Responsiveness in process calculi*

Lucia Acciai and Michele Boreale
Dipartimento di Sistemi e Informatica
Università di Firenze
Email: {lacciai, boreale}@dsi.unifi.it

Abstract

A system guarantees responsive usage of a channel r if a communication along r is guaranteed to eventually take place. Responsiveness is important, for instance, to ensure that any request to a service be eventually replied. We propose two distinct type systems, each of which statically guarantees responsive usage of names in well-typed pi-calculus processes. In the first system, we achieve responsiveness by combining techniques for deadlock and livelock avoidance with *linearity* and *receptiveness*. The latter is a guarantee that a name is ready to receive as soon as it is created. These conditions imply relevant limitations on the nesting of actions and on multiple use of names in processes. In the second system, we relax these requirements so as to permit certain forms of nested inputs and multiple outputs. We demonstrate the expressive power of the two systems by showing that primitive recursive functions – in the case of the first system – and Cook and Misra's service orchestration language ORC – in the case of the second system – can be encoded into well-typed processes.

Keywords: pi-calculus, type systems, responsiveness, receptiveness, linearity.

1 Introduction

A system guarantees responsive usage of a channel name r if a communication along r is guaranteed to eventually take place. That is, under a suitable assumption of fairness, all computations from the initial state contain at least one reduction with r as subject. We christen this property *responsiveness* as we are particularly interested in the case where r is a return channel passed to a service or function. As an example, a network of processes S may contain a service $!a(x,r).P$ invocable in RPC style: the caller sends at a an argument x and a return channel r. S's responsive usage of r implies that every request at a will be eventually replied. This may be a critical property in domains of applications such as service-oriented computing.

Our goal is to devise typing disciplines that statically guarantee responsiveness of significant classes of processes. In the past decade, several type systems for the pi-calculus have been proposed to analyze properties that share some similarities with responsiveness, such as linearity [10], uniform receptiveness [13], lock freedom [6, 7] and termination [5]; they will be examined throughout the paper. However

*Research partially supported by the EU within the FET-GC2 initiative, project SENSORIA. Corresponding author: Michele Boreale, Università di Firenze, Dipartimento di Sistemi e Informatica, Viale Morgagni 65, I-50134 Firenze. E-mail: boreale@dsi.unifi.it.

none of the above mentioned properties alone is sufficient, or even necessary, to ensure the property we are after, as we discuss below (further discussion is found in the concluding section).

The first system we propose builds around Sangiorgi's system for uniform receptiveness [13]. However, we discard uniformity and introduce additional constraints, as explained below. As expected, most difficulties in achieving responsiveness originate from responsive names being passed around. If an intended receiver of a responsive name r, say $a(x).P$, is not available "on time", r might never be delivered, hence used. In this respect, receptiveness is useful, because it can be used to ensure that inputs on a and on r are available as soon as these names are created.

Even when delivery of r is ensured, however, one should take care that r will be processed properly. Indeed, the recipient might just "forget" about r, like in $(\nu a, r)(a(x).\mathbf{0} \,|\, \overline{a}\langle r\rangle)$; or r might be passed from one recipient to another, its use as a subject being delayed indefinitely, like in

$$(\nu a, b, r)\big(\,!a(x).\overline{b}\langle x\rangle \mid !b(y).\overline{a}\langle y\rangle \mid \overline{a}\langle r\rangle\,\big)\,. \tag{1}$$

The first situation can be avoided by imposing that in the receiver $a(x).P$, name x occurs at least once in the body P. In fact, as we shall discuss in the paper, it is necessary that any responsive name be used *linearly*, that is, it appears exactly once in input and once in output. Indefinite delays like (1) can be avoided by using a stratification of names into *levels*, like in the type system for termination of Deng and Sangiorgi [5]. We will rule out divergent computations that involve responsive names infinitely often, but we'll do allow divergence in general.

Finally, even when a responsive name is eventually in place as a subject of an output action, one has to make sure that such action becomes eventually available. In other words, one must avoid cyclic waiting like in

$$r(x).\overline{s}\langle x\rangle \mid s(y).\overline{r}\langle y\rangle\,. \tag{2}$$

This will be achieved by building a graph of the dependencies among responsive names and then checking for its acyclicity.

In the first system, receptiveness and linearity impose relevant limitations on the syntax of well-typed processes: nested free inputs are forbidden, as well as multiple outputs on the same responsive name. On the other hand, the type system is expressive enough to enable a RPC programming style; in particular, we show that the usual CPS encoding of primitive recursive functions gives rise to well-typed processes.

In the second system we propose, the constraints on receptiveness and linearity are relaxed so as to allow certain forms of nested inputs and multiple outputs. Relaxation of linearity and receptiveness raises new issues, though. As an example, responsiveness might fail due to "shortage" of inputs or outputs, like in the following example, where a reduction on r is followed by one on s, while a communication on t cannot occur (r, s and t responsive):

$$\overline{r}\langle s\rangle|\overline{r}\langle t\rangle|r(x).\overline{x}|s|t \xrightarrow{\tau}\xrightarrow{\tau} \overline{r}\langle t\rangle|t\,.$$

These issues must be dealt with by carefully "balancing" inputs and outputs in typing contexts and in processes. The resulting system is flexible enough to let all orchestration patterns of Cook and Misra's ORC language [4] be encoded into well-typed processes. Due to a rather crude use of levels, however, only certain forms of (tail-)recursion are encodable. In fact, neither the first system is subsumed by the second one, nor vice versa.

The rest of the paper is organized as follows. Syntax and operational semantics of the calculus are presented in Section 2, and responsiveness is formally defined. Section 3 introduces the first type system, after an informal discussion on the requirements for responsiveness. The main results, subject reduction and type soundness, are presented in Section 4; there we also give a bound, depending on the

size of a process, on the number of reductions necessary before a given responsive name is used. Simple extensions of the first system (summation, `if-then-else` and recursion on data values) are presented in Section 5, where the encoding of primitive recursive functions is also discussed. The second system and its properties are discussed in Section 6 and 7. Several examples illustrating the extent and limits of the system are discussed in Section 8. Encoding of ORC into pi-calculus is discussed in Section 9. The concluding Section 10 contains some indications for further work and a detailed discussion of related work. For the sake of readability, the most technical proofs have been confined to separate Appendices (A-H).

2 Syntax and operational semantics

In this section we describe the syntax (processes and types) and the operational semantics of the calculus. On top of the operational semantics, we define the responsiveness property we are after.

2.1 Syntax

We focus on an asynchronous variant of the pi-calculus [14] without nondeterministic summations. Asynchrony is a natural assumption in a distributed environment and simplifies the technical treatment. Nondeterministic summation and `if-then-else` can be accommodated, but they would raise some notational burden in the proofs that we prefer to avoid at this stage: we defer its treatment to a later stage (see Subsection 5.1).

A countable set of names $\mathcal{N}$, ranged over by $a, b, \ldots, x, y, \ldots$, is presupposed. The set $\mathcal{P}$ of *processes* $P, Q, \ldots$ is defined as the set of terms generated by the following grammar.

$P ::=$	$\mathbf{0}$		*Inaction*
	$\mid \overline{a}\langle b\rangle$		*Output*
	$\mid a(x).P$	$x \notin \mathrm{in}(P)$	*Input prefix*
	$\mid\ !a(x).P$	$x \notin \mathrm{in}(P)$	*Replication*
	$\mid P\mid Q$		*Parallel composition*
	$\mid (\nu b)P$		*Restriction*

In a non blocking output action $\overline{a}\langle b\rangle$, name a is said to occur in *output subject position* and b in *output object position*. In an input prefix $a(x).P$, and in a replicated input prefix $!a(x).P$, name a is said to occur in *input subject position* and x in *input object position*. We denote by $\mathrm{in}(P)$ the set of names occurring free in input subject position in P. The condition $x \notin \mathrm{in}(P)$, for input and replicated input, means that names can be passed around with the output capability only. This assumption simplifies reasoning on types and does not significantly affect the expressiveness of the language (see e.g. [3, 11]). As usual, parallel composition, $P|Q$, represents the concurrent execution of P and Q and restriction, $(\nu b)P$, creates a fresh name b with initial scope P. Notions of free and bound names ($\mathrm{fn}(\cdot)$ and $\mathrm{bn}(\cdot)$), and alpha-equivalence ($=_\alpha$) arise as expected. We assume the set of names $\mathcal{N}$ is partitioned into a family of countable *sorts* $\mathcal{S}, \mathcal{S}', \ldots$. A fixed sorting à la Milner [12] is presupposed: that is, any sort $\mathcal{S}$ has an associated object sort $\mathcal{S}'$, and a name of sort $\mathcal{S}$ can only carry names of sort $\mathcal{S}'$. Alpha-equivalence is assumed to be sort-respecting: this means a bound name can be alpha-renamed only to a fresh name in the same sort. We only consider processes that are well-sorted in this system.

Notationally, we shall often abbreviate $a(x).\mathbf{0}$ as $a(x)$, and $(\nu a_1)\ldots(\nu a_n)P$ as $(\nu a_1,\ldots,a_n)P$ or $(\nu\tilde{a})P$, where $\tilde{a} = a_1,\ldots,a_n$. In a few examples, the object part of an action may be omitted if not relevant for the discussion; e.g., $a(x).P$ may be shortened into $a.P$.

2.2 Sorts and types

We assume a surjective mapping from the set of sorts to the set of types $\mathcal{T}$ defined below. We write $a : \mathsf{T}$ if a belongs to a sort $\mathcal{S}$ with associated type T.

A channel type $\mathsf{T}^{[\mathsf{u},k]}$ conveys three pieces of information: a type of carried objects T, a *usage* u, that can be *responsive* (ρ) or ω-*receptive* (ω), and an integer *level* $k \geq 0$. If $a : \mathsf{T}^{[\mathsf{u},k]}$ and $\mathsf{u} = \rho$ (resp. $\mathsf{u} = \omega$) we say that a is *responsive* (resp. ω-*receptive*). Informally, responsive names are guaranteed to be eventually used as subject in a communication, while ω-receptive names are guaranteed to be constantly ready to receive. Levels are used to bound the number of times a responsive name can be passed around, so to avoid infinite delay in their use as subject. We also consider a type I of *inert* names that cannot be used as subject of a communication – they just serve as tokens to be passed around. Finally, a type $\bot$ is introduced to collect those names that cannot be used at all: as we discuss below, $\bot$ is useful to formulate the subject reduction property while keeping the standard operational semantics.

Definition 1 (types). *The set $\mathcal{T}$ of* types *contains the constant $\bot$ and the set of terms generated by the grammar below. We use $\mathsf{T},\mathsf{S},\ldots$ to range over $\mathcal{T}$.*

$$\mathsf{T} ::= \quad \mathsf{I} \quad \big| \quad \mathsf{T}^{\mathsf{U}} \qquad \mathsf{U} ::= [\rho,k] \quad \big| \quad [\omega,k] \qquad (k \geq 0)$$

Note that even if $\bot \in \mathcal{T}$, the grammar above rules out compound types containing $\bot$, e.g. $\bot^{[\rho,k]}$ is not a type. For the sake of simplicity, recursive types are omitted. In particular we do not allow channels to carry names belonging to their own sort.

2.3 Operational semantics

The semantics of processes is given by a labelled transition system in the early style, whose rules are presented in Table 1. An *action* μ can be of the following forms: free output, $\overline{a}\langle b\rangle$, bound output, $\overline{a}(b)$, input $a(b)$, or internal move $\tau\langle a,b\rangle$. We define $\mathrm{n}(a(b)) = \mathrm{n}(\overline{a}\langle b\rangle) = \mathrm{n}(\overline{a}(b)) = \mathrm{n}(\tau\langle a,b\rangle) = \{a,b\}$. A substitution σ is a finite partial map from names to names; for any term P, we write $P\sigma$ for the result of applying σ to P, with the usual renaming conventions to avoid captures.

The rules are standard, but in τ-transitions $\xrightarrow{\tau\langle a,b\rangle}$ we keep track of the – *free or bound* – names a and b that are used as subject and object, respectively, of a communication. This extra information will be useful on several occasions. In rule (RES-ρ), a bound responsive subject a is renamed to a fresh name c of type $\bot$ – a sort of "casting" of a to type $\bot$. Informally, this casting is necessary because in a well-typed process, due to the linearity constraint on responsive names, name a must vanish after being used as subject. Rule (RES) deals with the remaining cases of restriction. Note that if type and sorting information are ignored, one gets back the standard operational semantics of pi-calculus.

Convention In the paper, *processes are identified modulo alpha-equivalence*. Formally, this means that we work with alpha-equivalence classes of terms, rather than with individual terms. A few caveats apply to this convention. For each alpha-equivalence class $[P]$, we choose a representative term in a canonical form, written $\mathrm{can}(P)$, having all bound names pairwise distinct and disjoint from the set of free names. All (syntax-directed) functions or relations taking $[P]$ as an argument are defined in terms

$$\text{(IN)}\ a(x).P \xrightarrow{a(b)} P[b/x] \qquad \text{(REP)}\ !a(x).P \xrightarrow{a(b)} !a(x).P|P[b/x]$$

$$\text{(OUT)}\ \overline{a}\langle b\rangle \xrightarrow{\overline{a}\langle b\rangle} \mathbf{0} \qquad \text{(PAR}_1\text{)}\ \frac{P \xrightarrow{\mu} P' \quad \mathrm{bn}(\mu)\cap \mathrm{fn}(Q)=\emptyset}{P|Q \xrightarrow{\mu} P'|Q}$$

$$\text{(OPEN)}\ \frac{P \xrightarrow{\overline{a}\langle b\rangle} P' \quad a\neq b}{(\nu b)P \xrightarrow{\overline{a}(b)} P'} \qquad \text{(CLOSE}_1\text{)}\ \frac{P \xrightarrow{\overline{a}(b)} P' \quad Q \xrightarrow{a(b)} Q' \quad b\notin \mathrm{fn}(Q)}{P|Q \xrightarrow{\tau\langle a,b\rangle} (\nu b)(P'|Q')}$$

$$\text{(COM}_1\text{)}\ \frac{P \xrightarrow{\overline{a}\langle b\rangle} P' \quad Q \xrightarrow{a(b)} Q'}{P|Q \xrightarrow{\tau\langle a,b\rangle} P'|Q'} \qquad \text{(RES-}\rho\text{)}\ \frac{P \xrightarrow{\tau\langle a,b\rangle} P' \quad a \text{ responsive} \quad c:\perp \quad c \text{ fresh}}{(\nu a)P \xrightarrow{\tau\langle a,b\rangle} (\nu c)P'[c/a]}$$

$$\text{(RES)}\ \frac{P \xrightarrow{\mu} P' \quad \text{if } a\in \mathrm{n}(\mu) \text{ then } \exists b\neq a: \begin{cases} \text{either} & \mu=\tau\langle b,a\rangle \\ \text{or} & \mu=\tau\langle a,b\rangle \text{ and } a \text{ not responsive}\end{cases}}{(\nu a)P \xrightarrow{\mu} (\nu a)P'}$$

Symmetric rules not shown.

Table 1: Rules for the labeled transition system

of $\mathrm{can}(P)$. In particular: (a) $\mathrm{bn}([P])$ is formally defined as the set of bound names of the term $\mathrm{can}(P)$, that is $\mathrm{bn}(\mathrm{can}(P))$; (b) the operational semantics given in Table 1, which does not mention a rule for alpha-equivalence, is formally defined over plain terms, not over equivalence classes. The semantics of equivalence classes is then given in terms of the semantics of their representatives. Formally, this is done via the rule below, which we assume implicitly:

$$\frac{\mathrm{can}(P) \xrightarrow{\mu} Q}{[P] \xrightarrow{\mu} [Q]}\ .$$

For the sake of simplicity, in the rest of the paper we shall write $[P]$ simply as P, and omit making explicit reference to $\mathrm{can}(P)$ when no ambiguity arises. So, for example, we write $\mathrm{bn}(P)$ instead of $\mathrm{bn}([P])$. Given any collection of terms, reductions etc. we shall assume that all bound names occurring in this collection are pairwise distinct and disjoint from free names.

Notation We shall often refer to a transition $P \xrightarrow{\tau\langle a,b\rangle} P'$, sometimes abbreviated as $P \xrightarrow{\tau} P'$, as a *reduction*. $P \xrightarrow{[a]} P'$ means $P \xrightarrow{\tau\langle a,b\rangle} P'$ for some free or bound name b. For a string $s = a_1 \cdots a_n \in \mathcal{N}^*$, $P \xrightarrow{[s]} P'$ means $P \xrightarrow{[a_1]} \cdots \xrightarrow{[a_n]} P'$, while $P \stackrel{[c]}{\Longrightarrow} P'$ means $P \xrightarrow{\tau}{}^* \xrightarrow{[c]} \xrightarrow{\tau}{}^* P'$. We use such abbreviations as $P \stackrel{[c]}{\Longrightarrow}$ to mean that there exists P' such that $P \stackrel{[c]}{\Longrightarrow} P'$.

We can now introduce the responsiveness property we are after. To motivate the definition below, think of a fair computation as a sequence of communications where for no name a a transition $\xrightarrow{[a]}$ is weakly (i.e. up to some reductions) enabled infinitely often without ever taking place. Assuming a fair scheduling of transitions in this sense, responsiveness of r is guaranteed if, in all states reachable without doing $\xrightarrow{[r]}$, a communication on r is weakly enabled.

Definition 2 (responsiveness). *Let P be a process and $c \in \mathrm{fn}(P)$. We say that P* guarantees responsiveness of r *if whenever $P \xrightarrow{[s]} P'$ ($s \in \mathcal{N}^*$) and r does not occur in s then $P' \overset{[r]}{\Longrightarrow}$.*

3 The type system $\vdash_1$

The type system consists of judgments of the form $\Gamma; \Delta \vdash_1 P$, where Γ and Δ are sets of names.

3.1 Overview of the system

Informally, names in Γ are those used by P in input, while in Δ are those used by P in output actions. There are several constraints on the usage of these names by P. A name in Γ must occur *immediately* (at top level) in input subject position, exactly once if it is responsive and replicated if it is ω-receptive. A responsive name in Δ must occur in P exactly once either in subject or in object output position, although not necessarily at top level, that is, occurrences in output actions underneath prefixes are allowed. There are no constraints on the use in output actions of ω-receptive names: they may be used an unbounded number of times, including zero. Linearity ("exactly once" usage) on responsive names is useful to avoid dealing with "dangling" responsive names, that might arise after a communication, like in (r responsive, object parts ignored):

$$(\nu r)(r.\mathbf{0}|\overline{r}|\overline{r}) \xrightarrow{\tau} (\nu r)(\mathbf{0}|\mathbf{0}|\overline{r})\,.$$

If the process on the LHS above were declared well-typed, this transition would violate the subject reduction property, as the process on the RHS above cannot be well-typed.

Linearity and receptiveness alone are not sufficient to guarantee a responsive usage of names. As discussed in the Introduction, we have also to avoid deadlock situations involving responsive names, like (2). This is achieved by building a *graph of dependencies* among responsive names of P (defined in the sequel) and checking for its acyclicity. We have also to avoid those situations described in the Introduction by which a responsive name is indefinitely "ping-pong"-ed among a group of replicated processes, like in (1). To this purpose, levels in types are introduced and the typing rules decree that sending a responsive name to a replicated receiver of level k may only trigger output of level less than k. This is similar to the use of levels in [5] to ensure termination. In our case, we just avoid divergent computations that involve responsive names infinitely often.

There is one more condition necessary for responsiveness, that is, the sets of input and output names must be "balanced", so as to ban situations like an output with no input counterpart. This constraint, however, is most easily formulated "on top" of well-typed-ness, and will be discussed later on.

3.2 Preliminary definitions

Formulation of the actual typing rules requires a few preliminary definitions. *Structural equivalence* is necessary in order to correctly formulate the absence of cyclic waiting on responsive names. We define structural equivalence $\equiv$ as the least equivalence relation over processes satisfying the axioms below and closed under restriction and parallel composition. Let us point out a couple of differences from the standard notion [12]. First, there is no rule for replication ($!P \equiv P|!P$), as its right-hand side would not be well-typed. Consider e.g. the process $!a(x).R$, with a ω-receptive name; the process $a(x).R\,|\,!a(x).R$ is not well-typed because ω-receptive names cannot be used as subjects of non-replicated inputs. For a similar reason, in the rule $(\nu a)\mathbf{0} \equiv \mathbf{0}$ we require $a : \bot$ or $a : \mathsf{I}$. Indeed, the type system requires that restricted ω-receptive or responsive names be used in input subject position at least once.

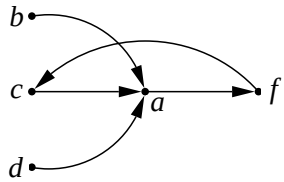
$$(\nu a)(\nu b)P \equiv (\nu b)(\nu a)P \qquad (\nu a)(P|Q) \equiv (\nu a)P|Q \text{ if } a \notin \mathrm{fn}(Q)$$
$$P|Q \equiv Q|P \qquad P|\mathbf{0} \equiv P$$
$$(P|Q)|R \equiv P|(Q|R) \qquad (\nu a)\mathbf{0} \equiv \mathbf{0} \qquad \text{if } a : \bot \text{ or } a : \mathsf{I}$$

Let us call a process P *prime* if P is of the form either $\overline{a}\langle b\rangle$, or $a(x).P'$ or $!a(x).P'$. A process P is in *normal form* if P is of the form $(\nu\tilde{d})(P_1|\cdots|P_n)$ ($n \geq 0$), where every P_i is prime and $\tilde{d} \subseteq \mathrm{fn}(P_1,...,P_n)$.

In a dependency graph, defined below, nodes are responsive names and there is an arc from a to b exactly when an output action that involves a depends on an input action on b. Although the following definition does not mention processes, one should think of the pairs (Γ_i,Δ_i) mentioned below as typing contexts – limited to responsive names – for prime P_i's in $P_1|\cdots|P_n$.

Definition 3 (dependency graph). *Let* $\{(\Gamma_i,\Delta_i) : i = 1,...,n\}$ *be a set of context pairs. The* dependency graph $\mathrm{DG}(\Gamma_i,\Delta_i)_{i=1,...,n}$ *is a graph* (V,T) *where:* $V = \bigcup_{i=1,...,n}(\Gamma_i \cup \Delta_i)$ *is the set of nodes and* $T = \bigcup_{i=1,...,n}(\Delta_i \times \Gamma_i)$ *is the set of arcs.*

Example 1. Consider the sets of names $\Gamma_1 = \{a\}$, $\Delta_1 = \{b,c,d\}$, $\Gamma_2 = \{f\}$, $\Delta_2 = \{a\}$, $\Gamma_3 = \{c\}$ and $\Delta_3 = \{f\}$. By applying the typing rules introduced in the next section, these contexts can be used for deriving well-typedness of the process $a.(\overline{b}\,|\,\overline{c}\,|\,\overline{d})\,|\,f.\overline{a}\,|\,c.\overline{f}$. The graph $\mathrm{DG}(\Gamma_i,\Delta_i)_{i=1,2,3}$ depicted below is cyclic.



We will have more to say on both structural equivalence and dependency graphs in Remark 1 at the end of the section. Like in [5], we will use a function $\mathrm{os}(P)$, defined below, that collects all – either free or bound – names in P that occur as subject of an *active* output action, that is, an output not underneath a replication (!).

$$\mathrm{os}(\mathbf{0}) = \emptyset \qquad \mathrm{os}(!a(b).P) = \emptyset \qquad \mathrm{os}(a(b).P) = \mathrm{os}(P)$$
$$\mathrm{os}(\overline{a}\langle b\rangle) = \{a\} \qquad \mathrm{os}((\nu a)P) = \mathrm{os}(P) \qquad \mathrm{os}(P|Q) = \mathrm{os}(P) \cup \mathrm{os}(Q)$$

Finally, some notation for contexts and types. For any name a, we set $\mathrm{lev}(a) = k$ if $a : \mathsf{T}^{[\mathsf{u},k]}$ for some T and u, otherwise we leave $\mathrm{lev}(a)$ undefined. Given a set of names V, define $V^{\rho} \stackrel{\triangle}{=} \{x \in V \mid x \text{ is responsive }\}$ and $V^{\omega} \stackrel{\triangle}{=} \{x \in V \mid x \text{ is } \omega\text{-receptive }\}$. For V and W sets of names, we define $V \ominus W \stackrel{\triangle}{=} V \setminus W^{\rho}$. If $\Delta \cap \Delta' = \emptyset$, we abbreviate $\Delta \cup \Delta'$ as Δ,Δ' and if $a \notin \Delta$, we abbreviate $\Delta \cup \{a\}$ as Δ,a; similarly for Γ.

3.3 The typing rules

The type system is displayed in Table 2. Recall that each sort has an associated type. Linear usage of responsive names is ensured by rules (T-NIL) and (T-OUT), by the disjointness conditions in (T-PAR) and by forbidding responsive names to occur free underneath replication (T-REP). Absence of cyclic waiting involving responsive names is checked in (T-PAR) and in (T-INP) ($a \notin \Delta$). Note the use of levels in rule (T-REP): communication involving a replicated input subject a and a responsive object can only

trigger outputs of level less than $\mathrm{lev}(a)$. This condition is meant to avoid those never-ending "ping-pongs" of responsive names mentioned above. Finally, rule (T-RES) ensures that bound responsive names are used both in input and in output and ω-receptive names are used at least as input subjects. Rule (T-RES-$\bot$) prevents from using a name of type $\bot$ and (T-RES-I) deals with inert names. We say that a process P is *well-typed* if there are Γ and Δ such that $\Gamma;\Delta \vdash_1 P$ holds.

$$\text{(T-NIL)}\quad \frac{\Delta^{\rho} = \emptyset}{\emptyset;\Delta \vdash_1 \mathbf{0}} \qquad \text{(T-OUT)}\quad \frac{a,b \in \Delta \quad a : \mathsf{T}^{\cup} \quad b : \mathsf{T} \quad \Delta^{\rho} \ominus \{a,b\} = \emptyset}{\emptyset;\Delta \vdash_1 \overline{a}\langle b\rangle}$$

$$\text{(T-STR)}\quad \frac{P \equiv Q \quad \Gamma;\Delta \vdash_1 Q}{\Gamma;\Delta \vdash_1 P} \qquad \text{(T-INP)}\quad \frac{a : \mathsf{T}^{[\rho,k]} \quad b : \mathsf{T} \quad a \notin \Delta \quad \emptyset;\Delta,b \vdash_1 P}{a;\Delta \vdash_1 a(b).P}$$

$$\text{(T-RES-}\bot\text{)}\ \frac{a : \bot \quad \Gamma;\Delta \vdash_1 P}{\Gamma;\Delta \vdash_1 (\nu a)P} \qquad \text{(T-RES-I)}\ \frac{a : \mathsf{I} \quad \Gamma;\Delta,a \vdash_1 P}{\Gamma;\Delta \vdash_1 (\nu a)P} \qquad \text{(T-RES)}\ \frac{a : \mathsf{T}^{\cup} \quad \Gamma,a;\Delta,a \vdash_1 P}{\Gamma;\Delta \vdash_1 (\nu a)P}$$

$$\text{(T-REP)}\quad \frac{a : \mathsf{T}^{[\omega,k]} \quad b : \mathsf{T} \quad \Delta^{\rho} = \emptyset \quad \emptyset;\Delta,b \vdash_1 P \quad (\ b \text{ responsive implies } \forall c \in \mathrm{os}(P) : \mathrm{lev}(c) < k\ )}{a;\Delta \vdash_1 !a(b).P}$$

$$\text{(T-PAR)}\quad \frac{P = P_1|\cdots|P_n\ \ (n>1) \quad \forall i :\ P_i \text{ is prime and } \Gamma_i;\Delta_i \vdash_1 P_i \qquad \forall i \neq j :\ \Gamma_i^{\rho} \cap \Gamma_j^{\rho} = \emptyset \text{ and } \Delta_i^{\rho} \cap \Delta_j^{\rho} = \emptyset \qquad \mathrm{DG}(\Gamma_i^{\rho},\Delta_i^{\rho})_{i=1,\ldots,n} \text{ is acyclic}}{\bigcup_{i=1,\ldots,n} \Gamma_i \ ;\ \bigcup_{i=1,\ldots,n} \Delta_i \ \vdash_1 P}$$

Bound names in processes are assumed to be different from free names and from names in contexts.

Table 2: Typing rules of $\vdash_1$

Remark 1. (1) Avoiding deadlock on responsive names might be achieved by using levels in rule (T-INP), in the same fashion as in rule (T-REP), rather than using graphs. In fact, this would rule out cyclic waiting such as the one in (2) in the Introduction. We shall pursue this approach in the system of Section 6, where there is no way of defining a meaningful notion of dependency graph. However, in the present system this way of dealing with cyclic waiting would be unnecessarily restrictive, in particular it would ban as ill-typed the usual encoding of recursive functions into processes (see also Section 8).

(2) We note that, despite the presence of a rule for structural equivalence, the type system may be viewed as essentially syntax driven, in the following sense. Given P in normal form, $P = (\nu\tilde{d})(P_1|\cdots|P_n)$, and ignoring structural equalities that just rearrange the $\tilde{d}$ or the P_i's, there is at most one rule one can apply with P in the conclusion. This is made formal below.

We define a normal derivation of $\Gamma;\Delta \vdash_1 P$ to be one where rule (T-STR) is applied only where strictly necessary:

Definition 4 (normal derivation). *A* normal derivation *of* $\Gamma;\Delta \vdash_1 R$ *is a derivation where at each application of rule* (T-STR) *(Table 2) the process P in the conclusion* is not *in normal-form, while the process Q in the premise* is *in normal form.*

For each well typed process P there exists a normal derivation (the proof is reported in Appendix A).

Lemma 1. *Suppose* $\Gamma;\Delta \vdash_1 P$*, then there exists a normal derivation of* $\Gamma;\Delta \vdash_1 P$*.*

Example 2. Consider the process

$$P \triangleq c\,|\,\overline{a}\langle c\rangle\,|\,a(x).(\nu b)\big(b.\overline{x}\,|\,\overline{f}\langle b\rangle\big)\,|\,f(y).\overline{y}$$

with $c,x : \mathsf{I}^{[\rho,k_c]} = \mathsf{T}$, $a : \mathsf{T}^{[\rho,k_a]}$, $b,y : \mathsf{I}^{[\rho,k_b]} = \mathsf{S}$, $f : \mathsf{S}^{[\rho,k_f]}$, for any k_c, k_a, k_b and k_f, and context $\Gamma = \Delta = \{a,c,f\}$. P is a parallel composition of well-typed prime processes, the resulting dependency graph is acyclic and rule (T-PAR) can be applied for deducing $\Gamma;\Delta \vdash_1 P$. As we will see in the next section (Theorem 2) responsiveness of c is guaranteed.

Example 3. As expected, there exist processes that guarantee responsiveness, but which are discarded by our type system. Consider e.g.

$$P \triangleq \overline{a}\langle c\rangle\,|\,!a(x).\overline{a}\langle x\rangle\,|\,a(x).(\overline{x}\,|\,x)\ .$$

Assume c is responsive. Process P is not well typed, because channel a is used in input as a subject of both a replicated input – hence it cannot be responsive, (T-REP) – and of a simple input – hence it cannot be ω-receptive, (T-INP). However, P guarantees responsiveness of c according to Definition 2. Indeed, along every computation not involving c, a reduction on c is weakly enabled at any stage (which guarantees that reduction $\xrightarrow{[c]}$ will take place under a fair scheduling assumption).

4 Subject reduction and type soundness for system $\vdash_1$

Subject reduction states that well-typedness is preserved through reductions, and it is our first step towards proving type soundness. Proofs omitted here are reported in Appendix B, C and D.

Theorem 1 (subject reduction). *Suppose* $\Gamma;\Delta \vdash_1 P$ *and* $P \xrightarrow{[a]} P'$. *Then* $\Gamma \ominus \{a\};\Delta \ominus \{a\} \vdash_1 P'$.

Our task is proving that any "balanced" well-typed process guarantees responsiveness (Definition 2) for all responsive names it contains. In the following definition we formally identify balanced processes.

Definition 5 (balanced processes). *A process* P *is* $(\Gamma;\Delta)$-balanced *if* $\Gamma;\Delta \vdash_1 P$, $\Gamma^\rho = \Delta^\rho$ *and* $\Delta^\omega \subseteq \Gamma^\omega$. *It is* balanced *if it is* $(\Gamma;\Delta)$*-balanced for some* Γ *and* Δ.

We need two main ingredients for the proof. The first one is given by the following proposition, stating that if the dependency graph of a process P is acyclic, then P always offers at least one output action involving a responsive name.

Proposition 1. *Suppose that* $\Gamma;\Delta \vdash_1 P$, *with* Γ, Δ *and* P *satisfying the conditions in the premise of rule* (T-PAR) *and* $\Gamma^\rho = \Delta^\rho \neq \emptyset$. *Then for some* $j \in \{1,\ldots,n\}$ *we have* $P_j = \overline{a}\langle b\rangle$ *with either* a *or* b *responsive.*

Following Deng and Sangiorgi's approach, we define a measure of processes that is decreased by reductions involving responsive names. We borrow from [5] the definition of *weight* of P, written $\mathrm{wt}(P)$. In particular, $\mathrm{wt}(P)$ is defined only if P is well-typed and is a vector $\langle w_k, w_{k-1}, \ldots, w_0\rangle$, where $k \geq 0$ is the highest level of names in $\mathrm{os}(P)$, and w_i is the number of occurrences in output subject position of names of level i in P. A formal definition is given below. It is woth to notice that in $\mathrm{wt}(\overline{a}\langle b\rangle)$ it can never be the case that a is of type I nor $\bot$, because otherwise $\overline{a}\langle b\rangle$ would not be well-typed. Here, "0_k" is an abbreviation for the vector $\langle 1,0,\ldots,0\rangle$ with k components "0" following "1". The vector with just one component that equals "0" is denoted by 0. Sum "$+$" between two vectors is performed component-wise if they are of the same length; if not, the shorter one is first "padded" by inserting on the left as many 0's as needed.

$$\mathrm{wt}(\mathbf{0}) = 0 \qquad \mathrm{wt}(!a(b).P) = 0 \qquad \mathrm{wt}(\overline{a}\langle b\rangle) = 0_k \text{ if } \mathrm{lev}(a) = k$$
$$\mathrm{wt}(a(b).P) = \mathrm{wt}(P) \qquad \mathrm{wt}((\nu a)P) = \mathrm{wt}(P) \qquad \mathrm{wt}(P|Q) = \mathrm{wt}(P) + \mathrm{wt}(Q)$$

The set of all vectors can be ordered lexicographically. Assuming two vectors are of equal length (if not, the shorter vector is padded with 0's on the left), we define $\langle w_k,\ldots,w_0\rangle \prec \langle w'_k,\ldots,w'_0\rangle$ if there is i in $0,\ldots,k$ such that $w_j = w'_j$ for all $k \geq j > i$ and $w_i < w'_i$. This order is total and well-founded, that is, there are no infinite descending chains of vectors. The next proposition states that the weight of a process is decreased by reductions involving a responsive name, and leads us to Theorem 2, which is the main result of the section.

Lemma 2. *For each P such that $\Gamma;\Delta \vdash_1 P$ there exists R in normal form such that $P \equiv R$.*

Proposition 2. *Suppose $\Gamma;\Delta \vdash_1 P$ and $P \xrightarrow{\tau\langle a,b\rangle} P'$, with either a or b responsive. Then $\mathrm{wt}(P') \prec \mathrm{wt}(P)$.*

Theorem 2 (type soundness). *Let P be $(\Gamma;\Delta)$-balanced and $r \in \Delta^{\rho}$. Then P guarantees responsiveness of r.*

PROOF: Assume $P \xrightarrow{[s']} R$, for any R, and $r \notin s'$. We have to show that $R \overset{[r]}{\Longrightarrow}$. Let P' be a process with a *minimal* $\mathrm{wt}(\cdot)$ satisfying $R \xrightarrow{[s'']} P'$ for some s'' such that $r \notin s''$: this P' must exist by well-foundedness of $\prec$. Let $s = s' \cdot s''$. By subject reduction we have that P' is $(\Gamma';\Delta')$-balanced, with $\Gamma' = \Gamma \ominus s$ and $\Delta' = \Delta \ominus s$.

Consider now a normal form of process P' (Lemma 2): $P' \equiv N \overset{\triangle}{=} (\nu\tilde{d})N'$ with $N' = P_1|\cdots|P_n$ for some $P_1,\cdots,P_n$ prime. By rule (T-STR), we get $\Gamma;\Delta \vdash_1 N$. Therefore, we deduce that it must be $n > 1$, as r occurs in both input and output and, by rule (T-INP), an output $\overline{r}$ cannot occur under an input on r.

By Lemma 1, there exists a normal derivation of $\Gamma';\Delta' \vdash_1 N$. In this derivation, $\Gamma';\Delta' \vdash_1 N$ is deduced from $\Gamma',\tilde{d};\Delta',\tilde{d} \vdash_1 P_1|\cdots|P_n$ by repeated applications of (T-RES) and (T-RES-I), and rule (T-PAR) must have been applied to infer $\Gamma',\tilde{d};\Delta',\tilde{d} \vdash_1 N'$. Hence it must be: $(\Gamma',\tilde{d}) = \bigcup_{i=1,\ldots,n}\Gamma_i$, and $(\Delta',\tilde{d}) = \bigcup_{i=1,\ldots,n}\Delta_i$, and $\Gamma_i;\Delta_i \vdash_1 P_i$, where Δ_i^{ρ} (resp. Γ_i^{ρ}) are pairwise disjoint and $\mathrm{DG}(\Gamma_i^{\rho},\Delta_i^{\rho})_{i=1,\ldots,n}$ is acyclic. Moreover, from balancing of Γ and Δ and definition of $\ominus$ we deduce the balancing of $\Gamma',\tilde{d}$ and $\Delta',\tilde{d}$, hence $(\Delta',\tilde{d})^{\rho} = (\Gamma',\tilde{d})^{\rho}$. By Proposition 1 there is a j such that $P_j = \overline{a}\langle b\rangle$ with a or b responsive. By (T-OUT) and $\Gamma',\tilde{d};\Delta',\tilde{d} \vdash_1 N'$ we have $a \in \Delta',\tilde{d}$. By $(\Delta',\tilde{d})^{\omega} \subseteq (\Gamma',\tilde{d})^{\omega}$ and receptiveness of responsive and ω-receptive names ((T-INP) and (T-REP)), there is a k such that $P_k = (!)a(x).P'_k$. This implies $N' \xrightarrow{\tau\langle a,b\rangle} N''$, by (COM), hence $N \xrightarrow{\tau\langle a,b\rangle} M$. Hence, $N \equiv P' \xrightarrow{\tau\langle a,b\rangle} P'' \equiv M$ as well and, since either a or b is responsive, by Proposition 2 we get $\mathrm{wt}(P'') \prec \mathrm{wt}(P')$. This implies $a = r$, as P' was assumed to be the process with minimal weight satisfying $R \xrightarrow{[s'']} P'$, for some s'' such that $r \notin s''$. Hence we have proved that $R \overset{[r]}{\Longrightarrow}$. □

Next, we establish an upper bound on the number of steps that are always sufficient for a given responsive name to be used as subject. This upper bound can be given as a function of the syntactic size of P, written $|P|$, and of name levels in P. A similar result was given in [5] for terminating processes. Here, since we deal with processes that in general may not terminate, the upper bound must be given relatively to a notion of *scheduling* of transitions, that is introduced below.

Definition 6 (responsive scheduling). *A* responsive scheduling *is a finite or infinite sequence of reductions $P = P_0 \xrightarrow{\tau\langle a_1,b_1\rangle} P_1 \xrightarrow{\tau\langle a_2,b_2\rangle} \cdots$ where the bound names in $\{(a_i,b_i)|i \geq 1\}$ are all distinct from the free names in P and for each $i \geq 0$, either a_i or b_i is responsive.*

The size of a process P, written $|P|$, is defined as

$$|\mathbf{0}| = 0 \qquad |a(x).P| = 1 + |P| \qquad |(\nu c)P| = |P|$$
$$|\overline{a}\langle b\rangle| = 1 \qquad |!a(x).P| = 1 + |P| \qquad |P|R| = |P| + |R| \,.$$

Note that structural equivalence preserves the size of a process.
We denote by $\mathrm{O}(P)$ the multiset of all output actions of P that are active, that is, not underneath a replication. $\mathrm{O}(P)$ is formally defined as follows

$$\mathrm{O}(\mathbf{0}) = \emptyset \qquad \mathrm{O}(a(b).P) = \mathrm{O}(P) \qquad \mathrm{O}(\overline{a}\langle b\rangle) = \{\overline{a}\langle b\rangle\}$$
$$\mathrm{O}(!a(b).P) = \emptyset \qquad \mathrm{O}((\nu a)P) = \mathrm{O}(P) \qquad \mathrm{O}(P|R) = \mathrm{O}(P) \uplus \mathrm{O}(R) \,.$$

We indicate by $\mathrm{O}^{\rho}(P)$ the multiset containing all output actions in $\mathrm{O}(P)$ involving a responsive name.

Theorem 3. *Let P be $(\Gamma;\Delta)$-balanced and $r \in \Delta^{\rho}$ and let k be the maximal level of names appearing in active responsive output actions of P, $\mathrm{O}^{\rho}(P)$. In all responsive schedulings, the number of reductions preceding a reduction on r is upper-bounded by $|P|^{k+1}$.*

The proof relies on Theorem 2 (type soundness), which ensures that a communication on r must take place. The maximal number of communications that can precede the reduction on r is estimated by considering that each reduction can increase the number of outputs – that is, of potential reductions – in the continuation, but this increase is limited by the initial size of the process (see Appendix D for a detailed proof).

5 Extensions of system $\vdash_1$

In this section we introduce two simple extensions of our type system.

5.1 Summation and if-then-else

We introduce guarded summation and `if-then-else` and extend the original definitions and results to the new constructs.

Summation. In the process syntax, plain input prefix is replaced by guarded summation

$$P ::= \cdots \Big| \sum_{i\in I} a_i(x_i).P_i$$

which, as expected, has the following transition rule:

$$(\textsc{sum})\ \frac{j \in I}{\sum_{i\in I} a_i(x_i).P_i \xrightarrow{a_j(b)} P_j[b/x_j]} \,.$$

A summation is well-typed if all its branches can be typed under one and the same context:

$$(\text{T-}\textsc{Sum})\ \frac{\forall i \in I:\ \Gamma;\Delta \vdash_1 a_i(x_i).P_i \quad |I| > 1}{\Gamma;\Delta \vdash_1 \sum_{i\in I} a_i(x_i).P_i} \,.$$

This rule implies that $a_i = a_j$ for each $i, j \in I$ and that all responsive names in Δ^{ρ} are used in output in each branch. The results introduced in Section 4 still hold for the extended calculus, modulo a few notational changes described below. First, processes of the form $\sum_{i\in I} a_i(x_i).P_i$ are prime. Concerning the functions defined on processes, we have

$$|\sum_{i\in I} a_i(x_i).P_i| = \max_{i\in I}|a_i(x_i).P_i| \quad \mathrm{os}(\sum_{i\in I} a_i(x_i).P_i) = \bigcup_{i\in I}\mathrm{os}(a_i(x_i).P_i)$$
$$\mathrm{O}(\sum_{i\in I} a_i(x_i).P_i) = \biguplus_{i\in I}\mathrm{O}(a_i(x_i).P_i) \quad \mathrm{wt}(\sum_{i\in I} a_i(x_i).P_i) = \sum_{i\in I}\mathrm{wt}(a_i(x_i).P_i)\ .$$

All proofs are obvious generalization (with summations replacing inputs) of those reported in Appendices B and C, hence omitted.

If-then-else. The syntax of processes is extended as follows:

$$P ::= \cdots \mid \texttt{if}\ G\ \texttt{then}\ P\ \texttt{else}\ P.$$

We leave the syntax of guards G unspecified, but assume guards can be formed using predicates over names (e.g. $(a = b)$). We assume an evaluation function that maps each guard G to *true* or $false$: $G \rightsquigarrow true$ or $G \rightsquigarrow false$.

The operationl semantics of the `if-then-else` construct is as usual:

$$(\textsc{if-t})\ \frac{G \rightsquigarrow true}{\texttt{if}\ G\ \texttt{then}\ P\ \texttt{else}\ Q \xrightarrow{[\varepsilon]} P} \qquad (\textsc{if-f})\ \frac{G \rightsquigarrow false}{\texttt{if}\ G\ \texttt{then}\ P\ \texttt{else}\ Q \xrightarrow{[\varepsilon]} Q}$$

There are no new structural rules. An `if-then-else` is well-typed if both branches are well-typed:

$$(\textsc{t-if})\ \frac{\Gamma;\Delta \vdash_1 P \quad \Gamma;\Delta \vdash_1 Q}{\Gamma;\Delta \vdash_1 \texttt{if}\ G\ \texttt{then}\ P\ \texttt{else}\ Q}\ .$$

The results presented in Section 4 can be extended to the calculus enriched with `if-then-else` (see Appendix E; for the sake of simplicity, we omit the extension of Theorem 3, which would require additional technicalities to take into account ε-transitions originated by (IF-T) and (IF-F)).

Example 4. A web portal, available at *portal*, allows users to subscribe to a given service, subject to an assessment of their reliability. Any client contacting the portal must supply its personal data, d. The portal passes the personal data to a sub-service, reachable at *assess*, who actually performs the assessment. The result of the assessment can be either "*high*" or "*low*" reliability. After receiving this piece of information from the assessment service, the portal produces a security token t, which is internally associated with the client's reliability and personal data, and then passes t onto the client in response. At a later time, the client contacts, at *subscribe*, the subscription service providing it the token t. The subscription service uses t to retrieve the client's reliability (and private information) and grants or denies subscription according to the following policy: subscription requests originating from clients with "*high*" reliability are always accepted, while those from clients with "*low*" reliability may be either accepted or rejected, depending on other circumstances which are left out of the model.

An abstract description of this system is given by $Sys \stackrel{\triangle}{=} P|A|S|C$ where (internal nondeterministic

sum $\overline{s}\langle\text{“}ok\text{”}\rangle \oplus \overline{s}\langle\text{“}nok\text{”}\rangle$ is an abbreviation for $(\nu a)(\overline{a} \mid a.\overline{s}\langle\text{“}ok\text{”}\rangle + a.\overline{s}\langle\text{“}nok\text{”}\rangle)$):

$$
\begin{aligned}
P &\triangleq !portal(d,r).(\nu s)\Big(\overline{assess}\langle d,s\rangle \mid s(x).(\nu t)\big(\texttt{if } x = \text{“}ok\text{”} \\
&\qquad\qquad \texttt{then } (\overline{r}\langle t\rangle \mid t(v).\overline{v}\langle\text{“}high\text{”}, d\rangle) \\
&\qquad\qquad \texttt{else } (\overline{r}\langle t\rangle \mid t(v).\overline{v}\langle\text{“}low\text{”}, d\rangle)\big)\Big) \\
A &\triangleq !assess(d,s).(\overline{s}\langle\text{“}ok\text{”}\rangle \oplus \overline{s}\langle\text{“}nok\text{”}\rangle) \\
S &\triangleq !subscribe(t,q).(\nu v)\big(\overline{t}\langle v\rangle \mid v(w,d).\texttt{if } w = \text{“}high\text{”} \\
&\qquad\qquad \texttt{then } \overline{q}\langle\text{“}done\text{”}\rangle \\
&\qquad\qquad \texttt{else } (\overline{q}\langle\text{“}deny\text{”}\rangle \oplus \overline{q}\langle\text{“}done\text{”}\rangle)\big) \\
C &\triangleq (\nu data)(\nu r)\Big(\overline{portal}\langle data, r\rangle \mid r(t).(\nu q)\big(\overline{subscribe}\langle t,q\rangle \mid q(x).C'\big)\Big)\,.
\end{aligned}
$$

It is worth noticing that the token t can be viewed as a temporary service that is delegated by *portal* to pass the client's personal information directly to the *subscribe* service.

Sys is balanced under the assumption that: C' is balanced, $r,t,q \notin \mathrm{fn}(C')$, $\mathrm{lev}(portal) > \mathrm{lev}(assess), \mathrm{lev}(r), \mathrm{lev}(v)$, $\mathrm{lev}(assess) > \mathrm{lev}(s)$ and $\mathrm{lev}(subscribe) > \mathrm{lev}(t), \mathrm{lev}(q)$, while *data*, "*high*", "*low*", "*ok*", "*done*", "*deny*", d, w and x are of sort inert. Therefore, a communication on q is guaranteed to take place and the client is ensured to receive a reply to its subscription request.

Note that, formally, we are not allowed to talk about responsiveness of q, which is bound in *Sys*. To get around this small difficulty, assume $C' \triangleq \overline{done}|C''$ and consider $Sys|done$, in place of *Sys*, with *done* responsive: then Theorem 2 ensures that a communication on *done* is guaranteed, which in turn implies that a communication on q must eventually take place.

5.2 Recursion on well-founded data values

The system presented in Section 3 bans as ill-typed processes implementing recursive functions. As an example, consider the traditional implementation of the factorial function, the process P below. For the purpose of illustration, let us consider a polyadic version of the calculus enriched with natural numbers, variables $(n,m,\ldots)$ and predicates/functions on them as expected.

$$P \triangleq !f(n,r).\texttt{if } n = 0 \texttt{ then } \overline{r}\langle 1\rangle \texttt{ else } (\nu r')\big(\overline{f}\langle n-1, r'\rangle \mid r'(m).\overline{r}\langle m * n\rangle\big). \tag{3}$$

It would be natural to see f as ω-receptive and r and r' as responsive, but under these assumptions P would not be well-typed: the recursive call $\overline{f}\langle n-1, r'\rangle$ violates the constraint on levels of output actions under replication (rule (T-REP)). Nevertheless, it is natural to see the output $\overline{f}\langle n-1, r'\rangle$, triggered by a recursive call at f, as "smaller" than the output $\overline{f}\langle n, r\rangle$ that has triggered it: at least, this is true if one takes into account the ordering relation on natural numbers. This means that the "weight" of the process decreases after each recursive call, and since natural numbers are well-founded, after some reductions no further recursive call will be possible, and a communication on r must take place. This idea from [5] is adapted here to our type system. For simplicity, we only consider the domain of natural values Nat. However, the results may be extended to any data type on which a well-founded ordering relation can be defined. We define an ordering relation "$<$" between (possibly open) integer expressions and variables as follows: $e < n$ if, for each evaluation ρ under which e is defined, $e\rho < \rho(n)$. E.g., $n-1 < n$. In the case of the monadic calculus, this relation is lifted to a "smaller than" relation $\lhd$ between output and input actions as follows. Below, d, d' denote either names or (open) expressions.

Definition 7 (ordering on actions). *We write* $\overline{c}\langle d\rangle \lhd a(d')$ *if either* $\mathrm{lev}(c) < \mathrm{lev}(a)$ *or* $\mathrm{lev}(c) = \mathrm{lev}(a)$ *and* $d = e < x = d'$.

The "$\lhd$" relation is used in the typing rule below, that replaces rule (T-REP).

$$\text{(T-REP')}\quad \frac{\begin{array}{c} a : \mathsf{T}^{[\omega,k]} \quad b : \mathsf{T} \quad \Delta^{\rho} = \emptyset \quad \emptyset; \Delta, b \vdash_1 P \\ (b : \mathsf{Nat} \text{ or } b \text{ responsive}) \quad \text{implies} \quad \forall \overline{c}\langle d\rangle \in \mathrm{O}(P) : \ \overline{c}\langle d\rangle \lhd a(b) \end{array}}{a; \Delta \vdash_1 !a(b).P}$$

In the polyadic case, "$\lhd$" compares first the subject and then the object parts of two actions lexicographically; this is a correct choice because the lexicograhic ordering is well-founded. (Actually, this is not the only possibility, see e.g. [5].) More precisely, "$\lhd$" is generalized as follows. We write $\overline{c}\langle d_1, \cdots, d_k\rangle \lhd a(d'_1, \cdots, d'_k)$ if either $\mathrm{lev}(c) < \mathrm{lev}(a)$ or $\mathrm{lev}(c) = \mathrm{lev}(a)$ and for some j, with $1 \leq j \leq k$, it holds that $d_j = e < n = d'_j$ and $d_i = d'_i$ for each $1 \leq i < j$. E.g. assuming that $\mathrm{lev}(c) = \mathrm{lev}(a)$, it holds that $\overline{c}\langle n-1, m+1, r\rangle \lhd a(n, m, s)$, for each channel name r and s. As an example of application of (T-REP'), it is easy to see that the process P in (3) is well-typed if $f : (\mathsf{Nat}, \mathsf{Nat}^{[\rho,0]})^{[\omega,1]}$ and $r, r' : \mathsf{Nat}^{[\rho,0]}$.

The proof of type soundness remains the same, modulo a change in function $\mathrm{wt}(\cdot)$. Here, we need a measure that records, for each output prefix, not only the level of the subject, but also the value of the corresponding object. This can be achieved by considering a *compound vector*, which consists of two parts: the weight and a multiset of natural values recording the objects' contribution to $\mathrm{wt}(\cdot)$. We omit the details of the formal definition, which can be found in [5].

Primitive Recursive Functions can be encoded into well-typed processes, with (T-REP) replaced by (T-REP'). The schema of the encoding is an easy generalization of that seen in (3) above for the factorial function. We have the following result (the proof is reported in Appendix F).

Proposition 3. *For every k-ary primitive recursive function* f *there is a well-typed process* $\langle f\rangle_b$ *such that: for each* $(v_1, \ldots, v_k)$ *in* Nat^k *the process* $G \stackrel{\triangle}{=} (\nu b)(\langle f\rangle_b | \overline{b}\langle v_1, \ldots, v_k, r\rangle \,|\, r(n).\mathbf{0})$, *with* b ω*-receptive and* $r : (\mathsf{Nat})^{[\rho,h]}$ *(*$h \geq 0$*), is balanced. Moreover,* $f(v_1, \ldots, v_k) = m$ *if and only if* $G \xrightarrow{\tau}{}^{*} \xrightarrow{\overline{r}\langle m\rangle}$.

6 Nested inputs, multiple outputs: the type system $\vdash_2$

The type system presented in Section 3 puts rather severe limitations on nesting of input actions and multiple use of channels. These limitations stem from the "immediate receptiveness" and linearity conditions imposed on responsive names. For instance, the following encoding of internal choice $\overline{r}\langle a\rangle \oplus \overline{r}\langle b\rangle$, where r is responsive and a, b inert, is not well-typed

$$(\nu c)(\overline{c}\langle a\rangle | \overline{c}\langle b\rangle \,|\, c(x).\overline{r}\langle x\rangle)\,. \tag{4}$$

Limitations are also built-in in process syntax, as for example replicated outputs, that clearly violate linearity, are not permitted. Replicated outputs might be useful to encode environments holding constants. As an example, in the process below an environment with one entry a is initialized with the first input received at r, and then repeatedly read at a (these situations do arise in the encoding of high-level languages into pi-calculus):

$$(\nu a)\big(r(x).!\overline{a}\langle x\rangle | a(y).P | a(y).Q | a(y).R\big)\,. \tag{5}$$

For another example, a process that receives two integers in a fixed order from two return channels, r_1 and r_2, and then outputs the max along s, may not be well-typed

$$r_1(n).r_2(m).\mathtt{if}\ n \geq m\ \mathtt{then}\ \overline{s}\langle n\rangle\ \mathtt{else}\ \overline{s}\langle m\rangle\,. \tag{6}$$

In fact, the type system $\vdash_1$ does not allow (free) input actions guarded by other inputs.

We present below a new type system $\vdash_2$ that overcomes the limitations discussed above. In fact, we will trade off flexibility for expressiveness in terms of encodable functions, as only certain patterns of (tail-)recursion will be well-typed in the new system.

6.1 Syntax and operational semantics

We extend the syntax of processes by introducing replicated output and the syntax of types by introducing a new responsive usage of names, ρ^+, as follows:

$$\begin{array}{lcl} P & ::= & \cdots \mid !\overline{a}\langle b\rangle \\ \mathsf{U} & ::= & \cdots \mid [\rho^+, k]. \end{array}$$

A name $a : \mathsf{T}^{[\rho^+,k]}$ is called *+-responsive*, as it is meant to be used *at least once* as subject of a communication. Therefore now we consider three different usages: ρ (for names used once), ρ^+ (for names used at least once) and ω (for names used an undefined number of times.) We point out that responsive names are not subsumed by +-responsive: in particular, as we shall see, the conditions on the type of carried objects are more liberal for responsive names. Operational semantics is enriched by adding the obvious rule for replicated output:

$$!\overline{a}\langle b\rangle \xrightarrow{\overline{a}\langle b\rangle} !\overline{a}\langle b\rangle \ .$$

6.2 Overview of the system

We give here an informal overview of the type system. Judgments are of the form $\Gamma;\Delta \vdash_2 P$ where in Γ and Δ each +-responsive name a is annotated with a *capability* t, written a^t. A capability t can be one of four kinds: n (*null*), s (*simple*), m (*multiple*) and p (*persistent*). Informally, capabilities have the following meaning (in the examples below, we ignore object parts of some actions and assume b is a (+-)responsive name):

- a^{n} indicates that a cannot be used at all. This capability has been introduced to uniformly account for +-responsive names that disappear after being used as subjects.
- a^{s} indicates that a appears exactly once and not under a replication. Examples: $a.P$, $b.a.P$, $\overline{a}$ and $b.\overline{a}$.
- a^{m} indicates that a appears at least once, even under replication, but never as subject of a replicated action. Examples: $a.P|a.Q$, $!b.a.P$ and $!b.\overline{a}$.
- a^{p} indicates that a only appears as subject of a replicated action. Examples: $!a.P$, $!\overline{a}$, $b.!\overline{a}$ and $!b.!\overline{a}$.

Note that a name a may be given distinct capabilities in input (Γ) and output (Δ). E.g. one may have, again ignoring the object parts, $\Gamma;\Delta \vdash_2 !a.P|\overline{a}|\overline{a}$, where $a^{\mathsf{p}} \in \Gamma$ and $a^{\mathsf{m}} \in \Delta$. Next we illustrate and motivate the constraints on name usages realized by the typing rules. They guarantee correct usage of +-responsive names under the balancing conditions discussed in the next section. Roughly, these conditions extend those in Definition 5 by ensuring that each input action involving a +-responsive subject is always matched by a corresponding output.

(1) If $a^{\mathsf{s}} \in \Gamma$ then a occurs exactly once in input subject position. This constraint relieves one from checking that there are "enough" $\overline{a}$ available. Indeed, using a more than once in input would require ensuring that the number of inputs involving each name does not exceed the number of available outputs. E.g. the process (a and b +-responsive names)

$$a \,|\, a.b \,|\, \overline{a} \,|\, \overline{b} \xrightarrow{[a]} a.b \,|\, \overline{b} \not\rightarrow \ . \tag{7}$$

has to be discarded, because a is used twice in input and only once in output.

(2) If $a^{\mathsf{m}} \in \Gamma$ and $a^{\ell} \in \Delta$ then $\ell = \mathsf{p}$. If a is used more than once in input and at least once as output subject, then deadlocks arising from not having enough output actions of subject a, like in (7), are avoided, because the output on a is replicated as in $a \,|\, a.b \,|\, !\overline{a} \,|\, \overline{b}$.

(3) If $a^{t} \in \Gamma$ and a carries (+-)responsive names, then $t = \mathsf{p}$ – hence a must be used as subject of a replicated input. This is to avoid deadlocks arising from having not enough input of subject a that carry (+-)responsive names, like in (a +-responsive, b and d (+-)responsive names):

$$\overline{a}\langle b\rangle \,|\, \overline{a}\langle d\rangle \,|\, a(x).\overline{x} \,|\, b \,|\, d \xrightarrow{[a]} \xrightarrow{[b]} \overline{a}\langle d\rangle \,|\, d \ .$$

(4) Concerning a^{p}, names with capability p (persistent) are required to occur *exactly once* in subject position (either in input or in output). This is necessary to avoid deadlock situations due to shortage of outputs like in (b and c +-responsive)

$$!a.\overline{b} \,|\, !a.\overline{c} \,|\, \overline{a} \,|\, c \,|\, b \xrightarrow{[a]} !a.\overline{b} \,|\, !a.\overline{c} \,|\, \overline{b} \,|\, c \,|\, b \xrightarrow{[b]} !a.\overline{b} \,|\, !a.\overline{c} \,|\, c \not\xrightarrow{\tau}$$

where a communication on c would never happen.

Moreover, we ban names persistent both in input and in output. This is a simplifying condition that relieve us from dealing with divergent computations involving +-responsive names, like in (a +-responsive)

$$!\overline{a} \,|\, !a.P \ .$$

To preserve both these conditions at run-time, we have also to forbid

(i) replicated actions guarded by replicated inputs. This is to avoid situations like

$$!a.!b \,|\, \overline{a} \xrightarrow{[a]} !a.!b \,|\, !b$$

where the RHS violates the requirement that $+-$responsive names can appear exactly once as subjects of replicated inputs;

(ii) persistent names passed around as objects.

(5) Names occurring under an (either simple or replicated) input must be assigned smaller levels than the input subject. The role of this condition is twofold, now. Under replicated inputs, it avoids infinite delays, like in the first system. Under simple inputs, it serves to avoid cyclic waiting, like in (a, b (+-)responsive):

$$a.\overline{b} \,|\, b.\overline{a} \ .$$

This was achieved by the use of dependency graphs in the first system. As announced in Remark 1, however, there appears to be no meaningful extension of this notion of graph in the present system.

In particular, acyclicity of the graph might not be preserved by reductions. E.g. consider the process

$$b(x).\overline{a}\langle x\rangle|c(x).a(y).\overline{x}\langle y\rangle|\overline{c}\langle b\rangle \ .$$

Its graph is acyclic, but after a reduction on c the process become

$$b(x).\overline{a}\langle x\rangle|a(y).\overline{b}\langle y\rangle$$

and the corresponding dependency graph has a cycle involving a and b. As a by product of discarding the dependency graph, we achieve a simplification of the typing rule for parallel composition. However, this rather crude use of levels to ban cyclic waiting is also the cause of the reduced expressiveness in terms of typable functions.

Finally, we introduce a syntactic restriction. As +-responsive names used once or more than once in output are treated in the same manner, we reserve capability s for inputs, and use only m and p for outputs. This choice alleviates some technicalities in the proof of the subject reduction theorem.

6.3 The typing rules

In what follows, we denote by $\mathrm{is}(P)$ the set of either *bound* or *free* names used in input subject position in P. Contexts Γ and Δ are sets of annotated names of the form a^t, where t is a capability. Each name occurs at most once in a context. +-responsive names are annotated with one of the four capabilities n, s (only in Γ), m or p, while non-+-responsive names are always annotated with a default "$-$" capability; when convenient a^- is abbreviated simply as a. Union and intersection of two contexts, written $\Gamma_1 \cup \Gamma_2$ and $\Gamma_1 \cap \Gamma_2$, are defined only if the contexts agree on capabilities of common names, that is whenever $a^{t_i} \in \Gamma_i$ for $i = 1,2$ then $t_1 = t_2$. We write Γ_1, Γ_2 in place of $\Gamma_1 \cup \Gamma_2$ if $\Gamma_1 \cap \Gamma_2 = \emptyset$, while Γ_1, a^t abbreviates $\Gamma_1, \{a^t\}$. For any context Γ and capability t, we define $\Gamma^t \stackrel{\triangle}{=} \{a | a^t \in \Gamma\}$. The set of names $\Gamma^{\mathsf{p}^+} \stackrel{\triangle}{=} \{\, a \,|$ a is +-responsive and $a^t \in \Gamma$ for some $t \neq \mathsf{n}$ $\}$ and Γ^{p}, Γ^{ω} are defined similarly. The typing rules are presented in Table 3. We briefly comment on the rules by re-considering conditions $(1-5)$ discussed in the preceding subsection.

(1) is ensured in (T_+-PAR) by checking the disjointness of Γ_1^{s} and Γ_2^{s} and in (T_+-INP), by requiring $a \notin \Gamma$;

(2) is ensured in (T_+-PAR) by $\Gamma^{\mathsf{m}} \cap \Delta^{\mathsf{m}} = \emptyset$;

(3) is ensured in (T_+-INP) by checking that +-responsive names used as subject of non-replicated inputs cannot carry (+-)responsive objects;

(4) all rules for input ensure that received names cannot be used as subjects of replicated outputs (by enforcing the capability of the received objects to be different from p); moreover, (T_+-REP) and (T_+-REP$^{\mathsf{p}}$) ensure that inputs on persistent names cannot be guarded by replicated inputs (by checking $\Gamma^{\mathsf{p}} = \emptyset$). Rules for outputs check that persistent names cannot be passed around. Finally, (T_+-PAR) ensures the linear usage of persistent names in both input and output subject (by checking the disjointness of ${\Gamma_1}^{\mathsf{p}}$ and ${\Gamma_2}^{\mathsf{p}}$ and of ${\Delta_1}^{\mathsf{p}}$ and ${\Delta_2}^{\mathsf{p}}$) and bans the usage of names with persistent capability in both input and output (by checking the disjointness of Γ^{p} and Δ^{p});

(5) is ensured in rules (T_+-INP), (T_+-REP) and (T_+-REP$^{\mathsf{p}}$), where the level of the input prefixes are compared against the level of each nested input and output.

$$(\mathrm{T}_+\text{-INP})\ \frac{\begin{array}{c} a : \mathsf{T}^{[\mathsf{u},k]} \text{ with } \mathsf{u} \neq \omega \quad b : \mathsf{T} \quad \forall c \in \mathrm{os}(P) \cup \mathrm{is}(P) : \mathrm{lev}(c) < k \\ \Gamma^{\omega} = \emptyset \quad a \text{ +-responsive implies } b \text{ not (+-)responsive} \\ \Gamma; \Delta, b^{t'} \vdash_2 P \quad t \neq \mathsf{n}, \mathsf{p} \quad t' \neq \mathsf{n}, \mathsf{p} \end{array}}{\Gamma, a^{t}; \Delta \vdash_2 a(b).P}$$

$$(\mathrm{T}_+\text{-REP})\ \frac{\begin{array}{c} a : \mathsf{T}^{[\omega,k]} \quad b : \mathsf{T} \quad \Delta^{\rho} = \Delta^{\rho^+} = \emptyset \quad \emptyset; \Delta, b^{t'} \vdash_2 P \quad t' \neq \mathsf{n}, \mathsf{p} \\ b \text{ (+-)responsive implies } \forall c \in \mathrm{os}(P) \cup \mathrm{is}(P) : \mathrm{lev}(c) < k \end{array}}{a^{-}; \Delta \vdash_2 !a(b).P}$$

$$(\mathrm{T}_+\text{-REP}^{\mathsf{p}})\ \frac{\begin{array}{c} a : \mathsf{T}^{[\rho^+,k]} \quad b : \mathsf{T} \quad \Gamma^{\ell} = \emptyset \text{ for } \ell \in \{\rho, \omega, \mathsf{s}, \mathsf{p}\} \quad \Delta^{\ell'} = \emptyset \text{ for } \ell' \in \{\mathsf{p}, \rho\} \\ \Gamma; \Delta, b^{t} \vdash_2 P \quad t \neq \mathsf{n}, \mathsf{p} \quad \forall c \in \mathrm{os}(P) \cup \mathrm{is}(P) : \mathrm{lev}(c) < k \end{array}}{\Gamma, a^{\rho}; \Delta \vdash_2 !a(b).P}$$

$$(\mathrm{T}_+\text{-OUT})\ \frac{a : \mathsf{T}^{\mathsf{U}} \quad b : \mathsf{T} \quad \Delta^{\rho} = \Delta^{\rho^+} = \emptyset \quad t' \neq \mathsf{n}, \mathsf{p} \quad t \neq \mathsf{n}, \mathsf{p}}{\emptyset; \Delta, a^{t}, b^{t'} \vdash_2 \overline{a}\langle b \rangle}$$

$$(\mathrm{T}_+\text{-OUT}^{\mathsf{p}})\ \frac{a : \mathsf{T}^{[\rho^+,k]} \quad b : \mathsf{T} \quad b \text{ not (+-)responsive} \quad \Delta^{\rho} = \Delta^{\rho^+} = \emptyset}{\emptyset; \Delta, a^{\rho}, b^{-} \vdash_2 !\overline{a}\langle b \rangle}$$

$$(\mathrm{T}_+\text{-NIL})\ \frac{\Delta^{\rho} = \Delta^{\rho^+} = \emptyset}{\emptyset; \Delta \vdash_2 \mathbf{0}} \qquad (\mathrm{T}_+\text{-RES})\ \frac{a : \mathsf{T}^{\mathsf{U}} \quad \Gamma, a^{t}; \Delta, a^{t'} \vdash_2 P}{\Gamma; \Delta \vdash_2 (\nu a)P}$$

$$(\mathrm{T}_+\text{-RES-}\bot)\ \frac{a : \bot \quad \Gamma; \Delta \vdash_2 P}{\Gamma; \Delta \vdash_2 (\nu a)P} \qquad (\mathrm{T}_+\text{-RES-I})\ \frac{a : \mathsf{I} \quad \Gamma; \Delta, a^{-} \vdash_2 P}{\Gamma; \Delta \vdash_2 (\nu a)P}$$

$$(\mathrm{T}_+\text{-WEAK-}\Gamma)\ \frac{\Gamma; \Delta \vdash_2 P}{\Gamma, a^{\mathsf{n}}; \Delta \vdash_2 P} \qquad (\mathrm{T}_+\text{-WEAK-}\Delta)\ \frac{\Gamma; \Delta \vdash_2 P}{\Gamma; \Delta, a^{\mathsf{n}} \vdash_2 P}$$

$$(\mathrm{T}_+\text{-PAR})\ \frac{\begin{array}{c} \Gamma = \Gamma_1 \cup \Gamma_2 \quad \Delta = \Delta_1 \cup \Delta_2 \quad \Gamma_i; \Delta_i \vdash_2 P_i \quad (i = 1, 2) \\ \Gamma_1^{\ell} \cap \Gamma_2^{\ell} = \emptyset \text{ for } \ell \in \{\rho, \mathsf{s}, \mathsf{p}\} \qquad \Delta_1^{\ell'} \cap \Delta_2^{\ell'} = \emptyset \text{ for } \ell' \in \{\rho, \mathsf{p}\} \\ \Gamma^{\mathsf{p}} \cap \Delta^{\rho} = \emptyset \quad \Gamma^{\mathsf{m}} \cap \Delta^{\mathsf{m}} = \emptyset \end{array}}{\Gamma; \Delta \vdash_2 P_1 | P_2}$$

Table 3: Typing rules of $\vdash_2$

Finally, linear usage of responsive names is ensured by the typing rules for replicated inputs ($\Delta^{\rho} = \Gamma^{\rho} = \emptyset$), by ($\mathrm{T}_+$-OUT) and ($\mathrm{T}_+$-NIL) ($\Gamma = \Delta^{\rho} = \emptyset$), by ($\mathrm{T}_+$-PAR) (${\Gamma_1}^{\rho} \cap {\Gamma_2}^{\rho} = \emptyset$ and ${\Delta_1}^{\rho} \cap {\Delta_2}^{\rho} = \emptyset$), by ($\mathrm{T}_+$-WEAK-$\Gamma$) and ($\mathrm{T}_+$-WEAK-$\Delta$) (only names annotated with capability n can be freely added to typing contexts) and by (T_+-INP) ($a \notin \Gamma$).

7 Subject reduction and type soundness for system $\vdash_2$

Subject reduction carries over to the new system, modulo a notational change. For Γ a typing context and V a set of names let us denote by $\Gamma \ominus^{+} V$ the typing context obtained by removing from Γ each a^{t} such that $a \in V$. Let us denote by $\mathrm{on}(P)$ the set of names occurring free in output position in P.

Theorem 4 (subject reduction for system $\vdash_2$). *$\Gamma;\Delta \vdash_2 P$ and $P \xrightarrow{[a]} P'$ imply $\Gamma';\Delta' \vdash_2 P'$, with $\Gamma' = \Gamma \ominus^+ (\{a\} \setminus \mathrm{in}(P'))$ and $\Delta' = \Delta \ominus^+ (\{a\} \setminus \mathrm{on}(P'))$.*

What follow are the analogs of Propositions 1 and 2 for system $\vdash_2$ (their proofs can be found in Appendix G). We consider the extension of $\mathrm{wt}(\cdot)$ to the system $\vdash_2$, written $\mathrm{wt}^+(\cdot)$, defined as follows.

$$\begin{array}{lll} \mathrm{wt}^+(\mathbf{0}) = 0 & \mathrm{wt}^+(!\overline{a}\langle b\rangle) = 0 & \mathrm{wt}^+(!a(b).P) = 0 \\ \mathrm{wt}^+(\overline{a}\langle b\rangle) = 0_{\mathrm{lev}(a)} & \multicolumn{2}{l}{\mathrm{wt}^+(P|R) = \mathrm{wt}^+(P) + \mathrm{wt}^+(R)} \\ \mathrm{wt}^+((\nu a)P) = \mathrm{wt}^+(P) & \multicolumn{2}{l}{\mathrm{wt}^+(a(b).P) = \mathrm{wt}^+(P) + 0_{\mathrm{lev}(a)}} \end{array}$$

Note the different clause for input $a(b).P$, where the level of a contributes to the weight of the whole process. This is necessary for guaranteeing that $\mathrm{wt}^+(\cdot)$ decreases through reductions involving replicated outputs.

Proposition 4. *$\Gamma;\Delta \vdash_2 P$ and $P \xrightarrow{\tau\langle a,b\rangle} P'$ with either a or b (+-)responsive, implies $\mathrm{wt}^+(P') \prec \mathrm{wt}^+(P)$.*

The balancing requirements are now more stringent. They include those for responsive and ω-receptive names necessary in the first system (condition 1 below). Concerning +-responsive names, "perfect balancing" between input and output is required only for those names that carry (+-)responsive names (condition 2). Moreover, the same requirements apply also to restricted +-responsive names (condition 3).

Given a set of names V let us define $V^\dagger = \{a \in V \,|\, a : \mathsf{T}$ and T is of the form $(\mathsf{S}^{[\mathsf{u},k]})^{[\mathsf{u}',h]}$ with $\mathsf{u} \in \{\rho, \rho^+\}\}$. Define $\mathrm{r}_\mathrm{i}^+(P)$ (resp. $\mathrm{r}_\mathrm{o}^+(P)$) as the set of restricted +-responsive names in P occurring in an input (resp. output) action in P, even underneath a replication. We have the following definition and results. Proofs omitted here are reported in Appendix G.

Definition 8 (strongly balanced processes). *A process P is* $(\Gamma;\Delta)$-strongly balanced *if $\Gamma;\Delta \vdash_2 P$ and the following conditions hold:*

1. $\Gamma^\rho = \Delta^\rho$ *and* $\Delta^\omega \subseteq \Gamma^\omega$;
2. $\Gamma^{\rho^+} \subseteq \Delta^{\rho^+}$ *and* $(\Delta^{\rho^+})^\dagger \subseteq (\Gamma^{\rho^+})^\dagger$;
3. $\mathrm{r}_\mathrm{i}^+(P) \subseteq \mathrm{r}_\mathrm{o}^+(P)$ *and* $(\mathrm{r}_\mathrm{o}^+(P))^\dagger \subseteq (\mathrm{r}_\mathrm{i}^+(P))^\dagger$.

Proposition 5. *Suppose P is $(\Gamma;\Delta)$-strongly balanced with $\Delta^\rho \cup \Gamma^{\rho^+} \neq \emptyset$. Then $P \xrightarrow{\tau\langle a,b\rangle}$ with either a or b (+-)responsive.*

The proof of the theorem below is non-trivial, as strong balancing is preserved through reductions only up to certain transformations on processes. The lemma below identifies such transformations.

Lemma 3. *Suppose P is $(\Gamma;\Delta)$-strongly balanced and $P \xrightarrow{\tau\langle a,b\rangle} P'$ with P' non strongly balanced. Assume $\Gamma';\Delta' \vdash_2 P'$, with Γ',Δ' as given by Theorem 4. Then for some R, R', b and $\tilde{d}$:*

1. $a \in (\Gamma'^{\rho^+} \setminus \Delta'^{\rho^+}) \cup (\mathrm{r}_\mathrm{i}^+(P') \setminus \mathrm{r}_\mathrm{o}^+(P'))$;
2. $P \equiv (\nu\tilde{d})(!a(x).R \,|\, \overline{a}\langle b\rangle \,|\, R')$ *and* $a \notin \mathrm{fn}(R, b, R')$;
3. $P' \equiv (\nu\tilde{d})(!a(x).R \,|\, R[{}^b/_x] \,|\, R')$ *and* $a \notin \mathrm{fn}(R[{}^b/_x], R')$;

4. $P'' = (\nu \tilde{d})(R[b/x] \,|\, R')$ *is strongly balanced.*

Theorem 5 (type soundness for system $\vdash_2$). *Suppose P is $(\Gamma;\Delta)$-strongly balanced and $r \in \Delta^{\rho} \cup \Gamma^{\rho^+}$. Then P guarantees responsiveness of r.*

PROOF: Suppose that $P \xrightarrow{[s]} P'$, with P' having a minimal weight among processes reachable from P with $r \notin s$ (this P' must exist by well-foundedness of $\prec$). Let $s = a_1 \cdots a_n$, and consider the sequence of reductions leading to P':

$$P = P_0 \xrightarrow{[a_1]} P_1 \xrightarrow{[a_2]} \cdots \xrightarrow{[a_n]} P_n = P' \,. \tag{8}$$

By $\Gamma;\Delta \vdash_2 P$ and subject reduction we have that $\Gamma_i;\Delta_i \vdash_2 P_i$ for $i = 0,\ldots,n$, where $\Gamma_0 = \Gamma$ and $\Delta_0 = \Delta$ and $\Gamma_i = \Gamma_{i-1} \ominus^+ (\{a_i\} \setminus \mathrm{in}(P_i))$ and $\Delta_i = \Delta_{i-1} \ominus^+ (\{a_i\} \setminus \mathrm{on}(P_i))$ for $i > 0$. We prove that $P' \overset{[r]}{\Longrightarrow}$ by induction on the number k of non-strongly balanced processes in the sequence of reductions (8), that is

$$k = \big|\{i \,|\, 0 \le i \le n \text{ and } P_i \text{ is not } (\Gamma_i,\Delta_i)\text{-strongly balanced } \}\big|.$$

$k = 0$**:** Then P' is strongly balanced. Since $r \in (\Delta_n{}^{\rho} \cup \Gamma_n{}^{\rho^+})$ (as $r \notin s$), by Proposition 5, $P' \xrightarrow{\tau\langle a,b\rangle} P''$, with either a or b (+-)responsive, and, by Proposition 4, $\mathrm{wt}^+(P'') \prec \mathrm{wt}^+(P')$. Hence $a = r$, because P' was assumed to have minimal weight among the processes reachable from P without using r as subject.

$k > 0$**:** Let P_j ($j > 0$) be the leftmost non-strongly balanced process in the sequence (8). Consider the reduction $P_{j-1} \xrightarrow{[a_j]} P_j$. Process P_{j-1} is strongly balanced while P_j is not, thus, by Lemma 3 (1, 2), $a_j \in (\Gamma_j{}^{\rho^+} \setminus \Delta_j{}^{\rho^+}) \cup (\mathrm{r_i}^+(P_j) \setminus \mathrm{r_o}^+(P_j))$ and $P_{j-1} \equiv (\nu \tilde{d})(!a_j(x).R \,|\, \overline{a}_j\langle c\rangle \,|\, S)$, with $a_j \notin \mathrm{fn}(R,c,S)$. Again by Lemma 3 (3), $P_j \equiv (\nu \tilde{d})(!a_j(x).R \,|\, R[c/x] \,|\, S)$ with $a_j \notin \mathrm{fn}(R[c/x],S)$. Moreover $P' \equiv (\nu \tilde{d'})(!a_j(x).R \,|\, P''')$ with $a_j \notin \mathrm{fn}(P''')$. Suppose for simplicity a_j free in P_j, that is $a \in (\Gamma_j{}^{\rho^+} \setminus \Delta_j{}^{\rho^+})$. Now, the process $P'_j = (\nu \tilde{d})(R[c/x] \,|\, S)$, obtained by erasing the term $!a_j(x).R$ from P_j, is strongly balanced (Lemma 3 (4)), and, as $a \notin \mathrm{fn}(R[c/x],S)$, it holds $P'_j \xrightarrow{[a_{j+1}]} \cdots \xrightarrow{[a_n]} P'_n = P''$, with $P'' \equiv (\nu \tilde{d'})\,P'''$. This sequence has $\le k-1$ unbalanced processes, and moreover P'' has minimal weight among the processes reachable from P'_j without using r as subject, because $\mathrm{wt}^+(P'') = \mathrm{wt}^+(P')$ (by definition of $\mathrm{wt}^+(\cdot)$ we have $\mathrm{wt}^+((\nu \tilde{d'})(!a_j(x).R \,|\, P''')) = \mathrm{wt}^+((\nu \tilde{d'})P''')$). Then, by induction hypothesis, $P'' \overset{[r]}{\Longrightarrow}$, which implies $P' \overset{[r]}{\Longrightarrow}$.

□

8 Examples

Let us now examine a few examples. We begin by considering processes (4)-(6), then a couple of examples useful to compare our system to type systems that guarantee lock freedom, and a recursive function. Finally, we show a more concrete example (a Web Service).

Basic examples. In what follows, unless otherwise stated, we assume that x,y are of sort inert, that a,b,c are +-responsive and that r,s are responsive. Conditions on levels are ignored when obvious.

Process (4) at the beginning of Section 6 is well-typed with c of capability multiple (m) in output and simple (s) in input; it is strongly balanced if put in parallel with an appropriate context of the form

$r(x).P$. Process (5) is well-typed with a of capability persistent (p) in output and simple (s) in input (also, P must be assumed strongly balanced, and not containing free persistent inputs or names of level greater than a's); it is strongly balanced if put in parallel with $\overline{r}\langle x\rangle$. Process (6) is well-typed assuming r_1 and r_2 of capability simple in input and n,m natural number variables (the obvious extension of the system with `if-then-else` and naturals is here assumed); again, it is strongly balanced if put in parallel with an appropriate context.

The next two examples involve non-linear usages of +-responsive names arising from replication and reference passing. We mention these examples also because they will help us to compare our system to existing type systems that enforce lock freedom, a property related to responsiveness (see the concluding section). The first example involves only replication, object parts play no role:

$$!a.\overline{b}\,|\,\overline{a}\,|\,b\,. \tag{9}$$

The above process is strongly balanced under the assumption that a has capability persistent in input and multiple in output, and b has capability simple in input and multiple in output; also, the level of b must be less than a's. In the next example, an agent "looks up" a directory a to get the address of a service b, and then calls this service:

$$!a(z).\overline{z}\langle b\rangle\,|\,(\nu r)(\overline{a}\langle r\rangle\,|\,r(w).\overline{w})\,|\,b\,. \tag{10}$$

This process is strongly balanced under the assumption that: a is persistent in input and multiple in output; b is simple in input and multiple in output; also, it must be $\mathrm{lev}(b) < \mathrm{lev}(r) < \mathrm{lev}(a)$ (the variant where the input b is replaced by $!b$ is also strongly balanced; in this case b is persistent in input.)

The type system $\vdash_2$ can be extended to the polyadic version of the calculus with naturals and variables exactly as seen in Section 5.2, i.e. by relying on the "$\lhd$" relation over actions in rules (T_+-INP), (T_+-REP) and (T_+-REP$^{\mathrm{P}}$).

Moreover, as already seen for $\vdash_1$, the results introduced in the previous section are still valid for the system $\vdash_2$ extended with summation and `if-then-else`. The proofs reported in Appendix G require some changes in the vein of those reported in Appendix E, hence are omitted. Now, consider the process implementing the factorial function in (3) and assume r,r' are (+-)responsive. It is easily seen that the process in (3) is not well-typed in the present system: in fact, because of the recursive call at f, it cannot be $\mathrm{lev}(r) < \mathrm{lev}(r')$. In general, the type system bans as ill-typed recursive calls of the form $g(h(g(i),i))$, thus ruling out the usual encoding of primitive recursion. Certain forms of recursion, like the tail-recursive version of factorial below, are however still well-typed

$$!f(n,m,r).\texttt{if}\ n=0\ \texttt{then}\ \overline{r}\langle m\rangle\ \texttt{else}\ \overline{f}\langle n-1,m*n,r\rangle\,.$$

A broker service. A broker service, available at *broker*, upon receiving from a client some travel information d and a reply channel r from a client, contacts agencies A_1 and A_2 and waits for their offers. Upon receiving a response from both agencies, the broker compares their offers and passes onto the client a link to contact the agency that made the most advantageous one; a refusal message is passed to the other agency. The client can now decide to either accept or decline the offer. In the former case, the selected agency replies to the client by sending the reservation details.

The scenario described above can be modeled as $Sys \stackrel{\triangle}{=} B\,|\,A_1\,|\,A_2\,|\,C$, where (as usual, this is an

abstract model where many details about internal computations are left out):

$$B \triangleq !broker(d,r).(\nu s,t)\Big(\overline{ag_1}\langle d,s\rangle \mid \overline{ag_2}\langle d,t\rangle \mid s(o_1,r_1).t(o_2,r_2).$$
$$\texttt{if } o_1 \preceq o_2 \texttt{ then } \overline{r}\langle o_1,r_1\rangle \mid (\nu v)(\overline{r_2}\langle \text{“}decline\text{”},v\rangle \mid v(x))$$
$$\texttt{else } \overline{r}\langle o_2,r_2\rangle \mid (\nu v)(\overline{r_1}\langle \text{“}decline\text{”},v\rangle \mid v(x))\Big)$$
$$A_i \triangleq !ag_i(d,r).(\nu s,offer)\big(\overline{r}\langle offer,s\rangle \mid s(x,t).\texttt{if } x = \text{“}accept\text{”} \texttt{ then } \overline{t}\langle \text{“}details\text{”}\rangle \texttt{ else } \overline{t}\langle \text{“}void\text{”}\rangle\big)$$
$$C \triangleq (\nu data,r)\Big(\overline{broker}\langle data,r\rangle \mid r(o,y).(\nu s)\big((\overline{y}\langle \text{“}accept\text{”},s\rangle \mid s(x).C') \oplus (\overline{y}\langle \text{“}decline\text{”},s\rangle \mid s(x))\big)\Big)$$

Sys is strongly balanced under the assumption that C' is $(\emptyset;\Delta)$-strongly balanced, for some Δ such that $\Delta^{\rho} = \emptyset$; that *broker* and ag_i are ω-receptive; “*decline*”, “*details*”, “*accept*”, “*void*”, *offer*, *data*, x,z,o_1,o_2,o and d are inert; the remaining names are (+-)responsive and the obvious requirements on levels. Therefore, a communication on s is guaranteed to eventually take place: the client is guaranteed to receive a confirmation request from a travel agency. This example emphasizes the usefulness of nested (free) inputs: without this feature, no broker could be defined, as comparison between two or more received data would be impossible.

9 Encoding the Structured Orchestration Language

ORC [4] is a recently proposed language for Web Services orchestration that supports a structured model of concurrent and distributed programming. This model assumes that basic services, performing basic sequential computations and data manipulations, are implemented by *primitive* sites, and provides constructs to orchestrate the concurrent invocation of sites to achieve a given goal. In this section we briefly introduce ORC and then show that it can be encoded into pi-calculus. Responsiveness on the target terms can be used to reason about responsiveness on the original ORC terms.

9.1 ORC: syntax and operational semantics

For the sake of simplicity, we consider a monadic version of this calculus, and we suppose that inert names, $c,c',\ldots$, are the only *data values* that can be exchanged among ORC services. We also consider a countable set of *variables* $x,y,\ldots$. ORC terms, ranged over $f,g,\ldots$, are defined by the following grammar:

p	::=	x	*Variable*
		$\mid c$	*Value*
f,g	::=	$\mathbf{0}$	*Inaction*
		$\mid M(p)$	*Site call*
		$\mid E(p)$	*Expression call*
		$\mid \texttt{let}(p)$	*Publication*
		$\mid f > x > g$	*Sequential composition*
		$\mid f \mid g$	*Symmetric parallel composition*
		$\mid g \texttt{ where } x :\in f$	*Asymmetric parallel composition*

$$(\text{PUB})\ \frac{}{\texttt{let}(c) \overset{!c}{\hookrightarrow} \mathbf{0}} \qquad (\text{SITE})\ \frac{}{M(c) \overset{\tau}{\hookrightarrow} \texttt{let}(F_M(c))}$$

$$(\text{PAR1})\ \frac{f \overset{\lambda}{\hookrightarrow} f'}{f \mid g \overset{\lambda}{\hookrightarrow} f' \mid g} \qquad (\text{PAR2})\ \frac{g \overset{\lambda}{\hookrightarrow} g'}{f \mid g \overset{\lambda}{\hookrightarrow} f \mid g'}$$

$$(\text{SEQ1})\ \frac{f \overset{\lambda}{\hookrightarrow} f' \quad \lambda \neq !c}{f > x > g \overset{\lambda}{\hookrightarrow} f' > x > g} \qquad (\text{SEQ2})\ \frac{f \overset{!c}{\hookrightarrow} f'}{f > x > g \overset{\tau}{\hookrightarrow} (f' > x > g) \mid g[c/x]}$$

$$(\text{WH1})\ \frac{f \overset{\lambda}{\hookrightarrow} f' \quad \lambda \neq !c}{g\ \texttt{where}\ x :\in f \overset{\lambda}{\hookrightarrow} g\ \texttt{where}\ x :\in f'} \qquad (\text{WH2})\ \frac{f \overset{!c}{\hookrightarrow} f'}{g\ \texttt{where}\ x :\in f \overset{\tau}{\hookrightarrow} g[c/x]}$$

$$(\text{WH3})\ \frac{g \overset{\lambda}{\hookrightarrow} g'}{g\ \texttt{where}\ x :\in f \overset{\lambda}{\hookrightarrow} g'\ \texttt{where}\ x :\in f} \qquad (\text{DEF})\ \frac{E(x) \overset{\triangle}{=} f}{E(p) \overset{\tau}{\hookrightarrow} f[p/x]}$$

where in (SITE) $F_M(c)$ is any function on data values.

Table 4: ORC operational semantics.

In the syntax, M is a primitive *site* name, p is a parameter (either a variable x or a name c) and for every expression name E there exists a declaration $E(x) \overset{\triangle}{=} f$, where x is the formal parameter and $\text{fv}(f) \subseteq \{x\}$. The primitives can be informally explained as follows. Each closed expression f *publishes* (returns) a (finite or infinite) sequence of zero or more values. A site call $M(c)$ always publishes a predefined value $F_M(c)$, where $F_M(\cdot)$ is the function associated with site M. An expression call $E(c)$ publishes the values returned by $f[c/x]$ if $E(x) \overset{\triangle}{=} f$. The expression $\texttt{let}(c)$ publishes the value c. In $f > x > g$, the execution of f is started, and every value c published by f triggers a new instance of g, $g[c/x]$; the sequence of values produced by all these instances of g running in parallel is published. In the following, $f >> g$ abbreviates $f > x > g$ when $x \notin \text{fv}(g)$. In $f|g$ a sequence obtained by interleaving values produced by f and g is published. In $g\ \texttt{where}\ x :\in f$ the values produced by g are published; however, the execution of f and g is started in parallel, and each subterm of g that depends on x is blocked until f produces the first value c, which causes x to be replaced everywhere by c; subsequent values published by f are discarded. The operational semantics is formally defined in Table 4. Labels, λ, λ', range over published values, written $!c$, and synchronizations, τ. We write $f \overset{!c}{\Rightarrow}$ if $f \overset{\tau}{\hookrightarrow}^* \overset{!c}{\hookrightarrow}$, that is if f publishes the value c possibly after some internal reductions.

9.2 Encoding

ORC terms are translated into pi-calculus by the function $[\![\cdot]\!]_s$, where s is a chosen "result channel", defined as follows

name	c, y	x	s	p	r	t	E	M
Type	I	$\mathsf{I}^{[\rho^+,k_x]}$	$\mathsf{I}^{[\rho^+,k_s]}$	$\mathsf{I}^{[\rho^+,k_p]}$	$\mathsf{I}^{[\rho^+,k_r]}$	$\mathsf{I}^{[\rho^+,k_t]}$	$(\mathsf{I},\mathsf{I}^{[\rho^+,k_s]})^{[\rho^+,k_E]}$	$(\mathsf{I},\mathsf{I}^{[\rho^+,k_r]})^{[\rho^+,k_M]}$
I-Cap.		m	s or p	s	s	p	p	p
O-Cap.	–	p	m	m	m	m	m	m
with: $k_x > k_s, k_p, k_E, k_M, \quad k_E > k_s, \quad k_M > k_p, \quad k_p > k_s, \quad k_r, k_t > k_s, k_x$								

Table 5: Typing assumptions.

$$
\begin{array}{ll}
[\![\mathtt{let}(x)]\!]_s = x(y).\overline{s}\langle y\rangle & [\![\mathtt{let}(c)]\!]_s = \overline{s}\langle c\rangle \\
[\![E(x)]\!]_s = x(y).\overline{E}\langle y,s\rangle & [\![E(c)]\!]_s = \overline{E}\langle c,s\rangle \\
[\![M(x)]\!]_s = x(y).(\nu p)(\overline{M}\langle y,p\rangle \mid p(z).\overline{s}\langle z\rangle) & [\![M(c)]\!]_s = (\nu p)(\overline{M}\langle c,p\rangle \mid p(y).\overline{s}\langle y\rangle) \\
[\![f \mid g]\!]_s = [\![f]\!]_s \mid [\![g]\!]_s & [\![f > x > g]\!]_s = (\nu t)\big([\![f]\!]_t \mid !t(y).(\nu x)(!\overline{x}\langle y\rangle \mid [\![g]\!]_s)\big)
\end{array}
$$

$$
[\![g\ \mathtt{where}\, x :\in f]\!]_s = (\nu r)\big([\![f]\!]_r \mid (\nu x)(r(y).!\overline{x}\langle y\rangle \mid [\![g]\!]_s)\big)\ .
$$

Encoding of a declaration $E(x) \stackrel{\triangle}{=} f$ is given by $!E(x,s).[\![f]\!]_s$. The encoding of the site M is $!M(x,s).\overline{s}\langle F_M(c)\rangle$. The encodings of $\mathtt{let}(p)$, $E(p)$ and $M(p)$ for $p = c$ correspond to outputting c on the result channel s and invoking expression E and site M with parameters c and s, respectively. When $p = x$, it is first necessary to retrieve the content of variable x (by reading on it) before proceeding by either outputting, calling E or calling M. The encoding of the parallel composition of two terms corresponds to the parallel composition of both encodings. The remaining two cases are more interesting. In $[\![f > x > g]\!]_s$ the execution of $[\![f]\!]_t$ is started and each published value is sent on t. For each of these values a new copy of $[\![g]\!]_s$ is started with a new "local variable" x containing such a value. In the case $[\![g\,\mathtt{where}\, x :\in f]\!]_s$, the executions of f and g are started in parallel, and only the first value published by f is considered (thanks to the non-replicated input on r). Note that the first publication of f does not stop f's execution, which does not interfere with the execution of g because the name r is no longer available.

The encoded terms are well typed if the typing assumptions in Table 5 can be enforced. Levels are left unspecified, but some constraints on the values they can assume are given on the bottom part of the table.

The following result can be used for reasoning about responsiveness of ORC expressions. More precisely, the theorem below ensures that each well-typed process, encoding of an ORC term – and site and expression it needs –, always publish at least one value. The proof is reported in Appendix H. In what follows, given an ORC term f, D_f stands for the parallel composition of the encodings of all declarations and sites involved in the definition of f, and $\tilde{d} = \mathrm{fn}(D_f)$.

Theorem 6. *Let f be a closed* ORC *term and suppose D_f is well typed. Under the typing assumptions of Table 5, $[\![f]\!]_s$ is well-typed and $F \stackrel{\triangle}{=} (\nu\tilde{d})([\![f]\!]_s \mid D_f \mid !s(x).\mathbf{0})$, with s and $\tilde{d}$ +-responsive, is strongly balanced. Moreover, $f \stackrel{!c}{\Rightarrow}$ if and only if $F \stackrel{\tau\langle s,c\rangle}{\Longrightarrow}$.*

Note that, as already discussed in the previous section, some recursive functions can be typed by extending $\vdash_2$ by considering "$\lhd$" in place of "$<$", as already seen for system $\vdash_1$ in Section 5.2. Hence some recursive expressions, more precisely the tail-recursive ones, can be handled. Specifically, for each encoding of expression $!E(x,s).[\![f]\!]_s$ the level associated to channel name E is deduced by forcing the

constraints identified in Table 5. E.g. if $E(x) \triangleq \texttt{let}(x)|E(x-1)$, where x is supposed to be an integer value, then $[\![E(x)]\!] = !E(x,s).\big(\overline{s}\langle x\rangle \,|\, \overline{E}\langle x-1,s\rangle\big)$ is well-typed under the assumption that $\text{lev}(E) > \text{lev}(s)$ and s is +-responsive. For another example of ORC tail-recursive function, which can be encoded into a well-typed term, consider *MN* defined in the following example.

Example 5. The following ORC expressions are taken from [4]. Consider two primitive sites, *CNN* and *BBC* that, when invoked with null argument, reply by publishing a piece of news. Consider also a site $Mail(m,a)$, which receives a message m and an e-mail address a, and notifies (publishes an uninteresting null value) after sending m to a. The ORC function below emails the first n pieces of news received from either *CNN* or *BBC* to address a, and publishes the current value of n after every sending and upon termination:

$$\begin{aligned} MN(n,a) \triangleq\ & \texttt{if } n=0 \ \texttt{ then } \ \texttt{let}(n) \\ & \texttt{else } \ (Mail(t,a) \gg \texttt{let}(n)) \ \texttt{ where } \ t :\in (CNN|BBC) \\ & \qquad\qquad | \ MN(n-1,a)\ . \end{aligned}$$

Consider the extension of the calculus with natural values, Nat, polyadic communication and an `if-then-else` construct. Suppose the encodings of sites *CNN*, *BBC* and *Mail* are, respectively, $!CNN(x).(\nu n)\overline{x}\langle n\rangle$, $!BBC(x).(\nu n')\overline{x}\langle n'\rangle$ and $!Mail(x,a,r).(\overline{a}\langle x\rangle\,|\,\overline{r})$, where n and n' represent pieces of news. Suppose that s is the result channel, the function *MN* is encoded as follows:

$$\begin{aligned} [\![MN(n,a)]\!]_s \triangleq\ & !Mn(n,a,s).\texttt{if } n=0 \texttt{ then } \overline{s}\langle n\rangle \\ & \texttt{else } \Big((\nu r)\Big(\overline{CNN}\langle r\rangle \,|\, \overline{BBC}\langle r\rangle \,|\, (\nu t)\big(r(y).!\overline{t}\langle y\rangle \,|\, (\nu r')(t(x).\overline{Mail}\langle x,a,r'\rangle \\ & \qquad\qquad | \, !r'(x).\overline{s}\langle n\rangle)\big)\Big) \,|\, \overline{Mn}\langle n-1,a,s\rangle\Big) \end{aligned}$$

$[\![MN(n,a)]\!]_s$ is well-typed assuming s, r, r' and t +-responsive, $\text{lev}(Mn) > \text{lev}(CNN)$, $\text{lev}(Mn) > \text{lev}(BBC)$ and $\text{lev}(CNN), \text{lev}(BBC) > \text{lev}(r) > \text{lev}(t) > \text{lev}(Mail) > \text{lev}(r') > \text{lev}(s)$.

Example 6. Not all ORC terms are encodable into well-typed processes. Consider the term $f = Inc(0)$, where the expression *Inc* is recursively defined as $Inc(n) \triangleq Succ(n) > m > Inc(m)$ and *Succ* is the successor function $Succ(n) \triangleq n+1$. The term F below is not well-typed.

$$\begin{aligned} F &\triangleq (\nu Succ, Inc)([\![f]\!]_s \,|\, !s(x).\mathbf{0} \,|\, D_f) \\ [\![f]\!]_s &\triangleq \overline{Inc}\langle 0,s\rangle \\ D_f &\triangleq\ !Succ(n,s).\overline{s}\langle n+1\rangle \\ &\qquad | \, !Inc(n,r).(\nu s)\big(\overline{Succ}\langle n,s\rangle \,|\, !s(m).(\nu w)(!\overline{w}\langle m\rangle \,|\, w(o).\overline{Inc}\langle o,r\rangle)\big) \end{aligned}$$

In fact, $!Inc(n,r).(\nu s)\big(\overline{Succ}\langle n,s\rangle \,|\, !s(m).(\nu w)(!\overline{w}\langle m\rangle \,|\, w(o).\overline{Inc}\langle o,r\rangle)\big)$ is not well-typed (not $\overline{Inc}\langle o,r\rangle \triangleleft Inc(n,r)$) and the premise of Theorem 6 are not satisfied.

10 Conclusions and related works

We have presented two type systems for statically enforcing responsive usage of names in pi-calculus. The first system combines linearity, receptiveness and techniques for deadlock and livelock avoidance.

In the second system, receptiveness and linearity are relaxed at the price of stronger requirements on levels and balancing: we lose some expressive power in terms of encodable recursive functions, but are able to type interesting processes, such as translations of ORC terms. Both systems are syntax driven, so that type checking should be straightforward and efficient to implement. Extensions with type inference and subtyping deserve further investigation, mainly due to the presence of levels. Implementation of the type checking algorithm and the study of its complexity are left as future work.

Beside the works, already discussed, on receptiveness [13] and termination [5], there are a few more works related to ours and that are discussed below.

Closely related to our system $\vdash_1$ are a series of papers by Berger, Honda and Yoshida on linearity-based type systems. In [17], they introduce a type system that guarantees termination and determinacy of pi-calculus processes, i.e. *Strong Normalization* (SN). Our techniques of system $\vdash_1$ are actually close to theirs, as far as the linearity conditions and cycle-detection graphs are concerned (see also the type system in [15]). However SN is stronger than responsiveness, in particular SN implies responsiveness on all linear names under a balancing condition. In fact, the system in [17] is stricter than our system $\vdash_1$, e.g. it does not allow linear subjects to carry linear objects, and bans ω-names, hence any form of nondeterminism and divergence, as these features would obviously violate SN. Yoshida's type system in [16], in turn a refinement of the systems in [17] and [2], is meant to ensure a *Linear Liveness* property, by which processes eventually prompt for a free output at a given channel. This property is related to responsiveness, the difference being that Linear Liveness does not imply synchronization, hence the corresponding input might not become available. Two kinds of names are considered in [16]: linear (used exactly once) and *affine* (used at most once). Linear subjects carrying linear objects are forbidden and internal mobility is assumed – only restricted names can be passed around.

Closely related to our system $\vdash_2$ are a series of papers by Kobayashi and collaborators. A type system for linearity in the pi-calculus was first introduced in [10]. This system can be used to ensure that any linear name in a process occurs exactly once in input and once in output; however, it cannot ensure that a linear name will be eventually used as a subject of a synchronization. Kobayashi's type systems in [6, 7] can be used to guarantee that, under suitable fairness assumptions, certain actions are lock free, i.e. are deemed to succeed in synchronization, if they become available ([8] is a further refinement, but the resulting system cannot be used to enforce responsiveness.) Channel types are defined in terms of *usages*: roughly, CCS-like expressions on the alphabet $\{I, O\}$, that define the order in which each channel must be used in input (I) and in output (O). Each I/O action is annotated with an *obligation* level, related to when the action must become available, and a *capability* level, related to when the action must succeed in synchronization if it becomes available. A level can be a natural number or infinity, the latter used to annotate actions that are not guaranteed to become available/succeed in synchronization. This scheme is fairly general, allowing e.g. for typing of shared-memory structures such as locks and semaphores, which are outside the scope of our systems. Concerning responsiveness, on the other hand, it appears that our $+$-responsive types cannot in general be encoded into lock-freedom types. More precisely, one can exhibit processes well-typed in our system two and containing $+$-responsive names that cannot be assigned a finite capability in Kobayashi's systems. For example, both the process (9) and the "service-lookup" (10) are well-typed (in fact, strongly balanced) in our system two, under a typing context where b is $+$-responsive. They are not in the systems of [6, 7], under any type context that assigns to b a finite capability: the reason is that in these systems a finite-capability input on b is required to be balanced by an instance of a finite-obligation output $\overline{b}$, that cannot be statically determined in the given processes[1]. Another difference from [6, 7] is that these systems partly rely on a form of dynamic analysis: the

[1] In the latest version of Kobayashi's TyPiCal tool [9], released after the publication of [1], these examples are handled, though.

reliability condition on usages, which roughly plays the same role played in our systems by balancing, is checked via a reduction to the reachability problem for Petri Nets. As previously noted, our systems are entirely static.

Acknowledgments We wish to thank Davide Sangiorgi and Naoki Kobayashi for stimulating discussions on the topics of the paper.

References

[1] L. Acciai and M. Boreale. Responsiveness in process calculi. In *Proc. of ASIAN*, 2006. *LNCS* 4435:136–150, 2007.

[2] M. Berger, K. Honda and N. Yoshida. Sequentiality and the π-calculus. In *Proc. of TCLA*, 2001. *LNCS* 2044:29–45, 2001.

[3] M. Boreale. On the Expressiveness of Internal Mobility in Name-Passing Calculi. *Theoretical Computer Science* 195(2):205–226, 1998.

[4] W. R. Cook and J. Misra. Computation Orchestration: A Basis for Wide-Area Computing. *Software and Systems Modeling* 6(1):83–110, 2007.

[5] Y. Deng and D. Sangiorgi. Ensuring Termination by Typability. In *Proc. of IFIP TCS*, pp.619–632, 2004. Full version in *Information and Computation* 204(7):1045–1082, 2006.

[6] N. Kobayashi. A type system for lock-free processes. *Information and Computation* 177(2):122–159, 2002.

[7] N. Kobayashi. Type-Based Information Flow Analysis for the Pi-Calculus. *Acta Informartica* 42(4-5): 291–347, 2005.

[8] N. Kobayashi. A New Type System for deadlock-Free Processes. In *Proc. of CONCUR*, 2006. *LNCS* 4137:233–247.

[9] N. Kobayashi. The `TyPiCal` tool, available at `http://www.kb.ecei.tohoku.ac.jp/~koba/typical/`.

[10] N. Kobayashi, B.C. Pierce and D.N. Turner. Linearity and the Pi-Calculus. In *Proc. of POPL*, 1996. Full version in *ACM Transactions on Programming Languages and Systems* 21(5):914–947, 1999.

[11] M. Merro and D. Sangiorgi. On asynchrony in name-passing calculi. In *Proc. of ICALP*, 1998. *LNCS* 1443:856–867. Full version in *Mathematical Structures in Computer Science* 14(5):715–767, 2004.

[12] R. Milner. The polyadic π-calculus: a tutorial. Tec.Rep. LFCS report ECS-LFCS-91-180, 1991. Also in *Logic and Algebra of Specification*, Springer-Verlag, pp.203–246, 1993.

[13] D. Sangiorgi. The name discipline of uniform receptiveness. In *Proc. of ICALP*, 1997. *TCS* 221(1–2):457-493, 1999.

[14] D. Sangiorgi and D. Walker. *The π-calculus: a Theory of Mobile Processes.* Cambridge University Press, 2001.

[15] N. Yoshida. Graph Types for Monadic Mobile Processes. In *Proc. of 16th FST/TCS*, 1996. *LNCS* 1180:371–386, 1996.

[16] N. Yoshida. Type-Based Liveness in the Presence of Nontermination and Nondeterminism. MCS Technical Report, 2002-20, University of Leicester, 2002.

[17] N. Yoshida, M. Berger and K. Honda. Strong Normalisation in the π-calculus. In *Proc. of LICS*, 2001. *IEEE*, pp.311–322. Also in *Information and Computation* 191(2):145–202, 2004.

A Proof of Lemma 1

In this section we prove that for each typing derivation in $\vdash_1$ there exists a syntax driven one with the same conclusion.

Lemma A.1. *Suppose $\Gamma;\Delta \vdash_1 P$ has been derived by a normal derivation. Then there exists R in normal form such that $R \equiv P$ and $\Gamma;\Delta \vdash_1 R$ is obtained by a normal derivation.*

PROOF: The proof proceeds by induction on the derivation of $\Gamma;\Delta \vdash_1 P$ and by distinguishing the last typing rule applied.

(T-NIL), (T-OUT), (T-INP), (T-REP), (T-PAR): there is nothing to prove, the process is already in normal form;

(T-RES-$\bot$): suppose $P = (\nu a)R$, by the premise of the rule we get $\Gamma;\Delta \vdash_1 R$. By applying the inductive hypothesis to R we get that there exists a Q in normal form such that $Q \equiv R$ and $\Gamma;\Delta \vdash_1 Q$ has been derived by a normal derivation. Given that it must be $a \notin \mathrm{fn}(R)$, we get $Q \equiv (\nu a)R$, hence the result.

(T-RES): suppose $P = (\nu a)R$, by the premise of the rule we get $\Gamma, a;\Delta, a \vdash_1 R$. The proof proceeds by applying the inductive hypothesis to R followed by an application of (T-RES) (note that in this case it must be $a \in \mathrm{fn}(R)$);

(T-RES-I): suppose $P = (\nu a)R$. The proof proceeds as already seen either for (T-RES) or for (T-RES-$\bot$), depending on $a \in \mathrm{fn}(R)$ or not;

(T-STR): by $\Gamma;\Delta \vdash_1 P$ and the premise of the rule we get $P \equiv Q$ and $\Gamma;\Delta \vdash_1 Q$. By definition of normal derivation, $\Gamma;\Delta \vdash_1 Q$ is derived by applying a normal derivation and Q is in normal form.

□

Lemma A.2. *If $P = (\nu a_1)\cdots(\nu a_n)\big(P_1 \mid \cdots \mid P_m\big)$ is in normal form and $Q \equiv P$ with Q in normal form then there exist permutations $i_1, \cdots, i_n$ and $j_1, \cdots, j_m$ such that $Q = (\nu a_{i_1})\cdots(\nu a_{i_n})\big(P_{j_1} \mid \cdots \mid P_{j_m}\big)$.*

PROOF: The proof is straightforward by induction on the derivation of $Q \equiv P$. □

Lemma A.3. *Suppose P is in normal form and $\Gamma;\Delta \vdash_1 P$ has been derived by a normal derivation. Then $Q \equiv P$ and Q in normal form imply that there is a normal derivation of $\Gamma;\Delta \vdash_1 Q$.*

PROOF: By Lemma A.2, we have

$$\begin{aligned} P &= (\nu a_1)\cdots(\nu a_n)(P_1 \mid \cdots \mid P_m) \\ Q &= (\nu a_{i_1})\cdots(\nu a_{i_n})(P_{j_1} \mid \cdots \mid P_{j_m})\,. \end{aligned}$$

Assume, for the sake of simplicity, that none of the a_i's is of type $\bot$ or I.

In the normal derivation of $\Gamma;\Delta \vdash_1 P$, rule (T-RES) has been applied in the last n steps, preceded by an application of (T-PAR):

$$\dfrac{\dfrac{\dfrac{\dfrac{\cdots}{\Gamma, a_1, \cdots, a_n : \Delta, a_1, \cdots, a_n \vdash_1 P_1 \mid \cdots \mid P_m}\,(\text{T-PAR})}{\Gamma, a_1, \cdots, a_{n-1} : \Delta, a_1, \cdots, a_{n-1} \vdash_1 (\nu a_n)(P_1 \mid \cdots \mid P_m)}\,(\text{T-RES})}{\vdots}\,(\text{T-RES})}{\Gamma;\Delta \vdash_1 P}\,(\text{T-RES})$$

By the premise of (T-PAR) we get: $\Gamma, a_1, \cdots, a_n = \bigcup_{i\in I} \Gamma_i$ and $\Delta, a_1, \cdots, a_n = \bigcup_{i\in I} \Delta_i$ and $\Gamma_i; \Delta_i \vdash_1 P_i$ for each i. Moreover, the disjointness constraints are satisfied and the dependency graph is acyclic. Note also that $\Gamma_i; \Delta_i \vdash_1 P_i$ is derived by a normal derivation and P_i is prime for each i.

Therefore, rule (T-PAR) can be applied for deducing, $\Gamma, a_1, \cdots, a_n; \Delta, a_1, \cdots, a_n \vdash_1 P_{j_1} \mid \cdots \mid P_{j_m}$ and by n applications of (T-RES), in order to bind $a_{i_n}, \cdots, a_{i_1}$ in this order, we obtain a normal derivation of $\Gamma; \Delta \vdash_1 Q$. □

Lemma A.4 (Lemma 1). *Suppose* $\Gamma; \Delta \vdash_1 P$*, then there exists a normal derivation of* $\Gamma; \Delta \vdash_1 P$.

PROOF: The proof proceeds by induction on the derivation of $\Gamma; \Delta \vdash_1 P$. The most interesting case is when rule (T-STR) is the last applied. By $\Gamma; \Delta \vdash_1 P$ and the premise of the rule, we get $P \equiv Q$ and $\Gamma; \Delta \vdash_1 Q$.

By applying the inductive hypothesis to Q, a normal derivation of $\Gamma; \Delta \vdash_1 Q$ exists.

Suppose Q is not in normal form. By Lemma A.1, there is R in normal form such that $R \equiv Q$ and a normal derivation of $\Gamma; \Delta \vdash_1 R$ exists. By the transitivity of $\equiv$ and $R \equiv Q$, we get $R \equiv P$. The proof proceeds by distinguishing two cases. If P is not in normal form then rule (T-STR) can be applied with premise $P \equiv R$ and $\Gamma; \Delta \vdash_1 R$ for deducing $\Gamma; \Delta \vdash_1 P$ with a normal derivation. If P is in normal form, Lemma A.3 can be applied to obtain a normal derivation of $\Gamma; \Delta \vdash_1 P$.

In case Q is in normal form the proof proceeds similarly. □

B Proof of Theorem 1

As usual a preliminary result on substitutions is needed.

Proposition B.1 (substitution). *Suppose* $\Gamma; \Delta, x \vdash_1 P$ *with* $x \notin \mathrm{in}(P)$, $x, b : T$ *and* $b \notin \Gamma$ *then*

1. $b \notin \Delta$ *implies* $\Gamma; \Delta, b \vdash_1 P[b/x]$*;*
2. $b \in \Delta$ *and* b *is either an* ω*-receptive or inert name imply* $\Gamma; \Delta \vdash_1 P[b/x]$.

PROOF: In both cases the proof proceeds by induction on the derivation of $\Gamma; \Delta, x \vdash_1 P$.

1. Consider the last typing rule applied in the derivation. The interesting case is (T-OUT), in the other cases the proof proceeds by applying the inductive hypothesis. In particular, concerning rule (T-PAR), $b \notin \Gamma \cup \Delta$ in the premise ensures that acyclicity of the graph and disjointness of Δ_i^{ρ}, for $i = 1, \ldots, n$, are preserved.

   (T-OUT) by $\emptyset; \Delta, x \vdash_1 P = \overline{a}\langle c \rangle$ and the premise of the rule, we get $a : \mathsf{S}^{\cup}$, $c : \mathsf{S}$ and $(\Delta, x)^{\rho} \ominus \{a, c\} = \emptyset$. We distinguish the following cases:

   $a, c \neq x$**:** $(\Delta, b)^{\rho} \ominus \{a, c\} = \emptyset$;
   $a = x$**:** $\overline{x}\langle c \rangle[b/x] = \overline{b}\langle c \rangle$, $\mathsf{T} = \mathsf{S}^{\cup}$ and $(\Delta, b)^{\rho} \ominus \{b, c\} = \emptyset$;
   $c = x$**:** $\overline{a}\langle x \rangle[b/x] = \overline{a}\langle b \rangle$, $\mathsf{T} = \mathsf{S}$ and $(\Delta, b)^{\rho} \ominus \{a, b\} = \emptyset$;
   in each case, by (T-OUT), $\emptyset; \Delta, b \vdash_1 \overline{a}\langle c \rangle[b/x]$.
   Note that it cannot be the case that $a = c = x$ because recursive types are not allowed.

2. The result follows by a straightforward induction on typing rules. Recall that rule (T-PAR) does not impose linearity on the usage of ω-receptive and inert names in output.

□

The following lemma ensures that structural congruent processes have the same behavior. Note that the presence of (T-STR) spares us from introducing a subject congruence proposition.

Lemma B.1. *If $P \equiv R$ and $P \xrightarrow{\mu} P'$ then $R \xrightarrow{\mu} R'$ and $P' \equiv R'$.*

PROOF: The proof is straightforward by induction on the derivation of $P \equiv R$. □

The following proposition represents the analog of subject reduction for visible transitions.

Proposition B.2. *Suppose $\Gamma;\Delta \vdash_1 P$.*

1. *Whenever $P \xrightarrow{a(b)} P'$, with $a : T^{\cup}$ and $b : T$ then*
   (a) *if $b \notin \Delta$ then $\Gamma \ominus \{a\};\Delta,b \vdash_1 P'$;*
   (b) *if $b \in \Delta$ and either $T = S^{[\omega,k]}$ or $T = I$ then $\Gamma \ominus \{a\};\Delta \vdash_1 P'$;*
2. *Whenever $P \xrightarrow{\overline{a}\langle b\rangle} P'$ then $\Gamma;\Delta \ominus \{a,b\} \vdash_1 P'$;*
3. *Whenever $P \xrightarrow{\overline{a}(b)} P'$ then*
   (a) *either $b : T^{\cup}$ and $\Gamma,b;(\Delta,b) \ominus \{a,b\} \vdash_1 P'$*
   (b) *or $b : I$ and $\Gamma;(\Delta,b) \ominus \{a,b\} \vdash_1 P'$.*

PROOF:

1. By induction on the derivation of $\Gamma;\Delta \vdash_1 P$; the proof proceeds by distinguishing the last typing rule applied:

   (T-NIL), (T-OUT): it cannot be the case that $P \xrightarrow{a(b)}$;

   (T-STR): the proof proceeds by applying the induction hypothesis, Lemma B.1 and (T-STR);

   (T-INP): In this case $P = a(x).Q$ and by well-formedness of P, $x \notin \text{in}(Q)$. Moreover, by $\Gamma;\Delta \vdash_1 a(x).Q$ and the premise of the rule, we get $\Gamma = \{a\}$, $a : \mathsf{T}^{\cup}$, $x : \mathsf{T}$ and $\emptyset;\Delta,x \vdash_1 Q$. By hypothesis $P = a(x).Q \xrightarrow{a(b)} Q[b/x] = P'$ and $b : \mathsf{T}$, therefore
   (a): if $b \notin \Delta$, by Proposition B.1 (1) (substitution), it follows that $\emptyset;\Delta,b \vdash_1 Q[b/x]$;
   (b): if $b : \mathsf{T}$ with either $\mathsf{T} = \mathsf{S}^{[\omega,k]}$ or $\mathsf{T} = \mathsf{I}$ and $b \in \Delta$, by Proposition B.1 (2) (substitution), it follows that $\emptyset;\Delta \vdash_1 Q[b/x]$;

   (T-REP): the proof proceeds as already seen for the previous case;

   (T-RES): by $\Gamma;\Delta \vdash_1 P = (\nu c)Q$ and the premises of the rule, we get $\Gamma,c;\Delta,c \vdash_1 Q$. By $P \xrightarrow{a(b)} P'$ and (RES), we get $a,b \neq c$ and $Q \xrightarrow{a(b)} Q'$, with $P' = (\nu c)Q'$. By applying the inductive hypothesis to Q, we get either $\Gamma \ominus \{a\},c;\Delta,b,c \vdash_1 Q'$, if $b \notin \Delta$, or $\Gamma \ominus \{a\},c;\Delta,c \vdash_1 Q'$, if $b \in \Delta$ and b is either an inert or ω-receptive name. Therefore, by (T-RES), we get either $\Gamma \ominus \{a\};\Delta,b \vdash_1 P'$, if $b \notin \Delta$, or $\Gamma \ominus \{a\};\Delta \vdash_1 P'$, if $b \in \Delta$ and b is either an inert or ω-receptive name;

   (T-RES-I), (T-RES-⊥): the proof proceeds as already seen for the previous case;

(T-PAR): by $\Gamma;\Delta \vdash_1 P$ and the premise of the rule, we get $P = P_1 | \cdots | P_n$ with P_i prime, $\Gamma_i;\Delta_i \vdash_1 P_i$ for each i, $\Gamma = \bigcup_i \Gamma_i$ and $\Delta = \bigcup_i \Delta_i$. Moreover, $\Gamma_i^\rho \cap \Gamma_j^\rho = \Delta_i^\rho \cap \Delta_j^\rho = \emptyset$, for each $i \neq j$, and $\mathrm{DG}(\Gamma_i^\rho, \Delta_i^\rho)_{i=1,\cdots,n}$ is acyclic.

$P \xrightarrow{a(b)} P'$ means that there is a $j \in \{1,\cdots,n\}$ such that $P_j \xrightarrow{a(b)} P'_j$.

The induction hypothesis can be applied to P_j: either $\Gamma_j \ominus \{a\};\Delta_j, b \vdash_1 P'_j$, if $b \notin \Delta_j$, or $\Gamma_j \ominus \{a\};\Delta_j \vdash_1 P'_j$ if $b \in \Delta_j$ and b is either an inert or ω-receptive name.

Suppose $b \notin \Delta$, hence $b \notin \Delta_i$ for each i.

In this case $\Gamma_j \ominus \{a\};\Delta_j, b \vdash_1 P'_j$ holds. If P'_j is prime, rule (T-PAR) can be applied for deducing $\Gamma \ominus \{a\};\Delta, b \vdash_1 P'$. (Note that in this case $b \notin \Delta$ guarantees that acyclicity of the graph and disjointness of Δ_i^ρ are preserved.) If P'_j is not prime then an equivalent normal form exists (Lemma 2): $P'_j \equiv (\nu \tilde{d})(Q_1 | \cdots | Q_m)$ with Q_i prime for each i. Suppose for simplicity the $\tilde{d}$ are all ω-receptive or responsive names. By $\Gamma_j \ominus \{a\};\Delta_j, b \vdash_1 P'_j$ and (T-RES), we get $\Gamma_j \ominus \{a\}, \tilde{d};\Delta_j, b, \tilde{d} \vdash_1 Q_1 | \cdots | Q_m$. Hence, by (T-PAR), each Q_i is well-typed, the dependency graph is acyclic and the responsive parts of Q_i's input and output contexts are disjoint. Therefore, $\Gamma \ominus \{a\}, \tilde{d};\Delta, b, \tilde{d} \vdash_1 P_1 | \cdots | P_{j-1} | Q_1 | \cdots | Q_m | P_{j+1} | \cdots | P_n$ can be inferred (by applying (T-PAR)) and, by (T-RES), $\Gamma \ominus \{a\};\Delta, b \vdash_1 (\nu \tilde{d})(P_1 | \cdots | P_{j-1} | Q_1 | \cdots | Q_m | P_{j+1} | \cdots | P_n) \equiv P'$, therefore, by (T-STR), $\Gamma \ominus \{a\};\Delta, b \vdash_1 P'$.

Suppose now that $b \in \Delta$ and either b is an inert or ω-receptive name. If $b \in \Delta_j$ then $\Gamma \ominus \{a\};\Delta \vdash_1 P'_j$ else $\Gamma \ominus \{a\};\Delta, b \vdash_1 P'_j$. In both cases, the proof proceeds as already seen in the previous case. Note that acyclicity of the graph and disjointness of Δ_i^ρ is guaranteed because b is either of type inert or ω-receptive.

2. By induction on the derivation of $\Gamma;\Delta \vdash_1 P$, the proof proceeds by distinguishing the last typing rule applied:

(T-NIL), (T-INP), (T-REP): it cannot be the case that $P \xrightarrow{\overline{a}\langle b\rangle}$;

(T-OUT): by $\emptyset;\Delta \vdash_1 \overline{a}\langle b\rangle$ and the premise of (T-OUT) we get $(\Delta \ominus \{a,b\})^\rho = \emptyset$. Moreover, by (OUT), $P = \overline{a}\langle b\rangle \xrightarrow{\overline{a}\langle b\rangle} \mathbf{0} = P'$ and $\emptyset;\Delta \ominus \{a,b\} \vdash_1 \mathbf{0} = P'$ by rule (T-NIL);

(T-STR): the proof proceeds by applying the induction hypothesis, Lemma B.1 and (T-STR);

(T-PAR): by $\Gamma;\Delta \vdash_1 P$ and the premise of the rule, we get $P = P_1 | \cdots | P_n$ with P_i prime, $\Gamma_i;\Delta_i \vdash_1 P_i$ for each i, $\Gamma = \bigcup_i \Gamma_i$ and $\Delta = \bigcup_i \Delta_i$. Moreover, $\Gamma_i^\rho \cap \Gamma_j^\rho = \Delta_i^\rho \cap \Delta_j^\rho = \emptyset$, for each $i \neq j$, and $\mathrm{DG}(\Gamma_i^\rho, \Delta_i^\rho)_{i=1,\cdots,n}$ is acyclic.

$P \xrightarrow{\overline{a}\langle b\rangle} P'$ means that there is a $j \in \{1,\cdots,n\}$ such that $P_j = \overline{a}\langle b\rangle \xrightarrow{\overline{a}\langle b\rangle} \mathbf{0}$.

The induction hypothesis can be applied to P_j: $\Gamma_j;\Delta_j \ominus \{a,b\} \vdash_1 \mathbf{0}$. Note that $\Delta_i^\rho \cap \Delta_j^\rho = \emptyset$ for $i \neq j$ implies that $\Delta \ominus \{a,b\} = \bigcup_{i \neq j} \Delta_i \cup (\Delta_j \ominus \{a,b\})$. Therefore, rule (T-PAR) can be applied for deducing $\Gamma;\Delta \ominus \{a,b\} \vdash_1 P_1 | \cdots | P_{j-1} | P_{j+1} | \cdots | P_n = P'$;

(T-RES): by $\Gamma;\Delta \vdash_1 P = (\nu c)Q$ and the premise of the rule, we get $\Gamma, c;\Delta, c \vdash_1 Q$. By hypothesis $P \xrightarrow{\overline{a}\langle b\rangle} P'$. By (RES), $a, b \neq c$ and $Q \xrightarrow{\overline{a}\langle b\rangle} Q'$, with $P' = (\nu c)Q'$. By applying the induction hypothesis to Q, we get $\Gamma, c;\Delta \ominus \{a,b\}, c \vdash_1 Q'$ and by (T-RES), $\Gamma;\Delta \ominus \{a,b\} \vdash_1 (\nu c)Q' = P'$;

(T-RES-I), (T-RES-⊥): the proof proceeds as already seen for the previous case.

3. By induction on the derivation of $\Gamma;\Delta \vdash_1 P$, the proof proceeds by distinguishing the last typing rule applied. The interesting cases are (T-RES) and (T-RES-I), in the other cases the proof proceeds by applying the induction hypothesis as already seen for the previous point.

(T-RES): by $P = (\nu c)Q$ and the premise of the rule, we get $c : \mathsf{T}^{\mathsf{U}}$ and $\Gamma, c; \Delta, c \vdash_1 Q$. If $c \neq b$ then $Q \xrightarrow{\overline{a}(b)} Q'$, $P \xrightarrow{\overline{a}(b)} P' = (\nu c)Q'$ and the induction hypothesis can be applied for deducing $\Gamma, b, c; (\Delta, b, c) \ominus \{a, b\} \vdash_1 Q'$ Hence, by (T-RES), $\Gamma, b; (\Delta, b) \ominus \{a, b\} \vdash_1 (\nu c)Q' = P'$. Suppose now $b = c$. In this case, $Q \xrightarrow{\overline{a}\langle b\rangle} Q'$ and $P \xrightarrow{\overline{a}(b)} P' = Q'$. By the previous point, we get $\Gamma, b; (\Delta, b) \ominus \{a, b\} \vdash_1 Q'$, hence the result;

(T-RES-I): in this case the proof proceeds similarly, recall that inert names are not added to the input context Γ.

□

Theorem B.1 (Theorem 1). *Suppose* $\Gamma; \Delta \vdash_1 P$ *and* $P \xrightarrow{[a]} P'$. *Then* $\Gamma \ominus \{a\}; \Delta \ominus \{a\} \vdash_1 P'$.

PROOF: Consider the normal form of P (Lemma 2) $P \equiv (\nu \tilde{d})(P_1 | \cdots | P_n) \triangleq N$. By Lemma B.1, $N \xrightarrow{[a]} N'$, with $N' \equiv P'$.

Consider a normal derivation of $\Gamma; \Delta \vdash_1 N$ (Lemma 1) and suppose for simplicity that in $\tilde{d}$ there are no channels of type $\bot$ or I. Suppose $\tilde{d}$ is not empty, the last typing rule applied in the derivation must be (T-RES). Before (T-RES), rule (T-PAR) must have been applied for deriving $\Gamma, \tilde{d}; \Delta, \tilde{d} \vdash_1 P_1 | \cdots | P_n$. Hence, by its premise we get that there are suitable Γ_i, Δ_i such that $\Gamma_i; \Delta_i \vdash_1 P_i$, for each i, $\Gamma, \tilde{d} = \cup_{i=1,\ldots,n}\Gamma_i$, $\Delta, \tilde{d} = \cup_{i=1,\ldots,n}\Delta_i$, $\Gamma_i^{\rho} \cap \Gamma_j^{\rho} = \Delta_i^{\rho} \cap \Delta_j^{\rho} = \emptyset$, for each $i, j \in \{1, \ldots, n\}$ and $i \neq j$, and $\mathrm{DG}(\Gamma_i^{\rho}, \Delta_i^{\rho})_{i=1,\ldots,n}$ is acyclic.

Let us proceed by considering the reduction $N \xrightarrow{[a]} N' \equiv P'$. Given that $(\nu \tilde{d})(P_1 | \cdots | P_n) \triangleq N$, it must be $P_1 | \cdots | P_n \xrightarrow{[a]} R$. Each P_i is prime, hence it must be $P_i = \overline{a}\langle b\rangle$ and $P_j = (!)a(x).P_j'$ for some b and $i, j \in \{1, \ldots, n\}$. Therefore $P_i \xrightarrow{\overline{a}\langle b\rangle} \mathbf{0} = P_i'$ and $P_j \xrightarrow{a(b)} P_j'[b/x]$. By Proposition B.2, it can be derived that $\emptyset; \Delta_i \ominus \{a, b\} \vdash_1 \mathbf{0}$. Note also that either $b \notin \Delta_j$ or $b \in \Delta_j$ and b is an ω-receptive or inert name. Indeed, if b were responsive and $b \in \Delta_j$, given that $\emptyset; \Delta_i \vdash_1 \overline{a}\langle b\rangle$, it would be $b \in \Delta_i$, hence $\Delta_i^{\rho} \cap \Delta_j^{\rho} \neq \emptyset$. Therefore, again by Proposition B.2, it can be derived that $\Gamma_j \ominus \{a\}; \Delta_j' \vdash_1 P_j'[b/x]$, with either $\Delta_j' = \Delta_j, b$, if $b \notin \Delta_j$, or $\Delta_j' = \Delta_j$, if $b \in \Delta_j$ and b is either an ω-receptive or an inert name.

It is easy to see that the premise of (T-PAR) are still satisfied: the graph is acyclic because nested inputs are not allowed hence no new arcs from b can be added to the graph; and if b is responsive $b \notin \Delta_j$ and $b \notin (\Delta_i \ominus \{a, b\})$. Hence $(\Gamma, \tilde{d}) \ominus \{a\}; (\Delta, \tilde{d}) \ominus \{a\} \vdash_1 P_1 | \cdots | P_i' | \cdots | P_j' | \cdots | P_n = R$.

We distinguish the following two main cases.

- Suppose either (COM_i), or (PAR_i), or (RES) is the last rule applied in the derivation of $N \xrightarrow{[a]} N'$. In this case, $N' = (\nu \tilde{d})R$ and from $(\Gamma, \tilde{d}) \ominus \{a\}; (\Delta, \tilde{d}) \ominus \{a\} \vdash_1 R$ and the typing rules for restriction, we get $\Gamma \ominus \{a\}; \Delta \ominus \{a\} \vdash_1 (\nu \tilde{d})R = N'$.

- Suppose (RES-ρ) is the last applied in the derivation of $N \xrightarrow{[a]} N'$. In this case, $N' = (\nu \tilde{d}[c/a])R[c/a]$, for some $c : \bot$. Given that a is responsive and $a \in \tilde{d}$ we get $a \notin (\Gamma, \tilde{d}) \ominus \{a\}$ and $a \notin (\Delta, \tilde{d}) \ominus \{a\}$. Hence, from $(\Gamma, \tilde{d}) \ominus \{a\}; (\Delta, \tilde{d}) \ominus \{a\} \vdash_1 R$ and the typing rules for restriction, we get $\Gamma; \Delta \vdash_1 (\nu \tilde{d'})R$, for $\tilde{d'} = \tilde{d} - a$, and $a \notin \mathrm{fn}(R)$. Hence, by (T-RES-$\bot$), $\Gamma; \Delta \vdash_1 (\nu c)(\nu \tilde{d'})R$, for each $c : \bot$. Therefore, by (T-STR), $\Gamma; \Delta \vdash_1 N'$.

Finally, from $\Gamma; \Delta \vdash_1 N'$ and (T-STR), we get $\Gamma; \Delta \vdash_1 P'$. □

C Proof of Theorem 2

In this section we prove the intermediate results needed for proving Theorem 2 (type soundness).

Proposition C.1 (Proposition 1). *Suppose that* $\Gamma;\Delta \vdash_1 P$*, with* Γ*,* Δ *and* P *satisfying the conditions in the premise of rule* (T-PAR) *and* $\Gamma^\rho = \Delta^\rho \neq \emptyset$*. Then for some* j *in* $1,\ldots,n$ *we have* $P_j = \overline{a}\langle b\rangle$ *with either* a *or* b *responsive.*

PROOF: Let $P = P_1 \mid \cdots \mid P_n$. For each i, process P_i is prime, $\Gamma_i;\Delta_i \vdash_1 P_i$, $\Gamma_i^\rho \cap \Gamma_j^\rho = \emptyset$ and $\Delta_i^\rho \cap \Delta_j^\rho = \emptyset$ for $i \neq j$. Moreover, $\Gamma = \bigcup_{i=1\ldots n}\Gamma_i$, $\Delta = \bigcup_{i=1\ldots n}\Delta_i$; and $\mathrm{DG}(\Gamma_i^\rho, \Delta_i^\rho)_{i=1,\ldots,n}$ is acyclic.

The acyclicity of the graph implies that there is at least one node c with no outgoing arcs. By definition of the graph and $\Gamma^\rho = \Delta^\rho$ we have that $\exists j \in \{1,\ldots,n\}$ s.t. $c \in \Delta_j^\rho$ and $\Gamma_j^\rho = \emptyset$. Consider the process P_j. By hypothesis P_j is prime and $\Gamma_j;\Delta_j \vdash_1 P_j$.

By contradiction, assume P_j is of the form $!a(b).R$. By $\Gamma_j;\Delta_j \vdash_1 !a(b).R$ and the premise of rule (T-REP), we get $\Delta_j^\rho = \emptyset$, but this is in contradiction with the hypothesis $c \in \Delta_j^\rho$, thus P_j is not of the form $!a(b).R$.

Again by contradiction, assume P_j is of the form $a(b).P$. By $\Gamma_j;\Delta_j \vdash_1 a(b).P$ and the premise of rule (T-INP), we get $\Gamma_j^\rho = \{a\}$, but this is in contradiction with the hypothesis $\Gamma_j^\rho = \emptyset$, thus P_j is not of the form $a(b).P$.

In conclusion, P_j prime implies that $P_j = \overline{a}\langle b\rangle$ with either $a = c$ or $b = c$, thus at least one of the two names is responsive. □

The lemma below ensures that substitutions preserve $\mathrm{wt}(\cdot)$.

Lemma C.1. *Suppose* $\Gamma;\Delta, x \vdash_1 P$ *and* $x, b : T$*. Then* $\mathrm{wt}(P) = \mathrm{wt}(P\{b/x\})$.

PROOF: The proof is straightforward by induction on the definition of $\mathrm{wt}(\cdot)$ (note that $x, b : \mathsf{T}$ implies that $\mathrm{lev}(x) = \mathrm{lev}(b)$). □

The following lemma ensures that the weight of a process is a good measure when considering responsive reductions, in fact it decreases after each communication involving responsive names. This is a consequence of the constraints on levels in the premise of rule (T-REP) and of the linearity of responsive names. The lemma below is a step forward this result.

Lemma C.2. *Suppose* $\Gamma;\Delta \vdash_1 P$*, then:*

1. *if* $a \in \Gamma$*,* $a : T^U$ *and* $b : T$ *then* $P \xrightarrow{a(b)} P'$ *and, if either* a *or* b *is responsive, then* $\mathrm{wt}(P') \prec \mathrm{wt}(P) + 0_{\mathrm{lev}(a)}$*;*
2. *if* $P \xrightarrow{\overline{a}\langle b\rangle} P'$ *(or* $P \xrightarrow{\overline{a}(b)} P'$*) then* $\mathrm{wt}(P') \preceq \mathrm{wt}(P) - 0_{\mathrm{lev}(a)}$.

PROOF: In both cases the proof proceeds by induction on the derivation of $\Gamma;\Delta \vdash_1 P$.

1. Consider the last typing rule applied in the derivation; the most interesting cases are rules (T-INP) and (T-REP). The other cases can be easily proved by applying the inductive hypothesis.

   (T-INP): Suppose $P = a(x).R$. By rule (IN), $a(x).R \xrightarrow{a(b)} R\{b/x\}$ and by the premise of (T-INP), $x : \mathsf{T}$. $\mathrm{wt}(a(x).R) + 0_{\mathrm{lev}(a)} = \mathrm{wt}(R) + 0_{\mathrm{lev}(a)} \succ \mathrm{wt}(R\{b/x\}) = \mathrm{wt}(R)$, by Lemma C.1.

   (T-REP): Suppose $P = !a(x).R$. By $a;\Delta \vdash_1 !a(x)R$ and the premise of the rule, we get $x : \mathsf{T}$ and $\forall c \in \mathrm{os}(R) : \mathrm{lev}(c) < \mathrm{lev}(a)$. By rule (REP), $!a(x).R \xrightarrow{a(b)} !a(x).R|R\{b/x\}$. If b is ω-receptive, there is nothing to prove. Otherwise, from $\forall c \in \mathrm{os}(R) : \mathrm{lev}(c) < \mathrm{lev}(a)$ and $x, b : \mathsf{T}$ we get $\forall c \in \mathrm{os}(R\{b/x\}) : \mathrm{lev}(c) < \mathrm{lev}(a)$. Hence, $\mathrm{wt}(!a(x).R) + 0_{\mathrm{lev}(a)} = 0_{\mathrm{lev}(a)} \succ \mathrm{wt}(R\{b/x\}) = \mathrm{wt}(!a(x).R|R\{b/x\})$;

2. Consider the last typing rule applied in the derivation; the most interesting cases are rules (T-OUT) and (T-RES). The other cases can be easily proved by applying the inductive hypothesis.

   (T-OUT): Suppose $P = \overline{a}\langle b\rangle$. By (OUT), $\overline{a}\langle b\rangle \xrightarrow{\overline{a}\langle b\rangle} \mathbf{0}$ and $\mathrm{wt}(\mathbf{0}) = \mathrm{wt}(\overline{a}\langle b\rangle) - 0_{\mathrm{lev}(a)}$;

   (T-RES): Suppose $P = (\nu d)R$ and $d : \mathsf{T}$ (the cases $d : \mathsf{I}$ and $d : \bot$ are proved similarly.) By the premise of (T-RES) and $\Gamma;\Delta \vdash_1 (\nu d)R$ we get $\Gamma, d; \Delta, d \vdash_1 R$. We distinguish two cases considering the transition rule applied:

   (OPEN): by $(\nu d)R \xrightarrow{\overline{a}\langle d\rangle} R'$ and the premise of the rule, we get $R \xrightarrow{\overline{a}\langle d\rangle} R'$ and, by inductive hypothesis, $\mathrm{wt}(R') \preceq \mathrm{wt}(R) - 0_{\mathrm{lev}(a)} = \mathrm{wt}((\nu d)R) - 0_{\mathrm{lev}(a)}$;

   (RES): by $(\nu d)R \xrightarrow{\overline{a}\langle b\rangle} (\nu d)R'$ and the premise of the rule, we get $R \xrightarrow{\overline{a}\langle b\rangle} R'$, $a, b \neq d$ and, by inductive hypothesis, $\mathrm{wt}(R') \preceq \mathrm{wt}(R) - 0_{\mathrm{lev}(a)}$. By definition of $\mathrm{wt}(\cdot)$, $\mathrm{wt}((\nu d)R') = \mathrm{wt}(R') \preceq \mathrm{wt}(R) - 0_{\mathrm{lev}(a)} = \mathrm{wt}((\nu d)R) - 0_{\mathrm{lev}(a)}$.

□

Lemma C.3. *$P \equiv R$ implies $\mathrm{wt}(P) = \mathrm{wt}(R)$.*

PROOF: The proof is straightforward by induction on the derivation of $P \equiv R$. □

Proposition C.2 (Proposition 2). *Suppose $\Gamma;\Delta \vdash_1 P$ and $P \xrightarrow{\tau\langle a,b\rangle} P'$, with either a or b responsive. Then $\mathrm{wt}(P') \prec \mathrm{wt}(P)$.*

PROOF: Consider the normal form of P (Lemma 2) $P \equiv (\nu \tilde{d})(P_1 | \cdots | P_n) \triangleq N$. By Lemma B.1, $N \xrightarrow{\tau\langle a,b\rangle} N'$, with $N' \equiv P'$.

Consider a normal derivation of $\Gamma;\Delta \vdash_1 N$ (Lemma 1) and suppose for simplicity that in $\tilde{d}$ there are no channels of type $\bot$ or I. The last typing rule applied in the derivation should be (T-RES) – if $\tilde{d}$ is not empty. Before (T-RES), rule (T-PAR) must have been applied for deriving $\Gamma, \tilde{d}; \Delta, \tilde{d} \vdash_1 P_1 | \cdots | P_n$. Hence, by its premise, we get that there are suitable Γ_i, Δ_i such that $\Gamma_i; \Delta_i \vdash_1 P_i$, for each $i = 1, \ldots, n$.

Let us proceed by considering the reduction $N \xrightarrow{\tau\langle a,b\rangle} N' \equiv P'$. It must be $R \triangleq P_1 | \cdots | P_n \xrightarrow{\tau\langle a,b\rangle} R'$. Each P_i is prime, hence it must be $P_i = \overline{a}\langle b\rangle$ and $P_j = (!)a(x).P'_j$ for some b and $i, j \in \{1, \ldots, n\}$. Suppose for simplicity $P_j = a(x).P'_j$. Therefore, $P_i \xrightarrow{\overline{a}\langle b\rangle} \mathbf{0} = P'_i$, $P_j \xrightarrow{a(b)} P'_j[{}^{b}/_{x}]$, $R' = P_1 | \cdots | P'_i | \cdots | P'_j | \cdots | P_n$ and $N' = (\nu \tilde{d'})R'$, with either $\tilde{d'} = \tilde{d}$ or $\tilde{d'} = \tilde{d}[{}^{a}/_{c}]$ for some $c : \bot$. Given that a or b is responsive, by Lemma C.2 we get $\mathrm{wt}(P'_j[{}^{b}/_{x}]) \prec \mathrm{wt}(P_j) + 0_{\mathrm{lev}(a)}$ and $\mathrm{wt}(P'_i) \preceq \mathrm{wt}(P_i) - 0_{\mathrm{lev}(a)}$. Hence, $\mathrm{wt}(R') = \mathrm{wt}(P_1) + \cdots + \mathrm{wt}(P'_i) + \cdots + \mathrm{wt}(P'_j) + \cdots + \mathrm{wt}(P_n) \prec \sum_i \mathrm{wt}(P_i) = \mathrm{wt}(R)$ and $\mathrm{wt}(N') \prec \mathrm{wt}(N)$, by definition of $\mathrm{wt}(\cdot)$. Finally, by Lemma C.3, $P \equiv N$ and $P' \equiv N'$, we get $\mathrm{wt}(P') \prec \mathrm{wt}(P)$. □

D Proof of Theorem 3

In what follows we introduce some notations and prove some preliminary results useful for proving Theorem 3. Note that proof of a similar statement is outlined in [5]; our proof proceeds along the same lines.

In the following, we write $|\mathrm{O}^{\rho}(P)|$ and $|\mathrm{O}(P)|$ for the cardinality of $\mathrm{O}^{\rho}(P)$ and $\mathrm{O}(P)$, respectively. The *height* of P, written $\mathrm{h}(P)$, is defined as the greatest size of a replicated term in P. E.g. $\mathrm{h}(!a(x).P) = 1 + |P|$.

First of all, we prove that size, weight and height of a process and the number of outputs it contains are preserved by structural equivalence and substitutions. Moreover we prove that the number of outputs

in a process P is upper bounded by $|P|$. Next, in Proposition D.1, we show that the number of output actions in P and its size may increase only after a reduction where the subject is an ω-receptive name, but this increase is limited by $\mathrm{h}(P)$.

Lemma D.1.

1. *If* $P \equiv R$ *then* $|P| = |R|$, $|\mathrm{O}^{\rho}(P)| = |\mathrm{O}^{\rho}(R)|$ *and* $\mathrm{h}(P) = \mathrm{h}(R)$.
2. *Let be* $x, b : T$. $\mathrm{wt}(P) = \mathrm{wt}(P[b/x])$, $|\mathrm{O}^{\rho}(P)| = |\mathrm{O}^{\rho}(P[b/x])|$ *and* $\mathrm{h}(P) = \mathrm{h}(P[b/x])$.

PROOF: The proof is straightforward by induction on the derivation of $P \equiv R$ and by definition of $|\cdot|$, $\mathrm{wt}(\cdot)$, $|\mathrm{O}^{\rho}(\cdot)|$ and $\mathrm{h}(\cdot)$. □

Lemma D.2. *If* $\Gamma;\Delta \vdash_1 P$ *then* $|\mathrm{O}^{\rho}(P)| \leq |P|$.

PROOF: By definition of $|\mathrm{O}^{\rho}(P)|$. □

Proposition D.1. *If* $\Gamma;\Delta \vdash_1 P$ *then:*

1. *if either* $P \xrightarrow{\overline{a}\langle b\rangle} P'$ *or* $P \xrightarrow{\overline{a}(b)} P'$ *and either* a *or* b *is responsive then* $|\mathrm{O}^{\rho}(P')| = |\mathrm{O}^{\rho}(P)| - 1$*;*
2. *if* $P \xrightarrow{a(b)} P'$*, with* a *responsive, then* $|\mathrm{O}^{\rho}(P')| = |\mathrm{O}^{\rho}(P)|$*;*
3. *if* $P \xrightarrow{a(b)} P'$*, with* a ω*-receptive, then* $|\mathrm{O}^{\rho}(P')| \leq |\mathrm{O}^{\rho}(P)| + \mathrm{h}(P)$*;*
4. *if* $P \xrightarrow{[a]} P'$*, with* a *responsive, then* $|\mathrm{O}^{\rho}(P')| \leq |\mathrm{O}^{\rho}(P)| - 1$*;*
5. *if* $P \xrightarrow{[a]} P'$*, with* a ω*-receptive carrying responsive names, then* $|\mathrm{O}^{\rho}(P')| \leq |\mathrm{O}^{\rho}(P)| + \mathrm{h}(P) - 1$.

PROOF: In all cases the proof proceeds by induction on the derivation of $P \xrightarrow{\mu} P'$ and distinguishes the last transition rule applied:

1. the interesting case is (OUT); the other cases ((PAR$_1$), (OPEN) and (RES)) can be proved by applying the inductive hypothesis.

   By (OUT), $\overline{a}\langle b\rangle \xrightarrow{\overline{a}\langle b\rangle} \mathbf{0}$ with a or b responsive, and $|\mathrm{O}^{\rho}(\overline{a}\langle b\rangle)| - 1 = 0 = |\mathrm{O}^{\rho}(\mathbf{0})|$;

2. the interesting case is (IN); the other cases ((PAR$_1$) and (RES)) can be proved by applying the inductive hypothesis.

   By (IN), $a(x).P \xrightarrow{a(b)} P[b/x]$. By the premise of (T-INP), $\Gamma;\Delta \vdash_1 a(x).P$ and $a : \mathsf{T}^{[\rho,k]}$, we get $x : \mathsf{T}$. $\mathrm{O}^{\rho}(a(x).P) = \mathrm{O}^{\rho}(P)$, thus $|\mathrm{O}^{\rho}(a(x).P)| = |\mathrm{O}^{\rho}(P)|$ and $|\mathrm{O}^{\rho}(P)| = |\mathrm{O}^{\rho}(P[b/x])|$ by $b, x : \mathsf{T}$ and Lemma D.1 (2);

3. the interesting case is (REP); the other cases ((PAR$_1$) and (RES)) can be proved by applying the inductive hypothesis.

   By (REP), $!a(x).P \xrightarrow{a(b)} !a(x).P|P[b/x]$. By the premise of (T-REP), $\Gamma;\Delta \vdash_1 !a(x).P$ and $a : \mathsf{T}^{[\omega,k]}$, it follows that $x : \mathsf{T}$.

   $\mathrm{O}^{\rho}(!a(x).P) = \emptyset$, $\mathrm{h}(!a(x).P) = 1 + |P|$ and $|\mathrm{O}^{\rho}(!a(x).P|P[b/x])| = |\mathrm{O}^{\rho}(P[b/x])|$. By Lemma D.2, $|\mathrm{O}^{\rho}(P)| \leq |P|$, hence by $b, x : \mathsf{T}$, Lemma D.1 (2) and $\mathrm{h}(!a(x).P) = 1 + |P|$, $|\mathrm{O}^{\rho}(P[b/x])| \leq \mathrm{h}(!a(x).P)$ and $|\mathrm{O}^{\rho}(!a(x).P|P[b/x])| \leq |\mathrm{O}^{\rho}(!a(x).P)| + \mathrm{h}(!a(x).P)$;

4. the interesting cases are rules (COM$_1$) and (CLOSE$_1$); the other cases ((PAR$_1$), (RES) and (RES-ρ)) can be proved by applying the inductive hypothesis.

   (COM$_1$): by $R|S \xrightarrow{\tau\langle a,b\rangle} R'|S'$, with a responsive name, and the premise of the rule, it follows that $R \xrightarrow{\overline{a}\langle b\rangle} R'$ and $S \xrightarrow{a(b)} S'$. By $\Gamma;\Delta \vdash_1 R|S$ and rules (T-STR), (T-RES), (T-RES-I) and (T-PAR), there are suitable contexts $\Gamma_1,\Delta_1,\Gamma_2$ and Δ_2, such that $\Gamma_1;\Delta_1 \vdash_1 R$ and $\Gamma_2;\Delta_2 \vdash_1 S$. Moreover, by well-typedness and $a : \mathsf{T}^{[\rho,k]}$ we get $b : \mathsf{T}$.
   By (1,2) it follows that $|\mathrm{O}^\rho(R')| = |\mathrm{O}^\rho(R)| - 1$ and $|\mathrm{O}^\rho(S')| = |\mathrm{O}^\rho(S)|$, thus $|\mathrm{O}^\rho(R'|S')| = |\mathrm{O}^\rho(R')| + |\mathrm{O}^\rho(S')| = |\mathrm{O}^\rho(R)| - 1 + |\mathrm{O}^\rho(S)| = |\mathrm{O}^\rho(R|S)| - 1$;

   (CLOSE$_1$): this case is similar to the previous one;

5. the interesting cases are rules (COM$_1$) and (CLOSE$_1$); the other cases ((PAR$_1$) and (RES)) can be proved by applying the inductive hypothesis.

   (COM$_1$): by $R|S \xrightarrow{\tau\langle a,b\rangle} R'|S'$, with a ω-receptive name, and the premise of the rule, it follows that $R \xrightarrow{\overline{a}\langle b\rangle} R'$ and $S \xrightarrow{a(b)} S'$. By $\Gamma;\Delta \vdash_1 R|S$ and rules (T-STR), (T-RES), (T-RES-I) and (T-PAR), there are suitable contexts $\Gamma_1,\Delta_1,\Gamma_2$ and Δ_2, such that $\Gamma_1;\Delta_1 \vdash_1 R$ and $\Gamma_2;\Delta_2 \vdash_1 S$. Moreover, by well-typedness and $a : \mathsf{T}^{[\omega,k]}$ it follows that $b : \mathsf{T}$.
   By (1,3) it can be deduced that $|\mathrm{O}^\rho(R')| = |\mathrm{O}^\rho(R)| - 1$ and $|\mathrm{O}^\rho(S')| \le |\mathrm{O}^\rho(S)| + \mathrm{h}(S)$, thus $|\mathrm{O}^\rho(R'|S')| = |\mathrm{O}^\rho(R')| + |\mathrm{O}^\rho(S')| \le |\mathrm{O}^\rho(R)| - 1 + |\mathrm{O}^\rho(S)| + \mathrm{h}(S) \le |\mathrm{O}^\rho(R|S)| + \mathrm{h}(R|S) - 1$ (note that $\mathrm{h}(R|S) \ge \mathrm{h}(S)$ by definition);

   (CLOSE$_1$): this case is similar to the previous one.

□

The height of a process is preserved by transitions:

Lemma D.3. *If $P \xrightarrow{\mu} P'$ then $\mathrm{h}(P') = \mathrm{h}(P)$.*

PROOF: The proof is straightforward by induction on the derivation of $P \xrightarrow{\mu} P'$. □

Each component of the weight vector of a process P gives the number of active outputs in P of the corresponding level.

Lemma D.4. *If $\Gamma;\Delta \vdash_1 P$ and $\mathrm{wt}(P) = \langle w_k,\ldots,w_0\rangle$ then in $\mathrm{O}^\rho(P)$ there are at most w_i outputs of level i for i in $0,\ldots,k$.*

PROOF: The proof is straightforward by induction on the structure of P. □

Theorem D.1 (Theorem 3). *Let P be $(\Gamma;\Delta)$-balanced and $r \in \Delta^\rho$ and let k be the maximal level of names appearing in active responsive output actions of P, $\mathrm{O}^\rho(P)$. In all responsive schedulings, the number of reductions preceding a reduction on r is upper-bounded by $|P|^{k+1}$.*

PROOF: (outline) By Theorem 2 (type soundness) we get that whenever $P \xrightarrow{[s]} P'$ with $s \in \mathcal{N}^*$ and $r \notin s$ then $P' \xrightarrow{[r]}$.

Let $S = P_0 \xrightarrow{\tau\langle a_1,b_1\rangle} P_1 \xrightarrow{\tau\langle a_2,b_2\rangle} P_2 \xrightarrow{\tau\langle a_3,b_3\rangle} \cdots$ be a maximal responsive scheduling not containing r as subject. The length of S, which is finite because each reduction step involving a responsive name decreases wt (Proposition 2), is an upper bound for n.

First, note that every process P_i in S is well typed by Theorem 1 (subject reduction). Moreover, $\mathrm{h}(P_i) = \mathrm{h}(P)$ for each $i = 1,2,\ldots$, Lemma D.3.

Suppose that $\mathrm{wt}(P) = \langle w_k, \ldots, w_0 \rangle$. In general, any reduction of involving a subject of level $i \in \{1, \ldots, k\}$ can directly activate at most $\mathrm{h}(P) \leq |P|$ reductions – that is, can add to $\mathrm{O}^{\rho}(P)$ at most $\mathrm{h}(P) \leq |P|$ outputs – of level at most $i-1$ (Proposition D.1, Lemma D.2, well typed-ness of P_i and (T_{+}-$\textsc{Rep}^{\rho}$)). Hence, each reduction of level i can directly or indirectly be causally involved in at most $\mathrm{f}(i)$ reductions of S, where the function f is defined as

$$\begin{cases} \mathrm{f}(0) = 1 \\ \mathrm{f}(i) \ = 1 + |P| * \mathrm{f}(i-1) \leq \sum_{j=0}^{i} |P|^j = \frac{|P|^{i+1}-1}{|P|-1}. \end{cases}$$

Note that $\mathrm{f}(i)$ is monotone on i by definition. By definition of $\mathrm{O}^{\rho}(\cdot)$, only output in $\mathrm{O}^{\rho}(P)$ can be involved in a responsive reduction from P. Moreover, by Lemma D.4, there are at most w_i active outputs of level i in $\mathrm{O}^{\rho}(P)$. Thus:

$$n \leq \sum_{i=0}^{k} w_i * \mathrm{f}(i) \ \text{ hence, by monotonicity, } \ \mathrm{f}(i) \leq \mathrm{f}(k) * (w_0 + \cdots + w_k) \ .$$

Now, $\mathrm{f}(k) \leq \frac{|P|^{k+1}-1}{|P|-1}$ and $w_k + \cdots + w_0 \leq |P|$, thus

$$n \leq \mathrm{f}(k) * (w_k + \cdots + w_0) \leq |P|^{k+1} \ .$$

□

E Proof of Theorem 1 and Theorem 2 in the extended calculus

In this section we prove that subject reduction and type safety are guaranteed in the calculus extended with `if-then-else`.

First of all some notations/definitions have to be extended. Each process of the form `if` G `then` P `else` Q is considered prime. The auxiliary functions over processes defined through the paper are extended as follows.

$$\begin{aligned} |\,\texttt{if}\ G\ \texttt{then}\ P\ \texttt{else}\ Q\,| &= 1 + \mathbf{max}(|P|, |Q|) \\ \mathrm{wt}(\texttt{if}\ G\ \texttt{then}\ P\ \texttt{else}\ Q) &= \mathrm{wt}(P) + \mathrm{wt}(Q) \\ \mathrm{O}(\texttt{if}\ G\ \texttt{then}\ P\ \texttt{else}\ Q) &= \mathrm{O}(P) \uplus \mathrm{O}(Q) \\ \mathrm{os}(\texttt{if}\ G\ \texttt{then}\ P\ \texttt{else}\ Q) &= \mathrm{os}(P) \cup \mathrm{os}(Q) \end{aligned}$$

In the following we report the enunciates and proofs of the results proved in Appendices A-C that require major changes.

Theorem E.1 (Theorem 1). *(i) Suppose* $\Gamma; \Delta \vdash_1 P$ *and* $P \xrightarrow{[a]} P'$. *Then* $\Gamma \ominus \{a\}; \Delta \ominus \{a\} \vdash_1 P'$. *(ii) Suppose* $\Gamma; \Delta \vdash_1 P$ *and* $P \xrightarrow{[\varepsilon]} P'$. *Then* $\Gamma; \Delta \vdash_1 P'$.

Proof: Point (i) has been already proved at page 32. Concerning the new point (ii), the proof proceeds by induction on the derivation of $P \xrightarrow{[\varepsilon]} P'$. The base case is when either (IF-T) or (IF-F) is the last rule applied. In the other cases the proof proceeds by applying the inductive hypothesis.

Suppose (IF-F) is the last applied; hence $\texttt{if}\ G\ \texttt{then}\ P_1\ \texttt{else}\ P_2 \xrightarrow{[\varepsilon]} P_1$. By $\Gamma;\Delta \vdash_1 \texttt{if}\ G\ \texttt{then}\ P_1\ \texttt{else}\ P_2$ and the premise of (T-IF), we get $\Gamma;\Delta \vdash_1 P_1$, hence the result.

The proof proceeds similarly if (IF-T) is the last applied. □

Proposition 1 does not hold for the extended calculus because of the possible presence top-level `if-then-else`'s. Therefore, it is necessary to limit that result to ε-stable processes (see Proposition E.1 below). A process is ε-*stable* if $P \not\xrightarrow{[\varepsilon]}$. The following holds.

Lemma E.1. *For each process P there exists an ε-stable P' such that $P \xRightarrow{[\varepsilon]} P'$.*

Proposition E.1. *Suppose that $\Gamma;\Delta \vdash_1 P$, with P ε-stable, Γ, Δ and P satisfying the conditions in the premise of rule* (T-PAR) *and $\Gamma^\rho = \Delta^\rho \neq \emptyset$. Then for some $j \in \{1,\ldots,n\}$ we have $P_j = \overline{a}\langle b\rangle$ with either a or b responsive.*

PROOF: The proof coincides with that on page 33. Note that it cannot be $P_j = \texttt{if}\ G\ \texttt{then}\ R_1\ \texttt{else}\ R_2$ because in this case it would be $P_j \xrightarrow{[\varepsilon]}$, hence $P \xrightarrow{[\varepsilon]}$. □

Lemma E.2. *Suppose $\Gamma;\Delta \vdash_1 P$ and $P \xrightarrow{\varepsilon} P'$ then* $\mathrm{wt}(P') \preceq \mathrm{wt}(P)$.

PROOF: Follows by definition of $\mathrm{wt}(\cdot)$; recall that $\mathrm{wt}(\texttt{if}\ G\ \texttt{then}\ P\ \texttt{else}\ Q) = \mathrm{wt}(P) + \mathrm{wt}(Q)$. □

Theorem E.2 (type soundness, Theorem 2). *Let P be $(\Gamma;\Delta)$-balanced, and $r \in \Delta^\rho$. Then P guarantees responsiveness of r.*

PROOF: The proof proceeds as that on page 10, with one difference: P' is taken to be an ε-stable process with minimal wt reachable from P without using r as communication subject. The rest of the proof proceeds unchanged, modulo relying on Proposition E.1 rather than on Proposition 1. □

F Proof of Proposition 3

In this section we prove that primitive recursive functions can be encoded into well typed processes and that the proposed encoding is correct. We will take advantage of some special properties enjoyed by type derivations for processes that encode primitive recursive functions. To this purpose, we introduce the notation $\Gamma;\Delta \vdash_{\mathrm{u}} P$ to mean that $\Gamma;\Delta \vdash_1 P$ can be deduced with the following extra constraint on ω-receptive names in the premise of (T-PAR): $\Gamma_i^\omega \cap \Gamma_j^\omega = \emptyset$. (Of curse (T-REP) is replaced by (T-REP') in both $\vdash_1$ and $\vdash_{\mathrm{u}}$.) The following result obviously holds true:

Lemma F.1. *$\Gamma;\Delta \vdash_{\mathrm{u}} P$ implies $\Gamma;\Delta \vdash_1 P$.*

The extra constraints of $\vdash_{\mathrm{u}}$ are related to confluence, which is a crucial ingredient for the proof of correctness of the encoding.

Proposition F.1. *If $\Gamma;\Delta \vdash_1 P$, $P \xrightarrow{\overline{a}\langle b\rangle}$ and $P \xrightarrow{\tau\langle a_1,b_1\rangle} \cdots \xrightarrow{\tau\langle a_n,b_n\rangle} P'$, with $a \neq a_i$ and $b \neq b_i$ for every i, then $P' \xrightarrow{\overline{a}\langle b\rangle}$. Moreover, if $P' \xrightarrow{\overline{r}\langle c\rangle}$ with $r \in \Delta^\rho$ then $r \neq a_i$ for each i.*

PROOF: The result follows by observing that $P \xrightarrow{\overline{a}\langle b\rangle}$ implies that $P \equiv (\nu\tilde{d})(\overline{a}\langle b\rangle|Q)$ for suitable $\tilde{d}$ and Q, and $a \neq a_i$ for each $i = 1,\ldots,n$ implies $Q \xrightarrow{\tau\langle a_1,b_1\rangle} \cdots \xrightarrow{\tau\langle a_n,b_n\rangle} Q'$ and $P' \equiv (\nu\tilde{d'})(\overline{a}\langle b\rangle|Q')$. Finally, $r \neq a_i$ follows from the linear usage of responsive names in well typed processes. □

Proposition F.2. *Suppose* $\Gamma;\Delta \vdash_{\mathrm{u}} P$. *If* $P \xrightarrow{\tau\langle a,b\rangle} P'$ *and* $P \xrightarrow{\tau\langle a',b'\rangle} P''$ *then there is a* P''' *such that* $P' \xrightarrow{\tau\langle a',b'\rangle} \equiv P'''$ *and* $P'' \xrightarrow{\tau\langle a,b\rangle} \equiv P'''$. *Moreover, if* $r \in \Gamma^{\mathrm{p}} \cap \Delta^{\mathrm{p}}$, $P \xrightarrow{\overline{r}\langle c\rangle}$ *and* $r \neq a, a'$ *then* $P''' \xrightarrow{\overline{r}\langle c\rangle}$.

PROOF: Consider the normal form of P: $P \equiv (\nu\tilde{d})(\prod_{i\in\{1,\ldots,n\}} P_i)$. First of all we prove the existence of P'''. Suppose that $P \xrightarrow{\tau\langle a,b\rangle} P'$ and $P \xrightarrow{\tau\langle a',b'\rangle} P''$ and both inputs on a, a' are replicated. In both cases, by the premise of rules (RES), (PAR_1), (COM_1), (OUT) and (REP), we have that:

- there are $j,k \in \{1,\ldots,n\}$ s.t. $P_j = \overline{a}\langle b\rangle$, $P_k = !a(x).P'_k$ and $P' = (\nu\tilde{d})(\prod_{i\in\{1,\ldots,n\}\setminus\{j\}} R_i)$, where $R_k = P_k|P'_k[b/x]$ and $R_i = P_i$ for $i \neq k$;
- there are $l,m \in \{1,\ldots,n\}$ s.t. $P_l = \overline{a'}\langle b'\rangle$, $P_m = !a'(x).P'_m$ and $P'' = (\nu\tilde{d})(\prod_{i\in\{1,\ldots,n\}\setminus\{l\}} R'_i)$, where $R'_m = P_m|P'_m[b'/x]$ and $R'_i = P_i$ for $i \neq m$.

We distinguish two cases depending on a and a':

$a = a'$: by linearity of ω-receptive names in input ($\Gamma^{\omega}_i \cap \Gamma^{\omega}_t = \emptyset$ for $i \neq t$ and $i,t = 1,\cdots,n$), it follows that $k = m$. If $j = l$ we have $P' = P''$. Otherwise:

- $P' \equiv (\nu\tilde{d})(\prod_{i\in\{1,\ldots,n\}\setminus\{j\}} R_i)$, where $R_k = P_k|P'_k[b/x]$ and $R_i = P_i$ for $i \neq k$; note that $R_l = P_l$, thus $P' \xrightarrow{\tau\langle a,b'\rangle} Q'$ where $Q' = (\nu\tilde{d})(\prod_{i\in\{1,\ldots,n\}\setminus\{j,l\}} Q'_i)$, with $Q'_k = P_k|P'_k[b/x]|P'_k[b'/x]$ and $Q'_i = P_i$ for $i \neq k$;
- $P'' \equiv (\nu\tilde{d})(\prod_{i\in\{1,\ldots,n\}\setminus\{l\}} R'_i)$, where $R'_k = P_k|P'_k[b'/x]$ and $R'_i = P_i$ for $i \neq k$; $R'_j = P_j$, thus $P'' \xrightarrow{\tau\langle a,b\rangle} Q''$ where $Q'' = (\nu\tilde{d})(\prod_{i\in\{1,\ldots,n\}\setminus\{l,j\}} Q''_i)$, with $Q''_k = P_k|P'_k[b'/x]|P'_k[b/x]$, and $Q''_i = P_i$ for $i \neq k$;

that is $Q' \equiv Q'' = P'''$.

$a \neq a'$: then $j \neq l$ and $k \neq m$. Look at the structure of P' and P'':

- $P' \equiv (\nu\tilde{d})(\prod_{i\in\{1,\ldots,n\}\setminus\{j\}} R_i)$, where $R_k = P_k|P'_k[b/x]$ and $R_i = P_i$ for $i \neq k$. Note that $R_l = P_l$ and $R_m = P_m$, thus $P' \xrightarrow{\tau\langle a',b'\rangle} Q'$ with $Q' = (\nu\tilde{d})(\prod_{i\in\{1,\ldots,n\}\setminus\{j,l\}} Q'_i)$, where $Q'_k = P_k|P'_k[b/x]$, $Q'_m = P_m|P'_m[b'/x]$ and $Q'_i = P_i$ for $i \neq k,m$;
- $P'' \equiv (\nu\tilde{d})(\prod_{i\in\{1,\ldots,n\}\setminus\{l\}} R'_i)$, where $R'_m = P_m|P'_m[b'/x]$ and $R'_i = P_i$ for $i \neq m$. $R'_j = P_j$ and $R'_k = P_k$, thus $P'' \xrightarrow{\tau\langle a,b\rangle} Q''$ with $Q'' = (\nu\tilde{d})(\prod_{i\in\{1,\ldots,n\}\setminus\{l,j\}} Q''_i)$, where $Q''_m = P_m|P'_m[b'/x]$, $Q''_k = P_k|P'_k[b/x]$ and $Q''_i = P_i$ for $i \neq k,m$;

that is: $Q' \equiv Q'' = P'''$.

Similar proofs in the other cases.

Finally, $P''' \xrightarrow{\overline{r}\langle c\rangle}$ follows from $r \neq a, a'$ and Proposition F.1. □

Corollary F.1 (confluence). *Suppose* $\Gamma;\Delta \vdash_{\mathrm{u}} P$. *If* $P \xrightarrow{\tau}{}^* P'$ *and* $P \xrightarrow{\tau}{}^* P''$ *then there is a* P''' *such that* $P' \xrightarrow{\tau}{}^* \equiv P'''$ *and* $P'' \xrightarrow{\tau}{}^* \equiv P'''$. *Moreover, if* $r \in \Gamma^{\mathrm{p}} \cap \Delta^{\mathrm{p}}$, $P \xrightarrow{\overline{r}\langle c\rangle}$ *and* r *is not used in the derivations from* P *to* P' *and* P'' *then* $P''' \xrightarrow{\overline{r}\langle c\rangle}$.

PROOF: The result follows by Proposition F.2. □

Proposition F.3 (Proposition 3). *For every k-ary primitive recursive function f there is a well-typed process $\langle f\rangle_b$ such that: for each $(v_1,\ldots,v_k)$ in Nat^k the process $G \triangleq (\nu b)(\langle f\rangle_b|\overline{b}\langle v_1,\ldots,v_k,r\rangle\,|\,r(n).\mathbf{0})$, with b ω-receptive and $r:(\mathsf{Nat})^{[\rho,h]}$ ($h\geq 0$), is balanced. Moreover, $f(v_1,\ldots,v_k)=m$ if and only if $G \xrightarrow{\tau}{}^{*} \xrightarrow{\overline{r}\langle m\rangle}$.*

PROOF: First of all, we show how to represent a primitive recursive function f using a well-typed process $\langle f\rangle_b$ and that the process G is balanced.

A function $f(\tilde{n})$ is represented as a replicated process $\langle f\rangle_b$, like $!b(\tilde{n},r).P$, where b receives a tuple of first-order arguments and a name r, which is a responsive name used as return channel for sending results. We encode the primitive recursive functions in a standard way (see e.g. [5]):

zero: $\langle n\rangle_b \triangleq !b(m,r).\overline{r}\langle 0\rangle$. If we suppose $b:(\mathsf{Nat},\mathsf{Nat}^{[\rho,1]})^{[\omega,2]}$ we have that $b;\emptyset\vdash_{\mathrm{u}}\langle n\rangle_b$; thus, considering $(\nu b)(\langle n\rangle_b|\overline{b}\langle v,r\rangle|r(m).\mathbf{0})$, with $r:\mathsf{Nat}^{[\rho,1]}$, we have that $r;r,v\vdash_{\mathrm{u}}(\nu b)(\langle n\rangle_b|\overline{b}\langle v,r\rangle|r(m).\mathbf{0})$;

successor: $\langle s\rangle_b \triangleq !b(m,r).\overline{r}\langle m+1\rangle$. If we suppose $b:(\mathsf{Nat},\mathsf{Nat}^{[\rho,1]})^{[\omega,2]}$ we have that $b;\emptyset\vdash_{\mathrm{u}}\langle s\rangle_b$; thus if we consider the process $(\nu b)(\langle s\rangle_b|\overline{b}\langle v,r\rangle|r(m).\mathbf{0})$ with $r:\mathsf{Nat}^{[\rho,1]}$ we have that $r;r,v\vdash_{\mathrm{u}}(\nu b)(\langle s\rangle_b|\overline{b}\langle v,r\rangle|r(m).\mathbf{0})$;

identity: $\langle u_i\rangle_b \triangleq !b(m_1,\ldots,m_k,r).\overline{r}\langle m_i\rangle$. If we suppose $b:(\widetilde{\mathsf{Nat}},\mathsf{Nat}^{[\rho,1]})^{[\omega,2]}$ we have that $b;\emptyset\vdash_{\mathrm{u}}\langle u_i\rangle_b$; thus if we consider the process $(\nu b)(\langle u_i\rangle_b|\overline{b}\langle\tilde{v},r\rangle|r(m).\mathbf{0})$ with $r:\mathsf{Nat}^{[\rho,1]}$ we have that $r;r,\tilde{v}\vdash_{\mathrm{u}}(\nu b)(\langle u_i\rangle_b|\overline{b}\langle\tilde{v},r\rangle|r(m).\mathbf{0})$;

composition: Suppose that $\langle g_i\rangle_{b_i}$ is defined for every g_i, with $b_i:(\widetilde{\mathsf{Nat}},\mathsf{Nat}^{[\rho,k_{r_i}]})^{[\omega,k_{b_i}]}$ for all i in $1,\ldots,l$, and $\langle f\rangle_d$ is defined for f with $d:(\widetilde{\mathsf{Nat}},\mathsf{Nat}^{[\rho,k_r]})^{[\omega,k_d]}$, $\langle f\rangle_d$ and all $\langle g_i\rangle_{b_i}$ are well typed by inductive hypothesis and $(\nu d)(\langle f\rangle_d|\overline{d}\langle\tilde{v}',r'\rangle|r'(m).\mathbf{0})$ and $(\nu b_i)(\langle g_i\rangle_{b_i}|\overline{b_i}\langle\tilde{v}_i,r_i\rangle|r_i(m).\mathbf{0})$ are balanced. We define $\langle h\rangle_b$ as follows:

$$\begin{aligned}\langle h\rangle_b \triangleq !b(\tilde{m},r).(\nu b_1,r_1)\big(\,&\langle g_1\rangle_{b_1}\,|\,\overline{b}_1\langle\tilde{m},r_1\rangle\,|\,r_1(n_1).(\nu b_2,r_2)(\langle g_2\rangle_{b_2}\,|\,\overline{b}_2\langle\tilde{m},r_2\rangle\,|\,r_2(n_2).(\nu b_3,r_3)(\ldots\\ &|\,r_{m-1}(n_{m-1}).(\nu b_m,r_m)(\langle g_m\rangle_{b_m}\,|\,\overline{b}_m\langle\tilde{m},r_m\rangle\,|\,r_m(n_m).(\nu d)(\overline{d}\langle\tilde{n},r\rangle\,|\,\langle f\rangle_d)\\ &)\ldots)))\big).\end{aligned}$$

If we consider $b:(\widetilde{\mathsf{Nat}},\mathsf{Nat}^{[\rho,k_r]})^{[\omega,k_b]}$ with $k_b>k_d>k_r$ and $k_b>k_{b_i}>k_{r_i}$ for $i=1,\ldots,n$ then, by repeatedly applying rules (T-RES) and (T-PAR), it is easy to see that $b;\emptyset\vdash_{\mathrm{u}}\langle h\rangle_b$; thus, if $r:\mathsf{Nat}^{[\rho,k_r]}$, we get $r;r,\tilde{v}\vdash_{\mathrm{u}}(\nu b)(\langle h\rangle_b|\overline{b}\langle\tilde{v},r\rangle|r(m).\mathbf{0})$;

primitive recursion: Suppose that $\langle f\rangle_d$ is defined for f, with $d:(\widetilde{\mathsf{Nat}},\mathsf{Nat}^{[\rho,k_r]})^{[\omega,k_d]}$, and $\langle g\rangle_e$ for g, with $e:(\widetilde{\mathsf{Nat}},\mathsf{Nat}^{[\rho,k_r]})^{[\omega,k_e]}$; both $\langle f\rangle_d$ and $\langle g\rangle_e$ are well typed by inductive hypothesis and $(\nu d)(\langle f\rangle_d|\overline{d}\langle\tilde{v}_f,r\rangle|r(m).\mathbf{0})$ and $(\nu e)(\langle g\rangle_e|\overline{e}\langle\tilde{v}_g,r\rangle|r(m).\mathbf{0})$ are balanced. Define $\langle r\rangle_b$ as follows:

$$\begin{aligned}\langle r\rangle_b \triangleq !b(\tilde{m},r).\,&\texttt{if } m_1=0 \texttt{ then } (\nu d)(\langle f\rangle_d\,|\,\overline{d}\langle m_2,\ldots,m_k,r\rangle)\\ &\texttt{else } (\nu r')(\overline{b}\langle m_1-1,m_2,\ldots,m_k,r'\rangle\,|\,r'(n).(\nu e)(\langle g\rangle_e\,|\,\overline{e}\langle m_1-1,n,m_2,\ldots,m_k,r\rangle)).\end{aligned}$$

If we consider $b:(\widetilde{\mathsf{Nat}},\mathsf{Nat}^{[\rho,k_r]})^{[\omega,k_b]}$, with $k_b>k_d>k_r$ and $k_b>k_e>k_r$, then $b;b\vdash_{\mathrm{u}}\langle r\rangle_b$; moreover, if $r:\mathsf{Nat}^{[\rho,k_r]}$, we get $r;r,\tilde{v}\vdash_{\mathrm{u}}(\nu b)(\langle r\rangle_b|\overline{b}\langle\tilde{v},r\rangle|r(m).\mathbf{0})$.

In all cases $\vdash_{\mathrm{u}}$ implies $\vdash_1$ (Lemma F.1), $\Delta^\rho=\Gamma^\rho=r$ and $\Delta^\omega=\Gamma^\omega=\emptyset$, hence each G is balanced.

Now we can show the two directions of the statement:

(⇒): we show that $\forall v_1,\ldots,v_k \in \mathbb{N}$ s.t. $f(v_1,\ldots,v_k) = n \Rightarrow (\nu b)(\langle f\rangle_b | \overline{b}\langle v_1,\ldots,v_k,r\rangle | r(m).\mathbf{0}) \xrightarrow{\tau}{}^* \xrightarrow{\overline{r}\langle n\rangle}$. The proof proceeds by lexicographic induction on the pair $(|\langle f\rangle_b|, \sum_{i=1,\ldots,k} v_i)$; we distinguish the different kinds of primitive recursive functions:

zero: $\forall v \in \mathbb{N}$ we have $N(v) = 0$ and $(\nu b)(\langle n\rangle_b | \overline{b}\langle v,r\rangle | r(m).\mathbf{0}) \to (\nu b)(\langle n\rangle_b | \overline{r}\langle 0\rangle | r(m).\mathbf{0}) \xrightarrow{\overline{r}\langle 0\rangle}$;

successor: $\forall v \in \mathbb{N}$ we have $S(v) = v+1$ and $(\nu b)(\langle s\rangle_b | \overline{b}\langle v,r\rangle | r(m).\mathbf{0}) \xrightarrow{\tau} (\nu b)(\langle s\rangle_b | \overline{r}\langle v+1\rangle | r(m).\mathbf{0}) \xrightarrow{\overline{r}\langle v+1\rangle}$;

identity: $\forall v_1,\ldots,v_k \in \mathbb{N}^k$ we have $U_i^{(k)}(v_1,\ldots,v_k) = v_i$ and $(\nu b)(\langle u_i\rangle_b | \overline{b}\langle v_1,\ldots,v_k,r\rangle | r(m).\mathbf{0}) \xrightarrow{\tau} (\nu b)(\langle u_i\rangle_b | \overline{r}\langle v_i\rangle | r(m).\mathbf{0}) \xrightarrow{\overline{r}\langle v_i\rangle}$;

composition: consider k values $v_1,\ldots,v_k$ s.t. $h(v_1,\ldots,v_k) = f(g_1(v_1,\ldots,v_k),\ldots,g_l(v_1,\ldots,v_k)) = n$, that is, if we have $g_j(v_1,\ldots,v_k) = m_j$ (for $j = 1,\ldots,l$) then $h(v_1,\ldots,v_k) = f(m_1,\ldots,m_l) = n$.

Suppose $\langle g_j\rangle_{b_j}$ is the encoding of g_j; given that $|\langle g_j\rangle_{b_j}| < |\langle h\rangle_b|$, by inductive hypothesis we have $(\nu b_j)(\langle g_j\rangle_{b_j} | \overline{b}_j\langle v_1,\ldots,v_k,r_j\rangle | r_j(m).\mathbf{0}) \xrightarrow{\tau}{}^* \xrightarrow{\overline{r}_j\langle m_j\rangle}$, for $j = 1,\ldots,l$. Similarly, if $\langle f\rangle_d$ is associated to f, by inductive hypothesis we have $(\nu d)(\langle f\rangle_d | \overline{d}\langle m_1,\ldots,m_l,r'\rangle | r'(m).\mathbf{0}) \xrightarrow{\tau}{}^* \xrightarrow{\overline{r'}\langle n\rangle}$. By looking at the definition of $\langle h\rangle_b$ it is easy to see that $(\nu b)(\langle h\rangle_b | \overline{b}\langle v_1,\ldots,v_k,r\rangle | r(m).\mathbf{0}) \xrightarrow{\tau}{}^* \xrightarrow{\overline{r}\langle n\rangle}$;

primitive recursion: consider $v_1,\ldots,v_k \in \mathbb{N}^k$ we distinguish two cases:

$v_1 = 0$**:** $r(0,v_2,\ldots,v_k) = f(v_2,\ldots,v_k) = n$. Let be $\langle f\rangle_d$ the encoding of f. By inductive hypothesis, given that $|\langle f\rangle_d| < |\langle r\rangle_b|$, we have that $f(v_2,\ldots,v_k) = n$ implies $(\nu d)(\langle f\rangle_d | \overline{d}\langle v_2,\ldots,v_k,r\rangle | r(m).\mathbf{0}) \xrightarrow{\tau}{}^* \xrightarrow{\overline{r}\langle n\rangle}$.
By looking at the definition of $\langle r\rangle_b$, if $v_1 = 0$ process $(\nu b)(\langle r\rangle_b | \overline{b}\langle 0,v_2,\ldots,v_k,r\rangle | r(m).\mathbf{0})$ reduces in a step into

$$(\nu b)(\langle r\rangle_b \,|\, (\nu d)(\langle f\rangle_d | \overline{d}\langle v_2,\ldots,v_k,r\rangle) \,|\, r(m).\mathbf{0})$$

that is $(\nu b)(\langle r\rangle_b | \overline{b}\langle 0,v_2,\ldots,v_k,r\rangle | r(m).\mathbf{0}) \xrightarrow{\tau}{}^* \xrightarrow{\overline{r}\langle n\rangle}$;

$v_1 \neq 0$**:** $r(v_1,\ldots,v_k) = g(v_1-1, r(v_1-1,v_2,\ldots,v_k), v_2,\ldots,v_k)$. It holds that $(v_1+v_2+\cdots+v_k) > (v_1-1+v_2+\cdots+v_k)$. Let be $r(v_1-1,v_2,\ldots,v_k) = n_R$. By inductive hypothesis, $(\nu b)(\langle r\rangle_b | \overline{b}\langle v_1-1,v_2,\ldots,v_k,r'\rangle | r'(m).\mathbf{0}) \xrightarrow{\tau}{}^* \xrightarrow{\overline{r'}\langle n_R\rangle}$ and, if $\langle g\rangle_d$ is the encoding of g, $g(v_1-1,n_R,v_2,\ldots,v_k) = n$ implies, by inductive hypothesis ($|\langle g\rangle_d| < |\langle r\rangle_b|$), $(\nu d)(\langle g\rangle_d | \overline{d}\langle v_1-1,n_R,v_2,\ldots,v_k,r\rangle | r(m).\mathbf{0}) \xrightarrow{\tau}{}^* \xrightarrow{\overline{r}\langle n\rangle}$. Given that $v_1 \neq 0$, process $\langle r\rangle_b$ proceeds by choosing the else clause, hence $(\nu b)(\langle r\rangle_b | \overline{b}\langle v_1,v_2,\ldots,v_k,r\rangle | r(x).\mathbf{0}) \xrightarrow{\tau} {}^* \xrightarrow{\overline{r}\langle n\rangle}$.

(⇐): By contradiction, suppose that $G = (\nu b)(\langle f\rangle_b | \overline{b}\langle \tilde{v},r\rangle | r(m).\mathbf{0}) \xrightarrow{\tau}{}^* G' \xrightarrow{\overline{r}\langle n\rangle}$ and $f(\tilde{v}) = n' \neq n$. By $(\Rightarrow)$ we have that $(\nu b)(\langle f\rangle_b | \overline{b}\langle \tilde{v},r\rangle | r(m).\mathbf{0}) \xrightarrow{\tau}{}^* G'' \xrightarrow{\overline{r}\langle n'\rangle}$. Given that r is responsive, by Proposition F.1, r is not used as subject of the communication in both sequences of reductions to G' and G''. By confluence (Corollary F.1) we have that $G' \xrightarrow{\tau}{}^* \equiv Q$ and $G'' \xrightarrow{\tau}{}^* \equiv Q$, in both cases without using r as subject of the communication; hence, again by Corollary F.1, $Q \xrightarrow{\overline{r}\langle n\rangle}$ and $Q \xrightarrow{\overline{r}\langle n'\rangle}$. By Theorem 1 (subject reduction) $\Gamma;\Delta \vdash_1 P$ implies $\Gamma';\Delta' \vdash_1 Q$ for suitable Γ' and Δ'. r is responsive and $r \in \mathrm{fn}(Q)$, thus it cannot be used twice in Q, (T-PAR); hence $n' = n$.

□

G Proofs of Theorem 4 and Theorem 5

In this section we prove that the subject reduction theorem is satisfied by type system $\vdash_2$ and the intermediate results needed for proving Theorem 5 (type soundness for system $\vdash_2$). Firstly we introduce some preliminary results.

Proposition G.1 (substitution). *Suppose* $\Gamma;\Delta,x^t \vdash_2 P$, *with* $t \neq p,n$ *and* $x,b:T$, *then*

1. $b \notin \Delta$ *and* $b^m \notin \Gamma$ *imply* $\Gamma;\Delta,b^t \vdash_2 P[b/x]$;
2. $b^t \in \Delta$, $b^m \notin \Gamma$ *and* $T \neq S^{[\rho,k]}$, *imply* $\Gamma;\Delta \vdash_2 P[b/x]$.

PROOF: In both cases the proof is straightforward by induction on the derivation of $\Gamma;\Delta,x^t \vdash_2 P$. The additional constraints on b and t ensure that, in case P is a parallel composition, the premise of rule (T_+-PAR) are still satisfied after substitution. □

Lemma G.1. $P \equiv R$ *and* $\Gamma;\Delta \vdash_2 P$ *imply* $\Gamma;\Delta \vdash_2 R$.

PROOF: The proof is straightforward by induction on the derivation of $P \equiv Q$. □

Proposition G.2. $\Gamma;\Delta \vdash_2 P$ *implies:*

1. *if* $P \xrightarrow{a(c)} P'$, *with* $a : T^U$ *and* $c : T$ *then*
   (a) *if* $c \notin \Delta$ *and* $c^m \notin \Gamma$ *then* $\Gamma \ominus^+ (\{a\} \setminus \mathrm{in}(P'));\Delta,c^t \vdash_2 P'$ *with* $t \neq n,p$;
   (b) *if* $c^t \in \Delta$, $c^m \notin \Gamma$, *with* $t \neq n,p$ *and* $T \neq S^{[\rho,k]}$, *then* $\Gamma \ominus^+ (\{a\} \setminus \mathrm{in}(P'));\Delta \vdash_2 P'$;
2. *if* $P \xrightarrow{\overline{a}\langle b\rangle} P'$ *then* $\Gamma;\Delta \ominus^+ (\{a,b\} \setminus \mathrm{on}(P')) \vdash_2 P'$;
3. *if* $P \xrightarrow{\overline{a}(b)} P'$ *then either* $\Gamma;\Delta \ominus^+ (\{a\} \setminus \mathrm{on}(P')),b \vdash_2 P'$ *and* $b : I$ *or* $\Gamma,b;(\Delta,b) \ominus^+ (\{a,b\} \setminus \mathrm{on}(P')) \vdash_2 P'$ *and* $b : T \neq I$.

PROOF: The proof proceeds by induction on the derivation of $P \xrightarrow{\mu} P'$. In each case we distinguish the last transition rule applied. Omitted cases can be easily proved by applying the inductive hypothesis.

1. (IN): $a(b).P \xrightarrow{a(c)} P[c/b]$. By the premise of ($T_+$-INP) and $\Gamma,a^{t''};\Delta \vdash_2 a(b).P$ we get $a : T^{[u,k]}$ with $u \neq \omega$, $b : T$ and $\Gamma;\Delta,b^t \vdash_2 P$ with $t \neq n,p$.
   Suppose $c^m \notin \Gamma$ and $c \notin \Delta$. By $c : T$ and by Proposition G.1 (1) (substitution), $\Gamma;\Delta,c^t \vdash_2 P[c/b]$ with $t \neq n,p$ (note that $\Gamma = (\Gamma,a) \ominus^+ (\{a\} \setminus \mathrm{in}(P[c/b]))$ because $a \notin \mathrm{in}(P[c/b])$ by (T_+-INP)).
   Suppose $c^m \notin \Gamma$, $c^{t'} \in \Delta$, with $t' \neq n,p$, and $T \neq S^{[\rho,k]}$. Thus, the capabilities t and t' are univocally determined by T (if $T = S^{[u,k]}$ then if $u = \omega$ then $t = t' = -$ and if $u = \rho^+$ then $t = t' = m$). By $b,c : T$, $t = t'$ and, by Proposition G.1 (2) (substitution), $\Gamma;\Delta \vdash_2 P[c/b]$.

   (REP): $!a(b).P \xrightarrow{a(c)} !a(b).P \mid P[c/b]$. Suppose a +-responsive, if a is an ω-receptive name the proof proceeds similarly.
   By the premise of (T_+-REP^p) and $\Gamma,a^p;\Delta \vdash_2 !a(b).P$, we get $a : T^{[\rho^+,k]}$, $b : T$, $\Delta^\rho = \Delta^p = \Gamma^\rho = \Gamma^s = \Gamma^\omega = \Gamma^p = \emptyset$ and $\Gamma;\Delta,b^t \vdash_2 P$ with $t \neq n,p$.

Suppose $c^{\mathsf{m}} \notin \Gamma$ and $c \notin \Delta$. By $c : \mathsf{T}$ and by Proposition G.1 (1) (substitution), $\Gamma;\Delta, c^t \vdash_2 P[c/b]$ with $t \neq \mathsf{n},\mathsf{p}$. Rule ($\mathrm{T}_+$-PAR) can be applied for deducing $\Gamma, a^{\mathsf{p}};\Delta, c^t \vdash_2 !a(b).P \mid P[c/b]$ (note that $\Gamma, a^{\mathsf{p}} = (\Gamma, a^{\mathsf{p}}) \ominus^+ (\{a\} \setminus \mathrm{in}(!a(b).P \mid P[c/b]))$).

Suppose $c^{\mathsf{m}} \notin \Gamma$ and $c^{t'} \in \Delta$ with $t' \neq \mathsf{n},\mathsf{p}$ and $\mathsf{T} \neq \mathsf{S}^{[\mathsf{p},k]}$. As already seen for rule (IN), $t = t'$ because both are univocally determined by T. By $c : \mathsf{T}$, $t = t'$ and, by Proposition G.1 (2) (substitution), $\Gamma;\Delta \vdash_2 P[c/b]$. Rule ($\mathrm{T}_+$-PAR) can be applied for deducing $\Gamma, a^{\mathsf{p}};\Delta \vdash_2 !a(b).P \mid P[c/b]$.

Note that in case (PAR_1) the premise of the rule are guaranteed by the additional constraints $t \neq \mathsf{n},\mathsf{p}$ and $c^{\mathsf{m}} \notin \Gamma$.

2. (OUT): $\overline{a}\langle b\rangle \xrightarrow{\overline{a}\langle b\rangle} \mathbf{0}$; by the premise of ($\mathrm{T}_+$-OUT) and $\emptyset;\Delta, a^t, b^{t'} \vdash_2 \overline{a}\langle b\rangle$, we get $\Delta^{\mathsf{p}} = \Delta^{\mathsf{p}^+} = \emptyset$; hence, by ($\mathrm{T}_+$-NIL), $\emptyset;(\Delta, a^t, b^{t'}) \ominus^+ \{a,b\} \vdash_2 \mathbf{0}$;

   ($\mathrm{OUT}^{\mathsf{p}}$): $!\overline{a}\langle b\rangle \xrightarrow{\overline{a}\langle b\rangle} !\overline{a}\langle b\rangle$; and $\emptyset;\Delta, a^{\mathsf{p}}, b^- \vdash_2 !\overline{a}\langle b\rangle$ $((\Delta, a^{\mathsf{p}}, b^-) \ominus^+ (\{a,b\} \setminus \mathrm{on}(!\overline{a}\langle b\rangle))) = (\Delta, a^{\mathsf{p}}, b^-)$).

3. (OPEN): by $(\nu b)P \xrightarrow{\overline{a}(b)} P'$ and the premise of the rule, we get $P \xrightarrow{\overline{a}\langle b\rangle} P'$. Suppose ($\mathrm{T}_+$-RES) is the last rule applied in the derivation of $\Gamma;\Delta \vdash_2 (\nu b)P$. By the premise of the rule, $b : \mathsf{T}^{\mathsf{U}}$ and $\Gamma, b^t;\Delta, b^{t'} \vdash_2 P$. Moreover, by Proposition G.2 (2), $\Gamma, b^t;(\Delta, b^{t'}) \ominus^+ (\{a,b\} \setminus \mathrm{on}(P')) \vdash_2 P'$.

   If (T_+-RES-I) is the last applied then $b : \mathsf{I}$ and from $\Gamma;\Delta, b \vdash_2 P$ and Proposition G.2 (2), $\Gamma;(\Delta, b) \ominus^+ (\{a,b\} \setminus \mathrm{on}(P')) \vdash_2 P'$.

   Note that it cannot be $b : \bot$ because $b \in \mathrm{on}(P)$.

□

Lemma G.2. *Suppose* $\Gamma;\Delta \vdash_2 P$. $P \xrightarrow{\overline{a}\langle b\rangle} P'$ *and* b *responsive name imply that* $b \notin \mathrm{on}(P')$.

PROOF: The proof is straightforward by induction on the derivation of $\Gamma;\Delta \vdash_2 P$. □

Theorem G.1 (Theorem 4). $\Gamma;\Delta \vdash_2 P$ *and* $P \xrightarrow{[a]} P'$ *imply* $\Gamma';\Delta' \vdash_2 P'$, *with* $\Gamma' = \Gamma \ominus^+ (\{a\} \setminus \mathrm{in}(P'))$ *and* $\Delta' = \Delta \ominus^+ (\{a\} \setminus \mathrm{on}(P'))$.

PROOF: The proof proceeds by induction on the derivation of $P \xrightarrow{[a]} P'$; we distinguish the last transition rule applied:

(COM_1): by $P|R \xrightarrow{[a]} P'|R'$ and the premise of the rule, $P \xrightarrow{\overline{a}\langle b\rangle} P'$ and $R \xrightarrow{a(b)} R'$. By $\Gamma;\Delta \vdash_2 P|R$ and (T_+-PAR), $\Gamma = \Gamma_P \cup \Gamma_R$, $\Delta = \Delta_P \cup \Delta_R$, $\Gamma_P;\Delta_P \vdash_2 P$, $\Gamma_R;\Delta_R \vdash_2 R$, $\Gamma_P^\ell \cap \Gamma_R^\ell = \emptyset$ for $\ell = \mathsf{p},\mathsf{s},\mathsf{p}$ and $\Delta_P^{\ell'} \cap \Delta_R^{\ell'} = \emptyset$ for $\ell' = \mathsf{p},\mathsf{p}$. Moreover, $\Gamma^{\mathsf{m}} \cap \Delta^{\mathsf{m}} = \emptyset$ and $\Gamma^{\mathsf{p}} \cap \Delta^{\mathsf{p}} = \emptyset$.

By $P \xrightarrow{\overline{a}\langle b\rangle} P'$, $\Gamma_P;\Delta_P \vdash_2 P$ and Proposition G.2 (2), $\Gamma_P;\Delta_P \ominus^+ (\{a,b\} \setminus \mathrm{on}(P')) \vdash_2 P'$. $b^t \in \Delta_P$ with $t \neq \mathsf{n},\mathsf{p}$ (because b is used as object of an output and because of (T_+-OUT), (T_+-$\mathrm{OUT}^{\mathsf{p}}$)), thus either $t = -$ or $t = \mathsf{m}$ and by $\Gamma^{\mathsf{m}} \cap \Delta^{\mathsf{m}} = \emptyset$ we have that if $t = \mathsf{m}$ then $b^{\mathsf{m}} \notin \Gamma$.

Suppose $b \notin \Delta_R$. By $R \xrightarrow{a(b)} R'$, $\Gamma_R;\Delta_R \vdash_2 R$, $b^{\mathsf{m}} \notin \Gamma_R \subseteq \Gamma$ and Proposition G.2 (1), $\Gamma_R \ominus^+ (\{a\} \setminus \mathrm{in}(R'));\Delta_R, b^{t'} \vdash_2 R'$ with $t' \neq \mathsf{n},\mathsf{p}$.

Let be $\Gamma'_P = \Gamma_P$, $\Gamma'_R = \Gamma_R \ominus^+ (\{a\} \setminus \mathrm{in}(R'))$, $\Delta'_P = \Delta_P \ominus^+ (\{a,b\} \setminus \mathrm{on}(P'))$ (by Lemma G.2 if b is a responsive name then $b \notin \mathrm{on}(P')$) and $\Delta'_R = \Delta_R, b^{t'}$. Note that $t = t'$ because both are different from n and p, hence univocally determined by the type of b (that is either $t = t' = -$ or $t = t' = \mathsf{m}$).

The premise of rule (T_+-PAR) is satisfied, hence $\Gamma';\Delta' \vdash_2 P'|R'$ with $\Gamma' = \Gamma'_P \cup \Gamma'_R = \Gamma \ominus^+ (\{a\} \setminus \text{in}(P'|R'))$ and $\Delta' = \Delta \ominus^+ (\{a\} \setminus \text{on}(P'|R'))$.

Similar proof if $b^t \in \Delta_R$. Note that it cannot be $b^{t'} \in \Delta_R$ with $t \neq t'$ because otherwise $\Delta_R \cup \Delta_P$ would not be defined;

(CLOSE_1): the proof proceeds similarly. Note that in this case it must be $b \notin \Delta_R$ because b is bound in P;

(RES), (RES-ρ), (PAR_1): the proof is straightforward by inductive hypothesis.

□

We now prove the intermediate results needed for proving the responsiveness theorem. Firstly, we show that each name carrying (+-)responsive objects has level greather than the carried object's.

Lemma G.3. *Suppose P is $(\Gamma;\Delta)$-strongly balanced and $\overline{a}\langle b\rangle \in \text{O}(P)$, with b (+-)responsive, then* $\text{lev}(a) > \text{lev}(b)$.

PROOF: P $(\Gamma;\Delta)$-strongly balanced imply that $\Gamma^\rho = \Delta^\rho$, $\Delta^\omega \subseteq \Gamma^\omega$, $\Gamma^{\rho^+} \subseteq \Delta^{\rho^+}$ and $(\Delta^{\rho^+})^\dagger \subseteq (\Gamma^{\rho^+})^\dagger$ (similar comments for bound +-responsive names). Hence, a is used as input subject in P.

Suppose the (possibly guarded) subprocess that uses a as subject of an input in P is $(!)a(x).R$. By well-typedness of P, $\text{lev}(b) = \text{lev}(x)$. From ($T_+$-NIL), ($T_+$-OUT), ($T_+$-$\text{OUT}^\text{P}$) and ($T_+$-PAR), x is used in R in output either as subject or object. Moreover, this output cannot be guarded by an ω-receptive input, (T_+-REP).

First of all, we prove that P $(\Gamma;\Delta)$-strongly balanced implies that each name in P carrying (+-)responsive names cannot have level equal to 0.

Consider a and, by contradiction, suppose $\text{lev}(a) = 0$. By well typedness of P, the subprocess $(!)a(x).R$ is well-typed and $\Gamma';\Delta',x^t \vdash_2 R$, with $t \neq \mathsf{n},\mathsf{p}$, for suitable Γ' and Δ' extending $\Gamma \setminus \{a\}$ and Δ with some names in $\text{bn}(P)$. Moreover, by the typing rules for input, $\forall c \in \text{os}(R) \cup \text{is}(R)$ it holds that $\text{lev}(c) < \text{lev}(a)$. Suppose $R = \mathbf{0}$, by rule (T_+-NIL), R wouldn't be well typed because x is (+-)responsive: contradiction. If either $R = (!)d(y).R'$ or $R = \overline{d}\langle e\rangle$, it would be $\text{lev}(d) < \text{lev}(a) = 0$, and this is not possible because levels are positive integers: contradiction. Similarly, it cannot be that $R = Q \,|\, Q'$ and $R = (\nu t)Q$, with $Q, Q' ::= (!)d(y).R' \,|\, \overline{d}\langle e\rangle$. Hence $\text{lev}(a) > 0$.

We continue by proving that $\text{lev}(b) < \text{lev}(a)$; the proof proceeds by induction on $\text{lev}(a)$.

$\text{lev}(a) = 1$**:** by (T_+-INP), (T_+-REP) and (T_+-REP^P), for each $c \in (\text{os}(R) \cup \text{is}(R))$ it holds that $\text{lev}(c) < \text{lev}(a)$, hence $\text{lev}(c) = 0$. The output action involving x cannot be guarded by an input (because otherwise the subject of the output should have a negative level, by typing rules for input). Moreover, x is the subject of such an action, because we have already shown that, if $\overline{a}\langle b\rangle$ with b (+-)responsive, then $\text{lev}(a) > 0$. In conclusion, $\text{lev}(x) = 0 < \text{lev}(a)$.

$\text{lev}(a) = n$**:** Suppose x is used as subject and the output is not guarded by a replicated input ($x \in \text{os}(R)$). By (T_+-INP), (T_+-REP) and (T_+-REP^P), for each $c \in (\text{os}(R) \cup \text{is}(R))$ it holds that $\text{lev}(c) < \text{lev}(a)$, that is $\text{lev}(b) = \text{lev}(x) < \text{lev}(a)$. Suppose the output is guarded by a replicated input, let's say on d (which is +-responsive because x is free in R). $d \in \text{is}(R)$ and $\text{lev}(d) < \text{lev}(a)$. By ($T_+$-$\text{REP}^\text{P}$), $\text{lev}(b) = \text{lev}(x) < \text{lev}(d) < \text{lev}(a)$.

Suppose x is used as object of an output action, let's say $\overline{e}\langle x\rangle$. As previously seen, we have $\text{lev}(e) < \text{lev}(a)$, and by applying the inductive hypothesis $\text{lev}(b) = \text{lev}(x) < \text{lev}(e) < \text{lev}(a)$.

□

The following proposition ensures that each strongly balanced process always has an enabled reduction involving a (+-)responsive name.

Proposition G.3 (Proposition 5). *Suppose P is $(\Gamma;\Delta)$-strongly balanced with $\Delta^{\mathsf{p}} \cup \Gamma^{\mathsf{p}^+} \neq \emptyset$. Then $P \xrightarrow{\tau\langle a,b\rangle}$ with either a or b (+-)responsive.*

PROOF: Let c be the (either free or bound) (+-)responsive name with highest level appearing as input subject in P anomg those not-guarded by a replicated input on an ω-receptive name. Since P is $(\Gamma;\Delta)$-strongly balanced, c is used in output in P.

By contradiction, suppose that P cannot reduce using c as subject or object of the communication and consider the normal form (Lemma 2) of $P \equiv (\nu\tilde{d})(P_1|\cdots|P_n)$.

If P cannot reduce using c as subject or object of the communication then either every output action $(!)\overline{a}\langle b\rangle$ involving c is guarded, or each corresponding input $(!)a(x).R$ is guarded.

Suppose that all outputs $(!)\overline{a}\langle b\rangle$, with a or b equal to c, are guarded. By the premise of rule (T_+-REP), none of them can be guarded by a replicated input on an ω-receptive name. By rules (T_+-INP) and (T_+-REP$^{\mathsf{p}}$), $(!)\overline{a}\langle b\rangle$ can be guarded by an input on a (+-)responsive name, say d, only if $\mathrm{lev}(d) > \mathrm{lev}(a) \geq \mathrm{lev}(c)$ (Lemma G.3). But this is a contradiction, because c has the highest level among the (+-)responsive (free or bound) names used in input in P. Hence none of the output $(!)\overline{a}\langle b\rangle$ involving c is guarded; that is for each of them there is a $j \in \{1,\ldots,n\}$ such that $P_j = (!)\overline{a}\langle b\rangle$.

Let us now look at the inputs.

Consider any $a \neq c$ for which there is a $P_j = (!)\overline{a}\langle c\rangle$ for some $j \in \{1,\ldots,n\}$. Since a carries the (+-)responsive name c, by definition of strong balancing, a must occur in input position in P. Assume this input is not available because it is guarded. Now, a cannot be an ω-receptive name because otherwise each input on a should be not-guarded, by (T_+-INP), (T_+-REP) and (T_+-REP$^{\mathsf{p}}$). Moreover, a cannot be (+-)responsive because, by Lemma G.3, $\mathrm{lev}(a) > \mathrm{lev}(c)$ and c has the highest level among the (+-)responsive free or bound names used in input in P.

Suppose $a = c$ and that the input on c, say $c(x)$, is guarded. As previously seen, by rule (T_+-REP), $c(x)$ cannot be guarded by a replicated input on an ω-receptive name, while, by rules (T_+-INP) and (T_+-REP$^{\mathsf{p}}$), $c(x)$ can be guarded by an input on a (+-)responsive name, say d. In the latter case, from well-typedness of P, it would follow that $\mathrm{lev}(d) > \mathrm{lev}(c)$ (rules (T_+-INP) and (T_+-REP$^{\mathsf{p}}$)): but c has the highest level among the (+-)responsive free or bound names used in input in P.

In both cases we have a contradiction. In conclusion, there are P_i and P_j such that $P_i = (!)\overline{a}\langle b\rangle \xrightarrow{\overline{a}\langle b\rangle}$ and $P_j = (!)a(x).P'_j \xrightarrow{a(b)}$, with either a or b equal to c; hence $P \xrightarrow{\tau\langle a,b\rangle}$ with either a or b equal to c. □

Lemma G.4. *Suppose $x, b : T$. $\mathrm{wt}^+(P) = \mathrm{wt}^+(P[b/x])$.*

PROOF: By definition of $\mathrm{wt}^+(\cdot)$. □

Lemma G.5. *Suppose $\Gamma;\Delta \vdash_2 P$, then:*

1. *$P \xrightarrow{a(b)} P'$, with either a or b (+-)responsive name, $a : T^U$ and $b : T$, implies*
   - *(a) $\mathrm{wt}^+(P') \prec \mathrm{wt}^+(P)$ if the input on a is not replicated;*
   - *(b) $\mathrm{wt}^+(P') \prec \mathrm{wt}^+(P) + 0_{\mathrm{lev}(a)}$ if the input on a is replicated;*
2. *$P \xrightarrow{\overline{a}\langle b\rangle} P'$ ($P \xrightarrow{\overline{a}(b)} P'$) implies*

(a) $\mathrm{wt}^+(P') \preceq \mathrm{wt}^+(P) - 0_{\mathrm{lev}(a)}$ *if the output on a is not replicated;*

(b) $\mathrm{wt}^+(P') = \mathrm{wt}^+(P)$ *if the output on a is replicated.*

PROOF: In each case the proof proceeds by induction on the derivation of $P \xrightarrow{\mu} P'$; we consider the last transition rule applied.

1. *(a)*,(IN): $a(x).P \xrightarrow{a(b)} P[b/x]$; $\mathrm{wt}^+(a(x).P) = \mathrm{wt}^+(P) + 0_{\mathrm{lev}(a)}$ and $\mathrm{wt}^+(P) = \mathrm{wt}^+(P[b/x])$ (by $x, b : \mathsf{T}$ and Lemma G.4). Thus, $\mathrm{wt}^+(P[b/x]) \prec \mathrm{wt}^+(P) + 0_{\mathrm{lev}(a)} = \mathrm{wt}^+(a(x).P)$;

   (b),(REP): $!a(x).P \xrightarrow{a(b)} !a(x).P \,|\, P[b/x]$; $\mathrm{wt}^+(!a(x).P) = 0$ and $\mathrm{wt}^+(!a(x).P|P[b/x]) \prec 0_{\mathrm{lev}(a)} = \mathrm{wt}^+(!a(x).P) + 0_{\mathrm{lev}(a)}$ because of the definition of $\mathrm{wt}^+(\cdot)$ and the premise of rule (T_+-REP$^{\mathsf{P}}$) or (T_+-REP) ($\forall c \in (\mathrm{os}(P) \cup \mathrm{is}(P)) : \mathrm{lev}(c) < \mathrm{lev}(a)$).

2. *(a)*:

   (OUT): $\overline{a}\langle b\rangle \xrightarrow{\overline{a}\langle b\rangle} \mathbf{0}$, $\mathrm{wt}^+(\overline{a}\langle b\rangle) = 0_{\mathrm{lev}(a)}$ and $\mathrm{wt}^+(\mathbf{0}) = 0 = \mathrm{wt}^+(\overline{a}\langle b\rangle) - 0_{\mathrm{lev}(a)}$;

   (OPEN): $(\nu b)P \xrightarrow{\overline{a}(b)} P'$ implies $P \xrightarrow{\overline{a}\langle b\rangle} P'$; by inductive hypothesis $\mathrm{wt}^+(P') \preceq \mathrm{wt}^+(P) - 0_{\mathrm{lev}(a)} = \mathrm{wt}^+((\nu b)P) - 0_{\mathrm{lev}(a)}$;

   (b):

   (OUT$^{\mathsf{P}}$): $!\overline{a}\langle b\rangle \xrightarrow{\overline{a}\langle b\rangle} !\overline{a}\langle b\rangle$;

   (OPEN): $(\nu b)P \xrightarrow{\overline{a}(b)} P'$ implies $P \xrightarrow{\overline{a}\langle b\rangle} P'$ and by inductive hypothesis $\mathrm{wt}^+(P') = \mathrm{wt}^+(P) = \mathrm{wt}^+((\nu b)P)$.

Omitted cases can be easily proved by applying the inductive hypothesis. □

The following proposition is the analog of Proposition 2 adapted to system $\vdash_2$ and show that $\mathrm{wt}^+(\cdot)$ is a good measure because decreases after each (+-)responsive reduction.

Proposition G.4 (Proposition 4). $\Gamma;\Delta \vdash_2 P$ *and* $P \xrightarrow{\tau\langle a,b\rangle} P'$ *with either a or b (+-)responsive, implies* $\mathrm{wt}^+(P') \prec \mathrm{wt}^+(P)$.

PROOF: By induction on the derivation of $P \xrightarrow{\tau\langle a,b\rangle} P'$, the proof proceeds by distinguishing the last transition rule applied:

(COM$_1$): by $P|R \xrightarrow{\tau\langle a,b\rangle} P'|R'$ and the premise of the rule, we get $P \xrightarrow{\overline{a}\langle b\rangle} P'$ and $R \xrightarrow{a(b)} R'$. By $\Gamma;\Delta \vdash_2 P \,|\, R$ and the premise of (T_+-PAR), $\Gamma_1;\Delta_1 \vdash_2 P$ and $\Gamma_2;\Delta_2 \vdash_2 R$ for suitable $\Gamma_1, \Gamma_2, \Delta_1$ and Δ_2. We consider the following cases:

both input and output are non-replicated: by Lemma G.5 (1a,2a), $\mathrm{wt}^+(R') \prec \mathrm{wt}^+(R)$ and $\mathrm{wt}^+(P') \preceq \mathrm{wt}^+(P) - 0_{\mathrm{lev}(a)}$; that is $\mathrm{wt}^+(P'|R') = \mathrm{wt}^+(P') + \mathrm{wt}^+(R') \prec \mathrm{wt}^+(P) + \mathrm{wt}^+(R) = \mathrm{wt}^+(P|R)$;

the input is replicated: by Lemma G.5 (1b,2a), $\mathrm{wt}^+(R') \prec \mathrm{wt}^+(R) + 0_{\mathrm{lev}(a)}$ and $\mathrm{wt}^+(P') \preceq \mathrm{wt}^+(P) - 0_{\mathrm{lev}(a)}$; that is $\mathrm{wt}^+(P'|R') = \mathrm{wt}^+(P') + \mathrm{wt}^+(R') \prec \mathrm{wt}^+(P) + \mathrm{wt}^+(R) = \mathrm{wt}^+(P|R)$;

the output is replicated: by Lemma G.5 (1a,2b), $\mathrm{wt}^+(R') \prec \mathrm{wt}^+(R)$ and $\mathrm{wt}^+(P') = \mathrm{wt}^+(P)$; hence, $\mathrm{wt}^+(P'|R') = \mathrm{wt}^+(P') + \mathrm{wt}^+(R') \prec \mathrm{wt}^+(P) + \mathrm{wt}^+(R) = \mathrm{wt}^+(P|R)$;

(CLOSE$_1$): in this case the proof proceeds in a similar way.

Omitted cases can be easily proved by applying the inductive hypothesis. □

The following lemma states that strong balancing is always preserved by responsive and ω-receptive reductions, while it can be violated by +-responsive reductions, but only if a replicated input is involved. Moreover, strong balancing can be re-established by erasing the subprocess guarded by this input, without affecting well-typedness.

Lemma G.6 (Lemma 3). *Suppose P is $(\Gamma;\Delta)$-strongly balanced and $P \xrightarrow{\tau\langle a,b\rangle} P'$ with P' non strongly balanced. Assume $\Gamma';\Delta' \vdash_2 P'$, with Γ',Δ' as given by Theorem 4. Then for some R,R',b and $\tilde{d}$:*

1. $a \in (\Gamma'^{\rho^+} \setminus \Delta'^{\rho^+}) \cup (\mathrm{r_i^+}(P') \setminus \mathrm{r_o^+}(P'))$;
2. $P \equiv (\nu\tilde{d})(!a(x).R \mid \overline{a}\langle b\rangle \mid R')$ *and* $a \notin \mathrm{fn}(R,b,R')$;
3. $P' \equiv (\nu\tilde{d})(!a(x).R \mid R[b/x] \mid R')$ *and* $a \notin \mathrm{fn}(R[b/x],R')$;
4. $P'' = (\nu\tilde{d})(R[b/x] \mid R')$ *is strongly balanced.*

PROOF: By Theorem 4 (subject reduction) we have $\Gamma' = \Gamma \ominus^+ (\{a\} \setminus \mathrm{in}(P'))$ and $\Delta' = \Delta \ominus^+ (\{a\} \setminus \mathrm{on}(P'))$.

1. P' non strongly balanced means that Definition 8 is not satisfied, hence at least one of its three points does not hold.

   It cannot be $\Gamma'^{\rho} \neq \Delta'^{\rho}$ because of the linearity of responsive names (rules ($\mathrm{T_+}$-PAR), ($\mathrm{T_+}$-INP), ($\mathrm{T_+}$-REP) and ($\mathrm{T_+}$-REP$^{\mathsf{P}}$)) and $\Gamma^{\rho} = \Delta^{\rho}$.

   It cannot be $\Delta'^{\omega} \not\subseteq \Gamma'^{\omega}$ because ω-receptive names are used as subject of replicated inputs (rules ($\mathrm{T_+}$-REP), ($\mathrm{T_+}$-INP) and ($\mathrm{T_+}$-REP$^{\mathsf{P}}$)), which cannot disappear, and $\Delta^{\omega} \subseteq \Gamma^{\omega}$.

   Similarly, it cannot be neither $(\Delta^{\rho^+})^{\dagger} \not\subseteq (\Gamma^{\rho^+})^{\dagger}$ nor $(\mathrm{r_o^+}(P))^{\dagger} \not\subseteq (\mathrm{r_i^+}(P))^{\dagger}$, because +-responsive names carrying (+)-responsive objects are used as subject of replicated inputs (by ($\mathrm{T_+}$-INP)), which cannot disappear.

   In conclusion, $a \in (\Gamma'^{\rho^+} \setminus \Delta'^{\rho^+}) \cup (\mathrm{r_i^+}(P') \setminus \mathrm{r_o^+}(P'))$.

2. We firstly prove that a is used (only) as subject of a replicated input in P. By contradiction, assume a is used as subject of non-replicated inputs in P. Then there are at least two of such inputs in P, because otherwise it cannot be $a \in (\Gamma'^{\rho^+} \setminus \Delta'^{\rho^+}) \cup (\mathrm{r_i^+}(P') \setminus \mathrm{r_o^+}(P'))$. Therefore a should have capability m in Γ. Hence, by strong balancing and rule ($\mathrm{T_+}$-PAR) (condition $\Gamma^{\mathsf{m}} \cap \Delta^{\mathsf{m}} = \emptyset$) a has to be used as subject of a replicated output (which cannot disappear), that is $a \in \Delta'^{\rho^+} \cup \mathrm{r_o^+}(P')$ and this is not the case. Thus, a is used as subject of a replicated input in P, hence in P'. By ($\mathrm{T_+}$-PAR) and ($\mathrm{T_+}$-REP$^{\mathsf{P}}$), a is used once in input subject position. Moreover, $P \xrightarrow{[a]}$ implies that such input cannot be guarded. Similarly, there should be a unique simple not-guarded output $\overline{a}\langle b\rangle$ in P, with $b \neq a$ (recall that we do not consider recursive types, hence channels cannot carry themselves).

   Therefore, $P \equiv (\nu\tilde{d})(!a(x).R \mid \overline{a}\langle b\rangle \mid R')$ and by Lemma G.1 ($\mathrm{T_+}$-PAR) and ($\mathrm{T_+}$-REP$^{\mathsf{P}}$) $a \notin \mathrm{fn}(R,b,R')$ ($a \notin \mathrm{fn}(R)$ is implied by the premise of ($\mathrm{T_+}$-REP$^{\mathsf{P}}$) and Lemma G.3).

3. By point (2) and the reduction $P \xrightarrow{\tau\langle a,b\rangle} P'$.

4. By points (1,2,3) and $a \notin \mathrm{on}(P')$, we get $\Gamma;\Delta \ominus^+ \{a\} \vdash_2 P' \equiv (\nu \tilde{d})(!a(x).R \,|\, R[b/x] \,|\, R')$.

   Suppose $a \in \mathrm{fn}(P)$ (hence $a \in \mathrm{fn}(P')$). By the typing rules for restriction (suppose for simplicity $\tilde{d}$ does not contain inert names) $\Gamma, \tilde{d}; \Delta \ominus^+ \{a\}, \tilde{d} \vdash_2 !a(x).R \,|\, R[b/x] \,|\, R'$. By ($\mathrm{T}_+$-PAR), $\Gamma, \tilde{d} = \Gamma_1 \cup \{a\} \cup \Gamma_2$ and $\Delta \ominus^+ \{a\}, \tilde{d} = \Delta_1 \cup \Delta_2$ with $\Gamma_1, a; \Delta_1 \vdash_2 !a(x).R$ and $\Gamma_2; \Delta_2 \vdash_2 R[b/x] \,|\, R'$. Moreover, by the premise of (T_+-REP$^\mathrm{P}$), it should be $\Gamma_1 \subseteq \Gamma_2$ and $\Delta_1 \subseteq \Delta_2$, hence $\Gamma_2 = \Gamma \ominus^+ \{a\}, \tilde{d}$ and $\Delta_2 = \Delta \ominus^+ \{a\}, \tilde{d}$. Again by the typing rules for restriction, $\Gamma \ominus^+ \{a\}; \Delta \ominus^+ \{a\} \vdash_2 (\nu \tilde{d})(R[b/x] \,|\, R') = P''$ and P'' is strongly balanced.

   The proof proceeds similarly in case $a \in \tilde{d}$. Note that in this case $\Gamma; \Delta \vdash_2 (\nu \tilde{d})(R[b/x] \,|\, R')$ follows by applying (T_+-WEAK-Γ) and (T_+-WEAK-Δ).

□

H Proof of Theorem 6

In this section, the encoding of ORC introduced in Section 9.2 is shown to be correct. In what follows, given an ORC term f, we write $\mathrm{fv}(f)$ for the set of free variables in f and $P \xrightarrow{\hat{\mu}} P'$ stands for $P \xrightarrow{\mu} P'$, if $\mu \neq \tau$ and for either $P \xrightarrow{\tau} P'$ or $P = P'$, if $\mu = \tau$. In the following, we first recall the definition of *expansion preorder*, $\gtrsim$, and of *strong bisimulation relation*, $\sim$.

Definition H.1 (expansion preorder). *A relation $\mathcal{R} \subseteq \mathcal{P} \times \mathcal{P}$ is an expansion preorder if $S \mathcal{R} P$ implies:*

1. *whenever $S \xrightarrow{\mu} S'$, there exists P' s.t. $P \xrightarrow{\hat{\mu}} P'$ and $S' \mathcal{R} P'$;*
2. *whenever $P \xrightarrow{\mu} P'$, there exists S' s.t. $S \xRightarrow{\mu} S'$ and $S' \mathcal{R} P'$.*

We say that S expands P, written $S \gtrsim P$, if $S \mathcal{R} P$ for some expansion $\mathcal{R}$.

Definition H.2 (strong bisimulation). *A symmetric relation $\mathcal{R} \subseteq \mathcal{P} \times \mathcal{P}$ is a strong bisimulation if $S \mathcal{R} P$ implies that whenever $S \xrightarrow{\mu} S'$, there exists P' such that $P \xrightarrow{\mu} P'$ and $S' \mathcal{R} P'$. We say that S is strongly bisimilar to P, written $S \sim P$, if $S \mathcal{R} P$ for some strong bisimulation $\mathcal{R}$.*

The following lemmas introduce some properties of $\sim$ and $\gtrsim$ that are useful for proving the correctness of the encoding. The (omitted) proofs rely on asynchrony and input locality of the calculus.

Lemma H.1.

1. $(\nu x)(!\overline{x}\langle c\rangle \,|\, P_1 \,|\, P_2) \sim (\nu x)(!\overline{x}\langle c\rangle \,|\, P_1) \,|\, (\nu x)(!\overline{x}\langle c\rangle \,|\, P_2)$ *if* $x \notin \mathrm{on}(P_1, P_2)$;
2. $(\nu a)(!a(y).P \,|\, P_1 \,|\, P_2) \sim (\nu a)(!a(y).P \,|\, P_1) \,|\, (\nu a)(!a(y).P \,|\, P_2)$ *if* $a \notin \mathrm{in}(P_1, P_2)$;
3. $(\nu x)(!\overline{x}\langle c\rangle \,|\, !a(y).P) \sim !a(y).(\nu x)(!\overline{x}\langle c\rangle \,|\, P)$ *if* $a, y \neq x$;
4. $(\nu x)(!x(z).P' \,|\, !a(y).P) \sim !a(y).(\nu x)(!x(z).P' \,|\, P)$ *if* $a, y \neq x$ *and* $a, y \notin \mathrm{fn}(P')$.

Lemma H.2. *$P' \gtrsim P$ implies:*

1. $P' \,|\, R \gtrsim P \,|\, R$;
2. $(\nu \tilde{d}) P' \gtrsim (\nu \tilde{d}) P$;
3. $\alpha.P' \gtrsim \alpha.P$ *with either* $\alpha = !a(y)$ *or* $\alpha = a(y)$.

In the following proofs, recall that given an ORC term f, in $[\![f]\!]_s$ all site and expression names are used only in output subject position and all variables only in input subject position. Moreover, if f is a closed term $[\![f]\!]_s$ can interact with the environment only by calling sites or expressions or by publishing (outputting) on s.

Proposition H.1. $(\nu\tilde{d})(D \mid (\nu x)(!\overline{x}\langle c\rangle \mid [\![g]\!]_s)) \gtrsim (\nu\tilde{d})(D \mid [\![g[c/x]]\!]_s)$.

PROOF: The proof is straightforward by induction on the structure of g. In cases $g ::= \mathtt{let}(p) \,|\, M(p) \,|\, E(p)$ we get $(\nu\tilde{d})(D \mid (\nu x)(!\overline{x}\langle c\rangle \mid [\![g]\!]_s)) \sim (\nu\tilde{d})(D \mid [\![g[c/x]]\!]_s)$ because $x \notin \mathrm{fn}([\![g]\!]_s)$. In cases $g ::= \mathtt{let}(x) \,|\, E(x) \,|\, M(x)$ the proof proceeds by defining a suitable relation $\mathcal{R}$ containing the pair $\langle (\nu\tilde{d})(D \mid (\nu x)(!\overline{x}\langle c\rangle \mid [\![g]\!]_s)) \;;\; (\nu\tilde{d})(D \mid [\![g[c/x]]\!]_s)\rangle$ and by showing that $\mathcal{R}$ is an expansion preorder up to $\sim$. In cases $g ::= (f \mid f') \,|\, f > y > f' \,|\, f \;\mathtt{where}\; y :\in f'$ the proof proceeds by performing some algebraic manipulation. The most interesting cases are described below.

$g = f > y > f'$**:**

$$
\begin{array}{ll}
& (\nu\tilde{d})\big(D \mid (\nu x)(!\overline{x}\langle c\rangle \mid [\![f > y > f']\!]_s)\big) \\
= & \text{(by definition of } [\![f]\!]_s) \\
& (\nu\tilde{d})\Big(D \mid (\nu x)\big(!\overline{x}\langle c\rangle \mid (\nu w)([\![f]\!]_w \mid !w(z).(\nu y)(!\overline{y}\langle z\rangle \mid [\![f']\!]_s))\big)\Big) \\
\sim & \text{(by Lemma H.1 (1,2))} \\
& (\nu w)\Big((\nu\tilde{d})\big(D \mid (\nu x)(!\overline{x}\langle c\rangle \mid [\![f]\!]_w)\big) \mid (\nu\tilde{d})\big(D \mid (\nu x)(!\overline{x}\langle c\rangle \mid !w(z).(\nu y)(!\overline{y}\langle z\rangle \mid [\![f']\!]_s))\big)\Big) \\
\sim & \text{(by Lemma H.1 (3,4))} \\
& (\nu w)\Big((\nu\tilde{d})\big(D \mid (\nu x)(!\overline{x}\langle c\rangle \mid [\![f]\!]_w)\big) \mid !w(z).(\nu\tilde{d},y)\big(!\overline{y}\langle z\rangle \mid D \mid (\nu x)(!\overline{x}\langle c\rangle \mid [\![f']\!]_s)\big)\Big) \\
\gtrsim & \text{(by inductive hypothesis and Lemma H.2)} \\
& (\nu w)\Big((\nu\tilde{d})(D \mid [\![f[c/x]]\!]_w) \mid !w(z).(\nu\tilde{d},y)(!\overline{y}\langle z\rangle \mid D \mid [\![f'[c/x]]\!]_s)\Big) \\
\sim & \text{(by Lemma H.1 (2))} \\
& (\nu\tilde{d})(D \mid (\nu w)([\![f[c/x]]\!]_w \mid !w(z).(\nu y)(!\overline{y}\langle z\rangle \mid [\![f'[c/x]]\!]_s))) \\
= & \text{(by definition of } [\![f]\!]_s) \\
& (\nu\tilde{d})(D \mid [\![(f > y > f')[c/x]]\!]_s \;.
\end{array}
$$

$g = f \,\texttt{where}\, y :\in f'$:

$$
\begin{array}{ll}
& (\nu \tilde{d})\big(D \,|\, (\nu x)(!\overline{x}\langle c\rangle \,|\, [\![f \,\texttt{where}\, y :\in f']\!]_s)\big) \\
= & \text{(by definition of } [\![f]\!]_s) \\
& (\nu \tilde{d})\Big(D \,|\, (\nu x)\big(!\overline{x}\langle c\rangle \,|\, (\nu w)([\![f']\!]_w \,|\, (\nu y)(w(z).!\overline{y}\langle z\rangle \,|\, [\![f]\!]_s))\big)\Big) \\
\sim & \text{(by Lemma H.1 (1,2))} \\
& (\nu w)\Big((\nu \tilde{d})\big(D \,|\, (\nu x)(!\overline{x}\langle c\rangle \,|\, [\![f']\!]_w)\big) \,|\, (\nu \tilde{d})\big(D \,|\, (\nu x,y)(!\overline{x}\langle c\rangle \,|\, w(z).!\overline{y}\langle z\rangle \,|\, [\![f]\!]_s)\big)\Big) \\
\sim & (\mathrm{fn}(w(z).!\overline{y}\langle z\rangle) \cap \{\tilde{d}, x\} = \emptyset) \\
& (\nu w)\Big((\nu \tilde{d})\big(D \,|\, (\nu x)(!\overline{x}\langle c\rangle \,|\, [\![f']\!]_w)\big) \,|\, (\nu y)\big(w(z).!\overline{y}\langle z\rangle \,|\, (\nu \tilde{d})(D \,|\, (\nu x)(!\overline{x}\langle c\rangle \,|\, [\![f]\!]_s))\big)\Big) \\
\gtrsim & \text{(by inductive hp. and Lemma H.2 (1,2))} \\
& (\nu w)\Big((\nu \tilde{d})\big(D \,|\, [\![f'[c/x]]\!]_w \,|\, (\nu y)(w(z).!\overline{y}\langle z\rangle \,|\, (\nu \tilde{d})(D \,|\, [\![f[c/x]]\!]_s))\big)\Big) \\
\sim & \text{(by Lemma H.1 (2))} \\
& (\nu \tilde{d})\Big(D \,|\, (\nu w)([\![f'[c/x]]\!]_w \,|\, (\nu y)(w(z).!\overline{y}\langle z\rangle \,|\, [\![f[c/x]]\!]_s))\Big) \\
= & \text{(by definition of } [\![f]\!]_s) \\
& (\nu \tilde{d})(D \,|\, [\![(f \,\texttt{where}\, y :\in f')[c/x]]\!]_s)\ .
\end{array}
$$

□

Proposition H.2. *Suppose D is a set of function and site definitions. $(\nu \tilde{d},y)(D|[\![f]\!]_y|P) \gtrsim (\nu \tilde{d})(D|P)$ if $y \notin \mathrm{fn}(P)$, $\tilde{d} = \mathrm{in}(D)$, $\tilde{d} \cap \mathrm{in}(P) = \emptyset$ and f is closed.*

PROOF: By Lemma H.1 (2) $(\nu \tilde{d},y)(D|[\![f]\!]_y|P) \sim (\nu \tilde{d},y)(D|[\![f]\!]_y) \,|\, (\nu \tilde{d})(D|P)$. Moreover, $(\nu \tilde{d},y)(D|[\![f]\!]_y) \gtrsim \mathbf{0}$ because $\mathrm{fn}([\![f]\!]_y) \subseteq \{\tilde{d},y\}$ and, by definition of $[\![\cdot]\!]_y$, name y cannot be extruded. Hence, by Lemma H.2 (2), $(\nu \tilde{d},y)(D|[\![f]\!]_y|P) \gtrsim (\nu \tilde{d})(D|P)$. □

The following proposition is a first step towards proving the correctness of the encoding.

In what follows λ represents a generic ORC's label and can be either $!c$ or τ. We define $[\![\lambda]\!]_s$ and $[\![\mu]\!]^{-1}$ as follows: $[\![!c]\!]_s = \overline{s}\langle c\rangle$, $[\![\tau]\!]_s = \tau$, $[\![\overline{s}\langle c\rangle]\!]^{-1} = !c$ and $[\![\tau]\!]^{-1} = \tau$.

Proposition H.3. *Let f be a closed* ORC *term.*

1. *$f \overset{\lambda}{\hookrightarrow} g$ implies $(\nu \tilde{d})(D \,|\, [\![f]\!]_s) \xrightarrow{[\![\lambda]\!]_s} \gtrsim (\nu \tilde{d})(D \,|\, [\![g]\!]_s)$;*
2. *$(\nu \tilde{d})(D \,|\, [\![f]\!]_s) \xrightarrow{\mu} (\nu \tilde{d})(D|P)$ implies $f \overset{[\![\mu]\!]^{-1}}{\hookrightarrow} g$, with $(\nu \tilde{d})(D|P) \gtrsim (\nu \tilde{d})(D \,|\, [\![g]\!]_s)$;*
3. *$f \overset{!c}{\hookrightarrow}$ implies $[\![f]\!]_s \xrightarrow{\overline{s}\langle c\rangle}$;*
4. *$[\![f]\!]_s \xrightarrow{\overline{s}\langle c\rangle}$ implies $f \overset{!c}{\hookrightarrow}$.*

PROOF:

1. This case is straightforward by induction on the derivation of $f \overset{\lambda}{\hookrightarrow} g$. The base cases are (PUB), (SITE) and (DEF). In the other cases the result is obtained by applying the inductive hypothesis and Lemma H.2. Moreover, in cases (SEQ2) and (WH2) also Proposition H.1 and H.2 are applied.

2. The proof proceeds by induction on the derivation of $\xrightarrow{\mu}$, by considering only closed ORC terms. The most interesting cases are sequential composition and asymmetric parallel composition. In the other cases the proof proceeds by applying the inductive hypothesis and Lemma H.2.

   $[\![f > x > g]\!]_s$**:** $(\nu \tilde{d})\big(D \mid (\nu y)([\![f]\!]_y \mid !y(z).(\nu x)(!\overline{x}\langle z\rangle \mid [\![g]\!]_s))\big) \xrightarrow{\mu} (\nu \tilde{d})(D \mid P)$ implies $(\nu \tilde{d})(D \mid [\![f]\!]_y) \xrightarrow{\mu'} (\nu \tilde{d})(D \mid P')$. By induction, $f \overset{[\![\mu']\!]^{-1}}{\hookrightarrow} f'$ and $(\nu \tilde{d})(D \mid P') \gtrsim (\nu \tilde{d})(D \mid [\![f']\!]_y)$. We distinguish two cases depending on μ':

   $\mu' \neq \overline{y}\langle c\rangle$**:** in this case $f \overset{[\![\mu']\!]^{-1}}{\hookrightarrow} f'$ implies, by (SEQ1), $f > x > g \overset{[\![\mu']\!]^{-1}}{\hookrightarrow} f' > x > g$; moreover, $(\nu \tilde{d})(D \mid P) = (\nu \tilde{d})\big(D \mid (\nu y)\big(P' \mid !y(z).(\nu x)(!\overline{x}\langle z\rangle \mid [\![g]\!]_s)\big)\big) \gtrsim (\nu \tilde{d})\big(D \mid (\nu y)\big([\![f']\!]_y \mid !y(z).(\nu x)(!\overline{x}\langle z\rangle \mid [\![g]\!]_s)\big)\big) = (\nu \tilde{d})(D \mid [\![f' > x > g]\!]_s)$ (Lemma H.2);

   $\mu' = \overline{y}\langle c\rangle$**:** in this case, by induction, $f \overset{!c}{\hookrightarrow} f'$ and, by (SEQ2), $f > x > g \overset{\tau}{\hookrightarrow} (f' > x > g) \mid g[c/x]$. By $(\nu \tilde{d})(D \mid P') \gtrsim (\nu \tilde{d})(D \mid [\![f']\!]_y)$, Lemma H.2 and Proposition H.1:

   $$\begin{aligned}
   & (\nu \tilde{d})(D \mid P) \\
   = \;& (\nu \tilde{d})\big(D \mid (\nu y)\big(P' \mid !y(z).(\nu x)(!\overline{x}\langle z\rangle \mid [\![g]\!]_s) \mid (\nu x)(!\overline{x}\langle c\rangle \mid [\![g]\!]_s)\big)\big) \\
   \gtrsim \;& (\nu \tilde{d})\big(D \mid (\nu y)\big([\![f']\!]_y \mid !y(z).(\nu x)(!\overline{x}\langle z\rangle \mid [\![g]\!]_s) \mid (\nu x)(!\overline{x}\langle c\rangle \mid [\![g]\!]_s)\big)\big) \\
   = \;& (\nu \tilde{d})(D \mid [\![f' > x > g]\!]_s \mid (\nu x)(!\overline{x}\langle c\rangle \mid [\![g]\!]_s)) \\
   \gtrsim \;& (\nu \tilde{d})(D \mid [\![f' > x > g]\!]_s \mid [\![g[c/x]]\!]_s)\,;
   \end{aligned}$$

   $[\![f \,\texttt{where}\, x :\in g]\!]_s$**:** $(\nu \tilde{d})\big(D \mid (\nu y)\big([\![g]\!]_y \mid (\nu x)(y(z).!\overline{x}\langle z\rangle \mid [\![f]\!]_s)\big)\big) \xrightarrow{\mu} (\nu \tilde{d})(D \mid P)$; we distinguish the following cases:

   $(\nu \tilde{d})(D \mid [\![g]\!]_y) \xrightarrow{\mu} (\nu \tilde{d})(D \mid P')$ **with** $\mu \neq \overline{y}\langle c\rangle$**:** by applying the inductive hypothesis, $g \overset{[\![\mu]\!]^{-1}}{\hookrightarrow} g'$ and $(\nu \tilde{d})(D \mid P') \gtrsim (\nu \tilde{d})(D \mid [\![g']\!]_y)$.
   Moreover, by Lemma H.2:

   $$\begin{aligned}
   & (\nu \tilde{d})(D \mid P) \\
   = \;& (\nu \tilde{d})\big(D \mid (\nu y)\big(P' \mid (\nu x)(y(z).!\overline{x}\langle z\rangle \mid [\![f]\!]_s)\big)\big) \\
   \gtrsim \;& (\nu \tilde{d})\big(D \mid (\nu y)\big([\![g']\!]_y \mid (\nu x)(y(z).!\overline{x}\langle z\rangle \mid [\![f]\!]_s)\big)\big) \\
   = \;& (\nu \tilde{d})(D \mid [\![f \,\texttt{where}\, x :\in g']\!]_s)
   \end{aligned}$$

   and $g \overset{[\![\mu]\!]^{-1}}{\hookrightarrow} g'$ implies, by (WH1), $f \,\texttt{where}\, x :\in g \overset{[\![\mu]\!]^{-1}}{\hookrightarrow} f \,\texttt{where}\, x :\in g'$;

   $(\nu \tilde{d})(D \mid [\![f]\!]_s) \xrightarrow{\mu} (\nu \tilde{d})(D \mid P')$**:** in this case the proof proceeds in a similar way;

   $(\nu \tilde{d})(D \mid [\![g]\!]_y) \xrightarrow{\overline{y}\langle c\rangle} (\nu \tilde{d})(D \mid P')$**:** by induction, $g \overset{!c}{\hookrightarrow} g'$, $(\nu \tilde{d})(D \mid P') \gtrsim (\nu \tilde{d})(D \mid [\![g']\!]_y)$ and $f \,\texttt{where}\, x :\in g \overset{\tau}{\hookrightarrow} f[c/x]$, by (WH2).
   Moreover, $(\nu \tilde{d})(D \mid P) = (\nu \tilde{d})\big(D \mid (\nu y)\big(P' \mid (\nu x)(!\overline{x}\langle c\rangle \mid [\![f]\!]_s)\big)\big) \gtrsim (\nu \tilde{d})\big(D \mid (\nu y)\big([\![g']\!]_y \mid (\nu x)(!\overline{x}\langle c\rangle \mid [\![f]\!]_s)\big)\big) \gtrsim (\nu \tilde{d})(D \mid [\![f[c/x]]\!]_s)$ by Proposition H.1 and Proposition H.2 (recall that f is a closed term and $y \notin \mathrm{fn}([\![f]\!]_s)$.)

3. By induction on transitions, we distinguish the following cases:

$\texttt{let}(c) \overset{!c}{\hookrightarrow}$: $[\![\texttt{let}(c)]\!]_s = \overline{s}\langle c\rangle \xrightarrow{\overline{s}\langle c\rangle}$;

$f \mid g \overset{!c}{\hookrightarrow}$: implies, by either (PAR1) or (PAR2), either $f \overset{!c}{\hookrightarrow}$ or $g \overset{!c}{\hookrightarrow}$; by induction either $[\![f]\!]_s \xrightarrow{\overline{s}\langle c\rangle}$ or $[\![g]\!]_s \xrightarrow{\overline{s}\langle c\rangle}$ and $[\![f \mid g]\!]_s = [\![f]\!]_s \mid [\![g]\!]_s \xrightarrow{\overline{s}\langle c\rangle}$;

$g \,\texttt{where}\, x :\in f \overset{!c}{\hookrightarrow}$: implies, (WH3), $g \overset{!c}{\hookrightarrow}$, and by induction $[\![g]\!]_s \xrightarrow{\overline{s}\langle c\rangle}$. $[\![g \,\texttt{where}\, x :\in f]\!]_s = (\nu y)\big([\![f]\!]_y \mid (\nu x)(y(z).!\overline{x}\langle z\rangle \mid [\![g]\!]_s)\big)$ and $(\nu y)\big([\![f]\!]_y \mid (\nu x)(y(z).!\overline{x}\langle z\rangle \mid [\![g]\!]_s)\big) \xrightarrow{\overline{s}\langle c\rangle}$.

4. We distinguish the following cases:

   $[\![\texttt{let}(c)]\!]_s \xrightarrow{\overline{s}\langle c\rangle}$: $[\![\texttt{let}(c)]\!]_s = \overline{s}\langle c\rangle \xrightarrow{\overline{s}\langle c\rangle}$ and $\texttt{let}(c) \overset{!c}{\hookrightarrow}$, (PUB);

   $[\![f \mid g]\!]_s \xrightarrow{\overline{s}\langle c\rangle}$: $[\![f \mid g]\!]_s \xrightarrow{\overline{s}\langle c\rangle}$ implies either $[\![f]\!]_s \xrightarrow{\overline{s}\langle c\rangle}$ or $[\![g]\!]_s \xrightarrow{\overline{s}\langle c\rangle}$. By induction, either $f \overset{!c}{\hookrightarrow}$ or $g \overset{!c}{\hookrightarrow}$, hence $f \mid g \overset{!c}{\hookrightarrow}$, by either (PAR1) or (PAR2);

   $[\![g \,\texttt{where}\, x :\in f]\!]_s \xrightarrow{\overline{s}\langle c\rangle}$: $(\nu y)\big([\![f]\!]_y \mid (\nu x)(y(z).!\overline{x}\langle z\rangle \mid [\![g]\!]_s)\big) \xrightarrow{\overline{s}\langle c\rangle}$ implies $[\![g]\!]_s \xrightarrow{\overline{s}\langle c\rangle}$ and by induction $g \overset{!c}{\hookrightarrow}$, that is $g \,\texttt{where}\, x :\in f \overset{!c}{\hookrightarrow}$, (WH3).

□

Proposition H.4. *Consider an* ORC *term f and suppose D_f well typed. If $\mathrm{fv}(f) = \tilde{x}$, then $F = (\nu \tilde{d}, \tilde{x})([\![f]\!]_s \mid \prod_{x\in\tilde{x}} !\overline{x}\langle c\rangle \mid D_f \mid !s(x).\mathbf{0})$, with $\mathrm{fn}(F) = \{s\}$, c inert and $\tilde{d}$, $\tilde{x}$ and s +-responsive names, is strongly balanced.*

PROOF: Well typedness of $[\![f]\!]_s$ is easy to prove by induction on the structure of f. In particular $\Gamma;\Delta \vdash_2 [\![f]\!]_s$, for suitable Γ and Δ such that $\Gamma^{\mathrm{p}} = \emptyset$, $\mathrm{dom}(\Gamma) = \mathrm{fv}(f)$, each name in Γ is annotated with capability m and $\mathrm{dom}(\Delta)$ contains only s and expression and site names, annotated with m. Hence well-typedness of F is ensured. Balancing of F may be proved by induction on the structure of f.

As an example, we consider the case $f = g_2 \,\texttt{where}\, y :\in g_1$. In this case

$$F = (\nu \tilde{x}, \tilde{d})\Big((\nu r)\big([\![g_1]\!]_r \mid (\nu y)(r(z).!\overline{y}\langle z\rangle \mid [\![g_2]\!]_s)\big) \mid \prod_{x\in\tilde{x}} !\overline{x}\langle c\rangle \mid D_f \mid !s(x)\Big)$$

where $\tilde{x} = \tilde{x}_1 \cup \tilde{x}_2$, with $\tilde{x}_1 = \mathrm{fv}(g_1)$ and $\tilde{x}_2 = \mathrm{fv}(g_2) \setminus \{y\}$, and $\tilde{d} = \tilde{d}_1 \cup \tilde{d}_2$, with $\tilde{d}_1$ and $\tilde{d}_2$ containing all names corresponding to sites and expressions called respectively by g_1 and g_2 (hence $D_f \sim D_1 | D_2$).

By induction, $G_1 = (\nu \tilde{d}_1, \tilde{x}_1)([\![g_1]\!]_r \mid \prod_{x\in\tilde{x}_1} !\overline{x}\langle c\rangle \mid D_1 \mid !r(z).\mathbf{0})$ and $G_2 = (\nu \tilde{d}_2, \tilde{x}_2, y)([\![g_2]\!]_s \mid \prod_{x\in\tilde{x}_2} !\overline{x}\langle c\rangle \mid !\overline{y}\langle c\rangle \mid D_2 \mid !s(x).\mathbf{0})$ are strongly balanced.

Note that in G_1 channel r is +-responsive and does not carry (+-)responsive names, hence if we replace $!r(z).\mathbf{0}$ with $r(z).\mathbf{0}$ then the resulting G_1' is still strongly-balanced. Thus, given that g_1 and g_2 can share only sites, expression names and variables (which are used only in output – resp. input – in $[\![g_1]\!]_r$ and $[\![g_2]\!]_s$ and replicated in input in D – resp. replicated in output in $\prod !\overline{x}\langle c\rangle$):

$$G_1'|G_2 \equiv (\nu \tilde{d}, \tilde{x}, y)\big([\![g_1]\!]_r \mid \prod_{x\in\tilde{x}_1} !\overline{x}\langle c\rangle \mid D_1 \mid D_2 \mid r(z).\mathbf{0} \mid [\![g_2]\!]_s \mid \prod_{x\in\tilde{x}_2} !\overline{x}\langle c\rangle \mid !\overline{y}\langle c\rangle \mid !s(x).\mathbf{0}\big) \triangleq G'.$$

Hence, given that G_1 and G_2 are strongly balanced, the process G' above is strongly balanced too. Similarly, the process G defined below, obtained from G' by applying the scope extension structural law to (νy) and by replacing D_1 and D_2 by D_f in such a manner to eliminate possible duplicate of site and expression definitions, is strongly balanced too:

$$G \stackrel{\triangle}{=} (\nu \tilde{d},\tilde{x})\Big((\nu y)\big([\![g_1]\!]_r \mid r(z).\mathbf{0} \mid !\overline{y}\langle c\rangle \mid [\![g_2]\!]_s\big) \mid \prod_{x\in\tilde{x}} !\overline{x}\langle c\rangle \mid D_f \mid !s(x).\mathbf{0}\Big)\ .$$

Finally, the process F below, obtained by bounding name r and by replacing $r(z).\mathbf{0} \mid !\overline{y}\langle c\rangle$ with $r(z).!\overline{y}\langle z\rangle$, is strongly balanced too. In fact, the balancing conditions are not influenced by restricting and by prefixing:

$$F \stackrel{\triangle}{=} (\nu \tilde{d},\tilde{x})\Big((\nu r)\big([\![g_1]\!]_r \mid (\nu y)(r(z).!\overline{y}\langle z\rangle \mid [\![g_2]\!]_s)\big) \mid \prod_{x\in\tilde{x}} !\overline{x}\langle c\rangle \mid D_f \mid !s(x).\mathbf{0}\Big)\ .$$

□

Theorem H.1 (Theorem 6). *Let f be a closed* ORC *term and suppose D_f is well typed.*

1. *$[\![f]\!]_s$ is well-typed and $F \stackrel{\triangle}{=} (\nu \tilde{d})([\![f]\!]_s \,|\, D_f \,|\, !s(x).\mathbf{0})$, with s and $\tilde{d}$ +-responsive, is strongly balanced;*

2. *$f \stackrel{!c}{\Rightarrow}$ if and only if $F \stackrel{\tau\langle s,c\rangle}{\Longrightarrow}$.*

PROOF:

1. Well-typedness of $[\![f]\!]_s$ and balancing of F follow by Proposition H.4.

2. ($\Rightarrow$)**:** $f \stackrel{!c}{\Rightarrow}$ means that $f \stackrel{\tau}{\hookrightarrow}{}^* g \stackrel{!c}{\hookrightarrow}$; by Proposition H.3 (1), $f \stackrel{\tau}{\hookrightarrow} f'$ implies $(\nu \tilde{d})(D_f \,|\, [\![f]\!]_s) \stackrel{\tau}{\rightarrow} (\nu \tilde{d})(D_f \,|\, P') \gtrsim (\nu \tilde{d})(D_f \,|\, [\![f']\!]_s)$, $f' \stackrel{\tau}{\hookrightarrow} f''$ implies $(\nu \tilde{d})(D_f \,|\, [\![f']\!]_s) \stackrel{\tau}{\rightarrow} (\nu \tilde{d})(D_f \,|\, P'') \gtrsim (\nu \tilde{d})(D_f \,|\, [\![f'']\!]_s)$ and by the definition and the transitivity of $\gtrsim$, $(\nu \tilde{d})(D_f|P') \stackrel{\tau}{\rightarrow} (\nu \tilde{d})(D_f|P''') \gtrsim (\nu \tilde{d})(D_f|[\![f'']\!]_s)$. This reasoning can be iterated for each τ transition from f to g. Thus, $f \stackrel{\tau}{\hookrightarrow}{}^* g$ implies $(\nu \tilde{d})(D_f \,|\, [\![f]\!]_s) \stackrel{\tau}{\rightarrow}{}^* (\nu \tilde{d})(D_f \,|\, P) \gtrsim (\nu \tilde{d})(D_f \,|\, [\![g]\!]_s)$ and $g \stackrel{!c}{\hookrightarrow}$ implies, by Proposition H.3 (3), $(\nu \tilde{d})(D_f \,|\, [\![g]\!]_s) \stackrel{\overline{s}\langle c\rangle}{\longrightarrow}$; thus by definition of $\gtrsim$, $(\nu \tilde{d})(D_f \,|\, P) \stackrel{\overline{s}\langle c\rangle}{\Longrightarrow}$ and $(\nu \tilde{d})(D_f \,|\, [\![f]\!]_s \,|\, !s(x).\mathbf{0}) \stackrel{\tau\langle s,c\rangle}{\Longrightarrow}$;

   ($\Leftarrow$)**:** in this case we can proceed similarly, the result follows by applying Proposition H.3 (2,4).

□