Spatial and behavioral types in the pi-calculus[☆]

Lucia Acciai, Michele Boreale

Dipartimento di Sistemi e Informatica
Università di Firenze

Abstract

We present a framework that combines ideas from spatial logics and behavioural type systems. Type systems for the pi-calculus are proposed where newly declared (restricted) names are annotated with spatial process properties, predicating on those names, that are expected to hold in the scope of the declaration. Types are akin to CCS terms and account for the process abstract behaviour and "shallow" spatial structure. Type checking relies on spatial model checking, but properties are checked against types rather than against processes. Type soundness theorems ensure that, for a certain class of spatial properties, well-typed programs are also well-annotated, in the sense that processes in the scope of any restriction do satisfy the corresponding annotation at run-time. The considered class of properties is rather general. Differently from previous proposals, it includes both safety and liveness ones, and is not limited to invariants. We also elaborate a distinction between locally and globally checkable properties.

Key words: pi-calculus, behavioural type systems, spatial logic.

1. Introduction

In the past few years, *spatial logics* [13, 10] have emerged as a promising tool for analyzing properties of concurrent systems. These logics aim at describing the spatial structure of processes, hence at expressing properties related to distribution and concurrency. An easy to grasp example is the race freedom property, stating that at any time, nowhere in the system there are two output actions ready on the same channel. The spectrum of properties that can be expressed by combination of simple spatial and behavioral connectives is very rich (see e.g. [10]). This richness is rather surprising, given the intensional nature of such logics: the process equivalences they induce coincide with, or come very close to, structural congruence (see e.g. [9]), a fine equivalence that only permits elementary rearrangements of the term structure.

A by now well-established trend in the field of process calculi is the use of *behavioural type systems* to simplify the analysis of concurrent message-passing programs [18, 16, 12]. Behavioural types are abstract representations of processes, yet sufficiently expressive to capture some interesting properties. In Igarashi and Kobayashi's generic type systems [16], the work

[☆]Extended and revised version of [2]. Work partially supported by the EU within the FET-GC2 initiative, project Sensoria.

Email addresses: `lacciai@dsi.unifi.it` (Lucia Acciai), `boreale@dsi.unifi.it` (Michele Boreale) that pioneered this approach, processes are pi-calculus terms, while types are akin to CCS terms. The crucial property enjoyed by the system is type soundness: in essence, for a certain class of properties (expressed in a simple modal logic), it holds that if a property is satisfied by a type then it is also satisfied by processes that inhabit that type. Results of this sort can in principle be used to effectively combine type checking and model checking. That is, in some cases it is possible to replace (expensive) model checking on message-passing processes by (cheaper) model checking on types. The paper [12] further elaborates on these themes.

A limitation of behavioural type systems proposed so far concerns the kind of properties that can be tackled this way. In [16, 12], properties for which a general type soundness theorem works are safety invariants – that is, properties of the form "nothing bad will ever happen". Moreover, *compositionality* may be an issue. Ideally, the type system should provide a means to performing model checking on types compositionally. In other words, model checking on the global types corresponding to entire programs should be avoided as much as possible.

In the present paper, we try to combine the expressiveness of spatial logics with the effectiveness of behavioural type systems. Building on Igarashi and Kobayashi's work on generic type systems, we present two distinct type systems for the pi-calculus where newly declared (restricted) names are annotated with properties that predicate on those names. A process in the scope of a restriction is expected to satisfy the restriction's annotation (property) at run-time. We shall focus on properties expressible in a spatial logic – the *Shallow Logic* – which is a fragment of Caires and Cardelli's logic [10]. Types are akin to CCS terms and account for the (abstract) behaviour and the "shallow" spatial structure of processes. The type system relies on (spatial) model checking: however, properties are checked against types rather than against processes. A general soundness theorem is proven stating that, for a certain class of properties, well-typed programs are also well-annotated, in the sense that annotations are satisfied as expected at run-time. The considered class of properties is rather general: unlike previous proposals [16, 12], it includes both safety and (weak) liveness ones, and is not limited to invariants. Several examples of such properties – including race freedom, deadlock freedom and many others – are given throughout the paper.

As another contribution of the paper, we elaborate a distinction between *locally* and *globally checkable* properties. Informally, a locally checkable property is one that can be model-checked against any type by looking at the (local) names it predicates about, while hiding the others; a globally checkable one requires looking also at names causally related to the local ones, hence in principle at names declared elsewhere in the process. These two classes of properties correspond in fact to two distinct type systems, exhibiting different degrees of compositionality and effectiveness (with the global one less compositional/effective). To sum up, we make the following contributions:

- we establish an explicit connection between spatial logics and behavioural type systems. In this respect, a key observation is that processes and their behavioural types share the same "shallow" spatial structure. This fact allows us to prove quite precise correspondences between processes and types and strong type soundness theorems;
- we syntactically identify classes of formulae for which type soundness is guaranteed;
- unlike previous proposals, our type soundness results are not limited to safety properties, nor to invariant properties;
- the verification process resulting from the system is compositional, in the sense that global checks on the type of the whole program are not necessary in general;

- we investigate a distinction between locally and globally checkable properties.

Structure of the paper. In Section 2 we introduce the language of processes, a standard polyadic pi-calculus. In Section 3 we introduce both spatial properties and the Shallow Logic, a simple language to denote them. In Section 4 the first type system, tailored to "local" properties, is presented and thoroughly discussed. Type soundness for this system is discussed in Section 5 and a few examples are discussed in Section 6. A "global" version of the type system is presented and discussed in Section 7, followed by a soundness result and a few examples in Section 8 and 9. A detailed comparison with Igarashi and Kobayashi's system is in Section 10. A few remarks on further and related work conclude the paper in Section 11. The most technical or lengthy proofs have been confined to Appendices A-D.

2. A process calculus

2.1. Types

Types are CCS terms with annotations on input prefixes and restrictions, and input-guarded replication in place of recursion. As usual, we presuppose a countable set $\mathcal{N}$ of *names*. We let lowercase letters $a, b, ..., x, y, ...$ range over names, and $\tilde{a}, \tilde{b}, \ldots, \tilde{x}, \tilde{y}, \ldots$ range over tuples of names. The set $\mathcal{T}$ of *types* $\mathsf{T}, \mathsf{S}, \mathsf{U}, \ldots$ is generated by the following grammar:

$$\begin{array}{rl} \textit{Prefixes} & \mu ::= a(\mathsf{t}) \;\big|\; \overline{a} \;\big|\; \tau \\ \textit{Channel types} & \mathsf{t} ::= (\tilde{x} : \tilde{\mathsf{t}})\mathsf{T} \qquad \text{(with } \tilde{x} \text{ a tuple of distinct names, } \tilde{x} \subseteq \mathrm{fn}(\mathsf{T}) \text{ and } \tilde{x}\#\tilde{\mathsf{t}}\text{)} \\ \textit{Process types} & \mathsf{T} ::= \sum_i \mu_i.\mathsf{T}_i \;\big|\; \mathsf{T}|\mathsf{T} \;\big|\; (\nu\tilde{a} : \tilde{\mathsf{t}})\mathsf{T} \;\big|\; !a(\mathsf{t}).\mathsf{T} \end{array}$$

where $\tilde{x}\#t$ means that $\tilde{x} \cap \mathrm{fn}(t) = \emptyset$. In a channel type $(\tilde{x} : \tilde{\mathsf{t}})\mathsf{T}$, we stipulate that $(\tilde{x} : \tilde{\mathsf{t}})$ is a binder with scope T, where $\tilde{x}$ and $\tilde{\mathsf{t}}$ represent, respectively, the formal parameters and types of objects that can be passed along the channel, while T is a process type prescribing a usage of those parameters. Note that, in $(\tilde{x} : \tilde{\mathsf{t}})\mathsf{T}$, it might in general be $\mathrm{fn}(\mathsf{T}) \setminus \tilde{x} \neq \emptyset$: the usage of received parameters prescribed by a channel type can depend on free names. Here, $a(\mathsf{t}).\mathsf{T}$ is a process type where a can transport names of channel type t. In what follows, we shall write $\mathbf{0}$ for the empty summation and we shall often omit trailing $\mathbf{0}$'s and the channel type $()\mathbf{0}$, writing e.g. $(x)\overline{x}$ instead of $(x : ()\mathbf{0})\overline{x}$ and sometimes shorten $\mathsf{T}_1|\cdots|\mathsf{T}_n$ by $\prod_{i=1,\cdots,n}\mathsf{T}_i$.

Notion of free and bound names arise as expected and types are identified up to alpha-equivalence. Notice that annotations contribute to the set of free names of types, e.g. $\mathrm{fn}(a(\mathsf{t}).\mathsf{T}) = \{a\} \cup \mathrm{fn}(\mathsf{t}) \cup \mathrm{fn}(\mathsf{T})$ and $\mathrm{fn}((\tilde{x} : \tilde{\mathsf{t}})\mathsf{T}) = \mathrm{fn}(\tilde{\mathsf{t}}) \cup \mathrm{fn}(\mathsf{T}) \setminus \tilde{x}$. This ensures that scope extrusion hence structural congruence work properly on types, as discussed later in Remark 3.

2.2. Processes

The language we consider is a synchronous polyadic pi-calculus [21] with guarded summations and replications. Terms $P, Q, R, \ldots$ are defined by the grammar below

$$\begin{array}{rl} \textit{Prefixes} & \alpha ::= a(\tilde{b}) \;\big|\; \overline{a}\langle\tilde{b}\rangle \;\big|\; \tau \\ \textit{Processes} & P ::= \sum_{i\in I} \alpha_i.P_i \;\big|\; P|P \;\big|\; (\nu\tilde{b} : \tilde{\mathsf{t}})P \;\big|\; !a(\tilde{b}).P\,. \end{array}$$

$$P|\mathbf{0} \equiv P \qquad (P|Q)|R \equiv P|(Q|R) \qquad P|Q \equiv Q|P \qquad (\nu\tilde{x}:\tilde{\mathsf{t}})P|Q \equiv (\nu\tilde{x}:\tilde{\mathsf{t}})(P|Q) \quad \text{if } \tilde{x}\#Q$$

Table 1: Laws for structural congruence $\equiv$ on processes

In the input prefix $a(\tilde{b})$. and in the restriction $(\nu\tilde{b}:\tilde{\mathsf{t}})$ the names in the tuple $\tilde{b}$ are assumed distinct. In the restriction clause, $\tilde{\mathsf{t}}$ is a tuple of channel types such that $|\tilde{\mathsf{t}}| = |\tilde{b}|$. Note that restriction acts on tuples of names, $\tilde{b}$, rather than on individual names. Indeed, the form $\nu\tilde{b}$ is equivalent to $\nu b_1 \cdots \nu b_n$ from an operational point of view. When we will introduce annotations (Section 4), however, the form $\nu\tilde{b}$ will allow us to specify properties that should hold of a group of names. Notions of free names $\mathrm{fn}(\cdot)$, of bound names and of alpha-equivalence arise as expected and terms are identified up to alpha-equivalence. In particular, we let $\mathrm{fn}((\nu\tilde{b}:\tilde{\mathsf{t}})P) = (\mathrm{fn}(P)\cup\mathrm{fn}(\tilde{\mathsf{t}}))\setminus\tilde{b}$. To avoid arity mismatches in communications, we shall only consider terms that are well-sorted in some fixed sorting system (see e.g. [21]), and call $\mathcal{P}$ the resulting set of *processes*.

Notation. In the sequel, $P_1|\cdots|P_n$ will be sometimes shortened as $\prod_{i=1,\cdots,n} P_i$. Given $n \geq 0$ tuples of names $\tilde{b}_1, ..., \tilde{b}_n$, we abbreviate $(\nu\tilde{b}_1:\tilde{\mathsf{t}}_1)\cdots(\nu\tilde{b}_n:\tilde{\mathsf{t}}_n)P$ as $(\tilde{\nu}\tilde{b}_i:\tilde{\mathsf{t}}_i)_{i=1,\cdots,n}P$, or simply as $(\tilde{\nu}\tilde{b})P$, with $\tilde{b} = (\tilde{b}_1, \cdots, \tilde{b}_n)$, when the identity of the $\tilde{\mathsf{t}}_i$ is unimportant. When writing down process terms, channel type annotations t may be omitted if not relevant for the discussion.

2.3. Structural congruence and reduction semantics

Over $\mathcal{P}$, we define *structural congruence* and *reduction semantics* as the least congruence $\equiv$ and as the least relation $\xrightarrow{\lambda}$ generated by the axioms in Table 1 and Table 2, respectively. Reductions are annotated with labels λ that carry information on the (free) subject name involved in the corresponding synchronization, if any:

$$\lambda ::= \langle a\rangle \mid \langle\epsilon\rangle \,.$$

Thus, either the subject of the reduction is a free name of the process, a, or ϵ is used to indicate that either the subject is restricted or the reduction originates from a τ-prefix. We shall need the piece of information represented by λ when defining process properties such as "a synchronization on a will eventually take place". We define a hiding operator on labels, written $\lambda\uparrow_{\tilde{b}}$, as follow: $\lambda\uparrow_{\tilde{b}} = \langle a\rangle$ if $\lambda = \langle a\rangle$ and $a \notin \tilde{b}$, $\lambda\uparrow_{\tilde{b}} = \langle\epsilon\rangle$ otherwise.

Remark 1. Concerning Table 1, note that, similarly to [16], we drop two laws for restrictions: $(\nu\tilde{x}:\tilde{\mathsf{t}})(\nu\tilde{y}:\tilde{\mathsf{t}}')P = (\nu\tilde{y}:\tilde{\mathsf{t}}')(\nu\tilde{x}:\tilde{\mathsf{t}})P$ and $(\nu\tilde{y}:\tilde{\mathsf{t}})\mathbf{0} = \mathbf{0}$. The first law is dropped because, on the left-hand side, $(\nu\tilde{x})$ might bind names occurring in $\tilde{\mathsf{t}}'$ that would become free in the right-hand side. The second law becomes problematic once restrictions are decorated with formulae (see Example 5 in Section 4). Also note the absence of a structural law for replication: this is replaced by an explicit reduction rule.

Over $\mathcal{T}$, we define notions of structural congruence and reduction relation similar to those introduced above for processes. Indeed, type annotations on input prefixes play no role in the reduction rules, where types are treated basically as 0-adic processes. As an example, we have $\bar{c}.\mathsf{T}|c(\mathsf{t}).\mathsf{S} \xrightarrow{\langle c\rangle} \mathsf{T}|\mathsf{S}$. Recall that annotations contribute to the set of free names, though: hence annotations do affect structural congruence.

$$(\textsc{com})\ \frac{\alpha_l = a(\tilde{x}) \quad \beta_n = \overline{a}\langle\tilde{b}\rangle \quad l \in I \quad n \in J}{\sum_{i\in I}\alpha_i.P_i | \sum_{j\in J}\beta_j.Q_j \xrightarrow{\langle a\rangle} P_l[\tilde{b}/\tilde{x}]|Q_n} \qquad (\textsc{tau})\ \frac{j \in I \quad \alpha_j = \tau}{\sum_{i\in I}\alpha_i.P_i \xrightarrow{\langle\epsilon\rangle} P_j}$$

$$(\textsc{rep-com})\ \frac{\beta_n = \overline{a}\langle\tilde{b}\rangle \quad n \in J}{!a(\tilde{x}).P | \sum_{j\in J}\beta_j.Q_j \xrightarrow{\langle a\rangle} !a(\tilde{x}).P|P[\tilde{b}/\tilde{x}]|Q_n} \qquad (\textsc{par})\ \frac{P \overset{\lambda}{\to} P'}{P|Q \overset{\lambda}{\to} P'|Q}$$

$$(\textsc{struct})\ \frac{P \equiv Q \quad Q \overset{\lambda}{\to} Q' \quad Q' \equiv P'}{P \overset{\lambda}{\to} P'} \qquad (\textsc{res})\ \frac{P \overset{\lambda}{\to} P'}{(\nu\tilde{x}:\tilde{\mathrm{t}})P \xrightarrow{\lambda\uparrow_{\tilde{x}}} (\nu\tilde{x}:\tilde{\mathrm{t}})P'}$$

Table 2: Rules for the reduction relation $\overset{\lambda}{\to}$ on processes.

Notation. In the sequel, for any sequence $s = \lambda_1 \cdot \cdots \cdot \lambda_n$, we let $P \overset{s}{\to} Q$ mean $P \xrightarrow{\lambda_1} \cdots \xrightarrow{\lambda_n} Q$, and $P \to Q$ (resp. $P \to^* Q$) mean $P \overset{\lambda}{\to} Q$ (resp. $P \overset{s}{\to} Q$) for some λ (resp. s). Moreover, we say that a process P has a *barb* a (resp. $\overline{a}$), written $P \searrow_a$ (resp. $P \searrow_{\overline{a}}$), whenever $P \equiv (\tilde{\nu}\tilde{b})(\sum_i \alpha_i.P_i + a(\tilde{x}).Q|R)$ or $P \equiv (\tilde{\nu}\tilde{b})(!a(\tilde{x}).Q|R)$ (resp. $P \equiv (\tilde{\nu}\tilde{b})(\sum_i \alpha_i.P_i + \overline{a}\langle\tilde{c}\rangle.Q|R)$), with $a \notin \tilde{b}$. Similar notations are defined for types.

3. Properties

We first take a general view of properties as *P-sets*: sets of processes and types, subject to certain conditions. Then we introduce *Shallow Logic*, a simple language to denote an interesting class of such properties. Although processes and types live in different worlds, for the purposes of this section it is possible and convenient to deal with them in a uniform manner. In what follows, we let $A, B, \ldots$ range over the set $\mathcal{U} \triangleq \mathcal{P} \cup \mathcal{T}$. Elements of $\mathcal{U}$ will be generically referred to as *terms*. The proofs not reported in this section can be found in Appendix A.

3.1. P-sets

Following [10, 9], a property set, P-set in brief, is a set of terms closed under structural congruence and having a finite support: the latter intuitively means that the set of names that are "relevant" for the property is finite (somewhat analogous to the notion of free names for syntactic terms). In the following, we let $\{a \leftrightarrow b\}$ denote the *transposition* of a and b, that is, the substitution that assigns a to b and b to a, and leaves the other names unchanged. For $\Phi \subseteq \mathcal{U}$, we let $A \models \Phi$ mean that $A \in \Phi$, and $\Phi\{a \leftrightarrow b\}$ denote the set $\{A\{a \leftrightarrow b\} \,|\, A \models \Phi\}$.

Definition 1 (support, P-set, least support [10]). Let $\Phi \subseteq \mathcal{U}$ and $N \subseteq \mathcal{N}$.

1. N is a *support* of Φ if for each $a, b \notin N$, it holds that $\Phi\{a \leftrightarrow b\} = \Phi$.
2. A *property set* (*P-set*) is a set of terms $\Phi \subseteq \mathcal{U}$ that is closed under $\equiv$ and has a finite support.
3. The *least support* of Φ, written $\mathrm{supp}(\Phi)$, is defined as $\mathrm{supp}(\Phi) \triangleq \bigcap_{N \text{ support of } \Phi} N$.

In other words, N is a support of Φ if renaming names *outside* N with fresh names does not affect Φ. P-sets have finite supports, and since countable intersection of supports is still a support, they also have a least support.

In the rest of the paper we will deal with properties that need not be invariant through reductions. This calls for a notion of λ-*derivative* of a P-set Φ, describing the set of terms reachable via λ-reductions from terms in Φ:

$$\Phi_\lambda \triangleq \{B | \exists A \text{ s.t. } A \models \Phi \text{ and } A \overset{\lambda}{\to} B\}\,.$$

The following property ensures that a λ-derivative of a P-set Φ is a P-set itself, provided λ involves a name in the support of Φ.

Proposition 1. *Let Φ be a P-set. If $\lambda = \langle a \rangle$ with $a \in \mathrm{supp}(\Phi)$ then Φ_λ is a P-set and* $\mathrm{supp}(\Phi_\lambda) \subseteq \mathrm{supp}(\Phi)$.

The $\mathrm{Ok}(\cdot)$ predicate introduced below identifies P-sets that enjoy certain desirable conditions. (1) requires a P-set to be closed under parallel composition with terms not containing free names. (2) demands a P-set to be invariant under reductions that do not involve names in its support. Finally, (3) requires preservation of (1) and (2) under derivatives. These requirements will be essential for guaranteeing the subject reduction property of our type systems. Note the coinductive form of the definition.

Definition 2 ($\mathrm{Ok}(\cdot)$ **predicate).** We define $\mathrm{Ok}(\cdot)$ as the largest predicate on P-sets such that whenever $\mathrm{Ok}(\Phi)$ then:

1. for any $A, B \in \mathcal{P}$ s.t. $\mathrm{fn}(B) = \emptyset$: $A \models \Phi$ if and only if $A|B \models \Phi$; similarly for $A, B \in \mathcal{T}$;
2. if $\lambda = \langle \epsilon \rangle$ or $\lambda = \langle b \rangle$ with $b \notin \mathrm{supp}(\Phi)$ then $\Phi_\lambda = \Phi$;
3. for each λ, $\mathrm{Ok}(\Phi_\lambda)$ holds.

Note that, by virtue of Proposition 1, if $\mathrm{Ok}(\Phi)$ then each Φ_λ is guaranteed to be a P-set.

3.2. Shallow Logic

The logic for the pi-calculus we introduce below can be regarded as a fragment of Caires and Cardelli's Spatial Logic [10]. We christen this fragment *Shallow Logic*, as it allows one to speak about the dynamics as well as the "shallow" spatial structure of processes and types. In particular, the logic does not provide for modalities that allow one to "look underneath" prefixes. Another relevant feature of this fragment is that the basic modalities focus on channel *subjects*, ignoring the object part at all. This selection of operators is sufficient to express a variety of interesting process properties (race freedom, unique receptiveness [23], deadlock freedom, responsiveness [5], to mention a few), while being tractable from the point of view of verification (see also Caires' [9]). Another important property of Shallow Logic will be discussed later in this section (see Lemma 2 below).

Definition 3 (Shallow Logic). The set $\mathcal{F}$ of *Shallow Logic* formulae $\phi, \psi, \ldots$ is given by the following syntax, where $a \in \mathcal{N}$ and $\tilde{a} \subseteq_{\mathrm{fin}} \mathcal{N}$:

$$\phi ::= \mathbf{T} \,\big|\, \phi \vee \phi \,\big|\, \neg\phi \,\big|\, \langle a \rangle \phi \,\big|\, \langle \tilde{a} \rangle^* \phi \,\big|\, \langle -\tilde{a} \rangle^* \phi \,\big|\, a \,\big|\, \overline{a} \,\big|\, \phi | \phi \,\big|\, \mathrm{H}^* \phi.$$

The free names of a formula ϕ, written $\mathrm{fn}(\phi)$, are defined as expected. We let $\mathcal{F}_{\tilde{x}} = \{\phi \in \mathcal{F} : \mathrm{fn}(\phi) \subseteq \tilde{x}\}$. The set of logical operators includes spatial $(a, \overline{a}, |, \mathrm{H}^*)$ as well as dynamic $(\langle a \rangle, \langle \tilde{a} \rangle^*, \langle -\tilde{a} \rangle^*)$ connectives, beside the usual boolean connectives, including a constant $\mathbf{T}$ for "true". The interpretation of $\mathcal{F}$ over the set of processes is given in Table 3. Connectives are

$$
\begin{array}{rcl@{\qquad}rcl}
[\![\mathbf{T}]\!] &=& \mathcal{U} & [\![\mathrm{H}^*\phi]\!] &=& \{A \mid \exists \tilde{a}, B : A \equiv (\nu\tilde{a})B,\ \tilde{a}\#\phi,\ B \in [\![\phi]\!]\} \\
[\![\phi_1 \vee \phi_2]\!] &=& [\![\phi_1]\!] \cup [\![\phi_2]\!] & [\![\phi_1|\phi_2]\!] &=& \{A \mid \exists A_1, A_2 : A \equiv A_1|A_2,\ A_1 \in [\![\phi_1]\!],\ A_2 \in [\![\phi_2]\!]\} \\
[\![\neg\phi]\!] &=& \mathcal{U} \setminus [\![\phi]\!] & [\![\langle a\rangle\phi]\!] &=& \{A \mid \exists B : A \xrightarrow{\langle a\rangle} B,\ B \in [\![\phi]\!]\} \\
[\![a]\!] &=& \{A \mid A \searrow_a\} & [\![\langle\tilde{a}\rangle^*\phi]\!] &=& \{A \mid \exists s, B : A \xrightarrow{s} B,\ s \in \{\langle b\rangle | b \in \tilde{a}\}^*,\ B \in [\![\phi]\!]\} \\
[\![\overline{a}]\!] &=& \{A \mid A \searrow_{\overline{a}}\} & [\![\langle -\tilde{a}\rangle^*\phi]\!] &=& \{A \mid \exists s, B : A \xrightarrow{s} B,\ \tilde{a}\#s,\ B \in [\![\phi]\!]\}
\end{array}
$$

Table 3: Interpretation of formulae over terms.

interpreted in the standard manner. In particular, concerning spatial modalities, the barb atom a (resp. $\overline{a}$) requires that A has an input (resp. output) barb on a; $\phi|\psi$ requires that A can be split into two independent threads satisfying ϕ and ψ; $\mathrm{H}^*\phi$ requires that A satisfies ϕ, up to some top level restrictions. Concerning the dynamic part, formula $\langle a\rangle\phi$ requires an interaction with subject a may lead A to a state where ϕ is satisfied; $\langle\tilde{a}\rangle^*\phi$ requires any number, including zero, of reductions with subject in $\tilde{a}$ may lead A to a state where ϕ is satisfied; $\langle -\tilde{a}\rangle^*\phi$ is similar, but it requires that the subjects of the reductions leading to such a state are *not* in $\tilde{a}$. We write $A \models \phi$ if $A \in [\![\phi]\!]$. Interpretations of formulae are P-sets, as stated below.

Lemma 1. *Let $\phi \in \mathcal{F}$. Then $[\![\phi]\!]$ is a P-set and $fn(\phi) \supseteq \mathrm{supp}([\![\phi]\!])$.*

Notation. In what follows, when no confusion arises, we shall often denote $\Phi = [\![\phi]\!]$ just as ϕ. We abbreviate $\neg\langle -\tilde{x}\rangle^*\neg\phi$ as $\Box^*_{-\tilde{x}}\phi$. Moreover, $\langle -\emptyset\rangle^*\phi$ and $\Box^*_{-\emptyset}\phi$ are abbreviated as $\Diamond^*\phi$ and $\Box^*\phi$, respectively. Note that $\Diamond^*$ and $\Box^*$ correspond to the standard "eventually" and "always" modalities as definable, e.g., in the mu-calculus.

Example 1 (some properties). The following formulae define properties depending on a generic channel name a. They will be reconsidered several times throughout the paper.

$$
\begin{array}{rrcl}
\text{Race freedom:} & \mathit{NoRace}(a) & \triangleq & \Box^*\ \ \neg\mathrm{H}^*(\overline{a}|\overline{a}) \\
\text{Unique receptiveness:} & \mathit{UniRec}(a) & \triangleq & \Box^*\ \ (a \wedge \neg\mathrm{H}^*(a|a)) \\
\text{Responsiveness:} & \mathit{Resp}(a) & \triangleq & \Box^*_{-a}\Diamond^*\langle a\rangle \\
\text{Deadlock freedom:} & \mathit{NoDead}(a) & \triangleq & \Box^*\Big(\ (\overline{a} \rightarrow \mathrm{H}^*(\overline{a}|\Diamond^*\,a))\ \wedge\ (a \rightarrow \mathrm{H}^*(a|\Diamond^*\,\overline{a}))\Big).
\end{array}
$$

NoRace(a) says that it will never be the case that there are two concurrent outputs competing for synchronization on a. *UniRec*(a) says that there will always be exactly one receiver ready on channel a. *Resp*(a) says that, until a reduction on a takes place, it is always possible to reach a reduction on a. If a is a return channel passed to some service, this means the service will, under a suitable fairness assumption, eventually respond (see also [5]). Finally, *NoDead*(a) says that each active output $\overline{a}$ will eventually have a chance of synchronizing with an input, and vice-versa for each active input a.

It is worth to notice that *NoRace*(a) and *UniRec*(a) belong to the class of safety and liveness properties, respectively. On the other hand, *Resp*(a) and *NoDead*(a) might be classified as "weak liveness", in the sense that they combine liveness and safety guarantees. ("No state is reachable from which a good state is unreachable", see Appendix A.2 of [17] for the definitions and more insights on the difference between weak and strong liveness.)

A further motivation for our specific selection of modalities is that satisfaction of any formula of $\mathcal{F}$ is, so to speak, invariant under parallel composition. In particular, whether A satisfies or not a property ϕ of a bunch of names $\tilde{x}$, should not depend on the presence of a *closed* parallel context B. This will be important to guarantee soundness of the scope extrusion law, hence of the subject congruence property, in our system. Formulae of Cardelli and Caires' Spatial Logic outside $\mathcal{F}$ do not, in general, meet this requirement. As an example, the requirement obviously fails for $\neg(\neg\mathbf{0}|\neg\mathbf{0})$, saying that there is at most one non-null thread in the system. As another example, take the formula $\Diamond\mathbf{T}$, where $\Diamond$ is the one-tau-step modality, saying that one reduction is possible, no matter about the subject: the reduction might be provided by the context B and not by A. This explains the omission of this modality from Shallow Logic. The following lemma formally states this property of Shallow Logic (this is in fact a rephrasing of condition 1 of the Ok predicate).

Lemma 2. *Let A be a term and $\phi \in \mathcal{F}_{\tilde{x}}$. For any term B such that $A|B$ is a term and* $\mathrm{fn}(B) = \emptyset$ *we have that $A \models \phi$ if and only if $A|B \models \phi$.*

We shall sometimes need to be careful about the placement of the modality $\langle -\tilde{a}\rangle^*$ with respect to negation $\neg$. To this purpose, it is convenient to introduce two subsets of formulae, positive and negative ones.

Definition 4 (positive and negative formulae). We say a formula ϕ is *positive* (resp. *negative*) if each occurrence of modality $\langle -\tilde{a}\rangle^*$ in ϕ is in the scope of an even (resp. odd) number of negations "$\neg$".

We let $\mathcal{F}^+$ (resp. $\mathcal{F}^-$) denote the subset of positive (resp. negative) formulae in $\mathcal{F}$. The sets $\mathcal{F}^+_{\tilde{x}}$ and $\mathcal{F}^-_{\tilde{x}}$ are defined as expected.

Example 2. Formulae not involving the operator $\langle -\tilde{a}\rangle^*$ are both positive and negative. As an example, a, $\langle a\rangle\mathbf{T}$, $\neg\langle a\rangle\mathbf{T}$, $(\neg a)|a$ are both negative and positive. On the other hand, $\neg\langle -\tilde{b}\rangle^* a$ is negative but not positive, while $\langle -\tilde{b}\rangle^*\neg a$ is positive but not negative. Concerning the formulae introduced in Example 1, note that *NoRace*(a) and *UniRec*(a) are negative, while *Resp*(a) and *NoDead*(a) are neither positive nor negative, as in both of them the modality $\Diamond^*$ occurs simultaneously in negative and in positive positions.

Remark 2. Note that our definitions of "positive" and "negative" are more liberal than the ones considered by Igarashi and Kobayashi [16], where the position of all spatial modalities – including the analogs of $|$, a and $\overline{a}$ – w.r.t. negation must be taken into account. For instance, unique receptiveness would not be considered neither as negative nor as positive in the classification of [16]. This difference has influential consequences on the generality of the type soundness theorems of the type systems.

In the rest of the paper, we shall mainly focus on formulae whose denotations are Ok P-sets. We write $\mathrm{Ok}(\phi)$ if $\mathrm{Ok}([\![\phi]\!])$ holds. The following lemma provides a sufficient syntactic condition for a formula to be Ok (note that condition (1) of $\mathrm{Ok}(\phi)$ is a direct consequence of Lemma 2).

Lemma 3. *Let ϕ be a Shallow Logic formula of the form either $\neg\diamond^*\neg\psi$ or $\neg\langle-\tilde{a}\rangle^*\neg\diamond^*\psi'$, where ψ' does not contain $\neg$. Then* $\mathrm{Ok}(\phi)$.

Example 3. formulae in Example 1 are in the format of Lemma 3, hence they are Ok.

4. A "Local" Type System

We present here our first type system. The adjective "local" refers to the controlled way P-set membership (that is, model checking, in practical cases) is checked within the system. Specifically, as we will see later on, in the restriction rule only the part of a process type T that depends on the restricted names is considered. We are now going to introduce a decorated version of the syntax, the typing rules and the basic properties of the system. Proofs not reported in this section can be found in Appendix B.

4.1. Annotated processes

As anticipated in Section 2, the type system works on annotated processes. Each restriction introduces a property, under the form of an Ok P-set, that depends on the restricted names and is expected to be satisfied by the process in the scope of the restriction. For annotated processes, the clause of restriction is

$$P \ ::= \ \cdots \ \Big|\, (\nu\tilde{a} : \tilde{\mathsf{t}}\,;\ \Phi)P \text{ with } \tilde{a} \supseteq \mathrm{supp}(\Phi) \text{ and } \mathrm{Ok}(\Phi)\ .$$

For brevity, when no confusion arises we shall omit writing explicitly channel types and properties in restrictions, especially when $\mathsf{t} = ()\mathbf{0}$ and $\Phi = [\![\mathbf{T}]\!]$. The reduction rule for restriction of annotated processes takes into account the λ-derivative of Φ in the continuation process. Hence the rule for restriction on annotated processes is

$$(\text{RES})\quad \frac{P \xrightarrow{\lambda} P'}{(\nu\tilde{x} : \tilde{\mathsf{t}}\,; \Phi)P \xrightarrow{\lambda\uparrow_{\tilde{x}}} (\nu\tilde{x} : \tilde{\mathsf{t}}; \Phi_\lambda)P'}\ .$$

Note that, by removing property annotations from restrictions, one gets back exactly the syntax and the reduction relation of plain processes. For an annotated process P, we take "$P \models \phi$" to mean that the plain process obtained by erasing all property annotations from P satisfies ϕ. A "good" process is one that satisfies its own annotations at an active position. Formally:

Definition 5 (well-annotated processes). A process $P \in \mathcal{P}$ is *well-annotated* if whenever $P \equiv (\tilde{\nu}\tilde{b})(\nu\tilde{a};\Phi)Q$ then $Q \models \Phi$.

4.2. Typing rules

Judgements of the type system are of the form $\Gamma \vdash P : \mathsf{T}$, where: $P \in \mathcal{P}$, $\mathsf{T} \in \mathcal{T}$ and Γ is a *context*, that is, a finite partial map from names $\mathcal{N}$ to channel types. We write $\Gamma \vdash a : \mathsf{t}$ if $a \in \mathrm{dom}(\Gamma)$ and $\Gamma(a) = \mathsf{t}$. We say that a context is *well-formed* if whenever $\Gamma \vdash a : (\tilde{x} : \tilde{\mathsf{t}})\mathsf{T}$ then $\mathrm{fn}(\mathsf{T}, \tilde{\mathsf{t}}) \subseteq \tilde{x} \cup \mathrm{dom}(\Gamma)$. In what follows *we shall only consider well-formed contexts*. In the type system, we make use of a "hiding" operation on types, $\mathsf{T}\downarrow_{\tilde{x}}$, which masks the use of names not in $\tilde{x}$ (as usual, in the definition we assume that all bound names in T and t are distinct from each other and disjoint from the set of free names and from $\tilde{x}$). $\mathsf{T}\downarrow_{\tilde{x}}$ is formally defined in Table 4.

$$(\overline{a}.\mathsf{T})\downarrow_{\tilde{x}} = \begin{cases} \tau.(\mathsf{T}\downarrow_{\tilde{x}}) & \text{if } a \notin \tilde{x} \\ \overline{a}.(\mathsf{T}\downarrow_{\tilde{x}}) & \text{otherwise} \end{cases} \qquad (a(\mathsf{t}).\mathsf{T})\downarrow_{\tilde{x}} = \begin{cases} \tau(\mathsf{t}\downarrow_{\tilde{x}}).(\mathsf{T}\downarrow_{\tilde{x}}) & \text{if } a \notin \tilde{x} \\ a(\mathsf{t}\downarrow_{\tilde{x}}).(\mathsf{T}\downarrow_{\tilde{x}}) & \text{otherwise} \end{cases}$$

$$(\mathsf{T}_1|\mathsf{T}_2)\downarrow_{\tilde{x}} = (\mathsf{T}_1\downarrow_{\tilde{x}})|(\mathsf{T}_2\downarrow_{\tilde{x}}) \qquad (\tau.\mathsf{T})\downarrow_{\tilde{x}} = \tau.(\mathsf{T}\downarrow_{\tilde{x}}) \qquad ((\nu\tilde{b}:\tilde{\mathsf{t}})\mathsf{T})\downarrow_{\tilde{x}} = (\nu\tilde{b}:\tilde{\mathsf{t}}\downarrow_{\tilde{x},\tilde{b}})(\mathsf{T}\downarrow_{\tilde{x},\tilde{b}})$$

$$(\textstyle\sum_i \mu_i.\mathsf{T}_i)\downarrow_{\tilde{x}} = \sum_i ((\mu_i.\mathsf{T}_i)\downarrow_{\tilde{x}}\ ) \qquad (!a(\mathsf{t}).\mathsf{T})\downarrow_{\tilde{x}} = !((a(\mathsf{t}).\mathsf{T})\downarrow_{\tilde{x}}\ ) \qquad ((\tilde{y}:\tilde{\mathsf{t}})\mathsf{T})\downarrow_{\tilde{x}} = (\tilde{y}:\tilde{\mathsf{t}}\downarrow_{\tilde{x},y})\mathsf{T}\downarrow_{\tilde{x},\tilde{y}}$$

Table 4: $\mathsf{T}\downarrow_{\tilde{x}}$.

$$\text{(L-Inp)}\ \frac{\Gamma \vdash a : (\tilde{x}:\tilde{\mathsf{t}})\mathsf{T} \quad \Gamma, \tilde{x}:\tilde{\mathsf{t}} \vdash P : \mathsf{T}|\mathsf{T}' \quad \tilde{x}\#\mathsf{T}'}{\Gamma \vdash a(\tilde{x}).P : a((\tilde{x}:\tilde{\mathsf{t}})\mathsf{T}).\mathsf{T}'} \qquad \text{(L-Tau)}\ \frac{\Gamma \vdash P : \mathsf{T}}{\Gamma \vdash \tau.P : \tau.\mathsf{T}}$$

$$\text{(L-Out)}\ \frac{\Gamma \vdash a : (\tilde{x}:\tilde{\mathsf{t}})\mathsf{T} \quad \Gamma \vdash \tilde{b}:\tilde{\mathsf{t}} \quad \Gamma \vdash P : \mathsf{S}}{\Gamma \vdash \overline{a}\langle\tilde{b}\rangle.P : \overline{a}.(\mathsf{T}[^{\tilde{b}}/_{\tilde{x}}]\,|\,\mathsf{S})} \qquad \text{(L-Eq)}\ \frac{\Gamma \vdash P : \mathsf{T} \quad \mathsf{T} \equiv \mathsf{S}}{\Gamma \vdash P : \mathsf{S}}$$

$$\text{(L-Sum)}\ \frac{|I| \neq 1 \quad \forall i \in I:\ \Gamma \vdash \alpha_i.P_i : \mu_i.\mathsf{T}_i}{\Gamma \vdash \sum_i \alpha_i.P_i : \sum_i \mu_i.\mathsf{T}_i} \qquad \text{(L-Rep)}\ \frac{\Gamma \vdash a(\tilde{x}).P : a(\mathsf{t}).\mathsf{T}}{\Gamma \vdash !a(\tilde{x}).P : !a(\mathsf{t}).\mathsf{T}}$$

$$\text{(L-Res)}\ \frac{\Gamma, \tilde{a}:\tilde{\mathsf{t}} \vdash P : \mathsf{T} \quad \mathsf{T}\downarrow_{\tilde{a}} \models \Phi}{\Gamma \vdash (\nu\tilde{a}:\tilde{\mathsf{t}};\Phi)P : (\nu\tilde{a}:\tilde{\mathsf{t}})\mathsf{T}} \qquad \text{(L-Par)}\ \frac{\Gamma \vdash P : \mathsf{T} \quad \Gamma \vdash Q : \mathsf{S}}{\Gamma \vdash P|Q : \mathsf{T}|\mathsf{S}}$$

Table 5: Typing rules for the local system.

As an example, we have that

$$(\nu a:\mathsf{t})(a(\mathsf{t}).b(\mathsf{t}')|a(\mathsf{t}).c(\mathsf{t}'')|\overline{c}|\overline{a})\downarrow_b = (\nu a:\mathsf{t}\downarrow_{a,b})(a(\mathsf{t}\downarrow_{a,b}).b(\mathsf{t}'\downarrow_{a,b})|a(\mathsf{t}\downarrow_{a,b}).\tau(\mathsf{t}''\downarrow_{a,b})|\tau|\overline{a})\ .$$

Notice that terms produced by the hiding operator are in general not in $\mathcal{T}$. Consider e.g. the $\tau(\mathsf{t}''\downarrow_{a,b})$ prefix above or $(!a(\mathsf{t}).\overline{c})\downarrow_c = !\tau(\mathsf{t}\downarrow_c).\overline{c}$. Formally, $\mathsf{T}\downarrow_{\tilde{x}}$ belongs to the set of terms defined by the grammar for types extended as follows:

$$\mathsf{T}\ ::=\ \cdots\ \big|\ !\tau(\mathsf{t}).\mathsf{T}\ \big|\ \tau(\mathsf{t}).\mathsf{T}\ .$$

Note, however, that, by the rules of the type system, such terms only arise when checking the premise of the restriction rule (L-Res). They cannot appear as types assigned to processes.

The rules of the type system are shown in Table 5. The structure of the system is along the lines of [16]; the main differences are discussed in Section 10. The key rules are (L-Inp), (L-Out), (L-Res) and (L-Eq). By and large, the system works as follows: given a process P, it computes an abstraction of P in the form of a type T. At any restriction $(\nu\tilde{a}:\tilde{\mathsf{t}};\Phi)P$ (rule (L-Res)), the abstraction T obtained for P is used to check that P's usage of names $\tilde{a}$ fulfills property Φ ($\mathsf{T}\downarrow_{\tilde{a}} \models \Phi$): in practical cases, Φ is a shallow logic formula and this is actually spatial model checking. Note that $\mathsf{T} \models \Phi$ might be undecidable, however significant decidable fragments have recently been identified (this is further discussed in Section 11). Also note that, thanks to $\downarrow_{\tilde{a}}$, this is checked without looking at the environment: only the part of T that depends on $\tilde{a}$, that is $\mathsf{T}\downarrow_{\tilde{a}}$, is considered, the rest is masked. In particular, in $\mathsf{T}\downarrow_{\tilde{a}}$, any masked parallel component at top-level can be safely discarded (a consequence of condition 1 of Ok). In this sense the type system is "local".

Similarly to [16], rules for input and output are asymmetric, in the sense that, when typing a receiver $a(\tilde{x}).P$, the type information on P that depends on the input parameters $\tilde{x}$ is moved to the sender side. The reason is that the transmitted names $\tilde{b}$ are statically known only by the sender (rule (L-OUT)). Accordingly, on the receiver's side (rule (L-INP)), one only keeps track of the part of the continuation that does not depend on the input parameters, that is T'. More precisely, the type of the continuation P is required to decompose – modulo structural congruence – as $\mathsf{T}|\mathsf{T}'$, where T is the type prescribed by the context Γ for a, and T', which does not mention the input parameters $\tilde{x}$, is anything else. In essence, in well typed processes, all receivers on a must share a common part that deals with the received names $\tilde{x}$ as prescribed by the type T.

Finally, rule (L-EQ) is related to sub-typing. As mentioned in the Introduction, a key point of our system is that types should reflect the (shallow) spatial structure of processes. When considering sub-typing, this fact somehow forces us to abandon preorders in favor of an equivalence relation that respects P-sets membership, which leads to structural congruence. Further discussion on this point is found in Section 10.

The judgements derivable in this type system are written as $\Gamma \vdash_{\mathrm{L}} P : \mathsf{T}$.

Example 4. Consider the formula $\phi = \Box^{*}\neg\mathrm{H}^{*}(\overline{a}|\overline{b})$ saying that it is not possible to reach a configuration where both an output barb on a and one on b are available at the same time. Ok(ϕ) holds by Lemma 3. Consider the process $P = (\nu a, b : \mathsf{t}, \mathsf{t}; \phi)Q$, where: $\mathsf{t} = ()\mathbf{0}$, $Q = ((\overline{d}\langle a\rangle + \overline{d}\langle b\rangle)|\,!a.\overline{b}|\,!b.\overline{a})|d(x).\overline{x}$ and a context Γ s.t. $\Gamma \vdash d : (x : \mathsf{t})\overline{x} = \mathsf{t}'$. By applying the typing rules for input, output, summation and parallel composition:

$$\Gamma, a : \mathsf{t}, b : \mathsf{t} \vdash_{\mathrm{L}} Q \ : \ (\overline{d}.\overline{a} + \overline{d}.\overline{b})|\,!a(\mathsf{t}).\overline{b}|\,!b(\mathsf{t}).\overline{a}|d(\mathsf{t}') \stackrel{\triangle}{=} \mathsf{T}\,.$$

$\mathsf{T}\downarrow_{a,b} = (\tau.\overline{a} + \tau.\overline{b})|\,!a(\mathsf{t}).\overline{b}|\,!b(\mathsf{t}).\overline{a}|\tau \models \phi$; hence, by (L-RES), $\Gamma \vdash_{\mathrm{L}} P : (\nu a, b : \mathsf{t}, \mathsf{t})\mathsf{T}$. This example shows clearly that the structural correspondence between types and processes is shallow: it breaks down as soon as we look underneath prefixes. Indeed, Q and T have the same number of parallel components and they offer the same barbs, regardless of the communication objects. But, as soon as we look at a deeper level, this correspondence breaks down: e.g. the output $\overline{d}\langle a\rangle$ in Q has no continuation, while the output $\overline{d}$ in T has the continuation $\overline{a}$. In this sense the correspondence is shallow.

Example 5. The following examples further illustrate the reasons for dropping the two rules $(\nu a)\mathbf{0} \equiv \mathbf{0}$ and $(\nu a)(\nu b)P \equiv (\nu b)(\nu a)P$ from structural congruence (as discussed in Section 2). Concerning the first rule, note that the process $\mathbf{0}$ is well typed in any context (by (L-SUM)), while e.g. $(\nu y : \mathsf{t}; \phi)\mathbf{0}$, with $\phi = \langle y\rangle\mathbf{T}$ requiring a communication on y, obviously is not. Concerning the second rule, the process $(\nu b : ()0; \mathbf{T})(\nu a : ()0; \diamond^{*}(a|\overline{a}))(a.b|\overline{a}.b)$ is well typed, while $(\nu a : ()0; \diamond^{*}(a|\overline{a}))(\nu b : ()0; \mathbf{T})(a.b|\overline{a}.b)$ is not. In other words, the presence of either rules would violate the subject congruence property (see the next subsection).

Despite the absence of law $(\nu a)(\nu b)P \equiv (\nu b)(\nu a)P$, swapping of two top level restrictions is still possible provided P can be split into two parts, each of them not containing respectively a and b.This swapping can be achieved by repeated applications of the scope extrusion structural law. For instance (a, b and c are assumed of type $()0$; annotations omitted for brevity):

$$(\nu b)(\nu a)(\overline{b}.\overline{c}|b|\overline{a}|c.a) \equiv (\nu b)(\overline{b}.\overline{c}|b)|(\nu a)(\overline{a}|c.a) \equiv (\nu a)(\nu b)(\overline{a}|b|\overline{b}.\overline{c}|c.a)$$

where at each step the scope extrusion rule is applied twice.

Remark 3 (on type annotations). We are now ready to motivate the annotations on types introduced at page 3. Basically, type annotations on both input prefixes and restrictions guarantee that the correspondence between the spatial structure of processes and types is preserved by the scope extrusion law. Consider the process

$$P = (\nu a;\phi)Q \ \text{ where } \ Q = (\nu b)(\nu c : (x)\overline{b}.x)(a.c(x).\overline{b}.x \,|\, \overline{a}.b)$$

and the formula $\phi = \Box^*_{-a}\Diamond^*(\overline{a}\,|\,a)$, which is Ok according to Lemma 3. Under a suitable context, the type system assigns to Q a type that, once annotations are removed, looks like

$$\mathsf{T} = (\nu b)(\nu c)(a.c|\overline{a}.b)\,.$$

Note that, according to the typing rules, the continuation of c is discarded when going from Q to T. Now, since in T names b and c occur in distinct threads, we can split $\mathsf{T} = \mathsf{T}\downarrow_a$ in two, thus

$$\mathsf{T} = \mathsf{T}\downarrow_a \equiv (\nu c)a.c\,|\,(\nu b)\overline{a}.b \models \phi\ .$$

Hence, ignoring annotations leads to concluding, by (L-Res), that P is well typed. The problem is that P is *not* well-annotated according to Definition 5. In fact, Q cannot be split in two independent threads, due to the sharing of the restricted name b

$$Q \not\models \phi$$

so that the soundness result presented in the next section would be violated. This shortcoming is avoided thanks to the presence of annotations in types. Indeed,

$$a : ()0 \vdash_{\mathrm{L}} Q : (\nu b)(\nu c : (x)\overline{b}.x)(a.c((x)\overline{b}.x)\,|\,\overline{a}.b)$$

where the type on the right cannot be split in two, since name b is shared by two threads. Hence

$$(\nu b)(\nu c : (x)\overline{b}.x)(a.c((x)\overline{b}.x)\,|\,\overline{a}.b)\downarrow_a = (\nu b)(\nu c : (x)\overline{b}.x)(a.c((x)\overline{b}.x)\,|\,\overline{a}.b) \not\models \phi$$

which entails, correctly, that P is not well typed. Note that, to guarantee the proper working of scope extrusion, it would be sufficient to annotate restrictions and input prefixes by sets of free names, rather than by whole channel types. We have preferred the latter form, which makes the structural correspondence between types and processes clearer.

Finally, note that, in the above example, the addition of the restriction operator H^* to ϕ would solve the problem: both T and Q satisfy $\phi' = \Box^*_{-a}\Diamond^*\mathrm{H}^*(\overline{a}\,|\,a)$. This points to the expressiveness of the Shallow logic. Indeed, in the logic considered in [16], the satisfaction of $\phi_1|\phi_2$ is defined up-to restrictions, hence it cannot logically distinguish Q above from $R = a.c(x).\overline{b}.x\,|\,\overline{a}.b$.

4.3. Basic properties

We state here the basic properties of the type system presented in the preceding subsection. Let us write $\Gamma \vdash_{\mathrm{NL}} P : \mathsf{T}$ if there exists a *normal* derivation of $\Gamma \vdash_{\mathrm{L}} P : \mathsf{T}$, that is, a derivation where rule (L-Eq) can only be found immediately above rule (L-Inp). Modulo $\equiv$, every judgment derivable in the type system admits a normal derivation.

Proposition 2 (normal derivation). *If* $\Gamma \vdash_{\mathrm{L}} P : \mathsf{T}$ *then* $\Gamma \vdash_{\mathrm{NL}} P : \mathsf{S}$ *for some* $\mathsf{S} \equiv \mathsf{T}$.

Normal derivations are syntax-directed, that is, processes and their types share the same shallow structure. This is formally stated in the two lemmas below.

Lemma 4. *Suppose that* $\Gamma \vdash_{\mathrm{NL}} P : \mathsf{T}$*, that* $\Gamma \vdash \tilde{b} : \tilde{\mathsf{t}}$ *and that* $\Gamma \vdash a : (\tilde{x} : \tilde{\mathsf{t}})\mathsf{U}$*. Then for any* Q, Q_1, Q_2*, and* $\alpha_i.Q_i$ $(i \in I)$*, it holds that:*

1. $P = a(\tilde{x}).Q$ *implies* $\mathsf{T} = a((\tilde{x} : \tilde{\mathsf{t}})\mathsf{U}).\mathsf{S}$ *for some* S *such that* $\Gamma, \tilde{x} : \tilde{\mathsf{t}} \vdash_{\mathrm{L}} Q : \mathsf{S}|\mathsf{U}$ *and* $\tilde{x}\#\mathsf{S}$*;*
2. $P = \overline{a}\langle\tilde{b}\rangle.Q$ *implies* $\mathsf{T} = \overline{a}.(\mathsf{S}|\mathsf{S}')$ *for some* S *and* S' *such that* $\Gamma \vdash_{\mathrm{NL}} Q : \mathsf{S}'$ *and* $\mathsf{S} = \mathsf{U}[\tilde{b}/\tilde{x}]$*;*
3. $P = \tau.Q$ *implies* $\mathsf{T} = \tau.\mathsf{S}$ *for some* S *such that* $\Gamma \vdash_{\mathrm{NL}} Q : \mathsf{S}$*;*
4. $P = (\nu\tilde{c} : \tilde{\mathsf{t}}'; \Phi)Q$ *implies* $\mathsf{T} = (\nu\tilde{c} : \tilde{\mathsf{t}}')\mathsf{S}$ *for some* S *such that* $\Gamma, \tilde{c} : \tilde{\mathsf{t}}' \vdash_{\mathrm{NL}} Q : \mathsf{S}$ *and* $\mathsf{S}\downarrow_{\tilde{c}} \models \Phi$*;*
5. $P = Q_1|Q_2$ *implies* $\mathsf{T} = \mathsf{S}_1|\mathsf{S}_2$ *for some* S_1 *and* S_2 *such that* $\Gamma \vdash_{\mathrm{NL}} Q_1 : \mathsf{S}_1$ *and* $\Gamma \vdash_{\mathrm{NL}} Q_2 : \mathsf{S}_2$*;*
6. $P = !a(\tilde{x}).Q$ *implies* $\mathsf{T} = !a((\tilde{x} : \tilde{\mathsf{t}})\mathsf{U}).\mathsf{S}$ *for some* S *such that* $\Gamma \vdash_{\mathrm{NL}} a(\tilde{x}).Q : a((\tilde{x} : \tilde{\mathsf{t}})\mathsf{U}).\mathsf{S}$*;*
7. $P = \sum_{i\in I}\alpha_i.Q_i$ *implies* $\mathsf{T} = \sum_{i\in I}\mu_i.\mathsf{S}_i$ *for some* S_i *and* μ_i *such that* $\Gamma \vdash_{\mathrm{NL}} \alpha_i.Q_i : \mu_i.\mathsf{S}_i$*, for each* $i \in I$*.*

Proof. The proof is straightforward by induction on the derivation of $\Gamma \vdash_{\mathrm{NL}} P : \mathsf{T}$. It proceeds by considering the last typing rule applied. All cases are obvious (recall that in a normal derivation, rule (L-Eq) cannot be the last applied one).

Lemma 5. *Suppose that* $\Gamma \vdash_{\mathrm{L}} P : \mathsf{T}$*, that* $\Gamma \vdash \tilde{b} : \tilde{\mathsf{t}}$ *and that* $\Gamma \vdash a : (\tilde{x} : \tilde{\mathsf{t}})\mathsf{U}$*. Then for any* $\mathsf{S}, \mathsf{S}_1, \mathsf{S}_2$*, and* $\mu_i.\mathsf{S}_i$ $(i \in I)$*, it holds that:*

1. $\mathsf{T} = a((\tilde{x} : \tilde{\mathsf{t}})\mathsf{U}).\mathsf{S}$ *implies* $P \equiv a(\tilde{x}).Q$ *for some* Q *such that* $\Gamma, \tilde{x} : \tilde{\mathsf{t}} \vdash_{\mathrm{L}} Q : \mathsf{U}|\mathsf{S}$ *and* $\tilde{x}\#\mathsf{S}$*;*
2. $\mathsf{T} = \overline{a}.\mathsf{S}$ *implies* $P \equiv \overline{a}\langle\tilde{b}\rangle.Q$ *for some* Q *and* S' *such that* $\Gamma \vdash_{\mathrm{L}} Q : \mathsf{S}'$ *and* $\mathsf{S} \equiv \mathsf{U}[\tilde{b}/\tilde{x}]|\mathsf{S}'$*;*
3. $\mathsf{T} = \tau.\mathsf{S}$ *implies* $P \equiv \tau.Q$ *for some* Q *such that* $\Gamma \vdash_{\mathrm{L}} Q : \mathsf{S}$*;*
4. $\mathsf{T} = (\nu\tilde{c} : \tilde{\mathsf{t}}')\mathsf{S}$ *implies* $P \equiv (\nu\tilde{c} : \tilde{\mathsf{t}}'; \Phi)Q$ *for some* Q *and* Φ *such that* $\Gamma, \tilde{c} : \tilde{\mathsf{t}}' \vdash_{\mathrm{L}} Q : \mathsf{S}$ *and* $\mathsf{S}\downarrow_{\tilde{c}} \models \Phi$*;*
5. $\mathsf{T} = \mathsf{S}_1|\mathsf{S}_2$ *implies* $P \equiv Q_1|Q_2$ *for some* Q_1 *and* Q_2 *such that* $\Gamma \vdash_{\mathrm{L}} Q_1 : \mathsf{S}_1$*,* $\Gamma \vdash_{\mathrm{L}} Q_2 : \mathsf{S}_2$*;*
6. $\mathsf{T} = !a((\tilde{x} : \tilde{\mathsf{t}})\mathsf{U}).\mathsf{S}$ *implies* $P \equiv !a(\tilde{x}).Q$ *for some* Q *such that* $\Gamma \vdash_{\mathrm{L}} a(\tilde{x}).Q : a((\tilde{x} : \tilde{\mathsf{t}})\mathsf{U}).\mathsf{S}$*;*
7. $\mathsf{T} = \sum_{i\in I}\mu_i.\mathsf{S}_i$*,* $|I| \neq 1$*, implies* $P \equiv \sum_{i\in I}\alpha_i.Q_i$ *for some* Q_i *and* α_i *such that* $\Gamma \vdash_{\mathrm{L}} \alpha_i.Q_i : \mu_i.\mathsf{S}_i$*, for each* $i \in I$*.*

Subject reduction relies on a few intermediate results. Let us first introduce some more terminology. In the remainder of the paper we define $((\tilde{y} : \tilde{\mathsf{t}})\mathsf{T})[\tilde{b}/\tilde{x}] \triangleq (\tilde{y} : \tilde{\mathsf{t}}[\tilde{b}/\tilde{x}])\mathsf{T}[\tilde{b}/\tilde{x}]$, for any $\tilde{x}$ and $\tilde{b}$, with implicit alpha renaming of bound names to avoid captures. This definition is generalized to Γ such that $\Gamma \vdash \tilde{x} : \tilde{\mathsf{t}}$ and $\Gamma \vdash \tilde{b} : \tilde{\mathsf{t}}$ as follows: $\Gamma[\tilde{b}/\tilde{x}](a) = \Gamma(a)[\tilde{b}/\tilde{x}]$, for any $a \in \mathrm{dom}(\Gamma)$. Notice that $\Gamma \vdash \tilde{x} : \tilde{\mathsf{t}}$ and $\Gamma \vdash \tilde{b} : \tilde{\mathsf{t}}$ imply that (i) $\Gamma[\tilde{b}/\tilde{x}] \vdash \tilde{x} : \tilde{\mathsf{t}}[\tilde{b}/\tilde{x}]$, (ii) $\Gamma[\tilde{b}/\tilde{x}] \vdash \tilde{b} : \tilde{\mathsf{t}}[\tilde{b}/\tilde{x}]$ and (iii) $\Gamma[\tilde{b}/\tilde{x}]$ is well-formed. Finally, let $\mathrm{fn}(\Gamma) \triangleq \bigcup_{a\in\mathrm{dom}(\Gamma)} \mathrm{fn}(\Gamma(a))$.

Proposition 3 (subject congruence). $\Gamma \vdash_{\mathrm{L}} P : \mathsf{S}$ *and* $P \equiv Q$ *implies* $\Gamma \vdash_{\mathrm{L}} Q : \mathsf{S}$*.*

Proposition 4 (substitution). *Suppose* $\Gamma, \tilde{x} : \tilde{\mathsf{t}} \vdash_{\mathrm{L}} P : \mathsf{T}$*, with* Γ *and* $\Gamma, \tilde{x} : \tilde{\mathsf{t}}$ *well-formed, and* $\Gamma \vdash \tilde{b} : \tilde{\mathsf{t}}$*. Then* $\Gamma[\tilde{b}/\tilde{x}] \vdash_{\mathrm{L}} P[\tilde{b}/\tilde{x}] : \mathsf{T}[\tilde{b}/\tilde{x}]$*.*

The following result establishes the operational correspondence between T and $\mathsf{T}\downarrow_{\tilde{x}}$. This correspondence is strict as long as reductions involve a name in $\tilde{x}$, a bound name or a τ-prefix (in T). Otherwise, a single step from T can be mimicked by two $\langle\epsilon\rangle$ reductions from $\mathsf{T}\downarrow_{\tilde{x}}$.

Proposition 5 (operational correspondence between T and $\mathsf{T}\downarrow_{\tilde{x}}$). *(i) If $\mathsf{T} \xrightarrow{\lambda} \mathsf{T}'$, with $\lambda ::= \langle\epsilon\rangle \,\big|\, \langle a\rangle$ and $a \in \tilde{x}$, then $\mathsf{T}\downarrow_{\tilde{x}} \xrightarrow{\lambda} \mathsf{T}'\downarrow_{\tilde{x}}$. (ii) If $\mathsf{T} \xrightarrow{\lambda} \mathsf{T}'$, with $\lambda = \langle a\rangle$ and $a \notin \tilde{x}$, then $\mathsf{T}\downarrow_{\tilde{x}} \xrightarrow{\langle\epsilon\rangle} \xrightarrow{\langle\epsilon\rangle} \mathsf{T}'\downarrow_{\tilde{x}}$. (iii) If $\mathsf{T} \xrightarrow{s} \mathsf{T}'$ then $\mathsf{T}\downarrow_{\tilde{x}} \xrightarrow{s'} \mathsf{T}'\downarrow_{\tilde{x}}$, with $\mathrm{fn}(s') \subseteq \mathrm{fn}(s)$. (iv) If $\mathsf{T}\downarrow_{\tilde{x}} \xrightarrow{\lambda} \mathsf{T}'$, with $\lambda = \langle a\rangle$ or $\lambda = \langle\epsilon\rangle$ and the reduction originates from a synchronization on a bound name or a τ–prefix in T, then $\mathsf{T} \xrightarrow{\lambda} \mathsf{S}$, with $\mathsf{T}' = \mathsf{S}\downarrow_{\tilde{x}}$.*

Theorem 1 (subject reduction). *$\Gamma \vdash_{\mathrm{L}} P : \mathsf{T}$ and $P \xrightarrow{\lambda} P'$ implies that there exists a T' such that $\mathsf{T} \xrightarrow{\lambda} \mathsf{T}'$ and $\Gamma \vdash_{\mathrm{L}} P' : \mathsf{T}'$.*

Proof. The proof proceeds by induction on the derivation of $P \xrightarrow{\lambda} P'$, where the last reduction rule applied is distinguished. We examine the most interesting cases below. Recall that by Proposition 2, a normal derivation for $\Gamma \vdash_{\mathrm{L}} P : \mathsf{T}$ exists, say $\Gamma \vdash_{\mathrm{NL}} P : \mathsf{S}$ with $\mathsf{T} \equiv \mathsf{S}$.

(struct). By $P \xrightarrow{\lambda} P'$ and the premise of the rule, we get $P \equiv Q$, $Q \xrightarrow{\lambda} Q'$ and $Q' \equiv P'$. By Proposition 3 (subject congruence), $\Gamma \vdash_{\mathrm{NL}} P : \mathsf{S}$ implies $\Gamma \vdash_{\mathrm{L}} Q : \mathsf{S}$; hence, by applying the induction hypothesis, we get $\mathsf{S} \xrightarrow{\lambda} \mathsf{S}'$ and $\Gamma \vdash_{\mathrm{L}} Q' : \mathsf{S}'$. Finally, by (struct), $\mathsf{S} \equiv \mathsf{T} \xrightarrow{\lambda} \mathsf{S}' = \mathsf{T}'$ and by Proposition 3 (subject congruence), $\Gamma \vdash_{\mathrm{L}} P' : \mathsf{T}'$.

(com). Assume for notational simplicity that $P = a(\tilde{x}).R|\overline{a}\langle\tilde{b}\rangle.Q \xrightarrow{\langle a\rangle} R[\tilde{b}/\tilde{x}]\,|\,Q = P'$ (the general case with arbitrary guarded summations is similar). By $\Gamma \vdash_{\mathrm{NL}} P : \mathsf{S}$ and Lemma 4, we get $\mathsf{S} = a.\mathsf{V}|\overline{a}.(\mathsf{U}[\tilde{b}/\tilde{x}]|\mathsf{V}')$ with $\Gamma \vdash_{\mathrm{L}} Q : \mathsf{V}'$, $\Gamma, \tilde{x} : \tilde{\mathrm{t}} \vdash_{\mathrm{L}} R : \mathsf{V}|\mathsf{U}$ with $\tilde{x}\#\mathsf{V}$, where $\Gamma \vdash_{\mathrm{L}} a : (\tilde{x} : \tilde{\mathrm{t}})\mathsf{U}$ and $\Gamma \vdash_{\mathrm{L}} \tilde{b} : \tilde{\mathrm{t}}$. We can also assume that $\tilde{x}\#\Gamma$.

By (com) and (struct), $\mathsf{S} \xrightarrow{\langle a\rangle} \mathsf{V}|\mathsf{U}[\tilde{b}/\tilde{x}]|\mathsf{V}'$ and $\mathsf{T} \xrightarrow{\langle a\rangle} \mathsf{V}|\mathsf{U}[\tilde{b}/\tilde{x}]|\mathsf{V}' \triangleq \mathsf{T}'$. By $\tilde{x}\#\Gamma$ it follows that $\Gamma[\tilde{b}/\tilde{x}] = \Gamma$. Moreover, by Proposition 4 (substitution) and (L-Par) we get $\Gamma \vdash_{\mathrm{L}} R[\tilde{b}/\tilde{x}]|Q : (\mathsf{V}|\mathsf{U})[\tilde{b}/\tilde{x}]|\mathsf{V}'$. Finally, by $\tilde{x}\#\mathsf{V}$ we get $(\mathsf{V}|\mathsf{U})[\tilde{b}/\tilde{x}]|\mathsf{V}' = \mathsf{V}|\mathsf{U}[\tilde{b}/\tilde{x}]|\mathsf{V}' = \mathsf{T}'$ and by (L-Eq), we get $\Gamma \vdash_{\mathrm{L}} P' : \mathsf{T}'$.

(res). By $P = (\nu\tilde{a} : \tilde{\mathrm{t}}; \Phi)P_0 \xrightarrow{\lambda\uparrow_{\tilde{a}}} (\nu\tilde{a} : \tilde{\mathrm{t}}; \Phi_\lambda)P_0' = P'$ and the premise of the rule, we get $P_0 \xrightarrow{\lambda} P_0'$. By $\Gamma \vdash_{\mathrm{NL}} (\nu\tilde{a} : \tilde{\mathrm{t}}; \Phi)P_0 : \mathsf{S}$ and Lemma 4, we get $\mathsf{S} = (\nu\tilde{a} : \tilde{\mathrm{t}})\mathsf{U}$, with $\Gamma, \tilde{a} : \tilde{\mathrm{t}} \vdash_{\mathrm{L}} P_0 : \mathsf{U}$ and $\mathsf{U}\downarrow_{\tilde{a}} \models \Phi$. Hence, by applying the induction hypothesis, we get $\mathsf{U} \xrightarrow{\lambda} \mathsf{U}'$ and $\Gamma, \tilde{a} : \tilde{\mathrm{t}} \vdash_{\mathrm{L}} P_0' : \mathsf{U}'$. By (res) we get $(\nu\tilde{a} : \tilde{\mathrm{t}})\mathsf{U} \xrightarrow{\lambda\uparrow_{\tilde{a}}} (\nu\tilde{a} : \tilde{\mathrm{t}})\mathsf{U}' \triangleq \mathsf{T}'$ and by (struct), $\mathsf{T} \xrightarrow{\lambda\uparrow_{\tilde{a}}} \mathsf{T}'$.

We have to prove that the premise of the rule (L-Res) is fulfilled. We already know that $\Gamma, \tilde{a} : \tilde{\mathrm{t}} \vdash_{\mathrm{L}} P_0' : \mathsf{U}'$, thus it suffices to show that $\mathsf{U}'\downarrow_{\tilde{a}} \models \Phi_\lambda$. We consider two possibilities (recall that $\tilde{a} \supseteq \mathrm{supp}(\Phi)$):

1. $\lambda = \langle a\rangle$, with $a \in \tilde{a}$, or $\lambda = \langle\epsilon\rangle$. $\mathsf{U}'\downarrow_{\tilde{a}} \models \Phi_\lambda$ follows by $\mathsf{U}\downarrow_{\tilde{a}} \models \Phi$ and $\mathsf{U}\downarrow_{\tilde{a}} \xrightarrow{\lambda} \mathsf{U}'\downarrow_{\tilde{a}}$, a consequence of $\mathsf{U} \xrightarrow{\lambda} \mathsf{U}'$ and Proposition 5.
2. otherwise $\Phi_\lambda = \Phi_{\langle\epsilon\rangle} = \Phi$, by $\mathrm{Ok}(\Phi)$. The reduction $\mathsf{U} \xrightarrow{\lambda} \mathsf{U}'$ has a free subject not in $\tilde{a}$. By Proposition 5, this reduction can be simulated by a pair of ϵ-reductions of $\mathsf{U}\downarrow_{\tilde{a}}$ that consume the corresponding prefixes, thus: $\mathsf{U}\downarrow_{\tilde{a}} \xrightarrow{\langle\epsilon\rangle} \xrightarrow{\langle\epsilon\rangle} \mathsf{U}'\downarrow_{\tilde{a}}$. By definition of Φ_λ and by $\mathrm{Ok}(\Phi)$, we have that: $\mathsf{U}'\downarrow_{\tilde{a}} \models \Phi_{\langle\epsilon\rangle\langle\epsilon\rangle} = \Phi_{\langle\epsilon\rangle} = \Phi = \Phi_\lambda$.

In both cases (L-Res) can be applied to deduce $\Gamma \vdash_{\mathrm{L}} P' = (\nu\tilde{a} : \tilde{\mathsf{t}}; \Phi_\lambda)P'_0 : (\nu\tilde{a} : \tilde{\mathsf{t}})\mathsf{U}' = \mathsf{T}'$.

The system $\vdash_{\mathrm{L}}$ enjoys a sort of "inverse subject reduction" property guaranteeing that processes simulate their types.

Theorem 2 (type subject reduction). *$\Gamma \vdash_{\mathrm{L}} P : \mathsf{T}$ and $\mathsf{T} \xrightarrow{\lambda} \mathsf{T}'$ implies that there exists a P' such that $P \xrightarrow{\lambda} P'$ and $\Gamma \vdash_{\mathrm{L}} P' : \mathsf{T}'$.*

5. Type Soundness for the Local System

In this section we prove a general type soundness result for our system: we study classes of properties for which well-typed-ness implies well-annotated-ness. We first identify the general class of properties for which, at least in principle, model checking on processes can be reduced to a type checking problem whose solution requires only a (local) use of model checking on types. We do so by the following coinductive definition.

Definition 6 (locally checkable properties). We let Lc be the largest predicate on P-sets such that whenever $\mathrm{Lc}(\Phi)$ then $\mathrm{Ok}(\Phi)$ and:

1. whenever $\Gamma \vdash_{\mathrm{L}} P : \mathsf{T}$ and $\tilde{x} \supseteq \mathrm{supp}(\Phi)$ and $\mathsf{T} \downarrow_{\tilde{x}} \models \Phi$ then $P \models \Phi$;
2. $\mathrm{Lc}(\Phi_\lambda)$ holds for each λ.

If $\mathrm{Lc}(\Phi)$ then we say Φ is *locally checkable*.

A formula $\phi \in \mathcal{F}$ is said to be locally checkable if $[\![\phi]\!]$ is locally checkable. The following theorem is quite expected.

Theorem 3 (type soundness). *Suppose $\Gamma \vdash_{\mathrm{L}} P : \mathsf{T}$ and P is decorated with locally checkable P-sets only. Then P is well-annotated.*

Proof. Suppose $\Gamma \vdash_{\mathrm{L}} P : \mathsf{T}$ and $P \equiv (\tilde{\nu b})(\nu\tilde{a} : \Phi)Q$. The P-set Φ is locally checkable by hypothesis. We have to prove that $Q \models \Phi$.

By Proposition 3 (subject congruence), $\Gamma \vdash_{\mathrm{L}} P : \mathsf{T}$ and $P \equiv (\tilde{\nu b})(\nu\tilde{a} : \Phi)Q$, we deduce that $\Gamma \vdash_{\mathrm{L}} (\tilde{\nu b})(\nu\tilde{a} : \Phi)Q : \mathsf{T}$. A normal derivation of this judgment exists (Proposition 2), $\Gamma \vdash_{\mathrm{NL}} (\tilde{\nu b})(\nu\tilde{a} : \Phi)Q : \mathsf{S} \equiv \mathsf{T}$ and, by Lemma 4, $\mathsf{S} \equiv (\tilde{\nu b})(\nu\tilde{a})\mathsf{T}'$ with $\Gamma, \tilde{b} : \tilde{\mathsf{t}}, \tilde{a} : \tilde{\mathsf{t}}' \vdash_{\mathrm{L}} Q : \mathsf{T}'$ and $\mathsf{T}' \downarrow_{\tilde{a}} \models \Phi$. By Definition 6, part 1, it follows that $Q \models \Phi$.

The following corollary is a consequence of type soundness and subject reduction.

Corollary 1 (run-time soundness). *Suppose that $\Gamma \vdash_{\mathrm{L}} P : \mathsf{T}$ and that P is decorated with locally checkable P-sets only. Then $P \to^* P'$ implies that P' is well-annotated.*

Our task is now providing sufficient *syntactic* conditions on a formula ϕ that guarantee $\mathrm{Lc}([\![\phi]\!])$. In the following, we will write $\Gamma \vdash_{\mathrm{K}} P : \mathsf{T}$ if well-typedness of P can be derived from the rules in Table 5 by omitting the check $\mathsf{T} \downarrow_{\tilde{a}} \models \Phi$ in the premise of rule (L-Res). The system $\vdash_{\mathrm{K}}$ can be seen as the kernel of $\vdash_{\mathrm{L}}$: its only purpose is to extract abstractions out of processes, without performing any check. The following proposition guarantees that processes and the corresponding types satisfy the same Shallow logic formulae.

Proposition 6. *Suppose* $\Gamma \vdash_K P : \mathsf{T}$ *and let* $\phi \in \mathcal{F}$*. Then* $\mathsf{T} \models \phi$ *if and only if* $P \models \phi$*.*

Proof. First, notice that $\Gamma \vdash_K P : \mathsf{T}$ if and only if $\Gamma \vdash_L P' : \mathsf{T}$, with P' obtained from P by replacing each P-set annotation with $[[\mathbf{T}]]$. Therefore, all basic properties of the system $\vdash_L$ introduced in Section 4.3 carry over to $\vdash_K$ provided that all Φs are replaced by $[[\mathbf{T}]]$.

The proof is straightforward by induction on the structure of ϕ. The cases of the boolean connectives easily follow from the induction hypothesis. The spatial connectives are accommodated relying on the structural correspondence between processes and their types (by Lemma 4 and 5). The dynamic connectives are accommodated relying on Theorem 1 and Theorem 2. Below, we cover in detail the two most interesting cases.

$\phi = \phi_1|\phi_2$.

(⇒). $\mathsf{T} \models \phi_1|\phi_2$ implies that $\mathsf{T} \equiv \mathsf{T}_1|\mathsf{T}_2$ with $\mathsf{T}_i \models \phi_i$ for $i = 1,2$. By rule (L-Eq), $\Gamma \vdash_K P : \mathsf{T}_1|\mathsf{T}_2$, hence by Lemma 5, $P \equiv P_1|P_2$, with $\Gamma \vdash_K P_1 : \mathsf{T}_1$ and $\Gamma \vdash_K P_2 : \mathsf{T}_2$. Therefore, by applying the induction hypothesis, we get $P_1 \models \phi_1$ and $P_2 \models \phi_2$, that is $P \models \phi$.

(⇐). The proof proceeds in a similar way, by applying Proposition 2 and Lemma 4 in place of Lemma 5.

$\phi = \langle -\tilde{b}\rangle^* \psi$.

(⇒). $\mathsf{T} \models \phi$ implies that $\mathsf{T} \xrightarrow{s} \mathsf{T}'$ and $\mathsf{T}' \models \psi$ for some T' and $\tilde{b}\#s$. By Theorem 2 (type subject reduction), we get $P \xrightarrow{s} P'$ and $\Gamma \vdash_K P' : \mathsf{T}'$.

From $\mathsf{T}' \models \psi$, $\Gamma \vdash_L P' : \mathsf{T}'$ and the induction hypothesis, we get $P' \models \psi$. Hence, $P \models \langle -\tilde{b}\rangle^* \psi$.

(⇐). The proof proceed in a similar way; Theorem 1 (subject reduction) is applied instead of Theorem 2.

We still miss two ingredients to obtain our main result. The first one relates the hiding operator to structural congruence. The second one is about the effect of the hiding operator on satisfiability. (The proof of Lemma 6 can be found in Appendix B.)

Lemma 6.

1. *Suppose* $a \in \tilde{x}$. $(\mathsf{T} \downarrow_{\tilde{x}}) \searrow_a$ *if and only if* $\mathsf{T} \searrow_a$.
2. *If* $\mathsf{T} \downarrow_{\tilde{x}} \equiv \mathsf{T}_1|\mathsf{T}_2$ *then there are* S_1 *and* S_2 *such that* $\mathsf{T} \equiv \mathsf{S}_1|\mathsf{S}_2$ *and* $\mathsf{S}_i \downarrow_{\tilde{x}} = \mathsf{T}_i$*, for* $i = 1,2$.
3. *If* $\mathsf{T} \equiv \mathsf{T}_1|\mathsf{T}_2$ *then there are* S_1 *and* S_2 *such that* $\mathsf{T} \downarrow_{\tilde{x}} \equiv \mathsf{S}_1|\mathsf{S}_2$ *and* $\mathsf{S}_i = \mathsf{T}_i \downarrow_{\tilde{x}}$*, for* $i = 1,2$.
4. *If* $\mathsf{T} \downarrow_{\tilde{x}} \equiv (\tilde{\nu}\tilde{a})\mathsf{S}$ *then there is* V *such that* $\mathsf{T} \equiv (\tilde{\nu}\tilde{a})\mathsf{V}$*, with* $\mathsf{V} \downarrow_{\tilde{x},\tilde{a}} = \mathsf{S}$.
5. *If* $\mathsf{T} \equiv (\tilde{\nu}\tilde{a})\mathsf{S}$ *then there is* V *such that* $\mathsf{T} \downarrow_{\tilde{x}} \equiv (\tilde{\nu}\tilde{a})\mathsf{V}$*, with* $\mathsf{V} = \mathsf{S} \downarrow_{\tilde{x},\tilde{a}}$.

Lemma 7. (*a*) *If* $\phi \in \mathcal{F}^-_{\tilde{x}}$ *and* $\mathsf{T} \downarrow_{\tilde{x}} \models \phi$ *then* $\mathsf{T} \models \phi$*.* (*b*) *If* $\phi \in \mathcal{F}^+_{\tilde{x}}$ *and* $\mathsf{T} \models \phi$ *then* $\mathsf{T} \downarrow_{\tilde{x}} \models \phi$*.*

Proof. The two statements (*a*) and (*b*) are proven by mutual induction on the structure of formula ϕ. We show only the most interesting cases: H^* and $\langle -\tilde{b}\rangle^*$. Barb and parallel composition cases follow from Lemma 6, while the others from the induction hypothesis. Notice that the mutual induction comes into play in the case $\phi = \neg\psi$.

$\phi = \mathrm{H}^*\psi$**.** Suppose ϕ (hence ψ) is a negative formula. $\mathsf{T}\downarrow_{\tilde{x}} \models \mathrm{H}^*\psi$ implies that $\mathsf{T}\downarrow_{\tilde{x}} \equiv (\tilde{\nu a})\mathsf{U}$ with $\mathsf{U} \models \psi$. By Lemma 6 (4), it follows that $\mathsf{T} \equiv (\tilde{\nu a})\mathsf{V}$, for some V such that $\mathsf{V}\downarrow_{\tilde{x},\tilde{a}} = \mathsf{U}$. Hence by $\phi \in \mathcal{F}^-_{\tilde{x}} \subseteq \mathcal{F}^-_{\tilde{x},\tilde{a}}$ and by the induction hypothesis, $\mathsf{V} \models \psi$ and $\mathsf{T} \equiv (\tilde{\nu a})\mathsf{V} \models \mathrm{H}^*\psi = \phi$.

Suppose ϕ (hence ψ) is positive. $\mathsf{T} \models \mathrm{H}^*\psi$ implies $\mathsf{T} \equiv (\tilde{\nu a})\mathsf{V}$, with $\mathsf{V} \models \psi$. By Lemma 6 (5), $\mathsf{T}\downarrow_{\tilde{x}} \equiv (\tilde{\nu a})\mathsf{U}$ with $\mathsf{U} = \mathsf{V}\downarrow_{\tilde{x},\tilde{a}}$. By applying the inductive hypothesis, $\mathsf{U} \models \psi$ and, by definition, $(\tilde{\nu a})\mathsf{U} \models \mathrm{H}^*\psi$, hence $\mathsf{T}\downarrow_{\tilde{x}} \models \phi$.

$\phi = \langle -\tilde{b}\rangle^*\psi$**.** By definition of positive formulae, both ϕ and ψ are positive. $\mathsf{T} \models \phi$ implies that $\mathsf{T} \xrightarrow{s} \mathsf{S}$, $\tilde{b}\#s$ and $\mathsf{S} \models \psi$ for some S. By Proposition 5, $\mathsf{T}\downarrow_{\tilde{x}} \xrightarrow{s'} \mathsf{S}\downarrow_{\tilde{x}}$, with $\mathrm{fn}(s') \subseteq \mathrm{fn}(s)$, and by applying the inductive hypothesis, $\mathsf{S}\downarrow_{\tilde{x}} \models \psi$. Therefore, by definition, $\mathsf{T}\downarrow_{\tilde{x}} \models \langle -\tilde{b}\rangle^*\psi$.

Theorem 4. *Any negative formula of the form $\Box^*\phi$ is locally checkable.*

Proof. Each formula of the given form is Ok (Lemma 3). Notice that well typedness in $\vdash_{\mathrm{L}}$ implies well typedness in $\vdash_{\mathrm{K}}$, hence by Lemma 7 *(a)* and Proposition 6, each (denotation of a) negative formula satisfies condition 1 of the definition of Lc. Moreover, $[\![\Box^*\phi]\!] = [\![\Box^*\phi]\!]_\lambda$ for each λ. This shows that $\{[\![\Box^*\phi]\!] \mid \phi \in \mathcal{F}^-\} \subseteq \mathrm{Lc}$.

The above result provides us a type soundness result for an interesting class of formulae, that includes both safety and liveness properties. Some examples will be given in the next section.

6. Examples in the Local System

The formulae *NoRace*(a) and *UniRec*(a) fit in the format given by Theorem 4, hence they are locally checkable. As an example, consider

$$P = (\nu a,b,c:()\mathbf{0},\mathsf{t}',\mathsf{t}\,;\ \mathit{UniRec}(a))\Big(\overline{c}\langle a\rangle \mid a + b(x).x \mid c(y).(\overline{b}\langle y\rangle|\overline{d}\langle y\rangle) \mid d(z).\overline{z}.z\Big)$$

where $\mathsf{t} = (x)\overline{b}.x$ and $\mathsf{t}' = (y)y$. Assume that $\Gamma \vdash_{\mathrm{L}} d : \mathsf{t}''$, with $\mathsf{t}'' = (z)\overline{z}.z$. By the typing rules, we easily derive

$$\Gamma, a:()\mathbf{0}, b:\mathsf{t}', c:\mathsf{t} \vdash_{\mathrm{L}} \overline{c}\langle a\rangle \mid a + b(x).x \mid c(y).(\overline{b}\langle y\rangle|\overline{d}\langle y\rangle) \mid d(z).\overline{z}.z \,:\, \mathsf{T}$$

with

$$\mathsf{T} \triangleq \overline{c}.(\overline{b}.a|\overline{d}.\overline{a}.a) \mid a + b(\mathsf{t}') \mid c(\mathsf{t}) \mid d(\mathsf{t}'')\ .$$

Since

$$\mathsf{T}\downarrow_{a,b,c} = \overline{c}.(\overline{b}.a|\tau.\overline{a}.a) \mid a + b(\mathsf{t}') \mid c(\mathsf{t}) \mid \tau(\mathsf{t}'') \models \mathit{UniRec}(a)$$

we can apply (L-Res) and get

$$\Gamma \vdash_{\mathrm{L}} P : (\nu a,b,c:()\mathbf{0},\mathsf{t}',\mathsf{t})\mathsf{T}\,.$$

For another example, consider the following access policy to a shared resource c. Before using the resource, a lock l must be acquired; the resource must then be used immediately, and the lock must be released immediately after that. If we identify an available resource with an input barb c, a use of c with a synchronization on c and the availability of l with an output barb $\overline{l}$, the above policy can be described by the following formula, where $[c]$ stands for $\neg\langle c\rangle\neg$:

$$\mathit{SafeLock}(l,c) \triangleq \Box^*((\overline{l} \to c)\ \wedge\ [c]\overline{l})\,.$$

Here $\bar{l} \to c = \neg\bar{l} \vee c$ means that presence of the lock implies presence of the resource, while $[c]\bar{l}$ guarantees that after using the resource the lock $\bar{l}$ must be made immediately available. This is a negative formula fitting the format of Theorem 4, hence it is locally checkable. As an example of use of this formula, the process

$$Q = (\nu c, l; SafeLock(l,c))(\bar{l}|c|\bar{a}\langle l,c\rangle) \,|\, a(x,y).!x.(\bar{y}.y|\bar{x})$$

is well typed under a Γ s.t. $\Gamma \vdash a : (x,y)!x.(\bar{y}.y|\bar{x})$. A more flexible version of *SafeLock*, not requiring an immediate release of the lock after using the resource, will be examined in Section 9.

Note that neither (the analog of) *UniRec*(a), nor *SafeLock*(l,c) are covered by the type soundness theorem of [16].

Finally, note that *Resp*(a) and *NoDead*(a) do not fit the format of Theorem 4. Indeed, these formulae are not locally checkable. E.g., consider $R = (\nu a; Resp(a))(c.a|\bar{a})$. This process is easily seen to be well-typed under $c : ()\mathbf{0}$, simply because the c blocking a is masked (turned into τ) in (L-Res). However, $c.a|\bar{a}$ clearly fails to satisfy *Resp*(a).

7. A "Global" Type System

The *Resp*(a) example at the end of the preceding section makes it clear that it is not possible to achieve type soundness for properties like responsiveness unless we drop the "locality" condition in the restriction rule, represented by the use of the hiding operator $\downarrow_{\tilde{x}}$ in $\mathsf{T} \downarrow_{\tilde{x}} \models \Phi$. Similar considerations apply to the case of deadlock-freedom. Those properties are inherently global, that is, they can be checked by looking also at names declared elsewhere in the environment to make sure that they do not interfere with the property being checked. More precisely, it appears that one must also consider the part of the type involving names on which the local (restricted) ones causally depend. In the previous example, where $\mathsf{T} = c.a|\bar{a}$, this means checking *Resp*(a) against $\mathsf{T} \downarrow_{a,c} = \mathsf{T}$, rather than against $\mathsf{T} \downarrow_a$, thus detecting the failure of the property. This can be regarded as an implicit form of assume-guarantee reasoning (this point of view will be further discussed in the concluding section).

Below, we introduce a new type system that pursues this idea. Note that dropping locality implies some loss of compositionality and effectiveness. However, that is done in a somewhat controlled way: not all names, but only some of them, causally related to the restricted ones, are considered when checking the validity of the property. The type system relies on the use of normal forms and dependency graphs, two technical devices introduced in the next subsection which help to keep track of causal dependencies among names in a type. Proofs omitted in this section can be found in Appendix C.

7.1. Dependency graphs and (head) normal forms

Let χ range over the set $\mathfrak{a} = \{\epsilon, \circ, \bullet\}$ of *annotations*. For $I \subseteq \mathcal{N}$, we let a set of annotated names $\hat{I}$ be a total function from I to $\mathfrak{a}$; by slight abuse of notation, we write $a^\chi \in \hat{I}$ rather than $\hat{I}(a) = \chi$. The informal meaning of annotations is: ϵ = free name, $\circ$ = input-bound name, $\bullet$ = restricted name. A *dependency graph* G is a pair $\langle V, E\rangle$, where: $V = \hat{I} \cup W$, with $W \subseteq \{(\nu\tilde{x}) \,\big|\, \tilde{x} \subseteq \mathcal{N}\}$, is a set of annotated names and restrictions, representing *vertices*, and $E \subseteq V \times V$ is a set of *edges*.

A dependency graph $G = \langle V, E\rangle$, with $V = \hat{I} \cup W$ ranged over by $u, v, \ldots$, encodes causal relations among (free or bound) names in I. Vertices of the form $(\nu\tilde{x})$ are introduced for delimiting the scope of restricted names. $(u,v) \in E$ is also written as $u \to_{\mathrm{G}} v$. Each edge (a,b) encodes a direct causal dependence of b from a. The reflexive and transitive closure of $\to_{\mathrm{G}}$, written $\to_{\mathrm{G}}^*$, encodes indirect causal dependencies. A *root* of G is a vertex $u \in V$ such that for no v, $v \rightarrow_{\mathrm{G}} u$; the set of G's roots is denoted by roots(G). Given a dependency graph $G = \langle V, E\rangle$, with $V = \hat{I} \cup W$, a name a is *critical in G with respect to* $\tilde{b}$, if it belongs to the set of names $G(\tilde{b})$ defined below.

$$G(\tilde{b}) \triangleq \{x \,\big|\, x^{\epsilon} \in \hat{I} \wedge \exists\, x \rightarrow_{\mathrm{G}} v_1 \rightarrow_{\mathrm{G}} \cdots \rightarrow_{\mathrm{G}} v_n = b \in \tilde{b} \text{ s.t. } \forall 1 \leq i < n :\ v_i = (\nu\tilde{y}) \text{ implies } b \notin \tilde{y}\} \quad (1)$$

The set of *critical names* in G, written cr(G), is defined as $\mathrm{cr}(G) \triangleq \bigcup_{b^{\bullet} \in \hat{I}} G(b)$. Finally, we define $G[\tilde{b}]$ as $G(\tilde{b}) \cup \tilde{b}$. In order to associate dependency graphs to types, we introduce four auxiliary operations on graphs:

(i) union $G_1 \cup G_2$ is defined componentwise as expected, provided the sets of vertices V_1 and V_2 agree on annotations of common names (otherwise union is not defined);

(ii) χ-update $G \uparrow^{\chi}_{\tilde{x}}$ changes into χ the annotation of all names in $\tilde{x}$ occurring in V;

(iii) a-rooting is defined as $a \rightarrow G \triangleq \langle\, V \cup \{a^{\epsilon}\}, E \cup \{(a,b) \,|\, b \in \mathrm{roots}(G)\}\,\rangle$, where $G = \langle V, E\rangle$, provided a does not occur in V with annotations different from ϵ (otherwise a-rooting is not defined);

(iv) $(\nu\tilde{x})$-rooting is defined as $(\nu\tilde{x}) \rightarrow G \triangleq \langle\, V, E \cup \{((\nu\tilde{x}), b) \,|\, b \in \mathrm{roots}(G)\}\,\rangle$.

We are interested in graphs G_{T} that correspond to types T in *normal form*, which are defined below.

Definition 7 ((head) normal forms). A type is *prime* if it is either of the form $\sum_{i \in I} \mu_i.\mathsf{T}_i$ with $I \neq \emptyset$ or $!a(\mathsf{t}).\mathsf{T}$. A type is in *head normal form* if it is of the form $(\tilde{\nu a})(\mathsf{T}_1|\cdots|\mathsf{T}_n)$ with $n \geq 0$ and the T_i's prime.

A type is in *normal form* if it is in head normal form and each term occurring underneath every prefix is, recursively, in normal form. Processes in (head) normal form are defined similarly.

Every type (and every process) is easily seen to be equivalent to one in normal form, as stated by the lemma below.

Lemma 8. *For each* $\mathsf{T} \in \mathcal{T}$ *there exists a* $\mathsf{S} \in \mathcal{T}$ *in normal form such that* $\mathsf{T} \equiv \mathsf{S}$. *For each* $P \in \mathcal{P}$ *there exist a* $Q \in \mathcal{P}$ *in normal form such that* $P \equiv Q$.

In what follows, we assume the existence of a function NF($\cdot$) that maps each type/process to a structurally congruent one in normal form. For channel types, we set $\mathrm{NF}((\tilde{x} : \tilde{\mathsf{t}})\mathsf{T}) \triangleq (\tilde{x} : \mathrm{NF}(\tilde{\mathsf{t}}))\,\mathrm{NF}(\mathsf{T})$. Note that the normal forms are unique modulo reordering of parallel components and swapping of top-level restrictions originated by applications of the scope extrusion law.

For any T and t in normal form, the dependency graphs G_{T} and G_{t} are defined by mutual induction on the structure of T and t as follows (it is assumed that in T and t bound names are distinct from each other and from free names):

$$G_{a(\mathsf{t}).\mathsf{T}} \;=\; a \rightarrow (G_{\mathsf{t}} \cup G_{\mathsf{T}}) \qquad G_{!a(\mathsf{t}).\mathsf{T}} \;=\; G_{a(\mathsf{t}).\mathsf{T}} \qquad G_{\overline{a}.\mathsf{T}} \;=\; a \rightarrow G_{\mathsf{T}}$$

$$G_{\sum_{i \in I} \mu_i.\mathsf{T}_i} = \bigcup_{i \in I} G_{\mu_i.\mathsf{T}_i} \quad |I| \neq 1 \qquad\qquad G_{\prod_i \mathsf{T}_i} = \bigcup_i G_{\mathsf{T}_i}$$

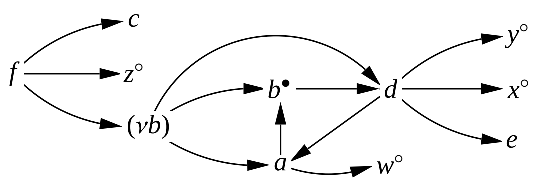
$$G_{(\nu\tilde{x}:\tilde{\mathsf{t}})\mathsf{T}} = (\nu\tilde{x}) \rightarrow ((G_{\mathsf{T}} \cup \bigcup_{\mathsf{t} \in \tilde{\mathsf{t}}} G_{\mathsf{t}}) \uparrow^{\bullet}_{\tilde{x}}) \qquad G_{(\tilde{x}:\tilde{\mathsf{t}})\mathsf{T}} = (G_{\mathsf{T}} \cup \bigcup_{\mathsf{t} \in \tilde{\mathsf{t}}} G_{\mathsf{t}}) \uparrow^{\circ}_{\tilde{x}} \;.$$

In essence, G_{T} encodes potential causal dependencies among – free or bound – names of T, as determined by the nesting of prefixes in T. Note that G_{T} is not expressive enough to describe *conflicts* between causes, e.g. a conflicts with b as a cause of d in $(a.c+b.c)|\overline{c}.d$. Anyway, such degree of accuracy is not necessary for our purposes. In the sequel, given any arbitrary type T, we shall write G_{T} for $G_{\mathrm{NF}(\mathsf{T})}$ (similarly for channel types) and abbreviate $\mathrm{cr}(G_{\mathsf{T}})$ and $G_{\mathsf{T}}[\tilde{b}]$, for any $\tilde{b} \subseteq \mathrm{fn}(\mathsf{T})$, as $\mathrm{cr}(\mathsf{T})$ and $\mathsf{T}[\tilde{b}]$, respectively.

Example 6 (a dependency graph). Consider

$$\mathsf{T} = \overline{f}.c\,|\,f((z:()0)z).(\nu b:(x)\overline{d}.\overline{x})\Big(\overline{a}.b((y)\overline{d}.\overline{y})\,|\,\overline{b}.\overline{d}.\overline{e}+\overline{d}.a((w:()0)w)\Big).$$

G_{T} can be graphically represented as the directed graph below (ϵ-annotations are omitted for the sake of notation).


$$\begin{tikzcd}
 & c & & & y^\circ \\
f \arrow[ur, bend left] \arrow[r] \arrow[dr, bend right] & z^\circ & b^\bullet \arrow[r] & d \arrow[ur, bend left] \arrow[r] \arrow[dr, bend right] \arrow[ddl] & x^\circ \\
 & (\nu b) \arrow[ur, bend left] \arrow[uurr, bend left] \arrow[dr, bend right] & & & e \\
 & & a \arrow[uu] \arrow[r, bend right] & w^\circ &
\end{tikzcd}$$

It is easy to compute the following sets: $\mathrm{cr}(\mathsf{T}) = \{a,d\}$, $\mathsf{T}[e] = \{f,a,d,e\}$ and $\mathsf{T}[c] = \{f,c\}$. Notice that $f \notin \mathrm{cr}(\mathsf{T}) = G_{\mathsf{T}}(b)$. Intuitively, this is the case because b is generated only after f is consumed, as shown by the path $f \longrightarrow (\nu b) \longrightarrow b^\bullet$ (recall the definition of $G(b)$ in (1)).

7.2. Typing rules

We need a more liberal definition of well-annotated process, that allows re-arranging of top-level restrictions before checking annotations. To see why this is necessary, consider $\phi = \Box^*(\Diamond^* a|\Diamond^* a)$, a typical property one would like to check in the new system. Consider the processes $P = (\nu b)(\nu a;\phi)R$ and $Q = (\nu a;\phi)(\nu b)R$, with $R = b.\overline{c}\,|\,b.\overline{d}\,|\,\overline{b}\,|\,\overline{b}\,|\,c.a\,|\,d.a$. We observe that $(\nu b)R \not\models \phi$, so that Q is not well-annotated according to Definition 5. On the other hand, $Q \equiv P$ and $R \models \phi$, which suggests that P, hence Q, could be considered as well-annotated up to a swapping of (νa) and (νb) obtained by applying the scope extrusion structural law twice. Recall that in general it is not possible to swap restrictions $(\nu\tilde{x})$ and $(\nu\tilde{y})$ in $(\nu\tilde{x})(\nu\tilde{y})P$, as already discussed on Section 4, Example 5.

Definition 8 (globally well-annotated processes). A process $P \in \mathcal{P}$ is *globally well-annotated* if whenever $P \equiv (\tilde{\nu}\tilde{b})(\nu\tilde{a};\,\Phi)(\tilde{\nu}\tilde{c})Q$, with Q a parallel composition of prime processes, then there is a permutation $\tilde{b}'\,\tilde{c}'$ of $\tilde{b}\,\tilde{c}$ such that $P \equiv (\tilde{\nu}\tilde{b}')(\nu\tilde{a};\,\Phi)(\tilde{\nu}\tilde{c}')Q$ and $(\tilde{\nu}\tilde{c}')Q \models \Phi$.

A channel type $(\tilde{x}:\tilde{\mathsf{t}})\mathsf{T}$ is said to be *well-formed* if $\tilde{x}\#\mathrm{cr}(\mathsf{T})$; in what follows, we only consider contexts Γ containing well-formed channel types. E.g. we discard channel types of the form $(x:\mathsf{t})(\nu c)(\overline{x}.c\,|\,\mathsf{S})$. For any type T we let $\mathsf{T}\Downarrow_{\tilde{x}}$ denote $\mathsf{T}\downarrow_{\mathsf{T}[\tilde{x}]}$ (note that $\mathrm{fn}(\mathsf{T}\Downarrow_{\tilde{x}}) \cup \tilde{x} = \mathsf{T}[\tilde{x}]$ by definition). Intuitively, in $\mathsf{T}\Downarrow_{\tilde{x}}$, we keep the names in $\tilde{x}$ and those that are causes of $\tilde{x}$ in T; the others are masked.

The global type system is reported in Table 6. Recall that in each rule the context Γ is assumed to contain only well-formed channel types as discussed above. The type system makes use of an auxiliary *property-type simulation* relation $\propto_{\tilde{x}}$ among P-sets and types, defined coinductively

$$(\text{G-Inp})\ \frac{\Gamma \vdash a : (\tilde{x} : \tilde{\mathsf{t}})\mathsf{T} \quad \Gamma, \tilde{x} : \tilde{\mathsf{t}} \vdash P : \mathsf{T}|\mathsf{T}' \quad \tilde{x}\#\mathsf{T}'}{\Gamma \vdash a(\tilde{x}).P : a((\tilde{x} : \tilde{\mathsf{t}})\mathsf{T}).\mathsf{T}'} \qquad (\text{G-Tau})\ \frac{\Gamma \vdash P : \mathsf{T}}{\Gamma \vdash \tau.P : \tau.\mathsf{T}}$$

$$(\text{G-Sum})\ \frac{|I| \neq 1 \quad \forall i \in I :\ \Gamma \vdash \alpha_i.P_i : \mu_i.\mathsf{T}_i}{\Gamma \vdash \sum_i \alpha_i.P_i : \sum_i \mu_i.\mathsf{T}_i} \qquad (\text{G-Res})\ \frac{\Gamma, \tilde{a} : \tilde{\mathsf{t}} \vdash P : \mathsf{T} \quad \Phi \propto_{\tilde{a}} \mathsf{T}}{\Gamma \vdash (\nu \tilde{a} : \tilde{\mathsf{t}}; \Phi)P : (\nu \tilde{a} : \tilde{\mathsf{t}})\mathsf{T}}$$

$$(\text{G-Rep})\ \frac{\Gamma \vdash a(\tilde{x}).P : a((\tilde{x} : \tilde{\mathsf{t}})\mathsf{T}).\mathsf{T}' \quad \mathrm{cr}(a((\tilde{x} : \tilde{\mathsf{t}})\mathsf{T}).\mathsf{T}') = \emptyset}{\Gamma \vdash !a(\tilde{x}).P : !a((\tilde{x} : \tilde{\mathsf{t}})\mathsf{T}).\mathsf{T}'} \qquad (\text{G-Eq-P})\ \frac{\Gamma \vdash P : \mathsf{T} \quad P \equiv Q}{\Gamma \vdash Q : \mathsf{T}}$$

$$(\text{G-Par})\ \frac{\Gamma \vdash P : \mathsf{T} \quad \Gamma \vdash Q : \mathsf{S} \quad \mathrm{cr}(\mathsf{T})\#\mathsf{S} \quad \mathrm{cr}(\mathsf{S})\#\mathsf{T}}{\Gamma \vdash P|Q : \mathsf{T}|\mathsf{S}} \qquad (\text{G-Eq})\ \frac{\Gamma \vdash P : \mathsf{T} \quad \mathsf{T} \equiv \mathsf{S}}{\Gamma \vdash P : \mathsf{S}}$$

$$(\text{G-Out})\quad \frac{\Gamma \vdash a : (\tilde{x} : \tilde{\mathsf{t}})\mathsf{T} \quad \Gamma \vdash \tilde{b} : \tilde{\mathsf{t}} \quad \Gamma \vdash P : \mathsf{S} \quad \tilde{b}\#\mathrm{cr}(\mathsf{T}) \quad \mathrm{cr}(\mathsf{T}[^{\tilde{b}}/_{\tilde{x}}])\#\mathsf{S} \quad \mathsf{T}[^{\tilde{b}}/_{\tilde{x}}]\#\mathrm{cr}(\mathsf{S})}{\Gamma \vdash \overline{a}\langle \tilde{b} \rangle.P : \overline{a}.(\mathsf{T}[^{\tilde{b}}/_{\tilde{x}}] \,|\, \mathsf{S})}$$

Table 6: Typing rules for the global system.

below (the use of this relation will be explained in the sequel). We first need some additional notations. A labeled transition relation on types is defined as expected: we write $\mathsf{T} \xrightarrow{a} \mathsf{T}'$ if $\mathsf{T} \equiv (\tilde{\nu d})(\sum_i \mu_i.\mathsf{T}_i + a(\mathrm{t}).\mathsf{T}''|\mathsf{S})$ or $\mathsf{T} \equiv (\tilde{\nu d})(!a(\mathrm{t}).\mathsf{T}''|\mathsf{S})$ and $\mathsf{T} \xrightarrow{\overline{a}} \mathsf{T}'$ if $\mathsf{T} \equiv (\tilde{\nu d})(\sum_i \mu_i.\mathsf{T}_i + \overline{a}.\mathsf{T}''|\mathsf{S})$, where in both cases $\mathsf{T}' \equiv (\tilde{\nu d})(\mathsf{T}''|\mathsf{S})$. In the following, we let γ range over $\langle \epsilon \rangle$, $\langle a \rangle$, a and $\overline{a}$, and define $a \downarrow_{\tilde{x}} = \overline{a} \downarrow_{\tilde{x}} = \langle \epsilon \rangle$ if $a \notin \tilde{x}$. Moreover, we write $\mathsf{T} >^{\gamma}_{\tilde{z}} \mathsf{T}'$ if

- either $\mathsf{T} \xrightarrow{\gamma} \mathsf{T}'$ with $\gamma ::= \langle \epsilon \rangle \,\big|\, \langle a \rangle$, for some $a \in \tilde{z}$
- or $\mathsf{T} \xrightarrow{\gamma} \mathsf{T}'$ with $\gamma ::= a \,\big|\, \overline{a}$ and $a \notin \tilde{z}$.

Intuitively, $\mathsf{T} >^{\gamma}_{\tilde{z}} \mathsf{T}'$ means that T can move to T' with a τ-action after hiding names in $\tilde{z}$.

Definition 9 (property-type simulation, $\propto_{\tilde{x}}$). We let *property-type simulation*, $\propto_{\tilde{x}}$, be the largest relation between P-sets and types such that whenever $\Phi \propto_{\tilde{x}} \mathsf{T}$ then $\mathrm{Ok}(\Phi)$ and:

1. $\mathsf{T} \Downarrow_{\tilde{x}} \models \Phi$;
2. for each γ, T' such that $\mathsf{T} >^{\gamma}_{\mathsf{T}[\tilde{x}]} \mathsf{T}'$ we have $\Phi_{\gamma\downarrow_{\mathsf{T}[\tilde{x}]}} \propto_{\tilde{x}} \mathsf{T}'$.

In the type system, we note the presence of a new structural rule for processes, (G-Eq-P) forcing subject congruence: this is not derivable from the other rules of the system. As an example, while $P = (\nu a : \mathrm{t};\ Resp(a))(b.a|\overline{b}|\overline{a})$ *can* be typed without using rule (G-Eq-P), the structurally congruent process $(\nu a : \mathrm{t};\ Resp(a))(b.a|\overline{a})|\overline{b}$ *cannot* be typed without using that rule. The condition on critical names in rule (G-Par) ensures that any Q put in parallel to P will not break well-annotated-ness of P (and vice-versa). In other words, the condition ensures that Q will not interfere with properties decorating restrictions $(\nu\tilde{a})$ inside P, as Q does not contain $\tilde{a}$-critical names. As an example, the condition prevents us from putting process $Q = b$ in parallel with P above: indeed, the resulting $P|b \equiv (\nu a : \mathrm{t}; \mathit{Resp}(a))(b.a|\overline{b}|\overline{a}|b)$ is not well-annotated. A similar remark applies to the rules for output and replication. In rule (G-Res), the property-type simulation relation $\propto_{\tilde{a}}$ ensures that each derivative of T satisfies the corresponding derivative of Φ, and this will be crucial in the proof of the substitution lemma and of the subject reduction theorem. It is worth noticing that checking $\Phi \propto_{\tilde{a}} \mathsf{T}$ might be undecidable, given that in general T might be infinite-state: at the end of the next section, however, we will identify a class of formulae for which checking $[\![\phi]\!] \propto_{\tilde{a}} \mathsf{T}$ can be done by checking the validity of $\mathsf{T} \Downarrow_{\tilde{a}} \models \phi$ (Proposition 10).

The judgements derivable in the new type system are written as $\Gamma \vdash_{\mathrm{G}} P : \mathsf{T}$.

Example 7. Consider the property $\phi = \Box^*(\Diamond^* a|\Diamond^* a)$ and the process P defined at the beginning of this subsection (for the sake of readability we omit channel types in annotations, which are always $()0$):

$$P = (\nu b)(\nu a; \phi)R \text{ with } R = b.\overline{c}\,|\,b.\overline{d}\,|\,\overline{b}\,|\,\overline{b}\,|\,c.a\,|\,d.a\,.$$

Assume that $\Gamma \vdash c, d : ()0$. It follows that $\Gamma \vdash_{\mathrm{G}} P : (\nu b)(\nu a)\mathsf{T}$, with $\mathsf{T} = b.\overline{c}\,|\,b.\overline{d}\,|\,\overline{b}\,|\,\overline{b}\,|\,c.a\,|\,d.a$. Indeed, by applications of typing for prefixes followed by applications of rule (G-Par), it can be deduced that $\Gamma, a : ()0, b : ()0 \vdash_{\mathrm{G}} R : \mathsf{T}$. Then by (G-Res) and $\mathsf{T} \Downarrow_a = \mathsf{T} \models \phi$ it follows that $\Gamma, b : ()0 \vdash_{\mathrm{G}} (\nu a; \phi)R : (\nu a)\mathsf{T}$. Finally, by applying (G-Res) again, $\Gamma \vdash_{\mathrm{G}} P : (\nu b)(\nu a)\mathsf{T}$.

7.3. Basic properties

The basic properties of the local type system carry over to the global one. In particular, we have a form of normal or syntax-directed derivations. The typing rules introduced in Table 6 are not syntax-directed. E.g. well-typedness of process $P = ((\nu a : \mathrm{t}; \mathbf{T})a)|b$ under $\Gamma \vdash_{\mathrm{L}} b : ()0$ can be derived either by applying the typing rules as dictated by the structure of P or by applying the rules as dictated by the structure of $(\nu a : \mathrm{t}; \mathbf{T})(a|b)$ followed by an application of rule (G-Eq-P). This is the case because applications of (G-Eq-P) and (G-Par) can be freely intertwined. We get syntax-directed derivations by disciplining the way these rules are used. Consider the rule

$$(\text{G-Par}^+)\quad \frac{\begin{array}{c}P = P_1|\cdots|P_n \quad n > 1 \quad P_i \text{ prime for each } i\\ \text{for each } i: \ \Gamma \vdash P_i : \mathsf{T}_i \quad \text{for each } i \neq j: \ \mathrm{cr}(\mathsf{T}_i)\#\mathsf{T}_j\end{array}}{\Gamma \vdash P : \mathsf{T}_1|\cdots|\mathsf{T}_n}\,.$$

Let us denote by $\Gamma \vdash^+_{\mathrm{G}} P : \mathsf{T}$ judgments that can be derived by applying the typing rules in Table 6 with (G-Par) replaced by (G-Par$^+$). We next prove the existence of normal derivations in $\vdash^+_{\mathrm{G}}$. A *normal* derivation of $\Gamma \vdash^+_{\mathrm{G}} P : \mathsf{T}$ is one where (G-Eq) can only be found immediately above (G-Inp), and (G-Eq-P) can only be applied with a process in head normal form in the premise and one *not* in head normal form in the conclusion. We write $\Gamma \vdash^+_{\mathrm{NG}} P : \mathsf{T}$ for judgments that can be derived in the new system with a normal derivation. The following proposition asserts that, modulo $\equiv$ on both processes and types, every judgment derivable in the system $\vdash^+_{\mathrm{G}}$ admits a normal derivation.

Proposition 7 (normal derivation). *$\Gamma \vdash^+_{\mathrm{G}} P : \mathsf{T}$ implies that there are $R \equiv P$ and $\mathsf{S} \equiv \mathsf{T}$ such that R and S are in head normal form and $\Gamma \vdash^+_{\mathrm{NG}} R : \mathsf{S}$.*

Reconsidering the example at the beginning of this subsection, we see that the only normal derivation in $\vdash^{+}_{\mathrm{G}}$ for deducing well-typedness of a normal form of P (that is of $P = ((\nu a : \mathsf{t}; \mathbf{T})a)|b)$ is the second one described at the beginning of this subsection.

$\vdash^{+}_{\mathrm{NG}}$-derivations are syntax-directed, in the sense that given P, there is at most an instance of one rule where P can appear in the conclusion. This fact can be again exploited to show that processes and their types share the same shallow structure. This is expressed by the following lemma, saying that it is possible to infer the structure of a type from that of the process (a dual of this result can also be proven but is not necessary for our purposes).

Lemma 9. $\Gamma \vdash^{+}_{\mathrm{NG}} (\tilde{\nu}\tilde{a} : \tilde{\mathsf{t}})(\nu\tilde{b} : \tilde{\mathsf{t}}'; \Phi)P : \mathsf{T}$ *implies* $\mathsf{T} = (\tilde{\nu}\tilde{a} : \tilde{\mathsf{t}})(\nu\tilde{b} : \tilde{\mathsf{t}}')\mathsf{S}$, *with* $\Gamma, \tilde{a} : \tilde{\mathsf{t}}, \tilde{b} : \tilde{\mathsf{t}}' \vdash^{+}_{\mathrm{NG}} P : \mathsf{S}$ *and* $\Phi \propto_{\tilde{b}} \mathsf{S}$.

It is an easy matter to prove that systems $\vdash_{\mathrm{G}}$ and $\vdash^{+}_{\mathrm{G}}$ are equivalent.

Proposition 8. $\Gamma \vdash_{\mathrm{G}} P : \mathsf{T}$ *if and only if* $\Gamma \vdash^{+}_{\mathrm{G}} P : \mathsf{T}$.

Finally, we have subject reduction. The proof of the theorem is not completely standard. In particular, it is convenient to reason with $\vdash^{+}_{\mathrm{G}}$ rather than with $\vdash_{\mathrm{G}}$, as the existence of normal derivations $\vdash^{+}_{\mathrm{NG}}$ makes the reasoning simpler. The idea is to consider a normal derivation $\Gamma \vdash^{+}_{\mathrm{NG}} Q : \mathsf{S}$ for a Q in head normal form and congruent to the original P, and S congruent to T. Lemma 9 ensures that Q and the associated type S share the same shallow spatial structure, thus any reduction from Q can be simulated by one from S.

Theorem 5 (subject reduction). $\Gamma \vdash_{\mathrm{G}} P : \mathsf{T}$ *and* $P \xrightarrow{\lambda} P'$ *implies that there exists a* T' *such that* $\mathsf{T} \xrightarrow{\lambda} \mathsf{T}'$ *and* $\Gamma \vdash_{\mathrm{G}} P' : \mathsf{T}'$.

8. Type Soundness for the Global System

Similarly to the local case, we firstly identify a general class of properties for which, at least in principle, model checking on well-typed processes can be reduced to a type checking problem whose solution requires only model checking on types. Then we give sufficient syntactic conditions for global-checkability. Proofs not reported in this section can be found in Appendix D. The definition of *globally checkable properties* is the same as the local one, except that the local hiding operator "$\downarrow_{\tilde{x}}$" is replaced by "$\Downarrow_{\tilde{x}}$":

Definition 10 (globally checkable properties). We let Gc be the largest predicate on P-sets such that whenever $\mathrm{Gc}(\Phi)$ then $\mathrm{Ok}(\Phi)$ and:

1. whenever $\Gamma \vdash_{\mathrm{G}} P : \mathsf{T}$ and $\tilde{x} \supseteq \mathrm{supp}(\Phi)$ and $\mathsf{T} \Downarrow_{\tilde{x}} \models \Phi$ then $P \models \Phi$;
2. $\mathrm{Gc}(\Phi_\lambda)$ holds for each λ.

If $\mathrm{Gc}(\Phi)$ then we say Φ is *globally checkable*.

A formula $\phi \in \mathcal{F}$ is said to be globally checkable if $[[\phi]]$ is globally checkable. The following result is quite expected. The subsequent corollary is a consequence of type soundness and of subject reduction.

Theorem 6 (type soundness). *Suppose* $\Gamma \vdash_{\mathrm{G}} P : \mathsf{T}$ *and P is decorated with globally checkable P-sets only. Then P is globally well-annotated.*

Proof. Assume $\Gamma \vdash_{\mathrm{G}} P : \mathsf{T}$ and $P \equiv (\tilde{\nu}\tilde{b})(\nu\tilde{a} : \tilde{\mathrm{t}}; \Phi)(\tilde{\nu}\tilde{c})Q$, with Q a parallel composition of prime processes. The P-set Φ is globally checkable by hypothesis. We have to prove that there is a permutation $\tilde{b}'\tilde{c}'$ of $\tilde{b}\tilde{c}$ such that $P \equiv (\tilde{\nu}\tilde{b}')(\nu\tilde{a} : \tilde{\mathrm{t}}; \Phi)(\tilde{\nu}\tilde{c}')Q$ and $(\tilde{\nu}\tilde{c}')Q \models \Phi$.

By Proposition 8, $\Gamma \vdash^{+}_{\mathrm{G}} P : \mathsf{T}$ and by Proposition 7, there are $R \equiv P$ and $\mathsf{S} \equiv \mathsf{T}$ such that R is in head normal form and $\Gamma \vdash^{+}_{\mathrm{NG}} R : \mathsf{S}$.

$P \equiv (\tilde{\nu}\tilde{b})(\nu\tilde{a} : \tilde{\mathrm{t}}; \Phi)(\tilde{\nu}\tilde{c})Q$ and $P \equiv R$ and R in head normal form imply $R = (\tilde{\nu}\tilde{b}')(\nu\tilde{a} : \tilde{\mathrm{t}}; \Phi)(\tilde{\nu}\tilde{c}')Q'$, with $\tilde{b}', \tilde{c}'$ permutations of $\tilde{b}, \tilde{c}$ and Q' parallel composition of prime processes such that $Q' \equiv Q$. By Lemma 9, $\mathsf{S} = (\tilde{\nu}\tilde{b}')(\nu\tilde{a} : \tilde{\mathrm{t}})\mathsf{U}$ with $\Gamma, \tilde{b}' : \tilde{\mathrm{t}}', \tilde{a} : \tilde{\mathrm{t}} \vdash^{+}_{\mathrm{NG}} (\tilde{\nu}\tilde{c}')Q' : \mathsf{U}$ and $\Phi \propto_{\tilde{a}} \mathsf{U}$. Therefore $\mathsf{U} \Downarrow_{\tilde{a}} \models \Phi$. By Definition 10, it follows that $(\tilde{\nu}\tilde{c}')Q' \models \Phi$, hence $(\tilde{\nu}\tilde{c}')Q \models \Phi$.

Corollary 2 (run-time soundness). *Suppose that $\Gamma \vdash_{\mathrm{G}} P : \mathsf{T}$ and that P is decorated with globally checkable P-sets only. Then $P \rightarrow^{*} P'$ implies that P' is globally well-annotated.*

Like in the local case, we can give syntactic conditions for a formula to be globally checkable. We need some intermediate results. First we note that well-typedness in $\vdash_{\mathrm{G}}$ implies well-typedness in the kernel system $\vdash_{\mathrm{K}}$.

Proposition 9. $\Gamma \vdash_{\mathrm{G}} P : \mathsf{T}$ *implies* $\Gamma \vdash_{\mathrm{K}} P : \mathsf{T}$.

Proof. This result can be proved by an easy inspection of the typing rules of $\vdash_{\mathrm{G}}$. Notice that in case (G-Eq-P) is the last rule applied, one exploits the $\vdash_{\mathrm{K}}$'s version of Proposition 3 (subject congruence). Indeed, if the premise is $P \equiv Q$ and $\Gamma \vdash_{\mathrm{G}} Q : \mathsf{T}$, by induction one can infer $\Gamma \vdash_{\mathrm{K}} Q : \mathsf{T}$ and by Proposition 3 $\Gamma \vdash_{\mathrm{K}} P : \mathsf{T}$.

Proposition 6 carries over to the global system as a corollary of the previous result.

Corollary 3. *Suppose $\Gamma \vdash_{\mathrm{G}} P : \mathsf{T}$ and $\phi \in \mathcal{F}$. Then $\mathsf{T} \models \phi$ if and only if $P \models \phi$.*

Lemma 10. *Let $\phi \in \mathcal{F}_{\tilde{x}}$ be of the form $\phi = \Box^{*}_{-\tilde{a}}\psi$ with negation not occurring underneath any $\langle -\tilde{b}\rangle^{*}$ in ψ. Then for any T, $\mathsf{T} \Downarrow_{\tilde{x}} \models \phi$ implies $\mathsf{T} \models \phi$.*

As a consequence of Lemma 3, Lemma 10 and Corollary 3 we get the following result.

Theorem 7. *Let ϕ be one of the following forms: (a) $\Box^{*}\psi$ with negation not occurring underneath any $\langle -\tilde{a}\rangle^{*}$ in ψ, or (b) $\Box^{*}_{-\tilde{a}}\Diamond^{*}\psi'$, with negation not occurring in ψ'. Then ϕ is globally checkable.*

Proof. Each formula of the given forms is Ok (Lemma 3). Assume $\Gamma \vdash_{\mathrm{G}} P : \mathsf{T}$. By Lemma 10 and Corollary 3 we get $\mathsf{T} \Downarrow_{\tilde{x}} \models \phi$ implies $P \models \phi$. That is, condition 1 of the definition of Gc is satisfied by formulae of the form *(a)* or *(b)*. It remains to show that $\mathrm{Gc}(\phi_\lambda)$ holds for each λ. We distinguish the two cases, *(a)* or *(b)*.

(a). $[\![\phi]\!]_\lambda = [\![\phi]\!]$ for each λ. This, together with the above considerations, entails that $[\![\phi]\!] \in \mathrm{Gc}$.

(b). If $[\![\Box^{*}_{-\tilde{a}}\Diamond^{*}\psi']\!] = \emptyset$ then $[\![\Box^{*}_{-\tilde{a}}\Diamond^{*}\psi']\!]_\lambda = \emptyset$. Otherwise, $[\![\Box^{*}_{-\tilde{a}}\Diamond^{*}\psi']\!] = [\![\Box^{*}_{-\tilde{a}}\Diamond^{*}\psi']\!]_\lambda$ for each $\lambda\#\tilde{a}$ and $[\![\Box^{*}_{-\tilde{a}}\Diamond^{*}\psi']\!]_\lambda = \mathcal{U}$ otherwise (see the proof of Lemma 3, Appendix A, for the details). This shows that

$$\{\emptyset, \mathcal{U}\} \cup \{[\![\Box^{*}_{-\tilde{a}}\Diamond^{*}\psi']\!] \mid \psi' \text{ does not contain negations}\} \subseteq \mathrm{Gc}\,.$$

The following proposition guarantees that for formulae that satisfy the hypotheses of Theorem 7 checking $[\![\phi]\!] \propto_{\tilde{a}} \mathsf{T}$ reduces to checking $\mathsf{T} \Downarrow_{\tilde{a}} \models \phi$.

Lemma 11. *Suppose $\phi \in \mathcal{F}_{\tilde{x}}$ with negation not occurring underneath any $\langle -\tilde{y} \rangle^*$ in ϕ. $\mathsf{T} \downarrow_{\tilde{w}} \models \phi$ and $\tilde{w} \supseteq \mathsf{T}[\tilde{x}]$ imply $\mathsf{T} \downarrow_{\mathsf{T}[\tilde{x}]} \models \phi$.*

Lemma 12. $\mathsf{T} \overset{\lambda}{\to} \mathsf{T}'$ *implies* $\mathsf{T}[\tilde{x}] \supseteq \mathsf{T}'[\tilde{x}]$.

Proposition 10. *Let $\phi \in \mathcal{F}_{\tilde{x}}$ be of the form (a) or (b) as specified in Theorem 7. If $\mathsf{T} \Downarrow_{\tilde{x}} \models \phi$ then $[\![\phi]\!] \propto_{\tilde{x}} \mathsf{T}$.*

Proof.

(*a*). $\Box^* \psi$, with negation not occurring underneath any $\langle -\tilde{a} \rangle^*$ in ψ. Define

$$\mathcal{R} \triangleq \{([\![\phi]\!], \mathsf{T}) \,\big|\, \phi = \Box^* \psi \text{ with negation not occurring underneath any } \langle -\tilde{a} \rangle^* \text{ in } \psi,\ \mathsf{T} \Downarrow_{\tilde{x}} \models [\![\phi]\!]\}.$$

It is enough to prove that $\mathcal{R} \subseteq \propto_{\tilde{x}}$. We have to prove that for each $(\Phi, \mathsf{T}) \in \mathcal{R}$ it holds that

1. $\mathsf{T} \Downarrow_{\tilde{x}} \models \Phi$ (this holds by definition of $\mathcal{R}$);
2. for each γ, T' such that $\mathsf{T} >^{\gamma}_{\mathsf{T}[\tilde{x}]} \mathsf{T}'$ it holds that $(\Phi_{\gamma \downarrow_{\mathsf{T}[\tilde{x}]}}, \mathsf{T}') \in \mathcal{R}$.
   By definition, for each λ, $\Phi_\lambda = \Phi$ because $[\![\Box^* \phi]\!] = [\![\phi]\!] = \Phi$. By $\mathsf{T} >^{\gamma}_{\mathsf{T}[\tilde{x}]} \mathsf{T}'$ we get $\mathsf{T} \overset{\gamma}{\to} \mathsf{T}'$ with either $\gamma ::= \langle \epsilon \rangle \,\big|\, \langle b \rangle$, for some $b \in \mathsf{T}[\tilde{x}]$ or $\gamma ::= c \,\big|\, \overline{c}$, for some $c \notin \mathsf{T}[\tilde{x}]$. Hence, by Proposition 5,
   $$\mathsf{T} \downarrow_{\mathsf{T}[\tilde{x}]} \xrightarrow{\gamma \downarrow_{\mathsf{T}[\tilde{x}]}} \mathsf{T}' \downarrow_{\mathsf{T}[\tilde{x}]} .$$
   By $\mathsf{T} \downarrow_{\mathsf{T}[\tilde{x}]} \models \Phi$ we have $\mathsf{T}' \downarrow_{\mathsf{T}[\tilde{x}]} \models \Phi_{\gamma \downarrow_{\mathsf{T}[\tilde{x}]}} = \Phi$. By Lemma 11 and $\mathsf{T}[\tilde{x}] \supseteq \mathsf{T}'[\tilde{x}]$ (Lemma 12) we get $\mathsf{T}' \downarrow_{\mathsf{T}'[\tilde{x}]} \models \Phi_{\gamma \downarrow_{\mathsf{T}[\tilde{x}]}} = \Phi$. Therefore, $(\Phi, \mathsf{T}') \in \mathcal{R}$.

(*b*). $\Box^*_{-\tilde{a}} \Diamond^* \psi'$, with negation not occurring in ψ'.
Define

$$\mathcal{R} \triangleq \{([\![\phi]\!], \mathsf{T}) \,\big|\, \phi = \Box^*_{-\tilde{a}} \Diamond^* \psi' \text{ with negation not occurring in } \psi',\ \mathsf{T} \Downarrow_{\tilde{x}} \models [\![\phi]\!]\} \cup \{(\mathcal{U}, \mathsf{T}) \,\big|\, \mathsf{T} \in \mathcal{T}\}.$$

It is enough to prove that $\mathcal{R} \subseteq \propto_{\tilde{x}}$. We have to prove that for each $(\Phi, \mathsf{T}) \in \mathcal{R}$ it holds that

1. $\mathsf{T} \Downarrow_{\tilde{x}} \models \Phi$ (this holds by definition of $\mathcal{R}$);
2. for each γ, T' such that $\mathsf{T} >^{\gamma}_{\mathsf{T}[\tilde{x}]} \mathsf{T}'$ it holds that $(\Phi_{\gamma \downarrow_{\mathsf{T}[\tilde{x}]}}, \mathsf{T}') \in \mathcal{R}$.
   By definition of $[\![\cdot]\!]$, $\Phi_{\gamma \downarrow_{\mathsf{T}[\tilde{x}]}} = \Phi$ for each $\gamma ::= \langle \epsilon \rangle \,\big|\, \langle y \rangle \,\big|\, b \,\big|\, \overline{b}$, with $y \notin \tilde{a}$ and $b \notin \mathsf{T}[\tilde{x}]$. Moreover, by $\mathsf{T} >^{\gamma}_{\mathsf{T}[\tilde{x}]} \mathsf{T}'$ we get $\mathsf{T} \overset{\gamma}{\to} \mathsf{T}'$. Hence, by Proposition 5,
   $$\mathsf{T} \downarrow_{\mathsf{T}[\tilde{x}]} \xrightarrow{\gamma \downarrow_{\mathsf{T}[\tilde{x}]}} \mathsf{T}' \downarrow_{\mathsf{T}[\tilde{x}]} .$$
   By the latter and $\mathsf{T} \downarrow_{\mathsf{T}[\tilde{x}]} \models \Phi$ we get $\mathsf{T}' \downarrow_{\mathsf{T}[\tilde{x}]} \models \Phi_{\gamma \downarrow_{\mathsf{T}[\tilde{x}]}} = \Phi$ and by Lemma 11 and $\mathsf{T}[\tilde{x}] \supseteq \mathsf{T}'[\tilde{x}]$ (Lemma 12) we get $\mathsf{T}' \downarrow_{\mathsf{T}'[\tilde{x}]} \models \Phi_{\gamma \downarrow_{\mathsf{T}[\tilde{x}]}} = \Phi$. Therefore, $(\Phi, \mathsf{T}') \in \mathcal{R}$.
   Now, suppose $\gamma = \langle y \rangle$, for some $y \in \tilde{a} \subseteq \tilde{x}$. As shown in the proof of Lemma 3 in Appendix A, $\Phi_{\langle y \rangle} = \mathcal{U} = [\![\mathbf{T}]\!]$ and $(\mathcal{U}, \mathsf{T}') \in \mathcal{R}$ by definition.

9. Examples in the Global System

All the properties defined in Example 1 fit the format of Theorem 7, hence they are globally checkable. As an example, consider $P = (\nu a : \mathit{Resp}(a))(\overline{c}\langle a\rangle)|Q$, where $Q = !c(x).(\overline{x}|x)|\overline{c}\langle b\rangle$. Under a suitable Γ, we derive $\Gamma \vdash_{\mathrm{G}} \overline{c}\langle a\rangle|Q : \overline{c}.(\overline{a}|a)|!c|\overline{c}.(\overline{b}|b) \triangleq \mathsf{T}$. Since $\mathsf{T} \Downarrow_a = \overline{c}.(\overline{a}|a)|!c|\overline{c}.(\tau|\tau) \models \mathit{Resp}(a)$, by (G-Res), we get $\Gamma \vdash_{\mathrm{G}} (\nu a : \mathit{Resp}(a))(\overline{c}\langle a\rangle|Q) : (\nu a)\mathsf{T}$, hence we can conclude that $\Gamma \vdash_{\mathrm{G}} P : (\nu a)\mathsf{T}$ using (G-Eq-P).

For another example, consider a somewhat more realistic variant of the *SafeLock* property introduced in Section 6. The new property, *SafeLockExt*, defines an access policy for a shared resource c. Before using the resource, a lock l must be acquired; resource c must then be used immediately, and the lock must be released (not necessarily immediately) after that:

$$\mathit{SafeLockExt}(l,c) \triangleq \Box^*((\overline{l} \to c) \ \ \wedge \ \ [c]\Diamond^*\overline{l}) \; .$$

This is a formula fitting the format of Theorem 7, hence it is globally checkable. As an example of use of this formula, the process

$$Q = (\nu c,d,l; \mathit{SafeLockExt}(l,c))\Big(\overline{l}|c|d|\overline{a}\langle l,c,d\rangle \,|\, a(x,y,z).(\tau.x.\overline{z} + \tau.x.\overline{y}.\overline{z}.(\overline{x}|y|z))\Big)$$

is well typed under a Γ s.t. $\Gamma \vdash a : (x,y,z)(\tau.x.\overline{z} + \tau.x.\overline{y}.\overline{z}.(\overline{x}|y|z))$.

It is worth to notice that (the analogs of) responsiveness and deadlock freedom are not covered by the type soundness theorem of [16]. In the case of deadlock freedom, though, a soundness result can still be proven by ad-hoc reasoning on certain basic properties of the system.

10. Discussion

We discuss here some limitations, and possible workarounds, of our approach, and contrast them with the generic type system approach of [16]. Generally speaking, these limitations arise from design choices that, on the one hand, reduce the flexibility of the systems and, on the other hand, allow to gain in precision and to widen the class of properties for which type soundness can be proven (e.g., the class includes interesting liveness properties).

A first point is the uniform behavior of input continuations imposed by our systems, which is somehow reminiscent of uniformity in Sangiorgi's receptiveness work [23]. Indeed, a process like $R = a(x).x\,|\,a(x).x.x\,|\,\overline{a}\langle b\rangle$ is discarded in both our systems, but is well typed in [16], for example assuming a of type $(x)(x\&x.x)$, which says that the input continuation of a can be either of type x or $x.x$. The absence of union types in our system is a design choice motivated by our search of type abstractions spatial correspondent to processes. Indeed, suppose we could assign type $(x)(x\&x.x)$ to a. This would lead to assigning R the type $a\,|\,a\,|\,\overline{a}.(b + b.b)$: here the spatial correspondence breaks down after one reduction.

A second point is that, in [16], the subtyping relation makes an essential use of a "sub-divide" law, $\mathsf{T} \equiv \mathsf{T}\uparrow_{\tilde{x}} |\mathsf{T}\downarrow_{\tilde{x}}$. This rule allows one to split *any* type into a part depending only on $\tilde{x}$, $\mathsf{T}\downarrow_{\tilde{x}}$, and a part not depending on $\tilde{x}$, $\mathsf{T}\uparrow_{\tilde{x}}$. As an example, one has $a.b.\overline{x} \equiv a.b.\tau|\tau.\tau.\overline{x}$. This law enhances the flexibility of the input rule, hence of the type system.

An example of a process that cannot be handled in our type systems because of the absence of the "sub-divide" law is

$$P = a(x).b(y).\overline{x}.\overline{y} \; .$$

Here, y causally depends on x: this makes the type of b depend on a bound name x, which cannot be expressed in our system. With a sub-divide law, the type of b's continuation $\overline{x}.\overline{y}$ can be decomposed as $\overline{x}.\tau|\tau.\overline{y}$, thus allowing one to assign b the type $(y)\tau.\overline{y}$ that ignores the dependency on x. Note that this specific problem does not arise in the sub-calculus enforcing input locality and asynchronous outputs, which is considered to be reasonably expressive. E.g. the process $a(x).b(y).(\overline{x}|\overline{y})$ is typable in our systems. For another, subtler example, consider the process

$$Q = !a(x).(\nu c)\Big(b(y).(\,(\nu z)(\overline{c}\langle x,z\rangle\,|\,z.\overline{y})\,|\,c(x,z).(\overline{x}|\overline{z})\,)\Big)\,.$$

Here, a can be viewed as a service invocation channel, x as a formal invocation parameter and y as an acknowledgement channel, introduced by another input (on b). It appears that y and x are related (via c), which makes the type of b dependent on the bound name x, which cannot be expressed in our system. This dependency could be discarded using the sub-divide law. Process Q cannot be dealt with by our type systems, for reasons similar to those discussed above.

To sum up, as shown in the examples above, union types and the sub-divide law of [16] make their system more flexible than ours: that is, when restricting to the class of properties handled by [16] (i.e. the properties expressed by negative formulae in the sense of [16]), then the set of processes typable in the system of [16] is larger than the set of processes typable in our systems. However, union types and the sub-divide law do not preserve the spatial structure of terms, even if this is partially mitigated by the presence of tags that keep track of certain correlations among names. In our system, we stick to spatial-preserving laws, thus trading off some flexibility for precision.

Let us now reconsider the two examples above. Suppose that the service invocation on a is intended to trigger an interaction between two parties (dyadic session): then the very dependency of y from x suggests a way to re-write the process into a conceptually equivalent one that can be dealt with in our systems. In particular, there appears to be no reason why y should be received at a moment later than x. E.g., P can be re-written as $a(x,y).\overline{x}.\overline{y}$, and Q as

$$!a(x,y).(\nu c)\Big(\,(\nu z)(\overline{c}\langle x,z\rangle\,|\,z.\overline{y})\,|\,c(x,z).(\overline{x}|\overline{z})\,\Big)\,.$$

Both of these processes are typable in our local system. Of course, this sort of rewriting does not make sense when the sessions triggered by invocations at a are multiparty, e.g. when it is a third party, and not the invoker, which sends a message on b.

11. Conclusion, further and related work

We have defined and investigated a framework that incorporates ideas from both spatial logics and behavioural type systems, drawing benefits from both. Our main results are: type soundness theorems that, for interesting classes of properties, basically reduce model checking on pi-processes to (local or global) model checking on CCS processes, via type-checking; and the definition of syntactic conditions identifying sets of interesting formulae belonging to such classes.

Implementation issues are not in the focus of this paper and are left for future work. The normal derivation property already provides us with syntax-directed systems. Of course, implementing the model checks $\mathsf{T} \models \phi$ is still an issue. In this respect, it is important to remark that, for a large class of properties, checking $\mathsf{T} \models \phi$ might be decidable in spite of the fact that $P \models \phi$ is not. For instance, Busi et al.'s have shown [6] that "weak" barbs $\diamond^* a$ are decidable in CCS with replication, while they are *not* in the pi-calculus. In [1], we have recently proved decidability of a subclass of Shallow logic formulae, expressive enough to describe interesting safety properties. As in the case of [6], the proofs rely on well-structured transition system techniques [11]. Another possibility would be re-using existing work on spatial model checking: Caires' work [9] seems to be a promising starting point. Also, approximations of possibly infinite-state CCS types with finite-state automata, in the vein of [19], seem useful to design effective tools.

Inference systems and their implementation are left for further work. Certain decidability results about structural congruence [15, 14] suggest their feasibility. It would also be interesting to cast our approach in more applicative scenarios, like calculi for service-oriented computing [3].

Our work has been mainly inspired by Igarashi and Kobayashi' paper on generic type systems [16]. The main differences between this work and ours have been already discussed throughout the paper. Some recent work by Kobayashi and collaborators has pointed out the intrinsic limits of behavioural type systems based on the use of simulation as a subtyping relation [20].

Also related to our approach are some recent proposals by Caires. In [8, 7], a logical semantics approach to types for concurrency is pursued. Specifically, in [8], a notion of spatial-behavioral typing suitable to discipline concurrent interactions and resource usage in a distributed object calculus is defined. Types, that can be viewed as a fragment of a spatial logic for concurrency, express resource ownership. The proposed system guarantees the availability of services and (resource access) race freedom. Closest to our work is [7], where a generic type system for the pi-calculus - parameterized on the subtyping relation - is proposed. The author identifies a family of types, the so called shared types, which allow to modularly and safely compose spatial and shared (classical invariants) properties and to safely factorize spatial properties.

Our dependency graphs are reminiscent of Yoshida's graph types [24]. The main idea is the same, to trace the nesting ordering among prefixes. However, Yoshida's graphs are meant to be quite more precise abstraction of the behavior of processes: this makes their derivation more complex than dependency graphs'. Indeed, graph types are built by means of several operations: prefixing, parallel composition and hiding. Intuitively, while prefixing is conceptually very similar to ours, parallel composition in [24] is a sort of merge, which "consumes" the possible communications by removing synchronizing nodes, while hiding (restriction) removes all nodes (and the corresponding arcs) mentioning the hidden name. Our dependency graphs, instead, are meant to over-approximate dependencies and can be easily built by inspection of types. Indeed, parallel composition of two types corresponds to the union of the sets of nodes and arcs of the two original graphs.

The side conditions on critical names in the typing rules (G-OUT) and (G-PAR) can be considered as giving rise to an assume-guarantee system allowing compositional reasoning at the level of types. I.e., in (G-OUT), the assumption concerning disjunction of critical names guarantees that e.g. any well-typed environment that might become ready to receive the output will not interfere with the sender by creating new dependencies and breaking well-annotated-ness. Related papers using assume-guarantee techniques in a somehow more explicit way are [17, 12]. In the already mentioned [12], the authors integrate the subtyping relation with an assume-guarantee rule for CCS with respect to open simulation. In [17], Kobayashi and Sangiorgi propose a hybrid type system for lock-freedom combining deadlock-freedom, termination and confluence analysis. The relation of the proposal in [17] to assume-guarantee reasoning is clear: capability and obligation annotations on channel usages represent respectively assumptions on the environment and consequent guarantees. The so called hybrid typing rules of [17] discard those processes that rely on the environment in order to fulfill their obligation. Hence well-typed processes are lock-free without making any assumption on the environment.

A preliminary investigation of the ideas presented in this paper, in a much simpler setting, is in [4].

Acknowledgments. We wish to thank Luis Caires, Michele Loreti and Gianluigi Zavattaro for stimulating discussions on the topics of the paper. Anonymous reviewers of CONCUR 2008 and Information and Computation provided valuable comments on the shorter [2] and earlier version of this paper.

References

[1] Acciai, L., Boreale, M.: Deciding safety properties in infinite-state pi-calculus via behavioural types. In: Albers, S., Marchetti-Spaccamela, A., Matias, Y., Nikoletseas, S.E., and Thomas, W. (eds.) ICALP'09. LNCS, vol. 5556, pp. 31–42 (2009)

[2] Acciai, L., Boreale, M.: Spatial and behavioral types in the pi-calculus. In: van Breugel, F., and Chechik, M. (eds.) CONCUR'08. LNCS, vol. 5201, pp. 372–386 (2008)

[3] Acciai, L., Boreale, M.: A type system for client progress in a service-oriented calculus. In: Degano, P. et al. (eds.) *Concurrency, Graphs and Models, Essays Dedicated to Ugo Montanari on the Occasion of His 65th Birthday*. LNCS, vol. 5065, pp. 642–658 (2008)

[4] Acciai, L., Boreale, M.: Type abstractions of name-passing processes. In: Arbab, F. and Sirjani, M. (eds.) FSEN'07. LNCS, vol. 4767, pp. 302–317 (2007)

[5] Acciai, L., Boreale, M.: Responsiveness in process calculi. Theoretical Computer Science, 409(1), 59–93 (2008)

[6] Busi, N., Gabbrielli, M., Zavattaro, G.: On the Expressive Power of Recursion, Replication, and Iteration in Process Calculi. In: Díaz, J., Karhumäki, J., Lepistö, A., and Sannella, D. (eds.) ICALP'04. LNCS, vol. 3142, pp. 307–319 (2004)

[7] Caires, L.: Logical Semantics of Types for Concurrency. In: Mossakowski, T., Montanari, U., Haveraaen, M. (eds.) CALCO'07. LNCS, vol. 4624, pp. 16–35 (2007)

[8] Caires, L.: Spatial-Behavioral Types, Distributed Services, and Resources. In: Montanari, U., Sannella, D., Bruni, R. (eds.) TGC'06. LNCS, vol. 4661, pp. 98–115 (2007)

[9] Caires, L.: Behavioral and Spatial Observations in a Logic for the pi-Calculus. In: Walukiewicz, I. (eds) FoSSaCS'04. LNCS, vol. 2987, pp. 72–89 (2004)

[10] Caires, L., Cardelli, L.: A spatial logic for concurrency (part I). Inf. Comput. 186(2), 194–235 (2003)

[11] Finkel, A. and Schnoebelen, Ph.: Well-Structured Transition Systems Everywhere! Theoretical Computer Science, 256(1-2), 63–92 (2001)

[12] Chaki, S., Rajamani, S.K., Rehof, J.: Types as models: model checking message-passing programs. POPL'02, pp. 45–57 (2002)

[13] Cardelli, L., Gordon, A.D.: Anytime, Anywhere: Modal Logics for Mobile Ambients. POPL'00, pp. 365–377 (2000)

[14] Engelfriet, J., Gelsema, Tj.: The decidability of structural congruence for replication restricted pi-calculus processes. Technical Report, LIACS 2004–07 (2004)

[15] Engelfriet, J., Gelsema, Tj.: Structural Inclusion in the pi-Calculus with Replication. Theoretical Computer Science, 258(1-2), 131–168 (2001)

[16] Igarashi, A., Kobayashi, N.: A generic type system for the Pi-calculus. Theoretical Computer Science 311(1-3), 121–163 (2004)

[17] Kobayashi, N., Sangiorgi, D.: A Hybrid Type System for Lock-Freedom of Mobile Processes. In: Gupta, A., and Malik, S. (eds) CAV 2008. LNCS, vol. 5123, pp. 80–93 (2008)

[18] Kobayashi, N.: Type-based information flow analysis for the pi-calculus. Acta Inf. 42(4-5), 291–347 (2005)

[19] Kobayashi, N., Suenaga, K., Wischik, L.: Resource Usage Analysis for the pi-Calculus. Logical Methods in Computer Science 2(3) (2006)

[20] Kobayashi, N., Suto, T.: Undecidability of 2-Label BPP Equivalences and Behavioral Type Systems for the pi -Calculus. In: Arge, L., Cachin, C., Jurdzinski, T., Tarlecki, A. (eds) ICALP'07. LNCS, vol. 4596, pp. 740-751 (2007)

[21] Milner, R.: The polyadic π-calculus: a tutorial. Logic and Algebra of Specification, Springer, pp. 203–246 (1993)

[22] Milner, R.: Communication and concurrency. Prentice-Hall (1989)

[23] Sangiorgi, D.: The Name Discipline of Uniform Receptiveness. Theoretical Computer Science, 221(1-2), 457–493 (1999)

[24] Yoshida, N.: Graph Types for Monadic Mobile Processes. In: Chandru, V., and Vinay, V. (eds.) FSTTCS'96. LNCS, vol. 1180, pp. 371–386 (1996)

A. Proofs of Section 3

This section reports the proofs omitted in Section 3, along with some intermediate results that will be useful here and in the rest of the paper. The following lemma introduces a somewhat expected result concerning substitutions and transpositions. It is needed in the proof of Proposition 1.

Lemma A.1. *$A \xrightarrow{\lambda} B$ implies $A\{x \leftrightarrow y\} \xrightarrow{\lambda\{x\leftrightarrow y\}} B\{x \leftrightarrow y\}$.*

Proof. The proof is straightforward by induction on the derivation of $A \xrightarrow{\lambda} B$.

Proposition A.1 (Proposition 1). *Let Φ be a P-set. If $\lambda = \langle a\rangle$ with $a \in \mathrm{supp}(\Phi)$ then Φ_λ is a P-set and $\mathrm{supp}(\Phi_\lambda) \subseteq \mathrm{supp}(\Phi)$.*

Proof. We first prove that $N = \mathrm{supp}(\Phi)$ is a support of Φ_λ. This, together with definition of least support, is enough to ensure that $\mathrm{supp}(\Phi_\lambda) \subseteq \mathrm{supp}(\Phi)$.

Consider a term $B \models \Phi_\lambda$ with $\lambda = \langle a\rangle$, for some $a \in \mathrm{supp}(\Phi)$. By definition, there is an $A \models \Phi$ such that $A \xrightarrow{\lambda} B$. Take any $x,y \notin N$. By definition of support, it holds that $A\{x \leftrightarrow y\} \models \Phi$. Given that $a \in \mathrm{supp}(\Phi)$, we get $\lambda\{x \leftrightarrow y\} = \lambda$ and, by Lemma A.1, $A\{x \leftrightarrow y\} \xrightarrow{\lambda} B\{x \leftrightarrow y\}$. Hence, by definition of Φ_λ we get $B\{x \leftrightarrow y\} \models \Phi_\lambda$. Thus, N is a support of Φ_λ.

Finally, Φ_λ is a P-set because it has a finite support and, by rule (struct), Φ_λ is closed under $\equiv$.

Let us now point out some properties of formulae that will be useful more than once in the sequel.

Lemma A.2. *Assume ψ does not contain $\neg$. Consider any $P, Q \in \mathcal{P}$, any $\mathsf{T}, \mathsf{S} \in \mathcal{T}$ and a sets of fresh names $\tilde{z}$ and $\tilde{w}$ of appropriate sort.*

1. *$(\tilde{\nu x})P \models \psi$ implies $(\tilde{\nu x})(P\,|\,Q) \models \psi$ and $(\tilde{\nu x})\mathsf{T} \models \psi$ implies $(\tilde{\nu x})(\mathsf{T}\,|\,\mathsf{S}) \models \psi$.*
2. *$P \models \Diamond^*\psi$ implies $\overline{a}\langle\tilde{z}\rangle\,|\,(a(\tilde{w}).Q + \tau.P) \models \Diamond^*\psi$. $\mathsf{T} \models \Diamond^*\psi$ implies $\overline{a}\,|\,(a.\mathsf{S} + \tau.\mathsf{T}) \models \Diamond^*\psi$.*
3. *$P \models \Box^*_{-\tilde{a}}\Diamond^*\psi$ implies $\overline{a}\langle\tilde{w}\rangle\,|\,(a(\tilde{z}).Q + \tau.P) \models \Box^*_{-\tilde{a}}\Diamond^*\psi$, for every $a \in \tilde{a}$. $\mathsf{T} \models \Box^*_{-\tilde{a}}\Diamond^*\psi$ implies $\overline{a}\,|\,(a.\mathsf{S} + \tau.\mathsf{T}) \models \Box^*_{-\tilde{a}}\Diamond^*\psi$, for every $a \in \tilde{a}$.*

Proof. In (1), the proof is straightforward by induction on the structure of the formula ψ. For (2) the proof relies on (1) with $\tilde{x} = \emptyset$, $Q = \overline{a}\langle\tilde{z}\rangle$ (resp. $\mathsf{S} = \overline{a}$) and on the definition of $[\![\cdot]\!]$ while for (3) the proof relies on (2) and on the definition of $[\![\cdot]\!]$.

Lemma A.3 (Lemma 1). *Let $\phi \in \mathcal{F}$. Then $[\![\phi]\!]$ is a P-set and $fn(\phi) \supseteq \mathrm{supp}([\![\phi]\!])$.*

Proof. The proof is straightforward by induction on the structure of ϕ. For the base cases $\phi = a$ and $\phi = \overline{a}$ it is obvious. In the other cases it proceeds by applying the inductive hypothesis. Consider the case $\phi = \langle -\tilde{a}\rangle^*\psi$ and suppose that there is $b\#\mathrm{fn}(\phi)$ such that $b \in \mathrm{supp}([\![\phi]\!])$, we prove that this assertion leads to a contradiction.

Take any $A \in [\![\phi]\!]$. By definition, $A \models \psi$ and for each B s.t. $A \xrightarrow{s} B$, with $\tilde{a}\#s$, we get $B \models \psi$. By applying the induction hypothesis to ψ we get both $A\{b \leftrightarrow c\} \models \psi$ and $B\{b \leftrightarrow c\} \models \psi$, for any $b,c\#\mathrm{fn}(\psi) \subseteq \mathrm{fn}(\phi)$. Moreover, by Lemma A.1, $A \xrightarrow{s} B$ implies $A\{b \leftrightarrow c\} \xrightarrow{s\{b\leftrightarrow c\}} B\{b \leftrightarrow c\}$. Hence, by choosing any $c\#\mathrm{fn}(\phi)$ we get $A\{b \leftrightarrow c\} \models \phi$. This holds for any $A \in [\![\phi]\!]$, therefore we have a contradiction and $b\#\mathrm{supp}([\![\phi]\!])$, for any $b\#\mathrm{fn}(\phi)$.

As expected, $\tilde{a}$ is contained in the support of (the denotation of) any non-trivial formula containing $\Box^*_{-\tilde{a}}$.

Lemma A.4. *Let* $\Phi = [\![\Box^*_{-\tilde{a}}\Diamond^*\phi]\!]$*, where* ϕ *does not contain* $\neg$*. Then* $\Phi \neq \mathcal{U}$ *and* $\Phi \neq \emptyset$ *imply* $\tilde{a} \subseteq \mathrm{supp}(\Phi)$.

Proof. Suppose $\Phi \neq \mathcal{U}$ and $\Phi \neq \emptyset$ and take any A and B such that $A|B$ is a term, $A \not\models \Phi$ ($A \not\models \Box^*_{-\tilde{a}}\Diamond^*\phi$) and $B \models \Phi$ ($B \models \Box^*_{-\tilde{a}}\Diamond^*\phi$). Note that $A \not\models \Phi$ implies $A\{x \leftrightarrow y\} \not\models \Phi$ for any $x, y \notin \mathrm{supp}(\Phi)$.

Suppose A and B are processes and consider the term $C = \overline{a}\langle\tilde{z}\rangle\,|\,(a(\tilde{z}).A + \tau.B)$, for any $a \in \tilde{a}$ and some fresh $\tilde{z}$ of suitable sort. By Lemma A.2 (3), $C \models \Box^*_{-\tilde{a}}\Diamond^*\phi$, therefore $C \in \Phi$.

Suppose by contradiction that $a \notin \mathrm{supp}(\Phi)$. By definition of support, we get $C\{a \leftrightarrow b\} \models \Phi$ (that is $C\{a \leftrightarrow b\} \models \Box^*_{-\tilde{a}}\Diamond^*\phi$) for each $b \notin \mathrm{supp}(\Phi)$, and in particular this holds for some $b \notin \tilde{a}$. Therefore, $C\{a \leftrightarrow b\} = \overline{b}\langle\tilde{z}\{a \leftrightarrow b\}\rangle\,|\,(b(\tilde{w}).A\{a \leftrightarrow b\} + \tau.B\{a \leftrightarrow b\}) \xrightarrow{\langle b\rangle} A\{a \leftrightarrow b\}$, and it must be $A\{a \leftrightarrow b\} \models \Box^*_{-\tilde{a}}\Diamond^*\phi$, hence $A\{a \leftrightarrow b\} \models \Phi$, and $A \models \Phi$, by definition of support: contradiction. Therefore, $\tilde{a} \subseteq \mathrm{supp}(\Phi)$.

The following proof enhances compositionality of spatial model checking. Indeed, it guarantees that satisfiability does not depend on non-interfering parallel threads. This result allows us to cut away some subterms when checking "$\models$".

Lemma A.5 (Lemma 2). *Let* A *be a term and* $\phi \in \mathcal{F}_{\tilde{x}}$*. For any term* B *such that* $A|B$ *is a term and* $\mathrm{fn}(B) = \emptyset$ *we have that* $A \models \phi$ *if and only if* $A|B \models \phi$*.*

Proof. We prove that for any term B such that $A|B$ is a term, $A\#B$ and $\tilde{x}\#B$ we have that $A \models \phi$ if and only $A|B \models \phi$. By this and $\mathrm{fn}(B) = \emptyset$ we get the result.

The proof proceeds by induction on the structure of ϕ. Cases $\phi = \mathbf{T}$, $\phi = a$ and $\phi = \overline{a}$, with $a \in \tilde{x}$, are obvious. Let us consider the other cases:

$\phi = \neg\psi$**.**

- ($\Rightarrow$) $A \models \neg\psi$ implies $A \not\models \psi$. By applying the induction hypothesis, $A|B \not\models \psi$ and by definition $A|B \models \neg\psi$.
- ($\Leftarrow$) In this case the proof proceeds similarly.

$\phi = \mathrm{H}^*\psi$**.**

- ($\Rightarrow$) $A \models \phi$ implies $A \equiv (\tilde{\nu}\tilde{a})A'$ with $A' \models \psi$ and $\tilde{a}\#B$. By induction hypothesis, $A'|B \models \psi$. Moreover, $A|B \equiv (\tilde{\nu}\tilde{a})(A'|B)$, and $A|B \models \mathrm{H}^*\psi$ by definition of "$[\![\mathrm{H}^*\psi]\!]$".
- ($\Leftarrow$) $A|B \models \phi$ means that $A|B \equiv (\tilde{\nu}\tilde{a})(\tilde{\nu}\tilde{b})(A'|B')$, with $A \equiv (\tilde{\nu}\tilde{a})A'$ ($\tilde{a}\#B, B'$), $B \equiv (\tilde{\nu}\tilde{b})B'$ ($\tilde{b}\#A, A'$) and $A'|B' \models \psi$. From $A\#B$, $\tilde{a}\#B'$, $\tilde{b}\#A'$ and $\tilde{x}\#B$, we get $A'\#B'$ and $\tilde{x}\#B'$, hence by applying the induction hypothesis, $A' \models \psi$. Finally, by $A \equiv (\tilde{\nu}\tilde{a})A'$ and definition of "$[\![\mathrm{H}^*\psi]\!]$", we get $A \models \phi$.

$\phi = \phi_1 \vee \phi_2$**.** In both cases, the proof proceeds by applying the induction hypothesis.

$\phi = \phi_1|\phi_2$**.**

- ($\Rightarrow$) $A \models \phi$ implies $A \equiv A_1|A_2$, with $A_1 \models \phi_1$ and $A_2 \models \phi_2$. By applying the induction hypothesis, we get $A_1|B \models \phi_1$ (similarly $A_2|B \models \phi_2$). Hence, by definition of $[\![\phi_1|\phi_2]\!]$, we get $A_1|B|A_2 \models \phi$, and given that $A|B \equiv A_1|B|A_2$, by definition of "$[\![\cdot]\!]$", $A|B \models \phi$.

($\Leftarrow$) $A|B \models \phi$ implies $A|B \equiv A_1|A_2|B_1|B_2$, with $A \equiv A_1|A_2$, $B \equiv B_1|B_2$, $A_1|B_1 \models \phi_1$ and $A_2|B_2 \models \phi_2$. From $A\#B$ and $\tilde{x}\#B$ we get $B_i\#A_i$, and $\tilde{x}\#B_i$, for $i = 1, 2$. By applying the induction hypothesis, we get $A_1 \models \phi_1$ and $A_2 \models \phi_2$. Hence, by definition of $[[\phi_1|\phi_2]]$, we get $A_1|A_2 \models \phi$, and given that $A \equiv A_1|A_2$, by definition of "$[[\cdot]]$", $A \models \phi$.

$\phi = \langle a\rangle\psi$.

($\Rightarrow$) $A \models \phi$ implies $A \xrightarrow{\langle a\rangle} A'$ with $A' \models \psi$. By applying the induction hypothesis, we get $A'|B \models \psi$. Moreover, by (PAR), $A|B \xrightarrow{\langle a\rangle} A'|B$, hence $A|B \models \langle a\rangle\psi$.

($\Leftarrow$) $A|B \models \phi$ implies $A|B \xrightarrow{\langle a\rangle} C$ with $C \models \psi$. From $\tilde{x}\#B$, the reduction with subject a originates from A and $A|B \xrightarrow{\langle a\rangle} C$ has been deduced by applying (PAR). Hence, $A \xrightarrow{\langle a\rangle} A'$ and $C \equiv A'|B$. Moreover, $B\#A'$ and $\tilde{x}\#B$ hold, hence by applying the induction hypothesis, we get $A' \models \psi$. Therefore, $A \models \langle a\rangle\psi$.

$\phi = \langle\tilde{a}\rangle^*\psi$, with $\tilde{a} \subseteq \tilde{x}$. this case can be proved as a generalization of the previous one to $n \geq 0$ reductions with subjects in $\tilde{a} \subseteq \tilde{x}$.

$\phi = \langle -\tilde{b}\rangle^*\psi$, with $\tilde{b} \subseteq \tilde{x}$.

($\Rightarrow$) $A \models \phi$ implies $A \xrightarrow{s} A'$, with $A' \models \psi$ and $\tilde{b}\#s$. By $B\#A'$, and $\tilde{x}\#B$ and by applying the induction hypothesis, we get $A'|B \models \psi$. Moreover, by (PAR), $A|B \xrightarrow{s} A'|B$ and $A|B \models \langle -\tilde{b}\rangle^*\psi$.

($\Leftarrow$) $A|B \models \phi$ implies $A|B \xrightarrow{s} C$, with $C \models \psi$ and $\tilde{b}\#s$. From $A\#B$, we get $C \equiv A'|B'$ with $A \xrightarrow{s_1} A'$, $B \xrightarrow{s_2} B'$ and s is some shuffle of s_1 and s_2. Again from $A\#B$ and $\tilde{x}\#B$, we get $A'\#B'$ and $\tilde{x}\#B'$. Hence, by applying the induction hypothesis, it follows that $A' \models \psi$ and $A \models \langle -\tilde{b}\rangle^*\psi$.

Recall that the following lemma syntactically identifies formulae that are guaranteed to be Ok. Lemma A.4 is used in the second part of its proof.

Lemma A.6 (Lemma 3). *Let ϕ be a Shallow Logic Formula of the form either $\Box^*\psi$ or $\Box^*_{-\tilde{a}}\Diamond^*\psi'$, where ψ' does not contain $\neg$. Then* $\mathrm{Ok}(\phi)$.

Proof. We examine the two cases separately.

$\phi = \Box^*\psi$. We prove that $\{[[\Box^*\psi]]\} \subseteq \mathrm{Ok}$, by showing this set satisfies points (1), (2) and (3) in Definition 2.

(1). It follows by Lemma 2.

(2). By definition, $[[\phi]] = [[\Box^*\phi]]$, therefore $[[\phi]] = [[\phi]]_\lambda$ for each λ.

(3). It follows by $[[\phi]] = [[\phi]]_\lambda$ for each λ, as shown above.

We have proved that $\{[[\Box^*\psi]]\} \subseteq \mathrm{Ok}$.

$\phi = \Box^*_{-\tilde{a}}\Diamond^*\psi'$ for some ψ' that does not contain $\neg$. As before, we prove that $\{[[\phi]], \mathcal{U}, \emptyset\} \subseteq \mathrm{Ok}$, by showing this set satisfies points (1), (2) and (3) in Definition 2. If $[[\phi]] = \emptyset$ or $[[\phi]] = \mathcal{U}$ this is obvious. Assume the contrary, then, by Lemma A.4, $\tilde{a} \subseteq \mathrm{supp}([[\phi]])$.

(1). It follows by Lemma 2.

(2). By definition, $[\![\Box^*_{-\tilde{a}}\Diamond^*\psi']\!] = [\![\Box^*_{-\tilde{a}}\Box^*_{-\tilde{a}}\Diamond^*\psi']\!]$ and $[\![\phi]\!]_\lambda = [\![\phi]\!]$ for λ *not* of the form $\langle a\rangle$ with $a \in \tilde{a}$.

(3). As already seen $[\![\phi]\!]_\lambda = [\![\phi]\!]$ for λ of the form $\langle a\rangle$ with $a \notin \tilde{a}$.
The proof proceeds by showing that $[\![\phi]\!]_\lambda = \mathcal{U}$ for $\lambda = \langle a\rangle$ with $a \in \tilde{a}$. Indeed, take any $A \in \mathcal{U}$ and $B \models [\![\phi]\!]$ such that $A|B$ is a term (such a B must exists because $[\![\phi]\!] \neq \emptyset$). Define $C = \overline{a}\langle\tilde{z}\rangle\,|\,(a(\tilde{z}).A + \tau.B)$, for $a \in \tilde{a}$ and some fresh $\tilde{z}$ of appropriate sort. By Lemma A.2 (3), we have $C \models [\![\phi]\!]$ and $C \xrightarrow{\langle a\rangle} A$. This proves that $[\![\phi]\!]_\lambda = \mathcal{U}$, with $\lambda = \langle a\rangle$ for some $a \in \tilde{a}$. Of course, $\mathcal{U}_\lambda = \mathcal{U}$ and $\emptyset_\lambda = \emptyset$ for any λ. This proves point (3).

We have proved that $\{[\![\phi]\!], \mathcal{U}, \emptyset\} \subseteq \mathrm{Ok}$, hence $\mathrm{Ok}(\phi)$.

B. Proofs of Section 4

The proofs of the basic properties of the local system omitted in Subsection 4.3 can be found here. Some additional results are also proved along the way.

The following lemma guarantees that, as already discussed in Remark 3, annotations on input prefixes and restrictions on types are sufficient to guarantee that the scope extrusion law preserves the spatial correspondence between processes and types. To prove this, it is necessary to prove that each free name in a process is also free in the corresponding type.

Lemma B.1. *If* $\Gamma \vdash_L P : \mathsf{T}$ *then* $\mathrm{fn}(P) \subseteq \mathrm{fn}(\mathsf{T})$ *and* $\mathrm{fn}(\mathsf{T}) \subseteq \mathrm{dom}(\Gamma)$. *If* $\Gamma, \tilde{x} : \tilde{\mathsf{t}} \vdash_L P : \mathsf{T}$ *and* $\tilde{x}\#P, \Gamma$ *then* $\tilde{x}\#\mathsf{T}$.

Proof. The proof is straightforward by induction on the derivation of $\Gamma \vdash_L P : \mathsf{T}$.

The usual weakening and contraction properties hold for the local system.

Proposition B.1 (weakening and contraction). *If* $\Gamma \vdash_L P : \mathsf{T}$, Γ *well-formed and* $\tilde{x}\#P, \Gamma$ *then* $\Gamma, \tilde{x} : \tilde{\mathsf{t}} \vdash_L P : \mathsf{T}$. *If* $\Gamma, \tilde{x} : \tilde{\mathsf{t}} \vdash_L P : \mathsf{T}$, Γ *well-formed and* $\tilde{x}\#P, \Gamma$ *then* $\Gamma \vdash_L P : \mathsf{T}$.

Proof. The proof is straightforward by induction on the derivation of $\Gamma \vdash_L P : \mathsf{T}$ and $\Gamma, \tilde{x} : \tilde{\mathsf{t}} \vdash_L P : \mathsf{T}$; it proceeds by distinguishing the last typing rule applied.

Recall that a normal derivation is a derivation where (L-Eq) is applied only before (T-Inp). Proposition 2 can be proved by an easy induction on the typing derivation.

Proposition B.2 (normal derivation, Proposition 2). *Suppose* $\Gamma \vdash_L P : \mathsf{T}$. *Then* $\Gamma \vdash_{NL} P : \mathsf{S}$ *for some* $\mathsf{S} \equiv \mathsf{T}$.

Proof. The proof proceeds by induction on the derivation of $\Gamma \vdash_L P : \mathsf{T}$ by distinguishing the last typing rule applied.

(L-Inp). By $\Gamma \vdash_L a(\tilde{x}).P : a((\tilde{x} : \tilde{\mathsf{t}})\mathsf{T}).\mathsf{T}'$ and the premise of the rule, we get $\Gamma \vdash_L a : (\tilde{x} : \tilde{\mathsf{t}})\mathsf{T}$, $\Gamma, \tilde{x} : \tilde{\mathsf{t}} \vdash_L P : \mathsf{T}|\mathsf{T}'$ and $\tilde{x}\#\mathsf{T}'$. By applying the induction hypothesis to $\Gamma, \tilde{x} : \tilde{\mathsf{t}} \vdash_L P : \mathsf{T}|\mathsf{T}'$, we derive that there is an $\mathsf{S}' \equiv \mathsf{T}|\mathsf{T}'$ such that $\Gamma, \tilde{x} : \tilde{\mathsf{t}} \vdash_{NL} P : \mathsf{S}'$. By applying (L-Eq) and then (L-Inp) to this normal derivation, we deduce that $\Gamma \vdash_{NL} a(\tilde{x}).P : a((\tilde{x} : \tilde{\mathsf{t}})\mathsf{T}).\mathsf{T}'$.

(L-Eq). By $\Gamma \vdash_{\mathrm{L}} P : \mathsf{T}$ and the premise of the rule, we get $\Gamma \vdash_{\mathrm{L}} P : \mathsf{S}$ with $\mathsf{S} \equiv \mathsf{T}$. By applying the induction hypothesis to $\Gamma \vdash_{\mathrm{L}} P : \mathsf{S}$, we get $\Gamma \vdash_{\mathrm{NL}} P : \mathsf{S}'$ for an $\mathsf{S}' \equiv \mathsf{S}$, and, by transitivity of $\equiv$, $\mathsf{S}' \equiv \mathsf{T}$.

(L-Res). In this case the proof relies on the induction hypothesis and on closure of P-sets with respect to $\equiv$.

(L-Out), (L-Tau), (L-Sum), (L-Par), (L-Rep). In these cases the proof proceeds by applying the induction hypothesis, for deriving a normal derivation for the premise, followed by application of the corresponding typing rule.

In what follows we write $\Gamma \vdash^{k}_{\mathrm{L}} P : \mathsf{T}$ if $\Gamma \vdash_{\mathrm{L}} P : \mathsf{T}$ can be deduced with a derivation of height $\leq k$. This additional annotation is necessary in order to guarantee Lemma 5, which comes as a corollary of the result below.

Lemma B.2. *Suppose that* $\Gamma \vdash^{k}_{\mathrm{L}} P : \mathsf{T}$*, that* $\Gamma \vdash \tilde{b} : \tilde{\mathsf{t}}$ *and that* $\Gamma \vdash a : (\tilde{x} : \tilde{\mathsf{t}})\mathsf{U}$*. Then for any* $\mathsf{S}, \mathsf{S}_1, \mathsf{S}_2$*, and* $\mu_i.\mathsf{S}_i$ $(i \in I)$*, it holds that:*

1. $\mathsf{T} = a((\tilde{x} : \tilde{\mathsf{t}})\mathsf{U}).\mathsf{S}$ *implies* $P \equiv a(\tilde{x}).Q$ *for some* Q *such that* $\Gamma, \tilde{x} : \tilde{\mathsf{t}} \vdash^{l}_{\mathrm{L}} Q : \mathsf{U}|\mathsf{S}$*,* $l < k$ *and* $\tilde{x}\#\mathsf{S}$.
2. $\mathsf{T} = \overline{a}.\mathsf{S}$ *implies* $P \equiv \overline{a}\langle\tilde{b}\rangle.Q$ *for some* Q *and* S' *such that* $\Gamma \vdash^{l}_{\mathrm{L}} Q : \mathsf{S}'$*,* $l < k$ *and* $\mathsf{S} \equiv \mathsf{U}[^{\tilde{b}}/_{\tilde{x}}]|\mathsf{S}'$.
3. $\mathsf{T} = \tau.\mathsf{S}$ *implies* $P \equiv \tau.Q$ *for some* Q *such that* $\Gamma \vdash^{l}_{\mathrm{L}} Q : \mathsf{S}$ *with* $l < k$.
4. $\mathsf{T} = (\nu\tilde{c} : \tilde{\mathsf{t}}')\mathsf{S}$ *implies* $P \equiv (\nu\tilde{c} : \tilde{\mathsf{t}}'; \Phi)Q$ *for some* Q *such that* $\Gamma, \tilde{c} : \tilde{\mathsf{t}}' \vdash^{l}_{\mathrm{L}} Q : \mathsf{S}$, $\mathsf{S} \downarrow_{\tilde{c}} \models \Phi$ *and* $l < k$.
5. $\mathsf{T} = \mathsf{S}_1|\mathsf{S}_2$ *implies* $P \equiv Q_1|Q_2$ *for some* Q_1 *and* Q_2 *such that* $\Gamma \vdash^{l_1}_{\mathrm{L}} Q_1 : \mathsf{S}_1$, $\Gamma \vdash^{l_2}_{\mathrm{L}} Q_2 : \mathsf{S}_2$ *and* $l_1, l_2 < k$*;*
6. $\mathsf{T} = !a((\tilde{x} : \tilde{\mathsf{t}})\mathsf{U}).\mathsf{S}$ *implies* $P \equiv !a(\tilde{x}).Q$ *for some* Q *such that* $\Gamma \vdash^{l}_{\mathrm{L}} a(\tilde{x}).Q : a((\tilde{x} : \tilde{\mathsf{t}})\mathsf{U}).\mathsf{S}$ *and* $l < k$*;*
7. $\mathsf{T} = \sum_{i \in I} \mu_i.\mathsf{S}_i$, $|I| \neq 1$*, implies* $P \equiv \sum_{i \in I} \alpha_i.Q_i$ *for some* Q_i *and* α_i *such that* $\Gamma \vdash^{l_i}_{\mathrm{L}} \alpha_i.Q_i : \mu_i.\mathsf{S}_i$ *and* $l_i < k$*, for each* $i \in I$.

Proof. The proof proceeds by induction on the derivation of $\Gamma \vdash^{k}_{\mathrm{L}} P : \mathsf{T}$ by considering the last typing rule applied.

Cases (L-Inp), (L-Out), (L-Tau), (L-Res), (L-Par), (L-Rep) and (L-Sum) are obvious.

Let us suppose that (L-Eq) is the last applied. By its premise we get $\mathsf{T} \equiv \mathsf{S}$ and $\Gamma \vdash^{k-1}_{\mathrm{L}} P : \mathsf{S}$. By applying the induction hypothesis and by $\Gamma \vdash^{k-1}_{\mathrm{L}} P : \mathsf{S}$, we know that

$$\text{points } \textit{(1–7)} \text{ hold for } \mathsf{S} \text{ and } P. \tag{2}$$

The proof proceeds by induction on the derivation of $\mathsf{T} \equiv \mathsf{S}$, i.e. by considering the last structural rule applied. Notice that in the following we consider not only the rules reported in Table 1 but also the standard ones for transitivity, symmetry, reflexivity and contexts.

Let us consider first the rules in Table 1. Notice how the heights of the typing derivations comes into play in each case.

$\mathsf{S} = \mathsf{S}'|0 \equiv \mathsf{S}' = \mathsf{T}$**.** By (2) we know that $P \equiv P_1|P_2$ with P_1 and P_2 such that $\Gamma \vdash^{l_1}_{\mathrm{L}} P_1 : \mathsf{S}'$ and $\Gamma \vdash^{l_2}_{\mathrm{L}} P_2 : 0$, with $l_1, l_2 < k - 1$. Hence $P_2 = \mathbf{0}$, and, by applying the structural rules, $P \equiv P_1$ with $\Gamma \vdash^{l_1}_{\mathrm{L}} P_1 : \mathsf{T}$. By $l_1 < k$ and external induction, points *(1–7)* hold for P_1 and T.

The proof for the converse is similar.

$\mathsf{S} = \mathsf{S}_1|\mathsf{S}_2 \equiv \mathsf{S}_2|\mathsf{S}_1 = \mathsf{T}$. By (2) we know that $P \equiv P_1|P_2$ with P_1 and P_2 such that $\Gamma \vdash^{l_1}_{\mathrm{L}} P_1 : \mathsf{S}_1$ and $\Gamma \vdash^{l_2}_{\mathrm{L}} P_2 : \mathsf{S}_2$, with $l_1, l_2 < k-1$. By applying the structural rules for processes $P \equiv P_2|P_1$.

The proof for the converse is similar.

$\mathsf{S} = (\mathsf{S}_1|\mathsf{S}_2)|\mathsf{S}_3 \equiv \mathsf{S}_1|(\mathsf{S}_2|\mathsf{S}_3) = \mathsf{T}$. By (2) we know that $P \equiv Q|P_3$ with Q and P_3 such that $\Gamma \vdash^{l_1}_{\mathrm{L}} Q : (\mathsf{S}_1|\mathsf{S}_2)$ and $\Gamma \vdash^{l_3}_{\mathrm{L}} P_3 : \mathsf{S}_3$, with $l_1, l_3 < k-1$. By external induction, $\Gamma \vdash^{l_1}_{\mathrm{L}} Q : (\mathsf{S}_1|\mathsf{S}_2)$ and $l_1 < k-1$ imply $Q \equiv P_1|P_2$ with P_1 and P_2 such that $\Gamma \vdash^{m_1}_{\mathrm{L}} P_1 : \mathsf{S}_1$ and $\Gamma \vdash^{m_2}_{\mathrm{L}} P_2 : \mathsf{S}_2$, with $m_1, m_2 < l_1$. Hence, $P \equiv (P_1|P_2)|P_3$ and by transitivity of "$\equiv$", $P \equiv P_1|(P_2|P_3)$. Moreover, by (L-Par), $\Gamma \vdash^{n}_{\mathrm{L}} (P_2|P_3) : (\mathsf{S}_2|\mathsf{S}_3)$ with $n = \max(m_2, l_3) + 1 < k$.

The proof for the converse is similar.

$\mathsf{S} = (\nu\tilde{x} : \tilde{\mathsf{t}})\mathsf{S}_1|\mathsf{S}_2 \equiv (\nu\tilde{x} : \tilde{\mathsf{t}})(\mathsf{S}_1|\mathsf{S}_2) = \mathsf{T}$ **with** $\tilde{x}\#\mathsf{S}_2$. By (2) we know that $P \equiv P_1|P_2$ with P_1 and P_2 such that $\Gamma \vdash^{l_1}_{\mathrm{L}} P_1 : (\nu\tilde{x} : \tilde{\mathsf{t}})\mathsf{S}_1$ and $\Gamma \vdash^{l_2}_{\mathrm{L}} P_2 : \mathsf{S}_2$, with $l_1, l_2 < k-1$. By $\tilde{x}\#\mathsf{S}_2$ and Lemma B.1 we deduce that $\tilde{x}\#P_2$. By external induction, $\Gamma \vdash^{l_1}_{\mathrm{L}} P_1 : (\nu\tilde{x} : \tilde{\mathsf{t}})\mathsf{S}_1$ and $l_1 < k-1$ imply $P_1 \equiv (\nu\tilde{x} : \tilde{\mathsf{t}}; \Phi)P'_1$ with P'_1 such that $\Gamma, \tilde{x} : \tilde{\mathsf{t}} \vdash^{m_1}_{\mathrm{L}} P'_1 : \mathsf{S}_1$ with $m_1 < l_1$ and $\mathsf{S}_1 \downarrow_{\tilde{x}} \models \Phi$, for some $\tilde{\mathsf{t}}$. Hence, $P \equiv (\nu\tilde{x} : \tilde{\mathsf{t}}; \Phi)P'_1|P_2$ and by transitivity of "$\equiv$", $P \equiv (\nu\tilde{x} : \tilde{\mathsf{t}}; \Phi)(P'_1|P_2)$. By $\mathrm{Ok}(\Phi)$, $\mathsf{S}_1 \downarrow_{\tilde{x}} \models \Phi$ and $\mathrm{fn}(\mathsf{S}_2 \downarrow_{\tilde{x}}) = \emptyset$, we get $(\mathsf{S}_1|\mathsf{S}_2) \downarrow_{\tilde{x}} \models \Phi$. Moreover, by (L-Par) and Proposition B.1 (weakening and contraction), $\Gamma, \tilde{x} : \tilde{\mathsf{t}} \vdash^{m}_{\mathrm{L}} (P'_1|P_2) : (\mathsf{S}_1|\mathsf{S}_2)$, with $m < k$.

The proof for the converse is similar.

Suppose now the last structural rule applied is one of the context rules. We distinguish the following cases.

$C[\cdot] = a((\tilde{x} : \tilde{\mathsf{t}})\mathsf{U}).[\cdot]$. Since $\mathsf{S} \triangleq C[\mathsf{S}']$ we know that $\mathsf{S} = a((\tilde{x} : \tilde{\mathsf{t}})\mathsf{U}).\mathsf{S}'$ and by (2) $P \equiv a(\tilde{x}).Q$ for a Q such that $\Gamma, \tilde{x} : \tilde{\mathsf{t}} \vdash^{l}_{\mathrm{L}} Q : \mathsf{U}|\mathsf{S}'$, $l < k-1$ and $\tilde{x}\#\mathsf{S}'$. Given that $\mathsf{S} \equiv \mathsf{T}$ and $\mathsf{T} = C[\mathsf{T}']$, we get $\mathsf{S}' \equiv \mathsf{T}'$. Hence, given that "$\equiv$" is preserved by parallel composition, we get $\Gamma, \tilde{x} : \tilde{\mathsf{t}} \vdash^{l+1}_{\mathrm{L}} Q : \mathsf{U}|\mathsf{T}'$, by (L-Eq), with $l+1 < k$ and $\tilde{x}\#\mathsf{T}'$.

$C[\cdot] = \overline{a}.[\cdot]$. Since $\mathsf{S} \triangleq C[\mathsf{S}']$ we know that $\mathsf{S} = \overline{a}.\mathsf{S}'$ and by (2) $P \equiv \overline{a}\langle\tilde{b}\rangle.Q$ for some $\tilde{b}$ such that $\Gamma \vdash \tilde{b} : \tilde{\mathsf{t}}$, $\Gamma \vdash^{l}_{\mathrm{L}} Q : \mathsf{S}''$, $l < k-1$ and $\mathsf{S}' \equiv \mathsf{U}[{}^{\tilde{b}}/_{\tilde{x}}]|\mathsf{S}''$, if $\Gamma \vdash a : (\tilde{x} : \tilde{\mathsf{t}})\mathsf{U}$. The result follows from $\mathsf{S} \equiv \mathsf{T}$ and transitivity of "$\equiv$".

$C[\cdot] = \tau.[\cdot]$. Since $\mathsf{S} \triangleq C[\mathsf{S}']$ we know that $\mathsf{S} = \tau.\mathsf{S}'$ and by (2) $P \equiv \tau.Q$ for a Q such that $\Gamma \vdash^{l}_{\mathrm{L}} Q : \mathsf{S}'$, $l < k-1$. Given that $\mathsf{S} \equiv \mathsf{T}$ and $\mathsf{T} = C[\mathsf{T}']$, we get $\mathsf{T}' \equiv \mathsf{S}'$ and by (L-Eq) we get $\Gamma \vdash^{l+1}_{\mathrm{L}} Q : \mathsf{T}'$, with $l+1 < k$.

$C[\cdot] = [\cdot]|\mathsf{U}$. Since $\mathsf{S} \triangleq C[\mathsf{S}']$ we know that $\mathsf{S} = \mathsf{S}'|\mathsf{U}$ and by (2) $P \equiv P_1|P_2$, $\Gamma \vdash^{l_1}_{\mathrm{L}} P_1 : \mathsf{S}'$ and $\Gamma \vdash^{l_2}_{\mathrm{L}} P_2 : \mathsf{U}$, for $l_1, l_2 < k-1$. By $\mathsf{S} \equiv \mathsf{T}$ and $\mathsf{T} = C[\mathsf{T}']$, we get $\mathsf{T}' \equiv \mathsf{S}'$. Hence, by (L-Eq), $\Gamma \vdash^{l+1}_{\mathrm{L}} P_1 : \mathsf{T}'$, with $l+1 < k$.

$C[\cdot] = \sum_{i \in I, |I|>1} \mu_i.\mathsf{S}_i + \mu.[\cdot]$. Since $\mathsf{S} \triangleq C[\mathsf{S}']$ we know that $\mathsf{S} = \sum_{i \in I, |I|>1} \mu_i.\mathsf{S}_i + \mu.\mathsf{S}'$ and by (2) $P \equiv \sum_{i \in I, |I|>1} \alpha_i.P_i + \alpha.Q$, $\Gamma \vdash^{l_i}_{\mathrm{L}} \alpha_i.P_i : \mu_i.\mathsf{S}_i$ and $\Gamma \vdash^{l}_{\mathrm{L}} \alpha.Q : \mu.\mathsf{S}'$, for $l_i, l < k-1$. The proof proceeds as already seen for $C[\cdot] = \mu.[\cdot]$, with $\mu ::= \overline{a} \mid b(\mathsf{t}) \mid \tau$, for proving that $\Gamma \vdash^{l+1}_{\mathrm{L}} \alpha.Q : \mu.\mathsf{T}'$, with $l+1 < k$ and $\mathsf{T} = C[\mathsf{T}']$.

$C[\cdot] = !a(\mathsf{t}).[\cdot]$. Since $\mathsf{S} \triangleq C[\mathsf{S}']$ we know that $\mathsf{S} = !a(\mathsf{t}).\mathsf{S}'$ and by (2) we get $P \equiv !a(\tilde{x}).Q$ for some $a(\tilde{x}).Q$ such that $\Gamma \vdash^{l}_{\mathrm{L}} a(\tilde{x}).Q : a(\mathsf{t}).\mathsf{S}'$, for an $l < k-1$. The proof proceeds as already seen for $C[\cdot] = a((\tilde{x} : \tilde{\mathsf{t}}')\mathsf{U}).[\cdot]$.

$C[\cdot] = (\nu\tilde{d} : \tilde{\mathsf{t}})[\cdot]$. Since $\mathsf{S} \triangleq C[\mathsf{S}']$ we know that $\mathsf{S} = (\nu\tilde{d} : \tilde{\mathsf{t}})\mathsf{S}'$ and by (2) we get $P \equiv (\nu\tilde{d} : \tilde{\mathsf{t}}; \Phi)Q$ for some Q such that $\Gamma, \tilde{d} : \tilde{\mathsf{t}} \vdash^{l}_{\mathrm{L}} Q : \mathsf{S}'$, for an $l < k-1$, and $\mathsf{S}' \downarrow_{\tilde{d}} \models \Phi$. By $\mathsf{T} \equiv \mathsf{S}$ and $\mathsf{T} = C[\mathsf{T}']$ we get $\mathsf{S}' \equiv \mathsf{T}'$. By definition, $\mathsf{S}' \downarrow_{\tilde{d}} \equiv \mathsf{T}' \downarrow_{\tilde{d}}$ and, by closure of P-sets with respect to $\equiv$, $\mathsf{T}' \downarrow_{\tilde{d}} \models \Phi$. Finally, by (L-Eq), $\Gamma \vdash^{l+1}_{\mathrm{L}} Q : \mathsf{T}'$, with $l+1 < k$.

Concerning the cases of the rules for transitivity, symmetry and reflexivity the proof proceeds by applying the induction hypothesis.

The subject congruence and substitution properties hold for the local system as expected. Notice that, for subject congruence, the existence of normal derivations is a key point in order to guarantee the structural correspondence between processes and types.

Proposition B.3 (subject congruence, Proposition 3). $\Gamma \vdash_{\mathrm{L}} P : \mathsf{S}$ *and* $P \equiv Q$ *implies* $\Gamma \vdash_{\mathrm{L}} Q : \mathsf{S}$.

Proof. The proof proceeds by induction on the derivation of $P \equiv Q$ by distinguishing the last structural rule applied. The most interesting case is when the scope extension rule is the last one.

Suppose $P = (\nu\tilde{x} : \tilde{\mathsf{t}}; \Phi)P_1 \,|\, P_2$ and $Q = (\nu\tilde{x} : \tilde{\mathsf{t}}; \Phi)(P_1 \,|\, P_2)$, with $\tilde{x}\#P_2$.

By $\Gamma \vdash_{\mathrm{L}} P : \mathsf{S}$ and Proposition 2, $\Gamma \vdash_{\mathrm{NL}} P : \mathsf{T}$ for some $\mathsf{T} \equiv \mathsf{S}$. By $\Gamma \vdash_{\mathrm{NL}} P : \mathsf{T}$ and Lemma 4, we deduce that $\mathsf{T} = \mathsf{T}_1|\mathsf{T}_2$ with $\Gamma \vdash_{\mathrm{NL}} (\nu\tilde{x} : \tilde{\mathsf{t}}; \Phi)P_1 : \mathsf{T}_1$ and $\Gamma \vdash_{\mathrm{NL}} P_2 : \mathsf{T}_2$. Again by Lemma 4, $\mathsf{T}_1 = (\nu\tilde{x} : \tilde{\mathsf{t}})\mathsf{S}_1$, with $\Gamma, \tilde{x} : \tilde{\mathsf{t}} \vdash_{\mathrm{L}} P_1 : \mathsf{S}_1$ and $\mathsf{S}_1 \downarrow_{\tilde{x}} \models \Phi$. Moreover, by $\Gamma \vdash_{\mathrm{L}} P_2 : \mathsf{T}_2$, by $\tilde{x}\#\Gamma$ and Lemma B.1, we get $\tilde{x}\#\mathsf{T}_2$. Hence, $\mathsf{T} \equiv (\nu\tilde{x} : \tilde{\mathsf{t}})(\mathsf{S}_1|\mathsf{T}_2)$. By Proposition B.1 (weakening and contraction) and $\Gamma \vdash_{\mathrm{L}} P_2 : \mathsf{T}_2$, we deduce that $\Gamma, \tilde{x} : \tilde{\mathsf{t}} \vdash_{\mathrm{L}} P_2 : \mathsf{T}_2$. Hence, by (L-Par), $\Gamma, \tilde{x} : \tilde{\mathsf{t}} \vdash_{\mathrm{L}} P_1|P_2 : \mathsf{S}_1|\mathsf{T}_2$. By $\mathrm{Ok}(\Phi)$ and $\mathrm{fn}(\mathsf{T}_2 \downarrow_{\tilde{x}}) = \emptyset$, we get $(\mathsf{S}_1|\mathsf{T}_2) \downarrow_{\tilde{x}} \models \Phi$. Finally, by (L-Res), $\Gamma \vdash_{\mathrm{L}} (\nu\tilde{x} : \tilde{\mathsf{t}}; \Phi)(P_1|P_2) : (\nu\tilde{x} : \tilde{\mathsf{t}})(\mathsf{S}_1|\mathsf{T}_2)$ and by (L-Eq), $\Gamma \vdash_{\mathrm{L}} (\nu\tilde{x} : \tilde{\mathsf{t}}; \Phi)(P_1|P_2) : \mathsf{S}$.

In case $P = (\nu\tilde{x} : \tilde{\mathsf{t}}; \Phi)(P_1 \,|\, P_2)$ and $Q = (\nu\tilde{x} : \tilde{\mathsf{t}}; \Phi)P_1 \,|\, P_2$ the proof proceeds similarly.

Proposition B.4 (substitution, Proposition 4). *Suppose* $\Gamma, \tilde{x} : \tilde{\mathsf{t}} \vdash_{\mathrm{L}} P : \mathsf{T}$, *with* Γ *and* $\Gamma, \tilde{x} : \tilde{\mathsf{t}}$ *well-formed. Then* $\Gamma \vdash \tilde{b} : \tilde{\mathsf{t}}$ *implies* $\Gamma[^{\tilde{b}}/_{\tilde{x}}] \vdash_{\mathrm{L}} P[^{\tilde{b}}/_{\tilde{x}}] : \mathsf{T}[^{\tilde{b}}/_{\tilde{x}}]$.

Proof. The proof is by induction on the derivation of $\Gamma, \tilde{x} : \tilde{\mathsf{t}} \vdash_{\mathrm{L}} P : \mathsf{T}$. We proceed by distinguishing the last typing rule applied. As an example consider the case when (L-Out) is the last applied one. By $\Gamma, \tilde{x} : \tilde{\mathsf{t}} \vdash_{\mathrm{L}} \overline{a}\langle\tilde{c}\rangle.P : \overline{a}.(\mathsf{T}[^{\tilde{c}}/_{\tilde{y}}] \,|\, \mathsf{S})$ and the premise of the rule we get:

- $\Gamma, \tilde{x} : \tilde{\mathsf{t}} \vdash_{\mathrm{L}} a : (\tilde{y} : \tilde{\mathsf{t}}')\mathsf{T}$ (with $\tilde{y}\#\tilde{x}, \tilde{b}$ because they are bound in the type associated to a)
- $\Gamma, \tilde{x} : \tilde{\mathsf{t}} \vdash_{\mathrm{L}} \tilde{c} : \tilde{\mathsf{t}}'$
- $\Gamma, \tilde{x} : \tilde{\mathsf{t}} \vdash_{\mathrm{L}} P : \mathsf{S}$

By applying the induction hypothesis to $\Gamma, \tilde{x} : \tilde{\mathsf{t}} \vdash_{\mathrm{L}} P : \mathsf{S}$, we get $\Gamma[^{\tilde{b}}/_{\tilde{x}}] \vdash_{\mathrm{L}} P[^{\tilde{b}}/_{\tilde{x}}] : \mathsf{S}[^{\tilde{b}}/_{\tilde{x}}]$. Moreover, by definition, $\Gamma[^{\tilde{b}}/_{\tilde{x}}] \vdash_{\mathrm{L}} \tilde{c}[^{\tilde{b}}/_{\tilde{x}}] : \tilde{\mathsf{t}}'[^{\tilde{b}}/_{\tilde{x}}]$ and $\Gamma[^{\tilde{b}}/_{\tilde{x}}] \vdash_{\mathrm{L}} a[^{\tilde{b}}/_{\tilde{x}}] : (\tilde{y} : \tilde{\mathsf{t}}'[^{\tilde{b}}/_{\tilde{x}}])\mathsf{T}[^{\tilde{b}}/_{\tilde{x}}]$. Therefore, by (L-Out), $\Gamma[^{\tilde{b}}/_{\tilde{x}}] \vdash_{\mathrm{L}} \overline{a[^{\tilde{b}}/_{\tilde{x}}]}\langle\tilde{c}[^{\tilde{b}}/_{\tilde{x}}]\rangle.P[^{\tilde{b}}/_{\tilde{x}}] : \overline{a}[^{\tilde{b}}/_{\tilde{x}}].(\mathsf{T}[^{\tilde{b}}/_{\tilde{x}}][^{\tilde{c}[^{\tilde{b}}/_{\tilde{x}}]}/_{\tilde{y}}] \,|\, \mathsf{S}[^{\tilde{b}}/_{\tilde{x}}])$, that is $\Gamma[^{\tilde{b}}/_{\tilde{x}}] \vdash_{\mathrm{L}} (\overline{a}\langle\tilde{c}\rangle.P)[^{\tilde{b}}/_{\tilde{x}}] : (\overline{a}.(\mathsf{T}[^{\tilde{c}}/_{\tilde{y}}] \,|\, \mathsf{S}))[^{\tilde{b}}/_{\tilde{x}}]$.

As expected, the proof of Proposition 5 relies mostly on the definition of $\downarrow_{\tilde{x}}$.

Proposition B.5 (Proposition 5). *(i) If* $\mathsf{T} \xrightarrow{\lambda} \mathsf{T}'$*, with* $\lambda ::= \langle\epsilon\rangle \,\big|\, \langle a\rangle$ *and* $a \in \tilde{x}$*, then* $\mathsf{T}\downarrow_{\tilde{x}} \xrightarrow{\lambda} \mathsf{T}'\downarrow_{\tilde{x}}$*. (ii) If* $\mathsf{T} \xrightarrow{\lambda} \mathsf{T}'$*, with* $\lambda = \langle a\rangle$ *and* $a \notin \tilde{x}$*, then* $\mathsf{T}\downarrow_{\tilde{x}} \xrightarrow{\langle\epsilon\rangle}\xrightarrow{\langle\epsilon\rangle} \mathsf{T}'\downarrow_{\tilde{x}}$*. (iii) If* $\mathsf{T} \xrightarrow{s} \mathsf{T}'$ *then* $\mathsf{T}\downarrow_{\tilde{x}} \xrightarrow{s'} \mathsf{T}'\downarrow_{\tilde{x}}$*, with* $\mathrm{fn}(s') \subseteq \mathrm{fn}(s)$*. (iv) If* $\mathsf{T}\downarrow_{\tilde{x}} \xrightarrow{\lambda} \mathsf{T}'$*, with* $\lambda ::= \langle\epsilon\rangle \,\big|\, \langle a\rangle$ *and the* ϵ*-reduction originated by a synchronization on a bound name or a* τ *prefix in* T*, then* $\mathsf{T} \xrightarrow{\lambda} \mathsf{S}$*, with* $\mathsf{T}' = \mathsf{S}\downarrow_{\tilde{x}}$*.*

Proof. It is sufficient to prove (*i*) and (*ii*), (*iii*) is a consequence of the two. The proof of (*iv*) is along the line of that of (*i*) and is omitted.

Suppose $\lambda = \langle a\rangle$. The proof proceeds by induction on the derivation of $\mathsf{T} \xrightarrow{\lambda} \mathsf{T}'$. The most interesting case is when (com) has been applied. In the other cases the proof proceeds by applying the inductive hypothesis. By (com)

$$\mathsf{T} = \sum_{i\in I}\mu_i.\mathsf{T}_i \,|\, \sum_{j\in J}\mu'_j.\mathsf{S}_j$$

$\mu_l = a$, $\mu'_k = \overline{a}$, for some $l \in I$ and $k \in J$, and $\mathsf{T}' = \mathsf{T}_l \,|\, \mathsf{S}_k$.

Suppose $a \in \tilde{x}$.

$$\mathsf{T}\downarrow_{\tilde{x}} = \sum_{i\in I\setminus\{l\}} (\mu_i.\mathsf{T}_i)\downarrow_{\tilde{x}} + a.(\mathsf{T}_l\downarrow_{\tilde{x}}) \,|\, \sum_{j\in J\setminus\{k\}} (\mu'_j.\mathsf{S}_j) + \overline{a}.(\mathsf{S}_k\downarrow_{\tilde{x}})$$

and $\mathsf{T}\downarrow_{\tilde{x}} \xrightarrow{\lambda} \mathsf{T}_l\downarrow_{\tilde{x}} \,|\mathsf{S}_k\downarrow_{\tilde{x}} = \mathsf{T}'\downarrow_{\tilde{x}}$. This proves (*i*) in case $\lambda = \langle a\rangle$.

Suppose $a \notin \tilde{x}$.

$$\mathsf{T}\downarrow_{\tilde{x}} = \sum_{i\in I\setminus\{l\}} (\mu_i.\mathsf{T}_i)\downarrow_{\tilde{x}} + \tau.(\mathsf{T}_l\downarrow_{\tilde{x}}) \,|\, \sum_{j\in J\setminus\{k\}} (\mu'_j.\mathsf{S}_j) + \tau.(\mathsf{S}_k\downarrow_{\tilde{x}})$$

and $\mathsf{T}\downarrow_{\tilde{x}} \xrightarrow{\langle\epsilon\rangle}\xrightarrow{\langle\epsilon\rangle} \mathsf{T}_l\downarrow_{\tilde{x}} \,|\mathsf{S}_k\downarrow_{\tilde{x}} = \mathsf{T}'\downarrow_{\tilde{x}}$. This proves (*ii*).

Suppose now $\lambda = \langle\epsilon\rangle$. If the reduction originates by (tau) the same rule can be applied to $\mathsf{T}\downarrow_{\tilde{x}}$. If the reduction originates from a communication on a restricted name again the proof is by induction on the derivation of $\mathsf{T} \xrightarrow{\lambda} \mathsf{T}'$. The interesting case is when (res) is applied. In this case $\mathsf{T} = (\nu\tilde{a})\mathsf{S}$, $\mathsf{T}' = (\nu\tilde{a})\mathsf{S}'$ and $\mathsf{S} \xrightarrow{\lambda'} \mathsf{S}'$, for some λ' such that $\lambda = \lambda' \uparrow_{\tilde{a}}$. $\mathsf{T}\downarrow_{\tilde{x}} = (\nu\tilde{a})(\mathsf{S}\downarrow_{\tilde{x},\tilde{a}})$, hence by applying the induction hypothesis to $\mathsf{S} \xrightarrow{\lambda'} \mathsf{S}'$ we get $\mathsf{S}\downarrow_{\tilde{x},\tilde{a}} \xrightarrow{\lambda'} \mathsf{S}'\downarrow_{\tilde{x},\tilde{a}}$. Notice that if $\lambda' = \langle a\rangle$ for some $a \in \tilde{a}$ then $a \in \tilde{x},\tilde{a}$. Finally, by (res), $\mathsf{T}\downarrow_{\tilde{x}} \xrightarrow{\lambda} \mathsf{T}'\downarrow_{\tilde{x}}$. This proves (*i*) in case $\lambda = \langle a\rangle$.

We now prove type subject reduction, the last theorem of Section 4. This result is in some sense the inverse of the subject reduction property, indeed it guarantees the operational correspondence between types and processes. Again, the proof proceeds without surprises by an induction on the reduction rules.

Theorem B.1 (type subject reduction, Theorem 2). $\Gamma \vdash_{\mathrm{L}} P : \mathsf{T}$ *and* $\mathsf{T} \xrightarrow{\lambda} \mathsf{T}'$ *implies that there exists a* P' *such that* $P \xrightarrow{\lambda} P'$ *and* $\Gamma \vdash_{\mathrm{L}} P' : \mathsf{T}'$*.*

Proof. The proof proceeds by induction on the derivation of $\mathsf{T} \xrightarrow{\lambda} \mathsf{T}'$ by distinguishing the last reduction rule applied.

(com). Assume for notational simplicity that $\mathsf{T} = a((\tilde{x} : \tilde{\mathsf{t}})\mathsf{S}).\mathsf{U}|\overline{a}.\mathsf{T}' \xrightarrow{\langle a \rangle} \mathsf{U}|\mathsf{T}'$ (the general case of arbitrary summations is similar). By $\Gamma \vdash_{\mathrm{L}} P : \mathsf{T}$ and Lemma 5, we get $P \equiv a(\tilde{x}).R|\overline{a}\langle\tilde{b}\rangle.Q$ with $\Gamma, \tilde{x} : \tilde{\mathsf{t}} \vdash_{\mathrm{L}} R : \mathsf{U}|\mathsf{S}$, $\mathsf{T}' \equiv \mathsf{S}[\tilde{b}/\tilde{x}]|\mathsf{S}'$, $\Gamma \vdash_{\mathrm{L}} Q : \mathsf{S}'$, $\tilde{x}\#\mathsf{U}$, $\Gamma \vdash a : (\tilde{x} : \tilde{\mathsf{t}})\mathsf{S}$ and $\Gamma \vdash \tilde{b} : \tilde{\mathsf{t}}$.

By (com) and (struct), $P \xrightarrow{\langle a \rangle} P' \equiv R[\tilde{b}/\tilde{x}]|Q$. Moreover, by Proposition B.4 (substitution) and (L-Par) we get $\Gamma \vdash_{\mathrm{L}} R[\tilde{b}/\tilde{x}]|Q : (\mathsf{U}|\mathsf{S})[\tilde{b}/\tilde{x}]|\mathsf{S}' = \mathsf{U}|\mathsf{S}[\tilde{b}/\tilde{x}]|\mathsf{S}'$. Hence, by (L-Eq), $\Gamma \vdash_{\mathrm{L}} R[\tilde{b}/\tilde{x}]|Q : \mathsf{U}|\mathsf{T}'$, and by Proposition 3 (subject congruence) $\Gamma \vdash_{\mathrm{L}} P' : \mathsf{U}|\mathsf{T}'$.

(rep-com). The proof proceeds similarly to (com).

(tau). By $\sum_{i \in I} \mu_i.\mathsf{T}_i \xrightarrow{\langle \epsilon \rangle} \mathsf{T}_j$ and the premise of the rule, we get $\mu_j = \tau$. By Lemma 5, $\Gamma \vdash_{\mathrm{L}} P : \sum_{i \in I} \mu_i.\mathsf{T}_i$ implies $P \equiv \sum_{i \in I} \alpha_i.P_i$ and for each $i \in I$ it holds that $\Gamma \vdash_{\mathrm{L}} \alpha_i.P_i : \mu_i.\mathsf{T}_i$. Again by Lemma 5 and $\mu_j.\mathsf{T}_j = \tau.\mathsf{T}_j$ we get $\alpha_j = \tau$ and $\Gamma \vdash_{\mathrm{L}} P_j : \mathsf{T}_j$. Finally, by (tau), $\sum_{i \in I} \alpha_i.P_i \xrightarrow{\langle \epsilon \rangle} P_j$ and by (struct), $P \xrightarrow{\langle \epsilon \rangle} P'$ with $P' \equiv P_j$ and $\Gamma \vdash_{\mathrm{L}} P' : \mathsf{T}_j$ by Proposition 3 (subject congruence).

(par). By $\mathsf{S}|\mathsf{U} \xrightarrow{\lambda} \mathsf{S}'|\mathsf{U}$ and the premise of the rule, we get $\mathsf{S} \xrightarrow{\lambda} \mathsf{S}'$. By $\Gamma \vdash_{\mathrm{L}} P : \mathsf{S}|\mathsf{U}$ and Lemma 5, we get $P \equiv Q|R$ with $\Gamma \vdash_{\mathrm{L}} Q : \mathsf{S}$ and $\Gamma \vdash_{\mathrm{L}} R : \mathsf{U}$. Hence, by applying the induction hypothesis, we get $Q \xrightarrow{\lambda} Q'$ and $\Gamma \vdash_{\mathrm{L}} Q' : \mathsf{S}'$. By (L-Par), we get $\Gamma \vdash_{\mathrm{L}} Q'|R : \mathsf{S}'|\mathsf{U}$ and by (par), $Q|R \xrightarrow{\lambda} Q'|R$. Moreover, by (struct), $P \equiv Q|R$ and $Q|R \xrightarrow{\lambda} Q'|R$, we get $P \xrightarrow{\lambda} P'$ with $P' \equiv Q'|R$, and by Proposition 3 (subject congruence), $\Gamma \vdash_{\mathrm{L}} P' : \mathsf{S}'|\mathsf{U}$.

(struct). By $\mathsf{T} \xrightarrow{\lambda} \mathsf{T}'$ and the premise of the rule, we get $\mathsf{T} \equiv \mathsf{S}$, $\mathsf{S} \xrightarrow{\lambda} \mathsf{S}'$ and $\mathsf{S}' \equiv \mathsf{T}'$. By (L-Eq), $\Gamma \vdash_{\mathrm{L}} P : \mathsf{T}$ implies $\Gamma \vdash_{\mathrm{L}} P : \mathsf{S}$; hence, by applying the induction hypothesis, we get $P \xrightarrow{\lambda} P'$ and $\Gamma \vdash_{\mathrm{L}} P' : \mathsf{S}'$. Again by (L-Eq), $\Gamma \vdash_{\mathrm{L}} P' : \mathsf{T}'$.

(res). By $(\nu\tilde{a} : \tilde{\mathsf{t}})\mathsf{T} \xrightarrow{\lambda \uparrow \tilde{a}} (\nu\tilde{a} : \tilde{\mathsf{t}})\mathsf{T}'$ and the premise of the rule, we get $\mathsf{T} \xrightarrow{\lambda} \mathsf{T}'$. By $\Gamma \vdash_{\mathrm{L}} P : (\nu\tilde{a} : \tilde{\mathsf{t}})\mathsf{T}$ and Lemma 5, we get $P \equiv (\nu\tilde{a} : \tilde{\mathsf{t}}; \Phi)Q$ with $\Gamma, \tilde{a} : \tilde{\mathsf{t}} \vdash_{\mathrm{L}} Q : \mathsf{T}$ and $\mathsf{T} \downarrow_{\tilde{a}} \models \Phi$. Hence, by applying the induction hypothesis, we get $Q \xrightarrow{\lambda} Q'$ and $\Gamma \vdash_{\mathrm{L}} Q' : \mathsf{T}'$. By (res) we get $(\nu\tilde{a} : \tilde{\mathsf{t}}; \Phi)Q \xrightarrow{\lambda \uparrow \tilde{a}} (\nu\tilde{a} : \tilde{\mathsf{t}}; \Phi_\lambda)Q'$ and by (struct), $P \xrightarrow{\lambda \uparrow \tilde{a}} P'$ with $P' \equiv (\nu\tilde{a} : \tilde{\mathsf{t}}; \Phi_\lambda)Q'$.

We have to prove that $\Gamma \vdash_{\mathrm{L}} (\nu\tilde{a} : \tilde{\mathsf{t}}; \Phi_\lambda)Q' : (\nu\tilde{a} : \tilde{\mathsf{t}})\mathsf{T}'$, in particular that $\mathsf{T}' \downarrow_{\tilde{a}} \models \Phi_\lambda$. We consider two possibilities separately (recall that $\tilde{a} \supseteq \mathrm{supp}(\Phi)$).

1. $\lambda = \langle a \rangle$, with $a \in \tilde{a}$, or $\lambda = \langle \epsilon \rangle$. $\mathsf{T}' \downarrow_{\tilde{a}} \models \Phi_\lambda$ follows by $\mathsf{T} \downarrow_{\tilde{a}} \models \Phi$ and $\mathsf{T} \downarrow_{\tilde{a}} \xrightarrow{\lambda} \mathsf{T}' \downarrow_{\tilde{a}}$, a consequence of $\mathsf{T} \xrightarrow{\lambda} \mathsf{T}'$.
2. otherwise $\Phi_\lambda = \Phi_{\langle \epsilon \rangle} = \Phi$, by Ok($\Phi$). The communication $\mathsf{T} \xrightarrow{\lambda} \mathsf{T}'$ has a free subject not in $\tilde{a}$. By Proposition 5, this reduction can be simulated by a pair of reductions that consume the corresponding prefixes, thus, when hiding, we get $\mathsf{T} \downarrow_{\tilde{a}} \xrightarrow{\langle \epsilon \rangle} \xrightarrow{\langle \epsilon \rangle} \mathsf{T}' \downarrow_{\tilde{a}}$ and, by definition, $\mathsf{T}' \downarrow_{\tilde{a}} \models \Phi_{\langle \epsilon \rangle \langle \epsilon \rangle} = \Phi_{\langle \epsilon \rangle} = \Phi_\lambda = \Phi$.

In both cases, (L-Res) can be applied for deducing $\Gamma \vdash_{\mathrm{L}} (\nu\tilde{a} : \tilde{\mathsf{t}}; \Phi_\lambda)Q' : (\nu\tilde{a} : \tilde{\mathsf{t}})\mathsf{T}'$. Hence, by Proposition 3 (subject congruence), $\Gamma \vdash_{\mathrm{L}} P' : (\nu\tilde{a} : \tilde{\mathsf{t}})\mathsf{T}'$.

Concerning the local system, there is still to prove the structural correspondence between types and their “hidden” versions.

Lemma B.3 (Lemma 6).

1. *Suppose $a \in \tilde{x}$.* $(\mathsf{T}\downarrow_{\tilde{x}}) \searrow_a$ *if and only if* $\mathsf{T} \searrow_a$.
2. *If* $\mathsf{T}\downarrow_{\tilde{x}} \equiv \mathsf{T}_1|\mathsf{T}_2$ *then there are* S_1 *and* S_2 *such that* $\mathsf{T} \equiv \mathsf{S}_1|\mathsf{S}_2$ *and* $\mathsf{S}_i \downarrow_{\tilde{x}} = \mathsf{T}_i$*, for* $i = 1,2$.
3. *If* $\mathsf{T} \equiv \mathsf{T}_1|\mathsf{T}_2$ *then there are* S_1 *and* S_2 *such that* $\mathsf{T}\downarrow_{\tilde{x}} \equiv \mathsf{S}_1|\mathsf{S}_2$ *and* $\mathsf{S}_i = \mathsf{T}_i \downarrow_{\tilde{x}}$*, for* $i = 1,2$.
4. *If* $\mathsf{T}\downarrow_{\tilde{x}} \equiv (\tilde{\nu a})\mathsf{S}$ *then there is* V *such that* $\mathsf{T} \equiv (\tilde{\nu a})\mathsf{V}$*, with* $\mathsf{V}\downarrow_{\tilde{x},\tilde{a}} = \mathsf{S}$.
5. *If* $\mathsf{T} \equiv (\tilde{\nu a})\mathsf{S}$ *then there is* V *such that* $\mathsf{T}\downarrow_{\tilde{x}} \equiv (\tilde{\nu a})\mathsf{V}$*, with* $\mathsf{V} = \mathsf{S}\downarrow_{\tilde{x},\tilde{a}}$.

Proof. Point (1) follows by definition of $\mathsf{T}\downarrow_{\tilde{x}}$ (Table 4). Points (2-5) can be proved by mutual induction on the derivation of $\equiv$. As an example, consider (4) and suppose the last structural rule applied is scope extension. Hence

$$\mathsf{T}\downarrow_{\tilde{x}} \equiv (\tilde{\nu a})\mathsf{S}_1|\mathsf{S}_2 \equiv (\tilde{\nu a})(\mathsf{S}_1|\mathsf{S}_2) \stackrel{\triangle}{=} (\tilde{\nu a})\mathsf{S}$$

with $\tilde{a}\#\,\mathrm{fn}(\mathsf{S}_2)$.

By applying the inductive hypothesis to $\mathsf{T}\downarrow_{\tilde{x}} \equiv (\tilde{\nu a})\mathsf{S}_1|\mathsf{S}_2$ we get $\mathsf{T} \equiv \mathsf{T}_1|\mathsf{T}_2$ with $\mathsf{T}_1\downarrow_{\tilde{x}} = (\tilde{\nu a})\mathsf{S}_1$ and $\mathsf{T}_2\downarrow_{\tilde{x}} = \mathsf{S}_2$. Again by induction, from $\mathsf{T}_1\downarrow_{\tilde{x}} = (\tilde{\nu a})\mathsf{S}_1$ we get $\mathsf{T}_1 \equiv (\tilde{\nu a})\mathsf{U}$, with $\mathsf{U}\downarrow_{\tilde{x},\tilde{a}} = \mathsf{S}_1$.

Without loss of generality, assume $\tilde{a}\#\,\mathrm{fn}(\mathsf{T}_2)$. Notice that $\tilde{a}$ can always be renamed with some $\tilde{b}$ such that $\tilde{b}\#\,\mathrm{fn}(\mathsf{T}_2)$. By scope extension, $(\tilde{\nu a})\mathsf{U}|\mathsf{T}_2 \equiv (\tilde{\nu a})(\mathsf{U}|\mathsf{T}_2) \stackrel{\triangle}{=} (\tilde{\nu a})\mathsf{V}$. By $\tilde{a}\#\,\mathrm{fn}(\mathsf{T}_2)$ and $\mathsf{T}_2\downarrow_{\tilde{x}} = \mathsf{S}_2$, we get $\mathsf{T}_2\downarrow_{\tilde{x},\tilde{a}} = \mathsf{S}_2$. Hence, by $\mathsf{U}\downarrow_{\tilde{x},\tilde{a}} = \mathsf{S}_1$ we get $\mathsf{V}\downarrow_{\tilde{x},\tilde{a}} = \mathsf{S}$. From this and $\mathsf{T} \equiv (\tilde{\nu a})\mathsf{V}$ we get the result.

C. Proofs of Section 7

Before proving the basic properties of the global system some additional notations and some preliminary results concerning properties of critical names are discussed. Definitions of $\mathrm{cr}(\cdot)$ and $(\cdot)[\tilde{y}]$ are extended to channel types as follow:

$$\mathrm{cr}(\mathrm{t}) = \mathrm{cr}(G_{\mathrm{t}}) \quad \text{and} \quad \mathrm{t}[\tilde{y}] = G_{\mathrm{t}}[\tilde{y}]$$

and to tuples of channel types $\tilde{\mathrm{t}}$ component-wise as expected.

The following lemma states a few properties of critical names that follows by definition of $\mathrm{cr}(\cdot)$. For the sake of completeness, we list all the properties we will need in the following.

Lemma C.1.

1. $\mathrm{cr}(\mathsf{T}|\mathsf{S}) \supseteq \mathrm{cr}(\mathsf{T}) \cup \mathrm{cr}(\mathsf{S})$*;*
2. $\mathrm{cr}(a(\mathrm{t}).\mathsf{T}) \supseteq \mathrm{cr}(\mathsf{T}) \cup \mathrm{cr}(\mathrm{t})$*;*
3. $\mathrm{cr}(\overline{a}.\mathsf{T}) = \mathrm{cr}(\mathsf{T})$*;*
4. $\mathrm{cr}(!a(\mathrm{t}).\mathsf{T}) = \mathrm{cr}(a(\mathrm{t}).\mathsf{T})$*;*
5. $\mathrm{cr}((\nu\tilde{x} : \tilde{\mathrm{t}})\mathsf{T}) \supseteq (\mathrm{cr}(\tilde{\mathrm{t}}) \cup \mathrm{cr}(\mathsf{T}) \cup \mathsf{T}[\tilde{x}]) \setminus \tilde{x}$*;*
6. $\mathrm{cr}(\sum_i \mu_i.\mathsf{T}_i) \supseteq \bigcup_i \mathrm{cr}(\mu_i.\mathsf{T}_i)$*;*
7. $\mathrm{cr}(\mathsf{T})\#\mathsf{S}$ *and* $\mathrm{cr}(\mathsf{S})\#\mathsf{T}$ *imply* $\mathrm{cr}(\mathsf{T}|\mathsf{S}) = \mathrm{cr}(\mathsf{T}) \cup \mathrm{cr}(\mathsf{S})$*;*
8. $\mathsf{T}[\tilde{x}]\#\mathsf{S}$ *implies* $(\mathsf{T}|\mathsf{S})[\tilde{x}] = \mathsf{T}[\tilde{x}]$*;*
9. $\mathsf{T} \xrightarrow{\lambda} \mathsf{T}'$ *implies* $\mathsf{T}[\tilde{x}] \supseteq \mathsf{T}'[\tilde{x}]$*;*

10. *suppose* $\mathsf{T}[\tilde{x}] \neq \emptyset$*; then* $a \in ((!)a(\mathsf{t}).\mathsf{T})[\tilde{x}]$ *and* $a \in (\overline{a}.\mathsf{T})[\tilde{x}]$*;*
11. *suppose* $\mathsf{T} \equiv (\tilde{\nu d})(\sum_i \mathsf{U}_i + a(\mathsf{t}).\mathsf{S}\,|\,\mathsf{V})$ *(resp.* $\mathsf{T} \equiv (\tilde{\nu d})(\sum_i \mathsf{U}_i + \overline{a}.\mathsf{S}\,|\,\mathsf{V})$ *or* $\mathsf{T} \equiv (\tilde{\nu d})(!a(\mathsf{t}).\mathsf{S}\,|\,\mathsf{V})$*). If* $a \notin \mathsf{T}[\tilde{x}]$ *then* $\mathrm{fn}(a(\mathsf{t}).\mathsf{S})\#\mathsf{T}[\tilde{x}]$ *(resp.* $\mathrm{fn}(\overline{a}.\mathsf{S})\#\mathsf{T}[\tilde{x}]$ *or* $\mathrm{fn}(!a(\mathsf{t}).\mathsf{S})\#\mathsf{T}[\tilde{x}]$*);*
12. *suppose* $\tilde{a}, \tilde{b}\#\mathrm{cr}(\mathsf{T})$*; then* $\mathrm{cr}(\mathsf{T}) = \mathrm{cr}(\mathsf{T}[^{\tilde{a}}/_{\tilde{b}}])$.

Proof. (*1-7*) follow by definition of $\mathrm{cr}(\mathsf{T})$ and of G_{T}. (*8*) follows by definition of $\mathsf{T}[\tilde{x}]$ and of G_{T}. (*9*) follows by definition of G_{T}: it can be easily seen that $G_{\mathsf{T}'}$ can be embedded into G_{T}. (*10-11*) follow by definition of G_{T} and $\mathsf{T}[\tilde{x}]$. In the rest of the proof we consider (*12*). For the sake of simplicity, suppose that $\tilde{a} = a$ and $\tilde{b} = b$. The proof can be easily generalized to the case of a generic substitution. Note that by $a, b\#\mathrm{cr}(\mathsf{T})$ we get $\mathrm{cr}(\mathsf{T}) \subseteq \mathrm{cr}(\mathsf{T}[^a/_b])$: indeed, since $b \notin \mathrm{cr}(\mathsf{T})$, each critical path in G_{T} is also found in $G_{\mathsf{T}[^a/_b]}$. Any other path in G_{T} corresponds to one in $G_{\mathsf{T}[^a/_b]}$ with each occurrence of b replaced by a. Suppose e.g.

$$\pi = x_1 \to \cdots \to x_n \to y^\bullet \text{ is a path in } G_{\mathsf{T}}$$

where, for each i, $x_i = (\nu\tilde{y})$ implies $y \notin \tilde{y}$: then π is a path in $G_{\mathsf{T}[^a/_b]}$ because b does not occur in π. Concerning the reverse inclusion, consider any $f \in \mathrm{cr}(\mathsf{T}[^a/_b])$. By definition, this means that in $G_{\mathsf{T}[^a/_b]}$ there exists a path π' of the form

$$\pi' = x_1 \to \cdots \to x_j (= f) \to \cdots \to x_n \to y^\bullet$$

where, for each i, $x_i = (\nu\tilde{y})$ implies $y \notin \tilde{y}$. Clearly, b does not occur in π'. If a does not occur either, then π' is also a path of G_{T}, hence $f \in \mathrm{cr}(\mathsf{T})$. Otherwise, let x_k be the rightmost occurrence of a in π'. In G_{T}, we therefore have the subpath $x_{k+1} \to \cdots \to y^\bullet$ and either $a \to x_{k+1}$ or $b \to x_{k+1}$. In the former case, we would get $a \in \mathrm{cr}(\mathsf{T})$, in the latter $b \in \mathrm{cr}(\mathsf{T})$, contradicting the assumption in both cases.

Lemma 2 extends to the property-type simulation relation $\propto_{\tilde{x}}$ as expected.

Lemma C.2. *Let* Φ *be a P-set such that* $\mathrm{Ok}(\Phi)$ *and* $\tilde{a} \supseteq \mathrm{supp}(\Phi)$*. If* $\Phi \propto_{\tilde{a}} \mathsf{T}$ *and* $\mathsf{T}[\tilde{a}]\#\mathsf{S}$ *then* $\Phi \propto_{\tilde{a}} (\mathsf{T}\,|\,\mathsf{S})$.

Proof. Define

$$\mathcal{R} \triangleq \{(\Phi, \mathsf{T}|\mathsf{U}) \mid \tilde{a} \supseteq \mathrm{supp}(\Phi),\ \Phi \propto_{\tilde{a}} \mathsf{T},\ \mathrm{Ok}(\Phi),\ \mathsf{T}[\tilde{a}]\#\mathsf{U}\}\,.$$

We prove that $\mathcal{R} \subseteq \propto_{\tilde{a}}$. That is, for each pair $(\Phi, \mathsf{T}|\mathsf{U}) \in \mathcal{R}$ we prove:

1. $(\mathsf{T}|\mathsf{U}) \downarrow_{(\mathsf{T}|\mathsf{U})[\tilde{a}]} \models \Phi$;
2. $\forall \mathsf{V}, \gamma$ such that $(\mathsf{T}|\mathsf{U}) \succ^{\gamma}_{(\mathsf{T}|\mathsf{U})[\tilde{a}]} \mathsf{V}$ it holds that $(\Phi_{\gamma\downarrow_{(\mathsf{T}|\mathsf{U})[\tilde{a}]}}, \mathsf{V}) \in \mathcal{R}$.

We prove separately the two points. First note that by $\mathsf{T}[\tilde{a}]\#\mathsf{U}$ and Lemma C.1 (8) we get $(\mathsf{T}|\mathsf{U})[\tilde{a}] = \mathsf{T}[\tilde{a}]$ and $\mathrm{fn}(\mathsf{U} \downarrow_{\mathsf{T}[\tilde{a}]}) = \emptyset$

1. By definition of $\mathcal{R}$ we get $\Phi \propto_{\tilde{a}} \mathsf{T}$, hence $\mathsf{T} \downarrow_{\mathsf{T}[\tilde{a}]} \models \Phi$. Therefore, by $\mathrm{Ok}(\Phi)$, $\mathrm{fn}(\mathsf{U} \downarrow_{\mathsf{T}[\tilde{a}]}) = \emptyset$ and $\mathsf{T} \downarrow_{\mathsf{T}[\tilde{a}]} \models \Phi$, we get $\mathsf{T} \downarrow_{\mathsf{T}[\tilde{a}]} | \mathsf{U} \downarrow_{\mathsf{T}[\tilde{a}]} \models \Phi$, hence $(\mathsf{T}|\mathsf{U}) \downarrow_{\mathsf{T}[\tilde{a}]} \models \Phi$, that is $(\mathsf{T}|\mathsf{U}) \downarrow_{(\mathsf{T}|\mathsf{U})[\tilde{a}]} \models \Phi$.
2. Take any $\mathsf{V} \prec^{\gamma}_{(\mathsf{T}|\mathsf{U})[\tilde{a}]} \mathsf{T}|\mathsf{U}$. By definition (page 21), $\gamma ::= \tau \mid \langle a \rangle \mid b \mid \overline{b}$, with $a \in (\mathsf{T}|\mathsf{U})[\tilde{a}]$ and $b \notin (\mathsf{T}|\mathsf{U})[\tilde{a}]$, such that $\mathsf{T}|\mathsf{U} \xrightarrow{\gamma} \mathsf{V}$. We distinguish the following cases.
   - Suppose $\mathsf{U} \xrightarrow{\gamma} \mathsf{U}'$ and $\mathsf{V} = \mathsf{T}|\mathsf{U}'$. Given that $\mathsf{T}[\tilde{a}]\#\mathsf{U}$ and $(\mathsf{T}|\mathsf{U})[\tilde{a}] = \mathsf{T}[\tilde{a}]$ we get $(\mathsf{T}|\mathsf{U})[\tilde{a}]\#\gamma$. Hence, $\gamma \downarrow_{(\mathsf{T}|\mathsf{U})[\tilde{a}]} = \langle\epsilon\rangle$ and, by $\mathrm{Ok}(\Phi)$, $\Phi_{\langle\epsilon\rangle} = \Phi$. Given that $\mathrm{fn}(\mathsf{U}') \subseteq \mathrm{fn}(\mathsf{U})$ and $\mathsf{T}[\tilde{a}]\#\mathsf{U}$ we get $(\Phi, \mathsf{T}|\mathsf{U}') \in \mathcal{R}$ by definition. Finally, given that $\Phi_{\gamma\downarrow_{(\mathsf{T}|\mathsf{U})[\tilde{a}]}} = \Phi$, we get $(\Phi_{\gamma\downarrow_{(\mathsf{T}|\mathsf{U})[\tilde{a}]}}, \mathsf{T}|\mathsf{U}') \in \mathcal{R}$.

- Suppose $\mathsf{T} \xrightarrow{\gamma} \mathsf{T}'$ and $\mathsf{V} = \mathsf{T}'|\mathsf{U}$. By $(\mathsf{T}|\mathsf{U})[\tilde{a}] = \mathsf{T}[\tilde{a}]$, by definition of $\succ^{\gamma}_{\tilde{x}}$ and by $\mathsf{T} \succ^{\gamma}_{(\mathsf{T}|\mathsf{U})[\tilde{a}]} \mathsf{T}'$, it follows that $\mathsf{T} \succ^{\gamma}_{\mathsf{T}[\tilde{a}]} \mathsf{T}'$. Consequently, by Definition 9, $\Phi_{\gamma\downarrow_{\mathsf{T}[\tilde{a}]}} \propto_{\tilde{a}} \mathsf{T}'$. Finally, given that $\mathsf{T}[\tilde{a}]\#\mathsf{U}$, $\mathsf{T}[\tilde{a}] \supseteq \mathsf{T}'[\tilde{a}]$ (Lemma C.1 (9)) and $(\mathsf{T}|\mathsf{U})[\tilde{a}] = \mathsf{T}[\tilde{a}]$, we get $(\Phi_{\gamma\downarrow_{(\mathsf{T}|\mathsf{U})[\tilde{a}]}}, \mathsf{T}'|\mathsf{U}) \in \mathcal{R}$ by definition.
- Suppose now that T and U interact and suppose the subject of the communication is b. Given that $\mathsf{T}[\tilde{a}]\#\mathsf{U}$ we get $b \notin \mathsf{T}[\tilde{a}]$ and either $\mathsf{T} \xrightarrow{b} \mathsf{T}'$, $\mathsf{U} \xrightarrow{\overline{b}} \mathsf{U}'$ and $\mathsf{T} \succ^{b}_{\mathsf{T}[\tilde{a}]} \mathsf{T}'$ or $\mathsf{T} \xrightarrow{\overline{b}} \mathsf{T}'$, $\mathsf{U} \xrightarrow{b} \mathsf{U}'$ and $\mathsf{T} \succ^{\overline{b}}_{\mathsf{T}[\tilde{a}]} \mathsf{T}'$. Given that $\mathsf{T}[\tilde{a}]\#\mathsf{U}$, $\mathrm{fn}(\mathsf{U}') \subseteq \mathrm{fn}(\mathsf{U})$ and $\mathsf{T}[\tilde{a}] \supseteq \mathsf{T}'[\tilde{a}]$ (Lemma C.1 (9)) we get $\mathsf{T}'[\tilde{a}]\#\mathsf{U}'$.

  Suppose $\mathsf{T} \xrightarrow{b} \mathsf{T}'$ (the proof proceeds similarly in the opposite case). Then by Definition 9, $\Phi_{b\downarrow_{\mathsf{T}[\tilde{a}]}} \propto_{\tilde{a}} \mathsf{T}'$. From $(\mathsf{T}|\mathsf{U})[\tilde{a}] = \mathsf{T}[\tilde{a}]$ and $\Phi_{b\downarrow_{\mathsf{T}[\tilde{a}]}} \propto_{\tilde{a}} \mathsf{T}'$ we get $\Phi_{b\downarrow_{(\mathsf{T}|\mathsf{U})[\tilde{a}]}} \propto_{\tilde{a}} \mathsf{T}'$. Moreover, by $\mathrm{Ok}(\Phi)$ we get $\Phi_{b\downarrow_{(\mathsf{T}|\mathsf{U})[\tilde{a}]}} = \Phi_{\langle b\rangle\downarrow_{(\mathsf{T}|\mathsf{U})[\tilde{a}]}} = \Phi$. Finally, by definition of $\mathcal{R}$: $(\Phi_{\langle b\rangle\downarrow_{(\mathsf{T}|\mathsf{U})[\tilde{a}]}}, \mathsf{T}'|\mathsf{U}') \in \mathcal{R}$.

Property-type simulation is preserved by (non-interfering) substitutions.

Lemma C.3. *Let* Φ *be a P-set such that* $\mathrm{Ok}(\Phi)$ *and* $\tilde{a} \supseteq \mathrm{supp}(\Phi)$. $\Phi \propto_{\tilde{a}} \mathsf{T}$ *and* $(\mathrm{dom}(\sigma) \cup \mathrm{ran}(\sigma))\#\mathsf{T}[\tilde{a}]$ *imply* $\Phi \propto_{\tilde{a}} \mathsf{T}\sigma$.

Proof. Define

$$\mathcal{R} \triangleq \{(\Phi, \mathsf{T}\sigma) \mid \tilde{a} \supseteq \mathrm{supp}(\Phi),\ \Phi \propto_{\tilde{a}} \mathsf{T},\ \mathrm{Ok}(\Phi),\ (\mathrm{dom}(\sigma) \cup \mathrm{ran}(\sigma))\#\mathsf{T}[\tilde{a}]\}\,.$$

We prove that $\mathcal{R} \subseteq \propto_{\tilde{a}}$. That is, for each pair $(\Phi, \mathsf{T}\sigma) \in \mathcal{R}$ we prove:

1. $\mathsf{T}\sigma \downarrow_{\mathsf{T}\sigma[\tilde{a}]} \models \Phi$;
2. $\forall \mathsf{V} : \mathsf{V} \prec^{\gamma}_{\mathsf{T}\sigma[\tilde{a}]} \mathsf{T}\sigma$ it holds that $(\Phi_{\gamma\downarrow_{\mathsf{T}\sigma[\tilde{a}]}}, \mathsf{V}) \in \mathcal{R}$.

We prove separately the two points.

1. By definition of $\mathcal{R}$ we get $\Phi \propto_{\tilde{a}} \mathsf{T}$, hence $\mathsf{T} \downarrow_{\mathsf{T}[\tilde{a}]} \models \Phi$. By $(\mathrm{dom}(\sigma) \cup \mathrm{ran}(\sigma))\#\mathsf{T}[\tilde{a}]$ and Lemma C.1 (12) we get $\mathsf{T}\sigma[\tilde{a}] = \mathsf{T}[\tilde{a}]$. Therefore, $\mathsf{T} \downarrow_{\mathsf{T}[\tilde{a}]} = \mathsf{T}\sigma \downarrow_{\mathsf{T}\sigma[\tilde{a}]}$ and $\mathsf{T}\sigma \downarrow_{\mathsf{T}\sigma[\tilde{a}]} \models \Phi$.
2. Take any V such that $\mathsf{V} \prec^{\gamma}_{\mathsf{T}\sigma[\tilde{a}]} \mathsf{T}\sigma$. By definition of $\succ^{\gamma}_{\tilde{x}}$ we get that $\mathsf{T}\sigma \xrightarrow{\gamma} \mathsf{V}$ for some $\gamma ::= \langle\epsilon\rangle \mid \langle a\rangle \mid b \mid \overline{b}$, with $a \in \mathsf{T}\sigma[\tilde{a}]$ or $b \notin \mathsf{T}\sigma[\tilde{a}]$. By $(\mathrm{dom}(\sigma) \cup \mathrm{ran}(\sigma))\#\mathsf{T}[\tilde{a}]$ and Lemma C.1 (12) it follows that $\mathsf{T}\sigma[\tilde{a}] = \mathsf{T}[\tilde{a}]$. Moreover, $\mathsf{T}\sigma \xrightarrow{\gamma} \mathsf{V}$ implies $\mathsf{T} \xrightarrow{\gamma'} \mathsf{V}'$, with $\gamma'\sigma = \gamma$ and $\mathsf{V}'\sigma = \mathsf{V}$. Notice also that $(\mathrm{dom}(\sigma) \cup \mathrm{ran}(\sigma))\#\mathsf{T}[\tilde{a}]$ and $\gamma ::= \langle\epsilon\rangle \mid \langle a\rangle \mid b \mid \overline{b}$, with $a \in \mathsf{T}\sigma[\tilde{a}]$ or $b \notin \mathsf{T}\sigma[\tilde{a}]$ imply $\gamma' ::= \langle\epsilon\rangle \mid \langle a\rangle \mid c \mid \overline{c}$ with $c \notin \mathsf{T}\sigma[\tilde{a}]$. By $\Phi \propto_{\tilde{a}} \mathsf{T}$ and Definition 9 we get $\Phi_{\gamma'\downarrow_{\mathsf{T}[\tilde{a}]}} \propto_{\tilde{a}} \mathsf{V}'$. In addition, $(\gamma' \downarrow_{\mathsf{T}[\tilde{a}]})\sigma = \gamma \downarrow_{\mathsf{T}\sigma[\tilde{a}]}$. Finally, by $\Phi_{\gamma'\downarrow_{\mathsf{T}[\tilde{a}]}} = \Phi_{\gamma\downarrow_{\mathsf{T}\sigma[\tilde{a}]}}$, it follows that $(\Phi_{\gamma\downarrow_{\mathsf{T}\sigma[\tilde{a}]}}, \mathsf{V}) \in \mathcal{R}$.

Proposition C.1 (normal derivations, Proposition 7). $\Gamma \vdash^{+}_{\mathrm{G}} P : \mathsf{T}$ *implies that there are* $R \equiv P$ *and* $\mathsf{S} \equiv \mathsf{T}$ *such that* R *and* S *are in head normal form and* $\Gamma \vdash^{+}_{\mathrm{NG}} R : \mathsf{S}$.

Proof. The proof proceeds by induction on the derivation of $\Gamma \vdash^{+}_{\mathrm{G}} P : \mathsf{T}$ by distinguishing the last typing rule applied in the derivation:

(G-Inp). By $\Gamma \vdash^{+}_{\mathrm{G}} a(\tilde{x}).P : a((\tilde{x} : \tilde{\mathsf{t}})\mathsf{T}).\mathsf{T}'$ and the premise of the rule, we get $\Gamma \vdash^{+}_{\mathrm{G}} a : (\tilde{x} : \tilde{\mathsf{t}})\mathsf{T}$ and $\Gamma, \tilde{x} : \tilde{\mathsf{t}} \vdash^{+}_{\mathrm{G}} P : \mathsf{T}|\mathsf{T}'$ and $\tilde{x}\#\mathsf{T}'$. By applying the induction hypothesis, we get that there are $R \equiv P$ and $\mathsf{S} \equiv \mathsf{T}|\mathsf{T}'$ such that $\Gamma, \tilde{x} : \tilde{\mathsf{t}} \vdash^{+}_{\mathrm{NG}} R : \mathsf{S}$. Hence, by applying (G-Eq) and (G-Inp) we get $\Gamma \vdash^{+}_{\mathrm{NG}} a(\tilde{x}).R : a((\tilde{x} : \tilde{\mathsf{t}})\mathsf{T}).\mathsf{T}'$, with $a(\tilde{x}).R \equiv a(\tilde{x}).P$.

(G-Out), (G-Tau), (G-Rep), (G-Sum), (G-Par$^+$). The proof proceeds by applying the induction hypothesis followed by the corresponding typing rule.

(G-Res). The proof proceeds by applying the induction hypothesis. It is enough to note that $\mathsf{T} \equiv \mathsf{S}$ implies $\mathsf{S}' \downarrow_{\tilde{a}} \equiv \mathsf{T}' \downarrow_{\tilde{a}}$, by definition. The latter and $\Phi \propto_{\tilde{a}} \mathsf{T}$ imply $\Phi \propto_{\tilde{a}} \mathsf{S}$ (by rule (struct) and closure of Φ with respect to structural congruence).

(G-Eq), (G-Eq-P). The proof proceeds by applying the induction hypothesis and relies on transitivity of $\equiv$.

The following two lemmas state some properties of normal derivations: both concern the structural correspondence of processes and types.

Lemma C.4. $\Gamma \vdash^{+}_{\mathrm{NG}} P : \mathsf{T}$ *with P prime implies that* T *is prime.*

Proof. The proof proceeds by inspection of the last typing rule applied in $\Gamma \vdash^{+}_{\mathrm{NG}} P : \mathsf{T}$.

Lemma C.5. $\Gamma \vdash^{+}_{\mathrm{NG}} (\nu\tilde{x}_1 : \tilde{\mathsf{t}}_1; \Phi_1)\cdots(\nu\tilde{x}_n : \tilde{\mathsf{t}}_n; \Phi_n)(P_1|\cdots|P_k) : \mathsf{T}$, *with $P_1, \cdots, P_k$ prime, implies*

- $\mathsf{T} = (\nu\tilde{x}_1 : \tilde{\mathsf{t}}_1)\cdots(\nu\tilde{x}_n : \tilde{\mathsf{t}}_n)(\mathsf{S}_1|\cdots|\mathsf{S}_k)$, *with* $\Gamma, \tilde{x}_1 : \tilde{\mathsf{t}}_1, \cdots, \tilde{x}_n : \tilde{\mathsf{t}}_n \vdash^{+}_{\mathrm{NG}} P_i : \mathsf{S}_i$, *for $i = 1, \cdots, k$,*
- *and* $\Phi_j \propto_{\tilde{x}_j} (\nu\tilde{x}_{j+1} : \tilde{\mathsf{t}}_{j+1})\cdots(\nu\tilde{x}_n : \tilde{\mathsf{t}}_n)(\mathsf{S}_1|\cdots|\mathsf{S}_k)$, *for $j = 1, \cdots, n$.*

Proof. If $n = 0$ take $\mathsf{S} = \mathsf{T}$. Suppose $n > 0$. By definition of normal derivation, the rule applied in the last n typing derivation must be (G-Res) preceded by an application of (G-Par). The result then follows by the premise of the rules and Lemma C.4.

Lemma 9 follows as a corollary of the result above.

Corollary C.1 (Lemma 9). $\Gamma \vdash^{+}_{\mathrm{NG}} (\tilde{\nu}\tilde{a} : \tilde{\mathsf{t}})(\nu\tilde{b} : \tilde{\mathsf{t}}'; \Phi)P : \mathsf{T}$ *implies* $\mathsf{T} = (\tilde{\nu}\tilde{a} : \tilde{\mathsf{t}})(\nu\tilde{b} : \tilde{\mathsf{t}}')\mathsf{S}$, *with* $\Gamma, \tilde{a} : \tilde{\mathsf{t}}, \tilde{b} : \tilde{\mathsf{t}}' \vdash^{+}_{\mathrm{NG}} P : \mathsf{S}$ *and* $\Phi \propto_{\tilde{b}} \mathsf{S}$.

As already seen for the local case, the weakening and contraction properties also hold for the global case. This result is used in the proof of Proposition 8 reported below.

Lemma C.6 (weakening and contraction). *Suppose* $\tilde{x}\#\,\mathrm{fn}(P), \mathrm{fn}(\mathsf{T})$ *and* Γ *is well-formed. Then* $\Gamma, \tilde{x} : \tilde{\mathsf{t}} \vdash^{+}_{\mathrm{G}} P : \mathsf{T}$ *if and only if* $\Gamma \vdash^{+}_{\mathrm{G}} P : \mathsf{T}$.

Proof. The proof is straightforward by induction on the derivation of $\Gamma, \tilde{x} : \tilde{\mathsf{t}} \vdash^{+}_{\mathrm{G}} P : \mathsf{T}$ and $\Gamma \vdash^{+}_{\mathrm{G}} P : \mathsf{T}$.

It is now possible to prove that the syntax directed system $\vdash^{+}_{\mathrm{G}}$ is equivalent to $\vdash_{\mathrm{G}}$.

Proposition C.2 (Proposition 8). $\Gamma \vdash_{\mathrm{G}} P : \mathsf{T}$ *if and only if* $\Gamma \vdash^{+}_{\mathrm{G}} P : \mathsf{T}$.

Proof.

($\Leftarrow$). Applications of rule (G-Par$^+$) to the parallel composition of n prime processes can be simulated by n applications of rule (G-Par).

($\Rightarrow$). The proof is by induction on the derivation of $\Gamma \vdash_G P : \mathsf{T}$ by distinguishing the last typing rule applied. The most interesting case is when (G-Par) is the last applied one. In this case $P = Q_1|Q_2$, $\mathsf{T} = \mathsf{T}_1|\mathsf{T}_2$ and $\Gamma \vdash_G Q_1|Q_2 : \mathsf{T}_1|\mathsf{T}_2$. By the premise of the rule, we get $\Gamma \vdash_G Q_1 : \mathsf{T}_1$ and $\Gamma \vdash_G Q_2 : \mathsf{T}_2$. Moreover, $\mathrm{cr}(\mathsf{T}_1)\#\mathsf{T}_2$ and $\mathrm{cr}(\mathsf{T}_2)\#\mathsf{T}_1$. By applying the induction hypothesis, we get $\Gamma \vdash^+_G Q_1 : \mathsf{T}_1$ and $\Gamma \vdash^+_G Q_2 : \mathsf{T}_2$.

By Proposition 7, there are n, k, m_1, m_2, and R_{i_j} (for $i = 1, 2$ and $j = i_1, \cdots, i_n$) such that

- $Q_1 \equiv (\nu\tilde{a}_1 : \tilde{\mathsf{t}}_1; \tilde{\Phi}_1) \cdots (\nu\tilde{a}_n : \tilde{\mathsf{t}}_n; \tilde{\Phi}_n)R_1$ with $R_1 = R_{1_1}|\cdots|R_{1_{m_1}}$
- $Q_2 \equiv (\nu\tilde{b}_1 : \tilde{\mathsf{t}'}_1; \tilde{\Psi}_1) \cdots (\nu\tilde{b}_k : \tilde{\mathsf{t}'}_k; \tilde{\Psi}_k)R_2$ with $R_2 = R_{2_1}|\cdots|R_{2_{m_2}}$

with

1. R_{i_j} prime for $i = 1, 2$ and $j = 1, \cdots, m_i$
2. $\tilde{a}_i \supseteq \mathrm{supp}(\Phi_i)$, for $i = 1, \cdots, n$, and $\tilde{b}_j \supseteq \mathrm{supp}(\Psi_j)$, for $j = 1, \cdots, k$ (by well-formedness of terms)
3. $\Gamma \vdash^+_{NG} (\nu\tilde{a}_1 : \tilde{\mathsf{t}}_1; \tilde{\Phi}_1) \cdots (\nu\tilde{a}_n : \tilde{\mathsf{t}}_n; \tilde{\Phi}_n)R_1 : \mathsf{S}_1 \equiv \mathsf{T}_1$
4. $\Gamma \vdash^+_{NG} (\nu\tilde{b}_1 : \tilde{\mathsf{t}'}_1; \tilde{\Psi}_1) \cdots (\nu\tilde{b}_k : \tilde{\mathsf{t}'}_k; \tilde{\Psi}_k)R_2 : \mathsf{S}_2 \equiv \mathsf{T}_2$.

By (3), (4) and Lemma C.5:

- $\mathsf{S}_1 = (\nu\tilde{a}_1 : \tilde{\mathsf{t}}_1) \cdots (\nu\tilde{a}_n : \tilde{\mathsf{t}}_n)\mathsf{U}$ with $\mathsf{U} = \mathsf{U}_1|\cdots|\mathsf{U}_{m_1}$, and $\Gamma, \tilde{a}_1 : \tilde{\mathsf{t}}_1, \cdots, \tilde{a}_n : \tilde{\mathsf{t}}_n \vdash^+_{NG} R_{1_i} : \mathsf{U}_i$, for $i = 1, \cdots, m_1$, and $\Phi_j \propto_{\tilde{a}_j} (\nu\tilde{a}_{j+1} : \tilde{\mathsf{t}}_{j+1}) \cdots (\nu\tilde{a}_n : \tilde{\mathsf{t}}_n)\mathsf{U}$, for $j = 1, \cdots, n$;
- $\mathsf{S}_2 = (\nu\tilde{b}_1 : \tilde{\mathsf{t}'}_1) \cdots (\nu\tilde{b}_k : \tilde{\mathsf{t}'}_k)\mathsf{V}$ with $\mathsf{V} = \mathsf{V}_1|\cdots|\mathsf{V}_{m_2}$, and $\Gamma, \tilde{b}_1 : \tilde{\mathsf{t}'}_1, \cdots, \tilde{b}_k : \tilde{\mathsf{t}'}_k \vdash^+_{NG} R_{2_i} : \mathsf{V}_i$, for $i = 1, \cdots, m_2$, and $\Psi_j \propto_{\tilde{b}_j} (\nu\tilde{b}_{j+1} : \tilde{\mathsf{t}'}_{j+1}) \cdots (\nu\tilde{b}_k : \tilde{\mathsf{t}'}_k)\mathsf{V}$, for $j = 1, \cdots, k$.

By $\mathrm{cr}(\mathsf{T}_1)\#\mathsf{T}_2$ and $\mathrm{cr}(\mathsf{T}_2)\#\mathsf{T}_1$ it follows that $\mathrm{cr}(\mathsf{S}_1)\#\mathsf{S}_2$, $\mathrm{cr}(\mathsf{S}_2)\#\mathsf{S}_1$, hence $\mathrm{cr}(\mathsf{U})\#\mathsf{V}$ and $\mathrm{cr}(\mathsf{V})\#\mathsf{U}$. Therefore, by Lemma C.6 and (G-Par$^+$) it follows that $\Gamma, \tilde{a}_1 : \tilde{\mathsf{t}}_1, \cdots, \tilde{a}_n : \tilde{\mathsf{t}}_n, \tilde{b}_1 : \tilde{\mathsf{t}'}_1, \cdots, \tilde{b}_k : \tilde{\mathsf{t}'}_k \vdash^+_{NG} R_1|R_2 : \mathsf{U}|\mathsf{V}$.

By definition of critical names, $\mathsf{U}[\tilde{a}_n] \subseteq \mathrm{cr}(\mathsf{S}_1) \cup \tilde{a}_n$ and by definition of free names $\mathrm{fn}(\mathsf{V}) \subseteq \mathrm{fn}(\mathsf{S}_2) \cup \tilde{b}_1 \cup \cdots \cup \tilde{b}_k$. Therefore, $\mathrm{cr}(\mathsf{U})\#\mathsf{V}$ and $\tilde{a}_n\#\mathsf{V}$ imply $\mathsf{U}[\tilde{a}_n]\#\mathsf{V}$. Moreover, $\tilde{a}_n \supseteq \mathrm{supp}(\Phi_n)$, hence by Lemma C.2 and $\Phi_n \propto_{\tilde{a}_n} \mathsf{U}$ it follows that $\Phi_n \propto_{\tilde{a}_n} (\mathsf{U}|\mathsf{V})$ and, by (G-Res), $\Gamma, \tilde{a}_1 : \tilde{\mathsf{t}}_1, \cdots, \tilde{a}_{n-1} : \tilde{\mathsf{t}}_{n-1}, \tilde{b}_1 : \tilde{\mathsf{t}'}_1, \cdots, \tilde{b}_k : \tilde{\mathsf{t}'}_k \vdash^+_{NG} (\nu\tilde{a}_n : \tilde{\mathsf{t}}_n; \Phi_n)(R_1|R_2) : (\nu\tilde{a}_n : \tilde{\mathsf{t}}_n)(\mathsf{U}|\mathsf{V})$.

A similar reasoning can be applied to the remaining $\tilde{a}_{n-1}, \cdots, \tilde{a}_1$ and $\tilde{b}_k, \cdots, \tilde{b}_1$ in order to obtain $\Gamma \vdash^+_{NG} (\nu\tilde{b}_1 : \tilde{\mathsf{t}'}_1; \tilde{\Psi}_1) \cdots (\nu\tilde{b}_k : \tilde{\mathsf{t}'}_k; \tilde{\Psi}_k)(\nu\tilde{a}_1 : \tilde{\mathsf{t}}_1; \tilde{\Phi}_1) \cdots (\nu\tilde{a}_n : \tilde{\mathsf{t}}_n; \tilde{\Phi}_n)(R_1|R_2) : (\nu\tilde{a}_1 : \tilde{\mathsf{t}}_1) \cdots (\nu\tilde{a}_n : \tilde{\mathsf{t}}_n)(\nu\tilde{b}_1 : \tilde{\mathsf{t}'}_1) \cdots (\nu\tilde{b}_k : \tilde{\mathsf{t}'}_k)(\mathsf{U}|\mathsf{V})$.

Finally, by (G-Eq) and (G-Eq-P), it follows that $\Gamma \vdash^+_G P : \mathsf{T}$.

As usual, the subject reduction property relies on the substitution lemma, that can be proved by means of a tedious, but not difficult, proof.

Lemma C.7 (substitution). *Suppose* Γ *is well-formed. If* $\Gamma, \tilde{x} : \tilde{\mathsf{t}} \vdash^+_G P : \mathsf{T}$ *and* $\Gamma \vdash^+_G \tilde{b} : \tilde{\mathsf{t}}$ *and* $\tilde{b}, \tilde{x}\#\mathrm{cr}(\mathsf{T})$ *then* $\Gamma[\tilde{b}/\tilde{x}] \vdash^+_G P[\tilde{b}/\tilde{x}] : \mathsf{T}[\tilde{b}/\tilde{x}]$.

Proof. The proof proceeds by induction on the derivation of $\Gamma, \tilde{x} : \tilde{\mathsf{t}} \vdash^+_G P : \mathsf{T}$.

(G-Eq), (G-Eq-P), (G-Inp). The proof relies on the induction hypothesis and on the fact that $\equiv$ is preserved by substitutions;

(G-Rep). The proof proceeds by applying the induction hypothesis and by noting that $\tilde{b},\tilde{x}\#\mathrm{cr}(\mathsf{T})$, $\mathrm{cr}(\mathsf{T})=\emptyset$ and Lemma C.1 (12) imply $\mathrm{cr}(\mathsf{T}[\tilde{b}/\tilde{x}])=\mathrm{cr}(\mathsf{T})=\emptyset$;

(G-Par). The proof proceeds by applying the induction hypothesis and by noting that $\tilde{b},\tilde{x}\#\mathrm{cr}(\mathsf{T})$, $\mathrm{cr}(\mathsf{T}_i)\#\mathsf{T}_j$, for $i,j=1,2$ with $i\neq j$, and Lemma C.1 (12) imply $\mathrm{cr}(\mathsf{T}_i[\tilde{b}/\tilde{x}])=\mathrm{cr}(\mathsf{T}_i)\#\mathsf{T}_j[\tilde{b}/\tilde{x}]$, for $i,j=1,2$ with $i\neq j$;

(G-Out). By $\Gamma,\tilde{x}:\tilde{\mathsf{t}}\vdash^{+}_{\mathrm{G}}\overline{a}\langle\tilde{c}\rangle.P:\overline{a}.(\mathsf{T}[\tilde{c}/\tilde{y}]\,|\,\mathsf{S})$ and the premise of the rule, we get:

- $\Gamma,\tilde{x}:\tilde{\mathsf{t}}\vdash^{+}_{\mathrm{G}} a:(\tilde{y}:\tilde{\mathsf{t}}')\mathsf{T}$, hence $\Gamma[\tilde{b}/\tilde{x}]\vdash^{+}_{\mathrm{G}} a[\tilde{b}/\tilde{x}]:(\tilde{y}:\tilde{\mathsf{t}}'[\tilde{b}/\tilde{x}])\mathsf{T}[\tilde{b}/\tilde{x}]$ (note that $\tilde{y}\#\tilde{x},\tilde{b}$ because names in $\tilde{y}$ are bound in T);
- $\Gamma,\tilde{x}:\tilde{\mathsf{t}}\vdash^{+}_{\mathrm{G}}\tilde{c}:\tilde{\mathsf{t}}'$, hence $\Gamma[\tilde{b}/\tilde{x}]\vdash^{+}_{\mathrm{G}}\tilde{c}[\tilde{b}/\tilde{x}]:\tilde{\mathsf{t}}'[\tilde{b}/\tilde{x}]$;
- $\Gamma,\tilde{x}:\tilde{\mathsf{t}}\vdash^{+}_{\mathrm{G}} P:\mathsf{S}$, hence by applying the induction hypothesis, $\Gamma[\tilde{b}/\tilde{x}]\vdash^{+}_{\mathrm{G}} P[\tilde{b}/\tilde{x}]:\mathsf{S}[\tilde{b}/\tilde{x}]$;
- $\tilde{c}\#\mathrm{cr}(\mathsf{T})$ moreover $\tilde{y}\#\mathrm{cr}(\mathsf{T})$ by well formedness of channel types, therefore by Lemma C.1 (12) $\mathrm{cr}(\mathsf{T})=\mathrm{cr}(\mathsf{T}[\tilde{c}/\tilde{y}])$. By $\tilde{b},\tilde{x}\#\mathrm{cr}(\overline{a}.(\mathsf{T}[\tilde{c}/\tilde{y}]\,|\,\mathsf{S}))$ and Lemma C.1 (3,1) we get $\tilde{b},\tilde{x}\#\mathrm{cr}(\mathsf{T}[\tilde{c}/\tilde{y}])=\mathrm{cr}(\mathsf{T})$. Hence, again by Lemma C.1 (12), $\mathrm{cr}(\mathsf{T})=\mathrm{cr}(\mathsf{T}[\tilde{b}/\tilde{x}])$. Therefore, $\tilde{c}\#\mathrm{cr}(\mathsf{T}[\tilde{b}/\tilde{x}])$ and $\tilde{c}[\tilde{b}/\tilde{x}]\#\mathrm{cr}(\mathsf{T}[\tilde{b}/\tilde{x}])$;
- $\mathrm{cr}(\mathsf{T}[\tilde{c}/\tilde{y}])\#\mathsf{S}$. As previously shown, $\mathrm{cr}(\mathsf{T}[\tilde{c}/\tilde{y}])=\mathrm{cr}(\mathsf{T})=\mathrm{cr}(\mathsf{T}[\tilde{b}/\tilde{x}])=\mathrm{cr}(\mathsf{T}[\tilde{c}/\tilde{y}][\tilde{b}/\tilde{x}])$. Given that $\tilde{y}\#\tilde{x},\tilde{b}$, it holds that $\mathsf{T}[\tilde{c}/\tilde{y}][\tilde{b}/\tilde{x}]=\mathsf{T}[\tilde{b}/\tilde{x}][\tilde{c}[\tilde{b}/\tilde{x}]/\tilde{y}]$ and $\mathrm{cr}(\mathsf{T}[\tilde{b}/\tilde{x}][\tilde{c}[\tilde{b}/\tilde{x}]/\tilde{y}])\#\mathsf{S}[\tilde{b}/\tilde{x}]$;
- $\mathsf{T}[\tilde{c}/\tilde{y}]\#\mathrm{cr}(\mathsf{S})$. By Lemma C.1 (3,1), $\mathrm{cr}(\mathsf{S}[\tilde{b}/\tilde{x}])=\mathrm{cr}(\mathsf{S})$ and, by $\tilde{y}\#\tilde{x},\tilde{b}$, $\mathrm{cr}(\mathsf{S}[\tilde{b}/\tilde{x}])\#\mathsf{T}[\tilde{c}/\tilde{y}][\tilde{b}/\tilde{x}]=\mathsf{T}[\tilde{b}/\tilde{x}][\tilde{c}[\tilde{b}/\tilde{x}]/\tilde{y}]$.

Therefore, by (G-Out), $\Gamma[\tilde{b}/\tilde{x}]\vdash^{+}_{\mathrm{G}}(\overline{a}\langle\tilde{c}\rangle.P)[\tilde{b}/\tilde{x}]:(\overline{a}.(\mathsf{T}[\tilde{c}/\tilde{y}]\,|\,\mathsf{S}))[\tilde{b}/\tilde{x}]=\overline{a}[\tilde{b}/\tilde{x}].(\mathsf{T}[\tilde{b}/\tilde{x}][\tilde{c}[\tilde{b}/\tilde{x}]/\tilde{y}]\,|\,\mathsf{S}[\tilde{b}/\tilde{x}])$.

(G-Res). By $\Gamma,\tilde{x}:\tilde{\mathsf{t}}\vdash^{+}_{\mathrm{G}}(\nu\tilde{a}:\tilde{\mathsf{t}}';\Phi)P:(\nu\tilde{a}:\tilde{\mathsf{t}}')\mathsf{T}$ and the premise of the rule, we get $\Gamma,\tilde{x}:\tilde{\mathsf{t}},\tilde{a}:\tilde{\mathsf{t}}'\vdash^{+}_{\mathrm{G}} P:\mathsf{T}$ and $\Phi\propto_{\tilde{a}}\mathsf{T}$. Moreover, $\tilde{a}\supseteq\mathrm{supp}(\Phi)$ by well-formedness of terms.

By applying the induction hypothesis to P, we get $(\Gamma,\tilde{a}:\tilde{\mathsf{t}}')[\tilde{b}/\tilde{x}]\vdash^{+}_{\mathrm{G}} P[\tilde{b}/\tilde{x}]:\mathsf{T}[\tilde{b}/\tilde{x}]$.

By $\tilde{b},\tilde{x}\#\mathrm{cr}((\nu\tilde{a}:\tilde{\mathsf{t}}')\mathsf{T})$ and $\tilde{b},\tilde{x}\#\tilde{a}$ it follows that $\tilde{b},\tilde{x}\#\mathsf{T}[\tilde{a}]$ and, by $\Phi\propto_{\tilde{a}}\mathsf{T}$, $\tilde{a}\supseteq\mathrm{supp}(\Phi)$ and Lemma C.3, we get $\Phi\propto_{\tilde{a}}\mathsf{T}[\tilde{b}/\tilde{x}]$.

Finally, by (G-Res), $\Gamma[\tilde{b}/\tilde{x}]\vdash^{+}_{\mathrm{G}}(\nu\tilde{a}:\tilde{\mathsf{t}}'[\tilde{b}/\tilde{x}];\Phi)P[\tilde{b}/\tilde{x}]:(\nu\tilde{a}:\tilde{\mathsf{t}}'[\tilde{b}/\tilde{x}])\mathsf{T}[\tilde{b}/\tilde{x}]$.

The following lemma is the equivalent of Lemma B.1 for the local system.

Lemma C.8. *$\Gamma\vdash_{\mathrm{G}} P:\mathsf{T}$ implies $\mathrm{fn}(P)\subseteq\mathrm{fn}(\mathsf{T})$. $\Gamma,\tilde{x}:\tilde{\mathsf{t}}\vdash_{\mathrm{G}} P:\mathsf{T}$, Γ well-formed and $\tilde{x}\#P$ imply $\tilde{x}\#\mathsf{T}$.*

Finally, the lengthy proof of the subject reduction property relies mainly on Proposition 8 and Proposition 7, which guarantee a syntax directed normal typing derivation, and on Lemma C.5, which guarantees that processes and types share the same shallow spatial structure.

Theorem C.1 (subject reduction, Theorem 5). *$\Gamma\vdash_{\mathrm{G}} P:\mathsf{T}$ and $P\xrightarrow{\lambda}P'$ implies that there exists a T' such that $\mathsf{T}\xrightarrow{\lambda}\mathsf{T}'$ and $\Gamma\vdash_{\mathrm{G}} P':\mathsf{T}'$.*

Proof. By Proposition 8, $\Gamma \vdash_{\mathrm{G}} P : \mathsf{T}$ implies $\Gamma \vdash^{+}_{\mathrm{G}} P : \mathsf{T}$, and by Proposition 7 there are $Q \equiv P$ and $\mathsf{S} \equiv \mathsf{T}$, with Q in head normal form, such that $\Gamma \vdash^{+}_{\mathrm{NG}} Q : \mathsf{S}$. Given that $Q \equiv P$ and $P \xrightarrow{\lambda} P'$ we get, by (struct), $Q \xrightarrow{\lambda} Q' \equiv P'$. In the following we prove that Q' is well-typed with an associated type S' such that $\mathsf{S} \xrightarrow{\lambda} \mathsf{S}'$. This in order to deduce well typedness of P' by applying rule (G-Eq-P); T' will be chosen equal to S'.

Suppose for simplicity that there is only one top level block of restricted names, thus

$$Q = (\nu\tilde{d} : \tilde{\mathsf{t}}; \Phi)(R_1|\cdots|R_k) \triangleq (\nu\tilde{d} : \tilde{\mathsf{t}}; \Phi)R$$

with R_i prime process for each $i = 1, \cdots, k$. The generalization to $n \geq 0$ top level restrictions is obvious. By $\Gamma \vdash^{+}_{\mathrm{NG}} (\nu\tilde{d} : \tilde{\mathsf{t}}; \Phi)(R_1|\cdots|R_k) : \mathsf{S}$ and Lemma C.5 we get

$$\mathsf{S} = (\nu\tilde{d} : \tilde{\mathsf{t}})(\mathsf{U}_1|\cdots|\mathsf{U}_k) \triangleq (\nu\tilde{d} : \tilde{\mathsf{t}})\mathsf{U} \tag{3}$$

with $\Gamma, \tilde{d} : \tilde{\mathsf{t}} \vdash^{+}_{\mathrm{NG}} R_i : \mathsf{U}_i$, for each $i = 1, \cdots, k$, and $\Phi \propto_{\tilde{d}} \mathsf{U}$. Moreover, by the proof of Lemma C.5, we get $\mathrm{cr}(\mathsf{U}_i)\#\mathsf{U}_j$ for $i \neq j$. We distingush two cases.

- Suppose $\lambda = \langle\epsilon\rangle$ and

$$R = R_1|\cdots|\underbrace{\tau.R'_l + \sum_i \alpha_i.Q_i}_{R_l}|\cdots|R_k \xrightarrow{\lambda} R_1|\cdots|R'_l|\cdots|R_k \triangleq R' .$$

  By $\Gamma, \tilde{d} : \tilde{\mathsf{t}} \vdash^{+}_{\mathrm{NG}} R_l : \mathsf{U}_l$ and the premise of (G-Tau) and (G-Sum) we have $\mathsf{U}_l = \tau.\mathsf{U}'_l + \sum_i \mu_i.\mathsf{W}_i$ and $\Gamma, \tilde{d} : \tilde{\mathsf{t}} \vdash^{+}_{\mathrm{NG}} R'_l : \mathsf{U}'_l$. Therefore, $\mathsf{U}_l = \tau.\mathsf{U}'_l + \sum_i \mu_i.\mathsf{W}_i \xrightarrow{\lambda} \mathsf{U}'_l$. Then

$$\mathsf{T} \equiv (\nu\tilde{d} : \tilde{\mathsf{t}})(\mathsf{U}_1|\cdots|\tau.\mathsf{U}'_l + \sum_i \mu_i.\mathsf{W}_i|\cdots|\mathsf{U}_k) \xrightarrow{\lambda} (\nu\tilde{d} : \tilde{\mathsf{t}})(\mathsf{U}_1|\cdots|\mathsf{U}'_l|\cdots|\mathsf{U}_k) \triangleq (\nu\tilde{d} : \tilde{\mathsf{t}})\mathsf{U}' \triangleq \mathsf{T}' .$$

  Hence, by (struct), $\mathsf{T} \xrightarrow{\lambda} \mathsf{T}'$.

  Now, we show that $\Gamma, \tilde{d} : \tilde{\mathsf{t}} \vdash_{\mathrm{G}} R' : \mathsf{U}'$. By Proposition 8, $\Gamma, \tilde{d} : \tilde{\mathsf{t}} \vdash^{+}_{\mathrm{NG}} R'_l : \mathsf{U}'_l$ implies $\Gamma, \tilde{d} : \tilde{\mathsf{t}} \vdash_{\mathrm{G}} R'_l : \mathsf{U}'_l$. By definition, $\mathrm{cr}(\mathsf{U}_l) \supseteq \mathrm{cr}(\mathsf{U}'_l)$ (Lemma C.1 (6)) hence we can repeatedly apply (G-Par) and deduce $\Gamma, \tilde{d} : \tilde{\mathsf{t}} \vdash_{\mathrm{G}} R_1|\cdots|R'_l|\cdots|R_k : \mathsf{U}_1|\cdots|\mathsf{U}'_l|\cdots|\mathsf{U}_k$. Moreover, $\mathsf{U} \xrightarrow{\langle\epsilon\rangle} \mathsf{U}'$ implies $\mathsf{U} \succ^{\langle\epsilon\rangle}_{\mathsf{U}\downarrow_{\tilde{d}}} \mathsf{U}'$ and, by Definition 9, $\Phi_{\langle\epsilon\rangle} = \Phi \propto_{\tilde{d}} \mathsf{U}'$.

  Hence, by (G-Res), $\Gamma \vdash_{\mathrm{G}} (\nu\tilde{d} : \tilde{\mathsf{t}}; \Phi_\lambda)R' : (\nu\tilde{d} : \tilde{\mathsf{t}})\mathsf{U}'$. Finally, by (G-Eq-P) and $\Gamma \vdash_{\mathrm{G}} (\nu\tilde{d} : \tilde{\mathsf{t}}; \Phi_\lambda)R' : (\nu\tilde{d} : \tilde{\mathsf{t}})\mathsf{U}'$, it follows that $\Gamma \vdash_{\mathrm{G}} P' : \mathsf{T}'$.

- Suppose $\lambda ::= \langle a\rangle \mid \langle\epsilon\rangle$ and there is a synchronization

$$R = R_1|\cdots|\underbrace{a(\tilde{y}).R'_l + \sum_{i\in I} \alpha_i.Q_i}_{R_l}|\cdots|\underbrace{\overline{a}\langle\tilde{b}\rangle.R'_m + \sum_{j\in J} \alpha'_j.Q'_j}_{R_m}|\cdots|R_k \xrightarrow{\lambda} R_1|\cdots|R'_l[^{\tilde{b}}/_{\tilde{y}}]|\cdots|R'_m|\cdots|R_k \triangleq R'$$

  and, for the sake of simplicity, take $|I| = |J| = 0$.

  The last rules applied in the derivations of $\Gamma, \tilde{d} : \tilde{\mathsf{t}} \vdash^{+}_{\mathrm{NG}} a(\tilde{y}).R'_l : \mathsf{U}_l$ and $\Gamma, \tilde{d} : \tilde{\mathsf{t}} \vdash^{+}_{\mathrm{NG}} \overline{a}\langle\tilde{b}\rangle.R'_m : \mathsf{U}_m$ are, respectively, (G-Inp) and (G-Out). By their premise, we deduce the following:

1. $\Gamma, \tilde{d} : \tilde{\mathsf{t}} \vdash^{+}_{\mathrm{G}} a : (\tilde{y} : \tilde{\mathsf{t}}')\mathsf{W}$, $\tilde{y}\#\mathrm{cr}(\mathsf{W})$ by well-formedness of channel types (note also that $\tilde{y}\#\,\mathrm{fn}(\Gamma, \tilde{d} : \tilde{\mathsf{t}})$);
2. $\Gamma, \tilde{d} : \tilde{\mathsf{t}}, \tilde{y} : \tilde{\mathsf{t}}' \vdash^{+}_{\mathrm{G}} R'_l : \mathsf{W}|\mathsf{U}'_l$, with $\tilde{y}\#\mathsf{U}'_l$;
3. $\mathsf{U}_l = a(\mathsf{t}'').\mathsf{U}'_l$;
4. $\Gamma, \tilde{d} : \tilde{\mathsf{t}} \vdash^{+}_{\mathrm{G}} \tilde{b} : \tilde{\mathsf{t}}'$;
5. $\tilde{b}\#\mathrm{cr}(\mathsf{W})$;
6. $\Gamma, \tilde{d} : \tilde{\mathsf{t}} \vdash^{+}_{\mathrm{G}} R'_m : \mathsf{V}$;
7. $\mathsf{U}_m = \overline{a}.(\mathsf{W}[\tilde{b}/\tilde{y}]|\mathsf{V})$;
8. $\mathrm{cr}(\mathsf{W}[\tilde{b}/\tilde{y}])\#\mathsf{V}$ and $\mathrm{cr}(\mathsf{V})\#\mathsf{W}[\tilde{b}/\tilde{y}]$.

Now, by $\mathsf{S} \equiv \mathsf{T}$ and (3), we have

$$\begin{aligned}
\mathsf{T} \;&\equiv\; (\nu\tilde{d} : \tilde{\mathsf{t}})\mathsf{U} = (\nu\tilde{d} : \tilde{\mathsf{t}})(\mathsf{U}_1|\cdots|\underbrace{a(\mathsf{t}'').\mathsf{U}'_l}_{\mathsf{U}_l}|\cdots|\underbrace{\overline{a}.(\mathsf{V}|\mathsf{W}[\tilde{b}/\tilde{y}])}_{\mathsf{U}_m}|\cdots|\mathsf{U}_k) \\
&\xrightarrow{\langle a\rangle} (\nu\tilde{d} : \tilde{\mathsf{t}})(\mathsf{U}_1|\cdots|\mathsf{U}'_l|\cdots|\mathsf{V}|\mathsf{W}[\tilde{b}/\tilde{y}]|\cdots|\mathsf{U}_k) \\
&\triangleq\; (\nu\tilde{d} : \tilde{\mathsf{t}})\mathsf{U}' \triangleq \mathsf{T}' .
\end{aligned}$$

Hence, by (STRUCT), $\mathsf{T} \xrightarrow{\lambda} \mathsf{T}'$.

Now, we first show that $\Gamma; \tilde{d} : \tilde{\mathsf{t}} \vdash_{\mathrm{G}} R' : \mathsf{U}'$. We do so by using system $\vdash_{\mathrm{G}}$ and rule (G-PAR). We show that the premise of the rule are fullfilled and more precisely that $R'_l[\tilde{b}/\tilde{y}]$ is well typed.

Consider W and split $\tilde{y}$ into two parts: $\tilde{y} = \tilde{y}_1 \cup \tilde{y}_2$ such that $\tilde{y}_1 \subseteq \mathrm{fn}(\mathsf{W})$ and $\tilde{y}_2\#\mathsf{W}$. As a consequence, $\tilde{b} = \tilde{b}_1 \cup \tilde{b}_2$ such that $\tilde{b}_1 \subseteq \mathrm{fn}(\mathsf{W}[\tilde{b}/\tilde{y}])$ and $\tilde{b}_2\#\mathsf{W}[\tilde{b}/\tilde{y}]$. Hence, $\mathsf{W}[\tilde{b}/\tilde{y}] = \mathsf{W}[\tilde{b}_1/\tilde{y}_1]$. Moreover, by $\mathrm{cr}(\mathsf{U}_l)\#\mathsf{U}_m$ and $\tilde{b}_1 \subseteq \mathrm{fn}(\mathsf{U}_m)$ we get $\tilde{b}_1\#\mathrm{cr}(\mathsf{U}'_l)$. By the latter, and points (1), (2) and (5) it follows that

$$\tilde{b}_1, \tilde{y}_1\#\mathrm{cr}(\mathsf{W}|\mathsf{U}'_l) . \tag{4}$$

By Lemma C.8 and $\tilde{y}_2\#(\mathsf{W}|\mathsf{U}'_l)$, we get $\tilde{y}_2\#R'_l$, hence $R'_l[\tilde{b}/\tilde{y}] = R'_l[\tilde{b}_1/\tilde{y}_1]$.

By point (1), and $\tilde{y}\#\,\mathrm{fn}(\Gamma, \tilde{d} : \tilde{\mathsf{t}})$, we get $(\Gamma, \tilde{d} : \tilde{\mathsf{t}})[\tilde{b}/\tilde{y}] = \Gamma, \tilde{d} : \tilde{\mathsf{t}}$. Moreover, by equation (2), point (4), and Lemma C.7 (substitution) we get

$$\Gamma, \tilde{d} : \tilde{\mathsf{t}} \vdash^{+}_{\mathrm{G}} R'_l[\tilde{b}_1/\tilde{y}_1] : (\mathsf{W}|\mathsf{U}'_l)[\tilde{b}_1/\tilde{y}_1] = \mathsf{U}'_l|\mathsf{W}[\tilde{b}_1/\tilde{y}_1]$$

that is

$$\Gamma, \tilde{d} : \tilde{\mathsf{t}} \vdash^{+}_{\mathrm{G}} R'_l[\tilde{b}/\tilde{y}] : \mathsf{W}[\tilde{b}/\tilde{y}]|\mathsf{U}'_l .$$

By (8), $\mathrm{cr}(\mathsf{U}_i)\#\mathsf{U}_j$ for $i \neq j$, and Lemma C.1 (2,7) we get $\mathrm{cr}(\mathsf{U}'_l) \cup \mathrm{cr}(\mathsf{W}[\tilde{b}/\tilde{y}]) = \mathrm{cr}(\mathsf{W}[\tilde{b}/\tilde{y}]|\mathsf{U}'_l)\#\mathsf{V}$ and vice-versa $\mathrm{cr}(\mathsf{V})\#\mathsf{U}'_l|\mathsf{W}[\tilde{b}/\tilde{y}]$. Therefore, we can apply repeatedly rule (G-PAR) and obtain

$$\Gamma, \tilde{d} : \tilde{\mathsf{t}} \vdash_{\mathrm{G}} R' \;=\; R_1|\cdots|R'_l|\cdots|R'_m|\cdots|R_k \;:\; \mathsf{U}_1|\cdots|\mathsf{U}'_l|\mathsf{W}[\tilde{b}/\tilde{y}]|\cdots|\mathsf{V}|\cdots|\mathsf{U}_k \;=\; \mathsf{U}' .$$

We have now to put the restriction on $\tilde{d}$ on top of R'. We distinguish two cases.

1. Suppose $\lambda = \langle a \rangle$ and $a \notin \mathsf{U}[\tilde{d}]$. Then $\mathsf{U} \xrightarrow{a} \mathsf{U}'' \xrightarrow{\overline{a}} \mathsf{U}'$. with $\mathsf{U}'' = \mathsf{U}_1 | \cdots | \mathsf{U}'_l | \cdots | \overline{a}.(\mathsf{V} | \mathsf{W}[\tilde{b}/\tilde{y}]) | \cdots | \mathsf{U}_k$. Hence, by definition, $\mathsf{U} \succ^{a}_{\mathsf{U}[\tilde{d}]} \mathsf{U}'' \succ^{\overline{a}}_{\mathsf{U}[\tilde{d}]} \mathsf{U}'$. By Definition 9 and $a \downarrow_{\mathsf{U}[\tilde{d}]} = \langle \epsilon \rangle$, this means that $\Phi_{\langle \epsilon \rangle} \propto_{\tilde{d}} \mathsf{U}''$ and $\Phi_{\langle \epsilon \rangle \langle \epsilon \rangle} \propto_{\tilde{d}} \mathsf{U}'$. Therefore, by $\mathrm{Ok}(\Phi)$ we get $\Phi = \Phi_{\langle \epsilon \rangle} = \Phi_{\langle \epsilon \rangle \langle \epsilon \rangle} = \Phi_{\langle a \rangle}$ and $\Phi_\lambda \propto_{\tilde{d}} \mathsf{U}'$.
2. Suppose $\lambda ::= \langle a \rangle \,\big|\, \langle \epsilon \rangle$ with $a \in \mathsf{U}[\tilde{d}]$ ($\lambda \downarrow_{\mathsf{U}[\tilde{d}]} = \lambda$). In this case, by definition of $\succ^{\lambda}_{\tilde{x}}$ it follows that $\mathsf{U} \succ^{\lambda}_{\mathsf{U}[\tilde{d}]} \mathsf{U}'$. Moreover, by $\Phi \propto_{\tilde{d}} \mathsf{U}$ and Definition 9 we get $\Phi_\lambda \propto_{\tilde{d}} \mathsf{U}'$.

In both cases we can apply (G-Res) and deduce $\Gamma \vdash_{\mathrm{G}} (\nu \tilde{d} : \tilde{\mathsf{t}}; \Phi_\lambda) R' : (\nu \tilde{d} : \tilde{\mathsf{t}}) \mathsf{U}'$ and finally, by rule (G-Eq-P) we get $\Gamma \vdash_{\mathrm{G}} P' : \mathsf{T}'$.

D. Proofs of Section 8

Proving Lemmas 10 and 11 requires some preliminary results on properties of "$\models$" with respect to compositionality and hiding. Lemma D.1 guarantees that restricted names can be opened without compromising satisfiability. Lemma D.2 and Lemma D.3 guarantee that non-interfering parallel threads can be cut away, even if they appear under some top-level restrictions.

Lemma D.1. *Consider $\phi \in \mathcal{F}_{\tilde{x}}$ not containing "$\neg$". Suppose $\mathsf{T} \models \phi$ and $\mathsf{T} \equiv (\tilde{\nu d})(\mathsf{T}_1 | \cdots | \mathsf{T}_n)$ with T_i prime for each i. Then $(\tilde{\nu b})(\mathsf{T}_1 | \cdots | \mathsf{T}_n) \models \phi$ for each $\tilde{b} \subseteq \tilde{d}$.*

Proof. The proof proceeds by induction on the structure of the formula ϕ. Note that $(\tilde{\nu d})(\mathsf{T}_1 | \cdots | \mathsf{T}_n) \models \phi$ follows by definition of $\models$ and $\mathsf{T} \models \phi$.

$\phi ::= \mathbf{T} \,\big|\, a \,\big|\, \overline{a}$. Obvious.

$\phi = \phi_1 \vee \phi_2$. The proof proceeds by applying the induction hypothesis.

$\phi = \mathrm{H}^*\psi$. $(\tilde{\nu d})(\mathsf{T}_1 | \cdots | \mathsf{T}_n) \models \mathrm{H}^*\psi$ means that $(\tilde{\nu d})(\mathsf{T}_1 | \cdots | \mathsf{T}_n) \equiv (\tilde{\nu d}')(\tilde{\nu d}'')(\mathsf{T}_1 | \cdots | \mathsf{T}_n)$ with $\tilde{d}' \cup \tilde{d}'' = \tilde{d}$ and $(\tilde{\nu d}'')(\mathsf{T}_1 | \cdots | \mathsf{T}_n) \models \psi$.

Given that $\tilde{b} \subseteq \tilde{d}$ we get $\tilde{b} = \tilde{b}' \cup \tilde{b}''$, for some $\tilde{b}' \subseteq \tilde{d}'$ and $\tilde{b}'' \subseteq \tilde{d}''$, and $(\tilde{\nu b})(\mathsf{T}_1 | \cdots | \mathsf{T}_n) \equiv (\tilde{\nu b}')(\tilde{\nu b}'')(\mathsf{T}_1 | \cdots | \mathsf{T}_n)$. Hence, by applying the induction hypothesis, we get $(\tilde{\nu b}'')(\mathsf{T}_1 | \cdots | \mathsf{T}_n) \models \psi$ and, by definition, $(\tilde{\nu b}')(\tilde{\nu b}'')(\mathsf{T}_1 | \cdots | \mathsf{T}_n) \models \phi$, that is $(\tilde{\nu b})(\mathsf{T}_1 | \cdots | \mathsf{T}_n) \models \phi$.

$\phi = \phi_1 | \phi_2$. $(\tilde{\nu d})(\mathsf{T}_1 | \cdots | \mathsf{T}_n) \models \phi_1 | \phi_2$ implies

$$(\tilde{\nu d})(\mathsf{T}_1 | \cdots | \mathsf{T}_n) \equiv (\tilde{\nu d}_1)(\mathsf{T}_{1,1} | \cdots | \mathsf{T}_{1,m_1}) \,|\, (\tilde{\nu d}_2)(\mathsf{T}_{2,1} | \cdots | \mathsf{T}_{2,m_2})$$

with $(\tilde{\nu d}_i)(\mathsf{T}_{i,1} | \cdots | \mathsf{T}_{i,m_i}) \models \phi_i$ for $i = 1, 2$.

Given that $\tilde{b} \subseteq \tilde{d}$ it follows that

$$(\tilde{\nu b})(\mathsf{T}_1 | \cdots | \mathsf{T}_n) \equiv (\tilde{\nu b}_1)(\mathsf{T}_{1,1} | \cdots | \mathsf{T}_{1,m_1}) \,|\, (\tilde{\nu b}_2)(\mathsf{T}_{2,1} | \cdots | \mathsf{T}_{2,m_2})$$

with $\tilde{b}_i \subseteq \tilde{d}_i$ for $i = 1, 2$. The result follows by applying the induction hypothesis to both $(\tilde{\nu b}_i)(\mathsf{T}_{i,1} | \cdots | \mathsf{T}_{i,m_i})$.

$\phi = \langle a \rangle \psi$. $(\tilde{\nu d})(\mathsf{T}_1 | \cdots | \mathsf{T}_n) \models \phi$ implies $(\tilde{\nu d})(\mathsf{T}_1 | \cdots | \mathsf{T}_n) \xrightarrow{\langle a \rangle} \mathsf{U}$ and $\mathsf{U} \models \psi$. Obviously, for $\tilde{b} \subseteq \tilde{d}$ we get $(\tilde{\nu b})(\mathsf{T}_1 | \cdots | \mathsf{T}_n) \xrightarrow{\langle a \rangle} \mathsf{U}'$. Moreover, if $\mathsf{U} \equiv (\tilde{\nu d})(\tilde{\nu a})(\mathsf{T}'_1 | \cdots | \mathsf{T}'_n)$, with T'_i prime, then $\mathsf{U}' \equiv (\tilde{\nu b})(\tilde{\nu a})(\mathsf{T}'_1 | \cdots | \mathsf{T}'_n)$, with $\tilde{b} \cup \tilde{a} \subseteq \tilde{d} \cup \tilde{a}$. Therefore the induction hypothesis can be applied to infer $\mathsf{U}' \models \psi$. That is $(\tilde{\nu b})(\mathsf{T}_1 | \cdots | \mathsf{T}_n) \models \phi$.

$\phi = \langle\tilde{a}\rangle^*\psi$**,** $\phi = \langle-\tilde{a}\rangle^*\psi$**.** In both cases the proof proceeds similarly.

Lemma D.2. *Suppose* $\phi \in \mathcal{F}_{\tilde{x}}$ *and* ϕ *does not contain* $\neg$. *Consider* T,S *and* $\tilde{d}$ *such that* $\mathsf{S}[\tilde{x}]\#\mathsf{T}$. $(\tilde{\nu}\tilde{d})(\mathsf{T}|\mathsf{S}) \xrightarrow{s} \mathsf{U} \models \phi$ *implies* $(\tilde{\nu}\tilde{d})\mathsf{S} \xrightarrow{s'} \mathsf{U}' \models \phi$*, with* $|s'| \leq |s|$ *and* $\mathrm{n}(s') \subseteq \mathrm{n}(s)$*.*

Proof. The proof proceeds by induction on $|s|$.

$|s| = 0$**.** $(\tilde{\nu}\tilde{d})\mathsf{S} \models \phi$ is deduced by Lemma D.3.

$|s| = n+1$**.** $s = \lambda \cdot s'$ and $(\tilde{\nu}\tilde{d})(\mathsf{T}|\mathsf{S}) \xrightarrow{\lambda} (\tilde{\nu}\tilde{d})\mathsf{V} \xrightarrow{s'} (\tilde{\nu}\tilde{d})\mathsf{U} \models \phi$, with $|s'| < |s|$ and obviously and $\mathrm{n}(s') \subseteq \mathrm{n}(s)$. The proof proceeds by distinguishing the following cases depending on the reduction $\xrightarrow{\lambda}$:

1. suppose $\mathsf{S} \xrightarrow{\lambda'} \mathsf{S}'$, with $\lambda' \uparrow_{\tilde{d}} = \lambda$ and $\mathsf{V} \equiv \mathsf{T}|\mathsf{S}'$. By Lemma C.1 (9), $\mathsf{T}\#\mathsf{S}[\tilde{x}] \supseteq \mathsf{S}'[\tilde{x}]$. Therefore, by applying the internal induction hypothesis to $(\tilde{\nu}\tilde{d})(\mathsf{T}|\mathsf{S}') \xrightarrow{s'} (\tilde{\nu}\tilde{d})\mathsf{U} \models \phi$, we get $(\tilde{\nu}\tilde{d})\mathsf{S}' \xrightarrow{s''} (\tilde{\nu}\tilde{d})\mathsf{U}' \models \phi$, with $|s''| \leq |s'|$ and $\mathrm{n}(s'') \subseteq \mathrm{n}(s')$. Hence $(\tilde{\nu}\tilde{d})\mathsf{S} \xrightarrow{\lambda} \xrightarrow{s''} (\tilde{\nu}\tilde{d})\mathsf{U}' \models \phi$, with $|\lambda \cdot s''| \leq |s|$ and $\mathrm{n}(\lambda \cdot s'') \subseteq \mathrm{n}(s)$.
2. Suppose $\mathsf{T} \xrightarrow{\lambda'} \mathsf{T}'$, with $\lambda' \uparrow_{\tilde{d}} = \lambda$ and $\mathsf{V} \equiv \mathsf{T}'|\mathsf{S}$. Again, $\mathsf{S}[\tilde{x}]\#\mathsf{T}'$. Therefore, by applying the internal induction hypothesis to $(\tilde{\nu}\tilde{d})(\mathsf{T}'|\mathsf{S}) \xrightarrow{s'} (\tilde{\nu}\tilde{d})\mathsf{U} \models \psi$, we get $(\tilde{\nu}\tilde{d})\mathsf{S} \xrightarrow{s''} (\tilde{\nu}\tilde{d})\mathsf{U}' \models \phi$, with $|s''| \leq |s'| < |s|$ and $\mathrm{n}(s'') \subseteq \mathrm{n}(s)$.
3. Suppose S and T interact with each other

$$\begin{aligned}
\mathsf{T} &\equiv (\tilde{\nu}\tilde{f})(\textstyle\sum_i \mu_i.\mathsf{W}_i + a(\mathrm{t}).\mathsf{W}'|\mathsf{W}'')\\
\mathsf{S} &\equiv (\tilde{\nu}\tilde{g})(\textstyle\sum_j \mu'_j.\mathsf{V}_j + \overline{a}.\mathsf{V}'|\mathsf{V}'')\\
\mathsf{V} &\equiv (\tilde{\nu}\tilde{f})(\mathsf{W}'|\mathsf{W}'')\,|\,(\tilde{\nu}\tilde{g})(\mathsf{V}'|\mathsf{V}'')\ .
\end{aligned}$$

for some $a \notin \mathsf{S}[\tilde{x}]$ (the proof proceeds similarly in case T contains the output, or in case the input prefix on a is replicated). Given that $\mathsf{S}[\tilde{x}] = \big((\tilde{\nu}\tilde{g})(\sum_j \mu'_j.\mathsf{V}_j + \overline{a}.\mathsf{V}'|\mathsf{V}'')\big)[\tilde{x}]$ and $a \notin \mathsf{S}[\tilde{x}]$ we get $\mathsf{S}[\tilde{x}]\#\overline{a}.\mathsf{V}'$ (Lemma C.1 (11)).
Now,

$$(\tilde{\nu}\tilde{d})\mathsf{V} \equiv (\tilde{\nu}\tilde{d})\big((\tilde{\nu}\tilde{g})((\tilde{\nu}\tilde{f})(\mathsf{W}'|\mathsf{W}'')\,|\,\mathsf{V}'\,|\,\mathsf{V}'')\big) \equiv (\tilde{\nu}\tilde{d}_0)(\mathsf{T}_0\,|\,\mathsf{V}'')$$

with $\tilde{d}_0 = \tilde{d}\tilde{g}$ and $\mathsf{T}_0 = (\tilde{\nu}\tilde{f})(\mathsf{W}'|\mathsf{W}''|\mathsf{V}')$. Moreover,

$$(\tilde{\nu}\tilde{d}_0)(\mathsf{T}_0\,|\,\mathsf{V}'') \xrightarrow{s'} (\tilde{\nu}\tilde{d})\mathsf{U} \models \psi$$

and $\mathsf{T}_0\#\mathsf{V}''[\tilde{x}] \subseteq \mathsf{S}[\tilde{x}] \cup \tilde{g}$.
Since $|s'| < |s|$ and $\mathrm{n}(s') \subseteq \mathrm{n}(s)$, by applying the internal induction hypothesis, we get $(\tilde{\nu}\tilde{d}_0)\mathsf{V}'' \xrightarrow{s''} (\tilde{\nu}\tilde{d})\mathsf{U}' \models \phi$, with $|s''| \leq |s'| < |s|$ and $\mathrm{n}(s'') \subseteq \mathrm{n}(s)$. Finally, by Lemma A.2 (1), $(\tilde{\nu}\tilde{d})(\tilde{\nu}\tilde{g})(\sum_j \mu'_j.\mathsf{V}_j + \overline{a}.\mathsf{V}'|\mathsf{U}') \models \phi$, hence $(\tilde{\nu}\tilde{d})\mathsf{S} = (\tilde{\nu}\tilde{d})((\tilde{\nu}\tilde{g})(\sum_j \mu'_j.\mathsf{V}_j + \overline{a}.\mathsf{V}'|\mathsf{V}'')) \xrightarrow{s''} (\tilde{\nu}\tilde{d})(\tilde{\nu}\tilde{g})(\sum_j \mu'_j.\mathsf{V}_j + \overline{a}.\mathsf{V}'|\mathsf{U}') \models \phi$, with $|s''| \leq |s'| < |s|$ and $\mathrm{n}(s'') \subseteq \mathrm{n}(s)$.

Lemma D.3. *Suppose* $\phi \in \mathcal{F}_{\tilde{x}}$ *and not containing "*$\neg$*". Suppose* $(\tilde{\nu}\tilde{d})(\mathsf{T}|\mathsf{S}) \models \phi$ *where* $\mathsf{S}[\tilde{x}]\#\mathsf{T}$*. Then* $(\tilde{\nu}\tilde{d})\mathsf{S} \models \phi$*.*

Proof. The proof proceeds by induction on the structure of the formula ϕ.

$\phi ::= \mathbf{T} \mid a \mid \overline{a}$ with $a \in \tilde{x}$. Obvious.

$\phi = \phi_1 \vee \phi_2$. The proof proceeds by applying the induction hypothesis.

$\phi = \mathrm{H}^*\psi$. By applying Lemma D.1 we first remove all $\tilde{d}$ and all top level restrictions in T and S. Therefore, if $\mathsf{T} \equiv (\tilde{\nu}\tilde{a})(\mathsf{T}_1|\cdots|\mathsf{T}_n)$ and $\mathsf{S} \equiv (\tilde{\nu}\tilde{b})(\mathsf{S}_1|\cdots|\mathsf{S}_m)$, with T_i and S_j prime for each i, j, then we get $\mathsf{T}_1|\cdots|\mathsf{T}_n|\mathsf{S}_1|\cdots|\mathsf{S}_m \models \mathrm{H}^*\psi$. Given that all T_i and S_j are prime, there are no top-level restrictions to collect, hence $\mathsf{T}_1|\cdots|\mathsf{T}_n|\mathsf{S}_1|\cdots|\mathsf{S}_m \models \psi$. Then the proof proceeds by applying the induction hypothesis in order to prove that $\mathsf{S}_1|\cdots|\mathsf{S}_m \models \psi$. Finally, by definition of $\models$, $(\tilde{\nu}\tilde{d})(\tilde{\nu}\tilde{b})(\mathsf{S}_1|\cdots|\mathsf{S}_m) \equiv (\tilde{\nu}\tilde{d})\mathsf{S} \models \mathrm{H}^*\psi$.

$\phi = \phi_1|\phi_2$. $(\tilde{\nu}\tilde{d})(\mathsf{T}|\mathsf{S}) \models \phi_1|\phi_2$ implies $(\tilde{\nu}\tilde{d})(\mathsf{T}|\mathsf{S}) \equiv \mathsf{U}|\mathsf{V}$ with $\mathsf{U} \equiv (\tilde{\nu}\tilde{d}_1)(\mathsf{T}_1|\mathsf{S}_1) \models \phi_1$ and $\mathsf{V} \equiv (\tilde{\nu}\tilde{d}_2)(\mathsf{T}_2|\mathsf{S}_2) \models \phi_2$, for some T_i and S_i such that $\mathsf{T} \equiv \mathsf{T}_1|\mathsf{T}_2$ and $\mathsf{S} \equiv \mathsf{S}_1|\mathsf{S}_2$ and $\tilde{d}_1\tilde{d}_2$ a permutation of $\tilde{d}$.

By $\tilde{x}\#\mathsf{T}$ we get $\tilde{x}\#\mathsf{T}_1, \mathsf{T}_2$. Similarly, given that $\mathsf{S}[\tilde{x}] \supseteq \mathsf{S}_1[\tilde{x}] \cup \mathsf{S}_2[\tilde{x}]$ (Lemma C.1), by $\mathsf{S}[\tilde{x}]\#\mathsf{T}$, we get $\mathsf{S}_1[\tilde{x}]\#\mathsf{T}_1$ and $\mathsf{S}_2[\tilde{x}]\#\mathsf{T}_2$. Hence, by applying the induction hypothesis, we get $(\tilde{\nu}\tilde{d}_1)\mathsf{S}_1 \models \phi_1$ and $(\tilde{\nu}\tilde{d}_2)\mathsf{S}_2 \models \phi_2$. Hence, $(\tilde{\nu}\tilde{d})\mathsf{S} \models \phi$.

$\phi = \langle a\rangle\psi$. $(\tilde{\nu}\tilde{d})(\mathsf{T}|\mathsf{S}) \models \phi$ implies $(\tilde{\nu}\tilde{d})(\mathsf{T}|\mathsf{S}) \xrightarrow{\langle a\rangle} \mathsf{U}$ and $\mathsf{U} \models \psi$. Given that $a \in \tilde{x}\#\mathsf{T}$, the reduction is originated by S; therefore $\mathsf{U} = (\tilde{\nu}\tilde{d})(\mathsf{T}|\mathsf{S}')$ where $\mathsf{S} \xrightarrow{\langle a\rangle} \mathsf{S}'$. Given that $\mathsf{T}\#\mathsf{S}[\tilde{x}] \supseteq \mathsf{S}'[\tilde{x}]$ (Lemma C.1 (9)), by applying the induction hypothesis to $(\tilde{\nu}\tilde{d})(\mathsf{T}|\mathsf{S}') \models \psi$, we get $(\tilde{\nu}\tilde{d})\mathsf{S}' \models \psi$ and $(\tilde{\nu}\tilde{d})\mathsf{S} \models \phi$.

$\phi = \langle\tilde{a}\rangle^*\psi$. The proof proceeds similarly.

$\phi = \langle-\tilde{a}\rangle^*\psi$. $(\tilde{\nu}\tilde{d})(\mathsf{T}|\mathsf{S}) \models \langle-\tilde{a}\rangle^*\psi$ implies $(\tilde{\nu}\tilde{d})(\mathsf{T}|\mathsf{S}) \xrightarrow{s} \mathsf{U}$ with $\mathsf{U} \models \psi$ and $\tilde{a}\#s$. By Lemma D.2, $(\tilde{\nu}\tilde{d})\mathsf{S} \xrightarrow{s'} \mathsf{S}'$ with $\mathrm{n}(s') \subseteq \mathrm{n}(s)$ and $\mathsf{S}' \models \psi$. Therefore $\tilde{a}\#s'$ and $(\tilde{\nu}\tilde{d})\mathsf{S} \models \langle-\tilde{a}\rangle^*\psi$ by definition.

Lemmas 10 and 11 follow as corollaries of the result below. It guarantees that non-critical names can be masked without compromising satisfiability.

Lemma D.4. *Suppose $\phi \in \mathcal{F}_{\tilde{x}}$ with negation not occurring underneath any $\langle-\tilde{y}\rangle^*$ in ϕ. Let $\tilde{w} \supseteq \mathsf{T}[\tilde{x}]$. $\mathsf{T}\downarrow_{\tilde{w}} \models \phi$ if and only if $\mathsf{T} \models \phi$.*

Proof. The proof proceeds by induction on the structure of the formula ϕ. The most interesting cases are dealt with below.

$\phi = \mathrm{H}^*\psi$.

($\Rightarrow$). $\mathsf{T}\downarrow_{\tilde{w}} \models \mathrm{H}^*\psi$ implies $\mathsf{T}\downarrow_{\tilde{w}} \equiv (\tilde{\nu}\tilde{a})\mathsf{V}$ and $\mathsf{V} \models \psi$. By Lemma 6 (4), we get $\mathsf{T} \equiv (\tilde{\nu}\tilde{a})\mathsf{S}$, for some S such that $\mathsf{S}\downarrow_{\tilde{w},\tilde{a}} = \mathsf{V}$. Moreover, by definition of $\mathsf{T}[\cdot]$ we get $\tilde{w} \cup \tilde{a} \supseteq \mathsf{S}[\tilde{x}]$. Hence, by applying the induction hypothesis to $\mathsf{S}\downarrow_{\tilde{w},\tilde{a}}$, we get $\mathsf{S} \models \psi$ and $\mathsf{T} \models \phi$.

($\Leftarrow$). $\mathsf{T} \models \mathrm{H}^*\psi$ implies $\mathsf{T} \equiv (\tilde{\nu}\tilde{a})\mathsf{S}$ and $\mathsf{S} \models \psi$. By definition of $\downarrow_{\tilde{w}}$, we get $\mathsf{T}\downarrow_{\tilde{w}} \equiv (\tilde{\nu}\tilde{a})\mathsf{S}\downarrow_{\tilde{w},\tilde{a}}$ and by applying the induction hypothesis, $\mathsf{S} \models \psi$ implies $\mathsf{S}\downarrow_{\tilde{w}'} \models \psi$, for each $\tilde{w}' \supseteq \mathsf{S}[\tilde{x}]$. In particular this holds for $\tilde{w}' = \tilde{w} \cup \tilde{a} \supseteq \mathsf{T}[\tilde{x}] \cup \tilde{a}$. Therefore, $\mathsf{T}\downarrow_{\tilde{w}} \models \mathrm{H}^*\psi$.

$\phi = \phi_1|\phi_2$.

- **($\Rightarrow$).** $\mathsf{T}\downarrow_{\tilde{w}} \models \phi_1|\phi_2$ implies $\mathsf{T}\downarrow_{\tilde{w}} \equiv \mathsf{T}_1\downarrow_{\tilde{w}} |\mathsf{T}_2\downarrow_{\tilde{w}}$ with $\mathsf{T}_i\downarrow_{\tilde{w}} \models \phi_i$ for $i = 1,2$. $\tilde{w} \supseteq \mathsf{T}[\tilde{x}] \supseteq \mathsf{T}_1[\tilde{x}] \cup \mathsf{T}_2[\tilde{x}]$. Hence, $\tilde{w} \supseteq \mathsf{T}_i[\tilde{x}]$. By applying the induction hypothesis, we get $\mathsf{T}_i \models \phi_i$. Moreover, by Lemma 6 (2), we get $\mathsf{T} \equiv \mathsf{T}_1|\mathsf{T}_2$. Therefore, $\mathsf{T} \models \phi_1|\phi_2$.
- **($\Leftarrow$).** $\mathsf{T} \models \phi_1|\phi_2$ implies $\mathsf{T} \equiv \mathsf{T}_1|\mathsf{T}_2$ with $\mathsf{T}_i \models \phi_i$ for $i = 1,2$. By applying the induction hypothesis, we get $\mathsf{T}_i\downarrow_{\tilde{w}'} \models \phi_i$, for each $\tilde{w}' \supseteq \mathsf{T}_i[\tilde{x}]$. Therefore, given that $\mathsf{T}[\tilde{x}] \supseteq \mathsf{T}_i[\tilde{x}]$, this holds for $\tilde{w} \supseteq \mathsf{T}[\tilde{x}]$. Moreover, by Lemma 6 (3), we get $\mathsf{T}\downarrow_{\tilde{w}} \equiv \mathsf{T}_1\downarrow_{\tilde{w}} |\mathsf{T}_2\downarrow_{\tilde{w}}$. Therefore, $\mathsf{T}\downarrow_{\tilde{w}} \models \phi_1|\phi_2$.

$\phi = \langle -\tilde{a}\rangle^* \psi$.

- **($\Rightarrow$).** $\mathsf{T}\downarrow_{\tilde{w}} \models \phi$ implies $\mathsf{T}\downarrow_{\tilde{w}} \xrightarrow{s} \mathsf{S}$ and $\mathsf{S} \models \psi$, with $\tilde{a}\#s$.
  The proof proceeds by induction on $l = |s|$.
  - **$l = 0$.** In this case $\mathsf{T}\downarrow_{\tilde{w}} \models \psi$ and by applying the external induction hypothesis, we get $\mathsf{T} \models \psi$, therefore $\mathsf{T} \models \phi$
  - **$l = n+1$.** $s = \lambda \cdot s'$, $\tilde{a}\#\lambda$ and $\mathsf{T}\downarrow_{\tilde{w}} \xrightarrow{\lambda} \mathsf{U} \xrightarrow{s'} \mathsf{S}$, with $\mathsf{S} \models \psi$, $|s'| = n$ and $\mathsf{U} \models \phi$. We distinguish two cases, depending on the reduction $\mathsf{T}\downarrow_{\tilde{w}} \xrightarrow{\lambda} \mathsf{U}$.

    1. $\lambda = \langle a\rangle$ or $\lambda = \langle \epsilon\rangle$ with the ϵ-reduction originated by a synchronization on a bound name or a τ prefix in T. By Proposition 5, we get $\mathsf{T} \xrightarrow{\lambda} \mathsf{V}$, for some V such that $\mathsf{V}\downarrow_{\tilde{w}} = \mathsf{U}$. Given that $\tilde{w} \supseteq \mathsf{T}[\tilde{x}] \supseteq \mathsf{V}[\tilde{x}]$ (Lemma C.1 (9)) by applying the internal induction hypothesis ($|s'| = n < l$) we get $\mathsf{V} \models \phi$, therefore $\mathsf{T} \models \phi$.
    2. $\lambda = \langle \epsilon\rangle$ with the ϵ-reduction originated by a prefix in T with subject a not in $\tilde{w}$. In this case

$$\mathsf{T}\downarrow_{\tilde{w}} \equiv ((\tilde{v}\tilde{d})((\sum_{i\in I}\mu_i.\mathsf{S}_i + \mu.\mathsf{W})|\mathsf{W}'))\downarrow_{\tilde{w}} = (\tilde{v}\tilde{d})((\sum_{i\in I}(\mu_i.\mathsf{S}_i)\downarrow_{\tilde{w},\tilde{d}} + \tau.(\mathsf{W}\downarrow_{\tilde{w},\tilde{d}}))|\mathsf{W}'\downarrow_{\tilde{w},\tilde{d}}) \tag{5}$$

with $\mu ::= a(\mathrm{t}) \,\big|\, \overline{a}$ for some $a \notin \tilde{w} \cup \tilde{d}$ (similar comments if $I = \emptyset$ and $\mu = !a(\mathrm{t})$) and

$$\mathsf{U} \equiv (\tilde{v}\tilde{d})(\mathsf{W}\downarrow_{\tilde{w},\tilde{d}} |\mathsf{W}'\downarrow_{\tilde{w},\tilde{d}}) = ((\tilde{v}\tilde{d})(\mathsf{W}|\mathsf{W}'))\downarrow_{\tilde{w}} \xrightarrow{s'} \mathsf{S} \models \psi\ .$$

Therefore, $((\tilde{v}\tilde{d})(\mathsf{W}|\mathsf{W}'))\downarrow_{\tilde{w}} \models \phi$.
Given that $a \notin \tilde{w} \supseteq \mathsf{T}[\tilde{x}]$, by Lemma C.1 (11) we deduce that $\mathsf{T}[\tilde{x}]\#\mathsf{W}$. Therefore, $\mathsf{T}\downarrow_{\tilde{w}} [\tilde{x}]\#\mathsf{W}\downarrow_{\tilde{w}}$, and by (5) $\mathsf{W}'\downarrow_{\tilde{w}} [\tilde{x}]\#\mathsf{W}\downarrow_{\tilde{w}}$. Given that ψ does not contain negations, by Lemma D.3 we get $((\tilde{v}\tilde{d})\mathsf{W}')\downarrow_{\tilde{w}} \models \phi$. Moreover, by Lemma D.2, $((\tilde{v}\tilde{d})\mathsf{W}')\downarrow_{\tilde{w}} \xrightarrow{s''} \mathsf{S}' \models \psi$, with $|s''| \leq |s'| < |s|$. By applying the internal induction hypothesis, we get $(\tilde{v}\tilde{d})\mathsf{W}' \models \phi$. Finally, by Lemma A.2 (1), we deduce $\mathsf{T} \equiv (\tilde{v}\tilde{d})((\sum_{i\in I}\mu_i.\mathsf{S}_i + \mu.\mathsf{W})|\mathsf{W}') \models \phi$.

- **($\Leftarrow$).** $\mathsf{T} \models \langle -\tilde{a}\rangle^*\psi$ implies $\mathsf{T} \xrightarrow{s} \mathsf{S}$, with $\tilde{a}\#s$ and $\mathsf{S} \models \psi$. By Proposition 5, $\mathsf{T}\downarrow_{\tilde{w}} \xrightarrow{s'} \mathsf{S}\downarrow_{\tilde{w}}$, with $\mathrm{fn}(s') \subseteq \mathrm{fn}(s)$. Moreover, $\tilde{w} \supseteq \mathsf{T}[\tilde{x}] \supseteq \mathsf{S}[\tilde{x}]$ (Lemma C.1 (9)). Hence, by applying the induction hypothesis, we get $\mathsf{S}\downarrow_{\tilde{w}} \models \psi$; therefore $\mathsf{T}\downarrow_{\tilde{w}} \models \phi$.

Corollary D.1 (Lemma 10). *Let $\phi \in \mathcal{F}_{\tilde{x}}$ be of the form $\phi = \Box^*_{-\tilde{a}}\psi$ with negation not occurring underneath any $\langle -\tilde{a}\rangle^*$ in ψ. Then for any* T, $\mathsf{T} \Downarrow_{\tilde{x}} \models \phi$ *implies* $\mathsf{T} \models \phi$.

Proof. Let $\tilde{w} = \mathsf{T}[\tilde{x}]$. $\mathsf{T} \downarrow_{\tilde{w}} \models \phi$ means that for each S such that $\mathsf{T} \downarrow_{\tilde{w}} \xrightarrow{s} \mathsf{S}$, with $\tilde{a}\#s$, it holds that $\mathsf{S} \models \psi$.

Take any U such that $\mathsf{T} \xrightarrow{s} \mathsf{U}$. By Proposition 5, $\mathsf{T} \downarrow_{\tilde{w}} \xrightarrow{s'} \mathsf{U} \downarrow_{\tilde{w}}$, with $\mathrm{fn}(s') \subseteq \mathrm{fn}(s)$. Therefore, $\mathsf{U} \downarrow_{\tilde{w}} \models \psi$. Moreover, $\tilde{w} = \mathsf{T}[\tilde{x}] \supseteq \mathsf{U}[\tilde{x}]$ (Lemma C.1 (9)).

Negation does not occur underneath any $\langle -\tilde{a} \rangle^*$ in ψ, therefore by Lemma D.4 we get $\mathsf{U} \models \psi$. This holds for each U such that $\mathsf{T} \xrightarrow{s} \mathsf{U}$, with $\tilde{a}\#s$; therefore $\mathsf{T} \models \phi$.

Corollary D.2 (Lemma 11). *Suppose $\phi \in \mathcal{F}_{\tilde{x}}$ with negation not occurring underneath any $\langle -\tilde{y} \rangle^*$ in ϕ. $\mathsf{T} \downarrow_{\tilde{w}} \models \phi$ and $\tilde{w} \supseteq \mathsf{T}[\tilde{x}]$ imply $\mathsf{T} \downarrow_{\mathsf{T}[\tilde{x}]} \models \phi$.*

Proof. By $\mathsf{T} \downarrow_{\tilde{w}} \models \phi$ and Lemma D.4, we get $\mathsf{T} \models \phi$ and again by Lemma D.4, $\mathsf{T} \downarrow_{\mathsf{T}[\tilde{x}]} \models \phi$.