu n(\mathcal{S})\}$,
- $E^\sharp_Q = [(m,n) \mapsto \bigsqcup_2\{I_Q(m,n,u) \mid u \in bn(\mathcal{S})\}, \forall m \in fn(Q), n \in \nu n(\mathcal{S})]$
  where $I_Q(m,n,u) = \prod_{(1,5)} sync(\{1,4\}, A(u) \bullet f^\sharp_{\text{can}}(\mu, m, n))$,
- $Escape^\sharp = [m \mapsto \mathcal{E}(m),\ \forall m \in \nu n(\mathcal{S})]$
  where $\mathcal{E}(m) = \bigsqcup_1\{\prod_{(5)}(sync(\{(1,4)\}, A(u) \bullet (f^\sharp_{\text{can}}(\mu, x_k, m)))) \mid 1 \leqslant k \leqslant n\}$.

(a) Abstract spied communication

Let $(f^\sharp_{\text{can}}, f^\sharp_{\text{esc}})$ be an abstract configuration,
if there is $\lambda = y?^i[y_1, ..., y_n]P$,
if $\bigsqcup_3\{A(u) \mid u \in \nu n(\mathcal{S})\} \neq \bot^\sharp_3$,
then $(f^\sharp_{\text{can}}, f^\sharp_{\text{esc}}) \rightsquigarrow_{\text{cfa}} \sqcup^\sharp \{(f^\sharp_{\text{can}}, f^\sharp_{\text{esc}}); \beta^\sharp(P, id^\sharp_P, E^\sharp_P)\}$ where

- $A(u) = sync(\{(2,3)\}, f^\sharp_{\text{can}}(\lambda, y, u) \bullet f^\sharp_{\text{esc}}(u))$,
- $id^\sharp_P = \bigsqcup_1\{\prod_{(1)}(A(u)) \mid u \in \nu n(\mathcal{S})\}$,
- $E^\sharp_P = [(m,n) \mapsto \bigsqcup_2\{I_P(m,n,u) \mid u \in \nu n(\mathcal{S})\}, \forall m \in fn(P), n \in \nu n(\mathcal{S})]$
  where $I_P(m,n,u) = \begin{cases} id^\sharp_P \bullet (f^\sharp_{\text{esc}}(n)) & \text{if } m = y_k \\ \prod_{(1,5)} sync(\{1,4\}, A(u) \bullet f^\sharp_{\text{can}}(\lambda, m, n)) & \text{otherwise.} \end{cases}$

(b) Abstract spoiled communication

Let $(f^\sharp_{\text{can}}, f^\sharp_{\text{esc}})$ be an abstract configuration,
if there is $\lambda = *y?^i[y_1, ..., y_n]P$,
if $\bigsqcup_3\{A(u) \mid u \in \nu n\} \neq \bot^\sharp_3$,
then $(f^\sharp_{\text{can}}, f^\sharp_{\text{esc}}) \rightsquigarrow_{\text{cfa}} \sqcup^\sharp \{(f^\sharp_{\text{can}}, f^\sharp_{\text{esc}}); \beta^\sharp(P, id^\sharp_P, E^\sharp_P)\}$ where

- $A(u) = sync(\{(2,3)\}, f^\sharp_{\text{can}}(\lambda, y, u) \bullet f^\sharp_{\text{esc}}(u))$,
- $id^\sharp_P = \bigsqcup_1\{\prod_{(1)}(push_{(i,0)}((\prod_{(1)}(A(u))) \bullet t^\sharp \bullet \top^\sharp_{\mathcal{M}})) \mid u \in \nu n(\mathcal{S})\}$,
- $E^\sharp_P = [(m,n) \mapsto \bigsqcup_2\{I_P(m,n,u) \mid u \in \nu n(\mathcal{S})\}, \forall m \in fn(P), n \in \nu n(\mathcal{S})]$
  where
  $I_P(m,n,u) = \begin{cases} (id^\sharp_P \bullet f^\sharp_{\text{esc}}(n)) & \text{if } m = y_k \\ push_{(i,0)}((\prod_{(1,6,5)} sync(\{1,4\}, A(u) \bullet f^\sharp_{\text{can}}(\lambda, m, n) \bullet t^\sharp))) & \text{otherwise.} \end{cases}$

(c) Abstract spoiled resource replication

Fig. 14. Abstract semantics: external interactions

Proposition 29 $\forall P \in \mathcal{P},\ \forall id \in \gamma_1(id^\sharp),\ \forall E \in (fn(P) \to (\nu n(\mathcal{S}) \times \mathcal{M}))$ *such that* $\forall m \in fn(P),\ \forall n \in \nu n(\mathcal{S}),\ \forall id_n \in \mathcal{M},\ [E(m) = (n, id_n) \implies (id, id_n) \in \gamma_2(E^\sharp(m, n))]$*, we have:*

$$\Sigma^* \times (\beta(P, id, E) \times \{\emptyset\} \times \wp(\{t_n \mid n \in \mathbb{N}\})) \subseteq \gamma(\beta^\sharp(P, id^\sharp, E^\sharp)).$$

The proof of Prop. 29 is shown in appendix F.

We now give some intuition about the abstract transition rules. For the sake of brevity, we focus on abstract communication. Abstract communication between two syntactic components λ and μ simulates all the possible communications between instances of both λ and μ. These communications are first quantified by the channel used by the communication, and more precisely, by the name restriction $(\nu\ u)$ used in opening this channel. So, for each $u \in \nu n(\mathcal{S})$, we compute an abstraction $A(u)$ of the 4-tuples of markers $(id_\lambda, id_u, id_\mu, id'_u)$ such that id_λ, id_μ may be simultaneously the markers of an instance of λ and μ while id_u and id'_u may be the markers of the instance that has opened the channel named u. We obviously enforce the synchronization condition between the second and the fourth components of those tuples, which allows us to detect relations between the first and the third ones. The abstract communication is allowed only if the set of 4-tuples is not empty. In that case, we add the new interactions caused by the continuation of both λ and μ. These continuations are calculated using the abstract extraction $\beta^\sharp$. Continuation marker descriptions are obtained by projecting $A(u)$ onto, respectively, its first and its third components. To calculate abstract environments, we have to deal with interaction communication. An interaction is communicated by abstractly concatenating it with each $A(u)$, enforcing the synchronization with the good agent marker, and then projecting the result in accordance with the marker of the agent it is passed to. New open channels are dealt automatically by the abstract extraction.

Theorem 30 $(\mathcal{C}^\sharp, \sqsubseteq^\sharp, \bigsqcup^\sharp, \bot^\sharp, \gamma, C_0^\sharp, \rightsquigarrow_{\text{cfa}}, \nabla)$ *is an abstraction.*

The proof of Thm. 30 is shown in appendix F.

7.2 Abstract domains

Various domains can be used to instantiate the family of parametric domains $(Id_n^\sharp)_{n\in\mathbb{N}}$, depending on the expected complexity and accuracy. We propose three particular instantiations. The first one abstracts away the information about markers. The result is a uniform control flow analysis. The second one keeps only the equality relations among markers, and gives an analysis which has similar behavior to with group creation [4]. The third one allows algebraic

comparisons of markers which is, to the best of our knowledge, beyond the scope of the analyses previously presented in the literature.

7.2.1 Uniform control flow analysis

Uniform control flow analysis consists in detecting the potential interactions between syntactic components, without keeping any information about markers. For each name restriction $(\nu\ x)$, it will capture an upper-approximation of the set of the syntactic components to which the names of the channels opened by this restriction may be communicated. Such an analysis can easily be obtained by instantiating all the elements of the family $(Id_n^\sharp)_{n\in\mathbb{N}}$ with the lattice $(\{\bot,\top\},\sqsubseteq)$. The domain $\{\bot,\top\}$ is related to $\wp(Id^n)$ by the concretization function γ_n defined by $\gamma_n(\bot)=\emptyset$ and $\gamma_n(\top)=Id^n$.

The abstract primitives are then defined as follows:

- $\varepsilon^\sharp=\top$;
- $t^\sharp=\top$;
- $\forall A\in\wp(\{\bot;\top\}),\ \bigsqcup_n(A)=\begin{cases}\bot & \textit{if } \top\notin A\\ \top & \textit{otherwise};\end{cases}$
- since there is no infinite sequence in $\{\bot;\top\}$ we can use the abstract union as a widening operator;
- $\forall a,b\in\{\bot,\top\},\ (a\bullet b)=\begin{cases}\bot & \textit{if } a=\bot \textit{ or } b=\bot\\ \top & \textit{if } a=\top \textit{ and } b=\top;\end{cases}$
- $\forall a\in\{\bot,\top\},\ \textit{sync}(A,a)=a,\ \prod_X(a)=a \textit{ and } \underset{(i,j)}{\textit{push}}(a)=a.$

The resulting analysis is always at least as precise as the analysis proposed in [1,3]. Nielson *et al.*'s analysis computes the least element of a Moore family defined as the solution set of a constraint system. Since our abstract union is exact, and we do not use widening operator, the result of our abstract semantics is the least fixpoint of the abstract endomorphism induced by $C_0^\sharp$ and $\rightsquigarrow_{\text{cfa}}$. This least fixpoint is also the least element of a Moore family defined as the solution set of a constraint system. Then, comparing the constraints involved in both analyses, it turns out that our constraint system is implied by Nielson *et al.*'s one. So, any solution of Nielson *et al.*'s system is also a solution of our system and the least solution of ours is more precise than Nielson *et al.*'s one. Roughly speaking, Nielson *et al.*'s analysis does not take into account action sequentiality: the constructed system only depends on the syntactic action set. Furthermore, it cannot infer distinct interaction sets for two distinct occurrences of the same channel name.

7.2.2 Confinement

We now focus on the equality relations between markers. This allows us to analyze whether or not the name of a channel can be communicated only to the recursive instance which has opened this channel.

For that purpose, we use a graph-based domain to represent equality relations between components of tuples. Each vertex describes a component; a path between two vertices expresses the fact that the two related components are always equal. We then lift that domain with an extra element in order to represent a non-satisfiable property. We first define the set $\mathcal{G}_n$ of all the undirected graphs 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:

- $\forall \curvearrowright_1,\curvearrowright_2 \in \wp([\![1;n]\!]^2), ([\![1;n]\!],\curvearrowright_1) \sqsubseteq_n ([\![1;n]\!],\curvearrowright_2) \iff \curvearrowright_2 \subseteq \curvearrowright_1$,
- $\forall([\![1;n]\!],\curvearrowright) \in \mathcal{G}_n,\ \gamma_n([\![1;n]\!],\curvearrowright) = \{(id_i)_{i\in[\![1;n]\!]} \mid k \curvearrowright l \implies id_k = id_l\}$;
- we also use $\bigsqcup_n$ as a widening operator since $\mathcal{G}_n$ is height bounded;
- $\varepsilon^\sharp = (\{1\},\emptyset)$;
- $t^\sharp = (\{1\},\emptyset)$;
- $\forall A \in \wp(\mathcal{G}_n),\ \bigsqcup_n(A) = ([\![1;n]\!];\curvearrowright_\cup)$
  where $i \curvearrowright_\cup j \iff \forall([\![1;n]\!],\curvearrowright) \in A, i \curvearrowright^* j$;
- $\forall a = ([\![1;n]\!],\curvearrowright_a) \in \mathcal{G}_n, b = ([\![1;m]\!],\curvearrowright_b) \in \mathcal{G}_m,\ (a \bullet b) = ([\![1;n+m]\!],\curvearrowright_\bullet)$
  where $i \curvearrowright_\bullet j \iff \begin{cases} i \curvearrowright_a j \ \textit{if}\ i,j \in [\![1;n]\!] \\ (i-n) \curvearrowright_b (j-n) \ \textit{if}\ i,j \in [\![n+1;n+m]\!]; \end{cases}$
- $\forall([\![1;n]\!],\curvearrowright) \in \mathcal{G}_n, A \in \wp([\![1;n]\!]^2),\ \mathit{sync}(A,([\![1;n]\!],\curvearrowright)) = ([\![1;n]\!], A \cup \curvearrowright)$;
- $\forall a = ([\![1;n]\!],\curvearrowright_a) \in \mathcal{G}_n,\ \underset{s}{\Pi}(a) = ([\![1;p]\!],\curvearrowright_\Pi)$
  where $s = (s_k)_{k\in[\![1;p]\!]}$ and $i \curvearrowright_\Pi j \iff i,j \in [\![1;p]\!],\ s_i \curvearrowright_a^* s_j$;
- $\forall a \in \mathcal{G}_3,\ \underset{(i,j)}{\mathit{push}}(a) = ([\![1;2]\!],\emptyset)$;

Roughly speaking, the partial order is the opposite of the constraint set inclusion because, the more constraints, the fewer solutions. The concretization of a graph is the set of all the tuples the components of which satisfy the equality relations described by the edges of this graph. The representation of the empty word is just a graph with one vertex. So is the set representation of the context markers. Gathering some abstract elements consists in intersecting the constraint sets they are described with. Abstract concatenation is just a non-relational union of two graphs, after a renaming of the vertices of the second one. Synchronization consists in adding new constraints. Projection consists in restricting the set of the vertices, keeping equality relations on the remaining vertices. When replicating a resource, we know that the new instance marker is fresh, but we can get no information of this fact. Before applying union and projection, we must close graphs, if not we may lose information.

We then lift each $\mathcal{G}_n$ by adding an extra element $\bot_n$. This element $\bot_n$ is the least element of $(\mathcal{G}_n \cup \{\bot_n\})$. The concretization of $\bot_n$ is the empty set. Abstract union is lifted as follows:

$$\bigsqcup_n(\emptyset) = \bot_n$$
$$\bigsqcup_n(A \cup \{\bot_n\}) = \bigsqcup_n(A).$$

All other abstract primitives are lifted to be strict, which means that they return the good element in $(\bot_i)_{i\in\mathbb{N}}$ with respect to their image domain as soon as one of their arguments is in $(\bot_i)_{i\in\mathbb{N}}$.

As in [4], this analysis can only prove that the name of a channel is confined inside the scope of the recursive instance which has opened this channel. It is unable to prove that the name of a channel which first exits the scope of the agent instance which has opened it, can then only be sent back to the recursive instance which has opened it. The main problem is that we can only propagate equality relations. When the name of a channel is communicated to an agent having a distinct marker than the one of the agent which has opened this channel, we have no information anymore. Then, if the channel name is returned to the agent which has previously opened the channel, we cannot infer the right relation. In order to do that we need an algebraic comparison between markers.

7.2.3 Non-uniform analysis with algebraic comparisons

We now propose an abstract domain which deals with abstract algebraic comparisons between markers. Following Prop. 5, we only abstract the right comb of each tree. We choose $m \in \{1; 2\}$, in accordance with the chosen simplification function ϕ_1 or ϕ_2 [6]. We introduce Σ_m as the alphabet $\mathcal{L}^m$ of the letters occurring in markers. We use a reduced product between two abstractions. Our first abstraction consists in abstracting component-wise the shape of the markers associated to threads. The second one infers a comparison between the Parikh vectors of the markers.

7.2.3.1 Regular approximation. We approximate the marker shape in regular languages. For the sake of efficiency, we only use the regular languages which can be described by a set of initial letters, a set of last letters and a succession relation between letters. The result is an efficient abstract domain of languages, the height of which is quadratic in the cardinal of the alphabet Σ_m, and abstract primitives can be computed with a $\mathcal{O}(|\Sigma_m|^2)$ worst-case time cost.

[6] Both functions have been introduced in Sect. 4, Prop. 5.

We introduce the set Id^{Reg} of tuples (i, f, t, b) such that $i, f \in \wp(\Sigma_m)$, $t \in (\Sigma_m \to \wp(\Sigma_m))$ and $b \in \{0; 1\}$. Each element $(i, f, t, b) \in Id^{\mathrm{Reg}}$ is related to a language on Σ_m via a concretization function γ^{Reg} defined as follows:

$$u \in \gamma^{\mathrm{Reg}}(i, f, t, b) \iff \begin{cases} |u| > 0 \implies u_1 \in i \\ |u| > 0 \implies u_{|u|} \in f \\ \forall i \in [\![1; |u|[\![,\ |u|_{i+1} \in t(u_i) \\ |u| = 0 \implies b = 1. \end{cases}$$

Roughly speaking, i is the set of the initial letters of the language words, f is the set of the final letters. The set $t(a)$ is the set of letters which may immediately follow an occurrence of the letter a. The boolean b is equal to 1 if the empty word belongs to the language.

The abstract domain Id^{Reg} is then fitted with a complete lattice structure $(Id^{\mathrm{Reg}}, \bot^{\mathrm{Reg}}, \subseteq^{\mathrm{Reg}}, \bigcap^{\mathrm{Reg}}, \bigcup^{\mathrm{Reg}}, \bot^{\mathrm{Reg}}, \top^{\mathrm{Reg}})$ as follows:

- $\bot^{\mathrm{Reg}} = (\emptyset, \emptyset, [\lambda \mapsto \emptyset], 0)$;
- $\subseteq^{\mathrm{Reg}}$, $\bigcup^{\mathrm{Reg}}$ and $\bigcap^{\mathrm{Reg}}$ are defined component-wise from the usual set operations $\subseteq$, $\bigcup$ and $\bigcap$;
- $\top^{\mathrm{Reg}} = (\Sigma_m, \Sigma_m, [\lambda \mapsto \Sigma_m], 1)$.

Furthermore, the language containing only the empty word is described by $(\emptyset, \emptyset, [\lambda \mapsto \emptyset], 1)$ and is denoted by $\varepsilon^{\mathrm{Reg}}$. At last, we can define the primitive $push^{\mathrm{Reg}} : Id^{\mathrm{Reg}} \times \Sigma_m \to Id^{\mathrm{Reg}}$ which adds a letter at the end of all the words of a language by $push^{\mathrm{Reg}}((i, f, t, b), \lambda) = (i', f', t', b')$ where:

$$\begin{cases} i' = \begin{cases} i \cup \{\lambda\} & \text{if } b = 1 \\ i & \text{otherwise;} \end{cases} \\ f' = \begin{cases} \{\lambda\} & \text{if } b = 1 \text{ or } i \neq \emptyset \\ \emptyset & \text{otherwise;} \end{cases} \\ t' = \left[a \mapsto \begin{cases} t(a) \cup \{\lambda\} & \text{if } a \in f \\ t(a) & \text{otherwise;} \end{cases} \right] \\ b' = 0. \end{cases}$$

The emptiness test is given by the following equivalence:

$$\gamma^{\mathrm{Reg}}(i, f, t, b) \neq \emptyset \iff b = 1 \text{ or } \exists a_0, ..., a_n, \begin{cases} a_0 \in i,\ a_n \in f, \\ \forall k \in [\![0; n[\![, a_{k+1} \in t(a_k). \end{cases}$$

We then introduce the set Id_n^{Reg} of the n-tuples of Id^{Reg}. It is related to $\wp(Id^n)$ by the following concretization function:

$$\gamma_n((A_i)_{i \in [\![1;n]\!]}) = \{(id_i)_{i \in [\![1;n]\!]} \mid \forall i \in [\![1; n]\!],\ \phi_m(id_i) \in \gamma^{\mathrm{Reg}}(A_i)\}.$$

Then the associated abstract primitives are defined as follows:

- $\varepsilon^\sharp = \varepsilon^{\mathrm{Reg}}$;
- $t^\sharp = (\{a_m\}, \{a_m\}, t_m, 0)$,
  where $a_1 = cont_!$, $a_2 = (cont_?, cont_!)$ and $t_m = \begin{cases} x \mapsto \{a_m\} & \text{if } x = a_m \\ x \mapsto \emptyset & \text{otherwise;} \end{cases}$
- $\bigsqcup_n$ applies $\bigcup^{\mathrm{Reg}}$ component-wisely;
- Π (resp. $\bullet$) is the classical projection (resp. concatenation) of tuples;
- $sync(A, Q)_i = \bigcap^{\mathrm{Reg}}\{Q_j \mid (i, j) \text{ in the reflexive and transitive closure of } A\}$;
- since there is no infinite ascending sequences, we define our widening operator ∇_n to be $\bigsqcup^n$;
- $\underset{\lambda}{push}(a, b, c) = push^{\mathrm{Reg}}(b, \lambda)$.

7.2.3.2 Numerical approximation. Our second abstraction captures relational comparisons between the occurrence number of each pattern inside sets of marker pairs. For each $n \in \mathbb{N}$, we introduce a set $\mathcal{V}_n$ of distinct variables $\{x_i^\lambda \mid i \in [\![1; n]\!],\ \lambda \in \Sigma_m\}$. The abstract domain $\wp(\mathbb{N}^{\mathcal{V}_n})$ is related to $\wp(Id^n)$ by the monotonic map γ_n:

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

The power set $\wp(\mathbb{N}^{\mathcal{V}_n})$ is then related to a numerical domain. Many relational numerical domains have been introduced in the literature [10, 17, 21, 24]. We propose two choices in accordance with the expected trade-off between complexity and accuracy. They both use the complete lattice of affine equality systems among a set of variables. This domain is described with its lattice operations in [21]. Given a set of variables $\mathcal{V}$, we denote by $\mathcal{K}_\mathcal{V}$ the domain of the affine equality systems among the elements of $\mathcal{V}$.

7.2.3.2.1 Component-wise affine comparison

The first choice for abstracting a set of word tuples consists in abstracting the affine relations between the occurrence number of λ in each tuple component, for each λ in Σ_m. This way, we introduce Id_n^{Rel} as the complete lattice $(\Sigma_m \to \mathcal{K}_{\{x_i \mid i \in [\![1;n]\!]\}})$ defined point-wise.

We then define abstract primitives as follows:

- $\varepsilon^\sharp$ maps each letter in Σ_m to the system composed of the only constraint $x_1 = 0$;
- $t^\sharp$ maps each letter in Σ_m to the system composed of the only constraint $x_1 = 0$, except the letter a_m which is mapped to the system that contains no constraint, where $a_1 = cont_!$ and $a_2 = (cont_?, cont_!)$;
- $\bigsqcup_n$ applies the affine hull point-wise;

- $\underset{(i_1,...,i_p)}{\Pi}(f)(\lambda)$ is obtained by using Gaussian elimination to collect all the constraints involving only the variables $\{x_{i_k}\}$ in the system $f(\lambda)$, and then by renaming each variable x_{i_k} into the variable x_k;
- $\forall f \in Id_p^{\mathrm{Rel}}, g \in Id_q^{\mathrm{Rel}},\ f \bullet g$ maps each letter $\lambda \in \Sigma_m$ to the system composed of the constraints in $f(\lambda)$ and those in $g(\lambda)$, where each occurrence of a variable matching x_i in the constraints of $g(\lambda)$ is replaced by the variable x_{i+p};
- for each $\lambda \in \Sigma_m$, $sync(\{(i_1, j_1), ..., (i_p, j_p)\}, f)(\lambda)$ is obtained by inserting all the constraints of the form $x_{i_k} = x_{j_k}$, for $k \in [\![1; p]\!]$, in the system $f(\lambda)$;
- since there is no infinite ascending sequence, we define our widening operator ∇_n to be $\bigsqcup_n$;
- $\underset{\lambda}{push}(f)$ is obtained by first replacing each occurrence of the variable x_2 in the constraints of the system $f(\lambda)$ by the expression $x_2 - 1$, and then applying the abstract projection $\underset{(2,3)}{\Pi}$.

7.2.3.2.2 Global affine comparison

The second choice consists in abstracting globally all constraints. Roughly speaking, the variable x_i^λ denotes the occurrence number of λ in the i-th component of marker tuples. We introduce Id^{Rel} as the complete lattice $\mathcal{K}_{\mathcal{V}_n}$ of affine equality systems on the set of variables $\mathcal{V}_n$. Lattice operations are described in [21]. We describe the remaining primitives as follows:

- $\varepsilon^\sharp$ is given by the system $\{x_1^\lambda = 0,\ \forall \lambda \in \Sigma$;
- $t^\sharp$ is given by the system $\left\{x_1^\lambda = 0,\ \forall \lambda \in \Sigma \setminus \{a_m\}\right.$,
  where $a_1 = cont_!$ and $a_2 = (cont_?, cont_!)$;
- given $K \in Id_p^{\mathrm{Rel}}$ and $K' \in Id_q^{\mathrm{Rel}}$, we obtain the abstract concatenation of K and K', by renaming each variable x_i^λ to x_{i+p}^λ in K', and gathering all the constraints of the two systems;
- $sync(\{(i_1, j_1), ..., (i_p, j_p)\}, K)$ corresponds to inserting all the constraints of the form $x_{i_k}^\lambda = x_{j_k}^\lambda,\ \forall k \in [\![1; p]\!],\ \lambda \in \Sigma$ in K;
- $\underset{(i_1,...,i_p)}{\Pi}(K)$ corresponds to collecting all the constraints involving only the variables $\{x_{i_k}^\lambda\}$ and then renaming each variable $x_{i_k}^\lambda$ into the variable x_k^λ;
- $\underset{(i,j)}{push}(K)$ is obtained by replacing first in each constraint each occurrence of the variable $x_2^{(i,j)}$ by the expression $x_2^{(i,j)} - 1$ and then applying the abstract projection $\underset{(2,3)}{\Pi}$.

Remark 31 *The global affine comparison domain has not been implemented yet.*

7.2.4 Examples

We now describe results obtained on our examples. All these results are obtained by setting $m = 2$. In the description of these results, we make no distinction between a marker and its abstraction by Φ_2.

Example 32 *In the* ftp*-server example , the analyzer proves that the name of a channel opened by the restriction* (ν ***address***) *can only be communicated to the variable* ***email*** *or to the variable* ***address****, and that the name of a channel opened by the restriction* (ν ***request***) *can only be communicated to the variable* ***request****, to the variable* ***data*** *or to the variable* ***rep****. More specifically, it discovers that each time an agent* ***email***$![$***rep***$]$ *is spawned, there exist p,q in* $\mathbb{N}$ *such that the agent marker is* $(1,16).(1,5)^p.((2,4).(6,3).(2,12))^q.(2,4).(6,3)$*; the variable* ***email*** *is linked to the name of a channel opened by the restriction* (ν ***address***) *of the instance the marker of which was* $(1,16).(1,5)^p$*; and the variable* ***rep*** *is linked to the name of a channel opened by the restriction* (ν ***request***) *of the instance the marker of which was* $(1,16).(1,5)^p$*. This is enough to prove that both variables* ***email*** *and* ***data*** *are linked to names of channels opened by the same instance of the client resource and so the answer to a query can only be sent back to the correct client.*

Example 33 *In the token-ring example, the analyzer discovers that in each instance of an agent* **mon**$![$***left***, ***right***$]$*, the variable* ***left*** *is either bound to the name of a channel opened by an instance of the* (ν ***right***) *restriction or to the name of a channel opened by an instance of the* (ν **left0**) *restriction, and the variable* ***right*** *is always bound to the name of a channel opened by an instance of the* (ν ***right***) *restriction. More specifically, in the case where the variable* ***left*** *is bound to the name of a channel opened by an instance of the* (ν ***right***) *restriction, it discovers that there exists* $n \in \mathbb{N}$ *such that the instance marker of* **mon**$![$***left***, ***right***$]$ *is* $(\mathbf{1},\mathbf{6})(\mathbf{1},\mathbf{3})^{\mathbf{n+1}}$*; the variable* ***left*** *is linked to the name of a channel opened by the restriction* (ν ***right***) *of the instance the marker of which was* $(\mathbf{1},\mathbf{6})(\mathbf{1},\mathbf{3})^{\mathbf{n}}$*; and the variable* ***right*** *is linked to the name of a channel opened by the restriction* (ν ***right***) *of the instance the marker of which was* $(\mathbf{1},\mathbf{6})(\mathbf{1},\mathbf{3})^{\mathbf{n+1}}$*. This is enough to prove that each process can only be linked to either the next one or to the first one.*

Remark 34 *Our confinement analysis is not simply an abstraction of our non-uniform analysis since 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 [27]). The equality of the Parikh vectors implies the equality of the markers if they are recognized by an automaton that contains only one acyclic path between an initial and a final state, 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 a reduced product of both our confinement analysis and our non-uniform control flow analysis to solve*

this problem.

Remark 35 *The uniform analysis is not complete with respect to the non-uniform one, this means, that computing the non-uniform analysis and then abstracting the result in order to ignore marker information can give more accurate results than directly computing the result of the uniform analysis. This is illustrated in example 36.*

Example 36 *We consider the following mobile system:*

$(\nu\ a)(\nu\ b)(\nu\ x)$
$(\ *x?^{1}[z]((\nu\ t)z!^{2}[t]t!^{3}[z])$
$|\ *\ repli?^{4}[]x!^{5}[a]$
$|\ *\ repli?^{6}[]x!^{7}[b]$
$|\ *\ a?^{8}[i]i?^{9}[j]trace!^{10}[j]$
$)$

This system is composed of four resources. The second and the third ones allow the spawning of an unbounded number of agents either of the form $\mathbf{x}!^{5}[\mathbf{a}]$ *or of the form* $\mathbf{x}!^{7}[\mathbf{b}]$*. The first resource can be replicated by an agent either of the form* $\mathbf{x}!^{5}[\mathbf{a}]$ *or of the form* $\mathbf{x}!^{7}[\mathbf{b}]$*, nevertheless the behavior of the spawned instance is deeply bound to which kind of agent has replicated the resource:*

- *when the resource is replicated with an agent of the form* $\mathbf{x}!^{5}[\mathbf{a}]$*: a channel is opened; its name* $\boldsymbol{t}$ *is sent via the channel named* $\mathbf{a}$*, so that* $\boldsymbol{t}$ *can be send to an instance of the fourth resource; this instance can then receive the name* $\mathbf{a}$ *via the channel denoted by* $\boldsymbol{t}$*; then the instance of the first resource can send the name* $\mathbf{a}$ *via the channel named* $\boldsymbol{t}$*, so that the instance of the fourth resource can receive the name* $\mathbf{a}$ *via the channel named* $\boldsymbol{t}$ *and send it through the channel named* **trace***; then an intruder can get the name* $\mathbf{a}$*, by spying the channel named* **trace***;*
- *when the resource is replicated with an agent of the form* $\mathbf{x}!^{7}[\mathbf{b}]$*: a channel is opened, its name* $\boldsymbol{t}$ *is sent via the channel named* $\mathbf{b}$*, but cannot be received, so the instance is stuck, and no intruder can get the name* $\mathbf{b}$*.*

The non-uniform analysis captures the fact that the name $\mathbf{b}$ *cannot be spied by an intruder, while the uniform does not. Roughly speaking, the main reason is that the non-uniform analysis relates the names communicated to an agent with the history of the replications which have led to the creation of this agent, while the uniform analysis abstracts this information away.*

8 Occurrence counting analysis

We now propose to count the occurrence number of agents during computation sequences. We first abstract away any information about markers and environments. This way, a configuration is just seen as a multi-set of syntactic agents. In order to capture many more properties, we relate occurrence number of agents to the number of performed transitions. In this section, we are not interested in channels and markers, so we will denote a configuration (C, U, F) by its set of threads C.

Our abstraction requires an abstract domain to describe natural number families. In the first subsection, we introduce a generic analysis, independently of this domain. In the next subsection, we propose a well-suited abstract domain.

8.1 Generic analysis

We denote by $\mathcal{P}$ the set of syntactic components of $\mathcal{S}$ and choose a set $\mathcal{T}$ of variables used to count performed transitions. Since the number of transition labels is quadratic, we may want to quotient this set into equivalence classes, in order to deal with fewer variables. The set $(\mathcal{L}_{\text{used}} \cup \{0\})^2$ is related to $\mathcal{T}$ by an onto map ψ. We propose three natural choices for $\mathcal{T}$ and ψ in accordance with the expected trade-off between accuracy and efficiency:

(1) we can use one extra variable by assigning
$\mathcal{T} = \{0\}$ and $\forall i, j \in \mathcal{L}_{\text{used}} \cup \{0\}, \psi(i, j) = 0$,
(2) we can use a linear number of extra variables by assigning
$\mathcal{T} = \{(0, j) \mid j \in \mathcal{L}_{\text{used}} \cup \{0\}\}$ and $\forall i, j \in \mathcal{L}_{\text{used}} \cup \{0\},\ \psi(i, j) = (0, j)$,
(3) we can use a quadratic number of extra variables by assigning
$\mathcal{T} = (\mathcal{L}_{\text{used}} \cup \{0\})^2$ and $\forall i, j \in \mathcal{L}_{\text{used}} \cup \{0\},\ \psi(i, j) = (i, j)$.

Let V be the set $\mathcal{P} \cup \mathcal{T}$. We introduce an abstraction $\Pi_{\mathbb{N}^V}$ which maps each concrete configuration (u, C) in $\Sigma^* \times \mathcal{C}(\mathcal{S})$ to a family of natural numbers indexed by the set V, as follows:

$$(\Pi_{\mathbb{N}^V}(u, C))_v = \begin{cases} \mathit{Card}(\{(P, id, E) \in C \mid P = v\}) & \text{if } v \in \mathcal{P} \\ \sum\limits_{\lambda \in \psi^{-1}(v)} |u|_\lambda & \text{if } v \in \mathcal{T}. \end{cases}$$

Proposition 37 *Let (u_1, C_1) and (u_2, C_2) be two configurations such that $C_1 \cap C_2 = \emptyset$, then for any $v \in V$, we have $\Pi_{\mathbb{N}^V}(u_1.u_2, C_1 \cup C_2)_v = (\Pi_{\mathbb{N}^V}(u_1, C_1))_v + (\Pi_{\mathbb{N}^V}(u_2, C_2))_v$.*

We then consider $\wp(\mathbb{N}^V)$, the complete lattice of sets of natural number families indexed by V. The power set $\wp(\mathbb{N}^V)$ is related to our concrete domain $\wp(\Sigma^* \times$

$\mathcal{C}(\mathcal{S})$) via a concretization function $\gamma_{\mathbb{N}^V}$, where $\forall A^\sharp$, $\gamma_{\mathbb{N}^V}(A^\sharp)$ is defined as follows:

$$\left\{(u, C) \in \Sigma^* \times \mathcal{C}(\mathcal{S}) \;\middle|\; \Pi_{\mathbb{N}^V}(u, C) \in A^\sharp\right\}.$$

We then introduce a generic pre-order $(\mathcal{N}_V, \sqsubseteq_{\mathcal{N}_V})$ to represent sets of natural number families indexed by V. It is related to $\wp(\mathbb{N}^V)$ via a monotonic concretization $\gamma_{\mathcal{N}_V}$. Furthermore, we introduce several generic primitives: a representation $\bot_{\mathcal{N}_V}$ of the empty set, an abstract union $\cup_{\mathcal{N}_V}$, a widening operator $\nabla_{\mathcal{N}_V}$, an abstract counterpart $+$ to the addition, an abstract counterpart $-$ to the subtraction, an abstract synchronization $sync_{\mathcal{N}_V}$. We also require the abstraction of some elementary family: an abstract element $0_{\mathcal{N}_V}$ to represent the singleton containing the 0 family which associates 0 to each element in V, and $\forall v \in V$, an abstract element $1_{\mathcal{N}_V}(v)$ to represent the singleton containing the family δ^v which associates 1 to the element v and 0 to any other elements.

These primitives should satisfy the following properties:

(1) $\gamma_{\mathcal{N}_V}(\bot_{\mathcal{N}_V}) = \emptyset$,
(2) $\forall a \in \mathcal{N}_V,\ \bot_{\mathcal{N}_V} \sqsubseteq_{\mathcal{N}_V} a$,
(3) $\forall A \in \wp_{\text{finite}}(\mathcal{N}_V),\ \cup_{\mathcal{N}_V}(A) \in \mathcal{N}_V$ and $\forall a \in A,\ a \sqsubseteq_{\mathcal{N}_V} \cup_{\mathcal{N}_V}(A)$,
(4) $\nabla_{\mathcal{N}_V}$ satisfies Def. 24.(7),
(5) $\forall a^\sharp, b^\sharp \in \mathcal{N}_V,\ (a^\sharp + b^\sharp) \in \mathcal{N}_V$ and
$\{(a_v + b_v)_{v\in\mathcal{V}} \mid a \in \gamma_{\mathcal{N}_V}(a^\sharp), b \in \gamma_{\mathcal{N}_V}(b^\sharp)\} \subseteq \gamma_{\mathcal{N}_V}(a^\sharp + b^\sharp)$,
(6) $\forall a^\sharp, b^\sharp \in \mathcal{N}_V,\ (a^\sharp - b^\sharp) \in \mathcal{N}_V$ and
$\{(a_v - b_v)_{v\in\mathcal{V}} \mid a \in \gamma_{\mathcal{N}_V}(a^\sharp), b \in \gamma_{\mathcal{N}_V}(b^\sharp),\ \forall v \in V,\ a_v \geqslant b_v\} \subseteq \gamma_{\mathcal{N}_V}(a^\sharp - b^\sharp)$,
(7) $\forall a^\sharp \in \mathcal{N}_V,\ I \in \wp(\mathcal{L}),\ sync_{\mathcal{N}_V}(I, a^\sharp) \in \mathcal{N}_V$ and
$\{a \mid a \in \gamma_{\mathcal{N}_V}(a^\sharp), \forall i \in I,\ a_i \geqslant 1\} \subseteq \gamma_{\mathcal{N}_V}(sync_{\mathcal{N}_V}(I, a^\sharp))$,
(8) $0_{\mathcal{N}_V} \in \mathcal{N}_V$ and $(0)_{i\in V} \in \gamma_{\mathcal{N}_V}(0_{\mathcal{N}_V})$,
(9) $\forall v \in V,\ 1_{\mathcal{N}_V}(v) \in \mathcal{N}_V$ and $(\delta_i^v)_{i\in V} \in \gamma_{\mathcal{N}_V}(1_{\mathcal{N}_V}(v))$.

Roughly speaking, $+$ is an abstract counterpart to the component-wise addition, $-$ is an abstract counterpart to the component-wise subtraction, $sync_{\mathcal{N}_V}$ is used to extract from an abstract value the representation of the configurations which simultaneously contain all the agents required by a computation step.

We define an abstract transition system $(C_0^{\mathcal{N}_V}, \rightsquigarrow_{\mathcal{N}_V})$ in Fig. 15. It uses an abstract extraction function $\beta_{\mathcal{N}_V}$ which computes an approximation of the syntactic agents which are spawned at the beginning of the system or when launching a continuation. Internal choice $\oplus$ and parallel composition $|$ are just abstracted by their respective abstract counterparts $\cup_{\mathcal{N}_V}$ and $+$.

Proposition 38 *$\forall P \in \mathcal{P},\ \forall id \in \mathcal{M},\ \forall E \in (fn(P) \to (\nu n(\mathcal{S}) \times \mathcal{M}))$, we have $Cont \in \beta(P, id, E) \implies (\varepsilon, Cont) \in (\gamma_{\mathbb{N}^V} \circ \gamma_{\mathcal{N}_V})(\beta_{\mathcal{N}_V}(P))$.*

The proof of Prop. 38 is shown in appendix G.

The initial abstract configuration is obtained by applying $\beta_{\mathcal{N}_V}$ to the initial system. An abstract communication between two syntactic components labelled i and j from an abstract state $C^\sharp$ first consists in computing a description of the set of the configurations described in $C^\sharp$ in which both syntactic components may occur simultaneously. If this set is empty, the components are mutually exclusive and the abstract communication is disabled. Otherwise, the result of the communication is obtained by translation: decreasing the occurrence number of the communicating agents which are not resources, and adding the description of both continuations. External interactions are dealt with in the same way, but only one agent is involved. We do not bother about the control flow, since we use a reduced product between this analysis and one of our control flow analyses.

Theorem 39 $(\mathcal{N}_V, \sqsubseteq_{\mathcal{N}_V}, \cup_{\mathcal{N}_V}, \bot_{\mathcal{N}_V}, \gamma_{\mathbb{N}^V} \circ \gamma_{\mathcal{N}_V}, C_0^{\mathcal{N}_V}, \rightsquigarrow_{\mathcal{N}_V}, \nabla_{\mathcal{N}_V})$ *is an abstraction.*

The proof of Thm. 39 is shown in appendix G.

8.2 Abstract domain

We only need to define an abstract domain to approximate sets of indexed family of natural numbers in which abstract primitives can be precisely and efficiently implemented. On the one hand, the primitive $sync_{\mathcal{N}_V}$ needs to express the property that several variables may simultaneously be greater than one, which is a relational information. On the other hand, the shape of the abstract computation rule suggests that the domain should be closed under addition. This problem is very similar to abstracting the occurrence number of ambients [18, 26], or even approximating the collecting semantics of a Petri net. We reject the use of usual numerical domains. We are unlikely to design a precise primitive $sync_{\mathcal{N}_V}$ in non-relational domains. Disjunctive completion can be used to lift a non-relational domain into a relational one: each configuration is abstracted in a finite non-relational domain, and then the abstraction of a configuration set is given by the collection of all the abstractions of its elements. Nevertheless, disjunctive completion often leads to a lack of accuracy and an exponential explosion. We cannot afford the domain of linear inequalities among a finite set of variables [10] because we deal with too many variables.

We propose the use of a reduced product between two domains: the interval lattice and the linear equalities lattice described in [21]. The first domain is based on the use of the interval lattice and is used for expressing properties of interest. This domain can represent all the properties we need to express

$$\begin{aligned}
\beta_{\mathcal{N}_V}((\nu\ n)P) &= \beta_{\mathcal{N}_V}(P) \\
\beta_{\mathcal{N}_V}(\mathbf{0}) &= 0_{\mathcal{N}_V} \\
\beta_{\mathcal{N}_V}(P \oplus Q) &= \beta_{\mathcal{N}_V}(P) \cup_{\mathcal{N}_V} \beta_{\mathcal{N}_V}(Q) \\
\beta_{\mathcal{N}_V}(P \mid Q) &= \beta_{\mathcal{N}_V}(P) + \beta_{\mathcal{N}_V}(Q) \\
\beta_{\mathcal{N}_V}(aP) &= 1_{\mathcal{N}_V}(aP)
\end{aligned}$$

(a) Abstract extraction

$$C_0^{\mathcal{N}_V} = \beta_{\mathcal{N}_V}(\mathcal{S})$$

(b) Initial configuration

Let $v^\sharp$ be an abstract configuration,
if there are $i, j \in \mathcal{L}_{\text{used}} \cup \{0\}$,
such that $sync_{\mathcal{N}_V}(\{agent(l) \mid l \in \{i,j\} \setminus \{0\}\}, v^\sharp) \neq \bot_{\mathcal{N}_V}$,
then
$v^\sharp \overset{(i,j)}{\rightsquigarrow}_{\mathcal{N}_V} (sync_{\mathcal{N}_V}(\{agent(l) \mid l \in \{i,j\}\setminus\{0\}\}, v^\sharp) + Act(i) + Act(j) + 1_{\mathcal{N}_V}(\psi(i,j)))$,
where

- $\forall l \in \mathcal{L}_{\text{used}} \setminus \{0\},\ agent(l) \in \mathcal{P} \subset V$ is the syntactic agent labelled with l,
- $Act(l) = \begin{cases} 0_{\mathcal{N}_V} & \text{if } l = 0 \\ \beta_{\mathcal{N}_V}(P) - 1_{\mathcal{N}_V}(aP) & \text{if } agent(l) = aP \\ & \text{and } a \text{ matches } y?^l[\overline{y}] \text{ or } x!^l[\overline{x}] \\ \beta_{\mathcal{N}_V}(P) & \text{if } agent(l) \text{ matches } *y?^l[\overline{y}]P. \end{cases}$

(c) Abstract transition rule

Fig. 15. Occurrence counting analysis

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 [21] and is used for expressing more complex properties, such as mutual exclusion, which allows for more precise calculations in the first domain. The power of our analysis directly comes from the use of an inexpensive algorithm, straightforwardly adapted from Linear Constraint Programming, to calculate an approximated reduction between these two domains.

8.2.1 Interval domain

The complete lattice $(\mathcal{I}, \sqsubseteq_{\mathcal{I}}, \bigsqcup_{\mathcal{I}}, \bot_{\mathcal{I}}, \sqcap_{\mathcal{I}}, \top_{\mathcal{I}})$ is the functional domain of natural number intervals, where lattice operations are defined point-wisely. The abstract domain $\mathcal{I}$ is related to $\wp(\mathbb{N}^V)$ via the monotonic map $\gamma_{\mathcal{I}}$ defined by:

$$\gamma_{\mathcal{I}}(f) = \{u \in \mathbb{N}^V \mid \forall i \in V,\ u_i \in f(i)\}.$$

A family $(\nabla^n_{\mathcal{I}})$ of widening operators on $\mathcal{I}$ is defined as follows:

$$[f \nabla^n_{\mathcal{I}} g](x) = f(x) \nabla^n g(x)$$
$$\text{where } [\![a; b]\!] \nabla^n [\![c; d]\!] = \begin{cases} [\![min\{a; c\}; +\infty[\![& \text{if } d > max\{b; n\} \\ [\![min\{a; c\}; max\{b; d\}]\!] & \text{otherwise.} \end{cases}$$

We can easily define abstract primitives in $\mathcal{I}$ as follows:

- $(f +_{\mathcal{I}} g) = [x \to f(x) + g(x)]$,
- $(f -_{\mathcal{I}} g) = [x \to (f(x) - g(x)) \cap [\![0; +\infty[\![\ ]$,
- $sync_{\mathcal{I}}(I, f) = \left[x \to \begin{cases} f(x) \cap [\![1; +\infty[\![& \text{if } x \in I \\ f(x) & \text{otherwise} \end{cases}\right]$,
- $0_{\mathcal{I}} = [x \to [\![0; 0]\!]\ ]$,
- $1_{\mathcal{I}}(v) = \left[\begin{cases} x \to [\![1; 1]\!] & \text{if } x = v \\ x \to [\![0; 0]\!] & \text{otherwise} \end{cases}\right]$.

8.2.2 Linear equalities domain

The complete lattice $(\mathcal{K}, \sqsubseteq_{\mathcal{K}}, \cup_{\mathcal{K}}, \top_{\mathcal{K}}, \cap_{\mathcal{K}}, \bot_{\mathcal{K}})$ of linear equality systems between the finite set of variables V is described with its lattice operations in [21]. This domain uses Gaussian elimination in order to normalize systems. It is related to the domain $\wp(\mathbb{N}^V)$ via the monotonic function $\gamma_{\mathcal{K}}$ which maps each system to the set of its solutions. Moreover, since there is no infinite ascending chain [21], we can choose $\cup_{\mathcal{K}}$ as a widening operator. To define the addition $+_{\mathcal{K}}$ and the subtraction $-_{\mathcal{K}}$ of two systems, we compute a particular solution of each system O_1 and O_2, and a linear direction $\overrightarrow{H_1}$ and $\overrightarrow{H_2}$. Then we use the following equalities:

$$(O_1 + \overrightarrow{H_1}) +_{\mathcal{K}} (O_2 + \overrightarrow{H_2}) = (O_1 + O_2) + (\overrightarrow{H_1} \cup_{\mathcal{K}} \overrightarrow{H_2}),$$
$$(O_1 + \overrightarrow{H_1}) -_{\mathcal{K}} (O_2 + \overrightarrow{H_2}) = (O_1 - O_2) + (\overrightarrow{H_1} \cup_{\mathcal{K}} \overrightarrow{H_2}).$$

Such a decomposition can be extracted from the normal form in linear time, so the cost of affine addition and subtraction is cubic.

Other primitives are defined as follows:

- $sync_{\mathcal{K}}(I, K) = K$,
- $0_{\mathcal{K}} = \{x = 0,\ \forall x \in V$,
- $1_{\mathcal{K}}(v) = \begin{cases} x = 1 \text{ if } x = v \\ x = 0 \text{ otherwise.} \end{cases}$

Roughly speaking, synchronization cannot be directly checked in $\mathcal{K}$. So we define it as the identity. Linear constraints will therefore be used to prove that synchronization interval constraints are incompatible, by reduction. Other primitive definitions are straightforward.

8.2.3 Approximated reduced product

Our numerical domain is the product $(\mathcal{I} \times \mathcal{K})$. It is partially ordered by the pair-wise order $\sqsubseteq_{\mathcal{I}} \times \sqsubseteq_{\mathcal{K}}$. It is related to $\wp(\mathbb{N}^V)$ by the concretization function $\gamma_{\mathcal{N}_V}$ where $\gamma_{\mathcal{N}_V}(i, s)$ is defined as $\gamma_{\mathcal{I}}(i) \cap \gamma_{\mathcal{K}}(s)$. Generic primitives are expressed as follows:

- $\bot_{\mathcal{N}_V}$, $\cup_{\mathcal{N}_V}$, $\nabla_{\mathcal{N}_V}$, $+$, $\sqsubseteq_{\mathcal{N}_V}$, $0_{\mathcal{N}_V}$, $1_{\mathcal{N}_V}$ are defined pair-wisely,
- $sync_{\mathcal{N}_V}(A, (i, s)) = \rho(i', s)$,
  where $\begin{cases} i'(x) = i(x) \cap [\![1; +\infty|\![& \forall x \in A \\ i'(x) = i(x) & \forall x \in \mathcal{P} \setminus A, \end{cases}$
  and ρ satisfies $\forall a \in \mathcal{I} \times \mathcal{K},\ \gamma_{\mathcal{N}_V}(a) \sqsubseteq_{\mathcal{N}_V} \gamma_{\mathcal{N}_V}(\rho(a))$.

The definition of $sync_{\mathcal{N}_V}$ uses a reduction operator ρ which satisfies the following property:

$$\forall a \in \mathcal{I} \times \mathcal{K},\ \gamma_{\mathcal{N}_V}(a) \sqsubseteq_{\mathcal{N}_V} \gamma_{\mathcal{N}_V}(\rho(a)).$$

Roughly speaking, the operator ρ is a reduction, it simplifies constraints without losing any solution. We now present a reduction ρ between $\mathcal{I}$ and $\mathcal{K}$. It consists in taking into account linear constraints in order to narrow interval ranges. 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 Gaussian elimination, 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 computing an approximate solution to this problem. The first step aims at narrowing infinite intervals into finite ones. It uses Gaussian elimination 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[7], which allows narrowing most of the infinite intervals into finite ones with a $\mathcal{O}(|V|^3)$ worst-case. 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 to propagate unidirectionally intervals from non-diagonal to diagonal variables. The result is a good reduction with a $\mathcal{O}(|V|^4)$ worst-case. In the case that the algorithm discovers an unsatisfiable constrain, the result of the reduction is set to $\bot_{\mathcal{N}_V}$.

8.2.3.1 Solving undefined forms Let V_{inf} be a subset of V and K a finite system of linear constraints on the variables V. We denote by K the system of equations:

$$\left\{ \sum_{v \in V} a_i^v.v = b_i, \forall i \in [\![1; n]\!] \right. .$$

We first define a positive form with respect to V_{inf} as follows:

Definition 40 *K is in positive form with respect to V_{inf} if and only if $\forall v \in V$, $\exists i_1, i_2 \in [\![1; n]\!]$ such that $a_{i_1}^v < 0 < a_{i_2}^v \implies v \notin V_{\text{inf}}$.*

This way, the variables which may occur in the matrix describing K with both a positive and a negative coefficient are known to be bounded[8]. Such a form can be computed by using the Gaussian elimination with a $\mathcal{O}(|V|^3)$ worst-case time cost. A positive form contains only few undefined forms. For each constraint in which all variables with an infinite range occur with the same sign, we can narrow the range of variables that occurs in this constraint into finite intervals. A dynamic resolution of such a system leads to a reduction step in $\mathcal{O}(|V|^2)$.

8.2.3.2 Narrowing finite intervals 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 to propagate unidirectionally intervals from non-diagonal to diagonal variables. The result is a good reduction with a $\mathcal{O}(|V|^4)$ worst-case time cost.

We use three kinds of reductions:

[7] An undefined form is a subtraction between two unbounded intervals

[8] V_{inf} denotes the set of the variables which are not proved to be bounded.

(1) Gaussian elimination:

$$\begin{cases} x+y+z=1 \\ x+y+t=2 \end{cases} \Longrightarrow \begin{cases} x+y+z=1 \\ t-z=1, \end{cases}$$

(2) interval propagation:

$$\begin{cases} x+y+z=3 \\ x \in [\![0;5]\!] \\ y \in [\![0;6]\!] \\ z \in [\![0;8]\!] \end{cases} \Longrightarrow \begin{cases} x+y+z=3 \\ x \in [\![0;\mathbf{3}]\!] \\ y \in [\![0;6]\!] \\ z \in [\![0;8]\!], \end{cases}$$

(3) redundancy introduction:

$$\begin{cases} x+y-z=3 \\ x \in [\![1;2|[\end{cases} \Longrightarrow \begin{cases} x+y-z=3 \\ y-z \in [\![1;2]\!] \\ x \in [\![1;2]\!]. \end{cases}$$

We first use Gaussian elimination to get a normal form of the linear constraint system, we then use interval propagation to narrow the range of the interval of the pivot of each constraint. Then, we forget about the pivot using redundancy introduction. We then get a new system involving only variables which were not a pivot of a constraint in the previous one. We then proceed recursively with it until constraints contain some variables. We then consider all the constraints we have computed which form a triangular system. We propagate the information we have collected on the interval ranges backward, by applying interval propagation from non-diagonal variables to diagonal ones, starting from the last constraint and ending with the first one.

8.2.4 Examples

We now describe some examples of mobile systems analyzed with our prototype. For the sake of brevity and simplicity, we do not present linear constraints. They are not of interest when considering the result as they are used internally to improve the interval information. Interval constraints are tagged with the actions of syntactic agents.

Example 41 *We give in Fig. 16 the result of occurrence analysis of the* ftp-*server. This result ensures that no more than three syntactic instances of the syntactic agent* ***deal***![***data***] *can simultaneously occur. So we can conclude that no more than three sessions can be active at the same time. This constraint is proved using the linear constraint which proves that the sum of the number of available ports* **port**![] *and activated sessions* ***deal***![***data***] *is always equal to three.*

Example 42 *We give in Fig. 17 the occurrence analysis result for the* token-ring. *This result ensures that only one syntactic instance of the agent* ***crit***$![]$ *may occur at any time. So only one process can proceed its critical section at the same time.*

Remark 43 *The analysis cannot succeed in proving that the occurrence number of the syntactic agent* $\mathbf{mon}![\boldsymbol{left}, \mathbf{left0}]$ *is less than or equal to* 1 *without counting the number of performed transitions. On the other hand, only a linear number of extra variables are required to prove this property.*

9 Complexity and benchmarks

9.1 About the complexity of our analyses

We briefly describe the complexity in the worst case of our analyses in Figs. 18 and 19. We denote by n the number of syntactic components in the initial system. The first line denotes the cost to scan all the possible labels of transition steps. The second line denotes the maximum number of stages in one abstract iteration. The third line denotes the cost to perform an abstract transition, it includes both abstract operation cost and information propagation. The worst-case time complexity of an analysis is then the product of those three partial worst-case time complexities.

In the third column of Fig. 18, m is a parameter of our analysis which denotes analysis precision. We have $m \in \{1; 2\}$ and in the case when $m = 1$ each transition label is abstracted by its second component, while the two components are kept when $m = 2$. In Fig. 19, i is also a parameter of our analysis such that the number of variables is in $\mathcal{O}(n^i)$. It is equal to 0 when no occurrence analysis is done; it is equal to 1 when only syntactic agents are counted; it is also equal to 1 when we also count the number of sent messages; it is equal to 2 when we count performed transitions.

To obtain the worst-case time complexity of the product of two analyzes, we add pair-wise the complexities due to transition label scanning, the maximum number of iterations, and the cost of an abstract transition and then make the product of these three complexities. So, for instance, the analysis obtained as the product between our non-uniform global analysis with $m = 2$ and our occurrence counting analysis with $i = 2$, has a $\mathcal{O}(n^2 \times n^7 \times n^{10})$ worst-case time cost.

$$
\begin{array}{l}
((\nu\ \mathrm{make})(\nu\ \mathrm{server})(\nu\ \mathrm{port}) \\
((*\mathrm{make}?^{1}[](\nu\ \mathit{address})(\nu\ \mathit{request}) \\
\qquad\qquad (\\
\qquad\qquad (*\mathit{address}?^{[\![0;+\infty[\![}[]\mathrm{server}!^{[\![0;+\infty[\![}[\mathit{address}, \mathit{request}]) \\
\qquad\qquad | \\
\qquad\qquad \mathit{address}!^{[\![0;+\infty[\![}[] \\
\qquad\qquad | \\
\qquad\qquad \mathrm{make}!^{[\![0;1]\!]}[] \\
\qquad\qquad)) \\
| \\
(*\mathrm{server}?^{1}[\mathit{email}, \mathit{data}](\nu\ \mathit{deal}) \\
\qquad\qquad (\\
\qquad\qquad \mathrm{port}?[]^{[\![0;+\infty[\![} \\
\qquad\qquad\qquad (\\
\qquad\qquad\qquad \mathit{deal}!^{[\![0;3]\!]}[\mathit{data}] \\
\qquad\qquad\qquad | \\
\qquad\qquad\qquad \mathit{deal}?^{[\![0;3]\!]}[\mathit{rep}](\mathit{email}!^{[\![0;+\infty[\![}[\mathit{rep}] \mid \mathrm{port}!^{[\![0;3]\!]}[]) \\
\qquad\qquad\qquad) \\
\qquad\qquad \oplus \\
\qquad\qquad \mathit{email}!^{[\![0;+\infty]\!]}[] \\
\qquad\qquad)) \\
| \ \mathrm{port}!^{[\![0;1]\!]}[] \\
| \ \mathrm{port}!^{[\![0;1]\!]}[] \\
| \ \mathrm{port}!^{[\![0;1]\!]}[] \\
| \ \mathrm{make}!^{[\![0;1]\!]}[]) \\
)
\end{array}
$$

Fig. 16. The *ftp*-server occurrence analysis

$$
\begin{array}{l}
((\nu\ \mathrm{make})(\nu\ \mathrm{mon})(\nu\ \mathrm{left0}) \\
(\\
((*\mathrm{make}?^{1}[\mathit{left}](\nu\ \mathit{right})(\mathrm{mon}!^{[\![0;+\infty[\![}[\mathit{left}, \mathit{right}] \mid \mathrm{make}!^{[\![0;+\infty[\![}[\mathit{right}])) \\
| (*\mathrm{make}?^{1}[\mathit{left}](\mathrm{mon}!^{[\![0;1]\!]}[\mathit{left}, \mathrm{left0}])) \\
| \ \mathrm{make}!^{[\![0;1]\!]}[\mathrm{left0}]) \\
| \\
((*\mathrm{mon}?^{1}[\mathit{prev}, \mathit{next}] \\
\qquad\qquad (*\mathit{prev}?^{[\![0;+\infty[\![}[](\nu\ \mathit{crit})(\mathit{crit}?^{[\![0;1]\!]}[]\mathit{next}!^{[\![0;1]\!]}[] \\
\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad | \ \mathit{crit}!^{[\![0;1]\!]}[]))) \\
| \ \mathrm{left0}!^{[\![0;1]\!]}[])))
\end{array}
$$

Fig. 17. The token-ring occurrence analysis

	0-CFA	confinement	non-uniform comp.-wise $m \in \{1;2\}$	non-uniform global $m \in \{1;2\}$
label scan	$\mathcal{O}(n^2)$	$\mathcal{O}(n^2)$	$\mathcal{O}(n^2)$	$\mathcal{O}(n^2)$
iteration height	$\mathcal{O}(n^3)$	$\mathcal{O}(n^3)$	$\mathcal{O}(n^{2\times m+3})$	$\mathcal{O}(n^{2\times m+3})$
transition complexity	$\mathcal{O}(n^4)$	$\mathcal{O}(n^4)$	$\mathcal{O}(n^{2\times m+4})$	$\mathcal{O}(n^{3\times m+4})$
total cost	$\mathcal{O}(n^9)$	$\mathcal{O}(n^9)$	$\mathcal{O}(n^{4\times m+9})$	$\mathcal{O}(n^{5\times m+9})$

Fig. 18. Control flow analysis complexity

	occurrence counting analysis $i \in \{0;1;2\}$
scan	$\mathcal{O}(n^2)$
iteration height	$\mathcal{O}((n^i)^2)$
transition complexity	$\mathcal{O}((n^i)^4)$
total cost	$\mathcal{O}(n^{6\times i+2})$

Fig. 19. Occurrence counting analysis complexity

ftp-server $n = 16$	0-CFA	non-uniform comp.-wise $m = 1$	non-uniform comp.-wise $m = 2$
$i = 0,\ \|V\| = 0$	0.03s	0.22s	0.21s
$i = 1,\ \|V\| = n$	0.44s	**0.71s**	**0.71s**
$i = 1,\ \|V\| = 2 \times n$	1.40s	**2.16s**	**2.13s**
$i = 2,\ \|V\| = n^2 + n$	1.42s	**2.17s**	**2.15s**

Fig. 20. The *ftp*-server analysis time

9.2 Analysis time

We report in Figs. 20 and 21 CPU-time used in analyzing our examples on a PENTIUM III, 500 MHz CPU with 128 MB RAM. We write the time in boldface when the analysis has succeeded in proving all the properties we were interested in.

Token-ring $n = 12$	0-CFA	non-uniform comp.-wise $m = 1$	non-uniform comp.-wise $m = 2$
$i = 0,\ \|V\| = 0$	0.03s	0.15s	0.14s
$i = 1,\ \|V\| = n$	0.24s	0.43s	0.44s
$i = 1,\ \|V\| = 2 \times n$	0.68s	**0.86s**	**0.88s**
$i = 2,\ \|V\| = n^2 + n$	0.79s	**1.01s**	**1.04s**

Fig. 21. The token ring analysis time

10 Conclusion

We have proposed a parametric framework for automatically inferring the properties of mobile systems. We have first introduced a powerful non-standard semantics which explicitly encodes the link between agents and the channels they have opened. This non-standard semantics handles the full π-calculus and allows us to describe the behavior of a system part, abstracting away the behavior of its context. We have proved both the soundness and the completeness of this abstraction. This means that our semantics takes into account the behavior of any context specified in the π-calculus and that each behavior described by our semantics can effectively be obtained by choosing an appropriate context written in the π-calculus.

We have then proposed a generic decidable approximation of this non-standard semantics. As an example, we proposed to focus on the properties which rely on both control flow and agent occurrence number. Our control flow analysis has successfully detected and proved some complex integrity properties: we have proved that in the case of an *ftp*-server used by an unbounded number of clients, the server always returns each query to the correct client and that, in the case of a token ring, each process of the ring can only pass the token to the next or to the first process of the ring. We can also capture information about causality, since our analysis captures comparison between the history of the replications that have led to the creation of agents. The analysis of the agent occurrence number has successfully detected and proved non-exhaustion of resources and mutual exclusion: in the case of the *ftp*-server, the analysis has discovered that the server can only run three simultaneous sessions, while in the case of the token ring, the analysis has proved that only one process of the ring can be in its critical section at a given time.

In this framework, we only capture properties about configurations. We plan to use our analyses to provide partitioning criteria, so that we can capture decidable approximations of system traces. This way we will be able to check

behavioral properties, in which channel instances are distinguished. Because of the approximation, we will only consider the properties such that the set of the traces that satisfies these properties are closed under inclusion. Moreover, our framework can very likely apply to other formalisms as well as to real languages. It may be a great challenge to infer the same kind of properties when agent dynamic creation is also controlled by complex data properties.

Acknowledgements

We deeply thank anonymous referees for their significant comments on early versions of this paper. We wish also to thank Patrick and Radhia Cousot, and Arnaud Venet, for their insightful comments and discussions. We are also very grateful to Bruno Blanchet, Francesco Logozzo, Antoine Miné, David Monniaux, Xavier Rival, Eben Upton and Yves Verhoeven who helped us in correcting previous versions of this work.

References

[1] C. Bodei, P. Degano, F. Nielson, and H. Riis Nielson. Control flow analysis for the π-calculus. In *Proc. CONCUR'98*, LNCS. Springer-Verlag, 1998.

[2] C. Bodei, P. Degano, F. Nielson, and H. Riis Nielson. Static analysis of processes for no read-up and no write-down. In *Proc. FoSSaCS'99*, LNCS. Springer-Verlag, 1999.

[3] C. Bodei, P. Degano, F. Nielson, and H. Riis Nielson. Static analysis for the π-calculus with applications to security. *Information and Computation*, 165, 2000.

[4] L. Cardelli, G. Ghelli, and A. D. Gordon. Secrecy and group creation. In *Proc. CONCUR'00*, LNCS. Springer-Verlag, 2000.

[5] C. K. Chiu and J. H. M. Lee. Interval linear constraint solving using the preconditioned interval Gauss-Seidel method. In *Proc. ICLP'95*, Logic Programming. The MIT Press, 1995.

[6] P. Cousot. Semantic foundations of program analysis. In *Program Flow Analysis: Theory and Applications*, chapter 10. Prentice-Hall, Inc., 1981.

[7] 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*. ACM Press, 1977.

[8] P. Cousot and R. Cousot. Abstract interpretation frameworks. *Journal of Logic and Computation*, 2(4), 1992.

[9] P. Cousot and R. Cousot. Comparing the Galois connection and widening-narrowing approaches to abstract interpretation. In *Proc. PLILP'92*, LNCS. Springer-Verlag, 1992.

[10] P. Cousot and N. Halbwachs. Automatic discovery of linear restraints among variables of a program. In *Proc. POPL'78*. ACM Press, 1978.

[11] N. G. de Bruijn. Lambda-calculus notation with nameless dummies: a tool for automatic formula manipulation with application to the Church-Rosser theorem. *Indagationes Mathematicae*, 34(5), 1972.

[12] N. Dershowitz and J.-. Jouannaud. Rewrite systems. In *Handbook of Theoretical Computer Science, Volume B: Formal Models and Sematics (B)*. Elsevier Science Publishers, 1990.

[13] D. Dolev and A. C. Yao. On the security of public key protocols (extended abstract). In *Proc. FOCS'81*. IEEE Press, 1981.

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

[15] J. Feret. Confidentiality analysis of mobile systems. In *Proc. SAS'00*, LNCS. Springer-Verlag, 2000.

[16] J. Feret. Occurrence counting analysis for the π-calculus. In *Proc. GETCO'00*, volume 39.2 of *ENTCS*. Elsevier Science Publishers, 2001.

[17] P. Granger. Static analysis of linear congruence equalities among variables of a program. In *Proc. TAPSOFT'91*, LNCS. Springer-Verlag, 1991.

[18] R. R. Hansen, J. G. Jensen, F. Nielson, and H. Riis Nielson. Abstract interpretation of mobile ambients. In *Proc. SAS'99*, LNCS. Springer-Verlag, 1999.

[19] M. Hennessy and J. Riely. Resource access control in systems of mobile agents. In *Proc. HLCL'98*, volume 16.3 of *ENTCS*. Elsevier Science Publishers, 1998.

[20] A. Igarashi and N. Kobayashi. A generic type system for the π-calculus. In *Proc. POPL'01*. ACM, 2001.

[21] M. Karr. Affine relationships among variables of a program. *Acta Informatica*, 1976.

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

[23] R. Milner, J. Parrow, and D. Walker. A calculus of mobile processes. *Information and Computation*, 1992.

[24] A. Miné. The octagon abstract domain. In *Proc. AST'01 in WCRE'01*. IEEE Press, 2001.

[25] F. Nielson and H. Seidl. Control-flow analysis in cubic time. In *Proc. ESOP'01*, LNCS. Springer-Verlag, 2001.

[26] H. Riis Nielson and F. Nielson. Shape analysis for mobile ambients. In *Proc. POPL'00*. ACM Press, 2000.

[27] R. J. Parikh. On context-free languages. *Journal of the ACM*, 13, 1966.

[28] S. K. Rajamani and J. Rehof. A behavioral module system for the π-calculus. In *Proc. SAS'01*, LNCS. Springer-Verlag, 2001.

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

[30] A. Venet. Abstract interpretation of the π-calculus. In *Proc. LOMAPS'97*, LNCS. Springer-Verlag, 1997.

[31] A. Venet. Automatic determination of communication topologies in mobile systems. In *Proc. SAS'98*, LNCS. Springer-Verlag, 1998.

A Correspondence between the standard and the naive semantics

In this section we give the proof of Thm. 8, which formalizes the relation between the standard and the naive semantics.

Theorem 8 *We have $\mathcal{S} = \Pi(\mathcal{C}_0^{\mathrm{n}}(\mathcal{S}))$, and for any non-standard configurations C and for any word $u \in (\mathcal{L}^2 \cup \{\varepsilon; \oplus\})^*$ such that $\mathcal{C}_0^{\mathrm{n}}(\mathcal{S}) \xrightarrow{u}{}^*_{\mathrm{n}} C$, we have:*

(1) $C \xrightarrow{\varepsilon}_{\mathrm{n}} C' \implies \Pi(C) \equiv \Pi(C')$;
(2) $\forall \lambda \in \mathcal{L}^2 \cup \{\oplus\},\ C \xrightarrow{\lambda}_{\mathrm{n}} C' \implies \Pi(C) \xrightarrow{\lambda} \Pi(C')$;
(3) $\forall \lambda \in \mathcal{L}^2 \cup \{\oplus\},\ \Pi(C) \xrightarrow{\lambda} P \implies \exists D,\ \exists E,\ \begin{cases} C \xrightarrow{\varepsilon}{}^*_{\mathrm{n}} D \xrightarrow{\lambda}_{\mathrm{n}} E \\ \Pi(E) \equiv P. \end{cases}$

PROOF. We have $\mathcal{C}_0^{\mathrm{n}}(\mathcal{S}) = (\mathcal{S}, \varepsilon, \emptyset)$, then $\Pi(\mathcal{C}_0^{\mathrm{n}}(\mathcal{S})) = \mathcal{S}$. Let C be a non-standard configuration and u a word in $(\mathcal{L}^2 \cup \{\varepsilon; \oplus\})^*$ such that $\mathcal{C}_0^{\mathrm{n}}(\mathcal{S}) \xrightarrow{u}{}^*_{\mathrm{n}} C$,

(1) let C' be a non-standard configuration such that $C \xrightarrow{\varepsilon}_{\mathrm{n}} C'$, we want to prove that $\Pi(C) \equiv \Pi(C')$ by case analysis on $C \xrightarrow{\varepsilon}_{\mathrm{n}} C'$:
   (a) in the case that $C \xrightarrow{\varepsilon}_{\mathrm{n}} C'$ consists in decomposing a thread into two concurrent threads, we have $\Pi(C) \equiv \Pi(C')$ thanks to the associativity and commutativity congruence rules;
   (b) in the case that $C \xrightarrow{\varepsilon}_{\mathrm{n}} C'$ consists in removing a thread the syntactic component of which is the empty agent, we have $\Pi(C) \equiv \Pi(C')$ thanks to the "end of an agent" congruence rule;

(c) in the case that $C \xrightarrow{\varepsilon}_{\mathrm{n}} C'$ consists in opening a channel, we have $\Pi(C) \equiv \Pi(C')$ thanks to the extrusion rule, the swapping rule, and the consistency of the marker allocation scheme (Cf. Prop. 4).

(2) let C' be a non-standard configuration such that $C \xrightarrow{\lambda}_{\mathrm{n}} C'$, we have $\Pi(C) \xrightarrow{\lambda} \Pi(C')$: in the case that C only contains the threads involved in the non-standard computation step, the property is true by definition of the two relations and thanks to the fact that a standard computation step can be performed inside name restriction; then we use the fact that a standard computation step can be performed both within parallel composition and within name restriction to prove it in the general case.

(3) let P be a standard configuration such that $\Pi(C) \xrightarrow{\lambda} P$. The binary relation $\xrightarrow{\varepsilon}_{\mathrm{n}}$ is nœtherian and locally confluent. So, following [12], we can take a non-standard configuration D such that $C \xrightarrow{\varepsilon}_{\mathrm{n}}{}^{*} D$ and such that for any configuration E, $D \not\xrightarrow{\varepsilon}_{\mathrm{n}} E$. In accordance with Thm. 8.(1), we have $\Pi(C) \equiv \Pi(D)$, then since $\xrightarrow{\lambda}$ is compatible with $\equiv$, we get that $\Pi(D) \xrightarrow{\lambda} P$. Besides, we can deduce from the fact that D cannot be reduced by a computation step labelled with ε, that the syntactic component of every thread of D is either of the form $P \oplus Q$ or of the form aP. So the computation step $\Pi(D) \xrightarrow{\lambda} P$ can be lifted in the non-standard semantics, so that we can choose a non-standard configuration E such that $D \xrightarrow{\lambda}_{\mathrm{n}} E$ and $\Pi(E) \equiv P$. □

B Correspondence between the standard and the intermediate semantics

In this section we give the proof of Thm. 11, which formalizes the relation between the standard and the intermediate semantics.

Proposition 9 *For any non-standard configuration C, we have $C \Downarrow \bigcup_{t \in C} \beta^{\mathrm{i}}(t)$.*

PROOF. Since the relation β^{i} separately acts on each thread, it is enough to prove the property in the case that C is a singleton $\{(P, id, E)\}$. This is done by induction on the syntax of P:

- in the case that $P = 0$,
  we have $\{(0, id, E)\} \xrightarrow{\varepsilon}_{\mathrm{n}} \emptyset$ and $\beta^{\mathrm{i}}(0, id, E) = \emptyset$,
  so $\{(0, id, E)\} \Downarrow \beta^{\mathrm{i}}(0, id, E)$;
- in the case that $P = (\nu\ n)Q$, we have $\{(P, id, E)\} \xrightarrow{\varepsilon}_{\mathrm{n}} \{(Q, id, E[n \mapsto (n, id)])\}$,
  and by induction $\{(Q, id, E[n \mapsto (n, id)])\} \Downarrow \beta^{\mathrm{i}}(Q, id, E[n \mapsto (n, id)])$,

since $\beta^{\mathrm{i}}(P, id, E) = \beta^{\mathrm{i}}(Q, id, E[n \mapsto (n, id)])$,
we obtain $\{(P, id, E)\} \Downarrow \beta^{\mathrm{i}}(P, id, E)$;

- in the case that $P = (P_1 \mid P_2)$,
we have $\{(P_1 \mid P_2, id, E)\} \xrightarrow{\varepsilon}_{\mathrm{n}} \{(P_1, id, E_{|fn(P_1)}); (P_2, id, E_{|fn(P_2)})\}$,
and by induction $\forall i \in \{1; 2\},\ \{(P_i, id, E_{|fn(P_i)})\} \Downarrow \beta^{\mathrm{i}}(P_i, id, E_{|fn(P_i)})$,
since[9] $\beta^{\mathrm{i}}(P_1 \mid P_2, id, E) = \beta^{\mathrm{i}}(P_1, id, E_{|fn(P_1)}) \cup \beta^{\mathrm{i}}(P_2, id, E_{|fn(P_2)})$,
we obtain $\{(P_1 \mid P_2, id, E)\} \Downarrow \beta^{\mathrm{i}}(P_1 \mid P_2, id, E)$;
- otherwise we have $\beta^{\mathrm{i}}(P, id, E) = \{(P, id, E_{|fn(P)})\}$,
and $\{(P, id, E_{fn(P)})\} \Downarrow \{(P, id, E_{|fn(P)})\}$,
so $\{(P, id, E)\} \Downarrow \beta^{\mathrm{i}}(P, id, E)$. □

Theorem 11 *We have $\mathcal{S} \equiv \Pi(\mathcal{C}_0^{\mathrm{i}}(\mathcal{S}))$, and for all non-standard configurations C and for all word $u \in (\mathcal{L}^2 \cup \{\varepsilon; \oplus\})^*$ such that $\mathcal{C}_0^{\mathrm{i}}(\mathcal{S}) \xrightarrow{u}{}^*_{\mathrm{i}} C$, we have:*

(1) $\forall \lambda \in \mathcal{L}^2 \cup \{\oplus\},\ C \xrightarrow{\lambda}_{\mathrm{i}} C' \implies \Pi(C) \xrightarrow{\lambda} \Pi(C')$;

(2) $\forall \lambda \in \mathcal{L}^2 \cup \{\oplus\},\ \Pi(C) \xrightarrow{\lambda} P \implies \exists D,\ \begin{cases} C \xrightarrow{\lambda}_{\mathrm{i}} D \\ \Pi(D) \equiv P. \end{cases}$

PROOF. The proof of these properties mainly relies on Thm. 8 and the fact that $\xrightarrow{\lambda}_{\mathrm{i}}$ is defined as the composition of $\xrightarrow{\lambda}_{\mathrm{n}}$ and $\Downarrow$: in accordance with Thm. 8, we know that $\mathcal{S} = \Pi(\mathcal{C}_0^{\mathrm{n}}(\mathcal{S}))$, then by definition of $\mathcal{C}_0^{\mathrm{i}}(\mathcal{S})$, we have $\mathcal{C}_0^{\mathrm{n}}(\mathcal{S}) \xrightarrow{\varepsilon}{}^*_{\mathrm{n}} \mathcal{C}_0^{\mathrm{i}}(\mathcal{S})$, and so we can deduce from Thm. 8.(1) that $\mathcal{S} \equiv \Pi(\mathcal{C}_0^{\mathrm{i}}(\mathcal{S}))$. Then let us take a non-standard configuration C and word $u \in (\mathcal{L}^2 \cup \{\varepsilon; \oplus\})^*$ such that $\mathcal{C}_0^{\mathrm{i}}(\mathcal{S}) \xrightarrow{u}{}^*_{\mathrm{i}} C$,

(1) let C' be a non-standard configuration such that $C \xrightarrow{\lambda}_{\mathrm{i}} C'$, there exists another non-standard configuration D such that $C \xrightarrow{\lambda}_{\mathrm{n}} D \xrightarrow{\varepsilon}{}^*_{\mathrm{n}} C'$, we obtain from Thm. 8.(2) that $\Pi(C) \xrightarrow{\lambda} \Pi(D)$ and from Thm. 8.(1) that $\Pi(D) \equiv \Pi(C')$, so $\Pi(C) \xrightarrow{\lambda} \Pi(C')$;

(2) let P be a standard configuration such that $\Pi(C) \xrightarrow{\lambda} P$, following Thm. 8.(3), there exists two non-standard configurations D and E, such that $C \xrightarrow{\varepsilon}{}^*_{\mathrm{n}} D$, $D \xrightarrow{\lambda}_{\mathrm{n}} E$ and $\Pi(E) \equiv P$, moreover, since $\mathcal{C}_0^{\mathrm{i}}(\mathcal{S}) \xrightarrow{u}{}^*_{\mathrm{i}} C$, we know that for any non-standard configuration D', we have $C \xrightarrow{\varepsilon}\!\!\!\!\!\!/\ _{\mathrm{n}} D'$, this means that $D = C$, so we have $C \xrightarrow{\lambda}_{\mathrm{n}} E$ and $\Pi(E) \equiv P$, then we introduce the non-standard configuration F such that $E \Downarrow F$, we know from Thm. 8.(1) that $\Pi(E) \equiv \Pi(F)$, and by definition of $\xrightarrow{\lambda}_{\mathrm{i}}$ that $C \xrightarrow{\lambda}_{\mathrm{i}} F$, so we can conclude that $P \equiv \Pi(F)$ and $C \xrightarrow{\lambda}_{\mathrm{i}} F$. □

[9] We use the property that $\beta^{\mathrm{i}}(R, id, E) = \beta^{\mathrm{i}}(R, id, E_{|fn(R)})$ which can be easily proved for all agents R by induction on the syntax of R.

C Correspondence between the standard and the efficient semantics

In this section we give the proof of Thm. 13, which formalizes the relation between the standard and the efficient semantics.

Proposition 12 *For any non standard configuration C, we have:*

$$\{b \mid C \implies b\} = \left\{\bigcup Cont_t \mid \forall t \in C,\ Cont_t \in \beta(t)\right\}.$$

PROOF. Since the β relation separately acts on each thread, it is enough to prove the property in the case that C is a singleton $\{(P, id, E)\}$. This is done by induction on the syntax of P:

- in the case that $P = 0$, only the garbage collection rule applies to the thread (P, id, E) and no rule applies on the empty set, so we conclude that $\{b \mid \{(0, id, E)\} \implies b\} = \{\emptyset\} = \beta(0, id, E)$;
- in the case that $P = (\nu\ n)Q$, only the name restriction rule can apply: we have $\{(P, id, E)\} \xrightarrow{\varepsilon}_{\mathrm{n}} \{(Q, id, E[n \mapsto (n, id)])\}$, $\{b \mid \{Q, id, E[n \mapsto (n, id)]\} \implies b\} = \beta(Q, id, E[n \mapsto (n, id)])$ (by the induction hypothesis), and $\beta(P, id, E) = \beta(Q, id, E[n \mapsto (n, id)])$,
  so we obtain $\{b \mid \{(P, id, E)\} \implies b\} = \beta(P, id, E)$;
- in the case that $P = (P_1 \mid P_2)$, only the rule which decomposes the thread can apply: we have $\{(P_1 \mid P_2, id, E)\} \xrightarrow{\varepsilon}_{\mathrm{n}} \{(P_1, id, E_{|fn(P_1)}); (P_2, id, E_{|fn(P_2)})\}$, so, since $\dashrightarrow$ separately acts on each thread, we obtain $\{b \mid \{(P, id, E)\} \implies b\} = \{b_1 \cup b_2 \mid \forall i \in \{1; 2\},\ \{(P_i, id, E)\} \implies b_i\}$; moreover by induction we know that for i in the set $\{1; 2\}$, we have $\{b_i \mid \{(P_i, id, E_{|fn(P_i)})\} \implies b\} = \beta(P_i, id, E_{|fn(P_i)})$, then by definition of $\beta(P_1 \mid P_2, id, E)$, we obtain [10] $\{b \mid \{(P_1 \mid P_2, id, E)\} \implies b\} = \beta(P_1 \mid P_2, id, E)$;
- in the case that $P = (P_1 \oplus P_2)$, only the two choice rules can apply: we have either $\{(P_1 \oplus P_2, id, E)\} \xrightarrow{\oplus}_{\mathrm{n}} \{(P_1, id, E_{|fn(P_1)})\}$ or $\{(P_1 \oplus P_2, id, E)\} \xrightarrow{\oplus}_{\mathrm{n}} \{(P_2, id, E_{|fn(P_2)})\}$, so $\{b \mid \{(P_1 \oplus P_2, id, E)\} \implies b\} = \cup_{i \in \{1;2\}} \{b \mid \{(P_i, id, E)\} \implies b\}$; moreover by induction we know that for i in the set $\{1; 2\}$, we have that $\{b_i \mid \{(P_i, id, E_{|fn(P_i)})\} \implies b\} = \beta(P_i, id, E_{|fn(P_i)})$, so since [10] $\beta(P_1 \oplus P_2, id, E) = \beta(P_1, id, E_{|fn(P_1)}) \cup \beta(P_2, id, E_{|fn(P_2)})$, we obtain $\{b \mid \{(P_1 \oplus P_2, id, E)\} \implies b\} = \beta(P_1 \oplus P_2, id, E)$;
- otherwise we have $\beta(P, id, E) = \{\{(P, id, E_{|fn(P)})\}\}$,
  and $\forall b,\ \{(P, id, E_{fn(P)})\} \not\dashrightarrow b$, so $\{b \mid \{(P, id, E)\} \implies b\} = \beta(P, id, E)$.

Theorem 13 *For any initial non-standard configuration $C_0 \in \mathcal{C}_0^{\mathrm{e}}(\mathcal{S})$, there*

[10] We use the property that $\beta(R, id, E) = \beta(R, id, E_{|fn(R)})$ which can be easily proved for all agents R by induction on the syntax of R.

exists $k \in \mathbb{N}$ *such that* $\mathcal{S} \xrightarrow{\oplus^k}{}^* \Pi(C_0)$ *and for all non-standard configurations* C *and for all word* $u \in (\mathcal{L}^2)^*$ *such that* $C_0 \xrightarrow{u}{}^*_{\mathrm{e}} C$*, we have:*

(1) $\forall \lambda \in \mathcal{L}^2,\ C \xrightarrow{\lambda}{}_{\mathrm{e}} C' \implies \exists k \in \mathbb{N},\ \exists P,\ \Pi(C) \xrightarrow{\lambda} P \xrightarrow{\oplus^k}{}^* \Pi(C')$*;*

(2) $\forall \lambda \in \mathcal{L}^2,\ \Pi(C) \xrightarrow{\lambda} P \implies \exists D,\ \begin{cases} C \xrightarrow{\lambda}{}_{\mathrm{e}} D \\ \textit{and} \begin{cases} \exists k > 0,\ P \xrightarrow{\oplus^k}{}^+ \Pi(D) \\ \textit{or } P \equiv \Pi(D). \end{cases} \end{cases}$

PROOF. Let $C_0 \in \mathcal{C}^{\mathrm{e}}_0(\mathcal{S})$ be an initial configuration, since $\mathcal{C}^{\mathrm{e}}_0(\mathcal{S}) = \beta(\mathcal{S}, \varepsilon, \emptyset)$, we obtain that $\mathcal{C}^{\mathrm{n}}_0(\mathcal{S})$ $\dashrightarrow^*$ C_0, then by definition of $\dashrightarrow$ and thanks to Thm. 8.(1) and Thm. 8.(2), there exists $k \in \mathbb{N}$ such that $\Pi(\mathcal{C}^{\mathrm{n}}_0(\mathcal{S})) \xrightarrow{\oplus^k}{}^* \Pi(C_0)$, since $\mathcal{S} = \Pi(\mathcal{C}^{\mathrm{n}}_0(\mathcal{S}))$ we get that $\mathcal{S} \xrightarrow{\oplus^k}{}^* \Pi(C_0)$. Then let us take a non-standard configuration C and a word $u \in (\mathcal{L}^2)^*$ such that $C_0 \xrightarrow{u}{}^*_{\mathrm{e}} C$,

(1) let C' be a non-standard configuration such that $C \xrightarrow{\lambda}{}_{\mathrm{e}} C'$, by definition of $\xrightarrow{\lambda}{}_{\mathrm{e}}$, there exists another non-standard configuration D such that we have $C \xrightarrow{\lambda}{}_{\mathrm{n}} D \dashrightarrow^* C'$; moreover, we obtain from Thm. 8.(2) that $\Pi(C) \xrightarrow{\lambda} \Pi(D)$ and we deduce from Thm. 8.(1) and Thm. 8.(2) that there exists $k \in \mathbb{N}$, such that $\Pi(D) \xrightarrow{\oplus^k}{}^* \Pi(C')$;

(2) let P be a standard configuration such that $\Pi(C) \xrightarrow{\lambda} P$, following Thm. 8.(3), there exist two non-standard configurations D and E which satisfy $C \xrightarrow{\varepsilon}{}^*_{\mathrm{n}} D$, $D \xrightarrow{\lambda}{}_{\mathrm{n}} E$ and $\Pi(E) \equiv P$; since $\mathcal{C}^{\mathrm{i}}_0(\mathcal{S}) \xrightarrow{u}{}^*_{\mathrm{e}} C$, we know that for any non-standard configuration D', we have $C \not\dashrightarrow D'$, this means that $D = C$ and $\lambda \neq \oplus$, so we have $C \xrightarrow{\lambda}{}_{\mathrm{n}} E$ and $\Pi(E) \equiv P$, then we introduce the non-standard configuration F such that $E \implies F$, by definition of $\xrightarrow{\lambda}{}_{\mathrm{e}}$, we have $C \xrightarrow{\lambda}{}_{\mathrm{e}} F$; moreover we have either $E \xrightarrow{\varepsilon}{}^*_{\mathrm{n}} F$ and thanks to Thm. 8.(1) $\Pi(E) \equiv \Pi(F)$, or $E \dashrightarrow^* E' \xrightarrow{\oplus}{}_{\mathrm{n}} E'' \dashrightarrow^* F$, and thanks to Thm. 8.(1) and Thm. 8.(2), there exists $k > 0$ such that $\Pi(E) \xrightarrow{\oplus^+}{}^* \Pi(F)$. □

D Correspondence between the semantics of closed and open systems

In this section we give the proof of Thms. 22 and 23, which formalize the relation between the semantics of closed and open systems.

We first recall Def. 21:

Definition 21 *Computation sequence projection is then defined as follows:*

Let $\tau = C_0 \xrightarrow{\lambda_1}_{\mathrm{e}} ... \xrightarrow{\lambda_n}_{\mathrm{e}} C_n$ *be a non-standard computation sequence, with* $C_0 \in \mathcal{C}_0^{\mathrm{e}}(\mathcal{S})$*. We define the projection of* τ*,* $\Pi_\tau(\mathcal{S}_{\mathrm{I}}, \Phi_{\mathcal{M}}, \Phi_{\mathcal{N}})(\tau)$ *as the non-standard computation sequence:*

$$(A_0, U_0, F_0) \overset{\Pi_\lambda(\lambda_{a_1})}{\twoheadrightarrow} ... \overset{\Pi_\lambda(\lambda_{a_p})}{\twoheadrightarrow} (A_p, U_p, F_p)$$

of the open system $\mathcal{S}_{\mathrm{I}}$*, where*

- $a_1, ..., a_p$ *is the strictly ascending sequence of the elements of the set* $\{i \in [\![1; n]\!] \mid \lambda_i \in \mathcal{L}^2 \setminus (\mathcal{L} \setminus \mathcal{L}_{\mathrm{I}})^2\}$*;*
- *the initial configuration* (A_0, U_0, F_0) *is the following triple:*

$$(\Pi_{\mathrm{C}}^{\Phi_{\mathcal{M}}, \Phi_{\mathcal{N}}}(C_0), en, \{t_n \mid n \in \mathbb{N}\});$$

- *for* $k \in [\![1; p]\!]$*, the configuration* (A_k, U_k, F_k) *is then defined as follows:*
  - $A_k = \Pi_{\mathrm{C}}^{\Phi_{\mathcal{M}}, \Phi_{\mathcal{N}}}(C_{a_k})$,
  - $U_k = \begin{cases} U_{k-1} & \text{if } fst(\lambda_{a_k}) \in \mathcal{L}_{\mathrm{I}}, \\ U_{k-1} \cup \{\Pi_{\mathcal{N}}^{\Phi_{\mathcal{M}}, \Phi_{\mathcal{N}}}(E(x_r)) \mid r \in [\![1; n]\!]\} & \text{otherwise,} \end{cases}$

    where, in the last case, $t = (x!^j[x_1, ..., x_n]P, id, E)$ *denotes the unique thread in* $C_{a_{k-1}} \setminus C_{a_k}$ *which matches this notation;*
  - $F_k = \begin{cases} F_{k-1} \text{ if } snd(\lambda_{a_k}) \in \mathcal{L}_{\mathrm{I}} \text{ or if } \xrightarrow{\lambda_{a_k}}_{\mathrm{e}} \text{ is not a resource replication,} \\ F_{k-1} \setminus \{\Phi_{\mathcal{M}}(snd(\lambda_{a_k}), id) \mid (P, id, E) \in C_{a_k - 1} \setminus C_{a_k}\} \text{ otherwise.} \end{cases}$

Then we introduce some lemma to help proving the correspondence.

The following lemma shows that the extraction function β and the configuration projection commute:

Lemma 44 *Let* (P, id, E) *be a thread such that* P *is a sub-term of* $\mathcal{S}_{\mathrm{I}}$*, then we have:*

$$\beta(P, \Pi_{\mathcal{M}}^{\Phi_{\mathcal{M}}}(id), [x \mapsto \Pi_{\mathcal{N}}^{\Phi_{\mathcal{M}}, \Phi_{\mathcal{N}}}(E(x))]) = \Pi_{\mathrm{C}}^{\Phi_{\mathcal{M}}, \Phi_{\mathcal{N}}}(\beta(P, id, E)).$$

PROOF. This lemma can easily by proved by induction on the syntax of P. We use the fact that each sub-term of a sub-term of $\mathcal{S}_{\mathrm{I}}$ is also a sub-term of $\mathcal{S}_{\mathrm{I}}$. □

The following lemma establishes the fact that communications between two threads of the context leave the threads of the system $\mathcal{S}_{\mathrm{I}}$ unchanged:

Lemma 45 *Let* $\tau = C_0 \xrightarrow{\lambda_1}_{\mathrm{e}} ... \xrightarrow{\lambda_n}_{\mathrm{e}} C_n$ *be a non-standard computation sequence, with* $C_0 \in \mathcal{C}_0^{\mathrm{e}}(\mathcal{S})$*; we denote by* $(A_0, U_0, F_0)...(A_p, U_p, F_p)$ *the computation sequence* $\Pi_\tau(\mathcal{S}_{\mathrm{I}}, \Phi_{\mathcal{M}}, \Phi_{\mathcal{N}})(\tau)$*, then we have* $\Pi_{\mathrm{C}}^{\Phi_{\mathcal{M}}, \Phi_{\mathcal{N}}}(C_n) = A_p$*.*

PROOF. We prove Lem. 45 by induction over the integer $n - a_p$.

(1) in the case that $n = a_p$, we have $\Pi_{\mathrm{C}}^{\Phi_{\mathcal{M}},\Phi_{\mathcal{N}}}(C_n) = A_p$ by Def. 21;
(2) we now suppose that there exists $m_0 \in \mathbb{N}$ such that Lem. 45 is satisfied provided that $n - a_p < m_0$, we now prove it in the case that $n - a_p = m_0$: we know from the induction hypothesis that $\Pi_{\mathrm{C}}^{\Phi_{\mathcal{M}},\Phi_{\mathcal{N}}}(C_{n-1}) = A_p$, then we have $\lambda_n \in (\mathcal{L} \setminus \mathcal{L}_{\mathrm{I}})^2$, so we can conclude that both the set $C_{n-1} \setminus C_n$ and the set $C_n \setminus C_{n-1}$ only contain threads the label of which is in $\mathcal{L} \setminus \mathcal{L}_{\mathrm{I}}$, thus with respect with Def. 21, we obtain that $\Pi_{\mathrm{C}}^{\Phi_{\mathcal{M}},\Phi_{\mathcal{N}}}(C_n) = \Pi_{\mathrm{C}}^{\Phi_{\mathcal{M}},\Phi_{\mathcal{N}}}(C_{n-1})$, since $\Pi_{\mathrm{C}}^{\Phi_{\mathcal{M}},\Phi_{\mathcal{N}}}(C_{n-1}) = A_p$, we conclude that $\Pi_{\mathrm{C}}^{\Phi_{\mathcal{M}},\Phi_{\mathcal{N}}}(C_n) = A_p$. □

The following lemma establishes some soundness conditions:

Lemma 46 *Let $\tau = C_0 \xrightarrow{\lambda_1}{}_{\mathrm{e}} \ldots \xrightarrow{\lambda_n}{}_{\mathrm{e}} C_n$ be a non-standard computation sequence, with $C_0 \in \mathcal{C}_0^{\mathrm{e}}(\mathcal{S})$. We consider the following computation sequence:*

$$\Pi_\tau(\mathcal{S}_{\mathrm{I}}, \Phi_{\mathcal{M}}, \Phi_{\mathcal{N}})(\tau) = (A_0, U_0, F_0)...(A_{a_p}, U_{a_p}, F_{a_p}).$$

Then, $\forall (P, id, E) \in C_n$ with $lab(P) \notin \mathcal{L}_I$:

(1) $\forall x \in fn(P)$, we have $\Phi_{\mathcal{N}}(E(x)) \in U_p$;
(2) $\Phi_{\mathcal{M}}(lab(P), id) \in F_p$.

PROOF. These two properties are easily proved by induction on n.

(1) (a) in the case that $n = 0$, we have $p = 0$; let (P, id, E) be a thread in C_0 with $lab(P) \notin \mathcal{L}_{\mathrm{I}}$; we have $C_0 \in \mathcal{C}_0^{\mathrm{e}}(\mathcal{S})$, let x be a free name in P; since $lab(P) \notin \mathcal{L}_{\mathrm{I}}$, we have $x \notin \mathcal{N}_{\mathrm{I}}$; then $E(x) = \Pi_{\mathcal{N}}^{\Phi_{\mathcal{M}},\Phi_{\mathcal{N}}}(x, \varepsilon) = \Phi_{\mathcal{N}}(x, \varepsilon) \in en = U_0$;
   (b) we suppose that there exists $n_0 \in \mathbb{N}$ such that Prop. 46.(1) is satisfied for any n smaller than n_0. We now prove this property for $n = n_0 + 1$: let (P, id, E) in C_{n_0+1} with $lab(P) \notin \mathcal{L}_{\mathrm{I}}$, and x be a free name in P; we denote $(y, id_y) = E(x)$;
      - in the case that there exists j smaller than n_0 such that (P, id, E) in C_j: by the induction hypothesis, there exists o smaller than p such that $\Phi_{\mathcal{N}}(E(x)) \in U_o \subseteq U_p$;
      - in the case that $y \notin \mathcal{N}_{\mathrm{I}}$, we have $\Phi_{\mathcal{N}}(E(x)) \in en = U_0 \subseteq U_p$;
      - otherwise we necessarily have $\lambda_{n_0+1} \in ((\mathcal{L} \setminus \mathcal{L}_{\mathrm{I}}) \times \mathcal{L}_{\mathrm{I}})$, and there exists a thread $t = (z!^j[z_1, ..., z_n]P, id_t, E_t) \in C_{n_0} \setminus C_{n_0+1}$ such that $E(x) \in \{E_t(z_l) \mid l \in [\![1; n]\!]\}$, so by definition of U_p, we get that $\Phi_{\mathcal{N}}(E(x)) \in U_p$.
(2) (a) in the case that $n = 0$, we have $p = 0$ and $F_p = \{t_n \mid n \in \mathbb{N}\}$, so $\Phi_{\mathcal{M}}(\{(lab(P), id) \mid (P, id, E) \in C_0,\ lab(P) \notin \mathcal{L}_{\mathrm{I}}\}) \subseteq F_p$;

(b) we suppose that there exists $n_0 \in \mathbb{N}$ such that Prop. 46.(2) is satisfied for any n smaller than n_0. We now prove this property for $n = n_0+1$: let (P, id, E) in C_{n_0+1} with $lab(P) \notin \mathcal{L}_{\mathrm{I}}$;
- in the case that $(P, id, E) \in C_{n_0}$ and $\lambda_{n_0+1} \in (\mathcal{L} \setminus \mathcal{L}_{\mathrm{I}})^2$: we have $\Phi_{\mathcal{M}}(lab(P), id) \in F_p$, by the induction hypothesis;
- in the case that $(P, id, E) \in C_{n_0}$ and $\lambda_{n_0+1} \notin (\mathcal{L} \setminus \mathcal{L}_{\mathrm{I}})^2$: we know by the induction hypothesis that $\Phi_{\mathcal{M}}(lab(P), id) \in F_{p-1}$, then the marker $\Phi_{\mathcal{M}}(lab(P), id)$ is also in F_p, otherwise we would have $(P, id, E) \in C_{n_0} \setminus C_{n_0+1}$, which is absurd;
- in the case that $(P, id, E) \notin C_{n_0}$, we have $\Phi_{\mathcal{M}}(lab(P), id) \in F_p$, otherwise there would exist an integer $i < p$ such that $\Phi_{\mathcal{M}}(lab(P), id) \in F_i \setminus F_{i+1}$, and so there would exist an integer $j < n_0 + 1$ such that $(P, id, E) \in C_j \setminus C_{j+1}$ which is in contradiction with the fact that $(P, id, E) \in C_{n_0+1}$ thanks to Prop. 4.

So in any case, we have $\Phi_{\mathcal{M}}(lab(P), id) \in F_p$. □

Theorem 22 (Soundness) *Let $\tau = C_0...C_n$ be a non-standard computation sequence of the following closed system:*

$$(\nu\ c_1)...(\nu\ c_k)(\mathcal{S}_{\mathrm{I}}(c_{i_1}, ..., c_{i_n}) \mid \mathcal{S}_{\mathrm{c}}(c_{j_1}, ...c_{j_l})),$$

with $C_0 \in \mathcal{C}_0^{\mathrm{e}}(\mathcal{S})$. Then $\Pi_\tau(\mathcal{S}_{\mathrm{I}}, \Phi_{\mathcal{M}}, \Phi_{\mathcal{N}})(\tau) = (A_0, U_0, F_0)...(A_p, U_p, F_p)$ is a non-standard computation sequence of the open system $\mathcal{S}_{\mathrm{I}}$ and $(A_0, U_0, F_0) \in \mathcal{C}_0^{\mathrm{o}}(\mathcal{S})$.

PROOF. Soundness is ensured by construction: we prove Thm. 22 by induction on the length of the computation sequences:

(1) We first prove that $(A_0, U_0, F_0) \in \mathcal{C}_0^{\mathrm{o}}(\mathcal{S})$: we have $C_0 \in \mathcal{C}_0^{\mathrm{e}}(\mathcal{S})$; so $C_0 \in \beta(\mathcal{S}, \varepsilon, \emptyset)$; so by definition of β, we have $C_0 \in \{A \cup B \mid A \in \beta(\mathcal{S}_{\mathrm{I}}, \varepsilon, [c_{i_k} \mapsto (c_{i_k}, \varepsilon)]),\ B \in \beta(\mathcal{S}_{\mathrm{c}}, \varepsilon, [c_{j_k} \mapsto (c_{j_k}, \varepsilon)])\}$; then we decompose C_0 into $A \cup B$ with $A \in \beta(\mathcal{S}_{\mathrm{I}}, \varepsilon, [c_{i_k} \mapsto (c_{i_k}, \varepsilon)])$ and with $B \in \beta(\mathcal{S}_{\mathrm{c}}, \varepsilon, [c_{j_k} \mapsto (c_{j_k}, \varepsilon)])$. Moreover, we have $A_0 = \Pi_{\mathrm{C}}^{\Phi_{\mathcal{M}}, \Phi_{\mathcal{N}}}(C_0) = \Pi_{\mathrm{C}}^{\Phi_{\mathcal{M}}, \Phi_{\mathcal{N}}}(A) \cup \Pi_{\mathrm{C}}^{\Phi_{\mathcal{M}}, \Phi_{\mathcal{N}}}(B)$; since $\Pi_{\mathrm{C}}^{\Phi_{\mathcal{M}}, \Phi_{\mathcal{N}}}(B) = \emptyset$, we have $A_0 = \Pi_{\mathrm{C}}^{\Phi_{\mathcal{M}}, \Phi_{\mathcal{N}}}(A)$. Thanks to Lem. 44, since $A \in \beta(\mathcal{S}_{\mathrm{I}}, \varepsilon, [c_{i_k} \mapsto (c_{i_k}, \varepsilon)])$, we obtain $A_0 \in \beta(\mathcal{S}_{\mathrm{I}}, \Pi_{\mathcal{M}}^{\Phi_{\mathcal{M}}}(id), [c_{i_k} \mapsto \Phi_{\mathcal{N}}(c_{i_k}, \varepsilon)])$; so, since $\Phi_{\mathcal{N}} : (\mathcal{N} \times \mathcal{M} \to en)$, we obtain the fact that $A_0 \in \bigcup\{\beta(\mathcal{S}_{\mathrm{I}}, \varepsilon, E) \mid E \in fn(P) \to en\}$; then since $U_0 = en$ and $F_0 = \{t_n \mid n \in \mathbb{N}\}$, we obtain, by definition of $\mathcal{C}_0^{\mathrm{o}}$, that $(A_0, U_0, F_0) \in \mathcal{C}_0^{\mathrm{o}}(\mathcal{S})$.

(2) We now assume that Thm. 22 is satisfied for any non-standard computation sequence τ containing at most n computation steps, and we prove that it is also satisfied for any non-standard computation sequence τ of containing $n + 1$ computation steps: let $\tau = C_0...C_n \xrightarrow{\lambda}_{\mathrm{e}} C_{n+1}$ be a non-standard computation sequence of length $n + 1$; by the induction

hypothesis $\Pi_\tau(\mathcal{S}_\mathrm{I}, \Phi_\mathcal{M}, \Phi_\mathcal{N})(C_0...C_n)$ is a non-standard computation sequence of the open system $\mathcal{S}_\mathrm{I}$, which we can denote:

$$(A_0, U_0, F_0)...(A_{a_p}, U_{a_p}, F_{a_p})$$

then we discuss several cases depending of $\lambda = (i, j)$:

(a) in the case that $\lambda \in (\mathcal{L} \setminus \mathcal{L}_\mathrm{I})^2$: by definition of $\Pi_\tau(\mathcal{S}_\mathrm{I}, \Phi_\mathcal{M}, \Phi_\mathcal{N})$, $\Pi_\tau(\mathcal{S}_\mathrm{I}, \Phi_\mathcal{M}, \Phi_\mathcal{N})(\tau) = \Pi_\tau(\mathcal{S}_\mathrm{I}, \Phi_\mathcal{M}, \Phi_\mathcal{N})((C_0, U_0, F_0)...(C_n, U_n, F_n))$; so $\Pi_\tau(\mathcal{S}_\mathrm{I}, \Phi_\mathcal{M}, \Phi_\mathcal{N})(\tau)$ is a non-standard computation sequence of the open system $\mathcal{S}_\mathrm{I}$;

(b) in the case that $\lambda \in \mathcal{L}_\mathrm{I}^2$ and $\xrightarrow{\lambda}_\mathrm{e}$ is a communication rule: there exist two threads $t_? = (y?^i[\overline{y}]P, id_?, E_?)$ and $t_! = (x!^j[\overline{x}], id_!, E_!)$ in C_n which satisfy that $E_?(y) = E_!(x)$ and $\lambda = (i, j)$, and two continuations $Cont_? \in \beta(P, id_?, E_?[y_i \mapsto x_i])$ and $Cont_! \in \beta(Q, id_!, E_!)$ such that $C_{n+1} = (C_n \setminus \{t_?; t_!\}) \cup Cont_? \cup Cont_!$; we have:

$$\begin{cases} (\Pi_\lambda(t_?), \Pi_\lambda(t_!)) \in (A_p)^2 \text{ (thanks to Lem. 45 and since } (i,j) \in \mathcal{L}_\mathrm{I}^2), \\ \Pi_\mathcal{N}^{\Phi_\mathcal{M}, \Phi_\mathcal{N}}(E_?(y)) = \Pi_\mathcal{N}^{\Phi_\mathcal{M}, \Phi_\mathcal{N}}(E_!(x)) \text{ (since } E_?(y) = E_!(x)), \\ \Pi_\mathrm{t}^{\Phi_\mathcal{M}, \Phi_\mathcal{N}}(Cont_?) \in \beta(P, \Pi_\mathcal{M}^{\Phi_\mathcal{M}}(id_?), E'_?) \text{ (thanks to Lem. 44)}, \\ \qquad \text{where } E'_? = [x \mapsto \Pi_\mathcal{N}^{\Phi_\mathcal{M}, \Phi_\mathcal{N}}(E_?[y_k \mapsto E_!(x_k)](x))] \\ \Pi_\mathrm{t}^{\Phi_\mathcal{M}, \Phi_\mathcal{N}}(Cont_!) \in \beta(Q, \Pi_\mathcal{M}^{\Phi_\mathcal{M}}(id_!), [x \mapsto \Pi_\mathcal{N}^{\Phi_\mathcal{M}, \Phi_\mathcal{N}}(E_!(x))]), \\ \qquad\qquad\qquad\qquad \text{(thanks to Lem. 44)}; \end{cases}$$

so $(A_p, U_p, F_p) \overset{\lambda}{\hookrightarrow} (A_p \setminus \{\Pi_\mathrm{t}^{\Phi_\mathcal{M}, \Phi_\mathcal{N}}(t_?); \Pi_\mathrm{t}^{\Phi_\mathcal{M}, \Phi_\mathcal{N}}(t_!)\} \cup \Pi_\mathrm{t}^{\Phi_\mathcal{M}, \Phi_\mathcal{N}}(Cont_?) \cup \Pi_\mathrm{t}^{\Phi_\mathcal{M}, \Phi_\mathcal{N}}(Cont_!), U_p, F_p)$; then since $\Pi_\lambda(\lambda) = \lambda$, we can conclude that $(A_p, U_p, F_p) \overset{\Pi_\lambda(\lambda)}{\hookrightarrow} (\Pi_\mathrm{C}^{\Phi_\mathcal{M}, \Phi_\mathcal{N}}(C_{n+1}), U_p, F_p)$;

(c) in the case that $\lambda \in \mathcal{L}_\mathrm{I}^2$ and $\xrightarrow{\lambda}_\mathrm{e}$ is a replication rule: there exist two threads $t_? = (*y?^i[\overline{y}]P, id_?, E_?)$ and $t_! = (x!^j[\overline{x}], id_!, E_!)$ in C_n which satisfy that $E_?(y) = E_!(x)$ and $\lambda = (i, j)$, and two continuations $Cont_? \in \beta(P, N((i,j), id_?, id_!), E_?[y_i \mapsto x_i])$ and $Cont_! \in \beta(Q, id_!, E_!)$ such that $C_{n+1} = (C_n \setminus \{t_!\}) \cup Cont_? \cup Cont_!$; since $(i, j) \in \mathcal{L}_\mathrm{I}^2$, we have $\Pi_\mathcal{M}^{\Phi_\mathcal{M}}(N((i,j), id_?, id_!)) = N((i,j), \Pi_\mathcal{M}^{\Phi_\mathcal{M}}(id_?), \Pi_\mathcal{M}^{\Phi_\mathcal{M}}(id_!))$; then:

$$\begin{cases} (\Pi_\lambda(t_?), \Pi_\lambda(t_!)) \in (A_p)^2 \text{ (thanks to Lem. 45 and since } (i,j) \in \mathcal{L}_\mathrm{I}^2), \\ \Pi_\mathcal{N}^{\Phi_\mathcal{M}, \Phi_\mathcal{N}}(E_?(y)) = \Pi_\mathcal{N}^{\Phi_\mathcal{M}, \Phi_\mathcal{N}}(E_!(x)) \text{ (since } E_?(y) = E_!(x)), \\ \Pi_\mathrm{t}^{\Phi_\mathcal{M}, \Phi_\mathcal{N}}(Cont_?) \in \beta(P, id_*, E_*) \text{ (thanks to Lem. 44)}, \\ \qquad \text{where } id_* = N((i,j), \Pi_\mathcal{M}^{\Phi_\mathcal{M}}(id_?), \Pi_\mathcal{M}^{\Phi_\mathcal{M}}(id_!)) \\ \qquad \text{and } E_* = [x \mapsto \Pi_\mathcal{N}^{\Phi_\mathcal{M}, \Phi_\mathcal{N}}(E_?[y_k \mapsto E_!(x_k)](x))], \\ \Pi_\mathrm{t}^{\Phi_\mathcal{M}, \Phi_\mathcal{N}}(Cont_!) \in \beta(Q, \Pi_\mathcal{M}^{\Phi_\mathcal{M}}(id_!), [x \mapsto \Pi_\mathcal{N}^{\Phi_\mathcal{M}, \Phi_\mathcal{N}}(E_!(x))]) \\ \qquad\qquad\qquad\qquad \text{(thanks to Lem. 44)}; \end{cases}$$

so we have $(A_p, U_p, F_p) \overset{\lambda}{\hookrightarrow} (A_p \setminus \{\Pi_{\mathrm{t}}^{\Phi_{\mathcal{M}},\Phi_{\mathcal{N}}}(t_!)\} \cup \Pi_{\mathrm{t}}^{\Phi_{\mathcal{M}},\Phi_{\mathcal{N}}}(Cont_?) \cup \Pi_{\mathrm{t}}^{\Phi_{\mathcal{M}},\Phi_{\mathcal{N}}}(Cont_!), U_p, F_p)$; then since $\Pi_\lambda(\lambda) = \lambda$, we can conclude that $(A_p, U_p, F_p) \overset{\Pi_\lambda(\lambda)}{\hookrightarrow} (\Pi_{\mathrm{C}}^{\Phi_{\mathcal{M}},\Phi_{\mathcal{N}}}(C_{n+1}), U_p, F_p)$;

(d) in the case that $i \in \mathcal{L}_{\mathrm{I}}$, $j \notin \mathcal{L}_{\mathrm{I}}$ and $\xrightarrow{\lambda}_{\mathrm{e}}$ is a communication rule: there exist two threads $t_? = (y?^i[\overline{y}]P, id_?, E_?)$ and $t_! = (x!^j[\overline{x}], id_!, E_!)$ in C_n which satisfy that $E_?(y) = E_!(x)$ and $\lambda = (i, j)$, and two continuations $Cont_? \in \beta(P, id_?, E_?[y_i \mapsto x_i])$ and $Cont_! \in \beta(Q, id_!, E_!)$ such that $C_{n+1} = (C_n \setminus \{t_?; t_!\}) \cup Cont_? \cup Cont_!$; we have:

$$\begin{cases} \Pi_\lambda(t_?) \in A_p \text{ (thanks to Lem. 45 and since } i \in \mathcal{L}_{\mathrm{I}}), \\ \Pi_{\mathcal{N}}^{\Phi_{\mathcal{M}},\Phi_{\mathcal{N}}}(E_?(y)) = \Pi_{\mathcal{N}}^{\Phi_{\mathcal{M}},\Phi_{\mathcal{N}}}(E_!(x)) \in U_p \text{ (since } E_?(y) = E_!(x), \\ \qquad\qquad \text{and thanks to Lem. 46.(1)}), \\ \Pi_{\mathrm{t}}^{\Phi_{\mathcal{M}},\Phi_{\mathcal{N}}}(Cont_?) \in \beta(P, \Pi_{\mathcal{M}}^{\Phi_{\mathcal{M}}}(id_?), E'_?) \text{ (thanks to Lem. 44)}, \\ \quad \text{where } E'_? = [x \mapsto \Pi_{\mathcal{N}}^{\Phi_{\mathcal{M}},\Phi_{\mathcal{N}}}(E_?[y_k \mapsto E_!(x_k)](x))] \\ \{\Pi_{\mathcal{N}}^{\Phi_{\mathcal{M}},\Phi_{\mathcal{N}}}(E_!(x_k))\} \subseteq U_p \text{ (thanks to Lem. 46.(1))}; \end{cases}$$

so $(A_p, U_p, F_p) \overset{(i,0)}{\hookrightarrow} (A_p \setminus \{\Pi_{\mathrm{t}}^{\Phi_{\mathcal{M}},\Phi_{\mathcal{N}}}(t_?)\} \cup \Pi_{\mathrm{t}}^{\Phi_{\mathcal{M}},\Phi_{\mathcal{N}}}(Cont_?), U_p, F_p)$; since $\Pi_\lambda(\lambda) = (i, 0)$, $(A_p, U_p, F_p) \overset{\Pi_\lambda(\lambda)}{\hookrightarrow} (\Pi_{\mathrm{C}}^{\Phi_{\mathcal{M}},\Phi_{\mathcal{N}}}(C_{n+1}), U_p, F_p)$;

(e) in the case that $i \in \mathcal{L}_{\mathrm{I}}$, $j \notin \mathcal{L}_{\mathrm{I}}$ and $\xrightarrow{\lambda}_{\mathrm{e}}$ is a replication rule: there exist two threads $t_? = (*y?^i[\overline{y}]P, id_?, E_?)$ and $t_! = (x!^j[\overline{x}], id_!, E_!)$ in C_n which satisfy that $E_?(y) = E_!(x)$ and $\lambda = (i, j)$, and two continuations $Cont_? \in \beta(P, N((i, j), id_?, id_!), E_?[y_i \mapsto x_i])$ and $Cont_! \in \beta(Q, id_!, E_!)$ such that $C_{n+1} = (C_n \setminus \{t_!\}) \cup Cont_? \cup Cont_!$; we have:

$$\begin{cases} \Pi_\lambda(t_?) \in A_p \text{ (thanks to Lem. 45 and since } i \in \mathcal{L}_{\mathrm{I}}), \\ \Pi_{\mathcal{N}}^{\Phi_{\mathcal{M}},\Phi_{\mathcal{N}}}(E_?(y)) = \Pi_{\mathcal{N}}^{\Phi_{\mathcal{M}},\Phi_{\mathcal{N}}}(E_!(x)) \in U_p \text{ (since } E_?(y) = E_!(x) \\ \qquad\qquad \text{and thanks to Lem. 46.(1))}, \\ \Pi_{\mathrm{t}}^{\Phi_{\mathcal{M}},\Phi_{\mathcal{N}}}(Cont_?) \in \beta(P, id_*, E_*), \text{ (thanks to Lem. 44)}, \\ \quad \text{where } id_* = N((i, 0), \Pi_{\mathcal{M}}^{\Phi_{\mathcal{M}}}(id_?), \Phi_{\mathcal{M}}(j, id_!)) \\ \quad \text{and } E_* = [x \mapsto \Pi_{\mathcal{N}}^{\Phi_{\mathcal{M}},\Phi_{\mathcal{N}}}(E_?[y_k \mapsto E_!(x_k)](x))], \\ \{\Pi_{\mathcal{N}}^{\Phi_{\mathcal{M}},\Phi_{\mathcal{N}}}(E_!(x_k))\} \subseteq U_p \text{ (thanks to Lem. 46.(1))}, \\ \Phi_{\mathcal{M}}(j, id_!) \in F_p \text{ (thanks to the Lem. 46.(2))}; \end{cases}$$

so $(A_p, U_p, F_p) \overset{(i,0)}{\hookrightarrow} (A_p \cup \Pi_{\mathrm{t}}^{\Phi_{\mathcal{M}},\Phi_{\mathcal{N}}}(Cont_?), U_p, F_p \setminus \{\Phi_{\mathcal{M}}(j, id_!)\})$; then since we gave $\Pi_\lambda(\lambda) = (i, 0)$, we can conclude that $(A_p, U_p, F_p) \overset{\Pi_\lambda(\lambda)}{\hookrightarrow} (\Pi_{\mathrm{C}}^{\Phi_{\mathcal{M}},\Phi_{\mathcal{N}}}(C_{n+1}), U_p, F_p \setminus \{\Phi_{\mathcal{M}}(j, id_!)\})$;

(f) in the case that $i \notin \mathcal{L}_{\mathrm{I}}$, $j \in \mathcal{L}_{\mathrm{I}}$ and $\xrightarrow{\lambda}_{\mathrm{e}}$ is a communication rule:

there exist two threads $t_? = (y?^i[\overline{y}]P, id_?, E_?)$ and $t_! = (x!^j[\overline{x}], id_!, E_!)$ in C_n which satisfy that $E_?(y) = E_!(x)$ and $\lambda = (i,j)$, and two continuations $Cont_? \in \beta(P, id_?, E_?[y_i \mapsto x_i])$ and $Cont_! \in \beta(Q, id_!, E_!)$ such that $C_{n+1} = (C_n \setminus \{t_?; t_!\}) \cup Cont_? \cup Cont_!$; we have:

$$\begin{cases} \Pi_\lambda(t_!) \in A_p \text{ (thanks to Lem. 45 and since } j \in \mathcal{L}_{\mathrm{I}}), \\ \Pi^{\Phi_\mathcal{M},\Phi_\mathcal{N}}_\mathcal{N}(E_!(y)) = \Pi^{\Phi_\mathcal{M},\Phi_\mathcal{N}}_\mathcal{N}(E_?(x)) \in U_p \text{ (since } E_?(y) = E_!(x) \\ \qquad\qquad \text{and thanks to Lem. 46.(1)}), \\ \Pi^{\Phi_\mathcal{M},\Phi_\mathcal{N}}_{\mathrm{t}}(Cont_!) \in \beta(Q, \Pi^{\Phi_\mathcal{M}}_\mathcal{M}(id_!), [x \mapsto \Pi^{\Phi_\mathcal{M},\Phi_\mathcal{N}}_\mathcal{N}(E_!(x))), \\ \qquad\qquad \text{(thanks to Lem. 44)}; \end{cases}$$

so $(A_p, U_p, F_p) \overset{(0,j)}{\hookrightarrow} (A_p \setminus \{\Pi^{\Phi_\mathcal{M},\Phi_\mathcal{N}}_{\mathrm{t}}(t_!)\} \cup \Pi^{\Phi_\mathcal{M},\Phi_\mathcal{N}}_{\mathrm{t}}(Cont_!), U_p \cup \{\Pi^{\Phi_\mathcal{M},\Phi_\mathcal{N}}_\mathcal{N}(E_!(x_k))\}, F_p)$; since $\Pi_\lambda(\lambda) = (0,j)$, we can conclude that:

$$(A_p, U_p, F_p) \overset{\Pi_\lambda(\lambda)}{\hookrightarrow} (\Pi^{\Phi_\mathcal{M},\Phi_\mathcal{N}}_{\mathrm{C}}(C_{n+1}), U_p \cup \{\Pi^{\Phi_\mathcal{M},\Phi_\mathcal{N}}_\mathcal{N}(E_!(x_k))\}, F_p);$$

(g) in the case that $i \notin \mathcal{L}_{\mathrm{I}}$, $j \in \mathcal{L}_{\mathrm{I}}$ and $\xrightarrow{\lambda}_{\mathrm{e}}$ is a replication rule: there exist two threads $t_? = (*y?^i[\overline{y}]P, id_?, E_?)$ and $t_! = (x!^j[\overline{x}], id_!, E_!)$ in C_n which satisfy that $E_?(y) = E_!(x)$ and $\lambda = (i,j)$, and two continuations $Cont_? \in \beta(P, N((i,j), id_?, id_!), E_?[y_i \mapsto x_i])$ and $Cont_! \in \beta(Q, id_!, E_!)$ such that $C_{n+1} = (C_n \setminus \{t_?; t_!\}) \cup Cont_? \cup Cont_!$; we have:

$$\begin{cases} \Pi_\lambda(t_!) \in A_p \text{ (thanks to Lem. 45 and since } j \in \mathcal{L}_{\mathrm{I}}), \\ \Pi^{\Phi_\mathcal{M},\Phi_\mathcal{N}}_\mathcal{N}(E_!(y)) = \Pi^{\Phi_\mathcal{M},\Phi_\mathcal{N}}_\mathcal{N}(E_?(x)) \in U_p, \text{ (since } E_?(y) = E_!(x) \\ \qquad\qquad \text{and thanks to Lem. 46.(1)}), \\ \Pi^{\Phi_\mathcal{M},\Phi_\mathcal{N}}_{\mathrm{t}}(Cont_!) \in \beta(Q, \Pi^{\Phi_\mathcal{M}}_\mathcal{M}(id_!), [x \mapsto \Pi^{\Phi_\mathcal{M},\Phi_\mathcal{N}}_\mathcal{N}(E_!(x))]), \\ \qquad\qquad \text{(thanks to Lem. 44)}; \end{cases}$$

so $(A_p, U_p, F_p) \overset{(0,j)}{\hookrightarrow} (A_p \setminus \{\Pi^{\Phi_\mathcal{M},\Phi_\mathcal{N}}_{\mathrm{t}}(t_!)\} \cup \Pi^{\Phi_\mathcal{M},\Phi_\mathcal{N}}_{\mathrm{t}}(Cont_!), U_p \cup \{\Pi^{\Phi_\mathcal{M},\Phi_\mathcal{N}}_\mathcal{N}(E_!(x_k))\}, F_p)$; since $\Pi_\lambda(\lambda) = (0,j)$, we get $(A_p, U_p, F_p) \overset{\Pi_\lambda(\lambda)}{\hookrightarrow} (A_{p+1}, U_p \cup \{\Pi^{\Phi_\mathcal{M},\Phi_\mathcal{N}}_\mathcal{N}(E_!(x_k))\}, F_p)$. □

Lemma 47 *Let $\mathcal{S}_*$ be a closed system of the following form:*

$$(\nu\, \overline{c})(\mathcal{S}_{\mathrm{I}}(c_{i_1}, ..., c_{i_p}) \mid \mathcal{S}_{\mathrm{c}}(c_{j_1}, ..., c_{j_q})),$$

we denote by $\mathcal{L}_{\mathrm{I}}$ the set of the label occurring in $\mathcal{S}_{\mathrm{I}}$. Let $\tau = C_0 \longrightarrow^*_{\mathrm{e}} C_n$ be a non-standard computation sequence, with $C_0 \in \mathcal{C}^{\mathrm{e}}_0(\mathcal{S})$ and $\Phi_\mathcal{M} : \mathcal{L} \times \mathcal{M} \to \{t_n \mid n \in \mathbb{N}\}$ and $\Phi_\mathcal{N} : \mathcal{N} \times \mathcal{M} \to en$ be two one-to-one maps, then:

(1) for any marker id occurring in a state of the computation sequence τ such that id matches $N((i,j), id_?, id_!)$, we have:

- $i \in \mathcal{L}_{\mathrm{I}} \implies \begin{cases} \textit{either } id_? = \varepsilon, \\ \textit{or } id_? \textit{ matches } N((i',j'), id'_?, id'_!) \textit{ where } i' \in \mathcal{L}_{\mathrm{I}}; \end{cases}$
- $j \in \mathcal{L}_{\mathrm{I}} \implies \begin{cases} \textit{either } id_! = \varepsilon, \\ \textit{or } id_! \textit{ matches } N((i',j'), id'_?, id'_!) \textit{ where } i' \in \mathcal{L}_{\mathrm{I}}; \end{cases}$

(2) let id_1 and id_2 be two markers occurring in a state of the computation sequence τ such that $\Pi_{\mathcal{M}}^{\Phi_{\mathcal{M}}}(id_1) = \Pi_{\mathcal{M}}^{\Phi_{\mathcal{M}}}(id_2)$, then $id_1 = id_2$;

(3) let c_1 and c_2 be two names in $\mathcal{N} \times \mathcal{M}$ occurring in a state of the computation sequence τ such that $\Pi_{\mathcal{N}}^{\Phi_{\mathcal{M}},\Phi_{\mathcal{N}}}(c_1) = \Pi_{\mathcal{N}}^{\Phi_{\mathcal{M}},\Phi_{\mathcal{N}}}(c_2)$, then $c_1 = c_2$.

PROOF.

(1) Prop. (1) can be proved using both the fact that:
   (a) if the label i of a thread t is in the set $\mathcal{L}_{\mathrm{I}}$, then either its marker is ε, or there exists a marker i' such that the syntactic agent labelled with i' contains the sub-term labelled with i, and that the marker of the thread t is necessarily of the form $N((i',j), id_?, id_!)$ where $i' \in \mathcal{L}_{\mathrm{I}}$,
   (b) a marker of the form $N((i,j), id_?, id_!)$ can be created only when a thread the label of which is i and the marker of which is $id_?$ interacts with a thread the label of which is j and the marker of which is $id_!$;

(2) Prop. (2) can be proved by induction over the height of the markers; this induction only uses Prop. (1) and the fact that the subtree of a marker occurring in a state of τ necessarily occurs in a previous state of τ;

(3) Prop. (3) follows from Prop. (2) and the fact that whenever a channel name has been opened by a restriction in $\mathcal{S}_{\mathrm{I}}$, its marker is also the marker of a former thread the label of which is in $\mathcal{L}_{\mathrm{I}}$. □

Theorem 23 (Completeness) *Let τ' be the non-standard computation sequence of an open system $\mathcal{S}_{\mathrm{I}}$, that we denote by:*

$$(C_0, U_0, F_0) \overset{(i_1,j_1)}{\curvearrowright} \ldots \overset{(i_n,j_n)}{\curvearrowright} (C_n, U_n, F_n),$$

where $(C_0, U_0, F_0) \in \mathcal{C}_0^{\mathrm{o}}(\mathcal{S}_{\mathrm{I}})$.

Then there exists:

- *a closed system $\mathcal{S}_* = (\nu\, \overline{c})(\mathcal{S}_{\mathrm{I}}(c_{i_1}, ..., c_{i_n}) \mid \mathcal{S}_{\mathrm{c}}(c_{j_1}, ...c_{j_l}))$,*
- *two one-to-one functions $\Phi_{\mathcal{N}}$ and $\Phi_{\mathcal{M}}$,*
- *a non-standard computation sequence τ of the system $\mathcal{S}_*$,*

such that $\Pi_\tau(\mathcal{S}_{\mathrm{I}}, \Phi_{\mathcal{M}}, \Phi_{\mathcal{N}})(\tau) = \tau'$.

We will prove the following stronger result:

Proposition 48 *Let τ' be the non-standard computation sequence of an open system $\mathcal{S}_{\mathrm{I}}$, defined as follows:*

$$(A_0, U_0, F_0) \overset{(i_1,j_1)}{\looparrowright} \ ... \ \overset{(i_n,j_n)}{\looparrowright} \ (A_n, U_n, F_n),$$

where $(A_0, U_0, F_0) \in \mathcal{C}_0^o(\mathcal{S}_{\mathrm{I}})$. Let $\mathcal{L}_{\mathrm{I}} \subseteq \mathcal{L}$ be the subset of the labels occurring in $\mathcal{S}_{\mathrm{I}}$ and $\mathcal{N}_{\mathrm{I}} \subseteq \mathcal{N}$ be the subset of the names occurring in name restrictions of $\mathcal{S}_{\mathrm{I}}$.

Then there exists:

- *a closed system of the form:*

$$\mathcal{S}_* = (\nu\ unsafe)(\nu\ x_1)...(\nu\ x_p)$$
$$(unsafe![x_1] \ |...|\ unsafe![x_p] \mid \mathcal{S}_I(x_{i_1}, ..., x_{i_n}) \mid \mathcal{S}_c(unsafe))$$

  where
  $\mathcal{S}_{\mathrm{c}} = (\nu\ new)$
  ($\boldsymbol{new} \mid \boldsymbol{repli}$
  $\mid \boldsymbol{spy}_0 \ |...|\ \boldsymbol{spy}_n$
  $\mid \boldsymbol{spoil}_0 \mid ... \mid \boldsymbol{spoil}_n$
  $\mid new![]$
)

  and
  - $\boldsymbol{new} := *new?[]((\nu\ channel)(unsafe![channel] \mid new![]))$
  - $\boldsymbol{repli} := *unsafe?[x](unsafe![x] \mid unsafe![x])$
  - $\boldsymbol{spy}_i := *unsafe?[c]c?[y_1, ..., y_i](unsafe![y_1] \mid ... \mid unsafe![y_i])$
  - $\boldsymbol{spoil}_i := *unsafe?[c]unsafe?[x_1]...unsafe?[x_i]c![x_1, ..., x_i]$
- *a non-standard computation sequence τ of the closed system $\mathcal{S}_*$,*
- *two into maps $\Phi_{\mathcal{N}}$ and $\Phi_{\mathcal{M}}$, such that:*
  - $\Phi_{\mathcal{N}} : \left\{ (y, id_y) \;\middle|\; \begin{array}{l} y \notin \mathcal{N}_{\mathrm{I}},\ \exists (P, id, E) \in \bigcup C_i,\ lab(P) \in \mathcal{L}_{\mathrm{I}}, \\ \exists x \in fn(P),\ E(x) = (y, id_y) \end{array} \right\} \rightarrow en;$
  - $\Phi_{\mathcal{M}} : \left\{ id \;\middle|\; \begin{array}{l} \exists (P, N((i,0), id_?, id_!), E) \in \bigcup C_i, \\ lab(P) \in \mathcal{L}_{\mathrm{I}} \end{array} \right\} \rightarrow \{t_n \mid n \in \mathbb{N}\}.$

such that:

(1) $\Pi_\tau(\mathcal{S}_{\mathrm{I}}, \Phi_{\mathcal{M}}, \Phi_{\mathcal{N}})(\tau) = \tau'$ [11] ;

[11] we make abusively no distinction neither between $\Phi_{\mathcal{N}}$ and any one-to-one extension of it defined over the set $\mathcal{N} \times \mathcal{M}$, nor between $\Phi_{\mathcal{M}}$ and any one-to-one extension of it defined over the set $\mathcal{M}$

(2) for any unsafe name $(x, id_x) \in U_n$, that occurs in a state of τ', there exists a thread in the last state of τ of the form $(unsafe![y], id, E)$ with $\Pi_{\mathcal{N}}^{\Phi_{\mathcal{M}},\Phi_{\mathcal{N}}}(E(y)) = (x, id_x)$.

PROOF. We prove Prop. 48 by induction on the length of the computation sequence τ':

(1) in the case that τ' is of the form $(A_0, U_0, F_0) \in \mathcal{C}_0^{\mathrm{o}}(\mathcal{S}_{\mathrm{I}})$, there exists $E \in fn(\mathcal{S}_{\mathrm{I}}) \to en$ such that $A_0 \in \beta(\mathcal{S}_{\mathrm{I}}, \varepsilon, E)$; we also have $U_0 = en$ and $F_0 = \{t_n \mid n \in \mathbb{N}\}$;
we take $\Phi_{\mathcal{M}} : \emptyset \to \{t_n \mid n \in \mathbb{N}\}$; we denote by e_1,...,e_u the elements of the set $\{E(x) \mid x \in fn(\mathcal{S}_{\mathrm{I}})\}$, we take $\Phi_{\mathcal{N}} : \{c_{i_k} \mid k \in [\![1; u]\!]\} \to en$, such that for any $k \in [\![1; u]\!]$, we have $\Phi_{\mathcal{N}}(c_{i_k}) = e_i$; thanks to Lem. 44, we have $\beta(\mathcal{S}_{\mathrm{I}}, \varepsilon, E) = \Pi_{\mathrm{C}}^{\Phi_{\mathcal{M}},\Phi_{\mathcal{N}}}(\beta(\mathcal{S}_*, \varepsilon, \Phi_{\mathcal{N}}^{-1} \circ E))$; so we can choose $C_0 \in \beta(\mathcal{S}_*, \varepsilon, \Phi_{\mathcal{N}}^{-1} \circ E)$ such that $A_0 = \Pi_{\mathrm{C}}^{\Phi_{\mathcal{M}},\Phi_{\mathcal{N}}}(C_0)$;
- we have $(A_0, U_0, F_0) = \Pi_\tau(\mathcal{S}_{\mathrm{I}}, \Phi_{\mathcal{M}}, \Phi_{\mathcal{N}})(C_0)$;
- for any unsafe name $(x, id_x) \in U_0$ occurring in A_0, there exists by construction a thread of the form (unsafe![y], id, E) with $\Pi_{\mathcal{N}}^{\Phi_{\mathcal{M}},\Phi_{\mathcal{N}}}(E(y)) = (x, id_x)$;

(2) in the case that τ' is of the form $(A_0, U_0, F_0)...(A_{p-1}, U_{p-1}, F_{p-1}) \overset{\lambda}{\Rightarrow} (A_p, U_p, F_p)$; we assume by the induction hypothesis that there exist:
- a non-standard computation sequence $\tau = C_0...C_n$ of the closed system $\mathcal{S}_*$ such that $C_0 \in \mathcal{C}_0^{\mathrm{e}}(\mathcal{S}_*)$,
- two into maps $\Phi_{\mathcal{N}}$ and $\Phi_{\mathcal{M}}$, such that:
  - $\Phi_{\mathcal{N}} : \left\{ (y, id_y) \;\middle|\; \begin{array}{l} y \notin \mathcal{N}_{\mathrm{I}},\ \exists (P, id, E) \in \bigcup C_i, \\ lab(P) \in \mathcal{L}_{\mathrm{I}}, \\ \exists x \in fn(P),\ E(x) = (y, id_y) \end{array} \right\} \to en,$
  - $\Phi_{\mathcal{M}} : \left\{ id \;\middle|\; \begin{array}{l} \exists (P, N((i,0), id_?, id_!), E) \in \bigcup C_i, \\ lab(P) \in \mathcal{L}_{\mathrm{I}} \end{array} \right\} \to \{t_n \mid n \in \mathbb{N}\};$

such that:
- $\Pi_\tau(\mathcal{S}_{\mathrm{I}}, \Phi_{\mathcal{M}}, \Phi_{\mathcal{N}})(\tau) = (A_0, U_0, F_0)...(A_{p-1}, U_{p-1}, F_{p-1})$;
- for any unsafe name $(x, id_x) \in U_n$ that occurs in a state of τ', there exists a thread in the last state of τ of the form (unsafe![y], id, E) with $\Pi_{\mathcal{N}}^{\Phi_{\mathcal{M}},\Phi_{\mathcal{N}}}(E(y)) = (x, id_x)$.

We proceed by case analysis in accordance with the kind of the next computation step:

(a) in the case that $(A_{p-1}, U_{p-1}, F_{p-1}) \overset{\lambda}{\Rightarrow} (A_p, U_p, F_p)$ is a communication rule: there exist two threads $t_? = (y?^i[\overline{y}]P, id_?, E_?)$ and $t_! = (x!^j[\overline{x}]Q, id_!, E_!)$ with $E_?(y) = E_!(x)$, two continuations $Cont_? \in \beta(P, id_?, E_?[y_i \mapsto E_!(x_i)])$ and $Cont_! \in \beta(Q, id_!, E_!)$ such that $A_p = (A_{p-1} \setminus \{t_?; t_!\}) \cup Cont_? \cup Cont_!$; moreover, we have $\Pi_{\mathrm{C}}^{\Phi_{\mathcal{M}},\Phi_{\mathcal{N}}}(C_n) =$

A_{p-1}, so by definition of $\Pi_{\mathrm{C}}^{\Phi_{\mathcal{M}},\Phi_{\mathcal{N}}}$ the state C_n contains two threads $t'_? = (y?^i[\overline{y}]P, id'_?, E'_?)$ and $t'_! = (x!^j[\overline{x}]Q, id'_!, E'_!)$ with $\Pi_{\mathcal{M}}^{\Phi_{\mathcal{M}}}(id'_?) = id_?$, $\Pi_{\mathcal{M}}^{\Phi_{\mathcal{M}}}(id'_!) = id_!$, $\Pi_{\mathcal{N}}^{\Phi_{\mathcal{M}},\Phi_{\mathcal{N}}} \circ E'_? = E_?$ and $\Pi_{\mathcal{N}}^{\Phi_{\mathcal{M}},\Phi_{\mathcal{N}}} \circ E'_! = E_!$; thanks to Lem. 44, there also exist $Cont'_? \in \beta(P, id'_?, E'_?[y_k \mapsto E'_!(x_k)])$ and $Cont'_! \in \beta(Q, id'_!, E'_!)$ such that $\Pi_{\mathrm{C}}^{\Phi_{\mathcal{M}},\Phi_{\mathcal{N}}}(Cont'_?) = Cont_?$ and $\Pi_{\mathrm{C}}^{\Phi_{\mathcal{M}},\Phi_{\mathcal{N}}}(Cont'_!) = Cont_!$;
we set $C_{n+1} = (C_n \setminus \{t'_?; t'_!\}) \cup Cont'_? \cup Cont'_!$;

- thanks to Lem. 47.(3), since $E_?(y) = E_!(x)$, we have $E'_?(y) = E'_!(x)$; so we have $C_n \xrightarrow{\lambda}_{\mathrm{e}} C_{n+1}$; moreover $\Pi_{\mathrm{C}}^{\Phi_{\mathcal{M}},\Phi_{\mathcal{N}}}(C_{n+1}) = A_p$; so $\Pi_\tau(\mathcal{S}_{\mathrm{I}}, \Phi_{\mathcal{M}}, \Phi_{\mathcal{N}})(C_0 ... C_n \xrightarrow{\lambda}_{\mathrm{e}} C_{n+1}) = \tau'$;
- the last computation step of τ' does not involve unsafe name and the computation step $C_n \xrightarrow{\lambda}_{\mathrm{e}} C_{n+1}$ does not consume any thread of the form $(\mathrm{unsafe}![y], id, E)$; so by the induction hypothesis, we get that for any unsafe name $(x, id_x) \in U_n$ occurring in a state of τ', there exists a thread in the last state of τ of the form $(\mathrm{unsafe}![y], id, E)$ with $\Pi_{\mathcal{N}}^{\Phi_{\mathcal{M}},\Phi_{\mathcal{N}}}(E(y)) = (x, id_x)$;

(b) in the case that $(A_{p-1}, U_{p-1}, F_{p-1}) \overset{\lambda}{\hookrightarrow} (A_p, U_p, F_p)$ is a resource replication; there are two threads $t_? = (*y?^i[\overline{y}]P, id_?, E_?)$ and $t_! = (x!^j[\overline{x}]Q, id_!, E_!)$ with $E_?(y) = E_!(x)$, two continuations $Cont_? \in \beta(P, N((i,j), id_?, id_!), E_?[y_i \mapsto E_!(x_i)])$ and $Cont_! \in \beta(Q, id_!, E_!)$ such that $A_p = (A_{p-1} \setminus \{t_!\}) \cup Cont_? \cup Cont_!$; moreover, we have $\Pi_{\mathrm{C}}^{\Phi_{\mathcal{M}},\Phi_{\mathcal{N}}}(C_n) = A_{p-1}$, so by definition of $\Pi_{\mathrm{C}}^{\Phi_{\mathcal{M}},\Phi_{\mathcal{N}}}$ there exist two threads $t'_? = (*y?^i[\overline{y}]P, id'_?, E'_?)$ and $t'_! = (x!^j[\overline{x}]Q, id'_!, E'_!)$ in C_n with $\Pi_{\mathcal{M}}^{\Phi_{\mathcal{M}}}(id'_?) = id_?$, $\Pi_{\mathcal{M}}^{\Phi_{\mathcal{M}}}(id'_!) = id_!$, $\Pi_{\mathcal{N}}^{\Phi_{\mathcal{M}},\Phi_{\mathcal{N}}} \circ E'_? = E_?$ and $\Pi_{\mathcal{N}}^{\Phi_{\mathcal{M}},\Phi_{\mathcal{N}}} \circ E'_! = E_!$; moreover we have $\Pi_{\mathcal{M}}^{\Phi_{\mathcal{M}}}(N((i,j), id'_?, id'_!)) = N((i,j), id_?, id_!)$; thanks to Lem. 44, there also exist two continuations $Cont'_! \in \beta(Q, id'_!, E'_!)$ and $Cont'_? \in \beta(P, N((i,j), id'_?, id'_!), E'_?[y_k \mapsto E'_!(x_k)])$ such that the properties $\Pi_{\mathrm{C}}^{\Phi_{\mathcal{M}},\Phi_{\mathcal{N}}}(Cont'_?) = Cont_?$ and $\Pi_{\mathrm{C}}^{\Phi_{\mathcal{M}},\Phi_{\mathcal{N}}}(Cont'_!) = Cont_!$ are satisfied;
we set $C_{n+1} = (C_n \setminus \{t'_!\}) \cup Cont'_? \cup Cont'_!$:

- thanks to Lem. 47.(3), since $E_?(y) = E_!(x)$, we have $E'_?(y) = E'_!(x)$; so we have $C_n \xrightarrow{\lambda}_{\mathrm{e}} C_{n+1}$; moreover $\Pi_{\mathrm{C}}^{\Phi_{\mathcal{M}},\Phi_{\mathcal{N}}}(C_{n+1}) = A_p$; so $\Pi_\tau(\mathcal{S}_{\mathrm{I}}, \Phi_{\mathcal{M}}, \Phi_{\mathcal{N}})(C_0 ... C_n \xrightarrow{\lambda}_{\mathrm{e}} C_{n+1}) = \tau'$;
- the last computation step of τ' does not involve unsafe name and the computation step $C_n \xrightarrow{\lambda}_{\mathrm{e}} C_{n+1}$ does not consume any thread of the form $(\mathrm{unsafe}![y], id, E)$; so by the induction hypothesis, we get that for any unsafe name $(x, id_x) \in U_n$ occurring in a state of τ', there exists a thread in the last state of τ of the form $(\mathrm{unsafe}![y], id, E)$ with $\Pi_{\mathcal{N}}^{\Phi_{\mathcal{M}},\Phi_{\mathcal{N}}}(E(y)) = (x, id_x)$;

(c) in the case that $(A_{p-1}, U_{p-1}, F_{p-1}) \overset{\lambda}{\hookrightarrow} (A_p, U_p, F_p)$ is a spied communication: there is a thread $t_! = (x!^j[x_1, ..., x_k]Q, id_!, E_!)$ with $E_!(x) \in U_{p-1}$, a continuation $Cont_! \in \beta(Q, id_!, E_!)$ such that $A_p = (A_{p-1} \setminus \{t_!\}) \cup Cont_!$, and $U_p = U_{p-1} \cup \{E_!(x_i) \mid i \in [\![1; k]\!]\}$; moreover, we have

$\Pi_{\mathrm{C}}^{\Phi_{\mathcal{M}},\Phi_{\mathcal{N}}}(C_n) = A_{p-1}$, so there exists a thread $t'_! = (x!^j[\overline{x}]Q, id'_!, E'_!)$ in C_n with $\Pi_{\mathcal{M}}^{\Phi_{\mathcal{M}}}(id'_!) = id_!$ and $\Pi_{\mathcal{N}}^{\Phi_{\mathcal{M}},\Phi_{\mathcal{N}}} \circ E'_! = E_!$; thanks to Lem. 44, there also exists $Cont''_! \in \beta(Q, id'_!, E'_!)$ such that $\Pi_{\mathrm{C}}^{\Phi_{\mathcal{M}},\Phi_{\mathcal{N}}}(Cont''_!) = Cont_!$; thanks to the induction hypothesis, there exists a thread $t_c = (unsafe![c], id_c, E_c)$ in C_n, such that $\Pi_{\mathcal{N}}^{\Phi_{\mathcal{M}},\Phi_{\mathcal{N}}}(E_c(c)) = E_!(x)$; we first use the resource **repli** to replicate the thread t_c: we obtain a configuration $C_{n+1} = C_n \setminus \{t_c\} \cup \{t'_c; t''_c\}$ where t'_c matches $(unsafe![c], id'_c, E'_c)$ and t''_c matches $(unsafe![c], id''_c, E''_c)$ with $E'_c(c) = E''_c(c) = E_c(c)$; we then replicate the resource $\mathbf{spy}_k$ by consuming the thread t'_c, and obtain the configuration $C_{n+2} = C_{n+1} \setminus \{t'_c\} \cup \{t_s\}$, where t_s matches $(c?[y_1, ..., y_k](unsafe![y_1] \mid ... \mid unsafe![y_k]), id_s, E_s)$ with $E_s(c) = E_c(c)$; we have $\Pi_{\mathcal{N}}^{\Phi_{\mathcal{M}},\Phi_{\mathcal{N}}}(E_s(c)) = \Pi_{\mathcal{N}}^{\Phi_{\mathcal{M}},\Phi_{\mathcal{N}}}(E'_!(x))$; so thanks to Lem. 47.(3), we deduce that $E_s(c) = E'_!(x)$; thus we perform the communication between the thread $t'_!$ and t_s to obtain the configuration $C_{n+3} = (C_{n+2} \setminus \{t_s; t'_!\}) \cup Cont''_! \cup \{t'_{u_i} \mid i \in [\![1; k]\!]\}$, where each t'_{u_i} matches $(unsafe![y_i], id'_i, E'_i)$ with $E'_i(y_i) = E'_!(x_i)$.

- the computation sequence $C_n \longrightarrow_{\mathrm{e}}^* C_{n+2}$ does not involve any thread of $\mathcal{S}_{\mathrm{I}}$; by Lem. 45, we have $\Pi_{\mathrm{C}}^{\Phi_{\mathcal{M}},\Phi_{\mathcal{N}}}(C_{n+2}) = A_{p-1}$; we conclude that $\Pi_{\mathrm{C}}^{\Phi_{\mathcal{M}},\Phi_{\mathcal{N}}}(C_{n+3}) = A_p$; moreover $C_{n+2} \xrightarrow{i',j}_{\mathrm{e}} C_{n+3}$ with $i' \notin \mathcal{L}_{\mathrm{I}}$; so $\Pi_\tau(\mathcal{S}_{\mathrm{I}}, \Phi_{\mathcal{M}}, \Phi_{\mathcal{N}})(C_0...C_{n+2} \xrightarrow{(i',j)}_{\mathrm{e}} C_{n+3}) = \tau'$.
- let (u, id_u) be an unsafe name occurring in a state of τ:
  - in the case that $(u, id_u) = E_!(x)$, t''_c is a thread of C_{n+3} and matches $(\text{unsafe}![y], id, E)$ with $\Pi_{\mathcal{N}}^{\Phi_{\mathcal{M}},\Phi_{\mathcal{N}}}(E(y)) = E_!(x)$;
  - in the case that $(u, id_u) = E_!(x_i)$, t_{u_i} is a thread of C_{n+3} and matches $(\text{unsafe}![y_i], id, E)$ which satisfies $\Pi_{\mathcal{N}}^{\Phi_{\mathcal{M}},\Phi_{\mathcal{N}}}(E(y_i)) = \Pi_{\mathcal{N}}^{\Phi_{\mathcal{M}},\Phi_{\mathcal{N}}}(E'_!(x_i))$; so $\Pi_{\mathcal{N}}^{\Phi_{\mathcal{M}},\Phi_{\mathcal{N}}}(E(y_i)) = E_!(x_i)$;
  - otherwise there is necessarily a thread in C_n which matches $(\text{unsafe}![y], id, E)$ with $\Pi_{\mathcal{N}}^{\Phi_{\mathcal{M}},\Phi_{\mathcal{N}}}(E(y)) = (x, id_x)$ and this thread is still in C_{n+3}.

(d) in the case that $(A_{p-1}, U_{p-1}, F_{p-1}) \overset{\lambda}{\hookrightarrow} (A_p, U_p, F_p)$ is a spoilt communication: there is a thread $t_? = (y?^i[y_1, ..., y_k]P, id_?, E_?)$ with $E_?(y) \in U_{p-1}$ (we set $(u_0, id_0) = E_?(y)$), there is a channel name $(u_i, id_i) \in U_{p-1}$ for each $i \in [\![1; k]\!]$, and a continuation $Cont_? \in \beta(P, id_?, E_?[y_i \mapsto (u_i, id_i)])$ such that $A_p = (A_{p-1} \setminus t_?) \cup Cont_?$, and $U_p = U_{p-1}$; moreover, we have $\Pi_{\mathrm{C}}^{\Phi_{\mathcal{M}},\Phi_{\mathcal{N}}}(C_n) = A_{p-1}$, so there exists a thread $t'_? = (y?^j[\overline{y}]P, id'_?, E'_?)$ in C_n with $\Pi_{\mathcal{M}}^{\Phi_{\mathcal{M}}}(id'_?) = id_?$ and $\Pi_{\mathcal{N}}^{\Phi_{\mathcal{M}},\Phi_{\mathcal{N}}} \circ E'_? = E_?$; we first deal with the names that the context sends into the system $\mathcal{S}_{\mathrm{I}}$ for the first time: for each element (u_i, id_i) of the set $\{(u_i, id_i) \mid i \in [\![1; k]\!]\}$ that does not occur in any state of τ, we create a new unsafe name using the resource **new**, as the result we get a fresh name $(channel, id_{chan_i})$ and, since (u_i, id_i) is necessarily in en, we extend the definition of $\Phi_{\mathcal{N}}$ with $\Phi_{\mathcal{N}}(channel, id_{chan_i}) = (u_i, id_i)$; we obtain a state C_{n+1} and an updated into map $\Phi_{\mathcal{N}}$, such that

there exists a thread $t_{c_i} = (\mathit{unsafe}![c], id_{c_i}, E_{c_i})$ in the state C_{n+1} such that $\Pi_{\mathcal{N}}^{\Phi_{\mathcal{M}},\Phi_{\mathcal{N}}}(E_{c_i}(c)) = (u_i, id_i)$, for each $i \in [\![0;k]\!]$; we then use the resource **repli** to replicate the threads t_{c_i} until we obtain a configuration $C_{n+2} = (C_{n+1} \setminus \{t_{c_i} \mid i \in [\![0;k]\!]\}) \cup \{t'_{c_i} \mid i \in [\![0;k]\!]\} \cup \{t''_{c_i} \mid 0 \in [\![1;k]\!]\}$ where t'_{c_i} matches $(\mathit{unsafe}![c], id'_{c_i}, E'_{c_i})$ and t''_{c_i} matches $(\mathit{unsafe}![c], id''_{c_i}, E''_{c_i})$ with $E'_{c_i}(c) = E''_{c_i}(c) = E_{c_i}(c)$ such that the threads t'_{c_i} and t''_{c_i} are all distinct from each other; we then replicate the resource $\mathbf{spoil}_k$ by consuming the thread t'_{c_0}, and obtain the configuration $C_{n+3} = (C_{n+2} \setminus \{t'_{c_0}\}) \cup \{t_s\}$, where t_s matches $(\mathit{unsafe}?[x_1]...\mathit{unsafe}?[x_k]c![\overline{x}], id_s, E_s)$ with $E_s(c) = E_{c_0}(c)$; we then use successively the threads t'_{c_i} to obtain a configuration $C_{n+4} = (C_{n+2} \setminus \{t'_{c_i} \mid i \in [\![0;k]\!]\}) \cup \{t_m\}$, where t_m matches $(x_0![\overline{x}], id_m, E_m)$ with $E_m(x_i) = E''_{c_i}(c)$, for each $i \in [\![0;k]\!]$; moreover the thread $t'_? = (y?^j[\overline{y}]P, id'_?, E'_?)$ is in C_{n+4} with $\Pi_{\mathcal{M}}^{\Phi_{\mathcal{M}}}(id'_?) = id_?$, $\Pi_{\mathcal{N}}^{\Phi_{\mathcal{M}},\Phi_{\mathcal{N}}} \circ E'_? = E_?$, and $\Pi_{\mathcal{N}}^{\Phi_{\mathcal{M}},\Phi_{\mathcal{N}}}(E''_{c_i}(c)) = (u_i, id_i)$, for each $i \in [\![1;k]\!]$; thanks to Lem. 44, there also exists $\mathit{Cont}'_? \in \beta(P, id'_?, E'_?[y_i \mapsto E''_{c_i}(c)])$ such that $\Pi_{\mathrm{C}}^{\Phi_{\mathcal{M}},\Phi_{\mathcal{N}}}(\mathit{Cont}'_?) = \mathit{Cont}_?$; thus we perform the communication between the thread $t'_?$ and t_m to obtain the configuration $C_{n+5} = C_{n+4} \setminus \{t'_?; t_m\} \cup \mathit{Cont}'_?$.

- the computation sequence $C_n \longrightarrow_{\mathrm{e}}^* C_{n+4}$ does not involve thread of $\mathcal{S}_{\mathrm{I}}$, by Lem. 45, we have $\Pi_{\mathrm{C}}^{\Phi_{\mathcal{M}},\Phi_{\mathcal{N}}}(C_{n+4}) = A_{p-1}$; then we conclude that $\Pi_{\mathrm{C}}^{\Phi_{\mathcal{M}},\Phi_{\mathcal{N}}}(C_{n+5}) = A_p$; moreover $C_{n+4} \xrightarrow{i,j'}_{\mathrm{e}} C_{n+5}$ with $j' \notin \mathcal{L}_{\mathrm{I}}$; so $\Pi_\tau(\mathcal{S}_{\mathrm{I}}, \Phi_{\mathcal{M}}, \Phi_{\mathcal{N}})(C_0...C_{n+4} \xrightarrow{(i,j')}_{\mathrm{e}} C_{n+5}) = \tau'$.
- let (u, id_u) be an unsafe name occurring in a state of τ:
  - in the case that $(u, id_u) = (u_i, id_i)$ with $i \in [\![0;k]\!]$, t''_{c_i} is a thread of C_{n+5} and matches $(\mathrm{unsafe}![c], id, E)$ with $\Pi_{\mathcal{N}}^{\Phi_{\mathcal{M}},\Phi_{\mathcal{N}}}(E(c)) = (u_i, id_i)$;
  - otherwise there is necessarily a thread in C_n which matches $(\mathrm{unsafe}![y], id, E)$ with $\Pi_{\mathcal{N}}^{\Phi_{\mathcal{M}},\Phi_{\mathcal{N}}}(E(y)) = (x, id_x)$ and this thread is still in C_{n+5}.

(e) in the case that $(A_{p-1}, U_{p-1}, F_{p-1}) \overset{\lambda}{\hookrightarrow} (A_p, U_p, F_p)$ is a spoilt replication: there is a thread $t_? = (*y?^i[y_1, ..., y_k]P, id_?, E_?)$ with $E_?(y) \in U_{p-1}$ (we set $(u_0, id_0) = E_?(y)$), there are several channel names $(u_i, id_i) \in U_{p-1}$, for $i \in [\![1;k]\!]$, a marker $id_! \in F_{p_1}$ and a continuation $\mathit{Cont}_? \in \beta(P, N((i,0), id_?, id_!), E_?[y_i \mapsto (u_i, id_i)])$, such that $A_p = A_{p-1} \cup \mathit{Cont}_!$, $U_p = U_{p-1}$ and $F_p = F_{p-1} \setminus \{id_!\}$; moreover, we have $\Pi_{\mathrm{C}}^{\Phi_{\mathcal{M}},\Phi_{\mathcal{N}}}(C_n) = A_{p-1}$, so there exists a thread $t'_? = (*y?^j[\overline{y}]P, id'_?, E'_?)$ in C_n with $\Pi_{\mathcal{M}}^{\Phi_{\mathcal{M}}}(id'_?) = id_?$ and $\Pi_{\mathcal{N}}^{\Phi_{\mathcal{M}},\Phi_{\mathcal{N}}} \circ E'_? = E_?$; we first deal with the names that the context sends into the system $\mathcal{S}_{\mathrm{I}}$ for the first time: for each element (u_i, id_i) of the set $\{(u_i, id_i) \mid i \in [\![1;k]\!]\}$ that does not occur in any state of τ, we create a new unsafe name using the resource **new**, as the result we get a fresh name $(\mathit{channel}, id_{\mathit{chan}_i})$ and, since (u_i, id_i) is necessarily in en, we extend the definition of

$\Phi_{\mathcal{N}}$ with $\Phi_{\mathcal{N}}(\mathit{channel}, \mathit{id}_{\mathit{chan}_i}) = (u_i, \mathit{id}_i)$; we obtain a state C_{n+1} and an updated into map $\Phi_{\mathcal{N}}$, such that there exists a thread $t_{c_i} = (\mathit{unsafe}![c], \mathit{id}_{c_i}, E_{c_i})$ in the state C_{n+1} such that $\Pi_{\mathcal{N}}^{\Phi_{\mathcal{M}},\Phi_{\mathcal{N}}}(E_{c_i}(c)) = (u_i, \mathit{id}_i)$, for each $i \in [\![0;k]\!]$; we then use the resource **repli** to replicate the threads t_{c_i} until we obtain a configuration $C_{n+2} = (C_{n+1} \setminus \{t_{c_i} \mid i \in [\![0;k]\!]\}) \cup \{t'_{c_i} \mid i \in [\![0;k]\!]\} \cup \{t''_{c_i} \mid 0 \in [\![1;k]\!]\}$ where t'_{c_i} matches $(\mathit{unsafe}![c], \mathit{id'}_{c_i}, E'_{c_i})$ and t''_{c_i} matches $(\mathit{unsafe}![c], \mathit{id''}_{c_i}, E''_{c_i})$ with $E'_{c_i}(c) = E''_{c_i}(c) = E_{c_i}(c)$, such that the threads t'_{c_i} and t''_{c_i} are all distinct from each other; we then replicate the resource $\mathbf{spoil}_k$ by consuming the thread t'_{c_0}, and obtain the configuration $C_{n+3} = C_{n+2} \setminus \{t'_{c_0}\} \cup \{t_s\}$, where t_s matches $(\mathit{unsafe}?[x_1]...\mathit{unsafe}?[x_k]c![\overline{x}], \mathit{id}_s, E_s)$ with $E_s(c) = E_{c_0}(c)$; we then use successively the threads t'_{c_i} to obtain a configuration $C_{n+4} = (C_{n+2} \setminus \{t'_{c_i} \mid i \in [\![0;k]\!]\}) \cup \{t_m\}$, where t_m matches $(x_0!^{s_k}[\overline{x}], \mathit{id}_m, E_m)$ with $E_m(x_i) = E''_{c_i}(c)$, for all $i \in [\![0;k]\!]$; by Prop. 4 it is the first time this syntactic component is tagged with the marker id_m; so we extend the definition of $\Phi_{\mathcal{M}}$ with $\Phi_{\mathcal{M}}(s_k, \mathit{id}_m) = \mathit{id}_!$); thus we have $\Pi_{\mathcal{M}}^{\Phi_{\mathcal{M}}}(N((i,s_k), \mathit{id}'_?, \mathit{id}_m)) = N((i,0), \mathit{id}_?, \mathit{id}_!)$; moreover the thread $t'_? = (y?^j[\overline{y}]P, \mathit{id}'_?, E'_?)$ is in C_{n+4} with $\Pi_{\mathcal{N}}^{\Phi_{\mathcal{M}},\Phi_{\mathcal{N}}} \circ E'_? = E_?$, and $\Pi_{\mathcal{N}}^{\Phi_{\mathcal{M}},\Phi_{\mathcal{N}}}(E''_{c_i}(c)) = (u_i, \mathit{id}_i)$, for each $i \in [\![1;k]\!]$; by Lem. 44, there also exists $\mathit{Cont}'_? \in \beta(P, N((i,s_k), \mathit{id}'_?, \mathit{id}_m), E'_?[y_i \mapsto E''_{c_i}(c))])$ such that $\Pi_{\mathrm{C}}^{\Phi_{\mathcal{M}},\Phi_{\mathcal{N}}}(\mathit{Cont}'_?) = \mathit{Cont}_?$; thus we perform the communication between the thread $t'_?$ and t_m to obtain the configuration $C_{n+5} = C_{n+4} \setminus \{t_m\} \cup \mathit{Cont}'_?$.

- the computation sequence $C_n \longrightarrow_{\mathrm{e}}^* C_{n+4}$ does not involves thread of $\mathcal{S}_{\mathrm{I}}$, by Lem. 45, we have $\Pi_{\mathrm{C}}^{\Phi_{\mathcal{M}},\Phi_{\mathcal{N}}}(C_{n+4}) = A_{p-1}$; we conclude that $\Pi_{\mathrm{C}}^{\Phi_{\mathcal{M}},\Phi_{\mathcal{N}}}(C_{n+5}) = A_p$; moreover $C_{n+4} \xrightarrow{i,j'}_{\mathrm{e}} C_{n+5}$ with $j' \notin \mathcal{L}_{\mathrm{I}}$; so $\Pi_\tau(\mathcal{S}_{\mathrm{I}}, \Phi_{\mathcal{M}}, \Phi_{\mathcal{N}})(C_0...C_{n+4} \xrightarrow{(i,j')}_{\mathrm{e}} C_{n+5}) = \tau'$.
- let (u, id_u) be an unsafe name occurring in a state of τ:
  - in the case that $(u, \mathit{id}_u) = (c_i, \mathit{id}_{c_i}$, for $i \in [\![0;k]\!]$, t''_{c_i} is a thread of the state C_{n+5} and matches $(\mathrm{unsafe}![c], \mathit{id}, E)$ where $\Pi_{\mathcal{N}}^{\Phi_{\mathcal{M}},\Phi_{\mathcal{N}}}(E(c)) = (u_i, \mathit{id}_i)$;
  - otherwise there is necessarily a thread in C_n which matches $(\mathrm{unsafe}![y], \mathit{id}, E)$ with $\Pi_{\mathcal{N}}^{\Phi_{\mathcal{M}},\Phi_{\mathcal{N}}}(E(y)) = (x, \mathit{id}_x)$ and this thread is still in C_{n+5}. □

E Abstract Interpretation framework

In this section we give the proof of Props. 27 and 28 which provide generic tools to compose abstractions.

Proposition 27 (Product) *Let* $(\mathcal{C}_1^\sharp, \sqsubseteq_1^\sharp, \bigsqcup_1^\sharp, \bot_1^\sharp, \gamma_1, C_{0_1}^\sharp, \rightsquigarrow_1, \nabla_1)$ *and* $(\mathcal{C}_2^\sharp, \sqsubseteq_2^\sharp, \bigsqcup_2^\sharp, \bot_2^\sharp, \gamma_2, C_{0_2}^\sharp, \rightsquigarrow_2, \nabla_2)$ *be two abstractions.*

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;*
- $\gamma \ : \ \begin{cases} \mathcal{C}^\sharp \to \wp(\Sigma^* \times \mathcal{C}) \\ (a_1^\sharp, a_2^\sharp) \mapsto \gamma_1(a_1^\sharp) \cap \gamma_2(a_2^\sharp); \end{cases}$
- $C_0^\sharp = (C_{0_1}^\sharp, C_{0_2}^\sharp)$;
- $\rightsquigarrow$ *is defined by:*
  $(a_1, a_2) \overset{\lambda}{\rightsquigarrow} (b_1, b_2)$ *if and only if* $a_1 \overset{\lambda}{\rightsquigarrow}_1 b_1$ *and* $a_2 \overset{\lambda}{\rightsquigarrow}_2 b_2$

is also an abstraction.

PROOF. The tuple $(\mathcal{C}^\sharp, \sqsubseteq^\sharp, \bigsqcup^\sharp, \bot^\sharp, \gamma, C_0^\sharp, \rightsquigarrow, \nabla)$ satisfies Def. 24:

- Props. (1),(2),(3),(7) are usual properties of the Cartesian product;
- Prop. (4) is satisfied, since both γ_1 and γ_2 are monotonic;
- Prop. (5) holds because we have both $\gamma(C_0^\sharp) = \gamma_1(C_{0_1}^\sharp) \cap \gamma_2(C_{0_2}^\sharp)$ and $\forall i \in \{1;2\}, \{\varepsilon\} \times \mathcal{C}_0^{\mathrm{o}}(\mathcal{S}) \subseteq \gamma_i(C_{0_i}^\sharp)$;
- Prop. (6) is satisfied:
  let $C^\sharp = (a_1, a_2)$ be an abstract element in $\mathcal{C}^\sharp$, (u, C) be a concrete element which satisfies $(u, C) \in \gamma(C^\sharp)$, λ be a transition label in Σ and $\overline{C}$ a concrete state such that $C \overset{\lambda}{\nrightarrow} \overline{C}$, we need to construct an abstract element $\overline{C}^\sharp \in \mathcal{C}^\sharp$ such that $C^\sharp \overset{\lambda}{\rightsquigarrow} \overline{C}^\sharp$ and $(u.\lambda, \overline{C}) \in \gamma(\overline{C}^\sharp)$: in accordance with Def. 24.6, for all $i \in \{1;2\}$, we can choose $b_i \in C_i^\sharp$, such that $a_i \overset{\lambda}{\rightsquigarrow}_i b_i$ and $(u.\lambda, \overline{C}) \in \gamma(b_i)$; so, by definition of $\rightsquigarrow$, we have $(a_1, a_2) \overset{\lambda}{\rightsquigarrow} (b_1, b_2)$ and, since $\gamma(b_1, b_2) = \gamma_1(b_1) \cap \gamma_2(b_2)$, we obtain that $(u.\lambda, \overline{C}) \in \gamma(b_1, b_2)$; so (b_1, b_2) is a valid candidate. □

Proposition 28 (Reduction) *Let* $(\mathcal{C}^\sharp, \sqsubseteq^\sharp, \bigsqcup^\sharp, \bot^\sharp, \gamma, C_0^\sharp, \rightsquigarrow, \nabla)$ *be an abstraction, and* ρ *be a reduction operator*[12] $\rho \ : \ \mathcal{C}^\sharp \to \mathcal{C}^\sharp$ *which satisfies:*

$$\forall a^\sharp \in \mathcal{C}^\sharp, \ \gamma(a^\sharp) \subseteq \gamma(\rho(a^\sharp)).$$

The following tuple $(\mathcal{C}^\sharp, \sqsubseteq^\sharp, \bigsqcup^\sharp, \bot^\sharp, \gamma, C_{0_\rho}^\sharp, \rightsquigarrow_\rho, \nabla)$ *where*

- $C_{0_\rho}^\sharp = \rho(C_0^\sharp)$;
- $\rightsquigarrow_\rho$ *is defined by:*
  $a \rightsquigarrow_\rho c$ *if and only if there exists* $b \in \mathcal{C}^\sharp$, *such that* $\rho(a) \rightsquigarrow b$ *and* $c = \rho(b)$

is also an abstraction.

[12] ρ simplifies the properties obtained in the abstract domain.

Furthermore, ρ can also be used to simplify the final result of the abstract iteration.

PROOF. The tuple $(\mathcal{C}^\sharp, \sqsubseteq^\sharp, \bigsqcup^\sharp, \bot^\sharp, \gamma, C^\sharp_{0_\rho}, \rightsquigarrow_\rho, \nabla)$ satisfies Def. 24:

- Props. (1),(2),(3),(4),(7) hold because $(\mathcal{C}^\sharp, \sqsubseteq^\sharp, \bigsqcup^\sharp, \bot^\sharp, \gamma, C^\sharp_0, \rightsquigarrow, \nabla)$ is an abstraction;
- Prop. (5) is satisfied because we have:
  · $\{(\varepsilon, c_0) \mid c_0 \in C_0(\mathcal{S})\} \subseteq \gamma(C^\sharp_0)$ (since $(\mathcal{C}^\sharp, \sqsubseteq^\sharp, \bigsqcup^\sharp, \bot^\sharp, \gamma, C^\sharp_0, \rightsquigarrow, \nabla)$ is an abstraction),
  · $\gamma(C^\sharp_0) \subseteq \gamma(\rho(C^\sharp_0))$ (by definition of ρ),
  · $C^\sharp_{0_\rho} = \rho(C^\sharp_0)$ (by definition of $C^\sharp_{0_\rho}$),
  so $\{\varepsilon\} \times \mathcal{C}^o_0(\mathcal{S}) \subseteq \gamma(C^\sharp_0) \subseteq \gamma(\rho(C^\sharp_0)) = \gamma(C^\sharp_{0_\rho})$;
- Prop. (6) is satisfied:
  let a be an element of $\mathcal{C}^\sharp$, (u, C) be a concrete element which satisfies $(u, C) \in \gamma(a)$, λ be a transition label in Σ and $\overline{C}$ a concrete state such that $C \overset{\lambda}{\Rightarrow} \overline{C}$, we need to construct an abstract element $\overline{b} \in \mathcal{C}^\sharp$ such that $C^\sharp \overset{\lambda}{\rightsquigarrow}_\rho \overline{b}$ and $(u.\lambda, \overline{C}) \in \gamma(\overline{b})$: we have $(u, C) \in \gamma(a) \subseteq \gamma(\rho(a))$ and $C \overset{\lambda}{\Rightarrow} \overline{C}$, so in accordance with Def. 24.6, we can choose $b \in \mathcal{C}^\sharp$ such that $\rho(a) \overset{\lambda}{\rightsquigarrow} b$ and $(u.\lambda, \overline{C}) \in \gamma(b)$; then, by definition of $\overset{\lambda}{\rightsquigarrow}_\rho$, we have $a \overset{\lambda}{\rightsquigarrow}_\rho \rho(b)$, and, since $\gamma(b) \subseteq \gamma(\rho(b))$, we obtain that $(u.\lambda, \overline{C}) \in \gamma(\rho(b))$; so $\rho(b)$ is a valid candidate. □

F Control flow analysis

In this section we give the proof of Thm. 30 which establishes the soundness of our control flow analysis.

Proposition 29 *$\forall P \in \mathcal{P}$, $\forall id \in \gamma_1(id^\sharp)$, $\forall E \in (fn(P) \to (\nu n(\mathcal{S}) \times \mathcal{M}))$ such that $\forall m \in fn(P)$, $\forall n \in \nu n(\mathcal{S})$, $\forall id_n \in \mathcal{M}$, $[E(m) = (n, id_n) \implies (id, id_n) \in \gamma_2(E^\sharp(m, n))]$, we have:*

$$\Sigma^* \times (\beta(P, id, E) \times \{\emptyset\} \times \wp(\{t_n \mid n \in \mathbb{N}\})) \subseteq \gamma(\beta^\sharp(P, id^\sharp, E^\sharp)).$$

PROOF. By definition of γ and since $x \in \emptyset$ implies anything, we only have to prove the following proposition *Prop*(P): given $id^\sharp \in Id^\sharp_1$ and $E^\sharp \in ((bn(\mathcal{S}) \times \nu n(\mathcal{S})) \to Id^\sharp_2)$, given a marker $id \in \mathcal{M}$ and an environment $E \in (fn(P) \to$

$(\nu n(\mathcal{S}) \times \mathcal{M}))$ such that:

$$\begin{cases} id \in \gamma_1(id^\sharp) \\ \forall m \in fn(P),\ [E(m) = (n, id_m) \implies (id, id_m) \in \gamma_2(E^\sharp(m, n))] \end{cases}$$

then for any thread $(Q, id_Q, E_Q) \in \bigcup \beta(P, id, E)$, and any free name x of Q, we have:

$$\begin{cases} \beta^\sharp(P, id^\sharp, E^\sharp) = (f, g) \\ E_Q(x) = (y, id_y) \end{cases} \implies (id_Q, id_x) \in \gamma_2(f(Q, x, y))$$

for any sub-term $P \in \mathcal{P}$.

We prove $\forall P \in \mathcal{P},\ Prop(P)$ by induction on the syntax of P: let $P \in \mathcal{P}$ be a sub-term, $id \in \gamma_1(id^\sharp)$ be a marker, $E \in (fn(P) \to (\nu n(\mathcal{S}) \times \mathcal{M}))$ be an environment such that for all m free names of P, for all $n \in \nu n(\mathcal{S})$, for all marker $id_n \in \mathcal{M}$, we have $[E(m) = (n, id_n) \implies (id, id_n) \in \gamma_2(E^\sharp(m, n))]$:

- if sub-term P matches $\mathbf{0}$,
  we have $\beta(P, id, E) = \emptyset$, so the property is trivial;
- if sub-term P matches aQ,
  we have $\beta(P, id, E) = \{\{aQ, id, E_{|fn(aQ)}\}\}$, let x be a free name of aQ, we denote $E(x) = (y, id_x)$ and $\beta^\sharp(P, id^\sharp, E^\sharp) = (f, g)$, by assumption we have $(id, id_x) \in \gamma_2(E^\sharp(x, y))$, besides we have $\beta^\sharp(P, id^\sharp, E^\sharp) = ([(P, m, n) \mapsto E^\sharp(m, n), \forall m \in fn(P), n \in \nu n(\mathcal{S})], \emptyset)$, so $f(aQ, x, y) = E^\sharp(x, y)$, and we obtain $(id, id_x) \in \gamma_2(f(aQ, x, y))$;
- if sub-term P matches $(\nu\ n)Q$,
  - we first prove that we can apply the induction hypothesis: we have $id \in \gamma_1(id^\sharp)$; let m be a free name of Q; in the case that $m = n$ we have $E[n \mapsto (n, id)](m) = (n, id)$ and $(id, id) \in \gamma_2(E^\sharp[(n, n) \mapsto dpush(id^\sharp)](n, n))$; otherwise we denote $E[n \mapsto (n, id)](m)$ by (o, id_m), since $E[n \mapsto (n, id)](m) = E(m)$, we have $E(m) = (o, id_m)$ and by assumption we get that $(id, id_m) \in \gamma_2(E^\sharp(m, o))$, then since $E^\sharp[(n, n) \mapsto dpush(id^\sharp)](m, o) = E^\sharp(m, o)$, we get that $(id, id_m) \in \gamma_2(E^\sharp[(n, n) \mapsto dpush(id^\sharp)](m, o))$; thus in both cases, the assertion $E[n \mapsto (n, id)](m) = (o, id_m)$ implies the fact that $(id, id_m) \in \gamma_2(E^\sharp[(n, n) \mapsto dpush(id^\sharp)](m, o))$;
  - let $T_R = (R, id_R, E_R)$ be a thread in $\bigcup \beta((\nu\ n)Q, id, E)$,
    we have $\beta((\nu\ n)Q, id, E) = \beta(Q, id, E[n \mapsto (n, id)])$ so T_R is also a thread in $\bigcup \beta(Q, id, E[n \mapsto (n, id)])$, so by the induction hypothesis, we obtain that for any free name x of R, we have:

$$\begin{cases} \beta^\sharp(Q, id^\sharp, E^\sharp[(n, n) \mapsto dpush(id^\sharp)]) = (f', g') \\ E[n \mapsto (n, id_R)](x) = (y, id_y) \end{cases} \implies (id_R, id_x) \in \gamma_2(f'(R, x, y));$$

then since $\beta^\sharp(Q, id^\sharp, E^\sharp[(n,n) \mapsto dpush(id^\sharp)]) = \beta^\sharp((\nu\ n)Q, id^\sharp, E^\sharp)$ we can conclude that for any free name x of R, we have:

$$\begin{cases} \beta^\sharp((\nu\ n)Q, id^\sharp, E^\sharp) = (f', g') \\ E[n \mapsto (n, id_R)](x) = (y, id_y) \end{cases} \implies (id_R, id_x) \in \gamma_2(f'(R, x, y)).$$

- if sub-term P matches $Q_1 \mid Q_2$ or with $Q_1 \oplus Q_2$,
  - · we first prove that we can apply the induction hypothesis on both Q_1 and Q_2: let i be either 1 or 2: we have $id \in \gamma_1(id^\sharp)$; let m be a free name in Q_i, we know that m is also a free name in P so by assumption we have $[E(m) = (n, id_m) \implies (id, id_m) \in \gamma_2(E^\sharp(m, n))]$;
  - · let $T = (R, id, E)$ be a thread in $\bigcup \beta(P, id, E)$,
    we know that $\bigcup \beta(P, id, E) = (\bigcup \beta(Q_1, id, E)) \cup (\bigcup \beta(Q_2, id, E))$, so T is either a thread in $\bigcup \beta(Q_1, id, E)$, or a thread in $\bigcup \beta(Q_2, id, E)$, let us assume by symmetry that T is a thread of $\bigcup \beta(Q_1, id, E)$; let x be a free name of R, we know by the induction hypothesis that:

$$\begin{cases} \beta^\sharp(Q_1, id^\sharp, E) = (f', g') \\ E(x) = (y, id_y) \end{cases} \implies (id_R, id_x) \in \gamma_2(f'(R, x, y));$$

    then, since we have $\beta^\sharp(P, id^\sharp, E^\sharp) = \bigsqcup^\sharp\{\beta^\sharp(Q_1, id^\sharp, E^\sharp); \beta^\sharp(Q_2, id^\sharp, E^\sharp)\}$, we get that $\beta^\sharp(Q_1, id^\sharp, E) \sqsubseteq^\sharp \beta^\sharp(P, id^\sharp, E)$, so we conclude that:

$$\begin{cases} \beta^\sharp(P, id^\sharp, E^\sharp) = (f', g') \\ E(x) = (y, id_y) \end{cases} \implies (id_R, id_x) \in \gamma_2(f'(R, x, y)).$$

□

Theorem 30 *$(\mathcal{C}^\sharp, \sqsubseteq^\sharp, \bigsqcup^\sharp, \bot^\sharp, \gamma, C_0^\sharp, \rightsquigarrow_{\text{cfa}}, \nabla)$ is an abstraction.*

PROOF. The tuple $(\mathcal{C}^\sharp, \sqsubseteq^\sharp, \bigsqcup^\sharp, \bot^\sharp, \gamma, C_0^\sharp, \rightsquigarrow_{\text{cfa}}, \nabla)$ satisfies Def. 24:

- Props. (1)(2)(3)(4)(7) follow from usual properties of both component-wise and pair-wise extension;
- Prop. (5) is satisfied: we denote $\beta^\sharp(\mathcal{S}, \varepsilon^\sharp, [(n, ext) \mapsto \varepsilon^\sharp \bullet t^\sharp,\ \forall n \in fn(\mathcal{S})])$ by (f,g), let E be an environment in $(fn(\mathcal{S}) \to en)$ and C be a configuration in $\beta(\mathcal{S}, \varepsilon, E)$, we want to prove that $(\varepsilon, (C, en, \{t_n \mid n \in \mathbb{N}\})) \in \gamma(C_0^\sharp)$:
  - · we have $\varepsilon \in \gamma_1(\varepsilon^\sharp)$; let m be a free name in $\mathcal{S}$: by definition of en, there exists $i \in \mathbb{N}$ such that $E(m) = (ext, t_i)$; since $\{\varepsilon\} \times \{t_n \mid n \in \mathbb{N}\} \subseteq \gamma_2^\sharp(\varepsilon^\sharp \bullet t^\sharp)$, we obtain that $(\varepsilon, t_i) \in \gamma_2([(n, ext) \mapsto \varepsilon^\sharp \bullet t^\sharp](m, ext))$; so by Prop. 29, we obtain that:

$$\Sigma^* \times (\beta(\mathcal{S}, \varepsilon, E) \times \{\emptyset\} \times \wp(\{t_n \mid n \in \mathbb{N}\})) \in \gamma((f, g)).$$

this proves that for all thread (P, id_P, E_P) in C, for all free name x in P such that $E(x) = (y, id_x)$, we have $(id_P, id_x) \in \gamma_2(f(P, x, y))$;

· we have $en = \{ext\} \times \{t_n \mid n \in \mathbb{N}\}$, and for all n in $\mathbb{N}$ we have $t_n \in \gamma_1(t^\sharp)$, so for all $(x, id) \in en$, we have $id \in \gamma_1([ext \mapsto t^\sharp](x))$, which means that $id \in \gamma_1(g(x))$;

- Prop. (6) is satisfied: let $C^\sharp = (f, g)$ be an abstract configuration, (u, C) be in the concretization $\gamma(C^\sharp)$, λ be a transition label and $\overline{C}$ be another configuration such that $C \overset{\lambda}{\hookrightarrow} \overline{C}$, we must construct $\overline{C}^\sharp$ such that $(u.\lambda, \overline{C}) \in \gamma(\overline{C}^\sharp)$ and $C^\sharp \overset{\lambda}{\rightsquigarrow} \overline{C}^\sharp$.

(1) In the case that $C \overset{\lambda}{\hookrightarrow} \overline{C}$ is a communication transition: we denote $C = (A, U, F)$; there exist two syntactic components P_0 and Q_0, such that P_0 matches $y?^i[\overline{y}]P$ and Q_0 matches $x!^j[\overline{x}]Q$, such that there exists two threads $(P_0, id_?, E_?)$ and $(Q_0, id_!, E_!)$ in A with $E_?(y) = E_!(x)$, two continuations $Cont_? \in \beta(P, id_?, E_?[y_i \mapsto E_!(x_i)])$ and $Cont_! \in \beta(Q, id_!, E_!)$ such that $\overline{C}$ is $((A \setminus \{(P_0, id_?, E_?); (Q_0, id_!, E_!)\}) \cup Cont_? \cup Cont_!, U, F)$; we denote $E_!(x) = (c, id_c)$, since $(u, C) \in \gamma(C^\sharp)$, we have $(id_?, id_c) \in \gamma_2(f(P_0, y, c))$ and $(id_!, id_c) \in \gamma_2(f(Q_0, x, c))$; then we can deduce that $(id_?, id_c, id_!, id_c) \in \gamma_4(sync(\{2, 4\}, f(P_0, y, c) \bullet f(Q_0, x, c)))$; since γ_4 is strict we obtain that $\bigsqcup_4\{sync(\{2, 4\}, f(P_0, y, u) \bullet f(Q_0, x, u) \mid u \in \nu n(\mathcal{S}))\} \neq \bot_4^\sharp$; so we can introduce $\overline{C}^\sharp = (f', g')$ such that $C^\sharp \overset{\lambda}{\rightsquigarrow} \overline{C}^\sharp$; we denote $\overline{C} = (\overline{A}, U, F)$, and introduce for each name $u \in \nu n(\mathcal{S})$, the abstract element $A(u) = sync(\{2, 4\}, f(P_0, y, u) \bullet f(Q_0, x, u))$, we also introduce two abstract markers $id_P^\sharp$ and $id_Q^\sharp$ and two abstract environments $E_P^\sharp$ and $E_Q^\sharp$ as follows:

$$id_P^\sharp = \bigsqcup_1\{\prod_{(1)}(A(u)) \mid u \in \nu n(\mathcal{S})\},$$
$$E_P^\sharp = [(m, n) \mapsto \bigsqcup_2\{I_P(m, n, u) \mid u \in \nu n(\mathcal{S})\}, \forall m \in fn(P), n \in \nu n(\mathcal{S})]$$

where

$$I_P(m, n, u) = \begin{cases} \prod_{(1,6)} sync(\{3, 5\}, A(u) \bullet f(Q_0, x_i, n)) & \text{if } m = y_i \\ \prod_{(1,6)} sync(\{1, 5\}, A(u) \bullet f(P_0, m, n)) & \text{otherwise.} \end{cases}$$

$$id_Q^\sharp = \bigsqcup_1\{\prod_{(3)}(A(u)) \mid u \in \nu n(\mathcal{S})\},$$
$$E_Q^\sharp = [(m, n) \mapsto \bigsqcup_2\{I_Q(m, n, u) \mid u \in \nu n(\mathcal{S})\}, \forall m \in fn(Q), n \in \nu n(\mathcal{S})]$$

where $I_Q(m, n, u) = \left\{ \prod_{(3,6)} sync(\{3, 5\}, A(u) \bullet f(Q_0, m, n)). \right.$

We now prove that $(u.\lambda, \overline{C}) \in \gamma(\overline{C}^\sharp)$:

(a) let $T_R = (R, id_R, E_R)$ be a thread of $\overline{A}$, x a free name of R and (y, id_x) such that $E_R(x) = (y, id_x)$, we need to prove that $(id_R, id_x) \in$

$\gamma_2(f'(R,x,y))$:

(i) if T_R is a thread of A: we have $(u,C) \in \gamma(\overline{C})$, so $(id_R, id_x) \in \gamma_2(f(R,x,y))$, then $(f,g) \sqsubseteq (f',g')$ and γ_2 is monotonic, so $(id_R, id_x) \in \gamma_2(f'(R,x,y))$;

(ii) if T_R is a thread of $\bigcup \beta(P, id_?, E_?[y_i \mapsto E_!(x_i)])$: we know that $(id_?, id_c, id_!, id_c) \in \gamma_4(A(c))$; this implies that $id_? \in \gamma_1(id_P^\sharp)$; let us take m be a free name in P, we denote $E_?[y_i \mapsto E_!(x_i)](m) = (n, id_n)$; in the case that m matches y_i, we have $(n, id_n) = E_!(x_i)$; since $(Q_0, id_!, E_!)$ is a thread of A and $(A,U,F) \in \gamma(f,g)$; we know that $(id_!, id_n) \in \gamma_2(f(Q_0, x_i, n))$ and $(id_?, id_c, id_!, id_c) \in \gamma_4(A(c))$, so $(id_?, id_c, id_!, id_c, id_!, id_n) \in \gamma_6(A(c) \bullet f(Q_0, x_i, n))$, then since $id_! = id_!$, we conclude that $(id_?, id_n) \in \gamma_2(E_P^\sharp(m,n))$; in the other case, m is a free name of $y?^i[\overline{y}]P$; since $(P_0, id_?, E_?)$ is a thread of A, and $(A,U,F) \in \gamma(f,g)$, we obtain that $(id_?, id_n) \in \gamma_2(f(\lambda, m, n))$; then since $(id_P, id_c, id_Q, id_c) \in \gamma_4(A(c))$, we get that $(id_?, id_c, id_!, id_c, id_?, id_n) \in \gamma_6(A(c) \bullet f^\sharp_{\text{can}}(P_0, m, n))$; then since $id_? = id_?$, we conclude that $(id_?, id_n) \in \gamma_2(E_P^\sharp(m,n))$; so in any case, we have $(id_?, id_n) \in \gamma_2(E_P^\sharp(m,n))$; we can then deduce from Prop. 29 that for any free name m of T_R, $E_R = (n, id_n)$ implies that $(id_R, id_n) \in \gamma_2(f'(R,m,n))$;

(iii) if T_R is a thread of $\bigcup \beta(Q, id_!, E_!)$: we have $(id_?, id_u, id_!, id_c) \in \gamma_4(A(c))$, so $id_! \in \gamma_1(id_Q^\sharp)$; let m be a free name in Q, we denote $E_!(m) = (n, id_n)$; since $(Q_0, id_!, E_!)$ is a thread of A, and $(A,U,F) \in \gamma(f,g)$, we have $(id_!, id_n) \in \gamma_2(f(Q_0, m, n))$; then $(id_?, id_c, id_!, id_c, id_!, id_n) \in \gamma_6(A(c) \bullet f(Q_0, m, n))$; then since $id_! = id_!$, we conclude that $(id_!, id_n) \in \gamma_2(E_Q^\sharp(m,n))$; then, we can deduce from Prop. 29 that for any free name m of T_R, $E_R = (n, id_n)$ implies that $(id_R, id_n) \in \gamma_2(f'(R,m,n))$;

so in any case, for any name x free in T_R, we have $E_R = (y, id_y)$ implies that $(id_R, id_y) \in \gamma_2(f'(R,x,y))$;

(b) let (x, id) be an element in U, we have $(A,U,F) \in \gamma((f,g))$ so $id \in \gamma_1(g(x))$, since $g = g'$ we obtain $id \in \gamma_1(g'(x))$.

So, in accordance with the definition of γ, we obtain that $(A',U,F) \in \gamma(\overline{C}^\sharp)$.

(2) In the case that $C \overset{\lambda}{\Rightarrow} \overline{C}$ is a resource replication: we denote $C = (A,U,F)$, there exist two syntactic components P_0 and Q_0, such that P_0 matches $*y?^i[\overline{y}]P$ and Q_0 matches with $x!^j[\overline{x}]Q$, such that there exist two threads $(P_0, id_?, E_?)$ and $(Q_0, id_!, E_!)$ in A with $E_?(y) = E_!(x)$, two continuations $Cont_? \in \beta(P, N((i,j), id_?, id_!), E_?[y_i \mapsto E_!(x_i)])$ and $Cont_! \in \beta(Q, id_!, E_!)$ such that $\overline{C}$ is $((A \setminus \{(Q_0, id_!, E_!)\}) \cup Cont_? \cup Cont_!, U, F)$; we denote $E_!(x) = (c, id_c)$; since $(u,C) \in \gamma(C^\sharp)$, we have $(id_?, id_c) \in \gamma_2(f(P_0, y, c))$ and $(id_!, id_c) \in \gamma_2(f(Q_0, x, c))$; so we can deduce that $(id_?, id_c, id_!, id_c) \in \gamma_4(sync(\{2,4\}, f(P_0, y, c) \bullet f(Q_0, x, c)))$; since γ_4 is strict we obtain that $\bigsqcup_4 \{sync(\{2,4\}, f(P_0, y, u) \bullet f(Q_0, x, u)) \mid u \in \nu n(\mathcal{S})\} \neq \bot_4^\sharp$; so we can

introduce $\overline{C}^\sharp = (f', g')$ such that $C^\sharp \overset{\lambda}{\rightsquigarrow} \overline{C}^\sharp$; we denote $\overline{C} = (\overline{A}, U, F)$, and introduce for each name $u \in \nu n(\mathcal{S})$, the abstract element $A(u) = sync(\{2,4\}, f(P_0, y, u) \bullet f(Q_0, x, u))$, we also introduce two abstract markers $id_P^\sharp$ and $id_Q^\sharp$ and two abstract environments $E_P^\sharp$ and $E_Q^\sharp$ as follows:

$$id_P^\sharp = \bigsqcup_1 \{\prod_{(1)} (\underset{(i,j)}{push}(\prod_{(1,3,4)} (A(u)))) \mid u \in \nu n(\mathcal{S})\},$$
$$E_P^\sharp = [(m,n) \mapsto \bigsqcup_2 \{I_P(m,n,u) \mid u \in \nu n(\mathcal{S})\}, \forall m \in fn(P), n \in \nu n(\mathcal{S})]$$

where

$$I_P(m,n,u) = \begin{cases} \underset{(i,j)}{push}(\prod_{(1,3,6)} sync(\{3,5\}, A(u) \bullet f(Q_0, x_i, n))) & \text{if } m = y_i \\ \underset{(i,j)}{push}(\prod_{(1,3,6)} sync(\{1,5\}, A(u) \bullet f(P_0, m, n))) & \text{otherwise.} \end{cases}$$

$$id_Q^\sharp = \bigsqcup_1 \{\prod_{(3)} (A(u)) \mid u \in \nu n(\mathcal{S})\},$$
$$E_Q^\sharp = [(m,n) \mapsto \bigsqcup_2 \{I_Q(m,n,u) \mid u \in \nu n(\mathcal{S})\}, \forall m \in fn(Q), n \in \nu n(\mathcal{S})]$$

where $I_Q(m,n,u) = \left\{ \prod_{(3,6)} sync(\{3,5\}, A(u) \bullet f(Q_0, m, n)). \right.$

We now prove that $(u.\lambda, \overline{C}) \in \gamma(\overline{C}^\sharp)$:

(a) let $T_R = (R, id_R, E_R)$ be a thread of $\overline{A}$, x a free name of R and (y, id_x) such that $E_R(x) = (y, id_x)$, we need to prove that $(id_R, id_x) \in \gamma_2(f'(R, x, y))$:

(i) if T_R is a thread of A: we have $(u, C) \in \gamma(\overline{C})$, so $(id_R, id_x) \in \gamma_2(f(R, x, y))$, then $(f, g) \sqsubseteq (f', g')$ and γ_2 is monotonic, so $(id_R, id_x) \in \gamma_2(f'(R, x, y))$;

(ii) if T_R is a thread of $\bigcup \beta(P, N((i,j), id_?, id_!), E_?[y_i \mapsto E_!(x_i)])$: we know that $(id_?, id_c, id_!, id_c) \in \gamma_4(A(c))$; so $(id_?, id_!, id_c) \in \gamma_3(\prod_{(1,3,4)} A(c))$ and $(N((i,j), id_?, id_!), id_c) \in \gamma_2(\underset{(i,j)}{push}(\prod_{(1,3,4)} (A(c))))$; then we conclude that $N((i,j), id_?, id_!) \in \gamma_1(id_P^\sharp)$; let us take m a free name in P, we denote $E_?[y_i \mapsto E_!(x_i)](m) = (n, id_n)$; in the case that m matches y_i, we have $(n, id_n) = E_!(x_i)$, since $(Q_0, id_!, E_!)$ is a thread of A and $(A, U, F) \in \gamma(f, g)$, we know that $(id_!, id_n) \in \gamma_2(f(Q_0, x_i, n))$; then since $(id_?, id_c, id_!, id_c) \in \gamma_4(A(c))$, we obtain that $(id_?, id_c, id_!, id_c, id_!, id_n) \in \gamma_6(A(c) \bullet f(Q_0, x_i, n))$; this way $(id_?, id_!, id_n) \in \gamma_3(\prod_{(1,3,6)} sync(\{3;5\}, A(c) \bullet f(Q_0, x_i, n)))$; so we can conclude that $(N((i,j), id_?, id_!), id_c) \in \gamma_2(E_P^\sharp(m,n))$; in the other case, m is a free name of $*y?^i[\overline{y}]P$; since $(P_0, id_?, E_?)$ is a thread of A, and $(A, U, F) \in \gamma(f, g)$, we get that $(id_?, id_n) \in \gamma_2(f(P_0, m, n))$; then since $(id_?, id_c, id_!, id_c) \in$

$\gamma_4(A(c))$, we obtain that $(id_?, id_c, id_!, id_c, id_?, id_n) \in \gamma_6(A(c) \bullet f(P_0, m, n)))$; since $id_? = id_?$, we can deduce that $(id_?, id_!, id_n) \in \gamma_3(\prod_{(1,3,6)}(sync(\{1,5\}, A(c) \bullet f(P_0, m, n))))$; so we can conclude that $(N((i,j), id_?, id_!), id_c) \in \gamma_2(E_P^\sharp(m,n))$; so in any case, we have $(N((i,j), id_?, id_!), id_n) \in \gamma_2(E_P^\sharp(m,n))$; we can then deduce from Prop. 29 that for any free name m of T_R, $E_R = (n, id_n)$ implies that $(id_R, id_n) \in \gamma_2(f'(R,m,n))$;

(iii) if T_R is a thread of $\bigcup \beta(Q, id_!, E_!)$: we have $(id_?, id_u, id_!, id_c) \in \gamma_4(A(c))$, so $id_! \in \gamma_1(id_Q^\sharp)$; let us take m a free name in Q; we denote $E_!(m) = (n, id_n)$; since $(Q_0, id_!, E_!)$ is a thread of A, and $(A, U, F) \in \gamma(f,g)$, we have $(id_!, id_n) \in \gamma_2(f(Q_0, m, n))$; then $(id_?, id_c, id_!, id_c, id_!, id_n) \in \gamma_6(A(c) \bullet f(Q_0, m, n))$; then since $id_! = id_!$, we conclude that $(id_!, id_n) \in \gamma_2(E_Q^\sharp(m,n))$; we can then deduce from Prop. 29 that for any free name m of T_R, $E_R = (n, id_n)$ implies that $(id_R, id_n) \in \gamma_2(f'(R,m,n))$;

so in any case, for any name x free in T_R, we have $E_R = (y, id_y)$ implies that $(id_R, id_y) \in \gamma_2(f'(R,x,y))$;

(b) let (x, id) be an element in U, we have $(A, U, F) \in \gamma((f,g))$ so $id \in \gamma_1(g(x))$, since $g = g'$ we obtain $id \in \gamma_1(g'(x))$.

So in accordance with the definition of γ, we obtain that $(A', U, F) \in \overline{C}^\sharp$.

(3) In the case that $C \overset{\lambda}{\hookrightarrow} \overline{C}$ is a spied communication transition: we denote $C = (A, U, F)$, there exists a syntactic component Q_0, such that Q_0 matches $x!^j[x_1, ..., x_n]Q$, such that there exists a thread $(Q_0, id_!, E_!)$ in A with $E_!(x) \in U$ and a continuation $Cont_! \in \beta(Q, id_!, E_!)$ which satisfy $\overline{C} = ((A \backslash \{(Q_0, id_!, E_!)\}) \cup Cont_!, U \cup \{E_!(x_i) \mid i \in [\![1; n]\!]\}, F)$; we denote $E_!(x) = (c, id_c)$; since $(u, C) \in \gamma(C^\sharp)$, we have $(id_!, id_c) \in \gamma_2(f(Q_0, x, c))$ and $(id_c) \in \gamma_1(g(c))$; then $(id_!, id_c, id_c) \in \gamma_3(sync(\{2,3\}, f(Q_0, x, c) \bullet g(c)))$; since γ_3 is strict, we can conclude that $\bigsqcup_3 \{sync(\{2,3\}, f(Q_0, x, u) \bullet g(u)) \mid u \in \nu n(\mathcal{S})\} \neq \bot_3^\sharp$; so we can introduce $\overline{C}^\sharp = (f', g')$ such that $C^\sharp \overset{\lambda}{\rightsquigarrow} \overline{C}^\sharp$; we denote $\overline{C} = (\overline{A}, \overline{U}, \overline{F})$, and introduce for each name $u \in \nu n(\mathcal{S})$, the abstract element $A(u) = sync(\{2,3\}, f(Q_0, x, u) \bullet g(u))$, we also introduce an abstract marker $id_Q^\sharp$ and an abstract environment $E_Q^\sharp$ as follows:

$$id_Q^\sharp = \bigsqcup_1 \{\prod_{(1)}(A(u)) \mid u \in \nu n(\mathcal{S})\},$$

$$E_Q^\sharp = [(m,n) \mapsto \bigsqcup_2 \{I_Q(m,n,u) \mid u \in \nu n(\mathcal{S})\}, \forall m \in fn(Q), n \in \nu n(\mathcal{S})]$$

where $I_Q(m,n,u) = \begin{cases} \prod_{(1,5)} sync(\{1,4\}, A(u) \bullet f(Q_0, m, n)). \end{cases}$

we now prove that $(u.\lambda, \overline{C}) \in \gamma(\overline{C}^\sharp)$:

(a) let $T_R = (R, id_R, E_R)$ be a thread of $\overline{A}$, x be a free name in R, we denote $E_R(x) = (y, id_x)$, we need to prove that $(id_R, id_x) \in \gamma_2(f'(R,x,y))$:

(i) if T_R is a thread of A: we have $(u, C) \in \gamma(\overline{C})$, so $(id_R, id_x) \in \gamma_2(f(R, x, y))$, then $(f, g) \sqsubseteq (f', g')$ and γ_2 is monotonic, so $(id_R, id_x) \in \gamma_2(f'(R, x, y))$;

(ii) if T_R is a thread of $\bigcup \beta(Q, id_!, E_!)$: since $(id_!, id_c, id_c) \in \gamma_3(A(c))$, we have that $id_! \in \gamma_1(id_Q^\sharp)$; let us take m a free name in Q; we denote $E_!(m) = (n, id_n)$; since $(Q_0, id_!, E_!)$ is a thread of A, and $(A, U, F) \in \gamma(f, g)$, we obtain that $(id_!, id_n) \in \gamma_2(f(Q_0, m, n))$; then $(id_!, id_c, id_c, id_!, id_n) \in \gamma_5(A(c) \bullet f(Q_0, m, n))$; then since $id_! = id_1$, we conclude that $(id_!, id_n) \in \gamma_2(E_Q^\sharp(m, n))$; we can then deduce from Prop. 29 that for any free name m of T_R, $E_R = (n, id_n)$ implies that $(id_R, id_n) \in \gamma_2(f'(R, m, n))$;

so in any case, for any name m free in T_R, the fact $E_R = (n, id_n)$ implies that $(id_R, id_n) \in \gamma_2(f'(R, m, n))$;

(b) let (u, id) be an element in $\overline{U}$; we need to prove that $id \in \gamma_1(g(u))$:

(i) in the case that $(u, id) \in U$, we have $C \in \gamma(C^\sharp)$, $C^\sharp \sqsubseteq^\sharp \overline{C}^\sharp$ and γ is monotonic, then $C \in \gamma(\overline{C}^\sharp)$ and $id \in \gamma_1(g'(u))$;

(ii) otherwise we have $(u, id) = E(x_k)$, so since $C \in \gamma(C^\sharp)$, we obtain that $(id_!, id_c, id_c, id_!, id) \in sync(\{(1, 4)\}, A(c) \bullet (f(Q_0, x_k, u)))$, so $id \in \gamma_1(g'(u))$;

so in any case, for any $(u, id) \in \overline{U}$, we have $id \in \gamma_1(g'(u))$.

So in accordance with the definition of γ, we obtain that $(\overline{A}, \overline{U}, \overline{F}) \in \overline{C}^\sharp$.

(4) In the case that $C \overset{\lambda}{\nrightarrow} \overline{C}$ is a spoiled communication. We denote $C = (A, U, F)$, there exists a syntactic component P_0, such that P_0 matches $y?^i[\overline{y}]P$, such that there exists one thread $(P_0, id_?, E_?)$ in A with $E_?(y) \in U$, some elements c_1,...,c_n in U and a continuation $Cont \in \beta(P, id_?, E_?[y_i \mapsto c_i])$ that $\overline{C} = ((A \backslash \{(P_0, id_?, E_?)\}) \cup Cont, U, F)$; we denote $E_?(y) = (c, id_c)$; we have $(u, C) \in \gamma(C^\sharp)$, so $(id_?, id_c) \in \gamma_2(f(P_0, y, c))$ and $(id_c) \in \gamma_1(g(c))$; then $(id_?, id_c, id_c) \in \gamma_3(sync(\{2, 3\}, f(P_0, y, c) \bullet g(c)))$; since γ_3 is strict we obtain that $\bigsqcup_3\{sync(\{2, 3\}, f(P_0, y, c) \bullet g(c)) \mid u \in \nu n(\mathcal{S})\} \neq \bot_3^\sharp\}$; so we can introduce $\overline{C}^\sharp = (f', g')$ such that $C^\sharp \overset{\lambda}{\rightsquigarrow} \overline{C}^\sharp$; we denote $\overline{C} = (\overline{A}, U, F)$, and introduce for each name $u \in \nu n(\mathcal{S})$, the abstract element $A(u) = sync(\{2, 3\}, f(P_0, y, u) \bullet g(u))$, we also introduce an abstract marker $id_P^\sharp$ and an abstract environment $E_P^\sharp$ as follows:

$$id_P^\sharp = \bigsqcup_1\{\coprod_{(1)}(A(u)) \mid u \in \nu n(\mathcal{S})\},$$

$$E_P^\sharp = [(m, n) \mapsto \bigsqcup_2\{I_P(m, n, u) \mid u \in \nu n(\mathcal{S})\}, \forall m \in fn(P), n \in \nu n(\mathcal{S})]$$

where

$$I_P(m, n, u) = \begin{cases} id_P^\sharp \bullet g(n) & \text{if } m = y_i \\ \coprod_{(1,5)} sync(\{1, 4\}, A(u) \bullet f(P_0, m, n)) & \text{otherwise.} \end{cases}$$

We now prove that $(u.\lambda, \overline{C}) \in \gamma(\overline{C}^\sharp)$:

(a) let $T_R = (R, id_R, E_R)$ be a thread of $\overline{A}$, x a free name of R and (y, id_x) such that $E_R(x) = (y, id_x)$, we need to prove that $(id_R, id_x) \in \gamma_2(f'(R, x, y))$:
  (i) if T_R is a thread of A: we have $(u, C) \in \gamma(\overline{C})$, so $(id_R, id_x) \in \gamma_2(f(R, x, y))$, then $(f, g) \sqsubseteq (f', g')$ and γ_2 is monotonic, so $(id_R, id_x) \in \gamma_2(f'(R, x, y))$;
  (ii) if T_R is a thread of $\bigcup \beta(P, id_?, E_?[y_i \mapsto c_k])$: since $(id_?, id_c, id_c) \in \gamma_3(A(c))$, we deduce that $id_? \in \gamma_1(id_P^\sharp)$; let us take m be a free name in P; we denote $E_?[y_i \mapsto E_!(x_i)](m) = (n, id_n)$; in the case that m matches y_i, we have $(n, id_n) = c_i$; since $c_i \in U$ and $(A, U, F) \in \gamma(f, g)$, we know that $id_n \in \gamma_1(g(n))$; then since $id_? \in \gamma_1(id_P^\sharp)$, we get that $(id_?, id_n) \in \gamma_2(id_P^\sharp \bullet g(c))$; then we conclude that $(id_?, id_n) \in \gamma_2(E_P^\sharp(m, n))$; in the other case, m is a free name of $y?^i[\overline{y}]P$; since $(P_0, id_?, E_?)$ is a thread of A, and $(A, U, F) \in \gamma(f, g)$, we obtain that $(id_?, id_n) \in \gamma_2(f(P_0, m, n))$; so, since $(id_?, id_c, id_c) \in \gamma_3(A(c))$, we have $(id_?, id_c, id_c, id_?, id_n) \in \gamma_5(A(c) \bullet (f(P_0, m, n)))$; then since $id_? = id_?$, we conclude that $(id_?, id_n) \in \gamma_2(E_P^\sharp(m, n))$; so in any case, we have $(id_?, id_n) \in \gamma_2(E_P^\sharp(m, n))$; we can then deduce from Prop. 29 that for any free name m of T_R, $E_R = (n, id_n)$ implies that $(id_R, id_n) \in \gamma_2(f'(R, m, n))$;
  so in any case, for any name m free in T_R, we have $E_R = (n, id_n)$ implies that $(id_R, id_n) \in \gamma_2(f'(R, m, n))$;
(b) let (x, id) be an element in U, we have $(A, U, F) \in \gamma((f, g))$ so $id \in \gamma_1(g(x))$, since $g = g'$ we obtain $id \in \gamma_1(g'(x))$.

So in accordance with the definition of γ, we obtain that $(A', U, F) \in \overline{C}^\sharp$.

(5) In the case that $C \overset{\lambda}{\Rightarrow} \overline{C}$ is a spoiled resource replication. We denote $C = (A, U, F)$, there exists a syntactic component P_0, such that P_0 matches $*y?^i[\overline{y}]P$, such that there exists one thread $(P_0, id_?, E_?)$ in A such that $E_?(y) \in U$, some elements c_1,...,c_n in U, a fresh marker $id_! \in F$ and a continuation $Cont \in \beta(P, id_?, E_?[y_i \mapsto c_i])$ that $\overline{C} = ((C \setminus \{(P_0, id_?, E_?)\}) \cup Cont, U, F \setminus \{id_!\})$; we denote $E_?(y) = (c, id_c)$, since $(u, C) \in \gamma(C^\sharp)$, we have $(id_?, id_c) \in \gamma_2(f(P_0, y, c))$ and $(id_c) \in \gamma_1(g(c))$, so we deduce that $(id_?, id_c, id_c) \in \gamma_3(sync(\{2, 3\}, f(P_0, y, c) \bullet g(c)))$, and since γ_3 is strict we obtain that $\bigsqcup_3\{sync(\{2, 3\}, f(P_0, y, c) \bullet g(c)) \mid u \in \nu n(\mathcal{S})\} \neq \bot_3^\sharp\})$; so we can introduce $\overline{C}^\sharp = (f', g')$ such that $C^\sharp \overset{\lambda}{\rightsquigarrow} \overline{C}^\sharp$; we denote $\overline{C} = (\overline{A}, U, F)$, and introduce for each name $u \in \nu n(\mathcal{S})$, the abstract element $A(u) = sync(\{2, 3\}, f(P_0, y, u) \bullet g(u))$, we also introduce an abstract marker $id_P^\sharp$

and an abstract environment $E_P^\sharp$ as follows:

$$id_P^\sharp = \bigsqcup_1 \{ \prod_{(1)} (push_{(i,0)}((\prod_{(1)}(A(u))) \bullet t^\sharp \bullet \top_{\mathcal{M}}^\sharp)) \mid u \in \nu n(\mathcal{S}) \},$$
$$E_P^\sharp = [(m,n) \mapsto \bigsqcup_2 \{ I_P(m,n,u) \mid u \in \nu n(\mathcal{S}) \}, \forall m \in fn(P), n \in \nu n(\mathcal{S})]$$

where

$$I_P(m,n,u) = \begin{cases} id_P^\sharp \bullet g(n) & \text{if } m = y_i \\ push_{(i,0)}(\prod_{(1,6,5)} sync(\{1,4\}, A(u) \bullet f(P_0,m,n) \bullet t^\sharp)) & \text{otherwise.} \end{cases}$$

We now prove that $(u.\lambda, \overline{C}) \in \gamma(\overline{C}^\sharp)$:

(a) let $T_R = (R, id_R, E_R)$ be a thread of $\overline{A}$, x a free name of R and (y, id_x) such that $E_R(x) = (y, id_x)$, we need to prove that $(id_R, id_x) \in \gamma_2(f'(R,x,y))$:
  (i) if T_R is a thread of A: we have $(u, C) \in \gamma(\overline{C})$, so $(id_R, id_x) \in \gamma_2(f(R,x,y))$, then $(f,g) \sqsubseteq (f',g')$ and γ_2 is monotonic, so $(id_R, id_x) \in \gamma_2(f'(R,x,y))$;
  (ii) if T_R is a thread of $\bigcup \beta(P, N((i,0), id_?, id_!), E_?[y_i \mapsto c_k])$: we know that $(id_?, id_c, id_c) \in \gamma_3(A(c))$; so $id_? \in \gamma_1(\prod_{(1)}(A(c))$; moreover we have $id_! \in t^\sharp$; then for all $id \in \mathcal{M}$, we have $(id_?, id_!, id) \in \gamma_3(\prod_{(1)}(A(c)) \bullet t^\sharp \bullet \top_{\mathcal{M}}^\sharp)$; so $N((i,0), id_?, id_!) \in \gamma_1(id_P^\sharp)$; let us take m be a free name in P; we denote $E_?[y_i \mapsto E_!(x_i)](m) = (n, id_n)$; in the case that m matches y_i, we have $(n, id_n) = c_i$; since $c_i \in U$ and $(A,U,F) \in \gamma(f,g)$, we know that $id_n \in \gamma_1(g(n))$; since $N((i,0), id_?, id_!) \in \gamma_1(id_P^\sharp)$, we get that $(N((i,0), id_?, id_!), id_n) \in \gamma_2(id_P^\sharp \bullet g(c))$; so $(N((i,0), id_?, id_!), id_n) \in \gamma_2(E_P^\sharp(m,n))$; in the other case, m is a free name of $y?^i[\overline{y}]P$; since $(P_0, id_?, E_?)$ is a thread of A, and $(A,U,F) \in \gamma(f,g)$; we obtain that $(id_?, id_n) \in \gamma_2(f(\lambda, m, n))$; since $(id_?, id_c, id_c) \in \gamma_3(A(c))$, we can deduce that $(id_?, id_c, id_c, id_?, id_n) \in \gamma_5(A(c) \bullet (f(P_0,m,n)))$; since $id_? = id_?$, we can conclude that $(N((i,0), id_?, id_!), id_n) \in \gamma_2(E_P^\sharp(m,n))$; so in any case, we have $(N((i,0), id_?, id_!), id_n) \in \gamma_2(E_P^\sharp(m,n))$; we can then deduce from Prop. 29 that for any free name m of T_R, $E_R = (n, id_n)$ implies that $(id_R, id_n) \in \gamma_2(f'(R,m,n))$;
  so in any case, for any name m free in T_R, we have $E_R = (n, id_n)$ implies that $(id_R, id_n) \in \gamma_2(f'(R,m,n))$;

(b) let (x, id) be an element in U, we have $(A,U,F) \in \gamma((f,g))$ so $id \in \gamma_1(g(x))$, since $g = g'$ we obtain $id \in \gamma_1(g'(x))$.

So in accordance with the definition of γ, we obtain that $(A', U, F) \in \overline{C}^\sharp$. □

G Occurrence counting abstraction

In this section we give the proof of Thm. 39 which establishes the soundness of our occurrence counting analysis.

Proposition 38 $\forall P \in \mathcal{P},\ \forall id \in \mathcal{M},\ \forall E \in (fn(P) \to (\nu n(\mathcal{S}) \times \mathcal{M}))$, *we have* $Cont \in \beta(P, id, E) \implies (\varepsilon, Cont) \in (\gamma_{\mathbb{N}^V} \circ \gamma_{\mathcal{N}_V})(\beta_{\mathcal{N}_V}(P))$.

PROOF. Prop. 38 is proved by induction on the syntax of P: let P be a sub-term in $\mathcal{P}$, id be a marker in $\mathcal{M}$, E be an environment in $(fn(P) \to (\nu n(\mathcal{S}) \times \mathcal{M}))$ and $Cont$ be a continuation in $\beta(P, id, E)$:

- in the case that P matches 0, we have $\beta(P, id, E) = \{\emptyset\}$; so $Cont = \emptyset$; then $\Pi_{\mathbb{N}^V}(\varepsilon, Cont) = (0)_{v \in V}$; since $(0)_{v \in V} \in \gamma_{\mathcal{N}_V}(0_{\mathcal{N}_V})$ and $\beta_{\mathcal{N}_V}(P) = (0)_{v \in V}$, we obtain that $(\varepsilon, \emptyset) \in (\gamma_{\mathbb{N}^V} \circ \gamma_{\mathcal{N}_V})(\beta_{\mathcal{N}_V}(P))$;
- in the case that P matches $(\nu\ n)Q$, since $\beta(P, id, E) = \beta(Q, id, E[n \mapsto (n, id)])$, we have $(\varepsilon, Cont) \in (\gamma_{\mathbb{N}^V} \circ \gamma_{\mathcal{N}_V})(\beta_{\mathcal{N}_V}(Q))$ (by the induction hypothesis); since $\beta_{\mathcal{N}_V}(P) = \beta_{\mathcal{N}_V}(Q)$, we obtain that $(\varepsilon, Cont) \in (\gamma_{\mathbb{N}^V} \circ \gamma_{\mathcal{N}_V})(\beta_{\mathcal{N}_V}(P))$;
- in the case that P matches aQ, we have $\beta(P, id, E) = \{\{(P, id, E)\}\}$; so $Cont = \{(P, id, E)\}$; then $\Pi_{\mathbb{N}^V}(\varepsilon, Cont) = (\delta_v^{aQ})_{v \in V}$; we obtain $(\varepsilon, Cont) \in (\gamma_{\mathbb{N}^V} \circ \gamma_{\mathcal{N}_V})(\beta_{\mathcal{N}_V}(aP))$, since $(\delta_v^{aQ})_{v \in V} \in \gamma_{\mathcal{N}_V}(1_{\mathcal{N}_V}(aP))$ and $\beta_{\mathcal{N}_V}(aP) = 1_{\mathcal{N}_V}(aP)$;
- in the case that P matches $P_1 \mid P_2$, we have $\beta(P, id, E) = \{A_1 \cup A_2 \mid A_i \in \beta(P_i, id, E)\}$; let us take $(Cont_1, Cont_2) \in (\beta(P_1, id, E) \times \beta(P_2, id, E))$ such that $Cont = Cont_1 \cup Cont_2$, by the induction hypothesis, we have for any i in the set $\{1; 2\}$, $(\varepsilon, Cont_i) \in (\gamma_{\mathbb{N}^V} \circ \gamma_{\mathcal{N}_V})(\beta_{\mathcal{N}_V}(P_i))$ (i.e. $\Pi_{\mathbb{N}^V}(\varepsilon, Cont_i) \in \gamma_{\mathcal{N}_V}(\beta_{\mathcal{N}_V}(P_i))$); moreover since $P \in \mathcal{P}$ is labelled with distinct labels, we have $Cont_1 \cap Cont_2 = \emptyset$; by applying Prop. 37, since $\Pi_{\mathbb{N}^V}(\varepsilon, Cont) = \Pi_{\mathbb{N}^V}(\varepsilon, Cont_1) + \Pi_{\mathbb{N}^V}(\varepsilon, Cont_2)$, we obtain $\Pi_{\mathbb{N}^V}(\varepsilon, Cont) \in \gamma_{\mathcal{N}_V}(\beta_{\mathcal{N}_V}(P_1) + \beta_{\mathcal{N}_V}(P_2))$; since $\beta_{\mathcal{N}_V}(P) = \beta_{\mathcal{N}_V}(P_1) + \beta_{\mathcal{N}_V}(P_2)$, we obtain $(\varepsilon, Cont) \in (\gamma_{\mathbb{N}^V} \circ \gamma_{\mathcal{N}_V})(\beta_{\mathcal{N}_V}(P))$;
- in the case that P matches $P_1 \oplus P_2$, we have $\beta(P_1 \oplus P_2, id, E) = \beta(P_1, id, E) \cup \beta(P_2, id, E)$; so let us take i in the set $\{1; 2\}$ such that $Cont \in \beta(P_i, id, E)$; by the induction hypothesis, we have $Cont \in (\gamma_{\mathbb{N}^V} \circ \gamma_{\mathcal{N}_V})(\beta_{\mathcal{N}_V}(P_i))$; since $\beta_{\mathcal{N}_V}(P) = \beta_{\mathcal{N}_V}(P_1) \cup_{\mathcal{N}_V} \beta_{\mathcal{N}_V}(P_2)$, we get that $Cont \in (\gamma_{\mathbb{N}^V} \circ \gamma_{\mathcal{N}_V})(\beta_{\mathcal{N}_V}(P))$. □

Theorem 39 $(\mathcal{N}_V, \sqsubseteq_{\mathcal{N}_V}, \cup_{\mathcal{N}_V}, \bot_{\mathcal{N}_V}, \gamma_{\mathbb{N}^V} \circ \gamma_{\mathcal{N}_V}, C_0^{\mathcal{N}_V}, \rightsquigarrow_{\mathcal{N}_V}, \nabla_{\mathcal{N}_V})$ *is an abstraction.*

PROOF. The tuple $(\mathcal{N}_V, \sqsubseteq_{\mathcal{N}_V}, \cup_{\mathcal{N}_V}, \bot_{\mathcal{N}_V}, \gamma_{\mathbb{N}^V} \circ \gamma_{\mathcal{N}_V}, C_0^{\mathcal{N}_V}, \rightsquigarrow_{\mathcal{N}_V}, \nabla_{\mathcal{N}_V})$ satisfies Def. 24:

- Props. (1)(2)(3)(7) are satisfied by our assumptions;
- Prop. (4) is satisfied since both $\gamma_{\mathbb{N}^V}$ and $\gamma_{\mathcal{N}_V}$ are monotonic;
- Prop. (5) is satisfied: we have $\mathcal{C}_0^{\mathrm{n}}(\mathcal{S}) = \beta(\mathcal{S}, \varepsilon, \emptyset)$ and $C_0^{\mathcal{N}_V} = \beta_{\mathcal{N}_V}(\mathcal{S})$, we conclude by Prop. 38 that $\{\varepsilon\} \times \mathcal{C}_0^{\mathrm{n}}(\mathcal{S}) \subseteq C_0^{\mathcal{N}_V}$;
- Prop. (6) is satisfied: let $v^\sharp$ be an abstract configuration, (u, C) be in the concretization $(\gamma_{\mathbb{N}^V} \circ \gamma_{\mathcal{N}_V})(v^\sharp)$, λ be a transition label and $\overline{C}$ be another configuration such that $C \overset{\lambda}{\nrightarrow} \overline{C}$, we must construct $\overline{C}^\sharp$ such that $(u.\lambda, \overline{C}) \in \gamma(\overline{C}^\sharp)$ and $C^\sharp \overset{\lambda}{\rightsquigarrow}_{\mathcal{N}_V} \overline{C}^\sharp$.

(1) in the case that $C \overset{\lambda}{\nrightarrow} \overline{C}$ is a communication transition: there exist two syntactic components P_0 and Q_0, such that P_0 matches with $y?^i[\overline{y}]P$ and Q_0 matches with $x!^j[\overline{x}]Q$, such that there exist two threads $(P_0, id_?, E_?)$ and $(Q_0, id_!, E_!)$ in C with $E_?(y) = E_!(x)$, two continuations $Cont_? \in \beta(P, id_?, E_?[y_i \mapsto E_!(x_i)])$ and $Cont_! \in \beta(Q, id_!, E_!)$ such that $\overline{C}$ is $((C \setminus \{(P_0, id_?, E_?); (Q_0, id_!, E_!)\}) \cup Cont_? \cup Cont_!)$; we have $(u, C) \in (\gamma_{\mathbb{N}^V} \circ \gamma_{\mathcal{N}_V})(v^\sharp)$; so $\Pi_{\mathbb{N}^V}(u, C) \in \gamma_{\mathcal{N}_V}(v^\sharp)$; then since $(P_0, id_?, E_?)$ and $(Q_0, id_!, E_!)$ are two threads of C, we have $(\Pi_{\mathbb{N}^V}(u, C))_{P_0} \geq 1$ and $(\Pi_{\mathbb{N}^V}(u, C))_{Q_0} \geq 1$; then $\Pi_{\mathbb{N}^V}(u, C) \in \gamma_{\mathcal{N}_V}(sync_{\mathcal{N}_V}(\{i, j\}, v^\sharp))$; since $\gamma_{\mathcal{N}_V}$ is strict, we can conclude that $sync_{\mathcal{N}_V}(\{i, j\}, v^\sharp) \neq \bot_{\mathcal{N}_V}$; so we can introduce $\overline{v}^\sharp$ such that $C^\sharp \overset{\lambda}{\rightsquigarrow}_{\mathcal{N}_V} \overline{C}^\sharp$; we now prove that $(u.\lambda, \overline{C}) \in (\gamma_{\mathbb{N}^V} \circ \gamma_{\mathcal{N}_V})(\overline{v}^\sharp)$: we have $\overline{C} = ((C \setminus \{(P_0, id_?, E_?); (Q_0, id_!, E_!)\}) \cup Cont_? \cup Cont_!)$; so $\Pi_{\mathbb{N}^V}(u.\lambda, \overline{C}) = \Pi_{\mathbb{N}^V}(u, C) - \left((\delta_v^{P_0})_{v \in V} + (\delta_v^{Q_0})_{v \in V}\right) + \Pi_{\mathbb{N}^V}(\varepsilon, Cont_?) + \Pi_{\mathbb{N}^V}(\varepsilon, Cont_!) + (\delta_v^{\psi(\lambda)})_{v \in V}$; then we can deduce from Prop. 38 that $\Pi_{\mathbb{N}^V}(\varepsilon, Cont_?) \in \gamma_{\mathbb{N}^V}(\beta_{\mathcal{N}_V}(P))$ and that $\Pi_{\mathbb{N}^V}(\varepsilon, Cont_!) \in \gamma_{\mathbb{N}^V}(\beta_{\mathcal{N}_V}(Q))$; besides $\Pi_{\mathbb{N}^V}(u, C) \in \gamma_{\mathcal{N}_V}(sync_{\mathcal{N}_V}(\{i, j\}, v^\sharp))$, so we conclude that $\Pi_{\mathbb{N}^V}(u.\lambda, \overline{C}) \in \gamma_{\mathcal{N}_V}(sync_{\mathcal{N}_V}(\{i, j\}, v^\sharp) - 1_{\mathcal{N}_V}(P_0) - 1_{\mathcal{N}_V}(Q_0) + 1_{\mathcal{N}_V}(\psi(\lambda)) + \beta_{\mathcal{N}_V}(P) + \beta_{\mathcal{N}_V}(Q))$; so $(u.\lambda, \overline{C}) \in (\gamma_{\mathbb{N}^V} \circ \gamma_{\mathcal{N}_V})(\overline{v}^\sharp)$;

(2) in the case that $C \overset{\lambda}{\nrightarrow} \overline{C}$ is a resource replication: there exist two syntactic components P_0 and Q_0, such that P_0 matches $*y?^i[\overline{y}]P$ and Q_0 matches with $x!^j[\overline{x}]Q$, such that there exist two threads $(P_0, id_?, E_?)$ and $(Q_0, id_!, E_!)$ in C with $E_?(y) = E_!(x)$, a marker $id_* \in \mathcal{M}$, two continuations $Cont_? \in \beta(P, id_*, E_?[y_i \mapsto E_!(x_i)])$ and $Cont_! \in \beta(Q, id_!, E_!)$ such that $\overline{C}$ is $((C \setminus \{(Q_0, id_!, E_!)\}) \cup Cont_? \cup Cont_!)$; we have $(u, C) \in (\gamma_{\mathbb{N}^V} \circ \gamma_{\mathcal{N}_V})(v^\sharp)$; so $\Pi_{\mathbb{N}^V}(u, C) \in \gamma_{\mathcal{N}_V}(v^\sharp)$; then since $(P_0, id_?, E_?)$ and $(Q_0, id_!, E_!)$ are two threads of C, we have $(\Pi_{\mathbb{N}^V}(u, C))_{P_0} \geq 1$ and $(\Pi_{\mathbb{N}^V}(u, C))_{Q_0} \geq 1$; so we deduce that $\Pi_{\mathbb{N}^V}(u, C) \in \gamma_{\mathcal{N}_V}(sync_{\mathcal{N}_V}(\{i, j\}, v^\sharp))$; since $\gamma_{\mathcal{N}_V}$ is strict, we get that $sync_{\mathcal{N}_V}(\{i, j\}, v^\sharp) \neq \bot_{\mathcal{N}_V}$; so we can introduce $\overline{v}^\sharp$ such that $C^\sharp \overset{\lambda}{\rightsquigarrow}_{\mathcal{N}_V} \overline{C}^\sharp$; we now prove that $(u.\lambda, \overline{C}) \in (\gamma_{\mathbb{N}^V} \circ \gamma_{\mathcal{N}_V})(\overline{v}^\sharp)$: we have $\overline{C} = ((C \setminus \{(Q_0, id_!, E_!)\}) \cup Cont_? \cup Cont_!)$; so $\Pi_{\mathbb{N}^V}(u.\lambda, \overline{C}) = \Pi_{\mathbb{N}^V}(u, C) - (\delta_v^{Q_0})_{v \in V} + \Pi_{\mathbb{N}^V}(\varepsilon, Cont_?) + \Pi_{\mathbb{N}^V}(\varepsilon, Cont_!) + (\delta_v^{\psi(\lambda)})_{v \in V}$; then we deduce from Prop. 38 that $\Pi_{\mathbb{N}^V}(\varepsilon, Cont_?) \in \gamma_{\mathbb{N}^V}(\beta_{\mathcal{N}_V}(P))$ and that $\Pi_{\mathbb{N}^V}(\varepsilon, Cont_!)) \in \gamma_{\mathbb{N}^V}(\beta_{\mathcal{N}_V}(Q))$, we also have $\Pi_{\mathbb{N}^V}(u, C) \in \gamma_{\mathcal{N}_V}(sync_{\mathcal{N}_V}(\{i, j\}, v^\sharp))$, and we conclude that $\Pi_{\mathbb{N}^V}(u.\lambda, \overline{C}) \in \gamma_{\mathcal{N}_V}(sync_{\mathcal{N}_V}(\{i, j\}, v^\sharp) - 1_{\mathcal{N}_V}(Q_0) + 1_{\mathcal{N}_V}(\psi(\lambda)) + \beta_{\mathcal{N}_V}(P) + \beta_{\mathcal{N}_V}(Q))$; so $(u.\lambda, \overline{C}) \in (\gamma_{\mathbb{N}^V} \circ \gamma_{\mathcal{N}_V})(\overline{v}^\sharp)$;

(3) in the case that $C \overset{\lambda}{\nrightarrow} \overline{C}$ is a spied communication: there exists one syntactic component Q_0, such that Q_0 matches $x!^j[\overline{x}]Q$, such that there exists one thread $(Q_0, id_!, E_!)$ in C, a continuation $Cont_! \in \beta(Q, id_!, E_!)$ such that $\overline{C}$ is $((C \setminus \{(Q_0, id_!, E_!)\}) \cup Cont_!)$; we have $(u, C) \in (\gamma_{\mathbb{N}^V} \circ \gamma_{\mathcal{N}_V})(v^\sharp)$; so $\Pi_{\mathbb{N}^V}(u, C) \in \gamma_{\mathcal{N}_V}(v^\sharp)$; and since $(Q_0, id_!, E_!)$ is a thread of C, we have $(\Pi_{\mathbb{N}^V}(u, C))_{Q_0} \geq 1$; then $\Pi_{\mathbb{N}^V}(u, C) \in \gamma_{\mathcal{N}_V}(sync_{\mathcal{N}_V}(\{i\}, v^\sharp))$; since $\gamma_{\mathcal{N}_V}$ is strict, we can conclude that $sync_{\mathcal{N}_V}(\{i\}, v^\sharp) \neq \bot_{\mathcal{N}_V}$; so we can introduce $\overline{v}^\sharp$ such that $C^\sharp \overset{\lambda}{\rightsquigarrow}_{\mathcal{N}_V} \overline{C}^\sharp$; we now prove that $(u.\lambda, \overline{C}) \in (\gamma_{\mathbb{N}^V} \circ \gamma_{\mathcal{N}_V})(\overline{v}^\sharp)$: we have $\overline{C} = ((C \setminus \{(Q_0, id_!, E_!)\}) \cup Cont_!)$; so $\Pi_{\mathbb{N}^V}(u.\lambda, \overline{C}) = \Pi_{\mathbb{N}^V}(u, C) - (\delta_v^{Q_0})_{v \in V} + \Pi_{\mathbb{N}^V}(\varepsilon, Cont_!) + (\delta_v^{\psi(\lambda)})_{v \in V}$; then we can deduce from Prop. 38 that $\Pi_{\mathbb{N}^V}(\varepsilon, Cont_!) \in \gamma_{\mathbb{N}^V}(\beta_{\mathcal{N}_V}(Q))$; we also have $\Pi_{\mathbb{N}^V}(u, C) \in \gamma_{\mathcal{N}_V}(sync_{\mathcal{N}_V}(\{i\}, v^\sharp))$; so $\Pi_{\mathbb{N}^V}(u.\lambda, \overline{C}) \in \gamma_{\mathcal{N}_V}(sync_{\mathcal{N}_V}(\{i\}, v^\sharp) - 1_{\mathcal{N}_V}(Q_0) + 1_{\mathcal{N}_V}(\psi(\lambda)) + \beta_{\mathcal{N}_V}(P))$; so we can conclude that $(u.\lambda, \overline{C}) \in (\gamma_{\mathbb{N}^V} \circ \gamma_{\mathcal{N}_V})(\overline{v}^\sharp)$;

(4) in the case that $C \overset{\lambda}{\nrightarrow} \overline{C}$ is a spoiled communication: there exists one syntactic component P_0, such that P_0 matches $y?^i[\overline{y}]P$ such that there exists one thread $(P_0, id_?, E_?)$ in C, and several tagged names $c_1, ..., c_n \in \mathcal{N} \times \mathcal{M}$, a continuation $Cont_? \in \beta(P, id_?, E_?[y_i \mapsto c_i])$ such that $\overline{C}$ is $((C \setminus \{(P_0, id_?, E_?)\}) \cup Cont_?)$; we have $(u, C) \in (\gamma_{\mathbb{N}^V} \circ \gamma_{\mathcal{N}_V})(v^\sharp)$; so $\Pi_{\mathbb{N}^V}(u, C) \in \gamma_{\mathcal{N}_V}(v^\sharp)$; then since $(P_0, id_?, E_?)$ is a thread of C, we have $(\Pi_{\mathbb{N}^V}(u, C))_{P_0} \geq 1$; then $\Pi_{\mathbb{N}^V}(u, C) \in \gamma_{\mathcal{N}_V}(sync_{\mathcal{N}_V}(\{i\}, v^\sharp))$; since $\gamma_{\mathcal{N}_V}$ is strict, we can conclude that $sync_{\mathcal{N}_V}(\{i\}, v^\sharp) \neq \bot_{\mathcal{N}_V}$; so we can introduce $\overline{v}^\sharp$ such that $C^\sharp \overset{\lambda}{\rightsquigarrow}_{\mathcal{N}_V} \overline{C}^\sharp$; we now prove that $(u.\lambda, \overline{C}) \in (\gamma_{\mathbb{N}^V} \circ \gamma_{\mathcal{N}_V})(\overline{v}^\sharp)$: we have $\overline{C} = ((C \setminus \{(P_0, id_?, E_?)\}) \cup Cont_?)$; so $\Pi_{\mathbb{N}^V}(u.\lambda, \overline{C}) = \Pi_{\mathbb{N}^V}(u, C) - (\delta_v^{P_0})_{v \in V} + \Pi_{\mathbb{N}^V}(\varepsilon, Cont_?) + (\delta_v^{\psi(\lambda)})_{v \in V}$; we deduce from Prop. 38 that $\Pi_{\mathbb{N}^V}(\varepsilon, Cont_?) \in \gamma_{\mathbb{N}^V}(\beta_{\mathcal{N}_V}(P))$; we have already proved $\Pi_{\mathbb{N}^V}(u, C) \in \gamma_{\mathcal{N}_V}(sync_{\mathcal{N}_V}(\{i\}, v^\sharp))$; so $\Pi_{\mathbb{N}^V}(u.\lambda, \overline{C}) \in \gamma_{\mathcal{N}_V}(sync_{\mathcal{N}_V}(\{i\}, v^\sharp) - 1_{\mathcal{N}_V}(P_0) + 1_{\mathcal{N}_V}(\psi(\lambda)) + \beta_{\mathcal{N}_V}(P))$; so we can conclude that $(u.\lambda, \overline{C}) \in (\gamma_{\mathbb{N}^V} \circ \gamma_{\mathcal{N}_V})(\overline{v}^\sharp)$;

(5) in the case that $C \overset{\lambda}{\nrightarrow} \overline{C}$ is a spoiled resource replication: there exists one syntactic component P_0, such that P_0 matches $*y?^i[\overline{y}]P$ such that there exists one thread $(P_0, id_?, E_?)$ in C, and several tagged names $c_1, ..., c_n \in \mathcal{N} \times \mathcal{M}$, a marker $id_* \in \mathcal{M}$, and a continuation $Cont_? \in \beta(P, id_*, E_?[y_i \mapsto c_i])$ such that $\overline{C}$ is $(C \cup Cont_?)$; we have $(u, C) \in (\gamma_{\mathbb{N}^V} \circ \gamma_{\mathcal{N}_V})(v^\sharp)$; so $\Pi_{\mathbb{N}^V}(u, C) \in \gamma_{\mathcal{N}_V}(v^\sharp)$; and since $(P_0, id_?, E_?)$ is a thread of C, we have $(\Pi_{\mathbb{N}^V}(u, C))_{P_0} \geq 1$; then $\Pi_{\mathbb{N}^V}(u, C) \in \gamma_{\mathcal{N}_V}(sync_{\mathcal{N}_V}(\{i\}, v^\sharp))$; since $\gamma_{\mathcal{N}_V}$ is strict, we can conclude that $sync_{\mathcal{N}_V}(\{i\}, v^\sharp) \neq \bot_{\mathcal{N}_V}$; so we can introduce $\overline{v}^\sharp$ such that $C^\sharp \overset{\lambda}{\rightsquigarrow}_{\mathcal{N}_V} \overline{C}^\sharp$; we now prove that $(u.\lambda, \overline{C}) \in (\gamma_{\mathbb{N}^V} \circ \gamma_{\mathcal{N}_V})(\overline{v}^\sharp)$: we have $\overline{C} = (C \cup Cont_?)$; so $\Pi_{\mathbb{N}^V}(u.\lambda, \overline{C}) = \Pi_{\mathbb{N}^V}(u, C) + \Pi_{\mathbb{N}^V}(\varepsilon, Cont_?) + (\delta_v^{\psi(\lambda)})_{v \in V}$; we deduce from Prop. 38 that $\Pi_{\mathbb{N}^V}(\varepsilon, Cont_?) \in \gamma_{\mathbb{N}^V}(\beta_{\mathcal{N}_V}(P))$; we also have $\Pi_{\mathbb{N}^V}(u, C) \in \gamma_{\mathcal{N}_V}(sync_{\mathcal{N}_V}(\{i\}, v^\sharp))$; so we can conclude that $\Pi_{\mathbb{N}^V}(u.\lambda, \overline{C}) \in \gamma_{\mathcal{N}_V}(sync_{\mathcal{N}_V}(\{i\}, v^\sharp) + 1_{\mathcal{N}_V}(\psi(\lambda)) + \beta_{\mathcal{N}_V}(P))$; so $(u.\lambda, \overline{C}) \in (\gamma_{\mathbb{N}^V} \circ \gamma_{\mathcal{N}_V})(\overline{v}^\sharp)$. □