On abstract interpretation of mobile ambients

Francesca Levi and Sergio Maffeis

Abstract

We introduce an abstract interpretation framework for mobile ambients, based on a new semantics called normal semantics. Then, we derive within this setting two analyses computing a safe approximation of the run-time topological structure of processes. Such a static information can be successfully used to establish interesting security properties.

1. Introduction

Mobile ambients (MA) [10] has recently emerged as a core programming language for the Web, and at the same time as a model for reasoning about properties of mobile processes. MA is based on the notion of ambient. An ambient is a bounded place, where multi-threaded computation takes place; roughly speaking, it generalises both the idea of agent and the idea of location. Each ambient has a name, a collection of local processes and a collection of subambients. Ambients are organised in a tree, which can be dynamically modified, according to three basic capabilities: \( \text{in} n \) allows an ambient to enter into an ambient \( n \) (\( \text{in} n. P_1 \mid P_2 \mid Q \rightarrow n[m[P_1 \mid P_2] \mid Q] \)); \( \text{out} n \) allows an ambient to exit from an ambient \( n \) (\( \text{out} n. P_1 \mid P_2 \mid Q \rightarrow m[P_1 \mid P_2] \mid n[Q] \)); \( \text{open} n \) allows to destroy the boundary of an ambient \( n \) (\( \text{open} n. P \mid n[Q] \rightarrow P \mid Q \)).
Several static techniques, formalised as Type Systems \([2,3,6–9,11,17,19,21]\) or Control Flow Analyses (CFA) in Flow Logic style \([5,16,20,25,26]\), have been devised to study and establish various security properties of MA, such as secrecy and information flow. These approaches are strictly related and compute safe approximations of similar information on the run-time topological structure of processes. Although these methods are proved sound with respect to a formal semantics, they are typically formulated in different styles. As a consequence, it is rather difficult to formally compare them, and the corresponding algorithms for constructing the least analysis or for type-inference.

In this paper we apply to MA the semantic-based approach to program analysis of Abstract Interpretation \([13,14]\). Abstract interpretation provides a rigorous theory to derive program analyses from the specification of the semantics. The typical abstract interpretation approach consists of: replacing the concrete domain of computation with an abstract domain modeling the property we are interested in; establishing a relation between the concrete and the abstract domain which formalises (through Galois connections) safeness and precision of approximations; deriving an approximate semantics over the abstract domain. The approximate semantics can be obtained in a systematic way which guarantees its safeness by construction. We refer the reader to Section 2 for more details on the basic concepts of the abstract interpretation theory.

One of the most important and critical steps for applying abstract interpretation consists of the choice of the concrete semantics one should start from. The standard reduction semantics of MA \([10]\) is not adequate to abstraction, because it heavily relies on the syntax by using structural rules and structural congruence to bring the participants of a potential reaction into contiguous positions. We therefore introduce a new semantics for MA, called normal semantics, which is indeed equivalent to the standard reduction semantics. The normal semantics is based on the simple observation that an MA process is essentially a tree, where each node is an ambient containing a set of local processes controlling its movements. Then we derive, by step-wise abstraction of the normal semantics, two analyses which are proved to be safe.

The first analysis is designed to compute an approximation of the following property of all the computations of a process \(P\): for any ambient \(n\), which ambients and capabilities may be contained (at top level) inside \(n\), when \(n\) is within an ambient \(h\). This is obtained by an abstraction which combines information about the number of occurrences of objects and about the context. The integration of these two aspects permits to achieve very accurate results. To substantiate this claim, we consider a typical example: an ambient \(n\) which moves inside an immobile ambient \(k\), and then is opened unleashing an immobile process inside \(k\). This kind of situation is critical in MA, if we want to prove statically the immobility of \(k\), as it is necessary to detect that any capability of movement inside \(n\) has been consumed before opening. Example 5.11 shows that our analysis achieves this result, in particular because it is able to argue on the temporal ordering of execution of capabilities. We are not aware of similar results in the setting of MA without adopting more complex techniques \([1,26]\). It is well known instead that this problem can be solved with simpler techniques for variants of MA, such as Safe Ambients (SA) \([21,22]\). The static techniques for SA \([2,16,17,19,21,22]\) are typically more precise due to the presence of coactions, which control when an interaction may happen. For instance, the coaction \(\text{open}\) simplifies the task of distinguishing what happens inside an ambient before and after it is opened. Similar results have been obtained also for MA extended with primitives for objective mobility \([7]\).

The second analysis is designed to compute an approximation of the following weaker property of all the computations of a process \(P\): for any ambient \(n\), which ambients and capabilities may be contained
inside $n$. This is obtained from the first analysis by dropping off both the contextual information and the information about the number of occurrences of objects. The analysis we obtain is a refined version of the CFA of [25]. The main difference with respect to [25] is that our analysis considers the effect of the continuation of a capability only if the capability may be exercised. Example 6.11 shows in details the difference with the CFA of [25].

The properties computed by both the analyses permit to control where an ambient may move and also where it may be opened. This is the basic information which is needed to statically establish most of the security properties studied in the literature for MA [5, 6, 9, 16, 17, 25]. To illustrate the relevance of the analysis for security we show the application to some well-known examples taken from [5, 16]. We focus on the first analysis which is more precise and interesting; the second analysis can be used, as the CFA of [25], to solve simpler problems, such as the firewall protocol of [10] and the Trojan Horse of [6].

The normal semantics is presented in Section 4, and the two derived abstractions in Sections 5 and 6, respectively. Section 7 shows some examples of security properties. The proof of the main theorems are shown in Appendixes A and B.

**Remark.** This paper is an extended and revised version of [23].

### 2. Some background on abstract interpretation

We briefly recall the basic concepts of the Galois connection based approach of abstract interpretation [13, 14]. Suppose we want to approximate a semantics $S$, which is computed as the least fixed-point of a monotonic function $F$ over some concrete domain $⟨C, ≤⟩$. The key step consists of the choice of an abstract domain $⟨A, ≤^a⟩$ modeling the property we want to statically establish. The notion of Galois connection formalises the relation of abstraction between the concrete and the abstract domain which is the basis to define safeness and precision of approximations.

**Definition 2.1** (Galois connection). Let $⟨C, ≤⟩$ and $⟨A, ≤^a⟩$ be complete lattices. A pair of monotonic functions $(α, γ)$, such that $α : C → A$ is the abstraction function and $γ : A → C$ is the concretization function, is a Galois connection between $⟨C, ≤⟩$ and $⟨A, ≤^a⟩$ iff, for each $c ∈ C$ and $a ∈ A$

1. $c ≤ γ(α(c))$;
2. $α(γ(a)) ≤^a a$.

When $α(γ(a)) = a$, then $(α, γ)$ is called a Galois insertion.

The ordering $≤^a$ is intended to model precision so that $a ≤^a a'$ means that $a'$ is a safe approximation of $a$. Therefore, the abstraction of the least fixed-point $α(S)$ gives the exact abstract property corresponding to $S$, and an approximate semantics $S^a$ over the abstract domain is a safe approximation of $S$ whenever $α(S) ≤^a S^a$. One of the main results of abstract interpretation is that a safe approximate semantics $S^a$ can be computed as the least fixed-point of an abstract function $F^a$ satisfying a condition of local safeness, namely that $α(F(c)) ≤^a F^a(α(c))$.

**Theorem 2.2** (Safeness). Let $(α, γ)$ be a Galois connection between $⟨C, ≤⟩$ and $⟨A, ≤^a⟩$. Moreover, let $F : C → C$ and $F^a : A → A$ be monotonic functions. If $α(F(c)) ≤^a F^a(α(c))$, for each $c ∈ C$, then $α(lfp F) ≤^a lfp F^a$. 

3. Mobile ambients

We introduce the pure mobile ambients calculus (see [10]) without communication primitives. Let \( \mathcal{N} \) be a set of names (ranged over by \( n, m, h, k, \ldots \)).

**Definition 3.1 (Processes).** The processes are defined over names \( \mathcal{N} \) according to the following syntax:

\[
\begin{align*}
M, N &::= \text{(capabilities)} \\
P, Q &::= \text{(processes)} \\
\text{in} n &\quad \text{enter } n \\
\text{out} n &\quad \text{exit } n \\
\text{open} n &\quad \text{open } n \\
\end{align*}
\]

\[
\begin{align*}
P &\rightarrow \text{inactivity} \\
(\nu n) P &\rightarrow \text{restriction} \\
P | Q &\rightarrow \text{parallel composition} \\
!P &\rightarrow \text{replication} \\
n[P] &\rightarrow \text{ambient} \\
M . P &\rightarrow \text{prefix}
\end{align*}
\]

Standard syntactical conventions are used: trailing zeros are omitted, and parallel composition has the least syntactic precedence. We refer to the usual notions of names, free names, and bound names of a process \( P \), denoted by \( n(P) \), \( fn(P) \), \( bn(P) \), respectively. We identify processes which are \( \alpha \)-convertible, that is, can be made syntactically equal by a change of bound names. We adopt also the standard notation for substitutions: \( P[m/n] \) denotes the process obtained by replacing in \( P \) any free occurrence of \( n \) with \( m \) (assuming the bound names of \( P \) are \( \alpha \)-converted to avoid the conflicts with \( m \)). Similarly, \( P\eta \) denotes the process obtained by applying the substitution \( \eta : \mathcal{N} \rightarrow \mathcal{N} \).

The core of the semantics of MA consists of the reductions in Table 1 corresponding to the execution of capabilities. The semantics has also standard structural rules (Table 2) which use structural congruence to bring the participants of a potential interaction into contiguous positions (Table 3). The definition of \( \equiv \) includes the standard rules for commuting the positions of parallel components, for stretching the scope of a restriction and for replication.

In the following we use \( \rightarrow^* \) for the transitive and reflexive closure of \( \rightarrow \). Moreover, we write \( P \rightarrow\equiv Q \) to say that either \( P \rightarrow Q \) or \( P \equiv Q \). Similarly for \( P \rightarrow^\equiv Q \).

4. The normal semantics

The normal semantics aims at making easier the application of abstract interpretation, which is complicated by structural congruence (including \( \alpha \)-conversion) and by the structural rules of the reduction semantics. The normal semantics is based on the intuitive representation of a process as a tree of ambients, each containing a set of active processes. We use a set, called a topology, to represent the tree of ambients, and a set, called a configuration, to represent the active processes contained in each ambient. For instance, the process

<table>
<thead>
<tr>
<th>Table 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Basic reductions of mobile ambients</td>
</tr>
<tr>
<td>---</td>
</tr>
<tr>
<td>( n[\text{in} m] . P \mid Q \mid [m[R] \rightarrow m[n[P \mid Q] \mid R] ) (In)</td>
</tr>
<tr>
<td>( m[n[\text{out} m] . P \mid Q \mid [R] \rightarrow n[P \mid Q] \mid m[R] ) (Out)</td>
</tr>
<tr>
<td>( \text{open} n . P \mid n[Q] \rightarrow P \mid Q ) (Open)</td>
</tr>
</tbody>
</table>
Table 2
Structural rules for mobile ambients

<table>
<thead>
<tr>
<th>Rule</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$P \rightarrow Q \Rightarrow (\nu n) P \rightarrow (\nu n) Q$</td>
<td>(Res)</td>
</tr>
<tr>
<td>$P \rightarrow Q \Rightarrow P</td>
<td>R \rightarrow Q</td>
</tr>
<tr>
<td>$P \rightarrow Q \Rightarrow n[P] \rightarrow n[Q]$</td>
<td>(Amb)</td>
</tr>
<tr>
<td>$(P' \rightarrow Q', P \equiv P', Q' \equiv Q) \Rightarrow P \rightarrow Q$</td>
<td>(Cong)</td>
</tr>
</tbody>
</table>

Table 3
Structural congruence

<table>
<thead>
<tr>
<th>Rule</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$P \equiv P$</td>
<td>(Refl)</td>
</tr>
<tr>
<td>$Q \equiv P \Rightarrow P \equiv Q$</td>
<td>(Symm)</td>
</tr>
<tr>
<td>$P \equiv Q, Q \equiv R \Rightarrow P \equiv R$</td>
<td>(Trans)</td>
</tr>
<tr>
<td>$(P</td>
<td>Q) \equiv (Q</td>
</tr>
<tr>
<td>$P \equiv Q \Rightarrow n[P] \equiv n[Q]$</td>
<td>(Amb)</td>
</tr>
<tr>
<td>$P \equiv Q \Rightarrow M \cdot P \equiv M \cdot Q$</td>
<td>(Pref)</td>
</tr>
<tr>
<td>$n \neq m \Rightarrow (\nu n) (\nu m) P \equiv (\nu m) (\nu n) P$</td>
<td>(Res-Com)</td>
</tr>
<tr>
<td>$n \neq fn(P) \Rightarrow (\nu n)(P</td>
<td>Q) \equiv P</td>
</tr>
<tr>
<td>$n \neq m \Rightarrow (\nu n) m[P] \equiv m[(\nu n) P]$</td>
<td>(Res-Amb)</td>
</tr>
<tr>
<td>$P</td>
<td>0 \equiv P$</td>
</tr>
<tr>
<td>$(\nu n) 0 \equiv 0$</td>
<td>(Nil-Res)</td>
</tr>
<tr>
<td>$! P \equiv P !</td>
<td>! P$</td>
</tr>
</tbody>
</table>

The translation of a process into an equivalent pair of topology and configuration, as shown for process (1) above, presents two subtle problems. We need to: (i) distinguish two different occurrences of the same object in the process; (ii) choose properly the names used for the removal of restrictions. In (1), for instance, we have eliminated the restriction operator by substituting $n$ with a fresh name (in this case it suffices to take $n$ itself).

To deal with these problems in a simple way we enhance the syntax of processes by properly attaching labels to capabilities, restrictions and ambients.

Provided that the labels assigned to capabilities, restrictions and ambients are distinct, we directly obtain a representation, where two copies of the same process or of an ambient with the same name are
distinguished. For instance, consider the following labeled version of process $n[in m] | n[in k]$, where labels $\lambda, \mu, \gamma, \xi$ are distinct one from each other

$$n_\lambda[in m_\gamma] | n_\mu[in k_\xi].$$ (2)

We obtain the following representation:

$$\langle \{ n_\lambda \@, n_\mu \@ \}, \{ n_\lambda \in m_\gamma, n_\mu \in k_\xi \} \rangle$$

where there are two copies of ambient $n$: one containing the capability $in m$ and the other one containing the capability $in k$.

We also use the labels attached to restrictions to find out the name, which is used to replace the bound name. To this aim, we adopt a special substitution function, which associates in a one to one fashion names to labels. Provided that all the labels are distinct and that the names associated to the labels of restrictions do not appear in the process, the names introduced by the removal of restrictions are fresh. For instance, consider the following labeled process:

$$(\nu n_\lambda)( n_\gamma \in m_\mu . P) | (\nu m_\beta) m_\xi [0]$$ (3)

where the labels $\lambda, \gamma, \mu, \beta$ are distinct one from each other. Assume also that $\hat{n}$ and $\hat{m}$ are the distinct names associated to $\lambda$ and $\beta$ and that they do not appear in the process. We obtain the following representation

$$\langle \{ \hat{n}_\gamma \@, \hat{m}_\xi \@ \}, \{ \hat{n} \in m_\mu . P \} \rangle.$$ $\hat{n}_\gamma$ is replaced by $n_\gamma$ and $\hat{m}_\xi$ is replaced by $m_\xi$.

The removal of the restrictions over $n$ and $m$ does not produce any conflict on names, as $\hat{m} \neq \hat{n}$, $\hat{m} \neq m$ and $\hat{n} \neq m$. The condition $\hat{m} \neq \hat{n}$ is implied by $\lambda \neq \beta$; the conditions $\hat{m} \neq m$ and $\hat{n} \neq m$ are ensured by the additional requirement concerning the names and the labels appearing in the process.

The requirements on labels and names explained above are formalised by the notion of well-labeled process (see Definition 4.2).

**Labeled processes.** Let $\mathcal{L}$ be a set of labels (ranged over by $\ell, \ell', \ldots$), and let $\mathcal{L}_I = \{ \ell_i \mid \ell \in \mathcal{L}, i \in I \}$ be the corresponding set of indexed labels (ranged over by $\lambda, \mu, \gamma, \ldots$). Let $\hat{\mathcal{N}}$ (ranged over by $\hat{n}, \hat{m}, \hat{\lambda}, \hat{\mu}, \hat{\gamma}, \ldots$) be a set of names, such that $\mathcal{N} \cap \hat{\mathcal{N}} = \emptyset$, and let $\hat{\mathcal{N}}_I = \{ \hat{n}_i \mid \hat{n} \in \hat{\mathcal{N}}, i \in I \}$ be the corresponding set of indexed names.

We use the names $\hat{\mathcal{N}}_I$ for the elimination of restrictions according to a substitution function $H_{\mathcal{L}_I}$ which assigns indexed names $\hat{\mathcal{N}}_I$ to indexed labels $\mathcal{L}_I$. This is formalised by an injective function $H_{\mathcal{L}} : \mathcal{L} \rightarrow \hat{\mathcal{N}}$ and by the corresponding injective function $H_{\mathcal{L}_I} : \mathcal{L}_I \rightarrow \hat{\mathcal{N}}_I$, such that $H_{\mathcal{L}_I}(\ell_i) = \hat{n}_i$ if $H_{\mathcal{L}}(\ell) = \hat{n}$.
To have a more compact notation we may use when the distinction is not relevant: \(n, m, h, \ldots\) to denote a generic element of \(\hat{N}_I \cup N; \hat{n}, \hat{m}, \hat{h}, \ldots\) to denote a generic element of \(\hat{N}_I\).

**Definition 4.1 (Labeled processes).** The labeled processes are defined over names \(N \cup \hat{N}_I\) and indexed labels \(L_I\) according to the following syntax:

\[
\begin{align*}
M, N, & ::= \text{(capabilities)} \\
P, Q, & ::= \text{(processes)} \\
\text{in } n & \quad \text{enter } n \\
\text{out } n & \quad \text{exit } n \\
\text{open } n & \quad \text{open } n \\
\in n & \quad \text{inactivity} \\
(\nu n)_P & \quad \text{restriction} \\
P \mid Q & \quad \text{parallel composition} \\
!P & \quad \text{replication} \\
\hat{n}_j[P] & \quad \text{ambient} \\
M\lambda. P & \quad \text{prefix}
\end{align*}
\]

We assume that all the notions presented in Section 3 are adapted in the obvious way to labeled processes. The definition of \(\alpha\)-conversion only presents a subtle point: we require that the bound names can be changed but not their labels. We mean, for instance, that \((\nu n)_P\) is \(\alpha\)-convertible to \((\nu k)_P[k/n]\), provided that \(k \not\in fn(P)\), and not to \((\nu k\mu)_P[k/n]\). In the following, we use \(A(P)\) to denote the set of labels occurring in a labeled process \(P\).

We introduce now the concept of well-labeled process, which formalises the requirements discussed for the processes (2) and (3) above. Conditions (i) and (ii) say that the labels are distinct and the names associated to the labels of restrictions are fresh names, meaning that they do not occur in the process. Example 4.10 shows, more in details, why these requirements are needed to translate a process into an equivalent representation.

**Definition 4.2 (Well-labeled processes).** A process \(P\) is well-labeled if: (i) for any \(\lambda \in A(P), H_{L_I}(\lambda) \not\in n(P)\); (ii) the (indexed) labels used in capabilities, ambients and restrictions are distinct one from each other.

Over labeled processes we define a notion of equivalence, which is used in the definition of the collecting semantics (see Definition 4.8). A renaming of indexed labels is a function \(\rho : L_I \rightarrow L_I\). The application of a renaming is denoted in the standard way by \(P\rho\) and \(P[\lambda/\mu]\). We denote by \(dom(\rho)\) and \(dom(\eta)\) the domains of a renaming \(\rho\) and a substitution \(\eta\), respectively. We also introduce a special class of renamings and substitutions:

- we say that \(\rho_I : L_I \rightarrow L_I\) is a re-indexing of labels if, \(\rho_I\) is injective, and for any \(\ell_i \in dom(\rho_I)\), we have \(\rho_I(\ell_i) = \ell_j\);
- we say that \(\eta_I : \hat{N}_I \rightarrow \hat{N}_I\) is a re-indexing of names if, \(\eta_I\) is injective and, for any \(\hat{n}_i \in dom(\eta_I)\), we have \(\eta_I(\hat{n}_i) = \hat{n}_j\).

We say that \(P\) and \(Q\) are equivalent up to re-indexing \((P \sim Q)\) iff \(P\rho_I\eta_I = Q\), for a re-indexing of labels \(\rho_I\) and a re-indexing of names \(\eta_I\).

In the following, we use \(A\) (ranged over by \(a, b, c, \ldots\)) for the set of labeled names \(n_\lambda\), such that \(n \in N \cup \hat{N}_I\) and \(\lambda \in L_I\), augmented with the distinct symbol \(@\) representing the outermost ambient. Furthermore, we say that a process \(P\) is active if \(P = M_\lambda. Q\) or \(P = !Q\). We use \(P\) and \(AP\) to denote the
set of well-labeled processes (referred to as processes) and the subset of active well-labeled processes, respectively.

**Remark 4.3.** It is worth mentioning that the labeling of processes is also exploited by the analyses of Sections 5 and 6. This approach is indeed typical of static techniques, in particular of Flow Logic [27]. The labeling of processes is used to gain precision, and also it allows the programmer to identify the exact piece of input syntax responsible for some detected security violation. The main difference here consists in the use of indexes both in labels $L_I$ and in names $\hat{N}_I$. The normal semantics and the second abstraction could have been defined also without introducing the indexes. Instead, the indexes are needed and fruitfully exploited by the first abstraction (see Examples 5.10 and 5.12).

**States and transitions.** A state is a pair which consists of a topology and a configuration: the topology is a set of pairs, son–father, which form a tree, and the configuration is a set of pairs associating each active process to its enclosing ambient.

**Definition 4.4 (States).** A state $S$ is a pair $(T, C)$ where
1. $T \in \wp((A \setminus \{\@\} \times A)$ is a tree over a set of nodes $N_S \subseteq A$;
2. $C \in \wp(A \times AP)$ such that for each $(a, P) \in C$, $a \in N_S$.

In a state $(T, C)$ we call $T$ a topology and $C$ a configuration. The meaning of $(a, b) \in T$ (for short $a^b$) is that $a$ is a son of $b$. The meaning of $(a, P) \in C$ (for short $a^P$) is that $P$ is an active process of ambient $a$.

We extend to states in the obvious way the notions of labels, renaming, substitution and equivalence up to re-indexing $\sim$. Since we are interested in states representing well-labeled processes we consider only well-labeled states. A state $S \in \mathcal{S}$ is well-labeled if: (i) for each $\lambda \in \text{afii9806}(S)$, $H_{LI}(\lambda) \notin n(S)$; (ii) for any label $\lambda \in \text{afii9806}(S)$ there is at most one object labeled by $\lambda$. In the following, we use $\mathcal{S}$ to denote the set of well-labeled states (referred to as states). Also, we assume $\subseteq$ and $\cup$ over states are defined component-wise.

In Table 4 we introduce the *normalisation function* $\delta : (A \times P) \to S$ which is used to translate processes into states. Intuitively, $\delta(a, P)$ (for short $a^P$) gives the state representing process $P$, assuming that $P$ is contained in ambient $a$. We use $\delta$ both to derive the initial state from a process, and to handle the processes which become executable after a step. The initial state corresponding to a process $P$ is therefore $\delta^{\@P}$.

Rule $\text{DRes}$ eliminates the restriction by replacing the bound name $n$ with the name $H_{LI}(\lambda)$ associated to the indexed label $\lambda$. The definition of well-labeling ensures that $H_{LI}(\lambda)$ is a fresh name provided that $P$ is a well-labeled process. Rule $\text{DAmb}$ adds ambient $b$ to the topology as son of the enclosing ambient $a$, and translates the process contained in $b$. Rule $\text{DPar}$ gathers the processes and the topologies built in each of its two branches. Rules $\text{DBang}$ and $\text{DPref}$ simply add the active process to the configuration.

The rules of Table 5 define the *transitions* between states. They realise the unfolding of replication, the movements in and out of ambients, and the opening of ambients. Due to the implicit representation of parallel composition, restriction and ambient in states, the standard structural rules and structural

---

1 We refer to the standard definition of tree and root of a tree.
Table 4
The normalisation function $\delta$

<table>
<thead>
<tr>
<th>Function</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>DRes $\delta a(\nu n\lambda) P$</td>
<td>$\delta a(P[H_L(\lambda)/n])$</td>
</tr>
<tr>
<td>DAmb $\delta a[P]$</td>
<td>$\delta bP \cup ({a}, \emptyset)$</td>
</tr>
<tr>
<td>DZero $\delta a$</td>
<td>$\emptyset$</td>
</tr>
<tr>
<td>DPar $\delta aP \mid Q$</td>
<td>$\delta aP \cup \delta aQ$</td>
</tr>
<tr>
<td>DBang $\delta a! P$</td>
<td>$\emptyset \cup {a!P}$</td>
</tr>
<tr>
<td>DPref $\delta aM_\lambda . P$</td>
<td>$\emptyset \cup {aM_\lambda . P}$</td>
</tr>
</tbody>
</table>

Table 5
Transitions $\mapsto$

<table>
<thead>
<tr>
<th>Transition</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bang $a_i P \in C$</td>
<td>$(T, C) \mapsto \delta a_{\text{new}(T,C)}(P) \cup (T, C)$</td>
</tr>
<tr>
<td>In $a_{\text{in}} m_y . P \in C$</td>
<td>$\delta aP \cup ((T \setminus {a_i}) \cup {a^{m_y}, C \setminus {t}})$</td>
</tr>
<tr>
<td>Out $a_{\text{out}} m_y . P \in C$</td>
<td>$\delta aP \cup ((T \setminus {a^{m_y}}) \cup {a_i}, C \setminus {t})$</td>
</tr>
<tr>
<td>Open $a_{\text{open}} m_y . P \in C$</td>
<td>$\delta aP \cup ((T \setminus {a^{m_y}}) \cup {a_i}){a/m_y}$</td>
</tr>
</tbody>
</table>

Congruence of the reduction semantics are not needed. For notational convenience use the following abbreviation. We write $(T, C)[a/b]$ to denote the state $(T[\{a_i/\ldots c_i\}, C[\{aQ/bQ\}])$ for any ambient $c$ and process $Q$.

We comment the rules below. Rule **Bang** creates a fresh copy (equivalent up to re-indexing of labels) of the process under replication. To this aim, we use $\text{new}(T,C)(P)$, which is defined as follows. Let $S \in S$ be a state, we let $\text{new}_S(P) = P_{\rho I}$, where

- $\rho I$ is a re-indexing of labels such that $\text{dom}(\rho I) = A(P)$;
- $P_{\rho I}$ is well-labeled;
- there is no $\lambda \in A(P_{\rho I})$, such that either $\lambda \in A(S)$ or $H_L(\lambda) \in \eta(S)$.

The definition of $\text{new}_S$ ensures that $\delta a_{\text{new}_S}(P) \cup S$ is a well-labeled state, provided that $S$ and $P$ are well-labeled.

The last three rules correspond to the usual reduction rules of movements and opening (shown in Table 1). They use the normalisation function to handle the continuations. Rule **In** is applicable whenever there exists a parallel ambient named $m$. The rule modifies both the topology and the configuration according to the movement: (i) it updates the father of $a$, which is now $m$, (ii) it removes the executed capability, and it adds the continuation to the set of processes local to $a$. Rule **Out** acts in an analogous way. Rule **Open** modifies both the topology and the configuration according to the opening of ambient $m$: (i) it removes ambient $m$; (ii) it modifies the pointer to the father of any ambient and process which was within $m$. These processes and ambients are therefore acquired by ambient $a$ when opening $m$. 
The following theorem states the agreement between the transitions of Table 5 and the standard reduction semantics of Section 3. Let $P$ be a well-labeled process. We denote by $\mathcal{E}(P)$ the process obtained by stripping off all the labels. We use $\mapsto^*$ for the transitive and reflexive closure of $\mapsto$.

We introduce also a condition on $a \in A$ which guarantees that $\delta^aP$ is a well-labeled state, provided that $P$ is well-labeled (as formalised by Proposition A.13 of Appendix A). We first extend the notions of names $n(a)$ and labels $\Lambda(a)$. Hence, we let $n(a) = n$ and $\Lambda(a) = \lambda$, when $a = n\lambda$, and we let $n(a) = \Lambda(a) = \emptyset$, when $a = \emptyset$.

We say that $a$ is fresh for a labeled process $P$ iff $\Lambda(a) \cap \Lambda(P) = \emptyset$, there is no $\mu \in \Lambda(a)$ such that $H_{\mathcal{L}}(\mu) \in (n(P) \cup n(a))$, and there is no $\mu \in \Lambda(P)$ such that $H_{\mathcal{L}}(\mu) \in n(a)$.

**Theorem 4.5 (Equivalence).** Let $P$ be a well-labeled process and let $a \in A$ which is fresh for $P$.

1. If $\delta^aP \mapsto S$, then there exist a well-labeled process $Q$, such that $\mathcal{E}(P) \mapsto \equiv \mathcal{E}(Q)$ and $\delta^aQ = S$.
2. If $\mathcal{E}(P) \rightarrow Q$, then there exist a state $S$ and a well-labeled process $Q'$, such that $\delta^aP \mapsto^* S$, $\delta^aQ' = S$ and $Q \equiv \mathcal{E}(Q')$.

The proof of Theorem 4.5 is rather complex and is shown in Appendix A.

**Corollary 4.6.** Let $P$ be a well-labeled process.

1. If $\delta^aP \mapsto S$, then there exist a well-labeled process $Q$, such that $\mathcal{E}(P) \mapsto \equiv \mathcal{E}(Q)$ and $\delta^aQ = S$.
2. If $\mathcal{E}(P) \rightarrow Q$, then there exist a state $S$ and a well-labeled process $Q'$, such that $\delta^aP \mapsto^* S$, $\delta^aQ' = S$ and $Q \equiv \mathcal{E}(Q')$.

**Proof.** From Theorem 4.5 using the fact that $\emptyset$ is fresh for any well-labeled process $P$. □

The result can be extended straightforwardly to weak reductions.

**Corollary 4.7.** Let $P$ be a well-labeled process.

1. If $\delta^aP \mapsto^* S$, then there exist a well-labeled process $Q$, such that $\mathcal{E}(P) \mapsto \equiv \mathcal{E}(Q)$ and $\delta^aQ = S$.
2. If $\mathcal{E}(P) \rightarrow^* Q$, then there exist a state $S$ and a well-labeled process $Q'$, such that $\delta^aP \mapsto^* S$, $\delta^aQ' = S$ and $Q \equiv \mathcal{E}(Q')$.

The collecting semantics. We define the core of the abstract interpretation framework, the collecting semantics. The domain is the power-set of (well-labeled) states up to re-indexing. We use $[S]$ to denote the equivalence class of a state $S$ with respect to $\sim$, and we use $S_{/\sim}$ to denote the corresponding quotient set. For readability, we use $\subseteq$ and $\cup$ for $\subseteq_{/\sim}$ and $\cup_{/\sim}$.

**Definition 4.8.** Let $S^c = \emptyset(S_{/\sim})$. The concrete domain is $(S^c, \subseteq)$.

The concrete semantics is defined in a standard way as the least fixed-point of a function, which collects all the states reachable from the initial state.

**Definition 4.9 (Collecting semantics).** Let $S_2 \in S$, $S^c_2 \in S^c$ and let $P$ be a well-labeled process. We define $\mathcal{E}_{\text{Coll}}[P] = \text{lfp} \ F(\delta^aP)$ for the function $F : S \rightarrow (S^c \rightarrow S^c)$ such that $F(S_2) = \Psi_{S_2}$ and
Examples. We start discussing the normalisation function $\delta$, and we explain why this is correct (in the sense of Theorem 4.5) only when applied to well-labeled processes.

Example 4.10. The condition (ii) of Definition 4.2 ensures that two occurrences of the same object are distinguished. Consider the not well-labeled version of process $n[\text{in } m] | n[\text{in } k]$,

$$P = n_\lambda[\text{in } m_\gamma] | n_\lambda[\text{in } k_\zeta].$$

We obtain the following representation:

$$\delta^n_\otimes P = (\{ n_\lambda \otimes \}, \{ n_\lambda[\text{in } m_\gamma], n_\lambda[\text{in } k_\zeta] \}).$$

This representation differs significantly from that shown at the beginning of the section for the well-labeled process (2). In fact there is only one ambient $n$ which contains both $\text{in } m$ and $\text{in } k$. This representation is obviously not correct as ambient $n$ may interact both with $m$ and with $k$.

The condition (i) of Definition 4.2 concerns the relation between the names in $\hat{N}_I$ and the labels $L_I$, and ensures precisely that there is no clash of names when the restrictions are removed. Consider the following not well-labeled process:

$$Q = (\nu n_\lambda)( n_\gamma[\text{in } \nu \hat{m}_\mu]. P) | (\nu m_\beta)[\text{open }] n_\beta[\text{in } k_\zeta]. Q_1,$$

where $H_{L_I}(\lambda) = \hat{n}$ and $H_{L_I}(\beta) = \hat{m}$. We obtain the following representation

$$\delta^n_\otimes Q = (\{ n_\gamma \otimes, n_\gamma \}, \{ n_\gamma[\text{in } \nu \hat{m}_\mu]. P \}).$$

This representation is not correct, differently from the one obtained for the well-labeled process (3) shown at the beginning of the section. The bound name $\hat{m}$ is known to the process contained inside $\hat{n}$, and consequently $\hat{n}$ can move inside $\hat{m}$.

We give some examples to illustrate the normal semantics. To simplify the presentation in the collecting semantics states stand for their equivalence classes up to re-indexing. The following example shows an ambient $n$, which moves inside an ambient $k$, and there is opened unleashing no capability of movement inside $k$.

Example 4.11. Consider the (well-labeled) process

$$P = n_\lambda[\text{in } k_\epsilon. m_\zeta[Q_2]] | k_\mu[\text{open } n_\beta. Q_1]$$

Fig. 2 shows some states, which are reachable from the initial state representing process $P$ which is state (A). State (B) is obtained from state (A) by applying rule $\text{In}$. This shows that ambient $n$ moves inside $k$ carrying any capability and ambient it contains. State (C) is obtained from state (B) by applying rule $\text{Open}$; ambient $n$, when opened inside $k$, unleashes ambient $m$ which has as a local process $Q_2$.

By assuming that $Q_1 = Q_2 = 0$, the collecting semantics of $P$ contains only states (A)-(C) of Fig. 2. We have $\Xi_{Col}[P] = \{ S_0, S_1, S_2 \}$ such that

$$\Psi_{S_2}(S^2) = \{ [S_2] \} \cup \bigcup_{S \in \{ S_2 | S \rightarrow S_2, [S_1] \in S^2 \}} \{ [S] \}.$$
The following example stresses an important aspect concerning indexes and replication: the unfolding of replication produces (by means of new) processes which are equivalent up to re-indexing of labels. The link between the processes produced by replication expressed by the indexes is suitably exploited by the first abstraction (see Examples 5.10 and 5.12). Also, the example explains better the technique used to remove the restriction operator and its interplay with replication.

**Example 4.12.** Consider the well-labeled process \( Q = n_\lambda[\text{\textsc{in}} n_\gamma], \) where \( \lambda = \ell_1 \) and \( \gamma = \ell_1'. \) The initial state modeling process \( !Q \) is \( (\emptyset, @!Q) \). Every unfolding of replication is modeled by the addition of \( \delta @((Q\rho_j)) \) (see rule \textsc{Bang}), where

\[
Q\rho_j = n_\ell_j[\text{\textsc{in}} n_\ell_j]
\]

for a new index \( j \). Hence, a new ambient \( n_\ell_j \) is added representing a new copy of ambient \( n \). For instance, after two applications of rule \textsc{Bang} the state (A) depicted in Fig. 3 is reached.\(^3\) Any ambient \( n_\ell_j \) may enter inside any other \( n_\ell_h \) provided that \( h \neq j \). For instance, by applying rule \textsc{In} state (B) of Fig. 3 is obtained.

Consider instead the well-labeled process \( (\nu n_\mu)Q \), where the name \( n \) is restricted and \( \mu = \ell''_1 \) such that \( H_L(\ell''_1) = \hat{n} \). The initial state modeling process \( !(\nu n_\mu)Q \) is \( (\emptyset, @!(\nu n_\mu)Q) \). Every unfolding of replication is modeled by the addition of \( \delta @(((\nu n_\mu)Q)\rho_j) \), where

\[
((\nu n_\mu)Q)\rho_j = (\nu n_{\ell_j'})(n_\ell_j[\text{\textsc{in}} n_{\ell_j'}])
\]

for a new index \( j \). Function \( H_L \) assigns to any label \( \ell''_j \) the new name \( \hat{n}_j \) which is used to substitute \( n \). Hence, a new ambient \( \hat{n}_j \) is added with a new name \( \hat{n}_j \). For instance, after two applications of rule \textsc{Bang} the state (C) depicted in Fig. 3 is reached from the initial state. Since the names \( \hat{n}_1, \hat{n}_2 \) are distinct, then the ambients cannot in this case interact with each other.

\(^3\) As usual we have omitted labels to simplify the picture.
The difference between $!Q$ and $!(\nu n \mu)Q$ is reflected by their collecting semantics shown below. We have $\Xi_{Coll} [!Q] = \bigcup_{j \in [0, \infty]} X_j$, where

- $X_0 = \{ (\emptyset, @!Q) \}$;
- $X_j$ is the minimal set of states $S = (T, C)$, such that $fn(S) = \{ n \}$, and $A(S) = \bigcup_{i \in [1, \ldots, j]} \{ \ell_i, \ell'_i \}$ and $@!Q \in C$. Moreover, for each $i \in \{1, \ldots, j\}$ either $n_i \in T$ and $n_i \in n_{\ell_i} \in C$, or $n_i \notin T$, with $h \neq i$, and $n_i \in n_{\ell'_{i}} \notin C$.

We have $\Xi_{Coll} [!(\nu n \mu)Q] = \bigcup_{j \in [0, \infty]} S_j$, where

$$S_0 = (\emptyset, @!(\nu n \mu)Q),$$
$$S_j = \left( \bigcup_{i \in [1, \ldots, j]} (\hat{n_i})_{\ell_i} \cap @!(\nu n \mu)Q, \bigcup_{i \in [1, \ldots, j]} (\hat{n_i})_{\ell_i} \cap (\hat{n_i})_{\ell'_i} \cup \{ @!(\nu n \mu)Q \} \right).$$

5. A first abstraction

We devise a first abstraction aimed at capturing the following property about all the states reachable from the initial state representing a process $P$: for each ambient $n$, which ambients and capabilities can be contained (at top level) inside $n$, when $n$ is within an ambient $h$. This is formalised by an abstraction, which merges a set of states into a unique abstract state, and modifies the topology and the configuration according to the following ideas.

- We add to each pair of the topology and of the configuration an additional information which refers to the father of the enclosing ambient.

Consider for instance the states

$$S_1 = (\{ a@, b@ \}, a \in k_\mu, \in m_\gamma),$$
$$S_2 = (\{ a@, b@ \}, a \in m_\gamma).$$

They are represented by the following abstract states, respectively

$$S_1^a = (\{ b@^\top, a@^\top \}, \{ a@^\top \in k_\mu, \in m_\gamma \}),$$
$$S_2^a = (\{ a@^\top, b@^\top \}, \{ a@^\top \in m_\gamma \}).$$
In $S_1^\circ$ we have that $a@\in k_\mu.\in m_\gamma$ as $\in k_\mu.\in m_\gamma$ is an active process inside ambient $a$, when $a$ is within $\oplus$. The same happens in the topology. For instance $a@^\top$ says that ambient $a$ is a son of the top level ambient $\oplus$, when $\oplus$ is within $\top$.

The abstract state $S_2^\circ$ is obtained similarly.

To understand the relevance of the information we have introduced, it is necessary to look at the abstraction of the set of states $\{S_1, S_2\}$. This is given by the union of $S_1^\circ$ and $S_2^\circ$ (depicted also in Fig. 4).

$$S^\circ = (\{a@, b@^\top, a@^\top\}, \{a@\in m_\gamma, a@\in k_\mu.\in m_\gamma\}).$$

The abstract version of $\{S_1, S_2\}$ shows that the abstract topologies, differently from the concrete topologies, may not form a tree. For instance, in $S^\circ$ ambient $a$ has two fathers, namely ambients $b$ and $\oplus$. The additional information permits to distinguish between the multiple fathers of ambient $a$, and consequently to argue that the processes and the ambients contained inside $a$ may depend on where $a$ is located. For instance, in $S^\circ$ we have that: when $a$ is within $\oplus$ process $\in k_\mu.\in m_\gamma$ is executable inside $a$; when $a$ is within $b$ instead process $\in m_\gamma$ is executable inside $a$.

We call this additional information, the partial topology, as it gives us a partial view of the shape of the tree-like structure (the topology) of the state, which contains the pair of ambients, son–father, or the pair associating each active process to its enclosing ambient.

- We abstract indexes by keeping only the following information: whether there is at most one occurrence or many occurrences of an object.

Consider for instance the following states

$$S_1 = (n_i@, n_i'^{open} m_{\ell_1}),$$

$$S_2 = (n_i@, \{n_i'^{open} m_{\ell_1}, n_i'^{open} m_{\ell_2}\}).$$

They are represented by the following abstract states $S_1^\circ$ and $S_2^\circ$, respectively

$$S_1^\circ = (n_i@^\top, n_i'^{open} m_{\ell_1}),$$

$$S_2^\circ = (n_i@^\top, n_i'^{open} m_{\ell_2}).$$

4 The extra symbol $\top$ is used to model the ambient enclosing $\oplus$ and is mentioned for uniformity.

5 Rounded arrows represent the partial topology, pointing from an object to the link representing the relevant pair son/father. As usual we have omitted labels.
The capability open \( m \) in state \( S_1 \) is represented by open \( m_{\ell_1} \), and the two copies of open \( m \) in state \( S_2 \) are represented by open \( m_{\ell_2} \). The label \( \ell_1 \) has multiplicity one, and shows that there is one occurrence of the corresponding object; the label \( \ell_2 \) has multiplicity \( \omega \), and shows that there are many occurrences of the corresponding object equivalent up to re-indexing.

By abstracting the set of states \( \{S_1, S_2\} \) we obtain the following abstract state

\[
S^0 = (n_{\ell_2}^{\top}, n_{\ell_2}^{\top} \text{open} m_{\ell_2}).
\]

In the abstract state \( S^0 \) both labels have multiplicity \( \omega \) showing that there are many occurrences of ambient \( n \) and of capability open \( m \). The abstract state \( S^0 \) is obtained by taking the least upper bound of \( S_1^0 \) and \( S_2^0 \) with respect to a particular ordering over abstract states which realises the union and modifies the multiplicity of objects accordingly.

The abstraction of indexes explained above is needed to achieve a computable analysis, in that we may have infinite processes equivalent up to re-indexing (see Example 4.12).

**Abstract domain.** Let \( L^\circ = \{\ell_1, \ell_\omega \mid \ell \in L\} \) (ranged over by \( \lambda^\circ, \mu^\circ, \rho^\circ, \ldots \)) be the set of abstract labels, and let \( N^\circ \cup \hat{N}^\circ \) (ranged over by \( n^\circ, m^\circ, k^\circ, h^\circ, \ldots \)) be the set of abstract names. Let \( A^\circ \) (ranged over by \( a^\circ, b^\circ, c^\circ, \ldots \)) be the set of abstract labeled names \( n^\circ \lambda^\circ \), augmented with the symbols \( \top \) and \( \otimes \). The relation between names and labels is modified accordingly. We define \( H_L^\circ : L^\circ \to \hat{N}^\circ \) such that \( H_L^\circ(\ell_1) = H_L^\circ(\ell_\omega) = H_L(\ell) \).

The abstract labeled processes are built according to the syntax of Definition 4.1 over names \( N^\circ \cup \hat{N}^\circ \) and labels \( L^\circ \). We assume that all the previously defined notions on processes are adapted to abstract processes in the expected way. As in the concrete case we consider only well-labeled processes.

**Definition 5.1 (Well-labeled).** An abstract process \( P^\circ \) is well-labeled if: (i) \( \ell_1 \in A(P^\circ) \) implies \( \ell_\omega \notin A(P^\circ) \); (ii) for any label \( \lambda \in L^\circ \), such that \( \lambda = \ell_1 \), there is at most one object labeled by \( \lambda \).

In the following we use \( P^\circ \) and \( A P^\circ \) to denote the set of well-labeled abstract processes (referred to as abstract processes) and active well-labeled abstract processes, respectively.

**Definition 5.2 (Abstract states).** An abstract state \( S^\circ \) is a pair \( (T^\circ, C^\circ) \), where

1. \( T^\circ \in \wp((A^\circ \setminus \{\top\}) \times (A^\circ \setminus \{\top\}) \times A^\circ); \)
2. \( C^\circ \in \wp(((A^\circ \setminus \{\top\}) \times A^\circ) \times A P^\circ). \)

In an abstract state \( S^\circ = (T^\circ, C^\circ) \) we call \( T^\circ \) the topology and \( C^\circ \) the configuration. The meaning of \( (a^\circ, b^\circ, c^\circ) \in T^\circ \) (for short \( a^\circ b^\circ c^\circ \)) is that ambient \( a^\circ \) is a son of ambient \( b^\circ \), when \( b^\circ \) is within \( c^\circ \). The meaning of \( ((a^\circ, b^\circ), P^\circ) \in C^\circ \) (for short \( a^\circ b^\circ P^\circ \)) is that \( P^\circ \) is executable inside ambient \( a^\circ \), when \( a^\circ \) is within \( b^\circ \).

We assume that all the previously defined notions on states are adapted to abstract states in the expected way. As in the concrete case we consider only well-labeled states. An abstract state \( S^\circ = (C^\circ, T^\circ) \) is well-labeled if conditions (i) and (ii) of Definition 5.1 hold (with \( P^\circ \) replaced by \( S^\circ \)). We use \( S^\circ \) to denote the set of well-labeled abstract states (referred to as abstract states).
We now introduce a proper ordering over abstract states.\footnote{As usual we assume that $\subseteq$ and $\cup$ are defined component-wise.}

**Definition 5.3.** We define $\subseteq$ as the minimal ordering over $S^\circ$, such that $S^\circ \subseteq S'^\circ$ implies $S^\circ \subseteq S'^\circ$, and such that $S^\circ \subseteq S^\circ[\ell_\omega/\ell_1]$. We use $\cup^\circ$ for the least upper bound with respect to $\subseteq^\circ$.

The ordering reflects the intuition that $\ell_1$ is more precise than $\ell_\omega$. For instance, assuming that $\lambda = \ell_1$ and $\gamma = \ell_\omega$, we have

\[
(\alpha^\circ((T, \gamma), b^\circ P^\circ), n^{\gamma b^\circ P^\circ}) \cup^\circ (n^{\gamma b^\circ P^\circ}, \eta^\circ) = (n^{\gamma b^\circ P^\circ}, n^{\gamma b^\circ P^\circ}).
\]

**Definition 5.4.** The abstract domain is $\langle S^\circ, \subseteq^\circ \rangle$.

To simplify the presentation in the following we may omit the over-script $-^\circ$ for any syntactic category, when the meaning is clear from the context.

**The Galois connection.** We present now the relation between the concrete and the abstract domain establishing a Galois connection (see Definition 2.1). We formalise the ideas explained at the beginning of the section. A single state is abstracted

1. by introducing the partial topology both in the topology and in the configuration;
2. by replacing the indexed labels $L_I$ with the abstract labels $L^\circ$, and by substituting the names $\hat{N}_I$ with the abstract names $\hat{N}$.

To remove the indexes according to (2), we introduce a special renaming, that depend on the state which is abstracted, and a special substitution. Let $S \in S$ be a state. We define a renaming $\rho^\circ_S : L_I \to L^\circ$ such that $\rho^\circ_S(\ell_i) = \ell_\omega$, if there exist $j$ with $i \neq j$ such that $\ell_i, \ell_j \in A(S)$, and $\rho^\circ_S(\ell_i) = \ell_1$ otherwise. Moreover, we define a substitution $\eta^\circ : \hat{N}_I \to \hat{N}$ such that $\eta^\circ(\hat{n}_i) = \hat{n}$.

A set of states is abstracted by taking the least upper bound with respect to $\subseteq^\circ$ of the abstraction of each element.

**Definition 5.5.** Let $S^2 \in S^2$, $(T, \gamma) \in S$ and $S^\circ \in S^\circ$. We define $\alpha^\circ : S^2 \to S^\circ$ and $\gamma^\circ : S^\circ \to S^2$ as follows:

1. $\alpha^\circ((T, C)) = (T^\circ, C^\circ)\rho^\circ_{(T, C)}\eta^\circ$, where\footnote{We are assuming that the symbols $\oplus$ and $\top$ are introduced, when needed, to give a father and grandfather to any ambient. In particular, using $\top$ for the father of $\oplus$, and $\oplus$ for the father of the root, when different from $\oplus$.}

   \[
   T^\circ = \{a^b | a^b, b^b \in T\},
   \]

   \[
   C^\circ = \{a^b P | a^b \in T, a^b P \in C\};
   \]

2. $\alpha^\circ(S^2) = \bigcup_{S \in S} \alpha^\circ(|S|)$, where $\alpha^\circ(|S|) = \bigcup_{S \in S} |\alpha^\circ(S)|$;

3. $\gamma^\circ(S^\circ) = \bigcup_{S \in S^\circ} |\alpha^\circ(|S|)| \subseteq S^\circ$.

Note that in the definition above (case (1)) we have introduced an auxiliary abstraction function $\alpha^\circ : S \to S^\circ$ which maps a state into an abstract state. This is used to define the abstraction function $\alpha^\circ : S^2 \to S^\circ$ which maps a set of states up to re-indexing into an abstract state (case (2)).
The pair of functions defined above is a Galois connection.

**Theorem 5.6.** The pair of functions \((\alpha^\circ, \gamma^\circ)\) is a Galois connection between \((S^\circ, \subseteq)\) and \((S^\circ, \subseteq^\circ)\).

The proof of Theorem 5.6 is shown in Appendix B.1.

**Abstract semantics.** The abstract semantics is defined by an abstract normalisation function and by abstract transitions, which adapt the normalisation function of Table 4 and the transitions of Table 5 to the abstract domain.

The abstract normalisation function \(\delta^\circ : (A^\circ \times A^\circ) \times \mathcal{P}^\circ \rightarrow S^\circ\) is defined in Table 6 (as usual \(\delta^\circ a^b P\) stands for \(\delta^\circ((a, b), P)\)). The main differences with respect to \(\delta\) are that \(\delta^\circ\) deals with the partial topology and with the multiplicity. For instance, rule \(D\text{Amb}^\circ\) adds \(e^b\) to the topology instead of \(e\). Similarly, rule \(D\text{Pref}^\circ\) adds \(a^b M_\lambda^\circ. P\) instead of \(a M_\lambda^\circ. P\). Also the rules use \(\cup^\circ\) in place of \(\cup\) to properly handle labels with multiplicity.

The transition rules are shown in Table 7. For notational convenience we use the following abbreviations. We write \((T, C)\{\!\!\! ad/bc \!\!\!\}\) to denote the abstract state \((T\{\!\!\! ead/ebc \!\!\!\}, C\{\!\!\! ad/ bc \!\!\!\})\) for any ambient \(e\) and process \(Q\).

The rules are rather complex, it is worth explaining the most interesting cases to point out especially the role of the partial topology and of the multiplicity. Notice that, in each rule, the abstract normalisation function \(\delta^\circ\) is used in place of \(\delta\) to handle the continuations.

**Bang**\(^\circ\) The rule unfolds replication by creating a copy of the process, where every label has multiplicity \(\omega\), instead of creating a fresh copy (equivalent up to re-indexing). We use \(\text{new}_\omega\), which is defined as \(\text{new}_\omega(P) = P_{\rho}\) for the renaming \(\rho\), where \(\rho(\ell_1) = \ell_\omega\) for any \(\ell_1 \in A(P)\).

**In**\(^\circ\) The rule is applicable whenever there exists an ambient named \(m\), which is contained in the father \(b\) of \(a\), when in both cases \(b\) is within \(c\). The multiplicity of ambient \(m\) influences the movement, meaning that \(m\) can move inside itself only if its multiplicity is \(\omega\) (see the side-condition of the rule \((a \equiv m_1 \Rightarrow \ell_1' \neq \mu)\)).

The movement is realised by a modification both of the topology and of the configuration: (i) \(a^b\) is added to the topology; (ii) the continuation \(P\) and the processes, which are active inside \(a\) in parallel with \(\text{In} m P\), are added to the set of processes active, when \(a\) is within \(m\) (similarly for the ambients contained inside \(a\)); (iii) the process \(\text{In} m P\) is added to the set of

<table>
<thead>
<tr>
<th>Table 6</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>The normalisation function (\delta^\circ)</strong></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
</tbody>
</table>
Abstract transitions $\mapsto^\circ$

<table>
<thead>
<tr>
<th>Process</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bang $\diamond$</td>
<td>$P \in C$ $(T, C) \mapsto^\circ \delta^\circ \Downarrow_{\text{new}(P)} (T, C)$</td>
</tr>
<tr>
<td>In $\diamond$</td>
<td>$t = a \in_m P \in C \quad a^c \cdot m^i \in T \quad (a = m_{\ell_1} \Rightarrow \ell_1 \neq \mu)$ $(T, C) \mapsto^\circ \delta^\circ \Downarrow_{\text{in}}^\circ (T, C) \cup^\circ (T \cup^\circ {a^m}, C \setminus^\circ {t}) \downarrow_{a^m / a^b}$</td>
</tr>
<tr>
<td>Out $\diamond$</td>
<td>$t = a \text{ out}<em>m P \in C \quad a^i \cdot m^i \in T \quad (a = m</em>{\ell_1} \Rightarrow \ell_1 \neq \mu)$ $(T, C) \mapsto^\circ \delta^\circ \Downarrow_{\text{out}}^\circ (T, C) \cup^\circ (T \cup^\circ {a^c}, C \setminus^\circ {t}) \downarrow_{a^c / a^m}$</td>
</tr>
<tr>
<td>Open $\diamond$</td>
<td>$a \text{ open}<em>m P \in C \quad m^i \cdot a^i \in T \quad (a = m</em>{\ell_1} \Rightarrow \ell_1 \neq \mu)$ $(T, C) \mapsto^\circ \delta^\circ \Downarrow_{\text{open}}^\circ (T, C) \cup^\circ (T \cup^\circ {a^c}, C \setminus^\circ {t}) \downarrow_{a^c / a^m}$</td>
</tr>
</tbody>
</table>

The abstract semantics is defined as follows.

**Definition 5.7 (The abstract semantics).** Let $S_1 \in S^\circ$, $S_2 \in S^\circ$, and let $P$ be a well-labeled process. We define $\Xi_{Coll}^\circ[|P|] = \text{lfp } F^\circ (\alpha^\circ (\delta^\circ P))$, for the function $F^\circ : S^\circ \rightarrow (S^\circ \rightarrow S^\circ)$ such that $F^\circ(S_2^\circ) = \Psi_2^\circ$ and

$$\Psi_2^\circ(S_1) = S_2^\circ \cup^\circ \bigcup_{S^\circ \in [S_1, S_2] \mapsto^\circ S_2^\circ} S^\circ.$$

The abstract semantics is a safe approximation of the collecting semantics. Safeness is stated in classical abstract interpretation style showing that the abstract semantics is an upper approximation of the property we are interested in.

**Lemma 5.8.** Let $S_2 \in S$ and $S^\circ \in S^\circ$. We have

$$\alpha^\circ (\Psi_2^\circ(S_2)) \subseteq^\circ \Psi_\alpha^\circ(S_2) (\alpha^\circ (S^\circ)).$$

The proof of Lemma 5.8 is shown in Appendix B.1. The proof exploits two main properties which show the safeness of: the abstract normalisation function $\delta^\circ$ (Proposition B.7) with respect to $\delta$; the abstract transitions $\mapsto^\circ$ with respect to the concrete transitions $\mapsto$ (Lemma B.8).

**Theorem 5.9 (Safeness).** Let $P$ be a well-labeled process. We have

$$\alpha^\circ (\Xi_{Coll}[|P|]) \subseteq^\circ \Xi_{Coll^\circ}[|P|].$$
Proof. By Definitions 5.7 and 4.9 we have to show that
\[ \alpha^\omega(lfp_\delta \Psi_{\delta \circ p}) \subseteq lfp_\omega \Psi^\omega_{\alpha \circ (\delta \circ p)}. \]
This follows from Lemma 5.8 using Theorem 2.2. □

Examples. We present some examples to summarise the most interesting aspects of the abstraction. The following example explains more in details the role of indexes in the abstraction. Any labeling of a process \( P \) respecting the requirements of Definition 4.2 is enough to have a correct normal semantics of \( P \). However, the choice of labels has dramatic consequences on the precision of the abstraction. Hence, a convenient annotation schema consists of keeping all labels distinct also up to re-indexing.

Example 5.10. Consider the processes

\[ P_1 = n_{\ell_1}[in_{k_\mu}] \mid n_{\ell_2}[in_{m_\gamma}] \mid m_\lambda[0], \]
\[ P_2 = n_\beta[\mu] \mid n_\delta[\epsilon], \]
where \( \{\mu, \gamma, \lambda, \beta, \epsilon\} \) are distinct also up to re-indexing and are not of the form \( \ell_i \). We have

\[ \mathcal{S}_{Coll}[P_1] = (\{n_{\ell_1}[\mu], n_{\ell_2}[\gamma], n_\lambda[\gamma]\}, \{n_{\ell_1}[\mu], n_{\ell_2}[\gamma]\}) \]
\[ \mathcal{S}_{Coll}[P_2] = (\{n_\beta[\mu], n_\delta[\epsilon], n_\lambda[\gamma]\}, \{n_\beta[\mu], n_\delta[\epsilon]\}). \]

Obviously the processes \( P_1 \) and \( P_2 \) are equivalent up to renaming of labels. Notice that only ambients \( n_{\ell_1} \) and \( n_\delta \) may end up inside \( m_\lambda \). In the abstract semantics we have (for readability we use \( \{\mu, \gamma, \lambda, \beta, \epsilon\} \) for the corresponding labels with multiplicity one)

\[ \mathcal{S}_{Coll'}[P_1] = (\{n_{\ell_1}[\mu], n_{\ell_2}[\gamma], n_\lambda[\gamma]\}, \{n_{\ell_1}[\mu], n_{\ell_2}[\gamma]\}), \]
\[ \mathcal{S}_{Coll'}[P_2] = (\{n_\beta[\mu], n_\delta[\epsilon], n_\lambda[\gamma]\}, \{n_\beta[\mu], n_\delta[\epsilon]\}). \]

Due to a different choice of labels the results reported by the analysis are different: for process \( P_1 \) the two ambients with name \( n \) are both represented by \( n_{\ell_1} \); while for process \( P_2 \) ambients \( n_\beta \) and \( n_\delta \) are keep distinct. Consequently, the analysis of \( P_1 \) is less precise; it says that both \( n_{\ell_1} \) and \( n_{\ell_2} \) may end up inside \( m \).

The following example shows the analysis of the process considered in Example 4.11, where an ambient \( n \) moves inside an ambient \( k \), and then is opened unleashing no capability of movement inside \( k \). Due to the combination of the multiplicity and of the partial topology, the analysis is sufficiently precise to capture what is executed inside \( n \) before and after \( n \) is opened. In particular, it argues that the capability of movement \( in_k \) has been consumed when \( n \) is opened. Consequently, it says that ambient \( k \) acquires, when opens the mobile ambient \( n \), only an immobile process.

Example 5.11. Consider the process shown in Example 4.11 (see the semantics in Fig. 2)

\[ P = n_\lambda[in_{k_\mu}.m_\xi[Q_2]] \mid k_\mu[open n_\beta. Q_1]. \]
We discuss the abstract semantics of the process $P$ assuming that $Q_1 = Q_2 = 0$ and that the indexed labels $\lambda, \epsilon, \zeta, \mu, \beta$ are distinct also up to re-indexing. The initial abstract state representing the process $P$ is $S_0^\diamond = (T_0^\diamond, C_0^\diamond)$, where

$$T_0^\diamond = \{ n_\lambda @^\top, k_\mu @^\top \},$$

$$C_0^\diamond = \{ n_\lambda \in k_\epsilon, m_\zeta [0], k_\mu \open n_\beta \}.$$

By applying rule $\text{In}^\diamond$ we have a transition $S_0^\diamond \rightarrow^\diamond S_1^\diamond$, where $S_1^\diamond = (T_1^\diamond, C_0^\diamond)$ and

$$T_1^\diamond = T_0^\diamond \cup \{ m_\epsilon n_\lambda k_\mu, n_\lambda k_\mu \}. $$

The capability $\in k$ is exercised inside $n$, when $n$ and $k$ are within $\@$. Its execution modifies the abstract topology: (i) $n_\lambda k_\mu @^\omega$ is added to model the movement of $n$ inside $k$; (ii) $m_\epsilon n_\lambda k_\mu$ is added because the continuation of $\in k$ ($m_\epsilon [0]$) becomes executable after $n$ has moved inside $k$. Notice that the capability $\in k$ has multiplicity one, and thus $n_\lambda k_\mu \in k_\epsilon, m_\zeta [0]$ does not belong to the abstract configuration. This says that $\in k$ has been consumed when $n$ is within $k$.

We observe that only rule $\text{In}^\diamond$ can be applied in state $S_0^\diamond$; the capability $\open n$ cannot be exercised since $n$ is not within $k$ ($n_\lambda k_\mu @^\omega \notin T_0^\diamond$). Rule $\text{Open}^\diamond$ becomes instead executable in state $S_1^\diamond$ where $k$ is one of the fathers of $n$.

By applying rule $\text{Open}^\diamond$ we have a transition $S_1^\diamond \rightarrow^\diamond S_2^\diamond$, where $S_2^\diamond = (T_2^\diamond, C_0^\diamond)$ and

$$T_2^\diamond = T_1^\diamond \cup \{ m_\epsilon k_\mu @^\omega \}. $$

The execution of $\open n$ inside $k$ produces the unleashing inside $k$ only of those processes and ambients which are contained inside $n$, when $n$ is within $k$. Those processes and ambients are determined using the partial topology. Since $m_\epsilon n_\lambda k_\mu \in T_1^\diamond$, then ambient $m$ ends up inside $k$, that is $m_\epsilon k_\mu @^\omega$ is added to the abstract topology. No other ambient or process is acquired by $k$, in particular the process $\in k_\epsilon, m_\zeta [0]$, which can be executed inside $n$ only when $n$ is inside $\@$.

Therefore, the abstract semantics is (depicted also in Fig. 5) $S_{\text{Coll}}^\diamond [[P]] = S_2^\diamond$. The analysis shows that: $k$ is an immobile ambient (there are no capabilities of movement inside $k$); $n$ is a mobile ambient

---

Footnote: As usual we have omitted labels to simplify the picture.
(the capability \(\hat{i}n k\) is exercised inside \(n\)); ambient \(n\) unleashes, when opened, an immobile process (that is \(m_{\xi}[0]\)). As we have explained above both the labels with multiplicity and the partial topology are needed to achieve this very accurate prediction.

The following example shows the analysis of the processes discussed in Example 4.12 and clarifies how the replicated processes are identified by the abstraction.

**Example 5.12.** Consider the process \(Q = n_\lambda[\hat{i}n n_\gamma]\) of Example 4.12, where \(\lambda = \ell_1\) and \(\gamma = \ell'_1\).

We have for \(\lambda \circ = \ell_\omega\) and \(\gamma \circ = \ell'_\omega\),

\[
\Sigma_{Coll}[\bullet Q] = (n_\lambda \in n_{\hat{y}}, n_\lambda \in n_{\gamma}),
\]

\[
\Sigma_{Coll}[[!Q]] = \{(n_\lambda \in n_{\hat{y}}, n_\lambda \in n_{\gamma}, \ell_\omega), (n_\lambda \in n_{\hat{y}}, n_{\lambda'} \in n_{\gamma}, \ell'_\omega)\},
\]

\[
\Sigma_{Coll}[[!\nu n_{\mu} Q]] = \{(n_\lambda \in n_{\hat{y}}, n_{\lambda'} \in n_{\gamma}, \ell_\omega), (n_{\lambda'} \in n_{\gamma}, \ell'_\omega)\},
\]

The labels with multiplicity permit to distinguish process \(!Q\) from process \(Q\). In the abstract semantics of \(Q\) the label of \(n\) is \(\ell_1\), which forbids the movement of \(n\) inside itself (see rule \(In^\circ\)). Conversely, in the abstract semantics of \(!Q\) the unfolding of recursion produces a label \(\ell_\omega\) for \(n\) and a label \(\ell'_\omega\) for \(\hat{i}n n\), which force this movement (see rule \(In^\circ\)). Consequently, we have both \(n_\lambda, n_{\lambda'}\) in the abstract topology. Recall that the unfolding of replication produces multiple copies of \(n\), which may interact with each other as we have shown in Fig. 3. In particular, any copy of \(n\) may enter within another copy of \(n\) which is top level (inside \(\hat{i}\)). This shows a subtle difference between these two statements: \(n_\lambda, n_{\lambda'}\) is necessary to have a safe approximation of the concrete semantics; instead \(n_\lambda, n_{\lambda'}\) is an approximation due to the multiplicity \(\omega\) of capability \(\hat{i}n n\).

The analysis infers the same information for both processes \(!Q\) and \(!\nu n_{\mu} Q\). In the abstract domain the distinct names \(\hat{n}_1, \hat{n}_2\ldots\), produced by the unfolding of replication, are represented by \(\hat{n}\). Thus, the ambients \(\hat{n}\) interact with each other (see rule \(In^\circ\)).

### 6. A second abstraction

On top of the previous abstraction, we define a new abstraction, aimed at computing more efficiently an approximation of a weaker property. We want to know the following information about all the states reachable from the initial state representing a process \(P\): for each ambient \(n\), which ambients and capabilities may be contained (at top level), inside \(n\). The abstraction is simply obtained from the analysis of Section 5 by dropping the multiplicity from labels and the partial topology from the topology and the configuration.

Consider for instance the states (4) and (5) shown at the beginning of Section 5

\[
S_1 = \{a \in k_{\mu}, a \in m_{\nu}\},
\]

\[
S_2 = \{b \in m_{\nu}\}.
\]
The set of states \( \{ S_1, S_2 \} \) is represented by the following abstract state which is simply their union (depicted also in Fig. 6)}

\[
S^\circ = \left( \{ a \in b, b \in @, a \in @ \}, \{ a \in m, a \in k, a \in m, m \in m \} \right).
\]

The abstract configuration says that both \( a \) is inside \( b \), \( b \) is inside \( @ \), \( a \) is inside \( k \), \( a \) is inside \( m \), \( m \) is inside \( m \). With respect to the abstraction of Section 5, shown in Fig. 4, we lose the information that the former is executable, when \( a \) is inside \( b \); while the latter is executable, when \( a \) is inside \( @ \). Similarly for the topology.

Moreover, consider the states (6) and (7) shown at the beginning of Section 5

\[
S_1 = (n_1, \{ n_1, \text{open } m_1 \}),
\]

\[
S_2 = (n_2, \{ n_2, \text{open } m_2, n_2, \text{open } m_2 \}).
\]

In the new abstraction \( S_1 \) and \( S_2 \) are represented by the same abstract state

\[
S^\circ = (n, n_1, \text{open } m_1).
\]

Therefore, we lose the ability to distinguish one occurrence from multiple occurrences of an object.

Abstract domain. The abstract labels are \( L \) and the abstract names are \( N \cup \hat{N} \). The relation between names and labels is given precisely by function \( H : L \rightarrow \hat{N} \). We use \( A \) (ranged over by \( a, b, c, \ldots \)) for the set of abstract labeled names \( n_1 \), such that \( n \in N \cup \hat{N} \) and \( \ell \in L \), augmented with the symbol \( @ \). The abstract processes are built according to the syntax of Definition 4.1 over names \( N \cup \hat{N} \) and labels \( L \). As usual we use \( P \) and \( AP \) to denote the set of abstract processes and active abstract processes.

**Definition 6.1 (Abstract states).** An abstract state \( S^\circ \) is a pair \((T^\circ, C^\circ)\), where

1. \( T^\circ \in \wp((A \setminus \{ @ \}) \times A) \);
2. \( C^\circ \in \wp(A \times AP) \).

In an abstract state \((T^\circ, C^\circ)\) we call \( T^\circ \) the topology and \( C^\circ \) the configuration. We assume that all the previously defined notions on states and processes are adapted to abstract states and processes in the expected way. We use \( S \) to denote the set of abstract states.

---

\(^9\) As usual we have omitted labels to simplify the picture.
The abstract domain is given by the abstract states ordered by inclusion.\(^{10}\)

**Definition 6.2.** The abstract domain is \langle S^\circ, \subseteq \rangle.

In the following we may omit the over-script \(-\circ\) for any syntactic category, when the meaning is clear from the context.

**The Galois connection.** The relation between the abstract domain of Definition 5.4 and the abstract domain of Definition 6.2 is established by a Galois connection (see Definition 2.1). An abstract state is abstracted, as explained at the beginning of the section, by dropping both the multiplicity from labels and the partial topology. To this purpose, we use a renaming \(\rho\circ : L\diamond \rightarrow L\), such that \(\rho\circ(\ell_1) = \rho\circ(\ell_\omega) = \ell\).

**Definition 6.3.** Let \((T\diamond, C\diamond) \in S\diamond\) and \(S^\circ \in S^\circ\). We define \(\alpha^\circ : S^\circ \rightarrow S^\circ\) and \(\gamma^\circ : S^\circ \rightarrow S^\circ\) as follows:

1. \(\alpha^\circ((T\diamond, C\diamond)) = (\{ a^b \mid a^b \in T^\circ \}, \{ a^P \mid a^P \in C^\circ \})\rho^\circ;\)
2. \(\gamma^\circ(S^\circ) = \bigcup_{S\diamond \in \{ S\diamond \mid \alpha^\circ(S\diamond) \subseteq S^\circ \}} S^\circ\).

The pair of functions defined above is a Galois connection.

**Theorem 6.4.** The pair of functions \((\alpha^\circ, \gamma^\circ)\) is a Galois connection between \langle S^\circ, \subseteq \rangle and \langle S^\circ, \subseteq \rangle.

Theorem 6.4 is shown in Appendix B.2.

**Abstract semantics.** The abstract normalisation function \(\delta^\circ : A^\circ \times P^\circ \rightarrow S^\circ\) is given by the rules of Table 4 with a minor modification. It is enough to replace the concrete labels \(L_I\) with the abstract labels \(L\), that is using the substitution function \(H^L\) in place of \(H^{L_I}\).

The abstract transitions are defined by the rules of Table 8. Rule \textbf{Bang} is used to unfold replication; it creates a copy of the replicated process without modifying the labels. The rules \textbf{In}, \textbf{Out}, \textbf{Open} realise the movements and the opening. They are similar to the corresponding rules of the abstract semantics in Table 7 in the case of multiplicity \(\omega\). The only relevant difference is that, due to the removal of the partial topology, the conditions to be checked for the execution of capabilities are weaker. For instance, rule \textbf{In} can be applied, whenever ambient \(a\) and an ambient with name \(m\) have a common father \(b\) in the topology. There is no check on the father of \(b\) to guarantee that ambients \(a\) and \(m\) are contained in \(b\) at the same time.

The abstract semantics is defined as follows.

**Definition 6.5 (The abstract semantics).** Let \(S_1^\circ, S_2^\circ \in S^\circ\), and let \(P\) be a well-labeled process. We define \(\Xi_{Coll}[[P]] = \bigcup F^\circ(\alpha^\circ(\delta^\circ(\delta^\circ\circ P)))\) for the function \(F^\circ : S^\circ \rightarrow (S^\circ \rightarrow S^\circ)\) such that \(F^\circ(S_2^\circ) = \Psi_{S_2^\circ}\) and

\[
\Psi_{S_2^\circ}(S_1^\circ) = S_2^\circ \cup \bigcup_{S^\circ \in \{ S_1^\circ[S_1^\circ] \rightarrow S_2^\circ \}} S^\circ.
\]

\(^{10}\) As usual we assume \(\subseteq\) and \(\cup\) defined component-wise.
Table 8
Abstract transitions $\mapsto^\circ$

<table>
<thead>
<tr>
<th>Transition</th>
<th>Expression</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bang</td>
<td>$\Rightarrow aP \in C \quad (T, C) \mapsto^\circ \alpha P \cup (T, C)$</td>
</tr>
<tr>
<td>In</td>
<td>$\Rightarrow a \in P \in C \quad a^b, m^b \in T \quad (T, C) \mapsto^\circ \alpha P \cup (T \cup {m^b}, C)$</td>
</tr>
<tr>
<td>Out</td>
<td>$\Rightarrow a \in P \in C \quad a^{m^b}, m^b \in T \quad (T, C) \mapsto^\circ \alpha P \cup (T \cup {m^b}, C)$</td>
</tr>
<tr>
<td>Open</td>
<td>$\Rightarrow a \in P \in C \quad m^a \in T \quad (T, C) \mapsto^\circ \alpha P \cup (T \cup {a}, C)$</td>
</tr>
</tbody>
</table>

The abstract semantics defined above is a safe approximation of the abstract semantics of Definition 5.7.

**Lemma 6.6.** Let $S^1, S^2 \in S^\circ$. We have

$$\alpha^\circ(\Psi_{S^1}(S^1)) \subseteq \Psi_{S^2}(\alpha^\circ(S^2)).$$

The proof of Lemma 6.6 is shown in Appendix B.2. As before, the proof relies on the safeness of the abstract normalisation function $\delta^\circ$ with respect to $\delta^\circ$ (Proposition B.13), and the abstract transitions $\mapsto^\circ$ with respect to the transitions $\mapsto$ (Lemma B.14).

**Theorem 6.7** (Safeness). Let $P$ be a well-labeled process. We have

$$\alpha^\circ(\Xi_{Coll}[[P]]) \subseteq \Xi_{Coll}[[P]].$$

**Proof.** By Lemma 6.6 and Theorem 2.2 similarly as in Theorem 5.9. $\square$

It is a well-known result of abstract interpretation that Galois connections are closed under composition. Therefore, an immediate consequence of Theorem 6.7 is that the new abstract semantics is a safe approximation of the collecting concrete semantics.

**Corollary 6.8.** Let $P$ be a well-labeled process. We have

$$\alpha^\circ(\alpha^\circ(\Xi_{Coll}[[P]])) \subseteq \Xi_{Coll}[[P]].$$

**Examples.** We discuss the differences between the abstraction presented in this section and the abstraction of Section 5. One relevant difference is that the second abstraction does not distinguish between one or many occurrences of an object. Consequently, the second abstraction infers the same information for the processes $Q, !Q$ and $(\nu n \mu)Q$ discussed in the Examples 5.12 and 4.12. Another loss of information is due to the removal of the partial topology. The following examples explain that, consequently, the ability to argue on the ordering of execution of capabilities is lost.

**Example 6.9.** Consider the process of Example 4.11 (see the semantics in Fig. 2)
Fig. 7. The abstract semantics of process $P$.

$$P = n_{\lambda}[\text{in} \ k \ . \ m_{\xi} \ Q_2] | k_{\mu}[\text{open} \ n_{\beta} \ Q_1].$$

Assuming that $Q_1 = Q_2 = 0$, we derive the abstract semantics (depicted also in Fig. 7) $\Xi_{Coll^C}[P] = (T^\circ, C^\circ)$ where (for readability we use $\{\lambda, \gamma, \mu, \epsilon, \zeta, \beta\}$ to denote the corresponding abstract labels without indexes)

$$T^\circ = \{ n_{\lambda}^\omega, m_{\xi}^\omega, k_{\epsilon}^\omega, n_{\lambda} k_{\mu}^\omega, m_{\mu} k_{\epsilon}^\omega, k_{\mu} k_{\mu}^\omega \},$$
$$C^\circ = \{ n_{\lambda}^\omega \text{in} m_{\mu} . \text{in} k_{\zeta}, n_{\lambda} m_{\mu}^\omega \text{in} k_{\zeta}, n_{\lambda} m_{\beta}^\omega \text{in} k_{\zeta}, @P_1, @P_2, @P_3 \}.$$

The result of the first analysis has been discussed in Example 5.11 (Fig. 5). The second analysis is substantially less precise; it is not able to capture that capability $\text{in} \ k$ has been consumed before opening. Consequently, it says that ambient $k$ acquires also $\text{in} \ k$, when opens $n$. Also, since the analysis cannot reason on how many occurrences of ambient $k$ are present, it says that ambient $k$, by exercising $\text{in} \ k$, enters inside itself (see rule $\text{In}^\circ$). Thus, $k$ is reported as mobile ambient.

**Example 6.10.** Consider the process $P = P_1 | P_2 | P_3$, where

$$P_1 = !n_{\lambda}[\text{in} \ m_{\mu} . \text{in} k_{\zeta}], \quad P_2 = !m_{\beta}[0], \quad P_3 = !k_{\gamma}[0].$$

Assume that labels $\{\lambda, \mu, \zeta, \beta, \gamma\}$ are distinct also up to re-indexing. In the first abstraction we have $\Xi_{Coll^C}[P] = (T^\circ, C^\circ)$ where (for readability we use $\{\lambda, \mu, \zeta, \beta, \gamma\}$ to denote the corresponding abstract labels annotated with $\omega$

$$T^\circ = \{ n_{\lambda}^\omega, k_{\gamma}^\omega, m_{\beta}^\omega, n_{\lambda} m_{\mu}^\omega \},$$
$$C^\circ = \{ n_{\lambda}^\omega \text{in} m_{\mu} . \text{in} k_{\zeta}, n_{\lambda} m_{\mu}^\omega \text{in} k_{\zeta}, n_{\lambda} m_{\beta}^\omega \text{in} k_{\zeta}, @P_1, @P_2, @P_3 \}.$$

The analysis shows that capability $\text{in} \ k$ is not exercised inside $n$. In fact, the partial topology says that, it is executable only when $n$ has moved inside $m$. Ambient $k$ does not move and, consequently, cannot be within $m$.

In the second abstraction we have (for readability we use $\{\lambda, \mu, \zeta, \beta, \gamma\}$ for the corresponding abstract labels without indexes)

$$\Xi_{Coll^C}[P] = ((\{ n_{\lambda}^\omega, k_{\gamma}^\omega, m_{\beta}^\omega, n_{\lambda} m_{\mu}^\omega \}), \{ n_{\lambda}^\omega \text{in} m_{\mu} . \text{in} k_{\zeta}, n_{\lambda}^\omega \text{in} k_{\zeta}, @P_1, @P_2, @P_3 \}).$$

---

11 As usual we have omitted labels to simplify the picture.
The analysis predicts that \( \text{in} k \) can be executed, because \( n \) and \( k \) have \( @ \) as a common father. Due to the removal of the partial topology, does not detect that \( \text{in} k \) becomes executable inside \( n \) only after the movement inside \( m \).

It is worth noticing that the result of the first analysis is not optimal, meaning that

\[ \alpha^2 (\Xi_{\text{ColI}}[\llbracket P \rrbracket]) \subset \Xi_{\text{ColI}}[\llbracket P \rrbracket]. \]

For instance, in the abstract semantics we have \( s_\mu^m \text{in} m_\mu \text{in} k_\xi \) which says that \( \text{in} m \) is still executable inside \( n \), when \( n \) is within \( m \). Instead, in any instance of ambient \( n \) capability \( \text{in} m \) has been obviously consumed at that time. This approximation is due to the removal of the indexes, which in this case identifies all ambients \( n \) and all capabilities \( \text{in} m \) (see rule \text{Bang}').

The abstraction presented in this section uses an abstract domain analogous to that of the CFA proposed in [25]. Our analysis is however more precise as the following example shows.

\textbf{Example 6.11.} Consider the process \( P = n_\lambda[\text{in} m_\mu \text{in} k_\xi] | k_\gamma[\text{in} m_\beta[0]]. \) We obtain (for readability we use \( \{\lambda, \mu, \xi, \beta, \gamma\} \) to denote the corresponding abstract labels without indexes)

\[ \Xi_{\text{ColI}}[\llbracket P \rrbracket] = (\{n_\lambda @, k_\gamma @, m_\beta^k \}, \{n_\lambda \text{in} m_\mu \text{in} k_\xi\}). \]

The analysis shows that the system is deadlocked: neither capability \( \text{in} m \) nor capability \( \text{in} k \) can be executed. The former because ambient \( m \) is not a sibling of \( n \), the latter because it is guarded by \( \text{in} m \).

The analysis of [25] considers the effect of the continuation of a capability regardless of whether the capability may be exercised. Consequently, for process \( P \) it predicts that \( n \) moves inside \( k \) and, consequently, also inside \( m \).

7. Applications to security

We show some examples to demonstrate that the analyses we have proposed can be used to establish interesting security properties. In particular, we show the results obtained using the abstraction of Section 5 for two simple examples found in the literature [16,5]. Another typical example is the firewall protocol, which can be proved correct also by applying the weaker analysis of Section 6. This example in fact can be checked also by the CFA of [25].

\textbf{Example 7.1 (Secrecy).} Degano et al. [16] consider a property of secrecy based on a standard classification of ambients into untrusted and trusted. Secrecy of data is preserved if an untrusted ambient can never open a trusted ambient, since opening an ambient gives indeed access to its content. They show that the property holds for the following system (actually for its SA version)

\[ SYS = (v \text{ mail}) (a[\text{mail[out } a . \text{ in b. msg[out } \text{ mail. D} ]]) | b[\text{open msg}] | C. \]

The pilot ambient \( \text{mail} \) goes out of \( a \), and then enters \( b \). Once there, \( \text{msg} \) goes out of \( \text{mail} \), and \( b \) acquires the data \( D \) by opening \( \text{msg} \). When the data \( D \) is secret, it is essential to guarantee that no ambient can open \( \text{msg} \) except for the designated receiver \( b \). Assume that \( \{b, \text{msg}\} \) is the set of trusted ambients, and that all the others (including \( @ \)) are untrusted. We wish to prove that no untrusted ambient can open \( \text{msg} \).
Assume that the parallel process $C$ is open $msg$ meaning that the untrusted ambient $@$ tries to read the data $D$. By applying the analysis of Section 5 we derive $\Sigma_{Coll} [[SYS]] = (T^\circ, S^\circ)$, where\(^{12}\)

$$T^\circ = \{ a^\top@, \text{mail}^\top@, b^\top@, \text{mail}^\topT, \text{mail}^\top\#, \text{msg}^\text{mail}b, \text{msg}^\#b \},$$

$$C^\circ = \{ a^\top\text{open msg}, b^\#\text{open msg}, \text{mail}^\text{out}a.\text{in b.msg}[\text{out mail}.D], \text{mail}^\text{in}b.\text{msg}[\text{out mail}.D], \text{msg}^\text{mail}out[D], \text{msg}^bD, b^\#D \}.$$

This result shows that only $b$ can open the messenger ambient $msg$. Consequently the secret data may end up in $b$ only, as shown by $\text{msg}^bD$ and $b^\#D$. Both the partial topology and the multiplicity are needed to achieve this result. The main observations concerning the analysis are:

- the capability $\text{open msg}$ cannot be exercised in $@$ because $msg$ cannot end up within $@$. This is reported by the abstract topology, in particular by $\text{msg}^\text{mail}b$ and $\text{msg}^b\#$;
- the execution of the capability $\text{out mail}$ inside $msg$, lets $msg$ go only inside ambient $b$, as $msg$ can be contained in $mail$, only when $mail$ is within $b$ (see rule $\text{Out}^\circ$). The latter condition is modeled by $\text{msg}^\text{mail}b$;
- the multiplicity of capabilities $\text{out a}$ and $\text{in a}$ is used to conclude that $msg$ can be contained in $mail$ only when $mail$ is within $b$ (see rules $\text{Out}^\circ$ and $\text{In}^\circ$).

The analyses of [25] and of Section 6 are too weak to prove the secrecy of this system. They predict that $msg$, when goes out of $mail$, may end up in any of the fathers of $mail$, namely $a$, $b$ and $@$. This example shows that the analysis of Section 5 gives results comparable to those obtained for SA in [16]. In SA, however, it is easier to get such an accurate prediction, because coactions control precisely when and where capabilities can be exercised.

**Example 7.2 (Security boundaries).** Braghin et al. [5] study multilevel security for mobile ambients. The original idea is that of introducing boundary ambients to protect high level information; high level data can be contained either in boundary ambients or in low level ambients which do not escape boundaries. They refine the analysis of [25] to establish more precisely the property above. In particular, they show the following motivating system

\[
SYS = a[send[\text{out a.in b} | hdata[\text{in filter}]]] \\
| b[open.send] | filter[\text{in send}] | open.filter.
\]

The boundary ambient $send$ carries the high level ambient $hdata$ out of $a$. Then, it lets a possibly low level $filter$ ambient enter, and then it enters the boundary ambient $b$. Once there, it is dissolved. The system satisfies the security property stated above: $hdata$ is always within a boundary ambient (either $send$ or $b$) or within the low level ambient $filter$. Notice that the ambient $filter$ does not carry the ambient $hdata$ out of the boundary $b$.

By applying the analysis of Section 5 we obtain $\Sigma_{Coll} [[SYS]] = (T^\circ, S^\circ)$ where\(^{13}\)

$$T^\circ = \{ a^\top@, \text{send}^\top@, \text{filter}^\top@, \text{hdata}^{senda}, b^\top@, \text{send}^\top@, \text{hdata}^{sendb}, \text{filter}^{sendb}, \text{filter}^\#b \}.$$
The analysis shows that the security property holds, as the abstract topology shows that \( hdata \) can be within \( filter \) only when \( filter \) is contained in a boundary ambient, either \( b \) or \( send \). This is modeled by \( hdata \) \( filter \) \( send \) and \( hdata \) \( filter \) \( b \).

The analysis of [25] as well as the analysis of Section 6 identify, instead, a potential (but practically impossible) attack. Since they do not use the partial topology, they cannot capture that \( hdata \) enters inside \( filter \) only when \( filter \) is within either \( send \) or \( b \). Consequently, they predict that \( hdata \) may end up inside the low level ambient \( \top \) as a consequence of the execution of \( open \) \( filter \).

8. Conclusions and related works

We have proposed an abstract interpretation framework for MA based on the normal semantics. The normal semantics uses an explicit representation of the hierarchical structure of processes, in terms of topology and configuration. This representation is more viable for abstraction than the standard reduction semantics. The normal semantics can be compared with the Gamma semantic framework for concurrency of [4]: it shares its view of symmetry and locality of interaction, and is based on an explicit representation of multisets.

In the abstract interpretation framework we have derived two safe approximations of the run-time topological structure of processes. To show that these analyses are effective program analysers, it is worth discussing their computational complexity. By restricting the attention to a process \( P \) of size \( n \), in the first case the topology of the greatest state contains at most \( O(n^3) \) elements and the configuration at most \( O(n^3) \) elements. Hence, the iterations before reaching the fixed-point are at most \( O(n^3) \). Any iteration has complexity \( O(n^5) \), because it requires to check at most \( O(n^2) \) conditions for any element of the configuration. Similarly, in the second case we have at most \( O(n^2) \) iterations, where any iteration has complexity \( O(n^3) \). Therefore, it is not difficult to devise a naive implementation of the first analysis in \( O(n^8) \) and of the second one in \( O(n^5) \) by using standard algorithms.

In the last few years there has been a growing interest in the analysis of MA (and its variants) and several CFA in Flow Logic style [5,16,20,25,26] have been proposed. The analysis of Section 6 is a refinement of the 0-CFA of [25]. The CFA of [25] is less precise, as shown by Example 6.11, and can be obtained in our framework by weakening the conditions on the execution of the continuation of a capability in the rules of Table 8. We refer the reader to [24] for the formal comparison of the two approaches.

The analysis of Section 5 combines together the information about the number of occurrences of objects and the contextual information (i.e., the partial topology). The idea of using the partial topology has been inspired by the 1-CFA of [16] for Safe Ambients. The integration of these two aspects gives accurate predictions as shown by Examples 5.11, 7.1 and 7.2. These systems are interesting because the considered properties require to have a detailed information about the local process of an ambient, when this is ready to engage into an interaction of opening or movement. We are not aware of similar results in setting of MA apart from those obtained by more complex exponential technique, which use sophisticated information about the context or a sort of causality information [1,26]. For SA instead
the static techniques are more precise due to the presence of coactions. The 1-CFA of [16] for SA, for instance, is simpler than our analysis and is sufficiently precise to prove the secrecy property for the SA process corresponding to that of Example 7.1.

It is worth mentioning that we have introduced the occurrence counting information in the analysis of Section 5 to fruitfully exploit the partial topology. This information is crucial to predict when capabilities may be consumed. The use of the partial topology without that of multiplicity would give limited benefits (see for instance Example 6.10). Other approaches have been proposed to more profitably exploit the information about the number of objects. For instance, Hansen et al. [20] show that the 0-CFA of [25] can be derived, by abstract interpretation, from a new more precise and exponential CFA. The refined CFA uses sets of abstract states rather than abstract states and a relational occurrence counting analysis, meaning that the number of occurrences is not counted globally (as in the abstraction of Section 5), but inside any ambient. The use of abstract interpretation in [20] shows several advantages: the CFA's are compared in terms of precision by construction and the properties (in particular the safeness) of the former one are directly derived from those of the latter one. This is precisely what we obtain with the abstraction of Section 6.

Although the interplay between abstract interpretation and CFA in Flow Logic style is not fully understood, these techniques are undoubtedly very similar from an algorithmic point of view and also their specifications are strictly related. For instance, the constraints, which specify the CFA of [25], could be derived by abstract interpretation in our framework; conversely it seems that constraints in Flow Logic style could be given corresponding to the analysis of Section 6. Having said that, it is clear that the approach of [20] is very close to ours. We believe, however, that this paper proposes another original and interesting contribution with respect to the proposal of [20]: the definition of a general abstract interpretation framework, based on the normal semantics. The normal semantics simplifies the development of analyses by means of abstract interpretation; for instance, the derivation of the analysis of Section 6 is rather straightforward once the abstract domain, namely the property we want to compute, has been chosen. Moreover, the derivation of analyses from the normal semantics can be done using standard abstract interpretation techniques to refine and combine domains.

By the time the full version of this paper has been completed, another paper [18] has appeared, which proposes an abstract interpretation framework based on a non-standard semantics similar to the normal semantics. The shape of states and of labels is however slightly different and permits to define an interesting non-uniform analysis where recursive instances of agents are kept distinct. Another CFA that refines the analysis of [25] has been recently proposed in [5]. This work is motivated by the system of Example 7.2 for which the property of multi-level security cannot be established using the approach of [25]. We have shown that this example can be handled also by our analysis. A formal comparison is difficult as the CFA of [5] is designed to establish specifically the property of multi-level security.

This work is part of a project aimed at studying the relationship among abstract interpretation, CFA and types. We believe that the formalisation also of types (for instance of [7,8]) in an abstract interpretation setting would be very interesting. First, this way we could formally compare the expressive power of CFA's and types, integrate them, understand the pros and cons of each approach, and possibly for which class of properties one method is more adequate than another. Moreover, the development of types as abstract interpretations of a denotational semantics has given very promising results for functional languages [15]. This approach gives in particular more accurate type inference algorithms, based on abstract fixed-point computations and widening operators, and more expressive type systems. It would be interesting to apply this approach also to MA starting for instance from the recent “logical”
denotational semantics for higher-order MA of [12]. We leave this investigation to future work. Notice
that the comparison with types requires to extend the analyses to the full language with communication.
A first step toward this extension has been done by Feret [18], which considers communication of names
only. This extension deserves undoubtedly further investigations especially for the analysis of Section 5.

Appendix

A. Proofs of Section 4

In this section we show the proof of Theorem 4.5, which formalises the relation between the normal
semantics of Section 4 and the standard reduction semantics of Section 3. For convenience we recall its
assertion:

Let \( P \) be a well-labeled process and \( a \in \mathcal{A} \) which is fresh for \( P \):

1. if \( \delta^{a}P \rightarrow S \), then there exist a well-labeled process \( Q \), such that \( \mathcal{E}(P) \rightarrow_{\equiv} \mathcal{E}(Q) \) and \( \delta^{a}Q = S \);
2. if \( \mathcal{E}(P) \rightarrow Q \), then there exist a state \( S \) and a well-labeled process \( Q' \), such that \( \delta^{a}P \rightarrow^{*} S, \delta^{a}Q' = S \) and \( Q \equiv \mathcal{E}(Q') \).

Part (1) shows soundness and the converse part (2) shows completeness. To simplify the proof, which
is rather complex, we extend the reduction semantics to well-labeled processes. The reduction semantics
for well-labeled processes is designed precisely to be closer to the normal semantics than the standard
reduction semantics. Then, we prove both soundness and completeness in two steps: (i) we show the
relation between the reductions of well-labeled processes and those of standard unlabeled processes
(Lemmas A.2 and A.9); (ii) we show the relation between the reductions of well-labeled processes and
the transitions between the states representing them (Lemmas A.17 and A.20).

In the following, to ease the use of induction in the proofs, we assume that also standard unlabeled
processes of Definition 3.1 can be defined over names \( \mathcal{N} \cup \hat{\mathcal{N}} \).

A.1. Reduction semantics of well-labeled processes

The reduction semantics for well-labeled processes is defined by the rules of Table A.1 and is the
obvious adaptation of the standard reduction semantics for the unlabeled processes (Tables 1 and 2). The
only difference is that in rule (Cong) we adopt a relation \( \gg \), which differs substantially from structural
congruence \( \equiv \) for unlabeled processes (Table 3). In particular,

1. we rule out the analogues of rules (Pref) and (Bang);

<table>
<thead>
<tr>
<th>Rule</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>( n_{\lambda}[\text{in},m_{\gamma}.,P \mid Q] \mid m_{\mu}[R] \rightarrow m_{\mu}[n_{\lambda}[P \mid Q] \mid R] )</td>
<td>(In)</td>
</tr>
<tr>
<td>( m_{\mu}[n_{\lambda}[\text{out},m_{\gamma}.,P \mid Q] \mid R] \rightarrow n_{\lambda}[P \mid Q] \mid m_{\mu}[R] )</td>
<td>(Out)</td>
</tr>
<tr>
<td>( \text{open},n_{\mu}.,P \mid n_{\lambda}[Q] \rightarrow P \mid Q )</td>
<td>(Open)</td>
</tr>
<tr>
<td>( P \rightarrow Q \Rightarrow (\nu n_{\lambda})P \rightarrow (\nu n_{\lambda})Q )</td>
<td>(Res)</td>
</tr>
<tr>
<td>( P \rightarrow Q \Rightarrow P \mid R \rightarrow Q \mid R )</td>
<td>(Par)</td>
</tr>
<tr>
<td>( P \rightarrow Q \Rightarrow n_{\lambda}[P] \rightarrow n_{\lambda}[Q] )</td>
<td>(Amb)</td>
</tr>
<tr>
<td>( (P' \rightarrow Q', P \gg P', Q' \gg Q) \Rightarrow P \rightarrow Q )</td>
<td>(Cong)</td>
</tr>
</tbody>
</table>
Table A.2
The relation $\gg$

<table>
<thead>
<tr>
<th>Rule</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>$P \gg P$</td>
<td>(Refl)</td>
</tr>
<tr>
<td>$P \gg Q$, $Q \gg R \Rightarrow P \gg R$</td>
<td>(Trans)</td>
</tr>
<tr>
<td>$P \gg Q \Rightarrow P \gg Q$</td>
<td>(Comm)</td>
</tr>
<tr>
<td>$(P \gg Q) \gg R, Q \gg R \Rightarrow P \gg R$</td>
<td>(Ass)</td>
</tr>
<tr>
<td>$P \gg Q \Rightarrow (\nu n\lambda)P \gg (\nu n\lambda)Q$</td>
<td>(Res)</td>
</tr>
<tr>
<td>$P \gg Q \Rightarrow P \gg P</td>
<td>R \gg Q</td>
</tr>
<tr>
<td>$(P \gg Q) \gg n_\lambda[P] \gg n_\lambda[Q]$</td>
<td>(Amb)</td>
</tr>
<tr>
<td>$n \not= m \Rightarrow (\nu n_\lambda)(\nu m_\mu)P \gg (\nu m_\mu)(\nu n_\lambda)P$</td>
<td>(Res-Com)</td>
</tr>
<tr>
<td>$n \not= f n(P) \Rightarrow (\nu n_\lambda)(P</td>
<td>Q) \gg P</td>
</tr>
<tr>
<td>$n \not= m \Rightarrow (\nu n_\lambda)(\nu m_\mu)P \gg (\nu m_\mu)(\nu n_\lambda)P$</td>
<td>(Res-Amb)</td>
</tr>
<tr>
<td>$P \gg Q \Rightarrow n_\lambda[P] \gg n_\lambda[Q]$</td>
<td>(Nil-Par)</td>
</tr>
<tr>
<td>$(\nu n_\lambda)\rho I$ is well-labeled; there is no $\lambda \in n_\lambda[Q]$ such that $H_{\mathcal{L}_I}(\lambda) \in n(P)$</td>
<td>(Nil-Res)</td>
</tr>
<tr>
<td>$!P \gg new(P)</td>
<td>!P$</td>
</tr>
</tbody>
</table>

(2) we assume that rule (Bang-Bang), which realises the fold/unfold of replication, can be applied only in one way, that is to produce a copy of the replicated process and not to remove it.

These choices are motivated by the aim of having a relation $\gg$ which better reflects the normal semantics. More in details, we want that two well-labeled processes, such that $P \gg Q$, are represented by “equivalent” states when translated via $\delta$ (see Lemma A.18). The rules (Pref) and (Bang) give problems as, in the normal semantics, some syntactical differences are removed only at execution time. Consider, for instance, two processes $M. P$ and $M. Q$, where $P \equiv Q$. These processes are represented by two different states (assuming a proper labeling) $\delta@M. P = (\emptyset, @M. P)$ and $\delta@M. Q = (\emptyset, @M. Q)$. The continuations $P$ and $Q$ are translated via function $\delta$ only after the execution of capability $M$.

Rule (Bang-Bang) gives a similar problem, as the unfolding of replication is modeled by a transition (Bang) in the normal semantics. Consider for instance two processes $!P$ and $!P | P$. These processes are represented by two different states (assuming a proper labeling) $\delta.@!P = S_1 = (\emptyset, @!P)$ and $\delta.@(!P | P) = S_2 = (\emptyset, @!P) \cup @P$. We have $S_1 \not\rightarrow S_2$ by rule Bang, but obviously $S_2 \not\rightarrow S_1$.

The relation $\gg$ for well-labeled processes is defined in Table A.2. As we have explained above $\gg$ is not symmetric, as there is only one way of (Bang-Bang).\(^{14}\) In rule (Bang-Bang) the labels of the replicated process are re-indexed to guarantee that new$(P) | !P$ is a well-labeled process provided that $P$ is well-labeled. To this aim, we use new$(P)$ which is adapted in the obvious way from the definition of new over states (see Section 4). Hence, we let new$(P) = P\rho_I$, where $\rho_I$ is a re-indexing of labels such that dom$(\rho_I) = \lambda(P)$; $P\rho_I$ is well-labeled; there is no $\lambda \in \lambda(P\rho_I)$ such that either $\lambda \in \lambda(P)$ or $H_{\mathcal{L}_I}(\lambda) \in n(P)$.

\(^{14}\) We have therefore removed rule (Symm) and introduced the other direction of the rules (Ass), (Res-Par), (Res-Amb), (Nil-Par) and (Nil-Res).
It is worth mentioning that $\gg$ and $\rightarrow$ are defined only over well-labeled processes. It means that rules (Res), (Par) and (Amb) of Table A.2 and the corresponding rules of Table A.1 can be applied only when the resulting processes are well-labeled. This guarantees that the well-labeling of processes is preserved, that is the labels of new($P$) are fresh. For instance, $R \mid !P \gg R \mid (\text{new}(P) \mid !P)$ can be derived by applying rule (Par) to the premise $!P \gg \text{new}(P) \mid !P$ provided that both $R \mid (\text{new}(P) \mid !P)$ and $R \mid !P$ are well-labeled.

We now show the relation between the reductions of well-labeled processes and those of standard processes. To simplify the proofs we assume that in the inference of a statement $P \equiv Q$ over unlabeled processes the symmetric rules are used directly in place of rule (Symm) (as in $\gg$).

**Soundness.** We show that any reduction between two well-labeled processes is simulated by a reduction between the corresponding unlabeled processes.

**Lemma A.1.** Let $P$ and $Q$ be well-labeled processes. If $P \gg Q$, then $\mathcal{E}(P) \equiv \mathcal{E}(Q)$.

**Proof.** It is enough to observe that for any case in Table A.2 there exists a corresponding case in Table 3. In the case of (Bang-Bang) we have $P = !P$ and $Q = !P \mid \text{new}(P)$. As $\mathcal{E}(\text{new}(P)) = \mathcal{E}(P)$, by definition of new, we conclude $!\mathcal{E}(P) \equiv !\mathcal{E}(P) \mid \mathcal{E}(\text{new}(P))$. □

**Lemma A.2.** Let $P$ be a well-labeled process. If $P \rightarrow Q$, then $\mathcal{E}(P) \rightarrow \mathcal{E}(Q)$.

**Proof.** The proof is straightforward using Lemma A.1 for the case (Cong). □

**Completeness.** The proof of completeness is more complex. Due to the difference between $\equiv$ and $\gg$, the converse of Lemmas A.1 and A.2 do not hold. Consider for instance the following unlabeled processes:

$$P = !R \mid R \mid m[0]\mid n[\in m. S],$$
$$Q = !R \mid n[\in m. S'] \mid m[0].$$  \hspace{1cm} (A.1)  
\hspace{1cm} (A.2)

We have a reduction $Q \rightarrow Q'$, where $Q' = !R \mid m[n[S'] \mid 0]$. Assuming that $S \equiv S'$ we have $P \equiv Q$, and therefore by rule (Cong) we have also $P \rightarrow Q'$. We observe that there are no well-labeled versions of $P$ and $Q$ such that $P_L \gg Q_L$ (where $\mathcal{E}(P_L) = P$ and $\mathcal{E}(Q_L) = Q$). The problem is that in $P \equiv Q$ we use: rule (Pref) to derive $\in m. S \equiv \in m. S'$, and rule (Bang-Bang) to derive $!R \mid R \equiv !R$. Both steps cannot be simulated by $\gg$ over labeled processes (see Table A.2).

We therefore show a weaker property (Lemma A.9): if $P \rightarrow Q'$ then there exist well-labeled processes $P_L$ and $Q'_L$, such that $P_L \rightarrow Q'_L$, $\mathcal{E}(P_L) = P$ and $\mathcal{E}(Q'_L) \equiv Q'$. The proof of this property is based on the following steps. We show that, when $P \equiv Q$ and $Q \rightarrow Q'$, there exists a special process $Q''$, such that:

1. $P \gg Q''$ and $Q'' \equiv Q$, where $\gg$ means that only the rules of Table 3 corresponding to those of Table A.2 have been used;
2. the derivation $P \gg Q''$ can be simulated in the labeled setting (meaning that there exist well-labeled processes $P_L$ and $Q''_L$, such that $P_L \gg Q''_L$, $\mathcal{E}(P_L) = P$ and $\mathcal{E}(Q''_L) \equiv Q''$);
3. due to the special form of $Q''$, $Q''_L$ can simulate the transition $Q \rightarrow Q'$ (meaning that $Q'_L \rightarrow Q'_L$, where $\mathcal{E}(Q'_L) \equiv Q'$).
For instance for the processes (A.1) and (A.2) illustrated above we can take

\[ Q'' = !R \mid R \mid n[i \in \sigma].S \mid m[0]. \quad (A.3) \]

We have \( P \gg Q'' \) by rules (Comm) and (Par) and we have a transition \( Q'' \to Q''', \) where \( Q''' = !R \mid R \mid m[n[S]] \mid 0. \) Moreover, since \( Q' = !R \mid m[n[S']] \mid 0 \) and \( S \equiv S' \) we have \( Q''' \equiv Q' \) by rules (Bang-Bang). (Pref) and (Par). It is immediate to check that both \( P \gg Q'' \) and \( Q'' \to Q''' \) can be simulated in the labeled setting.

To find out in a systematic way the process which satisfies the properties described above we introduce the following definition.

Let \( P \) and \( Q \) be processes. We say that a process \( Q \) is a normal form of a process \( P \) iff

1. \( Q = P = 0; \)
2. \( Q = M. Q' \) and \( P = M. P' \), where \( P' \equiv Q' ; \)
3. \( Q = Q_1 \mid Q_2 \) and \( P = P_1 \mid P_2 \), where \( Q_i \) is a normal form of \( P_i \), for any \( i \in \{1, 2\}; \)
4. \( Q = n[Q'] \) and \( P = n[P'] \), where \( Q' \) is a normal form of \( P' \);
5. \( Q = \langle v \rangle. Q' \) and \( P = \langle v \rangle. P' \), where \( Q' \) is a normal form of \( P' \);
6. \( Q = !Q' \) or \( \langle \rangle. Q \mid \langle i \in \{1, \ldots, n\} \rangle Q' \rangle \) and \( P = !P' \), where \( Q' \equiv P' \) and \( Q' = Q'_i \equiv P'_i \), for any \( i \in \{1, \ldots, n\}. \)

For instance the process (A.3) is a normal form of the process (A.2).

We give below some easy properties about the normal form.

**Proposition A.3.** Let \( P, Q \) be processes such that \( Q \) is a normal form of \( P \). We have \( P \equiv Q. \)

**Proof.** The proof proceeds by induction on the structure of \( P \) using the definition of normal form and the rules of Table 3. The most interesting case is when \( P = !P' \) and either \( Q = !Q' \) or \( Q = !Q' \mid \langle i \in \{1, \ldots, n\} \rangle Q'_i \rangle \), where \( Q' \equiv P' \) and \( Q'_i \equiv P'_i \) for any \( i \). In the former case \( P \equiv Q \) follows immediately by rule (Bang). In the latter case we have \( !P' \equiv !Q' \), by rule (Bang). Also, by rules (Bang-Bang), (Par) and (Trans) \( !Q' \equiv !Q' \mid \langle i \in \{1, \ldots, n\} \rangle Q'_i \rangle \), thus, by rule (Trans) we have \( P \equiv Q. \)

**Proposition A.4.** Let \( P_1, P_2, \) and \( P_3 \) be processes. If \( P_1 \) is a normal form of \( P_2 \) and \( P_2 \) is a normal form of \( P_3 \), then \( P_1 \) is a normal form of \( P_3. \)

**Proof.** All the cases are easy using the definition of normal form except from the case when \( P_3 = !Q \). By definition we have either \( P_2 != P \) or \( P_2 = !P \mid \langle i \in \{1, \ldots, n\} \rangle Q_i \rangle \), where \( Q \equiv P = Q_i \), for any \( i \). In the former case, we have \( P_1 = !R \) or \( P_1 = !R \mid \langle i \in \{1, \ldots, k\} \rangle R_i \rangle \), where \( R \equiv R = R_i \), for any \( i \). Since \( Q \equiv P \) we have also \( Q = \langle R \rangle \mid \langle i \in \{1, \ldots, k\} \rangle R_i \rangle \), for any \( i \). Consequently, \( P_1 \) is a normal form of \( P_3 \). In the latter case, we have \( P_1 = P_1.1 \mid P_2.1 \), where \( P_1.1 \) is a normal form of \( !P \) and \( P_2.1 \) is a normal form of \( \langle i \in \{1, \ldots, n\} \rangle Q_i \rangle \). It means that \( P_1.1 = !R \) or \( P_1.1 = !R \mid \langle i \in \{1, \ldots, k\} \rangle R_i \rangle \), where \( R \equiv R = R_i \), for any \( i \). Moreover, \( P_2.1 = \langle i \in \{1, \ldots, n\} \rangle S_l \), where \( S_l \) is a normal form of \( Q_l \). By Proposition A.3 we have \( S_l \equiv Q_l \) for any \( i \). Assume that \( P_1 = !R \mid \langle i \in \{1, \ldots, k\} \rangle R_i \rangle \), since \( Q \equiv P = Q_i \), \( S_l \equiv Q_l \), and \( P = R, P_1 \) is a normal form of \( P_3 \). Assume that \( P_1 = !R \mid \langle i \in \{1, \ldots, k\} \rangle R_i \rangle \mid \langle i \in \{1, \ldots, n\} \rangle S_l \) since \( Q \equiv P = R = R_i \) and \( S_l \equiv Q_l \equiv P \), then \( P_1 \) is a normal form of \( P_3 \).}

We show the main property of normal forms we have discussed above: if \( P \equiv Q \) then there exists a normal form \( Q' \) of \( Q \), such that \( P \gg Q' \) and \( Q' \equiv Q. \)
Lemma A.5. Let \( P, Q \) be processes such that \( P \equiv Q \). There exists a process \( Q' \), which is a normal form of \( Q \), such that \( P \gg Q' \) and \( Q' \equiv Q \).

Proof. We notice that, by Proposition A.3, when \( Q' \) is a normal form of \( Q \), we have also \( Q' \equiv Q \). Therefore, it is enough to find out a process \( Q' \), which is a normal form of \( Q \). The proof proceeds by induction on the depth of the inference of \( P \equiv Q \).

- The cases of (Refl), (Comm), (Ass), (Res-Com), (Res-Par), (Res-Amb), (Nil-Par) and (Nil-Res) are easy. They can be solved by taking \( Q' = Q \), as \( P \gg Q' \) follows from \( P \equiv Q \) (by applying the same rule of Table A.2).

- The cases of (Res), (Par) and (Amb) are similar and follow by applying the induction hypothesis; as an example we show (Amb). It means that \( P \equiv n[R'] \) and \( Q \equiv n[R] \), where \( R' \equiv R \). By induction hypothesis there exists \( R'' \), such that \( R'' \equiv R \) and \( R' \gg R'' \). We take \( Q' = n[R'']. \) We have \( P \gg Q' \) by applying rule (Amb) to the premise \( R' \gg R''. \) Moreover, since \( R'' \) is a normal form of \( R \), then \( Q' \) is a normal form of \( Q \).

- In case (Bang) we have \( P = !R' \) and \( Q = !R \), where \( R' \equiv R \). Taking \( Q' = P \) we immediately have \( P \gg Q' \) by rule (Refl). Moreover, since \( R' \equiv R \) (and conversely \( R \equiv R' \)) then \( Q' \) is a normal form of \( Q \).

- In case (Pref) we have \( P = M, R' \) and \( Q = M, R \), where \( R' \equiv R \). Taking \( Q' = P \) we have \( P \gg Q' \) by rule (Refl). Since \( R' \equiv R \), then \( Q' \) is a normal form of \( Q \).

- In case (Bang-Bang) there are two possibilities depending on the way the rule is applied. Therefore, either \( P = !R \mid R \) and \( Q = !R \) or \( P = !R \) and \( Q = !R \mid R \). In the latter case we take \( Q' = Q \) and we have \( P \gg Q' \) by rule (Bang-Bang). In the former case we take \( Q' = P \) and we have \( P \gg Q' \) by rule (Refl). Moreover, \( Q' \) is a normal form of \( Q \) using \( R \equiv R \).

- In case (Trans) we have \( P \equiv Q_1 \) and \( Q_1 \equiv Q \). By induction hypothesis there exist \( R_1, R_2 \) such that:
  
  (i) \( P \gg R_1 \) and \( R_1 \) is a normal form of \( Q_1 \); (ii) \( Q_1 \gg R_2 \) and \( R_2 \) is a normal form of \( Q \).

  This case is rather complex. The crux of the proof consists of showing that, since \( R_1 \) is a normal form of \( Q_1 \) (and thus by Proposition A.3 \( R_1 \equiv Q_1 \)) and \( Q_1 \gg R_2 \), then there exists a process \( Q' \), which is a normal form of \( R_2 \), such that \( R_1 \gg Q' \) (and by Proposition A.3 \( Q' \equiv R_2 \)). To prove this property we proceed by induction on the depth of the inference of \( Q_1 \gg R_2 \). The case (Refl) is obvious; we show the other cases below.

  - The cases of (Comm) and (Ass) are similar; as an example we show (Comm). We have \( Q_1 = S_1 \mid S_2 \) and \( R_2 = S_2 \mid S_1 \). Since \( R_1 \) is a normal form of \( Q_1 \) it means that \( R_1 = S'_1 \mid S'_2 \), where \( S'_i \) is a normal form of \( S_i \) for any \( i \in \{1, 2\} \). We take \( Q' = S'_2 \mid S'_1 \) and we have \( R_1 \gg Q' \) by rule (Comm). Moreover, \( Q' \) is a normal form of \( R_2 \), since \( S'_i \) is a normal form of \( S_i \) for any \( i \in \{1, 2\} \).

  - The cases of (Res), (Par) and (Amb) are similar; as an example we show (Amb). We have \( Q_1 = n[S] \) and \( R_2 = n[S'] \), where \( S \gg S' \). Since \( R_1 \) is a normal form of \( Q_1 \) it means that \( R_1 = n[S''] \), where \( S'' \) is a normal form of \( S \). As \( S'' \) is a normal form of \( S \) and \( S \gg S' \), by induction hypothesis there exists \( S''' \), which is a normal form of \( S' \), such that \( S'' \gg S''' \). We take \( Q' = n[S'''] \). As \( S'''' \) is a normal form of \( S' \), then \( Q' \) is a normal form of \( R_2 \). Moreover, we have \( R_1 \gg Q' \) by applying rule (Amb) to the premise \( S'' \gg S'''' \).

  - The cases of (Res-Com), (Res-Par), (Res-Amb), (Nil-Par) and (Nil-Res) are similar; as an example we show (Res-Par). There are two cases: either \( Q_1 = (vn) (S_1 \mid S_2) \) and \( R_2 = S_1 \mid (vn) S_2 \) or the converse. We show only the former case, the other is analogous.
Since $R_1$ is a normal form of $Q_1$, it means that $R_1 = (vn) S'_1 \mid S'_2$, where $S'_i$ is a normal form of $S_i$, for any $i \in \{1, 2\}$. Taking $Q' = S'_1 \mid (vn) S'_2$ we have that $Q'$ is a normal form of $R_2$. Moreover, we have $R_1 \Rightarrow Q'$ by applying rule (Res-Par).

- In case (Bang-Bang) we have $Q_1 = !S$ and $R_2 = !S \mid S$. Since $R_1$ is a normal form of $Q_1$, it means that either $R_1 = !S'$ or $R_1 = !S'_{i \in \{1, \ldots, n\}} S'_2$, where $S \equiv S'$ (and conversely $S' \equiv S$) and $S \equiv S'_i$ (and conversely $S'_i \equiv S$), for any $i \in \{1, \ldots, n\}$. We show only the former case; the other is analogous.

  We observe that $R_1$ is a normal form of $Q_1$, and $P \equiv Q_1$, and $P \Rightarrow R_1$, where $R_1 = !S'$ and $Q_1 = !S$. It means that rule (Bang) is applied in $P \equiv Q_1$ to the premise $S' \equiv S$ (see the case (Bang) above). Therefore, by applying the induction hypothesis to $S' \equiv S$, there exists $S''$, which is a normal form of $S$, such that $S' \Rightarrow S''$.

  We take $Q' = !S' \mid S''$. By applying rule (Bang-Bang) we have $R_1 \Rightarrow !S' \mid S'$. Moreover, by applying rule (Par) to the premise $S' \Rightarrow S''$ we obtain $!S' \mid S' \Rightarrow !S' \mid S''$. Hence, by rule (Trans) we have $R_1 \Rightarrow Q'$. We conclude by observing that $Q'$ is a normal form of $R_2$ as $S \equiv S'$ and $S''$ is a normal form of $S$.

- In case (Trans) we have $Q_1 \Rightarrow S_1$ and $S_1 \Rightarrow R_2$. As $R_1$ is a normal form of $Q_1$, then by induction hypothesis there exists a process $Q''$, which is a normal form of $S_1$, such that $R_1 \Rightarrow Q''$. Since $Q''$ is a normal form of $S_1$, it means that rule (Trans) is applied in $P \equiv Q_1$ to the premise $S' \equiv S$, which is a normal form of $S$, such that $S' \Rightarrow S''$. We conclude by observing that by applying rule (Trans) to the premises $R_1 \Rightarrow Q''$ and $Q'' \Rightarrow Q'$, we obtain $R_1 \Rightarrow Q'$.

Using the property above\(^{15}\) we now conclude the case (Trans). Since $Q'$ is a normal form of $R_2$ and $R_2$ is a normal form of $Q$ (condition (ii)), we have by Proposition A.4 that $Q'$ is a normal form of $Q$. Moreover, $P \Rightarrow Q'$ follows from $P \Rightarrow R_1$ (condition (ii)) and $R_1 \Rightarrow Q'$.

We present now two auxiliary properties of the relation $\Rightarrow$ and of the reduction relation over well-labeled processes. They show that the new labels introduced in a process by $\Rightarrow$ or by a reduction can be properly re-indexed. This is possible because new labels can be introduced only by rule (Bang-Bang) of Table A.2 by means of $new$.

**Proposition A.6.** Let $P$ and $Q$ be well-labeled processes such that $P \Rightarrow Q$. We have $fn(P) = fn(Q)$, and for each re-indexing of labels $\rho_1$, such that $dom(\rho_1) = \Lambda(Q) \setminus \Lambda(P)$, and $Q \rho_1$ is well-labeled, we have also $P \Rightarrow Q \rho_1$.

**Proof.** The proof proceeds by induction on the depth of the inference of $P \Rightarrow Q$. We observe that in any rule of Table A.2, $P \Rightarrow Q$ implies $fn(P) = fn(Q)$. Moreover, in any rule of Table A.2, $P \Rightarrow Q$ implies $\Lambda(P) = \Lambda(Q)$, except from rules (Bang-Bang), (Nil-Res) and rules (Res), (Par), (Amb), (Trans).

- Suppose that rule (Nil-Res) has been applied. We have $P = 0$ and $Q = (vn_{\lambda}) 0$ or vice versa. The latter case is immediate, in the former case we have $\Lambda(Q) \setminus \Lambda(P) = \{\lambda\}$. Hence, for any re-indexing of labels $\rho_1$ such that $Q\rho_1$ is well-labeled, we have $Q\rho_1 = (vn_{\rho_1(\lambda)}) 0$. We conclude as follows by $P \Rightarrow Q\rho_1$ by rule (Nil-Res).

- Suppose that rule (Bang-Bang) has been applied. We have $P = !P_1$ and $Q = !P_1 \mid new(P_1)$. It means that $new(P_1) = P_1' \rho_1'$ for a re-indexing of labels such that: $dom(\rho_1') = \Lambda(P_1)$; $P_1' \rho_1'$ is well-labeled;

\(^{15}\) There exists a process $Q'$, which is a normal form of $R_2$, such that $R_1 \Rightarrow Q'$. 

there is no $\lambda \in \Lambda(P_1 P_1')$ such that either $\lambda \in \Lambda(P_1)$ or $H_{\xi_1}(\lambda) \in n(P_1)$. These conditions ensure that $\text{new}(P_1) \nmid P_1$ is well-labeled, and therefore that $\Lambda(\text{new}(P_1)) \cap \Lambda(P_1) = \emptyset$. Let $\rho_1$ be a re-indexing of labels such that $\text{dom}(\rho_1) = \Lambda(Q) \setminus \Lambda(P)$ and $Q\rho_1 = (\text{new}(P_1) \mid P_1)\rho_1$ is well-labeled. Since $\Lambda(\text{new}(P_1)) \cap \Lambda(P_1) = \emptyset$, we have $\text{dom}(\rho_2) = \Lambda(\text{new}(P_1))$ and $(\text{new}(P_1) \mid P_1)\rho_1 = \text{new}(P_1)\rho_1 \mid P_1$. Since $\text{new}(P_1)\rho_1 \mid P_1$ is well-labeled, also $P_1 P_1' P_1$. Thus, we can apply rule (Bang-Bang) to conclude $P_1 \gg P_1 P_1' P_1 \mid P_1$.

- The cases of rules (Res), (Par), (Amb) and (Trans) are similar and follow by induction hypothesis using the well-labeling of $P$. We give as an example (Par) and (Res).

  Assume that $P = P_1 \mid P_2$ and $Q = Q_1 \mid P_2$, where $P_1 \gg Q_1$. Let $\rho_1$ be a re-indexing of labels, such that $\text{dom}(\rho_1) = \Lambda(Q) \setminus \Lambda(P)$, and $Q\rho_1$ is well-labeled. Since $P$ and $Q$ are well-labeled, then $\Lambda(P_1) \cap \Lambda(P_2) = \emptyset$ and $\Lambda(Q_1) \cap \Lambda(P_2) = \emptyset$. Therefore, we have $\Lambda(Q) \setminus \Lambda(P) = \Lambda(Q_1) \setminus \Lambda(P) = \emptyset$. Therefore, we have $\Lambda(Q_1) \setminus \Lambda(P)$ is well-labeled. Thus, by induction hypothesis $P_1 \gg Q_1\rho_1$. We conclude by applying rule (Par) to derive $P_1 \mid P_2 \gg Q_1\rho_1 \mid P_2$.

  Assume that $P = (\nu_{n_1}) P_1$ and $Q = (\nu_{n_2}) Q_1$, where $P_1 \gg Q_1$. Let $\rho_1$ be a re-indexing of labels, such that $\text{dom}(\rho_1) = \Lambda(Q) \setminus \Lambda(P)$, and $Q\rho_1$ is well-labeled. Since $Q$ is well-labeled, then $\lambda \notin \Lambda(Q_1)$, $H_{\xi_1}(\lambda) \notin n(Q_1)$, and there is no $\mu \in \Lambda(Q_1)$ such that $H_{\xi_1}(\mu) = n$. Therefore, we have $\lambda \notin \Lambda(Q) \setminus \Lambda(P)$, and consequently $Q\rho_1 = (\nu_{n_1}) (Q_1\rho_1)$. We observe that $Q_1\rho_1$ is well-labeled, as $Q\rho_1$ is well-labeled. Hence, by induction hypothesis $P_1 \gg Q_1\rho_1$. We conclude by applying rule (Res) to derive $(\nu_{n_1}) P_1 \gg (\nu_{n_2}) (Q_1\rho_1)$.

**Proposition A.7.** Let $P$ and $Q$ be well-labeled processes such that $P \Rightarrow Q$. We have $fn(Q) \subseteq fn(P)$, and for each re-indexing of labels $\rho_1$, such that $\text{dom}(\rho_1) = \Lambda(Q) \setminus \Lambda(P)$, and $Q\rho_1$ is well-labeled, we have also $P \Rightarrow Q\rho_1$.

**Proof.** The proof proceeds by induction on the depth of the inference of $P \Rightarrow Q$. The cases of (In), (Out), and (Open) are immediate given that $\Lambda(Q) \subseteq \Lambda(P)$. The case of rule (Cong) follows by Proposition A.6. The cases of (Par), (Amb) and (Res) can be proved by induction following a reasoning similar to that used in the corresponding cases of Proposition A.6.

The following lemma shows that the converse of Lemma A.1 holds for unlabeled processes related by $\gg$.

**Lemma A.8.** Let $P$ be a well-labeled process. If $E(P) \gg Q$, then there exists a well-labeled process $Q'$, such that $P \gg Q'$ and $E(Q') = Q$.

**Proof.** We proceed by induction on the derivation of $E(P) \gg Q$ using the fact that for any rule of Table 3, which could have been applied to derive $\gg$, there exists a corresponding case in Table A.2. We discuss the most interesting cases, the others are trivial.

- In case (Bang-Bang) we have $E(P) = \mathcal{E}(P_1)$ and $Q = \mathcal{E}(P_1) \mid E(P_1)$. Let $Q' = \mathcal{E}(P_1) \mid \text{new}(P_1)$, which is (by definition of $\mathcal{E}$) well-labeled. By rule (Bang-Bang) we have $P_1 \gg P_1 \mid \text{new}(P_1)$ and $E(Q') = \mathcal{E}(P_1) \mid \mathcal{E}(\text{new}(P_1)) = \mathcal{E}(P_1) \mid \mathcal{E}(P_1) = Q$.

- In cases (Res), (Par), (Amb) we apply the induction hypothesis using Proposition A.6 to find out the well-labeled process $Q'$. We show as an example the cases of (Par) and (Res).
Assume that $\mathcal{E}(P) \gg Q$ has been derived by rule (Par). It means that $P = P_1 | R$ and $Q = Q_1 | \mathcal{E}(R)$, where $\mathcal{E}(P_1) \gg Q_1$. By induction hypothesis, there exists a well-labeled process $Q'_1$ such that $\mathcal{E}(Q'_1) = Q_1$ and $P_1 \gg Q'_1$. Using Proposition A.6 we derive that $f n(Q'_1) = f n(P_1)$. Moreover, since $P_1 | R$ is well-labeled we have: (i) $A(P_1) \cap A(R) = \emptyset$; (ii) for each $\lambda \in A(P_1)$, $H_{L_{\lambda}}(\lambda) \notin n(R)$; conversely (iii) for each $\lambda \in A(R)$, $H_{L_{\lambda}}(\lambda) \notin n(P_1)$.

We now use the fact that the labels $A(Q'_1) \setminus A(P_1)$ can be re-indexed. Therefore, let $\rho_1$ be a re-indexing of labels, such that $\text{dom}(\rho_1) = A(Q'_1) \setminus A(P_1)$. $Q'_1 \rho_1$ is well-labeled, $A(Q'_1 \rho_1) \cap A(R) = \emptyset$ and, for each $\lambda \in A(Q'_1 \rho_1)$, $H_{L_{\lambda}}(\lambda) \notin n(R)$. As $Q'_1 \rho_1$ is well-labeled, then by Proposition A.6, we obtain $P_1 \gg Q'_1 \rho_1$.

We now observe that $f n(Q'_1) = f n(P_1)$ and $f n(Q'_1) = f n(Q'_1 \rho_1)$ and that the bound names of $Q'_1 \rho_1$ can be properly $\alpha$-converted. By condition (iii) above we derive that $H_{L_{\lambda}}(\lambda) \notin n(Q'_1 \rho_1)$ for any $\lambda \in A(R)$. Moreover, $\rho_1$ has been chosen to have $A(Q'_1 \rho_1) \cap A(R) = \emptyset$ and, for each $\lambda \in A(Q'_1 \rho_1)$, $H_{L_{\lambda}}(\lambda) \notin n(R)$. Therefore, $Q'_1 \rho_1 | R$ is a well-labeled process.

Let $Q = Q'_1 \rho_1 | R$. Since $P_1 \gg Q'_1 \rho_1$, then by rule (Par) of Table A.2 we have $P \gg Q$. Moreover, since $\mathcal{E}(Q'_1) = \mathcal{E}(Q'_1) = Q_1$ we conclude that $\mathcal{E}(Q'_1 \rho_1 | R) = \mathcal{E}(Q'_1) | \mathcal{E}(R) = Q_1 | \mathcal{E}(R) = Q$.

Assume that $\mathcal{E}(P) \gg Q$ has been derived by rule (Res). It means that $P = (v \lambda) P_1$ and $Q = (v n) Q_1$, where $\mathcal{E}(P_1) \gg Q_1$. By induction hypothesis, there exists a well-labeled process $Q'_1$ such that $\mathcal{E}(Q'_1) = Q_1$ and $P_1 \gg Q'_1$. Using Proposition A.6 we derive that $f n(Q'_1) = f n(P_1)$.

Moreover, since $(v \lambda) P_1$ is well-labeled we have: (i) $\lambda \notin A(P_1)$; (ii) for each $\mu \in A(P_1)$, $H_{L_{\mu}}(\mu) \neq n$; conversely (iii) $H_{L_{\lambda}}(\lambda) \notin n(P_1)$.

We now use the fact that the labels $A(Q'_1) \setminus A(P_1)$ can be re-indexed. Therefore, let $\rho_1$ be a re-indexing of labels, such that $\text{dom}(\rho_1) = A(Q'_1) \setminus A(P_1)$. $Q'_1 \rho_1$ is well-labeled, $\lambda \notin A(Q'_1 \rho_1)$ and, for each $\mu \in A(Q'_1 \rho_1)$, $H_{L_{\mu}}(\mu) \neq n$. As $Q'_1 \rho_1$ is well-labeled, then by Proposition A.6, we obtain $P_1 \gg Q'_1 \rho_1$.

We now observe that $f n(Q'_1) = f n(P_1)$ and $f n(Q'_1) = f n(Q'_1 \rho_1)$ and that the bound names of $Q'_1 \rho_1$ can be properly $\alpha$-converted. Since for each $\mu \in A(Q'_1 \rho_1)$, $H_{L_{\mu}}(\mu) \neq n$, and by conditions (i) and (iii) above, we derive that $(v \lambda) Q'_1 \rho_1$ is well-labeled.

Let $Q = (v \lambda) Q'_1 \rho_1$. Since $P_1 \gg Q'_1 \rho_1$, then by rule (Res) of Table A.2 we have $P \gg Q$. Moreover, since $\mathcal{E}(Q'_1 \rho_1) = \mathcal{E}(Q'_1) = Q_1$ we conclude that $\mathcal{E}(Q) = (v n) Q_1 = Q$.

Using Lemmas A.5 and A.8 and the shape of normal forms we can now prove the main result of completeness.

**Lemma A.9.** Let $P$ be a well-labeled process. If $\mathcal{E}(P) \rightarrow Q$, then there exists a well-labeled process $Q'$, such that $\mathcal{E}(Q') \equiv Q$ and $P \rightarrow Q'$.

**Proof.** We prove a more general result: if $\mathcal{E}(P) \equiv P_1$ and $P_1 \rightarrow Q$, then there exists a well-labeled process $Q'$, such that $\mathcal{E}(Q') \equiv Q$ and $P \rightarrow Q'$. For this we proceed by induction on the depth of the inference of $P_1 \rightarrow Q$.

- The cases of (In), (Out), and (Open) are similar; as an example we show (In). If $P_1 \rightarrow Q$ has been obtained by rule (In), it means that $P_1 = n[\exists m. R_1 | R_2] | m[S]$ and $Q = m[n[R_1 | R_2] | S]$.

  Since $\mathcal{E}(P) \equiv P_1$, then by Lemma A.5 there exists a process $P'_1$, which is a normal form of $P_1$, such that $\mathcal{E}(P) \gg P'_1$ and $P'_1 \equiv P_1$. 


We now apply Lemma A.8. As $E(P) \Rightarrow P'_1$, then there exists a well-labeled process $P''$ such that $P \Rightarrow P''$ and $E(P'') = P'_1$. Since $P'_1$ is a normal form of $P_1$, then it must be the case that

$$P'' = n_\mu | n_y . R_1 \mid R_2 \mid m_\mu [S']$$

where $E(R'_1) \equiv R_1$, $E(R'_2)$ is a normal form $R_2$ and $E(S')$ is a normal form of $S$.

By applying rule (In) we have a reduction $P'' \rightarrow Q'$, where

$$Q' = m_\mu [n_\mu | R_1 \mid R_2 | S']$$

Moreover, since $P \Rightarrow P''$ we have by rule (Cong) $P \rightarrow Q'$. We conclude by observing that $E(R'_1) \equiv R_2$ and $E(S') \equiv S$ (using Proposition A.3). Given that also $E(R'_2) \equiv R_1$, $E(Q') \equiv Q$ follows by applying rules (Par), (Amb) and (Trans).

- The cases of (Par), (Amb) and (Res) are similar; they follow by applying the induction hypothesis and by using Proposition A.7 to find out the well-labeled process $Q'$ (similarly as in the proof of Lemma A.8). We show as an example the case (Par).

Assume that $P_1 \rightarrow Q$ has been obtained by rule (Par). It means that $P_1 = Q_1 \mid R$ and $Q = Q_2 \mid R$, where $Q_1 \rightarrow Q_2$.

Since $E(P) \equiv P_1$, then by Lemma A.5 there exists a process $P'_1$, which is a normal form of $P_1$, such that $E(P) \Rightarrow P'_1$ and $P'_1 \equiv P_1$.

We now apply Lemma A.8. As $E(P) \Rightarrow P'_1$, then there exists a well-labeled process $P''$ such that $P \Rightarrow P''$ and $E(P'') = P'_1$. Since $P'_1$ is a normal form of $P_1$, then it must be the case that $P'' = Q'_1 \mid R'$, where $E(Q'_1)$ is a normal form of $Q_1$ and $E(R')$ is a normal form of $R$. By Proposition A.3 we have that $E(Q'_1) \equiv Q_1$ and $E(Q') \equiv R$. Since $E(Q'_1) \equiv Q_1$ and $Q_1 \rightarrow Q_2$, by induction hypothesis there exists a reduction $Q'_1 \rightarrow Q'_2$ and $E(Q'_2) \equiv Q_2$.

We now use Proposition A.7 to find out a re-indexing of labels $\rho_1$ such that $Q'_1 \rightarrow Q'_2 \rho_1$ and $Q'' = Q'_2 \rho_1 \mid R'$ is well-labeled (the reasoning follows an argument similar to that applied in the proof of Lemma A.8).

As $Q''$ is well-labeled, then we derive $Q'' \rightarrow Q'$ by applying rule (Par) to the premise $Q'_1 \rightarrow Q'_2 \rho_1$. Since $P \Rightarrow P''$ we have also $P \rightarrow Q'$ by rule (Cong).

It remains to show that $E(Q') \equiv Q$. We recall that $Q = Q_2 \mid R$ and $Q' = Q'_2 \rho_1 \mid R'$, where $E(Q'_2) \equiv Q_2$ and $E(R') \equiv R$. Given that $E(Q'_2) = E(Q'_2 \rho_1)$, $E(Q') \equiv Q$ follows therefore by rules (Par) and (Trans).

- If $P_1 \rightarrow Q$ has been obtained by rule (Cong) it means that $P_1 \equiv P_2$, $P_2 \rightarrow P_3$ and $P_3 \equiv Q$. As $E(P) \equiv P_1$ and $P_1 \equiv P_2$ we have by rule (Trans) $E(P) \equiv P_2$. Since $E(P) \equiv P_2$ and $P_2 \rightarrow P_3$, then by induction hypothesis there $P \rightarrow Q'$ such that $E(Q') \equiv P_3$. We conclude by observing that $E(Q') \equiv Q$ follows by applying rule (Trans) to the premises $E(Q') \equiv P_2$ and $P_3 \equiv Q$.

A.2. Relation between the normal semantics and the reductions of labeled processes

We start presenting the basic properties of the normalisation function $\delta$ (Table 4). The following proposition shows that $\alpha$-convertible processes are represented by the same state. We recall that $\alpha$-conversion over labeled processes can change a bound name but not its label.

Proposition A.10 ($\alpha$-Conversion). Let $P$ and $Q$ be two well-labeled processes which are $\alpha$-convertible. For any $a \in A$, which is fresh for $P$ and $Q$, we have $\delta a P = \delta a Q$. 

Proof. The main observation is the following: when \( P = \langle \nu n \lambda \rangle P_1 \) and \( Q = \langle \nu k \mu \rangle P_1[k/n] \), such that \( k \not\in fn(P_1) \), we have by rule \( \textsf{DRes} \delta^a P = \delta^a(P_1[H_{\mathcal{L}_1}(\lambda)/n]) = \delta^a Q = \delta^a(P_1[k/n][H_{\mathcal{L}_1}(\lambda)/k]) \). \( \square \)

We now discuss the relation between the (free and bound) names and the labels of a process and those of the corresponding state obtained via \( \delta \). To formalise this relation it is necessary to know precisely which restrictions are removed via \( \delta \). We therefore introduce the following concepts which use a special kind of contexts. A context \( C \) is a process expression with a single occurrence of a hole \([\ ]\), such that the hole does not appear underneath the scope of a prefix or of a bang. As usual we denote by \( C[P] \) the process obtained by filling the hole of \( C \) with the process \( P \).

Let \( P \) be a labeled process. If \( P = C[\langle \nu n \lambda \rangle Q] \) for some context \( C \), then we say that \( \langle \nu n \lambda \rangle \) is an unguarded restriction of \( P \); if also \( n \not\in fn(Q) \) we say that \( \langle \nu n \lambda \rangle \) is an unguarded and unnecessary restriction of \( P \).

For instance, the restriction \( \langle \nu n \lambda \rangle \) is unguarded and the restriction \( \langle \nu m \gamma \rangle \) is not unguarded in the following process \( P = a[\langle \nu n \lambda \rangle ![\langle \nu m \gamma \rangle Q] \).

The unguarded restrictions of a process are important, as they are removed by the normalisation function \( \delta \). For instance, we have for the process \( P \) above

\[
\delta \circ P = (\{ a \circ \}, \{ a!([\nu m \gamma ] Q[H_{\mathcal{L}_1}(\lambda)/n])] \).
\]

The difference between the unguarded and the unguarded and unnecessary restrictions of a process is the following: if \( \langle \nu n \lambda \rangle \) is an unguarded and unnecessary restriction of a process \( P \), then \( H_{\mathcal{L}_1}(\lambda) \) does not necessarily appear in the state modeling \( P \). For instance, assume that the process \( P \) above is well-labeled, that is \( H_{\mathcal{L}_1}(\lambda) \not\in n(Q) \cup n(a) \). The name \( H_{\mathcal{L}_1}(\lambda) \) appears in the state \( \delta \circ P \) only when \( n \in fn(Q) \).

These intuitive ideas are stated by the propositions below. In the following, we use \( \mathcal{U}(P) = \{ n_\lambda | \langle \nu n \lambda \rangle \} \) is an unguarded restriction of \( P \) and \( \mathcal{U}_u(P) = \{ n_\lambda | \langle \nu n \lambda \rangle \} \) is an unguarded and unnecessary restriction of \( P \).

Proposition A.11. Let \( P \) be a well-labeled process and let \( a \in A \) which is fresh for \( P \). We have
1. if \( \langle \nu n \lambda \rangle \) and \( \langle \nu m \mu \rangle \) are two distinct unguarded restrictions of \( P \), then \( \lambda \neq \mu \); 
2. for any \( n_\lambda \in \mathcal{U}(P), H_{\mathcal{L}_1}(\lambda) \not\in n(P) \); 
3. \( H_{\mathcal{L}_1}(\lambda) \neq H_{\mathcal{L}_1}(\mu) \) for any \( n_\lambda, m_\mu \in \mathcal{U}(P) \).

Proof. The conditions follow straightforwardly from the definition of well-labeled process (Definition 4.2). \( \square \)

Proposition A.12. Let \( P \) be a well-labeled process and \( a \in A \) which is fresh for \( P \). We have
1. \( \nu(a) = \nu(P) \setminus \{ \lambda | n_\lambda \in \mathcal{U}(P) \}; \)
2. \( n(a) = fn(P) \cup (bn(P) \setminus \{ n | n_\lambda \in \mathcal{U}(P) \}) \cup \{ H_{\mathcal{L}_1}(\lambda) | n_\lambda \in (\mathcal{U}(P) \setminus \mathcal{U}_u(P)) \}. \)

Proof. The requirements on \( \nu(a) \) and \( n(a) \) can be proved by induction on the structure of \( P \). Using Proposition A.11, the main observation is that, by definition of \( \delta \), only the unguarded restrictions are removed (see rules \( \textsf{DBang} \) and \( \textsf{DPref} \)). In case \( \textsf{DRes} \), we have for \( P = \langle \nu n \lambda \rangle Q \)

\[
\delta^a P = \delta^a(Q[H_{\mathcal{L}_1}(\lambda)/n]).
\]
This shows that the label $\lambda$ is removed and the name $n$ is replaced by $H_{L_1}(\lambda)$. We recall that, by Proposition A.11: for any $n_\beta \in U(P)$, $H_{L_1}(\lambda) \notin n(P)$ and, there is no other object in $P$ with label $\lambda$. Therefore, $H_{L_1}(\lambda) \in n(\delta aP)$ only when $n \in fn(Q)$, that is $(vn_\beta) \in U(P) \setminus U_a(P)$. □

The following proposition is needed in the proof of completeness (Lemma A.20); it says that the state representing a well-labeled process is well-labeled provided that the root $a$ is fresh for $P$. We recall that a state $S \in S$ is well-labeled if: (i) for each $\lambda \in A(S)$, $H_{L_1}(\lambda) \notin n(S)$; (ii) for any label $\lambda \in A(S)$ there is at most one object labeled by $\lambda$.

**Proposition A.13.** Let $P$ be a well-labeled process and let $a \in A$, such that $a$ is fresh for $P$. We have that $\delta aP$ is a well-labeled state with root $a$.

**Proof.** Straightforward by induction on the structure of $P$ using Propositions A.11 and A.12. □

The converse of Proposition A.13 does not hold. Consider, for instance, the following not well-labeled process:

$$P = (vn_\beta) m_\lambda[0]. \quad (A.4)$$

We have $\delta aP = (\{m_\lambda\}, \emptyset)$ which is obviously well-labeled.

The anomaly in process (A.4) is that $(vn_\beta)$ is an unguarded and unnecessary restriction; therefore the name $H_{L_1}(\lambda)$, that is used to replace the bound name $n$, does not appear in the state representing $P$ (see Proposition A.12). By contrast, the clash between the two occurrences of label $\lambda$ is necessarily reflected into the corresponding state, when the bound name appears in the process. Consider, for instance, the following not well-labeled process:

$$Q = (vn_\beta) m_\lambda[\text{out } n]. \quad (A.5)$$

We have $\delta aQ = (\{m_\lambda\}, \{m_\lambda \text{out } \hat{n}\})$, where $H_{L_1}(\lambda) = \hat{n}$. We observe that $\delta aQ$ is not well-labeled since $\hat{n} \in n(\delta aQ)$ and $\lambda \in A(\delta aQ)$.

There is a main difference between the processes (A.4) and (A.5). In case (A.4) the process can be properly rearranged and a well-labeled process $P'$ can be obtained, such that $\delta aP = \delta aP'$ and $P \gg P'$. For instance, taking $P' = m_\lambda[0]$, it is immediate to check that $\delta aP = \delta aP'$ and $P \gg P'$, since $n \notin fn(m_\lambda[0])$ (reflecting the idea that this restriction is unnecessary). For the process (A.5) instead there is no way to modify the labels using $\gg$.

The idea explained for the processes $P$ and $P'$ above is useful in the proof of soundness (Lemma A.17). We therefore formalise it by introducing a relation $\gg$ and by showing that: when $P \gg P'$, we have $\delta aP = \delta aP'$ and $P \gg P'$ (and vice versa $P' \gg P$). The intuitive idea behind $\gg$ is that $P'$ is obtained from $P$ by eliminating all the unguarded and unnecessary restrictions. We define the relation $\gg$ over labelled processes inductively as follows:

1. $0 \gg 0$, $!P \gg !P$, $M_\lambda \gg M_\lambda$, $P$;
2. $Q \gg Q'$, if $P \gg P'$ provided that $Q \gg Q'$ and $P \gg P'$;
3. $a[Q] \gg a[Q']$ provided that $Q \gg Q'$;
4. $(vn_\beta)Q \gg (vn_\beta)Q'$ provided that $Q \gg Q'$ and $n \notin fn(Q)$;
5. $(vn_\beta)Q \gg Q'$ provided that $Q \gg Q'$ and $n \notin fn(Q)$.
Notice that by condition 5 we have \( U_\lambda(P') = \emptyset \) when \( P \triangleright P' \). Moreover, we have immediately \( fn(P) = fn(P') \) and \( \Lambda(P') \subseteq \Lambda(P) \).

**Lemma A.14.** Let \( P \) and \( P' \) be labeled processes such that \( P \triangleright P' \). We have \( \delta^a P = \delta^a P' \) and \( E(P) \equiv E(P') \). Moreover, if \( P \) and \( P' \) are well-labeled, then \( P \triangleright P' \) (and \( P' \triangleright P \)).

**Proof.** The proof proceeds by induction on the structure of \( P \). We observe that the cases of bang, prefix and nil are obvious since \( P \triangleright P' \) implies \( P = P' \). We show below the other cases.

- Suppose that \( P = b[Q] \). By definition of \( \triangleright \), we have \( P' = b[Q'] \) where \( Q \triangleright Q' \). Hence, by induction hypothesis we have \( \delta^b Q = \delta^b Q' \) and \( E(Q) \equiv E(Q') \). Also, if \( Q \) and \( Q' \) are well-labeled, then \( Q \triangleright Q' \). Using \( \delta^b Q = \delta^b Q' \) we therefore obtain \( \delta^a P = (\{ a \}, \emptyset) \cup \delta^b Q = (\{ a \}, \emptyset) \cup \delta^b Q' = \delta^a P' \).

Moreover, \( E(Q) \equiv E(Q') \), implies, by rule (Amb) of Table 3, \( n[E(Q)] = n[E(Q')] \) assuming \( b = n_\lambda \). Suppose that \( P \) and \( P' \) are well-labeled. It means that \( Q \) and \( Q' \) also are well-labeled. Using \( Q \triangleright Q' \) we derive \( b[Q] \triangleright b[Q'] \) by rule (Amb) of Table A.2.

- Suppose that \( P = Q_1 \mid Q_2 \). The proof proceeds by induction similarly as in the preceding case.

- Suppose that \( P = (vn_\lambda) \ Q \). By definition of \( \triangleright \) there are two cases: either \( P' = (vn_\lambda) \ Q' \), where \( Q \triangleright Q' \) and \( n \in fn(Q) \), or \( n \notin fn(Q) \) and \( P' = Q' \), where \( Q \triangleright Q' \).

1. Suppose that \( P' = (vn_\lambda) \ Q' \) where \( Q \triangleright Q' \). The proof proceeds by induction similarly as in the preceding case.

2. Suppose that \( n \notin fn(Q) \) and \( P' = Q' \), where \( Q \triangleright Q' \). By induction hypothesis we have \( \delta^a Q = \delta^a Q' \) and \( E(Q) \equiv E(Q') \). Also, if \( Q \) and \( Q' \) are well-labeled we have \( Q \triangleright Q' \). Using \( n \notin fn(Q) \) we have immediately \( \delta^a P = \delta^a (Q[m/n]) = \delta^a Q = \delta^a Q' = \delta^a P' \).

We observe also that \( (vn) E(Q) \equiv E(Q) \) can be derived by applying the rules (Nil-Par), (Nil-Res) and (Res-Par) of Table 3 (using \( n \notin fn(Q) \)). Since \( E(Q) \equiv E(Q') \), then we have also \( E(P) \equiv E(P') \). Similarly, for the case when \( P \) and \( P' \) are well-labeled.

We conclude by observing that, when \( P \) and \( P' \) are well-labeled, \( P \triangleright P' \) implies \( P' \triangleright P \). In any case shown above only the symmetric rules of \( \triangleright \) have been applied (see Table A.2). \( \square \)

**Soundness.** The proof is rather complex; it is difficult in particular to reason about the well-labeling of the processes obtained in the inductive cases. We need some auxiliary properties. The following proposition shows a useful property of the reductions of well-labeled processes.

**Proposition A.15.** Let \( P \) and \( Q \) be well-labeled processes such that \( P \rightarrow Q \). If there exists \( \lambda \) such that \( \lambda \in \Lambda(Q) \) and \( H_{\lambda}^{L_{\lambda}}(\lambda) \in fn(P) \), then there exists a well-labeled process \( Q' \), such that \( P \rightarrow Q' \), \( \lambda \notin \Lambda(Q') \) and \( Q' \triangleright Q \).

**Proof.** We first observe that by definition of well-labeling: \( H_{\lambda}^{L_{\lambda}}(\lambda) \in fn(P) \) implies \( \lambda \notin \Lambda(P) \); analogously, \( \lambda \in \Lambda(Q) \) implies \( H_{\lambda}^{L_{\lambda}}(\lambda) \notin fn(Q) \). The proofs proceed by induction on the depth of the inference of \( P \rightarrow Q \). The cases of (In), (Out), and (Open) are immediate given that \( \Lambda(Q) \subseteq \Lambda(P) \). In case (Cong) we have \( P \triangleright P' \) and \( P' \rightarrow P'' \) and \( P'' \triangleright Q \). If \( \lambda \notin \Lambda(P'') \) we have finished. Otherwise, we observe...
that \( H_L(\lambda) \in fn(P') \), using \( H_L(\lambda) \in fn(P) \) and Proposition A.6. As \( P' \) is well-labeled, by induction hypothesis there exists \( R'' \) such that \( P' \rightarrow R'' \), \( \lambda \notin L(R'') \) and \( R'' \gg P'' \gg Q \). Hence, by rule (Cong) we derive \( P \rightarrow R'' \) such that \( \lambda \notin L(R'') \), \( R'' \gg Q \).

The other cases are similar and follow by induction hypothesis; we discuss as an example the case (Par). It means that \( P = P_1 \mid P_2 \) and \( Q = P_1' \mid P_2 \), where \( P_1 \rightarrow P_1' \). We have \( \lambda \in L(Q) \) and \( \lambda \notin L(P) \), \( H_L(\lambda) \in fn(P) \) and \( H_L(\lambda) \notin fn(Q) \). Since \( P \) and \( Q \) are well-labeled, the only possibility is therefore that \( \lambda \in L(P_1') \) and \( H_L(\lambda) \in fn(P_1) \). Hence, by induction hypothesis there exists \( P_1'' \) such that \( P_1 \rightarrow P_1'' \), \( \lambda \notin L(P_1'') \) and \( P_1'' \gg P_1' \). We observe that, by Proposition A.6, \( P_1'' \gg P_1' \) implies \( fn(P_1'') = fn(P_1') \). Given \( fn(P_1') = fn(P_1'') \) and \( P_1' \mid P_2 \) is well-labeled, there exists a re-indexing of labels \( \rho_1 \), such that \( dom(\rho_1) = A(P_1'') \setminus A(P_1'), \lambda \notin \rho_1 \) and \( Q' = P_1'' \rho_1 \mid P_2 \) is well-labeled. By Propositions A.6 and A.7 we obtain both \( P_1'' \rho_1 \gg P_1' \) and \( P_1 \rightarrow P_1'' \rho_1 \). Since \( Q' \) is well-labeled, we derive \( P_1 \mid P_2 \rightarrow Q' \) by applying rule (Par) to the premise \( P_1 \rightarrow P_1'' \rho_1 \). Moreover, we have \( P_1'' \rho_1 \mid P_2 \gg P_1' \mid P_2 \) by applying rule (Par) to the premise \( P_1'' \rho_1 \gg P_1' \). \( \square \)

To reason about the well-labeling of processes it is convenient to know precisely which new labels are introduced by a transition \( S_1 \rightarrow S_2 \) between two states \( S_1 \) and \( S_2 \). To this aim we use \( \text{new}(S_1 \rightarrow S_2) \) to denote the set of labels which could have been introduced by an application of \textit{new}, that is by the unfolding of \textit{new}. Formally,

1. \( \text{new}(S_1 \rightarrow S_2) = \emptyset \) when \( S_1 \rightarrow S_2 \) has been obtained by one of the rules \textit{In}, \textit{Out} or \textit{Open};
2. \( \text{new}(S_1 \rightarrow S_2) = A(\text{new}_{S_1}(P)) \) when \( S_1 \rightarrow S_2 \) has been obtained by rule \textit{Bang} and \( S_2 = S_1 \cup \delta^a\text{new}_{S_1}(P) \).

The following proposition shows that the well-labeling of states is preserved by the transitions of Table 5 and clarifies in which sense the labels introduced by means of \textit{new} in rule \textit{Bang} are fresh.

**Proposition A.16.** Let \( S_1 \) be a well-labeled state. If \( S_1 \rightarrow S_1' \), then \( S_1' \) is well-labeled. Moreover, assume that \( S_1 \cup S_2 \) is a well-labeled state such that \( S_1 \cup S_2 \rightarrow S_1' \cup S_2 \). For any \( \lambda \in \text{new}(S_1 \rightarrow S_1') \) we have \( \lambda \notin L(S_2) \) and \( H_L(\lambda) \notin n(S_2) \).

**Proof.** The proof is by cases on the rule applied to obtain \( S_1 \rightarrow S_1' \). Let \( S_i = (T_i, C_i) \), for any \( i \in \{ 1, 2 \} \).

- As cases of \textit{Open}, \textit{In} and \textit{Out} are similar, we discuss case \textit{In} only. When \( S_1 \rightarrow S_1' \) has been obtained by rule \textit{In}, we have \( t = a \in m_P \), \( P \in C_1 \), \( a^b, m^b_\mu \in T_1 \) such that \( a \neq m_\mu \). Moreover, \( S_1' = S_1'' \cup \delta^aP \), where

\[
S_1'' = ((T_1 \setminus \{ a^b \}) \cup \{ m^a \}, C_1 \setminus \{ t \}).
\]

Since \( S_1 \) is well-labeled and \( t \in C_1 \), then \( P \) is well-labeled and \( a \) is fresh for \( P \). By Proposition A.13 we have that \( \delta^aP \) is well-labeled. We also observe that \( S_1'' \) is well-labeled, as \( S_1 \) is well-labeled. Hence, \( S_1' \) is not well-labeled only when there exists a label \( \lambda \), such that one of the following cases holds: (a) \( \lambda \in A(\delta^aP) \setminus A(a) \) and either \( \lambda \in A(S_1'') \) or \( H_L(\lambda) \in n(S_1'') \); (b) \( \lambda \in A(S_1'') \) and \( H_L(\lambda) \in n(\delta^aP) \setminus n(a) \).

To discuss (a) and (b) we need to know the relation between the names and the labels of \( P \) and those of \( \delta^aP \). By Proposition A.12, we have

1. \( A(\delta^aP) \setminus A(a) = A(P) \setminus \{ \lambda \mid n_\lambda \in U(P) \}; \)
2. \( n(\delta^aP) \setminus n(a) = fn(P) \cup (bn(P) \setminus \{ n \mid n_\lambda \in U(P) \}) \cup \{ H_L(\lambda) \mid n_\lambda \in (U(P) \setminus U_a(P)) \}. \)
We show case (b). Assume that \( \lambda \in A(S''_1) \) and \( H_{\mathcal{L}_t}(\lambda) \in n(\delta aP) \setminus n(a) \). Given (2) we derive that either \( H_{\mathcal{L}_t}(\lambda) \in n(P) \) or \( H_{\mathcal{L}_t}(\lambda) \in \{ H_{\mathcal{L}_t}(\lambda) \mid n_\lambda \in (U(P) \setminus U_a(P)) \} \).

In the former case, since \( H_{\mathcal{L}_t}(\lambda) \in n(P) \) and \( t \in C_1 \) we have \( H_{\mathcal{L}_t}(\lambda) \in n(S_1) \). Moreover, we have \( A(S''_1) \subseteq A(S_1) \). We obtain \( \lambda \in A(S_1) \) and \( H_{\mathcal{L}_t}(\lambda) \in n(S_1) \), which contradicts the well-labeling of \( S_1 \).

In the latter case, we have \( \lambda \in A(P) \) and \( \lambda \in A(S''_1) \). Hence, there is an object with label \( \lambda \) in \( S''_1 \).

Since \( t \) has been removed from the configuration and \( t \in C_1 \), there are two objects with label \( \lambda \) in \( S_1 \), which contradicts again the well-labeling of \( S_1 \).

Case (a) follows by applying a similar argument using condition (1), \( n(S''_1) \subseteq n(S_1) \) and \( A(S''_1) \subseteq A(S_1) \).

Let \( S_1 \cup S_2 \) be a well-labeled state such that \( S_1 \cup S_2 \mapsto S'_1 \cup S_2 \). We conclude by observing that \( \text{new}(S_1 \mapsto S_2) = \text{new}(S_1 \cup S_2 \mapsto S'_1 \cup S_2) = \emptyset \).

- Suppose that \( S_1 \mapsto S'_1 \) has been obtained by rule \textbf{Bang}. It means that \( S'_1 = (C_1, T_1) \cup \delta '\text{new}_{S_1}(Q) \) for some \( \text{bang} \in C_1 \). By definition, we have \( \text{new}_{S_1}(Q) = Q_{\rho_1} \) for a re-indexing of labels \( \rho_1 \) such that \( \text{dom}(\rho_1) = A(Q) \) and
  (1) \( Q_{\rho_1} \) is well-labeled;
  (2) there is no \( \lambda \in A(Q_{\rho_1}) \), such that either \( \lambda \in A(S_1) \) or \( H_{\mathcal{L}_t}(\lambda) \in n(S_1) \).

By conditions (1) and (2), \( Q_{\rho_1} \) is well-labeled and \( c \) is fresh for \( Q_{\rho_1} \). Consequently, by Proposition A.13, \( \delta '\text{new}_{S_1}(Q) \) is well-labeled. Since \( S_1 \) and \( \delta '\text{new}_{S_1}(Q) \) are well-labeled, \( S'_1 \) is not well-labeled only when there exists \( \lambda \) such that one of the following cases holds: (a) \( \lambda \in A(\delta '\text{new}_{S_1}(Q)) \setminus A(c) \) and either \( \lambda \in A(S_1) \) or \( H_{\mathcal{L}_t}(\lambda) \in n(S_1) \); (b) \( \lambda \in A(S_1) \) and \( H_{\mathcal{L}_t}(\lambda) \in \delta '\text{new}_{S_1}(Q) \setminus n(c) \).

By Proposition A.12, we have

(i) \( A(\delta '\text{new}_{S_1}(Q)) \setminus A(c) = A(\text{new}_{S_1}(Q)) \setminus \{ \lambda \mid n_\lambda \in U(\text{new}_{S_1}(Q)) \} \);  
(ii) \( n(\delta '\text{new}_{S_1}(Q)) \setminus n(c) = f(n(\text{new}_{S_1}(Q)) \cup (\text{bang}(\text{new}_{S_1}(Q)) \setminus \{ n \mid n_\lambda \in U(\text{new}_{S_1}(Q)) \}) \cup \{ H_{\mathcal{L}_t}(\lambda) \mid n_\lambda \in (U(\text{new}_{S_1}(Q)) \setminus U_a(\text{new}_{S_1}(Q))) \} \).

In case (a) we have \( \lambda \in A(\delta '\text{new}_{S_1}(Q)) \setminus A(c) \), and consequently \( \lambda \in A(\text{new}_{S_1}(Q)) \) using (i). When either \( \lambda \in A(S_1) \) or \( H_{\mathcal{L}_t}(\lambda) \in n(S_1) \) we have a contradiction with the requirement (2) above.

In case (b) we have \( H_{\mathcal{L}_t}(\lambda) \in \delta '\text{new}_{S_1}(Q) \setminus n(c) \). Using (ii) we obtain that either \( H_{\mathcal{L}_t}(\lambda) \in n(\text{new}_{S_1}(Q)) \) or \( \lambda \in A(\text{new}_{S_1}(Q)) \). In the latter case, we have \( \lambda \in A(\text{new}_{S_1}(Q)) \) and \( \lambda \in A(S_1) \), which contradicts the requirement (2) above. In the former case we have \( H_{\mathcal{L}_t}(\lambda) \in n(\text{new}_{S_1}(Q)) \) and \( \lambda \in A(S_1) \). We observe that \( n(Q) = n(\text{new}_{S_1}(Q)) \) and \( \text{bang} \in C_1 \). Hence, we have \( H_{\mathcal{L}_t}(\lambda) \in n(S_1) \) and \( \lambda \in A(S_1) \), which contradicts the well-labeling of \( S_1 \).

Let \( S_1 \cup S_2 \) be a well-labeled state such that \( S_1 \cup S_2 \mapsto S'_1 \cup S_2 \). We observe that it is necessary to have \( \text{new}_{S_1 \cup S_2}(Q) = Q_{\rho_1} \), that is (besides condition (1) above): there is no \( \lambda \in A(Q_{\rho_1}) \), such that either \( \lambda \in A(S_1 \cup S_2) \) or \( H_{\mathcal{L}_t}(\lambda) \in n(S_1 \cup S_2) \). Given \( \text{new}(S_1 \mapsto S'_1) = A(Q_{\rho_1}) \), we have finished. \( \square \)

Now we show the main result of soundness.

**Lemma A.17.** Let \( P \) be a well-labeled process and let \( \delta aP = S_1 \) where \( a \in A \) is fresh for \( P \). If \( S_1 \mapsto S_2 \), then there exists a well-labeled process \( Q \), such that \( a \) is fresh for \( Q \), \( \delta Q = S_2 \), \( P \rightarrow Q \) and \( A(Q) \setminus A(P) \subseteq \text{new}(S_1 \mapsto S_2) \).
The proof is by induction on the structure of $P$.

- Assume $P = 0$ or $P = M\lambda. P_1$. We have $\delta^0 = (\emptyset, \emptyset) = S_1$ and $\delta^aM\lambda. P_1 = (\emptyset, [a]M\lambda. P_1) = S_1$, respectively. In both cases the proof is trivial because there is no transition from $S_1$.

- Assume $P = !P_1$. We have $\delta^a!P_1 = (\emptyset, ![a]P_1) = (T_1, C_1) = S_1$. Transition $S_1 \rightarrow S_2$ could have been obtained only by applying rule **Bang**. It means that $S_2 = (\emptyset, ![a]P_1) \cup \delta^a\text{new}S_1(P_1)$. Let $Q = !P_1 | \text{new}S_1(P_1)$. We observe that by definition of $\text{new}$ and since $!P_1 \in C_1$, then $Q$ is a well-labeled process. Therefore, by rule (Bang-Bang) of Table A.2 we derive $!P_1 \gg !P_1 | \text{new}S_1(P_1)$. We also have $\delta^aQ = (\emptyset, ![a]P_1) \cup \delta^a\text{new}S_1(P_1)$. We conclude by noticing that $A(Q) \setminus A(P) = A(\text{new}S_1(P_1)) = \text{new}(S_1 \rightarrow S_2)$.

- Assume $P = (\nu n\lambda) P_1$. We have $\delta^a(\nu n\lambda) P_1 = \delta^aP_1'[\nu n\lambda] = S_1$, where $m = H_{\lambda n}(\lambda)$ and $P_1'[\nu n\lambda] = P_1[m/n]$. Since $P_1$ is well-labeled, $\lambda \notin A(P_1')$, and consequently $P_1'[\nu n\lambda]$ is well-labeled. Hence, by induction hypothesis there exists a well-labeled process $Q_1$ such that $\delta^aQ_1 = S_2$, $P_1' \rightarrow Q_1$ and $A(Q_1) \setminus A(P_1') \subseteq \text{new}(S_1 \rightarrow S_2)$.

There are two cases: either $P_1' \gg Q_1$ or $P_1' \rightarrow Q_1$. We show only the latter one, the other being analogous. We show the existence of a well-labeled process $Q$, such that $P \rightarrow Q$, $\delta^aQ = S_2$ and $A(Q) \setminus A(P) = \text{new}(S_1 \rightarrow S_2)$.

The crucial observation to find out the right process $Q$ is that $Q_1$ is a well-labeled process: it cannot be the case that $\lambda \in A(Q_1)$ and $m \in n(Q_1)$, where $m = H_{\lambda n}(\lambda)$.

1. Assume that $\lambda \notin A(Q_1)$. Let $k$ be a new name, such that $k \neq m$ and $k \notin n(Q_1) \cup n(P_1)$ and there is no $\mu \in (A(Q_1) \cup A(P_1))$ with $H_{\lambda n}(\mu) = k$. We take $Q = (\nu k\lambda) Q_1[k/m]$. Since $\lambda \notin A(Q_1)$ we have also $\lambda \notin A(Q_1)[k/m]$. Considering $k$ has been properly chosen, $Q$ is well-labeled. Moreover, we have $\delta^aQ = \delta^a(Q_1[k/m][m/k]) = \delta^aQ_1 = S_2$.

We now show that $P \rightarrow Q$. Since $P_1' \rightarrow Q_1$ and $k$ is a new name, we have also $P_1'[k/m] \rightarrow Q_1[k/m]$. Therefore, we derive $(\nu k\lambda) P_1'[k/m] \rightarrow (\nu k\lambda) Q_1[k/m]$ by applying rule (Res) to the premise $P_1'[k/m] \rightarrow Q_1[k/m]$. We also observe that $(\nu n\lambda) P_1$ is $\alpha$-convertible to $(\nu k\lambda) P_1[m/n][k/m]$.

It remains to show that $A(Q) \setminus A(P) \subseteq \text{new}(S_1 \rightarrow S_2)$. Since $\lambda \notin A(Q_1)$ we have $A(Q) \setminus A(P) = (A(\nu k\lambda) Q_1[k/m]) \setminus (A(\nu n\lambda) P_1[m/n]) = A(Q_1) \setminus A(P_1') \subseteq \text{new}(S_1 \rightarrow S_2)$.

2. Assume that $\lambda \in A(Q_1)$ and $m \notin n(Q_1)$. We take $Q = Q_1$. Since $Q_1$ is well-labeled and $\delta^aQ_1 = S_2$, it remains to show that $P \rightarrow Q_1$. The proof proceeds by considering the following two cases: $m \notin fn(P_1')$ or $m \in fn(P_1')$.

When $m \notin fn(P_1')$, we observe that $\lambda \notin fn(P_1)$, that is $P_1' = P_1$. Using $n \notin fn(P_1)$ we derive, by rules (Nil-Par), (Nil-Res) and (Res-Par), $(\nu n\lambda) P_1 \gg P_1$. Since $P_1 = P_1'$ and $P_1' \rightarrow Q_1$ we obtain by rule (Cong) $P \rightarrow Q_1$.

If $m \in fn(P_1')$ the proof is more complex. We use the fact that $P_1'$ is well-labeled, that is $\lambda \notin A(P_1')$. Since $P_1' \rightarrow Q_1$ and $m \notin fn(Q_1)$ we can apply Proposition A.15. We derive that there exists $Q_1'$ such that $Q_1' \gg Q_1$, $P_1' \rightarrow Q_1'$ and $\lambda \notin A(Q_1')$.

Since $\lambda \notin A(Q_1')$, the process $(\nu k\lambda) Q_1'[k/m]$ is well-labeled, where $k$ is a new name chosen as in case 1. above. Moreover, by applying rule (Res) to the premises $P_1' \rightarrow Q_1'$ we obtain $(\nu k\lambda) P_1'[k/m] \rightarrow (\nu k\lambda) Q_1'[k/m]$.

We now deduce $(\nu k\lambda) P_1'[k/m] \rightarrow Q_1$ from $(\nu k\lambda) P_1'[k/m] \rightarrow (\nu k\lambda) Q_1'[k/m]$. Since $Q_1' \gg Q_1$ and $m \notin n(Q_1)$, then by Proposition A.6, $m \notin fn(Q_1')$, that is $k \notin fn(Q_1'[k/m])$. Hence, by applying rules (Nil-Par), (Nil-Res) and (Res-Par) we obtain $(\nu k\lambda) Q_1'[k/m] \gg Q_1'$. Using $Q_1' \gg
\(Q_1\) we have also \((vk_2) Q'_1[k/m] \gg Q_1\). By rule (Cong) we therefore obtain \((vk_2) P'_1[k/m] \rightarrow Q_1\). Moreover, we have that \((vn_1) P_1\) is \(\alpha\)-convertible to \((vk_2) P_1[m/n][k/m]\).

We conclude by observing that \(A(Q) \setminus A(P) = A(Q_1) \setminus (A(P_1) \cup \{\lambda\})\). Since \(\lambda \notin A(P_1)\) and \(A(P_1) = A(P'_1)\) we have therefore \(A(Q_1) \setminus A(P) \subseteq A(Q) \setminus A(P'_1) \subseteq \text{new}(S_1 \mapsto S_2)\).

- Assume \(P = b[P_1]\). We have \(\delta^a b[P_1] = (\{b^a\}, \emptyset) \cup \delta^b P_1 = S_1\). Transition \(S_1 \mapsto S_2\) could have been obtained in two ways: either only \(P_1\) contributes to the action or also ambient \(b\) participates. Notice that ambient \(a\) cannot be involved as \(a\) is fresh for \(P\) and \(P\) is well-labeled. This guarantees that \(S_1\) is a well-labeled state with root \(a\) (see Proposition A.13). Let \(S'_1 = \delta^b P_1 = (T'_1, C'_1)\).

(1) If only \(P_1\) contributes to the action it means that \(S'_1 \mapsto S'_2\) and \(S_2 = S'_2 \cup (\{b^a\}, \emptyset)\). As \(P\) is well-labeled, \(P_1\) also is well-labeled and \(b\) is fresh for \(Q\). Therefore, by induction hypothesis there exist a well-labeled process \(Q_1\), such that \(\delta^b Q_1 = S'_2, P_1 \rightarrow Q_1\) and \(A(Q_1) \setminus A(P_1) \subseteq \text{new}(S'_1 \mapsto S'_2)\).

There are two cases: either \(P_1 \gg Q_1\) or \(P_1 \rightarrow Q_1\). We show only the latter case, the other being analogous. The proof proceeds by showing that \(b[Q_1]\) is well-labeled and that \(a\) is fresh for \(b[Q_1]\). The well-labeling of \(Q\) is a necessary condition to derive a reduction \(b[P_1] \rightarrow b[Q_1]\) by applying rule (Amb) to the premise \(P_1 \mapsto Q_1\). Let \(Q = b[Q_1]\).

Assume that either \(Q\) is not well-labeled or \(a\) is not fresh for \(Q\). We recall that \(Q_1\) is well-labeled and that \(S_1 = S'_1 \cup (\{b^a\}, \emptyset)\) is a well-labeled state. Therefore, the only possibility is that there exists a label \(\lambda\), such that one of the following cases holds: (i) \(\lambda \in A(Q_1)\) and either 
\[\lambda \in A(a) \cup A(b)\] and \(H_{C_1}(\lambda) \in n(a) \cup n(b)\); (ii) \(\lambda \in A(a) \cup A(b)\) and \(H_{C_1}(\lambda) \in n(Q_1)\).

We consider case (ii) first. Since the bound names of \(Q_1\) can be \(\alpha\)-converted, when needed, the interesting case is when \(H_{C_1}(\lambda) \in f n(Q_1)\). In this case we use \(P_1 \mapsto Q_1\) and we derive, by Proposition A.7, \(f n(Q_1) \subseteq f n(P_1)\). Since \(H_{C_1}(\lambda) \in f n(Q_1)\) we have therefore \(H_{C_1}(\lambda) \in f n(P_1)\), and also \(H_{C_1}(\lambda) \notin f n(P)\). Given that \(\lambda \in A(a) \cup A(b)\) this contradicts either the well-labeling of \(P\) or the freshness of \(a\) for \(P\).

In case (i) we have \(\lambda \in A(Q_1)\). We observe that it is not possible that \(\lambda \in A(P)\). This because \(\lambda \in A(a) \cup A(b)\) and \(\lambda \in A(P)\) contradict either the well-labeling of \(P\) or the freshness of \(a\) for \(P\). Similarly for \(H_{C_1}(\lambda) \in n(a) \cup n(b)\) and \(\lambda \in A(P)\). Therefore, we have \(\lambda \notin A(P_1)\) and \(\lambda \in A(Q_1) \setminus A(P_1)\). We now use the fact that \(A(Q_1) \setminus A(P_1) \subseteq \text{new}(S'_1 \mapsto S'_2)\) and we deduce \(\lambda \in \text{new}(S'_1 \mapsto S'_2)\).

We observe that \(S_1 \mapsto S_2\), where \(S_1 = S'_1 \cup (\{b^a\}, \emptyset)\) and \(S_2 = S'_2 \cup (\{b^a\}, \emptyset)\). Therefore, by Proposition A.16, there is no \(\mu \in \text{new}(S'_1 \mapsto S'_2)\) such that either \(\mu \in A((\{b^a\}, \emptyset))\) or \(H_{C_1}(\mu) \in n((\{b^a\}, \emptyset))\). Since \(A((\{b^a\}, \emptyset)) = A(a) \cup A(b)\) and \(n((\{b^a\}, \emptyset)) = n(a) \cup n(b)\) we have: \(\lambda \in \text{new}(S'_1 \mapsto S'_2)\) and either \(\lambda \in A(a) \cup A(b)\) or \(H_{C_1}(\lambda) \in n(a) \cup n(b)\). This is a contradiction.

Since \(Q\) is well-labeled, then a reduction \(b[P_1] \rightarrow b[Q_1]\) can be obtained by applying rule (Amb) to the premise \(P_1 \mapsto Q_1\). Moreover, we have that \(a\) is fresh for \(Q\) and

\[\delta^a Q = (\{b^a\}, \emptyset) \cup \delta^b Q_1 = (\{b^a\}, \emptyset) \cup S'_2 = S_2.\]

It remains to show that \(A(Q) \setminus A(P) \subseteq \text{new}(S_1 \mapsto S_2)\). This follows immediately using \(A(Q_1) \setminus A(P_1) \subseteq \text{new}(S'_1 \mapsto S'_2)\), \(\text{new}(S'_1 \mapsto S'_2) = \text{new}(S_1 \mapsto S_2)\) and \(A(Q_1) \setminus A(P_1) = A(Q) \setminus A(P)\) (as \(a \notin A(Q_1) \cup A(P_1)\)).
(2) If both \( P_1 \) and \( b \) participate to the action, the only possibility is that some ambient \( c \), which is top level inside \( b \), goes out of \( b \). It means that transition \( S_1 \mapsto S_2 \) has been obtained by rule Out. Therefore, there exist \( e \in T'_1 \) and \( \{ b \} \in C'_1 \), such that \( b = n_y \), and
\[
S'_1 = \delta b T \cup (\{ b \}, \emptyset) \cup (\emptyset, {\text{out}} n_y, R) \cup \delta c U
\]
for some processes \( T \) and \( U \). Moreover, the state \( S_2 \) reached from \( S_1 \) (by rule Out) is
\[
S_2 = (\{ b \}, \emptyset) \cup (\{ c \}, \emptyset) \cup \delta b T \cup \delta c U \cup \delta c R.
\]

We now use \( \delta b P_1 = S'_1 \) and the shape of \( S'_1 \) to infer the structure of \( P_1 \). Examining the cases in the definition of \( \delta \), we observe that: the components \( (\{ b \}) \) and \( (\emptyset, {\text{out}} n_y, R) \) tell us that rules DAmb and DPref (possibly after rules DRes and DPar) have been used. Therefore, we have
\[
P_1 \Rightarrow (v \tilde{P}_1) (T' \mid c'[\text{out} n_y, R' \mid U'])
\]
where \( c = c' \eta \) and \( T = T' \eta \), \( U = U' \eta \) and \( R = R' \eta \) for the substitution \( \eta : \mathcal{N} \to \mathcal{N}_I \) such that \( \eta(p) = H_{\mathcal{L}_I}(\mu) \).

Notice that we have grouped together the (eventual) unguarded restrictions by means of \( \gg \). This result is based on the underlying assumption that the bound names \( \tilde{P} \) can be \( \alpha \)-converted and on the following properties due to the well-labeling if \( P \): (i) \( H_{\mathcal{L}_I}(\mu) \neq \emptyset \) for any \( \mu \in \tilde{P} \); (ii) \( n \notin \tilde{P} \). Condition (i) follows from \( n \in \mathcal{N}(P) \) using Proposition A.11. Condition (ii) follows from the fact that the restrictions are unguarded, since by Proposition A.12 any unguarded restriction is removed. Consequently, \( n \notin \tilde{P} \) implies \( n \notin \delta b P_1 \), which contradicts \( \text{out} n_y, R \in C'_1 \).

We now exploit the condition \( n \notin \tilde{P} \) to derive, by applying rules (Amb) and (Res-Amb), that \( P \gg P' \) where
\[
P' = (v \tilde{P}_1) (n_y [T' \mid c'[\text{out} n_y, R' \mid U'])
\]

Let \( Q = (v \tilde{P}_1) b[T'] \mid c'[R' \mid U'] \) which is obviously well-labeled. Moreover, we have \( \delta b Q = S_2 \) and by rules (Out) and (Res) \( P' \to Q' \). We therefore derive \( P \to Q \) by applying rule (Cong).

We conclude by observing that \( \lambda(Q) \subseteq \lambda(P) \). Thus, we have \( \lambda(Q) \setminus \lambda(P) = \text{new}(S_1 \mapsto S_2) = \emptyset \).

• Assume \( P = P_1 \mid P_2 \). We have \( \delta a P_1 \mid P_2 = \delta a P_1 \cup \delta a P_2 = S_1 \). Transition \( S_1 \mapsto S_2 \) could have been obtained in two ways: either only one of \( P_1 \) and \( P_2 \) participates to the action or the two processes interact with each other. In the latter case, we observe that ambient \( a \) cannot be involved as \( a \) is fresh for \( P \). This guarantees that the topology is a tree with root \( a \) (see Proposition A.13). Therefore, \( S_1 \mapsto S_2 \) could have been obtained by the application either of rule In or of rule Open. In both cases the interaction may involve only processes and ambients which are top level inside \( a \). Let \( \delta a P_1 = (T_1, C_1) = S'_1 \) and \( \delta a P_2 = (T_2, C_2) = S'_2 \).

(1) Suppose that only \( P_1 \) contributes to the action. We have \( S_1 = S'_1 \cup S'_2 \) and \( S_2 = S'_0 \cup S'_2 \), where \( S'_1 \mapsto S'_2 \). Since \( P \) is well-labeled and \( a \) is fresh for \( P \), then also \( P_1 \) is well-labeled and \( a \) is fresh for \( P_1 \), for any \( i \in \{1, 2\} \). Hence, by induction hypothesis, we have \( P_1 \mapsto P'_1 \) for a well-labeled process \( P'_1 \), such that \( a \) is fresh for \( P'_1 \), \( \delta a P'_1 = S'_0 \) and \( \lambda(P'_1) \setminus \lambda(P_1) \subseteq \text{new}(S'_1 \mapsto S'_2) \).

There are two cases: either \( P_1 \gg P'_1 \) or \( P_1 \mapsto P'_1 \). We show only the latter case, the other being analogous.

Similarly to the case of ambient we can apply rule (Par) to derive a transition \( P_1 \mid P_2 \mapsto P'_1 \mid P_2 \) only when \( P'_1 \mid P_2 \) is well-labeled. This case is however more complex as it may be the case that
$P_1' \mid P_2$ is not well-labeled. We therefore consider a slightly different process $Q = P_1' \mid P_2'$, where $P_2 \triangleright P_2'$. We observe that, by definition of $\triangleright$, $fn(P_2) = fn(P_2')$ and $A(P_2') \subseteq A(P_2)$. Therefore, $P_2'$ is well-labeled and $a$ is fresh for $P_1'$, as $P_2$ is well-labeled and $a$ is fresh for $P_2$. Moreover, by Lemma A.14, we have $P_2 \gg P_2'$ and $\delta^a P_2' = \delta^a P_2 = S_2'$.

We now show that $Q = P_1' \mid P_2'$ is a well-labeled process. Assume that this is not the case. Since $P_1'$ and $P_2'$ are well-labeled the only possibility is that there exists a label $\lambda$ such that one of the following cases hold: (i) $\lambda \in A(P_1')$ and either $\lambda \in A(P_2')$ or $H_{L_i}(\lambda) \in n(P_2')$; (ii) $\lambda \in A(P_2')$ and $H_{L_i}(\lambda) \in n(P_2')$.

We discuss before case (ii). Since the bound names of $P_1'$ can be $\alpha$-converted, when needed, the interesting case is when $H_{L_i}(\lambda) \in f n(P_1')$. We use $P_1 \rightarrow P_1'$ and we obtain, by Proposition A.7, $fn(P_1') \subseteq fn(P_1)$. Hence, we have $H_{L_i}(\lambda) \in fn(P_1)$. Given that $A(P_1') \subseteq A(P_2)$ we obtain $\lambda \in A(P_2)$ and $H_{L_i}(\lambda) \in fn(P_1)$. This contradicts the well-labeling of $P_1 \mid P_2$.

In case (i) we have $\lambda \in A(P_1')$. We observe that it cannot be the case that also $\lambda \in A(P_1)$. This because the well-labeling of $P_1 \mid P_2$ contradicts $\lambda \in A(P_1')$ (which follows from $\lambda \in A(P_2')$). Similarly for $\lambda \in A(P_1)$ and $H_{L_i}(\lambda) \in n(P_2)$ (which follows from $H_{L_i}(\lambda) \in n(P_2')$). Therefore, we have $\lambda \notin A(P_1)$ and $\lambda \in A(P_1')$, that is $\lambda \in A(P_1') \setminus A(P_1)$. We now use the fact that $A(P_1') \setminus A(P_1) \subseteq \text{new}(S_1' \rightarrow S_1')$ and we derive $\lambda \in \text{new}(S_1' \rightarrow S_1')$.

We recall that $S_1 \mapsto S_2$, where $S_1 = S_1' \cup S_1'$ and $S_2 = S_1'' \cup S_2'$. Therefore, by Proposition A.16, there is no $\mu \in \text{new}(S_1' \rightarrow S_1')$ such that either $\mu \in A(S_2')$ or $H_{L_i}(\mu) \in n(S_2')$. Hence, it must be the case that (a) $\lambda \notin A(S_1')$ and (b) $H_{L_i}(\lambda) \notin n(S_2')$.

We now use the fact that $\delta^a P_2' = S_2'$. By Proposition A.12, we have

\[
\begin{align*}
& A(S_1') \setminus A(a) = A(P_2') \setminus \{\lambda \in S_1' \mid n_2 \in U(P_2')\}; \\
& n(S_1') \cap n(a) = fn(P_2') \cup bn(P_2') \setminus \{n \in U(P_2') \mid n_2 \in U(P_2') \setminus U(a(P_2'))\}; \\
& H_{L_i}(\lambda) \in n(S_1') \setminus \text{new}(S_1' \rightarrow S_1'). \end{align*}
\]

Using the results above, we now show that both possibilities $\lambda \in A(P_2')$ and $H_{L_i}(\lambda) \in n(P_2')$ contradict either (a) or (b).

Assume that $H_{L_i}(\lambda) \in n(P_2')$. As usual the interesting case is when $H_{L_i}(\lambda) \in fn(P_2')$. Given the previous conditions we have $fn(P_2') \subseteq n(S_1')$. Therefore, $H_{L_i}(\lambda) \in fn(P_2')$ implies $H_{L_i}(\lambda) \in n(S_1')$ which contradicts (b).

Assume that $\lambda \in A(S_2')$. Given the previous conditions we have two possibilities: either $\lambda \in A(S_2')$ or $n_2 \in U(P)$. The former case contradicts immediately (a). In the latter case, we use $P_2 \triangleright P_2'$, which says that $P'_2$ has no unguarded and unnecessary restrictions ($U_0(P_2') = \emptyset$). Consequently, when $n_2 \in U(P)$, then $H_{L_i}(\lambda) \in n(S_1')$. This contradicts condition (b).

We now show that there exists a reduction $P_1 \mid P_2 \rightarrow Q$, where $Q = P'_1 \mid P_2'$, we observe that $A(P_2') \subseteq A(P_2)$, and thus $P_1 \mid P_2$ is well-labeled. Since also $P'_1 \mid P_2'$ is well-labeled, by applying rule (Par) to the premise $P_1 \mid P_2'$, we obtain $P_1 \mid P_2' \rightarrow P'_1 \mid P_2'$, since $P_2 \gg P_2'$ we have also $P_1 \mid P_2 \gg P_1 \mid P_2'$. We therefore derive $P_1 \mid P_2 \rightarrow P'_1 \mid P_2'$ by applying rule (Cong).

Moreover, it is immediate to check that

$$
\delta^a P_1' \mid P_2' = S_1' \cup S_2' = S_2.
$$

It remains to show that $A(Q) \setminus A(P) \subseteq \text{new}(S_1 \rightarrow S_2)$. We observe that, since $P_1 \mid P_2$ and $P'_1 \mid P'_2$ are well-labeled, $A(P_1) \cap A(P_2') = \emptyset$ and $A(P_1') \cap A(P_2') = \emptyset$. Moreover, $A(P_2') \subseteq A(P_2)$.

Therefore, $A(Q) \setminus A(P) = (A(P_1') \cup A(P_2')) \setminus (A(P_1) \cup A(P_2)) = A(P_1') \setminus (A(P_1) \cup A(P_2)) \subseteq \text{new}(S_1 \rightarrow S_2)$. Therefore, we have a reduction $P_1 \mid P_2 \rightarrow Q$. This concludes the proof of Proposition A.16.
Lemma A.18. Let the relation between the states representing two well-labeled processes which are structural congruent.

Completeness. To show completeness we need some auxiliary properties. The following lemma shows the relation between the states representing two well-labeled processes which are structural congruent.

Lemma A.18. Let $P$ and $Q$ be well-labeled processes and let $a \in \mathcal{A}$, such that $a$ is fresh for $P$ and $Q$. If $P \succ Q$, then either $\delta^a P = \delta^a Q$ or $\delta^a P \mapsto \delta^a Q$.

Proof. By induction on the depth of $P \succ Q$. It is easy to check that in any case of Table A.2 the states obtained via $\delta$ are equal apart from the case (Bang-Bang). In case (Bang-Bang) we have $P = !R$ and
$Q = \bang R \mid \new(R)$. Hence, we have $\delta aP = S_1 = (\emptyset, a!R)$ and $\delta aQ = S_2 = (\emptyset, a!R) \cup \delta a\new(R)$. We observe that $\delta aP \mapsto S_2$ by rule $\bang$. □

**Proposition A.19.** Let $S_1$ be a well-labeled state such that $S_1 \mapsto S_1'$. If $S_2$ is a well-labeled state such that $S_1 \cup S_2$ and $S_1' \cup S_2$ is well-labeled, then we have also $S_1 \cup S_2 \mapsto S_1' \cup S_2$.

**Proof.** The proof is by cases on the rule applied to derive $S_1 \mapsto S_1'$. The cases of $\in$, $\out$, and $\open$ are trivial; the side conditions impose constraints which hold also for $S_1 \cup S_2$. In the case $\bang$ instead we have $S_1 = (T_1, C_1)$ and $S_1' = (T_1, C_1) \cup \delta a\new(Q)$ for some $\bang Q \in C_1$. We obtain $S_1 \cup S_2 \mapsto S_1' \cup S_2$, as $\new(Q) = \new(S_1) \cup \new(C_1) \cup \bang$. □

**Lemma A.20.** Let $P$ be a well-labeled process such that $P \mapsto Q$. For any $c \in A$ which is fresh for $P$, we have $\delta cP \mapsto^* \delta cQ$.

**Proof.** The proof is by induction on the depth of the derivation of $P \mapsto Q$. The last rule used could have been $\in$, $\out$, or $\open$, one of the structural rules $(\res)$, $(\par)$, $(\amb)$ or rule $(\cond)$.

- Assume that $P \mapsto Q$ has been obtained by applying rule $(\in)$. It means that $P = a[\in m_\lambda].P' \mid Q'$, where $a = n_\mu$ and $b = m_\gamma$, and $Q = b[a[P' \mid Q'] \mid R'$.

  By definition of $\delta$ we have

  \[
  \delta cP = \delta c[a[\in m_\lambda].P' \mid Q'] \cup \delta c\bang[R'] = \{a, b\} \cup \emptyset \cup (\emptyset, a[\in m_\lambda].P') \cup \delta aQ' \cup \delta bR'.
  \]

  Therefore, by applying rule $\in$ we obtain a transition $\delta cP \mapsto S$, where

  \[
  S = \{a, b\} \cup \delta aQ' \cup \delta bR' \cup \delta aP'.
  \]

- We conclude by observing that, by definition of $\delta$,

  \[
  \delta cQ = \{a, b\} \cup \delta b[a[P' \mid Q'] \mid R']
  = \{a, b\} \cup \delta aP' \cup \delta aQ' \cup \delta bR' = S.
  \]

- Assume that $P \mapsto Q$ has been obtained by applying rule $(\out)$. It means that $P = b[a[\out m_\lambda].P' \mid Q'] \mid R'$, where $a = n_\mu$ and $b = m_\gamma$, and $Q = b[R'] \mid a[P' \mid Q']$.

  By definition of $\delta$ we have

  \[
  \delta cP = \{a, b\} \cup \delta b[a[\out m_\lambda].P' \mid Q'] \cup R'
  = \{a, b\} \cup \emptyset \cup (\emptyset, a[\out m_\lambda].P') \cup \delta aQ' \cup \delta bR'.
  \]

  Moreover, by applying rule $\out$ we obtain a transition $\delta cP \mapsto S$, where

  \[
  S = \{a, b\} \cup \delta aP' \cup \delta aQ' \cup \delta bR'.
  \]

- We conclude by observing that, by definition of $\delta$,

  \[
  \delta cQ = \delta c\bang[R'] \cup \delta c[a[P' \mid Q'] \mid R'] = \{a, b\} \cup \delta aP' \cup \delta aQ' \cup \delta bR'.
  \]

- Assume $P \mapsto Q$ has been obtained by applying rule $(\open)$. It means that $P = \open n_\lambda. P' \mid a[R']$, where $a = n_\mu$, and $Q = P' \mid R'$.

  By definition of $\delta$ we have

  \[
  \delta cP = \delta c\open n_\lambda. P' \cup \delta c[a[R'] \mid Q'] \cup (\emptyset, \open n_\lambda. P') \cup \delta aR'.
  \]

  Moreover, by applying rule $\open$ we obtain a transition $\delta cP \mapsto S$, where

  \[
  S = \{a, b\} \cup \delta aP' \cup \delta aQ' \cup \delta bR'.
  \]
\[ S = \delta \cdot P' \cup (T[\ d' / \ d'], C[\ cR / \ aR]) \]

\[ \delta \cdot aR' = (T, C) \]

We conclude by observing that, by definition of \( \delta \),

\[ \delta \cdot Q = \delta \cdot P' \cup \delta \cdot R'. \]

Since \( c \) is fresh, we also have \((T[\ d' / \ d'], C[\ cR / \ aR]) = \delta \cdot R'. \) We therefore conclude \( \delta \cdot Q = S \).

- Assume \( P \rightarrow Q \) has been obtained by applying rule (Amb). It means that \( P = a[P_1] \), where \( a = n_\lambda \), and \( Q = a[P_2] \), where \( P_1 \rightarrow P_2 \). By definition of \( \delta \) we have

\[ \delta \cdot P = \delta \cdot a[P_1] = (a^c, \emptyset) \cup \delta \cdot aP_1. \]

Since \( P \) is well-labeled, then \( P_1 \) is well-labeled and \( a \) is fresh for \( P_1 \). Hence, by induction hypothesis we have \( \delta \cdot aP_1 \mapsto S' \), where \( \delta \cdot aP_2 = S' \). We now observe that \( Q = a[P_2] \) is well-labeled. Hence, by Proposition A.13, we have that \( \delta \cdot Q \) is well-labeled. Also, by definition of \( \delta \) we have

\[ \delta \cdot Q = (a^c, \emptyset) \cup \delta \cdot aP_2 = (a^c, \emptyset) \cup S'. \]

We conclude by applying Proposition A.19. Since \((a^c, \emptyset) \cup S' \) is well-labeled and \( \delta \cdot aP_1 \mapsto S' \), then we have also

\[ \delta \cdot P \mapsto (a^c, \emptyset) \cup S'. \]

- Assume \( P \rightarrow Q \) has been obtained by applying rule (Par). It means that \( P = P_1 \ | \ P_2 \) and \( Q = P'_1 \ | \ P_2 \), where \( P_1 \rightarrow P'_1 \). By definition of \( \delta \) we have

\[ \delta \cdot P = \delta \cdot P_1 \cup \delta \cdot P_2. \]

Since \( P \) is well-labeled and \( c \) is fresh for \( P \), then also \( P_1 \) is well-labeled and \( c \) is fresh for \( P_1 \). Hence, by induction hypothesis we have \( \delta \cdot P_1 \mapsto S' \), where \( \delta \cdot P_1 = S' \).

We now observe that \( Q = P'_1 \ | \ P_2 \) is well-labeled. Hence, by Proposition A.13, we have that \( \delta \cdot Q \) is well-labeled. Also, by definition of \( \delta \) we have

\[ \delta \cdot Q = \delta \cdot P'_1 \cup \delta \cdot P_2 = S' \cup \delta \cdot P_2. \]

We conclude by applying Proposition A.19. Since \( S' \cup \delta \cdot P_2 \) is well-labeled and \( \delta \cdot P_1 \mapsto S' \), then we have also

\[ \delta \cdot P \mapsto S' \cup \delta \cdot P_2. \]

- Assume \( P \rightarrow Q \) has been obtained by applying rule (Res). It means that \( P = (vn_\lambda) \ P_1 \) and \( Q = (vn_\lambda) \ P_2 \), where \( P_1 \rightarrow P_2 \). By definition of \( \delta \) we have

\[ \delta \cdot P = \delta \cdot (P_1[m/n]), \]

where \( m = Hc_{ij}(\lambda) \).

We observe that since \( P \) is well-labeled, then \( m \notin n(P_1) \). Since \( P_1 \rightarrow P_2 \), then by Proposition A.7, \( fn(P_2) \subseteq fn(P_1) \), and consequently also \( m \notin n(P_2) \). Considering the bound names can be \( \alpha \)-converted, if needed, we derive \( P_1[m/n] \rightarrow P_2[m/n] \) from \( P_1 \rightarrow P_2 \).

Since \( P \) is well-labeled, then \( \lambda \notin A(P_1[m/n]) \). Consequently, \( P_1[m/n] \) is a well-labeled process. By induction hypothesis we have \( \delta \cdot (P_1[m/n]) \mapsto S' \), where \( \delta \cdot (P_2[m/n]) = S' \). We conclude by observing that

\[ \delta \cdot Q = \delta \cdot (P_2[m/n]) = S'. \]
• Assume $P \rightarrow Q$ has been obtained by applying rule (Cong). It means that $P_1 \rightarrow Q_1$ for some processes $P_1, Q_1$, such that $P \gg P_1$ and $Q_1 \gg Q$. By induction hypothesis we have $\delta^c P_1 \mapsto^* S$, where $S = \delta^c Q_1$. By Lemma A.18, we have $\delta^c P \mapsto^* S$. Again by Lemma A.18, we have either $\delta^c Q_1 = \delta^c Q$ or $\delta^c Q_1 \mapsto \delta^c Q$. In both cases $\delta^c P \mapsto^* \delta^c Q$. □

A.3. Equivalence

We show the proof of Theorem 4.5.

Soundness: if $\delta^a P \mapsto Q$, then by Lemma A.17 there exists a well-labeled process $Q$, such that $\delta^a Q = S$ and $P \gg Q$. By Lemmas A.1 and A.2 we have $E(P) \mapsto E(Q)$.

Completeness: if $E(P) \rightarrow Q$, then by Lemma A.9 there exists a well-labeled process $Q'$, such that $E(Q') = Q$ and $P \rightarrow Q'$. By Lemma A.20 we have $\delta^a P \mapsto^* \delta^a Q'$.

B. Safeness of the abstractions

The following proposition recalls some well-known results of domain theory which are useful in the proofs.

Proposition B.1.
(1) Given any set $S$, $(\wp(S), \subseteq)$ is a complete lattice.
(2) Given two complete lattices $(S_1, \subseteq_1)$, $(S_2, \subseteq_2)$, the product $(S_1 \times S_2, \subseteq_{cw})$, where $\subseteq_{cw}$ is the component-wise induced ordering, is a complete lattice.

B.1. First abstraction

We first show that the pair of functions $(\alpha^\diamond, \gamma^\diamond)$ forms a Galois connection between $(S^\flat, \subseteq)$ and $(S^\flat, \subseteq^\diamond)$ (Theorem 5.6).

Proposition B.2.  
(i) The concrete domain $(S^\flat, \subseteq)$ is a complete lattice; (ii) the abstract domain $(S^\flat, \subseteq^\diamond)$ is a complete lattice.

Proof. (i) The concrete domain $S^\flat = \wp(S/\sim)$ is a complete lattice by case (1) of Proposition B.1. (ii) The abstract domain $(S^\flat, \subseteq^\diamond)$ is a complete lattice by case (2) of Proposition B.1. Notice that, by definition of $\subseteq^\diamond$ (Definition 5.3), given two well-labeled states $S_1^\flat$ and $S_2^\flat$, $S_1^\flat \cup S_2^\flat$ is a well-labeled state as well. □

The following proposition states the basic properties of the concretisation and abstraction functions.

Proposition B.3. Function $\alpha^\flat : (S^\flat, \subseteq) \rightarrow (S^\flat, \subseteq^\diamond)$ is monotonic and continuous, and function $\gamma^\flat : (S^\flat, \subseteq^\diamond) \rightarrow (S^\flat, \subseteq)$ is monotonic.

Proof. Straightforward by Definition 5.5. □

The properties stated above are enough to prove Theorem 5.6.
Proof (of Theorem 5.6). We show that \((\alpha^\circ, \gamma^\circ)\) is a Galois connection (see Definition 2.1). By Proposition B.2 the concrete and abstract domains are complete lattices. Also, by Proposition B.3 both \(\alpha^\circ\) and \(\gamma^\circ\) are monotonic. Hence, it remains two show that, for \(S^2 \in S^2\) and \(S^2 \in S^2\), we have

\[ S^2 \subseteq \gamma^\circ(\alpha^\circ(S^2)), \]

\[ \alpha^\circ(\gamma^\circ(S^2)) \subseteq S^2. \]

Both assertions follow rather obviously from Definition 5.5. We have \(S^2 \subseteq \gamma^\circ(\alpha^\circ(S^2))\) since by definition of \(\gamma^\circ\) and \(\alpha^\circ\),

\[ \gamma^\circ(\alpha^\circ(S^2)) = \bigcup \{ [S] \mid \alpha^\circ([S]) \subseteq \gamma^\circ \bigcup_{\{S\} \in S^2} \alpha^\circ([S]) \}. \]

Moreover, by definition of \(\alpha^\circ\) and \(\gamma^\circ\), and by continuity of \(\alpha^\circ\) (Proposition B.3) we have

\[ \alpha^\circ(\gamma^\circ(S^2)) = \alpha^\circ \big( \bigcup \{ [S] \mid \alpha^\circ([S]) \subseteq \gamma^\circ \bigcup_{\{S\} \in S^2} \alpha^\circ([S]) \} \big) \]

\[ = \bigcup \alpha^\circ \big( \{ [S] \mid \alpha^\circ([S]) \subseteq \gamma^\circ \bigcup_{\{S\} \in S^2} \alpha^\circ([S]) \} \big). \]

By definition of least upper bound on a complete lattice we conclude therefore

\[ \bigcup \alpha^\circ \big( \{ [S] \mid \alpha^\circ([S]) \subseteq \gamma^\circ \bigcup_{\{S\} \in S^2} \alpha^\circ([S]) \} \big) \subseteq S^2. \]

We now show some basic properties of the concrete and abstract semantic functions which are needed to establish the safeness of the abstraction (Lemma 5.8).

Lemma B.4. Let \(S^1, S^1 \in S^2\) be well-labeled abstract states such that \(S^1 \preceq S^2\). If \(S^1 \rightarrow S^2\), then there exists a transition \(S^1 \rightarrow S^2\), such that \(S^1 \preceq S^2\).

Proof. There are two cases depending on whether \(S^1 \subseteq S^2\) or not. In the former case the proof is straightforward. In the latter case, it means that there exists an abstract state \(S^1\), such that \(S^1 \rho S^2\) for a renaming \(\rho : L_1 \rightarrow L_2\), where either \(\rho(\ell_1) = \ell_1\) or \(\rho(\ell_1) = \ell_{\omega}\), and \(S^1 \subseteq S^2\). It is easy to check by cases on the rules of Table 7) that we have \(S^1_1 \rightarrow S^2_1\) such that \(S^2_1 \subseteq S^1_2\). Since \(S^1 \subseteq S^2\) and \(S^1_1 \rightarrow S^2_2\), then we have also \(S^1 \rightarrow S^2\) such that \(S^1 \subseteq S^2\). We conclude because \(S^1 \subseteq S^2\).

Lemma B.5. Let \(S \in S\) and \(S^2 \in S^2\). The functions \(\Psi_S : (S^2, \subseteq) \rightarrow (S^2, \subseteq)\) and \(\Psi^\circ_S : (S^2, \subseteq) \rightarrow (S^2, \subseteq)\) are monotonic.

Proof. The proof follows immediately by Lemma B.4 using Definitions 4.9 and 5.7.

We state some relevant properties of the auxiliary abstraction function \(\alpha^\circ : S \rightarrow S^2\) which maps a state into an abstract state (see Definition 5.5, case (1)). The following lemma says that \(\alpha^\circ\) is continuous for union of states with a special shape (recall that the abstraction over sets of states \(\alpha^\circ : S^2 \rightarrow S^2\) is continuous as shown by Proposition B.3). To state formally this result we need to introduce an auxiliary concept. Let \(S_1, S_2\) be two well-labeled states. We say that \(S_2\) is a sub-tree of \(S_1\) with root \(a \in A\) if \(a\) is the root of \(S_2\), and only ambient \(a\) occurs both in \(S_1\) and \(S_2\).
We introduce a convention which is useful in the following proofs. We recall that any object may have several abstractions depending on the global number of occurrences of its labels in the state. In the abstraction $\alpha^\circ$ (see Definition 5.5) this is formalised by: the renaming $\rho_S^\circ$, which depend on the state $S$ and introduce the multiplicity counting the indexes; the substitution $\eta^\circ$ which simply removes indexes. When the renaming $\rho_S^\circ$ is clear from the context we may use: $a^\circ$ to denote the abstract version of $a$; $P^\circ$ to denote that abstract version of $P$.

**Lemma B.6.** Let $S_1, S_2$ be two well-labeled states, such that $S_1 \cup S_2$ also is well-labeled. If $S_2$ is a sub-tree of $S_1$ with root $a$, then we have

$$a^\circ(S_1 \cup S_2) = a^\circ(S_1) \cup a^\circ(S_2)[a^b/a^@]$$

where $b$ is the father of $a$ in $S_1$.

**Proof.** Let $\alpha^\circ(S_1 \cup S_2) = (T^\circ, C^\circ), S_i = (T_i, C_i)$ and $\alpha^\circ(S_i) = (T_i^\circ, C_i^\circ)$ for $i \in \{1, 2\}$. We recall that, by definition of $\alpha^\circ$ (Definition 5.5), we have $(T^\circ, C^\circ) = (T^\circ_1, C^\circ_1) \rho_{S_1 \cup S_2}^{\circ} \eta^\circ$, where

$$T^\circ = \{ a^b \mid a^b, b^c \in T_1 \cup T_2 \}$$

$$C^\circ = \{ a^b P \mid a^b \in T_1 \cup T_2, a^P \in C_1 \cup C_2 \}.$$  

Analogously, for $i \in \{1, 2\}$, we have $(T_i^\circ, C_i^\circ) = (T_i^\circ_1, C_i^\circ_1) \rho_{S_i}^{\circ} \eta^\circ$ where

$$T_i^\circ = \{ a^b \mid a^b, b^c \in T_i \}$$

$$C_i^\circ = \{ a^b P \mid a^b \in T_i, a^P \in C_i \}.$$  

We first show that $T_i^\circ \subseteq T_1^\circ \cup T_2^\circ[a^b/a^@]$. Let us consider a generic element $c^{de} \in T_i^\circ$. It means that $c^d, d^e \in T_i \cup T_2$. There are several possibilities:

1. Both $c^d \in T_1$ and $d^e \in T_1$. It is immediate to check that we have also $c^{de} \in T_1^\circ$.
2. Both $c^d \in T_2$ and $d^e \in T_2$. Similarly as in the previous case we have $c^{de} \in T_2^\circ$. We now observe that $c$ and $d$ cannot be $a$, because $a$ is the root of $S_2$. We therefore conclude that $c^{de} \in T_2^\circ[a^b/a^@]$.
3. One element belongs to $T_1$ and the other one to $T_2$. Since $S_2$ is a sub-tree of $S_1$ with root $a$ the only possibility is that $c^d \in T_2, d^e \in T_1$ and $d = a$. Moreover, since $b$ is the father of $a$ in $S_1$, it means that $e = b$. It is immediate to check that $c^{de} \in T_2^\circ$, so that $c^{de} \in T_2^\circ[a^b/a^@]$.

We now show the converse $T_i^\circ \supseteq T_1^\circ \cup T_2^\circ[a^b/a^@]$. Let us consider a generic element $c^{de} \in T_1^\circ \cup T_2^\circ[a^b/a^@]$. There are two possibilities:

1. If $c^{de} \in T_1^\circ$, then both $c^d, d^e \in T_1$. It follows that both $c^d, d^e \in T_1 \cup T_2$, and thus $c^{de} \in T_1^\circ$.
2. If $c^{de} \in T_2^\circ[a^b/a^@]$, then either $c^{de} \in T_2^\circ$ or $d = a, e = b$ and $c^{de} \in T_2^\circ$. The former case is analogous to (1) above. In the latter case we observe that $c^a \in T_2$. Since $a^b \in T_1$, we have $c^a, a^b \in T_1 \cup T_2$, and thus $c^{de} \in T_1^\circ$.

A similar argument applies also to the configuration. Hence, we have

$$(T^\circ, C^\circ) = (T_1^\circ, C_1^\circ) \cup (T_2^\circ, C_2^\circ)[a^b/a^@].$$

---

16 Meaning that $a^b \in T_1$ for $S_1 = (T_1, C_1)$. 
Therefore, we have also
\[(T^{\circ}, C^{\circ})\rho_{S_1 \cup S_2}^\circ \eta^\circ = (T_1^{\circ}, C_1^{\circ})\rho_{S_1 \cup S_2}^\circ \eta^\circ \cup (T_2^{\circ}, C_2^{\circ})\|a^b/a^\circ \| \rho_{S_1 \cup S_2}^\circ \eta^\circ .\]

Using \(a^\circ = a\rho_{S_1 \cup S_2}^\circ \eta^\circ\) and \(b^\circ = b\rho_{S_1 \cup S_2}^\circ \eta^\circ\), we obtain
\[(T^{\circ}, C^{\circ})\rho_{S_1 \cup S_2}^\circ \eta^\circ = (T_1^{\circ}, C_1^{\circ})\rho_{S_1 \cup S_2}^\circ \eta^\circ \cup (T_2^{\circ}, C_2^{\circ})\rho_{S_1 \cup S_2}^\circ \eta^\circ\|a^{\circ b}/a^{\circ \circ} \| .\]

Now we observe that the equality is preserved, when the renamings \(\rho_{S_i}^\circ\) are used for \(i \in \{1, 2\}\) in place of \(\rho_{S_1 \cup S_2}^\circ\) and \(\cup\) is replaced by \(\cup^\circ\). This because \(\cup^\circ\) modifies the multiplicity counting the number of the occurrences of the union. Therefore, we conclude
\[(T^{\circ}, C^{\circ})\rho_{S_1 \cup S_2}^\circ \eta^\circ = (T_1^{\circ}, C_1^{\circ})\rho_{S_1}^\circ \eta^\circ \cup (T_2^{\circ}, C_2^{\circ})\rho_{S_2}^\circ \eta^\circ\|a^{\circ b}/a^{\circ \circ} \| .\]

The following proposition shows the safeness of the abstract normalisation function \(\delta^\circ\). Notice that \(a\) is the root of \(\delta^\circ aP\) so that the abstraction \(a^\circ\) assigns \(\circ\) as father of \(a^\circ\). It is therefore necessary to replace \(\circ\) with \(b^\circ\).

**Proposition B.7.** Let \(P\) be a well-labeled process and \(a \in A\) such that \(a\) is fresh for \(P\). We have
\[a^\circ(\delta^\circ aP)\|a^{\circ b}/a^{\circ \circ} \| \subseteq a^\circ \delta^\circ a^\circ P^\circ .\]

**Proof.** The proof proceeds by induction on the structure of \(P\) using the definition of \(\delta^\circ\) (Table 6).

We show the most interesting cases.

- Assume that \(P = c[P_1]\). We have
  \[\delta^\circ aP = ([c^d], \emptyset) \cup \delta^\circ cP_1 .\]

  By Proposition A.13, \(\delta^\circ aP\) is well-labeled state. Moreover, we observe that \(\delta^\circ cP_1\) is a sub-tree of \(([c^d], \emptyset)\) with root \(c\). Thus, by Lemma B.6 we have
  \[a^\circ(\delta^\circ aP) = a^\circ(([c^d], \emptyset)) \cup \delta^\circ a^\circ(\delta^\circ cP_1)\|c^{\circ a}/c^{\circ \circ} \|.\]

  By induction hypothesis we have
  \[a^\circ(\delta^\circ cP_1)\|c^{\circ a}/c^{\circ \circ} \| \subseteq a^\circ \delta^\circ c^{\circ a} P_1^\circ .\]

  Moreover, by definition of \(a^\circ\) we have \(a^\circ([c^d], \emptyset) = ([c^{\circ a\circ}], \emptyset)\). Therefore, we have
  \[a^\circ(\delta^\circ aP) \subseteq a^\circ([c^{\circ a\circ}], \emptyset) \cup \delta^\circ \delta^\circ c^{\circ a} P_1^\circ .\]

  We now observe that the replacement \(\|a^{\circ b}/a^{\circ \circ} \|\) cannot affect \(\delta^\circ c^{\circ a} P_1^\circ\) because the abstract topology of a single state is a tree. Therefore, we have
  \[a^\circ(\delta^\circ aP)\|a^{\circ b}/a^{\circ \circ} \| \subseteq a^\circ([c^{\circ a\circ}], \emptyset)\|a^{\circ b}/a^{\circ \circ} \| \cup \delta^\circ \delta^\circ c^{\circ a} P_1^\circ .\]

  We conclude because by definition of \(\delta^\circ\) we have
  \[\delta^\circ a^{\circ b} P^\circ = ([c^{\circ a\circ}], \emptyset) \cup \delta^\circ c^{\circ a} P_1^\circ .\]

- Assume that \(P = (\nu n_\lambda P_1)\). We have
  \[\delta^\circ aP = \delta^\circ (P_1[\hat{n}_i/n]) .\]

  where \(\hat{n}_i = H_{L_i}(\lambda)\) and \(\lambda = \ell_1\).
Since $P$ is well-labeled, then $\lambda \notin A(P_1)$. Therefore, $P_1[\tilde{n}_1/n]$ is well-labeled, and by induction hypothesis we have
\[
\alpha^o(\delta^\omega(P_1[\tilde{n}_1/n])) [a^{\omega}b^\omega a_\omega^\omega] \subseteq \delta^o a^{\omega} \alpha^o(P_1[\tilde{n}_1/n]).
\]

Let $P^o = \alpha^o(P) = (\nu n_\omega^\omega) P_1^o$. By definition of $\alpha^o$ we have $\alpha^o(P_1[\tilde{n}_1/n]) = P_1^o[\tilde{n}/n]$. Now we use $H_{\omega}\omega(\lambda^\omega) = \tilde{n}$ and we obtain by definition of $\delta^o$
\[
\delta^o a^{\omega}P^o = \delta^o a^{\omega} P_1^o[\tilde{n}/n].
\]
\[
\square
\]

The following lemma is the core of the proof of safeness; it states the agreement between concrete and abstract transitions.

**Lemma B.8.** Let $S, S' \in S$ be well-labeled states. For any $S \mapsto S'$ there exists an abstract state $S'^o$, such that $\alpha^o(S) \mapsto \alpha^o(S') \subseteq S'^o$.

**Proof.** The proof is by cases on the rule applied to obtain the transition $S \mapsto S'$. One of the rules Bang, In, Out and Open of Table 5 could have been applied. Assume that $S = (T, C)$, by definition of $\alpha^o$ (Definition 5.5), we have $\alpha^o(S) = (T^o, C^o) = (T^o, C^o) \rho_\omega^o \eta^o$ where
\[
T^o = \{ b^f \mid a, b \in T \}
\]
\[
C^o = \{ a^P \mid a^P \in T, \ a^P \in C \}.
\]

As usual we use $a^o$ to denote the abstract version of $a$, that is $a^o = \eta^o$. Similarly, for the other ambients and processes.

**Bang** It means that $a^o P \in C$ and that
\[
S' = S \uplus a^o new_S(P).
\]

By definition of $\alpha^o$ we have $a^{\omega}b^\omega P^o \in C^o$, where $b$ is the father of $a$ in $S$, i.e., either $a^b \in T$, or $a$ is the root of $T$ and $b = \ominus$. Hence, by applying rule Bang, we obtain a transition $\alpha^o(S) \mapsto \alpha^o(S')'$ where
\[
\delta^o \omega = \alpha^o(S) \cup \delta^o a^{\omega} \omega new_\omega(P^o).
\]

It remains to show that $\alpha^o(S') \subseteq \alpha^o(S')$, that is
\[
\alpha^o(S) \cup \delta^o a^o new_\omega(S(P)) \subseteq \alpha^o(S) \cup \delta^o a^{\omega} \omega new_\omega(P^o).
\]

We observe that $\delta^o a^o new_\omega(S(P))$ is a sub-tree of $S$ with root $a$. Hence, by Lemma B.6, we have
\[
\alpha^o(S \cup \delta^o a^o new_\omega(S(P))) = \alpha^o(S) \cup \alpha^o(\delta^o a^o new_\omega(S(P))) [a^{\omega} / a^{\omega}].
\]

By Proposition B.7 we have also
\[
\alpha^o(\delta^o a^o new_\omega(S(P))) [a^{\omega} / a^{\omega} \omega] \subseteq \delta^o a^{\omega} \omega new_\omega(P^o).
\]

We now observe that the function $new_\omega$ gives multiplicity $\omega$ to any label of $P^o$. It means that
\[
\delta^o a^{\omega} \omega new_\omega(P^o) \subseteq \delta^o a^{\omega} \omega new_\omega(P^o).
\]
We therefore conclude
\[ \alpha^\circ(\delta^a new_\mu(S)) \parallel a^b \cap \alpha^\circ \cap_\mu \subseteq \delta^a a^b new_\mu(P^\circ). \]

It means that \( a^b, m^\mu b \in T \) and \( t = a^b m^\mu b, P \in C \), where \( a \neq m^\mu b \) and \( a \neq \emptyset \). Moreover,
\[
S' = \delta^a P \cup ((T \setminus \{ a^b \}) \cup \{ a^m^\mu b \}, C \setminus \{ t \}).
\]

By definition of \( \alpha^\circ \) we have \( a^b \cap m^\mu b \in C^\circ \), since \( b \) is the father of \( a \) (\( a^b \in T \)). Moreover, it is immediate to check that there exists \( c^\circ \) such that \( a^b c^\circ, m^\mu b c^\circ \in T^\circ \). Notice that, since \( a \neq \emptyset \), either \( c \) is the father of \( b \) in \( T \), or \( b = \emptyset \) and \( c^\circ = \top \), or \( b \) is the root of \( T \) and \( c^\circ = \emptyset \).

We now observe that the side condition of rule \( \text{In}^\circ \) is satisfied (if \( a^\circ = m^\beta \ell_1 \) then \( \mu^\circ = \ell_1 \)). Since \( a \neq m^\mu \), there are two cases: either \( a = k_\lambda \) or \( a = m^\lambda \) with \( \lambda = \neq \mu \). In the former case the side condition is immediately satisfied. In the latter case it depends on whether \( \lambda \) and \( \mu \) differ in the indexes only. In particular, when \( \lambda = \ell_j \) and \( \mu = \ell_h \) for indexes \( j, h \), such that \( j \neq h \), the side condition is satisfied, because \( \lambda^\circ = \mu^\circ = \ell_\omega \) by definition of the abstraction.

By applying rule \( \text{In}^\circ \), we obtain a transition \( \alpha^\circ(S) \rightarrow S^\circ \) where
\[
S^\circ = \delta^a a^m^\mu b \cap P^\circ \cup S^\circ_2
\]

\[
S^\circ_2 = \alpha^\circ(S) \cup \cap_\mu (T^\circ \cup \{ a^m^\mu b \}, C^\circ \setminus \{ t^\circ \}) \parallel a^m^\mu b \cap \alpha^\circ \cap_\mu \cup S^\circ_1
\]

It remains to show that \( \alpha^\circ(S') \subseteq S^\circ \), that is \( \alpha^\circ(\delta^a P \cup ((T \setminus \{ a^b \}) \cup \{ a^m^\mu b \}, C \setminus \{ t \})) \subseteq S^\circ \).

We observe that \( \delta^a P \) is a sub-tree of \( S' \) with root \( a \) and that the father of \( a \) in \( S' \) is \( m^\mu \). Hence, by Lemma B.6, we have
\[
\alpha^\circ(\delta^a P) = \alpha^\circ(\delta^a P) \parallel a^m^\mu b \cap \alpha^\circ \cap_\mu \cup S^\circ_1
\]

\[
S^\circ_1 = \alpha^\circ((T \setminus \{ a^b \}) \cup \{ a^m^\mu b \}, C \setminus \{ t \}).
\]

By Proposition B.7, we have also
\[
\alpha^\circ(\delta^a P) \parallel a^m^\mu b \cap \alpha^\circ \cap_\mu \subseteq \delta^a a^m^\mu b \cap P^\circ.
\]

Hence, to conclude it is enough to show that \( S^\circ_1 \subseteq S^\circ_2 \), that is \( \alpha^\circ((T \setminus \{ a^b \}) \cup \{ a^m^\mu b \}, C \setminus \{ t \}) \subseteq \alpha^\circ((T \setminus \{ a^b \}) \cup \{ a^m^\mu b \}, C \setminus \{ t \}) \parallel a^m^\mu b \cap \alpha^\circ \cap_\mu \).

In the following we assume that \( S^\circ_i = (T^\circ, C^\circ) \) for \( i \in \{ 1, 2 \} \) (we recall also that \( \alpha^\circ(S) = (T^\circ, C^\circ) \subseteq (T^\circ, C^\circ) \cap_\mu \)).

\textbf{We show} \( T^\circ_1 \subseteq T^\circ_2 \). Let \( d^{e^\circ} \in T^\circ_1 \), by definition of \( \alpha^\circ \) we have \( d^e, e^f \in (T \setminus \{ a^b \}) \cup \{ a^m^\mu b \} \). There are several cases to consider depending on how the ambient \( a \), whose father has changed, is involved.

Assume that none of \( d^e \) and \( e^f \) is equal to \( a \). It is easy to check that \( d^e, e^f \in T \). Hence, we have \( d^{e^f} \in T^\circ \) and, consequently, also \( d^{e^f} \in T^\circ_2 \).

Assume that \( d^e = a \). Since \( m^\mu b \in (T \setminus \{ a^b \}) \cup \{ a^m^\mu b \} \) then \( e^f = m^\mu b \) and \( f^e = b^f \).

We conclude because \( a^m^\mu b \in T^\circ_2 \).
Assume that $a^\circ = a^\circ$. It means that $f^\circ = m_\mu^a$ as $a^m_\mu \in (T \setminus \{ b^\circ \}) \cup \{ a^m_\mu \}$. Therefore, we have $d^a, b \in T$ and $d^a \in T^\circ$. Moreover, we have $d^a \in T^\circ \{ a^m_\mu \}$ and, consequently, $a^m_\mu \in T^\circ$.

We show $C_1 \subseteq C_2^\circ$ by considering a generic element $a^\circ Q \in C_1^\circ$. The proof is similar to the one shown for the topology; the only interesting case is when the process $Q$ is local to $a$, that is $d^\circ = a^\circ$ and $e^\circ = m_\mu^a$. By definition of $a^\circ$ it means that $aQ \in C \setminus \{ t \}$ and $a^m_\mu \in (T \setminus \{ b^\circ \}) \cup \{ a^m_\mu \}$, and consequently $aQ \in C$ and $a^b \in T$. By definition of $a^\circ$ we obtain $a^\circ Q \in C^\circ$. Now, we use the definition of $a^\circ$; there are two cases depending on whether the label $\gamma$ is either $\ell_1^a$ or $\ell_\omega$.

When $\gamma = \ell_1^a$, we have $C^\circ \setminus \{ t_\gamma \} = C^\circ$. Since $a^\circ Q \in C^\circ$ then we conclude $a^\circ Q \in C^\circ \setminus \{ t_\gamma \}$ by applying rule In above.

When $\gamma = \ell_\omega$, we have $C^\circ \setminus \{ t_\gamma \} = C^\circ \setminus \{ t_\gamma \}$. We observe that it cannot be the case that $Q^\circ = t^\circ$, as $\gamma = \ell_1$ shows that there is only object with label $\gamma$. Therefore, we have $a^\circ Q \in C^\circ \setminus \{ t_\gamma \}$. We then conclude as before.

**Out** Similar to the case of rule In above.

**Open** It means that $m_\mu^a \in T$ and $t = \\text{open}_m \gamma$, $P \in C$, where $a \neq m_\mu$.

Moreover,

$$S' = \delta^a P \cup ((T \setminus \{ m_\mu^a \}), (C \setminus \{ t \})) \{ a/m_\mu \}$$

By definition of $a^\circ$ we have $a^\circ Q \in C^\circ$, where $b$ is the father of $a$ in $T$, i.e., either $a^b \in T$ or $a$ is the root of $T$ and $b = \emptyset$. Moreover, since $m_\mu^a \in T$ we have also $a^\circ Q \in T^\circ$.

We observe that the side condition of rule Open is satisfied since $a \neq m_\mu$ (by applying a reasoning similar to that for In). Hence, by applying rule Open, we obtain a transition $a^\circ (S) \rightarrow S^\circ$ where

$$S^\circ = \delta^a P \cup a^\circ (S) \cup a^\circ (S) \{ a^b/m_\mu^a \} \{ a^\circ a^\circ a^\circ \}$$

It remains to show that $a^\circ (S') \subseteq S^\circ$, that is

$$\delta^a P \cup ((T \setminus \{ m_\mu^a \}), (C \setminus \{ t \})) \{ a/m_\mu \} \subseteq S^\circ.$$  

We observe that $\delta^a P$ is a sub-tree of $S$ with root $a$. Hence, by Lemma B.6, we have

$$a^\circ (S') = a^\circ (\delta^a P) \{ a^b/m_\mu^a \} \{ a^\circ a^\circ \} \subseteq a^\circ ( ((T \setminus \{ m_\mu^a \}), (C \setminus \{ t \})) \{ a/m_\mu \}).$$

By Proposition B.7 we have also

$$a^\circ (\delta^a P) \{ a^b/m_\mu^a \} \subseteq \delta^a P.$$

Therefore, to conclude it is enough to show that

$$a^\circ ( ((T \setminus \{ m_\mu^a \}), (C \setminus \{ t \})) \{ a/m_\mu \}) \subseteq a^\circ (S) \cup a^\circ (S) \{ a^b/m_\mu^a \} \{ a^\circ a^\circ \}.$$

This can be shown following the reasoning used for the similar inclusion in rule In above. It is worth giving some details only about the substitutions. The substitution $\{ a^b/m_\mu^a \}$ guarantees that the opening ambient $a$ acquires any ambient and process local to $m_\mu$. Similarly, the substitution $a^\circ a^\circ$ guarantees that the removal of $m_\mu$ is propagated also to the processes and ambients local to an ambient, which is a son of $m_\mu$. □
We can now prove the main result, that is Lemma 5.8. We recall its assertion for clarity:

Let \( S_2 \in \mathcal{S} \) and \( S\ Nate \in \mathcal{S} \). We have

\[
\alpha \diamond \left( \psi_{S_2}(S\ Nate) \right) \subseteq \psi_{\alpha(S_2)}(\alpha(S\ Nate)).
\]

**Proof (of Lemma 5.8).** We first notice that, by definition of \( \sim \), when \( S_1 \sim S_2 \) we have \( \alpha \diamond (S_1) = \alpha \diamond (S_2) \). Moreover, for any \( S_1 \mapsto S_1' \) we have \( S_2 \mapsto S_2' \) such that \( S_1' \sim S_2' \). This observation permits us to simplify the proof by using, with an abuse of notation, \( S \in \mathcal{S} \) in place of \( [S] \in \mathcal{S} \). By definition of \( \psi_{S_2} \) (Definition 4.9) we therefore have

\[
\psi_{S_2}(S\ Nate) = \{ [S_2] \} \cup \bigcup_{S \in \{ S | S_1 \mapsto S, S_1 \in S\ Nate \}} [S].
\]

Thus, by continuity of \( \alpha \) (Proposition B.3) and using \( \alpha \diamond ([S_2]) = \alpha \diamond (S_2) \) and \( \alpha \diamond ([S]) = \alpha \diamond (S) \), we obtain

\[
\alpha \diamond (\psi_{S_2}(S\ Nate)) = \alpha \diamond (S_2) \cup \bigcup_{S \in \{ S | S_1 \mapsto S, S_1 \in S\ Nate \}} \alpha \diamond (S).
\]

By Lemma B.8 we have that, for each \( S_1 \in S\ Nate \) and for each \( S_1 \mapsto S_3 \), there exists \( \alpha \diamond (S_1) \mapsto \alpha \diamond (S_3) \) such that \( \alpha \diamond (S_3) \subseteq S_3 \). Since \( \{ S_1 \} \subseteq S\ Nate \), then by monotonicity of \( \alpha \) (Proposition B.3) we have \( \alpha \diamond ([S_1]) = \alpha \diamond (S_1) \mapsto \alpha \diamond (S) \). Hence, by Lemma B.4, we have also \( \alpha \diamond (S\ Nate) \mapsto S_4 \) such that \( \alpha \diamond (S_3) \subseteq S_3 \subseteq S_4 \).

We conclude, because by Definition of \( \psi_{\alpha \diamond (S_2)} \) (Definition 5.7), we have

\[
\psi_{\alpha \diamond (S_2)}(\alpha \diamond (S\ Nate)) = \alpha \diamond (S_2) \cup \bigcup_{S \in \{ S | \alpha \diamond (S) \mapsto S_3 \}} S_4. \quad \square
\]

**B.2. Second abstraction**

We first show that the pair of functions \( (\alpha^\circ, \gamma^\circ) \) forms a Galois connection between \( \langle S^\circ, \subseteq \rangle \) and \( \langle S^\circ, \subseteq \rangle \) (Theorem 6.4).

**Proposition B.9.** *The abstract domain \( \langle S^\circ, \subseteq \rangle \) is a complete lattice.*

**Proof.** Straightforward by Proposition B.1. \( \square \)

The following proposition states the basic properties of the concretization and abstraction functions.

**Proposition B.10.** *Function \( \alpha^\circ : \langle S^\circ, \subseteq \rangle \rightarrow \langle S^\circ, \subseteq \rangle \) is monotonic and continuous, and function \( \gamma^\circ : \langle S^\circ, \subseteq \rangle \rightarrow \langle S^\circ, \subseteq \rangle \) is monotonic.*

**Proof.** Trivial by Definition 6.3. \( \square \)

The properties stated above are enough to prove Theorem 6.4.
Proof (of Theorem 6.4). We show that \((\alpha^\circ, \gamma^\circ)\) is a Galois connection (see Definition 2.1). By Propositions B.2 and B.9 both abstract domains are complete lattices. Also, by Proposition B.10 both \(\alpha^\circ\) and \(\gamma^\circ\) are monotonic. Hence, it remains two show that, for \(S^\circ \subseteq S^\circ\) and \(S^\circ \subseteq S^\circ\), we have

\[
S^\circ \subseteq \gamma^\circ(\alpha^\circ(S^\circ))
\]

\[
\alpha^\circ(\gamma^\circ(S^\circ)) \subseteq S^\circ
\]

Both assertions follow straightforwardly from Definition 6.3. We have \(S^\circ \subseteq \gamma^\circ(\alpha^\circ(S^\circ))\) since, by definition of \(\gamma^\circ\) and \(\alpha^\circ\),

\[
\gamma^\circ(\alpha^\circ(S^\circ)) = \bigcup\{S^\circ \mid \alpha^\circ(S^\circ) \subseteq \alpha^\circ(S^\circ)\}.
\]

Moreover, by definition of \(\gamma^\circ\) and \(\alpha^\circ\) and by continuity of \(\alpha^\circ\) (Proposition B.10), we have

\[
\alpha^\circ(\gamma^\circ(S^\circ)) = \alpha^\circ\left(\bigcup\{S^\circ \mid \alpha^\circ(S^\circ) \subseteq S^\circ\}\right) = \bigcup\alpha^\circ(\{S^\circ \mid \alpha^\circ(S^\circ) \subseteq S^\circ\}).
\]

By definition of least upper bound on a complete lattice we conclude therefore

\[
\bigcup\alpha^\circ(\{S^\circ \mid \alpha^\circ(S^\circ) \subseteq S^\circ\}) \subseteq S^\circ.
\]

We now show the safeness of the second abstraction (Lemma 6.6). The proof uses some auxiliary lemmata similar to those shown for the first abstraction.

Lemma B.11. Let \(S^\circ_1, S^\circ_2 \in S^\circ\) be well-labeled abstract states such that \(S^\circ_1 \subseteq S^\circ_2\). if \(S^\circ_1 \rightarrow^\circ S^\circ_1\), then there exists a transition \(S^\circ_2 \rightarrow^\circ S^\circ_2\), such that \(S^\circ_1 \subseteq S^\circ_2\).

Proof. The proof is straightforward by cases on the rules of Table 8. □

Lemma B.12. Let \(S^\circ \in S^\circ\). The function \(\Psi^\circ_{S^\circ} : \langle S^\circ, \subseteq \rangle \rightarrow \langle S^\circ, \subseteq \rangle\) is monotonic.

Proof. This follows from Lemma B.11 using Definition 6.5. □

To simplify the notation we use the following convention: \(a^\circ\) denotes the abstract version of \(a\), that is \(a\rho^\circ\) where \(\rho^\circ\) is the renaming which forgets multiplicities (i.e., \(\rho^\circ(\ell_1) = \rho^\circ(\ell_\omega) = \ell\)). Similarly for processes \(P^\circ\) is the abstract version of \(P\).

Proposition B.13. Let \(P\) be a well-labeled abstract process. We have

\[
\alpha^\circ(\delta^\circ \cdot a^\circ P^\circ) = \delta^\circ a^\circ P^\circ.
\]

Proof. The proof is easy proceeding by induction on the structure of \(P\) and using the definition of \(\alpha^\circ\) (Definition 6.3). We recall that

\[
\alpha^\circ((T^\circ, C^\circ)) = (\{a^\circ b^\circ \mid a^\circ b^\circ \in T^\circ\}, \{a^\circ P \mid a^\circ P \in C^\circ\})\rho^\circ.
\]

Since function \(\alpha^\circ\) removes the partial topology, the information that \(b\) is father of \(a\) is lost. □

Lemma B.14. Let \(S^\circ, S^\circ \in S^\circ\) be well-labeled abstract states. For any \(S^\circ \rightarrow^\circ S^\circ\) there exists an abstract state \(S^\circ\), such that \(\alpha^\circ(\gamma^\circ(S^\circ)) \rightarrow^\circ S^\circ\) and \(\alpha^\circ(S^\circ) \subseteq S^\circ\).
Proof. The proof is by cases on the rule applied to obtain the transition $S^o \rightarrow S^o$. One of the rules Bang$^o$, In$^o$, Out$^o$ and Open$^o$ of Table 7 could have been applied. Let $S^o = (T^o, C^o)$ and $\alpha^o(S^o) = (T^o, C^o)$. We recall that, by definition of $\alpha^o$ (Definition 6.3), we have

$$\alpha^o(S^o) = \{ \{ a^b | a^b \in T^o \}, \{ a^P | a^P \in C^o \} \} \rho^o$$

Bang$^o$ It means that $\beta^! P \in C^o$ and that

$$S^o = S^o \cup \delta^o \beta^! new_\omega(P).$$

By definition of $\alpha^o$ we derive that $\alpha^{o \beta^! P} \in C^o$. Hence, by applying rule Bang$^o$ of Table 8, we obtain a transition $\alpha^o(S^o) \rightarrow S^o$ where

$$S^o = \alpha^o(S^o) \cup \delta^o \beta^! P^o.$$

It remains to show that $\alpha^o(S^o) \subseteq S^o$. By continuity of $\alpha^o$ (Proposition B.10) we have

$$\alpha^o(S^o) = \alpha^o(S^o) \cup \alpha^o(\delta^o \beta^! new_\omega(P)).$$

We notice that, since the abstraction $\alpha^o$ forgets any multiplicity, we have

$$\alpha^o(\delta^o \beta^! new_\omega(P)) = \alpha^o(\delta^o \beta^! P).$$

We conclude, because by Proposition B.13 we have

$$\alpha^o(\delta^o \beta^! P) = \delta^o \alpha^o P^o.$$  

In$^o$ It means that $\beta^c, m^c \in T^o$ and $\beta^c \in m^c, P \in C^o$, and that

$$S^o = S^o \cup \delta^o \beta^c m^c P \cup (T^o \cup \{ a^m \}, C^o \setminus \{ c \})\|a^m / a^b\|.$$

By definition of $\alpha^o$ we have that $\alpha^o(\beta^c m^c \in T^o$ and $\alpha^o(\beta^c \in m^c, P \in C^o$. Hence, by applying rule In$^o$ of Table 8, we obtain a transition $\alpha^o(S^o) \rightarrow S^o$ where

$$S^o = \alpha^o(S^o) \cup \delta^o \beta^c P^o \cup (\{ a^m \}, \emptyset).$$

It remains to show that $\alpha^o(S^o) \subseteq S^o$. By continuity of $\alpha^o$ (Proposition B.10) we have

$$\alpha^o(S^o) = \alpha^o(S^o) \cup \alpha^o(\delta^o \beta^c m^c P)$$

$$\cup \alpha^o(T^o \cup \{ a^m \}, C^o \setminus \{ c \})\|a^m / a^b\|.$$ 

By Proposition B.13 we have that

$$\alpha^o(\delta^o \beta^c m^c P) \subseteq \delta^o \alpha^o P^o.$$ 

We observe that

$$\alpha^o(T^o \cup \{ a^m \}, C^o \setminus \{ c \})\|a^m / a^b\| = \alpha^o(T^o \cup \{ a^m \}, C^o \setminus \{ c \}).$$

In fact, the operation of replacement only affects the partial topology which is removed by the abstraction $\alpha^o$. Furthermore, we have also using the continuity of $\alpha^o$ and the definition of $\setminus$ $\alpha^o(T^o \cup \{ a^m \}, C^o \setminus \{ c \}) \subseteq \alpha^o(T^o \cup \{ a^m \}, C^o) = \alpha^o(S^o) \cup \alpha^o(( a^m, \emptyset)).$ 

Since $\alpha^o(( a^m, \emptyset)) = (\{ a^m \}, \emptyset)$ we conclude that

$$\alpha^o(T^o \cup \{ a^m \}, C^o \setminus \{ c \})\|a^m / a^b\| \subseteq \alpha^o(S^o) \cup (\{ a^m \}, \emptyset).$$
Out° The proof is similar to that of rule In° above.

Open° It means that \( m_\mu^{ab} \in T° \) and \( a° \text{open } m_\lambda \). \( P \in C° \) and that

\[
S° = S° \cup ° \delta° ° b P \cup ° S° \| a° / m_\mu ° a \| \| c° / c° m_\mu ° \|.
\]

By definition of \( \alpha° \) we have \( m_\mu ° a° \in T° \) and \( a° \text{open } m_\lambda \). \( P° \in C° \). Hence, by applying rule Open° of Table 8, we obtain a transition \( \alpha°(S°) \rightarrow ° S° \) where

\[
S° = \alpha°(S°) \cup ° \delta° ° a° P° \cup ° \alpha°(S°) \| a° / m_\ell ° c°.
\]

It remains to show that \( \alpha°(S°) \subseteq S° \). By continuity of \( \alpha° \) (Proposition B.10) we have

\[
\alpha°(S°) = \alpha°(S°) \cup ° \alpha°(\delta° ° b P) \cup ° \alpha°(S° \| a° / m_\ell ° c°).
\]

By Proposition B.13 we have that

\[
\alpha°(\delta° ° b P) \subseteq ° \delta° ° a° P°.
\]

We observe also that, since the abstraction \( \alpha° \) forgets the partial topology, we have

\[
\alpha°(S° \| a° / m_\mu ° a \| \| c° / c° m_\mu ° \|) = \alpha°(S° \| a° / m_\mu ° a \|)
\]

\[
\alpha°(S° \| a° / m_\mu ° a \| \| c° / c° m_\mu ° \|) = \alpha°(S°) \| a° / m_\ell ° c°.
\]

Hence, we conclude that

\[
\alpha°(S° \| a° / m_\mu ° a \| \| c° / c° m_\mu ° \|) \subseteq \alpha°(S°) \| a° / m_\ell ° c°). \quad \Box
\]

We can now show the proof of Lemma 6.6. We recall its assertion:

Let \( S_1°, S_2° \in S° \). We have

\[
\alpha°(\Psi°_{S_1°}(S_1°)) \subseteq \Psi°_{\alpha°(S_1°)}(\alpha°(S_1°))
\]

Proof (of Lemma 6.6). The proof is analogous to that of Lemma 5.8 using Lemma B.14 and Lemma B.11. \( \Box \)

References