The Journal of Logic and Algebraic Programming 79 (2010) 550–577

Contents lists available at ScienceDirect

The Journal of Logic and Algebraic Programming

journal homepage: www.elsevier.com/locate/jlap

Termination in higher-order concurrent calculi ☆,☆☆

Romain Demangeon [a,*], Daniel Hirschkoff [a], Davide Sangiorgi [b]

[a] *Équipe PLUME, Laboratoire de l'Informatique du Parallélisme (LIP), ENS Lyon, 46 allée d'Italie, F-69364 Lyon Cedex, France*
[b] *Dipartimento di Scienze dell'Informazione, Università di Bologna, Mura Anteo Zamboni, 7, I-40126 Bologna, Italy*

ARTICLE INFO

Article history:
Available online 15 July 2010

Keywords:
Concurrency theory
Process algebra
Ttermination
Type systems
Higher-order calculi
π-calculus

ABSTRACT

We study termination of programs in concurrent higher-order languages. A higher-order concurrent calculus combines features of the λ-calculus and of the message-passing concurrent calculi. However, in contrast with the λ-calculus, a simply-typed discipline need not guarantee termination and, in contrast with message-passing calculi such as the π-calculus, divergence can be obtained even without a recursion (or replication) construct.

We first consider a higher-order calculus where only processes can be communicated. We propose a type system for termination that borrows ideas from termination in Rewriting Systems (and following the approach to termination in the π-calculus in [3]). We then show how this type system can be adapted to accommodate higher-order functions in messages. Finally, we address termination in a richer calculus that includes localities and a passivation construct, as well as name-passing communication. We illustrate the expressiveness of the type systems on a few examples.

1. Introduction

A system is terminating when it cannot perform an infinite number of reduction steps. Termination is a difficult property to ensure: typically, the termination of a rewriting system is not decidable in the general case. The problem of termination has been widely studied in sequential languages, including higher-order ones such as the λ-calculus, exploiting static analyses, and especially type systems.

Ensuring termination for concurrent and mobile systems is even more challenging, as such systems are rarely confluent. The presence of mobility, under the form of an evolving topology of communication (new communication endpoints can be created, information travels across the system along dynamically evolving connections), adds even more complexity to the task. Previous works on this subject [3,9,14] rely on type systems to ensure termination in a concurrent context, in the setting of the π-calculus (π) [10]. In some of these systems, weights are assigned to π-calculus channels, and typability guarantees that, at each reduction step that involves the firing of a replicated term, the total weight associated to the process decreases.

In this work, we want to address the problem of termination in languages that include powerful primitives for distributed programming. The most important primitive that we focus on is *process passing*, that is, the ability to transmit an entity of computation along messages. We therefore study higher-order concurrent languages, and focus on the higher-order π-calculus, HOpi [8], as working formalism to analyse termination in this setting.

To our knowledge, there exists no result on termination for higher-order concurrent processes. In some sense, formalisms like HOpi combine features from both the λ-calculus and the π-calculus, and ensuring termination in such a setting involves the control of difficulties related both to the higher-order aspects and to the concurrency aspects of the model.

☆ This article expands on the talk given at the 20th Nordic Workshop on Programming Theory, NWPT 2008, Tallinn, 19–21 Nov. 2008.

☆☆ Work supported by the FP7 ICT integrated project 231620 HATS and the French ANR projects CHoCo and Complice.

* Corresponding author.
E-mail addresses: Romain.Demangeon@ens-lyon.fr (R. Demangeon), Daniel.Hirschkoff@ens-lyon.fr (D. Hirschkoff), Davide.Sangiorgi@cs.unibo.it (D. Sangiorgi).

1567-8326/$ - see front matter © 2010 Elsevier Inc. All rights reserved.
doi:10.1016/j.jlap.2010.07.007

In contrast with name-passing concurrent languages such as the π-calculus, where recursion (or a similar operator such as replication) is needed in order to build non-terminating programs, in HOpi, similarly to the λ-calculus, non-termination can show up already in the fragment without recursion. As an example, consider the following process:

$$Q_0 = P_0 \mid \overline{a}\langle P_0\rangle, \quad \text{where} \quad P_0 = a(X).(X \mid \overline{a}\langle X\rangle).$$

P_0 receives a process on channel a, spawns the received process and emits a copy of this process on a again. In turn, Q_0 consists of a copy of P_0 emitted on a, in parallel with an active copy of P_0. Q_0 can only reduce to itself, giving rise to a divergence.

Another difference with the situation in the λ-calculus is related to typing. In the λ-calculus, termination can be ensured by adopting a type discipline, such as that of the simply-typed λ-calculus, which rules out recursive types. On the other hand, the HOpi process Q_0 is typable without resorting to recursive types (Q_0 is a process of simply-typed HOpi, where name a is used to carry processes, and X, Y are process variables).

To sum up, calculi like HOpi combine ideas from π-calculus and λ-calculus, and in both these calculi termination has been studied (using type systems). We cannot however directly adapt existing ideas. On the one hand, the type systems for termination in the π-calculus essentially impose constraints on the recursion (or replication) operators; we cannot directly adopt the idea in HOpi because HOpi has no recursion. On the other hand, the type systems for termination in the λ-calculus put constraints on self-applications, notably by forbidding recursive types. We cannot directly adopt these either, because of non-terminating examples like the one above.

The goal of this paper is to study type disciplines for higher-order concurrent calculi that allow us to rule out non-terminating programs such as process Q_0 above, while retaining a non-trivial expressiveness.

A solution to follow this program could be to exploit the standard encoding of HOpi in π [10], that respects termination, and use it, together with existing type systems for π, to infer termination in HOpi. However this would not be applicable in extensions of HOpi that are not encodable in π (or that appear difficult to encode), for instance, in distributed versions of the calculus. If one wishes to handle models for distributed computing (including explicit locations and mobility of locations), the techniques and type systems for termination should be directly formulated on HOpi. Further, a direct formulation would allow one to make enhancements of the techniques that are tailored to (and therefore more effective on) higher-order concurrency. We nevertheless analyse the approach via the encoding in the π-calculus in Section 2.3, to compare it with our system in terms of expressiveness.

In this paper, we first (Section 2) analyse termination in HOpi_2, a higher-order calculus where processes are the only values exchanged. We propose a type system for termination using techniques from term-rewriting, in which termination is guaranteed by a decreasing weight associated to processes. This is also the approach followed in [3] for termination in the π-calculus. The technical details and the proofs are however rather different, for the reasons outlined earlier (e.g., name passing vs process passing, absence of replication or recursion). We present the basic type system and its soundness proof (Section 2.2), and provide an analysis of its expressiveness (Section 2.3).

The system for HOpi_2 is a starting point, from which we build a similar type system for HOpi_ω, a richer higher-order calculus where the values communicated also include higher-order functions (Section 3 – the names HOpi_2 and HOpi_ω are inspired from [10]). The additional constructs for functions have to be controlled in order to rule out diverging behaviours.

In Section 4, we further extend our framework to analyse termination in a richer calculus, called $\text{HOpi}^!_\omega$. The type system for $\text{HOpi}^!_\omega$ goes beyond those of Sections 2 and 3, both because the calculus includes specific constructs, and because the analysis made using types is finer. We illustrate the flexibility of our approach by studying an encoding of the choice operator (Section 4.3), which involves non-trivial backtracking mechanisms that are difficult to analyse.

In Section 5, we explore another path in the analysis of the expressiveness and the flexibility of the system of Section 2. We indeed study termination in Paπ, a calculus that extends HOpi_2 with π-calculus-like name passing, and with powerful primitives such as explicit localities and *passivation*. Passivation is the operation of capturing a running computation in a preemptive way, in order to be able to modify the process being executed (for instance to discard, duplicate or update it). We provide several examples to illustrate the expressive power given by the combination of primitives in Paπ. Analysing and controlling interaction in Paπ is a challenging task. We discuss how the ideas we developed to control process passing in HOpi_2 can be combined with the approach to name passing of [3] in order to guarantee termination.

This paper extends [7]. The presentation we give here is more complete: we include the detailed proofs of our results, which were omitted for lack of space in [7]. Moreover, the developments we present in Section 4 (calculus $\text{HOpi}^!_\omega$ and its type system, typing the encoding of choice) were only sketched in the reference mentioned above.

2. HOpi_2

This section is dedicated to the study of HOpi_2, a basic higher-order process calculus, with processes as the only communication values (the index 2 in HOpi_2 is inherited from the notation in [10, Part V]).

2.1. The calculus

We shall use symbols P, Q, R, S for processes, X, Y for process variables and names a, b, c for channels.

Table 1
Laws of structural congruence.

$$P_1 \mid (P_2 \mid P_3) \equiv (P_1 \mid P_2) \mid P_3 \qquad P_1 \mid P_2 \equiv P_2 \mid P_1 \qquad P \mid \mathbf{0} \equiv P \qquad (\nu c)(\nu d)\, P \equiv (\nu d)(\nu c)\, P$$
$$(\nu c)\, \mathbf{0} \equiv \mathbf{0} \qquad (\nu c)\, (P_1 \mid P_2) \equiv P_1 \mid (\nu c)\, P_2 \text{ if } c \text{ is not free in } P_1$$

Table 2
Reduction relation in HOpi_2.

$$(\textbf{Com})\ \frac{}{\overline{a}\langle Q\rangle.P_1 \mid a(X).P_2 \to P_1 \mid P_2[Q/X]} \qquad (\textbf{Spect})\ \frac{P_1 \to P_1'}{P_1 \mid P_2 \to P_1' \mid P_2} \qquad (\textbf{Scop})\ \frac{P \to P'}{(\nu a)P \to (\nu a)P'}$$
$$(\textbf{Cong})\ \frac{Q \equiv P \quad P \to P' \quad P' \equiv Q'}{Q \to Q'}$$

The grammar for processes of HOpi_2 is the following:

$$P \quad ::= \quad \mathbf{0} \mid P|P \mid \overline{a}\langle P\rangle.P \mid a(X).P \mid X \mid (\nu a)\, P.$$

X (resp. a) is bound in $a(X).P$ (resp. $(\nu a)\, P$). Structural congruence ($\equiv$) is defined as the smallest equivalence relation that satisfies the laws of Table 1 and that is closed under contexts. We shall omit trailing occurrences of $\mathbf{0}$ in processes of the form $\overline{a}\langle P\rangle.\mathbf{0}$. Reduction is defined by the rules of Table 2. $P[Q/X]$ stands for the capture avoiding substitution of variable X with process Q in P. A process P is *terminating* if there exists no infinite sequence of reductions emanating from P. We suppose that all processes we shall manipulate obey a *Barendregt convention*: all bound names are pairwise distinct and different from all free names.

Reusing notations. In the following sections of the paper, we shall examine different calculi, and introduce in each case a dedicated type system for termination. These process calculi represent enrichments of HOpi_2. We will often reuse the notations and conventions we introduce (for terms, operational semantics and type system) for HOpi_2. When doing so, the process calculus we will be dealing with should be clear from the context. Only when necessary, that is, when reasoning about processes belonging to different calculi, we shall use specialised notations – this will be the case in Section 2.3, where we manipulate processes from HOpi_2 and from the pure π-calculus, and in Section 4.2, where an auxiliary calculus is introduced to construct the soundness proof for the type system for $\mathrm{HOpi}^{!}_{\omega}$.

To see how HOpi_2 processes interact, consider

$$S_1 = \overline{a}\langle \overline{b}\langle \mathbf{0}\rangle\rangle.\overline{a}\langle b(Z).\mathbf{0}\rangle \quad \text{and} \quad S_2 = a(X).a(Y).(X \mid Y).$$

S_1 is a process which sends on a the code of a process emitting $\mathbf{0}$ on b, and then sends on a the code of a process receiving on b. S_2 is a process which upon reception of two processes on channel a (in sequence) executes these in parallel. Process $S_1 \mid S_2$ performs two reductions to become $\overline{b}\langle \mathbf{0}\rangle \mid b(Z).\mathbf{0}$, after which a synchronisation on b can take place.

As discussed above, recursive outputs ("self-emissions") can lead to diverging behaviours in HOpi_2: in process Q_0 from Section 1, a process containing an output on a is sent over channel a itself in $\overline{a}\langle P_0\rangle$, and we have $Q_0 \to Q_0$. The type system we introduce below puts constraints on self-emissions in order to control divergence.

2.2. A type system to ensure termination in HOpi_2

We now define a type system for termination in HOpi_2. This type system associates types to channels, of the form $Ch^n(\diamond)$, where $\diamond$ is interpreted as the type of processes (throughout the paper, we use the syntax $Ch(T)$ to denote the type of a channel carrying values of type T), and n is a natural number, called the *level* of the channel being typed. Processes are typed using simply a natural number. We use Γ to range over typing contexts, that are lists of typing hypotheses of the form $a : Ch^n(\diamond)$ or $X : n$ with at most one hypothesis for each a or X. In the case where Γ contains a hypothesis $a : Ch^n(\diamond)$, we shall write $\Gamma(a) = Ch^n(\diamond)$, and $lvl_\Gamma(a) = n$. Moreover, when writing a typing context of the form $\Gamma, a : T$ (resp. $\Gamma, X : n$), we will always implicitly suppose that Γ does not contain a typing hypothesis about a (resp. X).

Before introducing formally the type system, we discuss an example. To type-check process $\overline{a}\langle \overline{c}.\mathbf{0}\rangle \mid a(X).(\overline{b}\langle X\rangle \mid X)$, we must be able to assign a level n_a to a (thus assigning type $Ch^{n_a}(\diamond)$ to a), and similarly n_b, n_c for b, c. The emission of $\overline{c}.\mathbf{0}$ on a imposes $n_a > n_c$. The structure of the continuation of the input process, $\overline{b}\langle X\rangle \mid X$, imposes that an hypothesis $X : k$ is present in the typing context, with $n_b > k$. Moreover, in order to allow the transmission on b of the process transmitted on a, we must insure $n_b \geq n_a$. This finally gives the constraints $n_b \geq n_a > n_c$, which allow us to type-check the example process.

Table 3 presents the rules of our type system for HOpi_2. These define a judgement of the form $\Gamma \vdash P : n$. We use notation $\mathcal{D} : (\Gamma \vdash P : n)$ to mean that $\mathcal{D}$ is a derivation of the typing judgement $\Gamma \vdash P : n$.

We briefly comment on the definition of the type system. The actual control takes place in rule (**Out**), where we ensure that the level of the transmitted process is strictly smaller than the level of the carrying channel: this way, we exclude "self-emissions". This discipline is at the basis of the termination proof: when a communication is performed, an output of weight n is traded for possibly several new outputs appearing in the process, that all have a weight smaller than n.

Table 3
HOpi$_2$: Typing rules.

$$(\textbf{Nil})\ \frac{}{\Gamma \vdash \mathbf{0} : 0} \qquad (\textbf{Var})\ \frac{\Gamma(X) = n}{\Gamma \vdash X : n} \qquad (\textbf{Res})\ \frac{\Gamma, a : Ch^k(\diamond) \vdash P : n}{\Gamma \vdash (\nu a)P : n} \qquad (\textbf{Par})\ \frac{\Gamma \vdash P_1 : n_1 \quad \Gamma \vdash P_2 : n_2}{\Gamma \vdash P_1 \mid P_2 : \max(n_1, n_2)}$$

$$(\textbf{In})\ \frac{\Gamma, X : (k-1) \vdash P : n \quad \Gamma(a) = Ch^k(\diamond)}{\Gamma \vdash a(X).P : n}$$

$$(\textbf{Out})\ \frac{\Gamma \vdash Q : m \quad \Gamma \vdash P : n \quad \Gamma(a) = Ch^k(\diamond) \quad m < k}{\Gamma \vdash \bar{a}\langle Q\rangle.P : \max(k, n)}$$

We can check that process Q_0 from Section 1 is ruled out by our system: as P_0 contains an output on a, its level is at least the level of a. As a consequence, the rule (**Out**) forbids P_0 to be sent on a itself, and Q_0 is not typable.

2.2.1. Soundness

We now turn to the proof that all typable processes terminate. The type systems we shall present in the following sections enrich the one for HOpi$_2$. The structure of their soundness proof will be similar to the one we present now, although, in some cases, they will involve more technicalities (this will notably be the case for the system of Section 4).

Our type system, the way it is defined, does not satisfy a sub-typing property of the form "$\Gamma \vdash P : n$ and $n \leq n_1$ implies $\Gamma \vdash P : n_1$". As a result, the level of a process is not preserved by reduction, because a process variable with level k can be instantiated by a process whose level is $k' < k$. Therefore, it can be proved that the level of a process can only decrease after a reduction step.

The type system of Table 3 satisfies some standard properties, which are given by Lemmas 2.1 and 2.2.

Lemma 2.1. *If a does not occur free in P then, for any T, $\Gamma \vdash P : n$ iff $\Gamma, a : T \vdash P : n$.*
If X does not occur free in P then, for any k, $\Gamma \vdash P : n$ iff $\Gamma, X : k \vdash P : n$.

Proof. These results are proved by induction on the derivation of $\Gamma \vdash P : n$. □

Lemma 2.2. *If $P \equiv Q$ then $\Gamma \vdash P : n$ iff $\Gamma \vdash Q : n$.*

Proof. This result is established by induction on the derivation of $P \equiv Q$, using the fact that the max operator satisfies laws of associativity and commutativity. □

To establish soundness of our type system, we introduce a measure on processes, which is defined only in case a process is typable – the measure is actually defined on typing derivations, rather than on 'bare' processes. Recall that notation $\mathcal{D} : (\Gamma \vdash P : n)$ means that $\mathcal{D}$ is a derivation of the typing judgment $\Gamma \vdash P : n$. Given such a derivation, we introduce $m_{\mathcal{D}}(P)$, the measure associated to the typing derivation $\mathcal{D}$ for P, which is given as a multiset of natural numbers (P is redundant in the notation $m_{\mathcal{D}}(P)$, since it can be deduced from $\mathcal{D}$ – we keep it for readability purposes).

We are therefore led to introduce and manipulate several notations related to multisets. These notations will be also useful in the sequel of the paper.

Notations and results on multisets. We use M, M', N to range over multisets of natural numbers, and $\uplus$ to denote multiset sum, $\cap$ to denote multiset intersection and $-$ to denote multiset difference. For instance, $\{1, 1, 2\} \uplus \{2, 3\} = \{1, 1, 2, 2, 3\}$, $\{2, 2, 2, 3\} \cap \{2, 2, 1\} = \{2, 2\}$ and $\{1, 1, 2, 2, 3\} - \{2, 1\} = \{1, 2, 3\}$. $\biguplus_{1 \leq i \leq k} M_i$ will stand for $M_1 \uplus \cdots \uplus M_k$.

$<_{mul}$ denotes the (strict) multiset extension of the standard ordering on natural numbers (written $<$) defined by: $M_1 <_{mul} M_2$ if $M_1 \neq M_2$ and $N_1 = M_1 - (M_1 \cap M_2)$, $N_2 = M_2 - (M_1 \cap M_2)$ and $\forall e_1 \in N_1, \exists e_2 \in N_2, e_1 < e_2$. Notice that, as $<$ is total, if N_1 and N_2 are non-empty, the latter condition amounts to $\max(N_1) < \max(N_2)$. We have, for instance, $\{2, 2\} <_{mul} \{3\}$ and $\{4, 2, 1\} <_{mul} \{4, 3\}$.

$\leq_{mul}$ is defined as $(<_{mul} \cup =)$, and $>_{mul}$ is the converse of $<_{mul}$. By a standard result of rewriting theory [12], $>_{mul}$ is a terminating relation, an important property that will be used in the proofs below.

We furthermore let $\max_{mul}(M, M')$ stand for the maximum of multisets M and M' according to $<_{mul}$ (which is a total relation, since $<$ is). We use the notation $c.M$ to denote the multiset sum of c copies of M. We define $succ(M)$ as $M \uplus \{0\}$. We can remark that $succ(M)$ is the smallest multiset M' s.t. $M <_{mul} M'$.

Finally, we will need two standard results on multisets of natural numbers: if n is a natural number, M, N are two multisets of natural numbers, and if $\{n\} >_{mul} M$ and $\{n\} >_{mul} N$, then $\{n\} >_{mul} M \uplus N$. As a consequence, if n, c are two natural numbers and M is a multiset of natural numbers such that $\{n\} >_{mul} M$, then $\{n\} >_{mul} c.M$.

In the following definition, as well as in similar definitions in the remainder of the paper, when we describe a typing derivation, we sometimes omit some side conditions (such as, e.g., $\Gamma(a) = Ch^k(\diamond)$) – we shall only state them explicitly when necessary.

Definition 2.3. If $\mathcal{D} : (\Gamma \vdash P : n)$, we define $m_{\mathcal{D}}(P)$ by induction over the structure of $\mathcal{D}$ as follows:

- $m_{\mathcal{D}}(\mathbf{0}) = m_{\mathcal{D}}(X) = \emptyset$;
- $m_{\mathcal{D}}(P_1 \mid P_2) = m_{\mathcal{D}^1}(P_1) \uplus m_{\mathcal{D}^2}(P_2)$ where $\mathcal{D}$ is obtained from premises $\mathcal{D}^1 : (\Gamma \vdash P_1 : n_1)$ and $\mathcal{D}^2 : (\Gamma \vdash P_2 : n_2)$, for some n_1, n_2;
- $m_{\mathcal{D}}((\nu a)\, P_1) = m_{\mathcal{D}^1}(P_1)$ where $\mathcal{D}$ is obtained from premise $\mathcal{D}^1 : (\Gamma, a : Ch^k(\diamond) \vdash P_1 : n)$, for some k;
- $m_{\mathcal{D}}(a(X).P_1) = m_{\mathcal{D}^1}(P_1)$ where $\mathcal{D}$ is obtained from premise $\mathcal{D}^1 : (\Gamma, X : k - 1 \vdash P_1 : n)$, for some k;
- $m_{\mathcal{D}}(\overline{a}\langle Q\rangle.P_1) = m_{\mathcal{D}^1}(P_1) \uplus \{lvl_\Gamma(a)\}$ where $\mathcal{D}$ is obtained from premise $\mathcal{D}^Q : (\Gamma \vdash Q : n_1), \mathcal{D}^1 : (\Gamma \vdash P_1 : n_Q)$ for some n_1, n_Q.

The type system of Table 3, as well as the other type systems in the paper, satisfies an important property: there is one typing rule per construct of the calculus. However, we are compelled to define the measure *on typing derivations*, and not on processes. Indeed, according to Definition 2.3, $\overline{a}\langle Q\rangle$ contributes to $m_{\mathcal{D}}(P)$, even if a is bound by restriction in P: in this case, the contribution of $\overline{a}\langle Q\rangle$ is determined by the typing hypothesis about a. This is the main reason why $m_{\mathcal{D}}(P)$ is defined by analysing a typing derivation, and not simply the process syntax and the typing context.

Lemma 2.4. *Let M be a multiset of integers. If $P \equiv Q$ then there exists $\mathcal{D}$ s.t. $\mathcal{D} : (\Gamma \vdash P : n)$ and $m_{\mathcal{D}}(P) = M$ iff there exists $\mathcal{D}'$ s.t. $\mathcal{D}' : (\Gamma \vdash Q : n)$ and $m_{\mathcal{D}'}(Q) = M$.*

Proof. We prove this result by induction on the derivation of $P \equiv Q$, using the definition of $m_{\mathcal{D}}$. □

The following lemma shows that typability is preserved when substituting a process variable with a typable process, provided some conditions are met. It also explains how the measure evolves when doing so.

Lemma 2.5. *If $\mathcal{D} : (\Gamma, X : m \vdash P : n)$ and $\mathcal{D}^Q : (\Gamma \vdash Q : m')$ with $m' \leq m$, then there exist $\mathcal{D}'$, n', c s.t. $\mathcal{D}' : (\Gamma \vdash P[Q/X] : n')$, $n' \leq n$ and $m_{\mathcal{D}}(P[Q/X]) = m_{\mathcal{D}}(P) \uplus c.m_{\mathcal{D}^Q}(Q)$.*

Proof. We reason by induction on the typing derivation:

- Case (**Nil**) is immediate.
- Case (**Par**). We have $P = P_1 \mid P_2$. We use the induction hypothesis, the rule (**Par**), the fact that $(P_1 \mid P_2)[Q/X] = (P_1[Q/X]) \mid (P_2[Q/X])$ and Definition 2.3.
- Case (**Res**). We have $P = (\nu a)\, P_1$. We use the induction hypothesis, rule (**Res**) and Definition 2.3.
- Case (**Var**). The case where $P = Y$ and $Y \neq X$ is immediate. Suppose $P = X$. As $X[Q/X] = Q$, $m' \leq m$ and $m_{\mathcal{D}}(X) = \emptyset$, we set $\mathcal{D}' = \mathcal{D}^Q$ and $c = 1$.
- Case (**In**). We have $P = a(Y).P_1$. As our processes abide Barendregt Convention, if X occurs free in P, then $X \neq Y$, and we can suppose that Y is not in the domain of Γ. Thus $(a(Y).P_1)[Q/X] = a(Y).(P_1[Q/X])$, and we can rely on the induction hypothesis on P_1 to conclude using. rule (**In**) and Definition 2.3.
- Case (**Out**). We have $P = \overline{a}\langle S\rangle.P_1$. There exists l s.t. $lvl(a) = l$ and $\mathcal{D}$ is obtained using rule (**Out**) from premises $\mathcal{D}^S : (\Gamma, X : m \vdash S : n_S)$, $\mathcal{D}^1 : (\Gamma, X : m \vdash P_1 : n_1)$, with $l > n_S$. Note that $n = \max(l, n_1)$. By induction we have $\mathcal{D}^{(1)}, \mathcal{D}^{(S)}$ s.t. $\mathcal{D}^{(S)} : (\Gamma \vdash S[Q/X] : n'_S \leq n_S), \mathcal{D}^{(1)} : (\Gamma \vdash P_1[Q/X] : n'_1 \leq n_1), m_{\mathcal{D}^{(1)}}(P_1[Q/X]) = m_{\mathcal{D}^1}(P_1) \uplus c_1.m_{\mathcal{D}^Q}(Q)$. As $l > n_S \geq n'_S$, we can construct

$$\mathcal{D}' = (\mathbf{Out})\ \frac{\mathcal{D}^{(1)} \qquad \mathcal{D}^{(S)}}{\Gamma \vdash \overline{a}\langle S[Q/X]\rangle.P_1[Q/X] : \max(l, n'_1)}$$

and we have $\max(l, n'_1) \leq \max(l, n_1)$. Definition 2.3 gives $m_{\mathcal{D}}(P) = \{l\} \uplus m_{\mathcal{D}^1}(P_1)$ and $m_{\mathcal{D}'}(P[Q/X]) = \{l\} \uplus m_{\mathcal{D}^{(1)}}(P_1[Q/X]) = \{l\} \uplus m_{\mathcal{D}^1}(P_1) \uplus c_1.m_{\mathcal{D}^Q}(Q)$. This allows us to conclude by setting $c = c_1$. □

We shall need the following auxiliary lemma:

Lemma 2.6. *If $\mathcal{D} : (\Gamma \vdash P : n)$, then $m_{\mathcal{D}}(P) <_{mul} \{n + 1\}$.*

Proof. By induction on the structure of $\mathcal{D}$:

- Case (**Nil**). Immediate, as $\{1\} >_{mul} .\emptyset$
- Case (**Res**). We have $P = (\nu a)\, P_1$. There exists T s.t. $\mathcal{D}$ is obtained using (**Res**) from premise $\mathcal{D}^1 : (\Gamma, a : T \vdash P_1 : n)$. We have $m_{\mathcal{D}}(P) = m_{\mathcal{D}^1}(P_1)$. The induction hypothesis gives $m_{\mathcal{D}'}(P_1) < \{n + 1\}$. Thus we have $m_{\mathcal{D}}(P) < \{n + 1\}$.
- Case (**Var**). We have $P = X$. By definition of the measure, $m_{\mathcal{D}}(X) = \emptyset$, hence the result.

- Case (**Par**). We have $P = P_1 \mid P_2$. $\mathcal{D}$ is obtained using rule (**Par**) from premises $\mathcal{D}^1 : (\Gamma \vdash P_1 : n_1)$, $\mathcal{D}^2 : (\Gamma \vdash P_2 : n_2)$. We have $n = \max(n_1, n_2)$. By the inductive hypotheses, $\{n_1 + 1\} >_{mul} m_{\mathcal{D}^1}(P_1)$ and $\{n_2 + 1\} >_{mul} m_{\mathcal{D}^2}(P_2)$. As $\max(n_1, n_2) + 1 \geq n_1 + 1$ and $\max(n_1, n_2) + 1 \geq n_2 + 1$, we deduce $\{\max(n_1, n_2) + 1\} >_{mul} m_{\mathcal{D}^1}(P_1) \uplus m_{\mathcal{D}^2}(P_2)$
- Case (**In**). We have $P = a(X).P_1$. $\mathcal{D}$ is obtained using (**In**) from premise $\mathcal{D}' : (\Gamma, X : k - 1 \vdash P_1 : n)$. The induction hypothesis gives $\{n + 1\} > m_{\mathcal{D}'}(P_1)$, and, by definition, $m_{\mathcal{D}}(a(X).P_1) = m_{\mathcal{D}'}(P_1)$. We thus conclude that $\{n + 1\} > m_{\mathcal{D}}(a(X).P_1)$.
- Case (**Out**). We have $P = \overline{a}\langle Q_2\rangle.P_1$. There exists k s.t. $lvl(a) = k$ and $\mathcal{D}$ is obtained using rule (**Out**) from premises $\mathcal{D}^2 : (\Gamma \vdash Q_2 : n_2)$, $\mathcal{D}^1 : (\Gamma \vdash P_1 : n_1)$. We have $m_{\mathcal{D}}(\overline{a}\langle P_1\rangle.Q) = m_{\mathcal{D}^1}(P_1) \uplus \{k\}$. By induction, we have $\{n_1 + 1\} >_{mul} m_{\mathcal{D}^1}(P_1)$. We conclude that $\{\max(k, n_1) + 1\} >_{mul} m_{\mathcal{D}}(\overline{a}\langle P_1\rangle.Q)$. □

The following proposition states the key property of our type system: when a typable process P reduces to P', not only is P' typable (hence, Proposition 2.7 contains the subject reduction property), but the measure decreases.

Proposition 2.7. *If $\mathcal{D} : (\Gamma \vdash P : n)$ and $P \to P'$ then there exist $\mathcal{D}'$ and n' such that $\mathcal{D}' : (\Gamma \vdash P : n')$ and $m_{\mathcal{D}'}(P') <_{mul} m_{\mathcal{D}}(P)$.*

Proof. By induction on the derivation of $P \to P'$.

- Case (**Com**). We have $P = \overline{a}\langle Q\rangle.P_1 \mid a(X).P_2 \to P' = P_1 \mid P_2[Q/X]$. From $\mathcal{D} : (\Gamma \vdash P : n)$ we obtain

$$\mathcal{D} = \dfrac{\dfrac{\mathcal{D}^Q \quad \mathcal{D}^1}{\Gamma \vdash \overline{a}\langle Q\rangle.P_1 : n_1} \quad \dfrac{\mathcal{D}^2}{\Gamma \vdash a(X).P_2 : n_2}}{\Gamma \vdash P : n}$$

  with $\mathcal{D}^1 : (\Gamma \vdash P_1 : n_1')$, $\mathcal{D}^Q : (\Gamma \vdash Q : m)$, $\mathcal{D}^2 : (\Gamma, X : l - 1 \vdash P_2 : n_2)$ and $lvl_\Gamma(a) = l > m$ for some m, n_1, n_1', n_2. By applying Lemma 2.5, we get $\mathcal{D}^{(2)} : (\Gamma \vdash P_2[Q/X] : n_2')$ with $n_2' \leq n_2$ and $m_{\mathcal{D}^{(2)}}(P_2[Q/X]) = m_{\mathcal{D}^2}(P_2) \uplus c.m_{\mathcal{D}^Q}(Q)$. This allows us to construct

$$\mathcal{D}' = \dfrac{\mathcal{D}^1 \quad \mathcal{D}^{(2)}}{\Gamma \vdash P' : n'}$$

  with $n' = \max(n_1', n_2')$. From Definition 2.3, we deduce that $m_{\mathcal{D}}(P) = m_{\mathcal{D}^1}(P_1) \uplus m_{\mathcal{D}^2}(P_2) \uplus \{l\}$ and $m_{\mathcal{D}'}(P') = m_{\mathcal{D}^1}(P_1) \uplus m_{\mathcal{D}^{(2)}}(P_2[Q/X]) = m_{\mathcal{D}^1}(P_1) \uplus m_{\mathcal{D}^2}(P_2) \uplus c.m_{\mathcal{D}^Q}(Q)$. From Lemma 2.6 we get $m_{\mathcal{D}^Q}(Q) <_{mul} \{m + 1\}$. This implies that $c.m_{\mathcal{D}^Q}(Q) <_{mul} \{m + 1\}$, and we finally obtain $c.m_{\mathcal{D}^Q}(Q) <_{mul} \{l\}$. Thus $m_{\mathcal{D}}(P) >_{mul} m_{\mathcal{D}'}(P')$.
- Case (**Spect**) we use the induction hypothesis and the compatibility of $\uplus$ with the multiset ordering.
- Case (**Cong**) we use the induction hypothesis, Lemma 2.2 and the fact that $m_{\mathcal{D}}$ is invariant by $\equiv$.
- Case (**Res**) follows from the induction hypothesis and Definition 2.3. □

Corollary 2.8. *If $\Gamma \vdash P : n$, then P terminates.*

Proof. Consider, towards a contradiction, an infinite sequence of reductions $(P_i)_{i \geq 0}$ emanating from $P = P_0$ (that is, $P_i \to P_{i+1}$ for $i \geq 0$). Proposition 2.7 allows us to construct an infinite sequence $(\mathcal{D}^i : (\Gamma \vdash P_i : n_i))_i$.

The infinite sequence $(m_{\mathcal{D}^i}(P_i))_i$ is strictly decreasing for $>_{mul}$, which is contradictory since $>_{mul}$ is well-founded. □

Clearly, our type system fails to capture all terminating processes: there are processes that are not typable and that do not exhibit infinite computations. An example is given by $\overline{a}\langle\overline{a}\langle\mathbf{0}\rangle\rangle \mid a(X).X$, in which the recursive output on a prevents us from type-checking the process.

2.3. An analysis of the type system for HOpi$_2$

We now compare the expressiveness of our type system with the expressiveness induced on HOpi$_2$ by the translation into π and the existing type system [3] for the π-calculus. We first recall the standard encoding from HOpi$_2$ to π, and the type system from [3] that we exploit to ensure termination of π-calculus processes. We then discuss the relationship to our type system for HOpi$_2$.

2.3.1. Translating HOpi$_2$ processes

We rely on (an adaptation of) the standard encoding of HOpi$_2$ into the π-calculus [8] (see also [13]).

The target calculus uses two kinds of channels: CCS-like channels (which are used only for synchronisation), ranged over h, and first-order channels, which are used to transmit CCS-like channels, ranged over using a, b, c. The grammar of (the version we study of) the π-calculus is as follows:

$$P \quad ::= \quad \mathbf{0} \mid P|P \mid (\nu c)\,P \mid (\nu h)\,P \mid !h.P \mid a(h).P \mid \overline{a}\langle h\rangle.P \mid \overline{h}.$$

Table 4
Typing termination in the π-calculus.

$$(\mathbf{Nil}_{\mathrm{Pi}})\ \frac{}{\Gamma \vdash_{\mathbf{pi}} \mathbf{0} : 0} \qquad (\mathbf{Res0}_{\mathrm{Pi}})\ \frac{\Gamma, h : \updownarrow^k \vdash_{\mathbf{pi}} P : n}{\Gamma \vdash_{\mathbf{pi}} (\nu h)\, P : n} \qquad (\mathbf{Res1}_{\mathrm{Pi}})\ \frac{\Gamma, a : \#(\updownarrow^k) \vdash_{\mathbf{pi}} P : n}{\Gamma \vdash_{\mathbf{pi}} (\nu a)\, P : n}$$

$$(\mathbf{Par}_{\mathrm{Pi}})\ \frac{\Gamma \vdash_{\mathbf{pi}} P_1 : n_1 \quad \Gamma \vdash_{\mathbf{pi}} P_2 : n_2}{\Gamma \vdash_{\mathbf{pi}} P_1 \mid P_2 : \max(n_1, n_2)} \qquad (\mathbf{Out0}_{\mathrm{Pi}})\ \frac{\Gamma(h) = \updownarrow^k}{\Gamma \vdash_{\mathbf{pi}} \overline{h} : k}$$

$$(\mathbf{Out1}_{\mathrm{Pi}})\ \frac{\Gamma \vdash_{\mathbf{pi}} P : n \quad \Gamma(a) = \#(\updownarrow^k) \quad \Gamma(h) = \updownarrow^k}{\Gamma \vdash_{\mathbf{pi}} \overline{a}\langle h\rangle.P : n} \qquad (\mathbf{In}_{\mathrm{Pi}})\ \frac{\Gamma(a) = \#(\updownarrow^k) \quad \Gamma, h : \updownarrow^k \vdash_{\mathbf{pi}} P : n}{\Gamma \vdash_{\mathbf{pi}} a(h).P : n}$$

$$(\mathbf{Rep}_{\mathrm{Pi}})\ \frac{\Gamma \vdash_{\mathbf{pi}} P : n \quad \Gamma(h) = \updownarrow^k \quad k > n}{\Gamma \vdash_{\mathbf{pi}} !h.P : 0}$$

We do not recall the operational semantics of this calculus, which is standard [10, Part I]. We overload notations, and write $P \to P'$ for reduction in the π-calculus. Note that the version of the π-calculus we work with is rather limited in terms of expressiveness, since we restrict name passing by allowing only depth 0 or 1 in the order of channels.

We write $[\![P]\!]$ for the π-calculus encoding of a HOpi_2 process P. The definition of $[\![P]\!]$ is rather standard. We recall it here (an unambiguous correspondence between HOpi_2 process variables – X – and their counterpart as CCS-like channels – h_X – is implicitly assumed):

$$[\![\mathbf{0}]\!] = \mathbf{0} \qquad [\![P \mid Q]\!] = [\![P]\!] \mid [\![Q]\!] \qquad [\![(\nu c)P]\!] = (\nu c)[\![P]\!] \qquad [\![a(X).P]\!] = a(h_X).[\![P]\!] \qquad [\![X]\!] = \overline{h_X}$$
$$[\![\overline{a}\langle P\rangle.Q]\!] = (\nu h_a)\, \overline{a}\langle h_a\rangle.(\, [\![Q]\!] \mid !h_a.[\![P]\!]\,) \quad h_a \text{ fresh.}$$

A higher-order output action $\overline{a}\langle P\rangle.Q$ is translated into the emission of a new name (h_a), which intuitively represents the address where process P can be accessed. This encoding respects termination, as expressed by the following result.

Proposition 2.9. *For any HOpi_2 process P, P terminates iff $[\![P]\!]$ terminates.*

Proof. Follows from Theorem 13.1.18 in [10]. □

In particular, the non-terminating process Q_0 of Section 1 is translated into

$$[\![Q_0]\!] \;=\; (\nu h_a)\, \overline{a}\langle h_a\rangle.!h_a.P' \;\mid\; P', \qquad \text{where} \quad P' \;=\; a(h_X).(\nu h'_a)\, \overline{a}\langle h'_a\rangle.(!h'_a.\overline{h_X} \mid \overline{h_X}).$$

2.3.2. Typing π-calculus processes

We rely on the first type system of [3] to type the encoding of a HOpi_2 process. This type system assigns levels to π-calculus names, in order to control replicated processes. We give here a new presentation of this system, which is equivalent (in terms of expressiveness, and from the point of view of type inference as well – see [2]) to the original system from [3], while being more tractable to study the encoding.

The types assigned to names are of two kinds, according to the distinction between CCS-like and first-order channels:

$$T \quad ::= \quad \updownarrow^n \quad \mid \#(\updownarrow^n).$$

The typing judgment for π-calculus processes is noted $\Gamma \vdash_{\mathbf{pi}} P : n$, and is defined by the rules of Table 4. The most important typing rule is ($\mathbf{Rep}_{\mathrm{Pi}}$), the one for replicated inputs: it basically imposes, for $!h.P$ to be well-typed, that the level assigned to h should dominate the level of all first-order names that are used in output subject position in P, where outputs occurring under a replication in intuitively, the counterpart of rule (**Out**) in rule (**Out**) in Table 3.

All processes typable using this type system are terminating [3].

2.3.3. Comparing the two analyses on HOpi_2 processes

We have two approaches to ensure termination of HOpi_2 processes: on the one hand, the type system from Section 2.2; on the other hand, the method consisting in type-checking the translation of a HOpi_2 process into π.

It is no surprise, that process $[\![Q_0]\!]$ (see above) is rejected by the system of [3]: first observe that the levels of h_a and h_X are necessarily equal, since they are both transmitted on channel a. This entails that subprocess $!h_a.P'$ is not typable, because of the output on h_X in P'.

There do moreover exist HOpi_2 processes that can be proved to terminate using the type system for HOpi_2, but whose encoding fails to be typable using the type system for π. A very simple example is given by $R_0 = a(X).\overline{a}\langle X\rangle$. We indeed have

$$[\![R_0]\!] \;=\; a(h_X).(\nu h_a)\, \overline{a}\langle h_a\rangle.!h_a.\overline{h_X},$$

a process which is not typable: indeed, h_X and h_a necessarily have the same type (both are transmitted on a), which prevents subprocess $!h_a.\overline{h_X}$ from being typable.

This example suggests a way to establish a relationship between the type systems in HOpi_2 and in π. Consider for that the system for HOpi_2 obtained by replacing rule (**In**) in Table 3 with the following one, the other rules remaining unchanged (the typing judgment for this modified type system shall be written $\Gamma \vdash_m P : n$):

$$(\mathbf{In}') \ \frac{\Gamma, X : k \vdash_m P : n \qquad lvl_\Gamma(a) = k}{\Gamma \vdash_m a(X).P : n}.$$

Clearly, the modified type system is more restrictive, that is, $\Gamma \vdash_m P : n$ implies $\Gamma \vdash P : n$, but not the converse (cf. process R_0 seen above).

Using this system, we can establish the following property, that allows us to draw a comparison between typability in HOpi_2 and in the π-calculus.

Proposition 2.10. *Let P be a HOpi_2 process. If $\Gamma \vdash_m P : n$, then there exists Δ, a typing context for the π-calculus, such that $\Delta \vdash_{\mathbf{pi}} [\![P]\!] : n$.*

Proof. The encoding presented above induces a translation of HOpi_2 typing contexts, defined as follows (we write $[\![\Gamma]\!]$ for the encoding of Γ):

$$[\![\emptyset]\!] = \emptyset \qquad [\![\Gamma, X : n]\!] = [\![\Gamma]\!], h_X : \updownarrow^n \qquad [\![\Gamma, a : Ch^n(\diamond)]\!] = [\![\Gamma]\!], a : \#(\updownarrow^n)$$

We reason by induction on the derivation of $\Gamma \vdash_m P : n$ to prove that $\Gamma \vdash_m P : n$ implies $[\![\Gamma]\!] \vdash_{\mathbf{pi}} [\![P]\!] : n$.

- The cases corresponding to rules (**Res**) and (**Par**) are treated easily by relying on the induction hypothesis. Case (**Nil**) is trivial.
- Case (**Var**). We can apply rule ($\mathbf{Out0}_{\text{Pi}}$) to derive $[\![\Gamma]\!] \vdash_{\mathbf{pi}} [\![X]\!] : n$, since $[\![X]\!] = \overline{h_X}$.
- Case (**In**′). We have $[\![a(X).P]\!] = a(h_X).[\![P]\!]$.
  We know by induction that $[\![\Gamma, X : k]\!] \vdash_{\mathbf{pi}} [\![P]\!] : n$, that is, $[\![\Gamma]\!], h_X : \updownarrow^k \vdash_{\mathbf{pi}} [\![P]\!] : n$. We moreover know $\Gamma(a) = Ch^k(\diamond)$, which gives $[\![\Gamma]\!](a) = \#(\updownarrow^k)$. This allows us to use rule ($\mathbf{In}_{\text{Pi}}$) to derive $[\![\Gamma]\!] \vdash_{\mathbf{pi}} [\![a(X).P]\!] : n$.
- Case (**Out**). Recall that $[\![\overline{a}\langle P\rangle.Q]\!] = (\nu h_a)\ \overline{a}\langle h_a\rangle.(\ [\![Q]\!] \mid !h_a.[\![P]\!]\ )$, for some fresh h_a.
  We know by induction that $[\![\Gamma]\!] \vdash_{\mathbf{pi}} [\![P]\!] : k$ and $[\![\Gamma]\!] \vdash_{\mathbf{pi}} [\![Q]\!] : m$. By hypothesis, we also have $\Gamma(a) = Ch^n(\diamond)$, which gives $[\![\Gamma]\!](a) = \#(\updownarrow^n)$.
  We can thus derive $[\![\Gamma]\!], h_a : \updownarrow^n \vdash_{\mathbf{pi}} !h_a.[\![P]\!] : 0$, using rule ($\mathbf{Rep}_{\text{Pi}}$) together with weakening which holds for the type system for the π-calculus ($k < n$ holds by hypothesis). This gives (rule ($\mathbf{Par}_{\text{Pi}}$)) $[\![\Gamma]\!], h_a : \updownarrow^n \vdash_{\mathbf{pi}} [\![Q]\!] \mid !h_a.[\![P]\!] : m$.
  We can now apply rule ($\mathbf{Out1}_{\text{Pi}}$) together with weakening (using $m \leq \max(m, n)$) to derive the judgment $[\![\Gamma]\!], h_a : \updownarrow^n \vdash_{\mathbf{pi}} \overline{a}\langle h_a\rangle.(\ [\![Q]\!] \mid !h_a.[\![P]\!]\ ) : \max(m, n)$ – we indeed have $[\![\Gamma]\!](a) = \#(\updownarrow^n)$, as remarked above. Finally, we can use ($\mathbf{Res}_{\text{Pi}}$) to obtain the expected result. □

In case (**In**′) of the proof above, we remark that the typing hypothesis $X : k$ in the original HOpi_2 derivation allows us to construct the π-calculus typing. If we were using rule (**In**) from Table 3, we could not conclude.

Remark 2.11 (*The limits of our type system*)**.** Proposition 2.10 shows that typability of a HOpi_2 process (in the sense of the modified type system) entails typability of its encoding. By Proposition 2.9, going via the encoding in π therefore provides a procedure to ensure termination of HOpi_2 processes.

We can observe that there do exist terms that can be typed via the encoding, but that are rejected by our type system for HOpi_2 (using neither the modified type system nor the type system from Section 2.2). This observation, together with the discussion about process R_0 above, shows that the two approaches to ensure termination of HOpi_2 processes are incomparable.

Consider indeed processes

$$R_1 = \overline{a}\langle\overline{a}\langle\mathbf{0}\rangle\rangle \mid a(X).\mathbf{0} \quad \text{and} \quad R_2 = a(X).b(Y).X \mid \overline{a}\langle\overline{a}\langle\mathbf{0}\rangle\rangle \mid \overline{b}\langle\mathbf{0}\rangle.$$

None of them is typable, because they contain "self-emissions" (an output action on channel a occurring inside a process emitted on a). However, R_1 and R_2 are terminating. Their encodings in π are

$$[\![R_1]\!] = (\nu h_a)\ \overline{a}\langle h_a\rangle.!h_a.(\nu h'_a)\ \overline{a}\langle h'_a\rangle.!h'_a.\mathbf{0} \mid a(h_X).\mathbf{0} \quad \text{and}$$
$$[\![R_2]\!] = a(h_X).b(h_Y).\overline{h_X} \mid (\nu h_a)\ \overline{a}\langle h_a\rangle.!h_a.(\nu h'_a)\ \overline{a}\langle h'_a\rangle.!h'_a.\mathbf{0} \mid (\nu h_b)\ \overline{b}\langle h_b\rangle.!h_b.\mathbf{0},$$

which are both typable using the system of Table 4. A suitable assignment for R_1 is, e.g., $lvl(h_a) = lvl(h'_a) = 1$. Both replications are typed as they have no first-order outputs in their continuation. R_2 can be typed with the same level assignment, extended with $lvl(h_b) = lvl(h'_b) = 1$.

It thus appears that self-emissions can be innocuous, while they are systematically rejected by the system of Section 2.2. Self-emissions in R_1 and R_2 are reminiscent of recursive calls in continuations of replicated π processes, like, e.g., in $!a(x).b(y).\overline{a}\langle y\rangle$. It turns out that constructions like the one we find in R_2 show up in examples, and, in particular, will be used in the encoding of choice (Section 4.3).

As pointed out in Section 1, a direct type system can be the basis for refinements and extensions. Indeed, as we expose in the next sections of this paper, some refinements and extensions of the system of Section 2.2 allow us to handle processes that go well beyond those that can be treated via encodings into the π-calculus.

3. HOPi$_\omega$: transmitting higher-order functions

3.1. The calculus

We now present HOpi$_\omega$, a calculus inspired from HOPi$^{\mathtt{unit},\to,\diamond}$ in [10]. The main difference between HOpi$_\omega$ and HOpi$_2$ is that the values communicated in HOpi$_\omega$ can be $\star$, the unique element of type `unit`, or functions (precisely parametrised processes) of arbitrarily high order (the order indicating the level of arrow nesting in the type). The grammar of HOpi$_\omega$ defines both processes and values, and is given below. We distinguish between channels (a, b, c) and variables (x, y), and use v, w to range over values.

$$P \quad ::= \quad \mathbf{0} \mid P|P \mid \overline{a}\langle v\rangle.P \mid v\lfloor v\rfloor \mid a(x).P \mid (\nu a)P \qquad\qquad v \quad ::= \quad x \mid \star \mid x \mapsto P.$$

Here, $x \mapsto P$ is a parametrised process, and $v\lfloor w\rfloor$ stands for the application of a function v to argument w. We will restrict ourselves to meaningful usages of (higher-order) functions; this can be ensured by adopting a standard type discipline, which we introduce in Section 3.2.

The operational semantics of HOpi$_\omega$ is given by the following rules (rules for closure w.r.t. parallel composition, restriction and structural congruence are the same as in Section 2.1, and are thus omitted):

$$(\mathbf{Com}_\omega)\frac{}{\overline{a}\langle v\rangle.Q_1 \mid a(x).Q_2 \to Q_1 \mid Q_2[v/x]} \qquad\qquad (\mathbf{Beta}_\omega)\frac{}{(x \mapsto P)\lfloor v\rfloor \to P[v/x]}.$$

Communication involves the transmission of a value, and β-reduction takes place when a function is applied to a value – $P[v/x]$ denotes here the process obtained by replacing variable x with value v in P without introducing variable capture.

We can remark that HOpi$_2$ processes can be seen as HOpi$_\omega$ processes by replacing communication of processes with communication of values of type $\mathtt{unit} \to \diamond$ (with the obvious meaning — types in HOpi$_\omega$ are introduced formally below), and, accordingly, usages of process variables with an application to $\star$. For instance, the diverging example Q_0 in HOpi$_2$ becomes $\overline{a}\langle x \mapsto S_0\rangle \mid S_0$ in HOpi$_\omega$, where $S_0 = a(y).(y\lfloor\star\rfloor \mid \overline{a}\langle y\rangle)$.

The following is another example HOpi$_\omega$ process:

$$S_3 \quad = \quad \overline{a}\langle x \mapsto (x\lfloor\star\rfloor \mid x\lfloor\star\rfloor)\rangle \mid \overline{b_1}\langle x_1 \mapsto \overline{c}\langle\star\rangle\rangle.\overline{b_2}\langle x_2 \mapsto c(z).\mathbf{0}\rangle \mid b_1(y_1).b_2(y_2).a(y_3).(y_3\lfloor y_1\rfloor \mid y_3\lfloor y_2\rfloor).$$

Process S_3 can do two communications on b_1 and b_2. Then, a function (in this case, a duplicator) can be transmitted on a, and successively applied to the functions sent on b_1 and b_2 (corresponding to processes respectively emitting and receiving on c). After these three reductions, we obtain the process $\overline{c}\langle\star\rangle \mid \overline{c}\langle\star\rangle \mid c(z).\mathbf{0} \mid c(z').\mathbf{0}$, which can still perform two synchronisations.

3.2. A type system for termination in HOpi$_\omega$

The grammar for types for HOpi$_\omega$ includes types for values, given by $T ::= \mathtt{unit} \mid T \to^n \diamond$, and channel types, of the form $Ch^n(T)$. In the example above (process S_3), c has thus type $Ch(\mathtt{unit})$, channels b_1, b_2 have type $Ch(\mathtt{unit} \to^n \diamond)$, and channel a has a type of the form $Ch((\mathtt{unit} \to^n \diamond) \to^k \diamond)$.

In manipulating types, we restrict ourselves to using only *well-formed* value types, defined as follows:

Definition 3.1 (*Well-formed value types*)**.** We say that T is a *well-formed value type* at level n w.r.t. a typing context Γ (written $Lvl_\Gamma(T) = n$ or simply $Lvl(T) = n$ when there is no ambiguity on Γ), whenever either $T = \mathtt{unit}$ and $n = 0$, or T' is a well-formed value type at level n', $T = T' \to^n \diamond$ and $n' < n$.

The rules defining our type system for HOpi$_\omega$ are presented in Table 5. Since there is no risk of confusion, we adopt the same notation as in Section 2 for typing judgements, and write them $\Gamma \vdash P : n$. We implicitly impose that every value type

Table 5
Typing rules for HOpi$_\omega$.

Typing values

$$\frac{}{\Gamma \vdash \bigstar : \mathtt{unit}^0} \qquad \frac{\Gamma, x : T \vdash P : n}{\Gamma \vdash x \mapsto P : T \to^{n+1} \diamond} \qquad \frac{}{\Gamma, x : T \vdash x : T}$$

Typing processes

$$\frac{}{\Gamma \vdash \mathbf{0} : 0} \qquad \frac{\Gamma, a : Ch^k(T) \vdash P : n}{\Gamma \vdash (\nu a)P : n} \qquad \frac{\Gamma \vdash P_1 : n_1 \quad \Gamma \vdash P_2 : n_2}{\Gamma \vdash P_1 \mid P_2 : \max(n_1, n_2)} \qquad \frac{\Gamma \vdash v_1 : T \to^n \diamond \quad \Gamma \vdash v_2 : T}{\Gamma \vdash v_1 \lfloor v_2 \rfloor : n}$$

$$\frac{\Gamma, x : T \vdash P : n \quad \Gamma(a) = Ch^k(T)}{\Gamma \vdash a(x).P : n}$$

$$\frac{\Gamma \vdash v : T \quad \Gamma \vdash P : n' \quad \Gamma(a) = Ch^n(T) \quad Lvl(T) = k \quad n > k}{\Gamma \vdash \overline{a}\langle v \rangle.P : \max(n, n')}$$

appearing in these rules is a well-formed value type. As in Section 2, types are annotated with a level, and the type assigned to a process is given by a natural number. The type of a process P is bound to dominate both the maximum level of outputs contained in P (not occurring inside a message), and, for any process of the form $v_1 \lfloor v_2 \rfloor$ that occurs in P not inside a message, the maximum level associated to the function v_1.

As before, we associate to a process a measure that decreases along reductions. We cannot focus our analysis, as above, only on the multiset of names used in output subject position in P (called $os(P)$ below), because β-reduction may let this multiset grow. For instance, if we take $P = (x \mapsto (\overline{a}\langle\bigstar\rangle \mid \overline{a}\langle\bigstar\rangle))\ \lfloor\bigstar\rfloor$, P has no output in subject position (the two outputs on a being guarded by the abstraction on x), so that $os(P) = \emptyset$. P can however reduce to $P' = \overline{a}\langle\bigstar\rangle \mid \overline{a}\langle\bigstar\rangle$, with $os(P') = \{a, a\}$.

Definition 3.2 (*Measure on processes in HOpi$_\omega$*)**.** Let P be a well-typed HOpi$_\omega$ process. We define $\mathcal{M}^\omega(P) = os(P) \uplus fun(P)$, where:

(*i*) $os(P)$ is the multiset of the levels of the channel names that are used in an output in P, without this output occurring in object position.
(*ii*) $fun(P)$ is defined as the multiset union of all $\{k\}$, for all $v_1 \lfloor v_2 \rfloor$ occurring in P not within a message, such that v_1 is of type $T \to^k \diamond$.

We do not enter the details of the correctness proof for the type system for HOpi$_\omega$, as it is subsumed by the proof of Theorem 4.13 (Section 4.2).

Proposition 3.3 (Soundness)**.** *If $\Gamma \vdash P : n$ for some HOpi$_\omega$ process P, then P terminates.*

Proposition 3.3 is established by observing that $\mathcal{M}^\omega(P)$ decreases at each step of transition:

- If the transition is a communication, the continuations of the processes involved in the communication contribute to the global measure the same way they did before communication, because a type preserving substitution is applied. $\mathcal{M}^\omega(P)$ decreases because an output has been consumed.
- If the transition is a β-reduction involving a function of level k, a process of level strictly smaller than k is spawned in P. Therefore, all new messages and active function applications that contribute to the measure are of a level strictly smaller than l, and $\mathcal{M}^\omega(P)$ decreases.

4. An expressive type system for parametrised process passing

We now move to the definition of a rich type system, that refines the systems of Sections 2 and 3 from several points of view. Before presenting the formal definitions (Section 4.2), we discuss the main ideas behind these refinements in Section 4.1. We shall show in Section 4.3 how this framework allows us to analyse and validate the encoding of a choice operator in an extension of HOpi$_\omega$.

4.1. Towards richer analyses

The framework we study in this section is more powerful than those of Sections 2 and 3 for two main reasons. First, the language we work with is richer than HOpi$_\omega$ (which in turn extends HOpi$_2$). Second, we make a finer analysis of termination, by defining a more complex (and more expressive) type system.

The main extension to the process calculus, beyond the addition of primitive booleans and an if-then-else construct to manipulate these, is to include a primitive construct for replication in a higher-order formalism. This in principle does not add expressiveness to the calculus, because replication is encodable in HOpi$_2$ (using a process similar to Q_0 from Section 1). However, in terms of typability, having a primitive replication, and a dedicated typing rule for it, helps in dealing with examples. The type system to handle replication in presence of higher-order communications controls divergences that can arise both from self-emissions and from recursions in replications (as they appear in the setting of [3]).

We now turn to the description of the refinements we add to the type analysis.

4.1.1. Introducing weights and capacities, using multisets

A first refinement we make to our termination analysis consists in attaching two pieces of information to a channel, instead of simply a level: a weight and a capacity (in the type systems seen above, the weight and the capacity are merged into a single information, namely the level). A channel a has a *weight*, which stands for the contribution of active outputs on a to the global weight of a process. For instance, in the process $U_1 = \overline{a_1}\langle U_2\rangle$, with $U_2 = \overline{b_1}\langle Q_1\rangle \mid \overline{b_2}\langle Q_2\rangle$, the global weight of U_2 is equal to the sum of the weights attached to names b_1 and b_2. We also associate a capacity to a channel a: this is an upper bound on the weight of processes that may be sent on a. U_1 is well-typed provided the capacity of a_1 is strictly greater than the global weight of U_2.

The distinction we make between the weight and capacity of a channel recalls the observations we have made in Remark 2.11 about the limitations of the type system of Section 2. Indeed, in the π-calculus processes $[\![R_1]\!]$ and $[\![R_2]\!]$ analysed in Remark 2.11, the level of a (resp. of h_a) somehow would play the rôle of the weight (resp. of the capacity) associated to the encoding of the HOpi$_2$ channel a.

As a second extension to our type system, we represent the weight and the capacity attached to a channel, as well as the type attached to a process, using *multisets of natural numbers*. For instance, if the outputs on a (resp. on b) weigh $\{1\}$ (resp. $\{2\}$), the process

$$S_4 \stackrel{\text{def}}{=} \overline{a}\langle P\rangle \mid \overline{a}\langle P'\rangle \mid \overline{b}\langle Q\rangle$$

has global weight $\{2, 1, 1\}$, and the message $\overline{c}\langle S_4\rangle$ is well-typed provided the capacity associated to c is strictly greater than $\{2, 1, 1\}$.

Using these ideas, certain forms of 'self-emission' can be typed. In the setting of HOpi$_2$, process $R_1 = \overline{a}\langle\overline{a}\langle\mathbf{0}\rangle\rangle$ can for instance be accepted, if we assign weight $\{1\}$ and capacity $\{2\}$ to a: the output is well-typed because process $\overline{a}\langle\mathbf{0}\rangle$ has weight $\{1\} <_{mul} \{2\}$.

4.1.2. A further refinement: handling successive input prefixes

Inspired by the third type system presented in [3], we introduce the possibility of treating sequences of input prefixes as a kind of 'single input action', that has the effect of decreasing the weight of the process being executed.

Let us sketch the main idea behind this approach, again by working in the setting of HOpi$_2$. Consider a process of the form $P = a_1(X_1)\ldots a_k(X_k).P'$. To type-check P, we make sure that the weight associated to the sequence of inputs is strictly greater than the weight associated to (some of the occurrences of) the process variables X_is in the continuation P'. The former quantity is equal to $M_1^1 \uplus \cdots \uplus M_1^k$, if the weight associated to a_i is given by multiset M_1^i. To compute the latter quantity, we must take into account the multiplicity of the instances of the X_is in the process P'; this involves some technicalities, which we expose below.

Because we work with sequences of input prefixes, we have a better expressiveness: without this possibility, we would be compelled to rely on the weight associated to a_k only to try and type-check P. For instance, in a HOpi$_2$ process like $a_1(X_1).a_2(X_2).X_1 \mid \overline{a_1}\langle\overline{a_1}\langle\mathbf{0}\rangle\rangle$, we can use the weights associated to a_1 *and* a_2 in order to type-check the apparent 'recursive call' on a_1 and accept the process as terminating. Note that this process is a simple 'variation' on $a_1(X_1).X_1 \mid \overline{a_1}\langle\overline{a_1}\langle\mathbf{0}\rangle\rangle$, which is not typable because of the self-emission on a_1: we can rely on the weight associated to a_2 to make type-checking possible.

4.2. An expressive type system for termination

We now present an enriched version of HOpi$_\omega$, that we call HOpi$^!_\omega$, for which we develop an expressive type system. The calculus HOpi$^!_\omega$ extends HOpi$_\omega$ by including primitive constructs for computation over booleans, and a replication operator. To present the grammar of HOpi$^!_\omega$, we rely on the same syntactic conventions as in the previous section, the set of values being extended with booleans `true` and `false`.

Values

$$v ::= \bigstar \mid x \mid (x \mapsto P) \mid \texttt{true} \mid \texttt{false}$$

Processes

$$P ::= \mathbf{0} \mid (\nu a)P \mid P|P \mid v\lfloor v \rfloor \mid a(x).P \mid \overline{a}\langle v\rangle.P \mid !a(x).P \mid \texttt{if}\ v\ \texttt{then}\ P\ \texttt{else}\ P.$$

Note that we restrict usages of the replication operator by applying it to inputs only.

Operational semantics. Reduction is defined by giving the following rules for the reduction of the new operators.

$$(\mathbf{CondT}_\omega)\ \frac{P \to P'}{\texttt{if true then}\ P\ \texttt{else}\ Q\ \to\ P'} \qquad (\mathbf{CondF}_\omega)\ \frac{Q \to Q'}{\texttt{if false then}\ P\ \texttt{else}\ Q\ \to\ Q'}$$

$$(\mathbf{Trig}_\omega)\ \frac{}{\overline{a}\langle v\rangle.Q_1 \mid !a(x).Q_2 \to Q_1 \mid Q_2[v/x] \mid !a(x).Q_2}.$$

Some care has to be taken when defining structural congruence. Since, as explained in Section 4.1.2, we treat sequences of inputs as a whole when type-checking processes, we are compelled to restrict the definition of structural congruence: $\equiv$ is the smallest equivalence relation that satisfies the axioms given in Section 2.1, and that is closed under contexts in which the hole does not occur under a prefix.

To see why we must proceed this way, consider the (tentative) equality

$$a(x).b(y).P \quad \equiv \quad a(x).(\mathbf{0} \mid b(y).P),$$

which is derived by rewriting $b(y).P$ into $\mathbf{0} \mid b(y).P$ under the prefix $a(x)$. It might well be the case that our type system recognises the left-hand side process as typable, by analysing the sequence of prefixes $a(x).b(y)$, and that typability fails for the right-hand side process, because in that case prefixes $a(x)$ and $b(y)$ must be treated separately. Such a situation would prevent subject congruence (the counterpart of Lemma 2.2) to hold, which would compromise subject reduction. Hence, in other words, we forbid $\equiv$ to be applied under prefixes, so that this relation 'preserves sequences of prefixes'.

Types. The types for channels in $\text{HOpi}^!_\omega$ are of the form $Ch^{M_1,M_2}(T)$, where T ranges over types for values, defined as follows:

$$T\ ::=\ \texttt{unit}^{\emptyset} \mid \texttt{bool}^{\emptyset} \mid (T \to^M \diamond).$$

In order to introduce the typing rules, we need to extend the definition of well-formed types (Definition 3.1) to handle multisets:

Definition 4.1. We say that T is a *well-formed value type* of $\text{HOpi}^!_\omega$ of weight M w.r.t. a typing context Γ (written $Lvl_\Gamma(T) = M$ or simply $Lvl(T) = M$ when there is no ambiguity on Γ), whenever M either $T = \texttt{unit}^{\emptyset}$ or $T = \texttt{bool}^{\emptyset}$ and $M = \emptyset$, or T' is a well-formed value type of weight M', $T = T' \to^M \diamond$ and $M' <_{mul} M$.

We sometimes use a shortened notation: we shall write $Lvl_\Gamma(v_j) = M$ when $\Gamma(v_j) = T_j$ and $Lvl_\Gamma(T_j) = M$.

Definition 4.2. The *M-contribution of x in P*, written $o(M, P, x)$, is defined as follows:

$$o(M, \mathbf{0}, x) = \emptyset$$

$$o(M, v_1\lfloor v \rfloor, x) = \begin{cases} M & \text{if } v_1 = x \\ \emptyset & \text{if } v_1 \neq x \end{cases}$$

$$o(M, P_1 \mid P_2, x) = o(M, P_1, x) \uplus o(M, P_2, x)$$

$$o(M, a(x').P, x) = \begin{cases} \emptyset & \text{if } x' = x \\ o(M, P, x) & \text{otherwise} \end{cases}$$

$$o(M, !a(x').P, x) = \emptyset$$

$$o(M, \overline{a}\langle Q\rangle.P, x) = o(M, (\nu a)\, P, x) = o(M, P, x)$$

$$o(\texttt{if}\ v\ \texttt{then}\ P\ \texttt{else}\ Q) = \max_{mul}(o(M, P, x), o(M, Q, x)).$$

$o(M, P, x)$ is the multiset union of c copies of the multiset M, where c is the number of occurrences of x that appear neither in messages nor under a replication in P. This is reminiscent of the integer c appearing in Lemma 2.5. We may remark that if $M \leq_{mul} N$, then $o(M, P, x) \leq_{mul} o(N, P, x)$.

Table 6
Typing rules for $\mathrm{HOpi}^{!}_{\omega}$.

Typing values

$$(\mathbf{Uni}_{\mathrm{T}})\ \frac{}{\Gamma \vdash \bigstar : \mathtt{unit}^{\emptyset}} \qquad (\mathbf{Bool}_{\mathrm{T}})\ \frac{}{\Gamma \vdash \mathtt{true}, \mathtt{false} : \mathtt{bool}^{\emptyset}} \qquad (\mathbf{Var}_{\mathrm{T}})\ \frac{}{\Gamma, x : T \vdash x : T}$$

$$(\mathbf{Fun}_{\mathrm{T}})\ \frac{\Gamma, x : T \vdash P : N}{\Gamma \vdash x \mapsto P : T \rightarrow^{\mathrm{succ(N)}} \diamond}$$

Typing processes

$$(\mathbf{Nil}_{\mathrm{T}})\ \frac{}{\Gamma \vdash \mathbf{0} : \emptyset} \qquad (\mathbf{Res}_{\mathrm{T}})\ \frac{\Gamma, a : Ch^{M_1,M_2}(T) \vdash P : N}{\Gamma \vdash (\nu a)P : N} \qquad (\mathbf{Par}_{\mathrm{T}})\ \frac{\Gamma \vdash P_1 : N_1 \quad \Gamma \vdash P_2 : N_2}{\Gamma \vdash P_1 \mid P_2 : N_1 \uplus N_2}$$

$$(\mathbf{App}_{\mathrm{T}})\ \frac{\Gamma \vdash v_1 : T \rightarrow^{M} \diamond \quad \Gamma \vdash v_2 : T}{\Gamma \vdash v_1\lfloor v_2 \rfloor : M} \qquad (\mathbf{If}_{\mathrm{T}})\ \frac{\Gamma \vdash v : \mathtt{bool}^{\emptyset} \quad \Gamma \vdash P_1 : N_1 \quad \Gamma \vdash P_2 : N_2}{\Gamma \vdash \mathtt{if}\ v\ \mathtt{then}\ P\ \mathtt{else}\ Q : \max_{mul}(N_1, N_2)}$$

$$(\mathbf{In}_{\mathrm{T}})\ \frac{\begin{array}{c}\Gamma, x_1 : T_1, \ldots, x_k : T_k \vdash P : N \quad \forall i, \Gamma(a_i) = Ch^{M_1^i, M_2^i}(T_i) \\ \forall i, Lvl(T_i) <_{mul} M_2^i \quad \biguplus M_1^i >_{mul} \biguplus o(M_2^i, x_i, P)\end{array}}{\Gamma \vdash a_1(x_1)\ldots a_k(x_k).P : N}$$

$$(\mathbf{Out}_{\mathrm{T}})\ \frac{\begin{array}{c}\Gamma(a) = Ch^{M_1,M_2}(T) \quad \Gamma \vdash P : N \\ \Gamma \vdash v : T \quad M_2 >_{mul} Lvl(T)\end{array}}{\Gamma \vdash \overline{a}\langle v \rangle.P : M_1 \uplus N}$$

$$(\mathbf{Rep}_{\mathrm{T}})\ \frac{\begin{array}{c}\Gamma, x_1 : T_1, \ldots, x_k : T_k \vdash P : N \quad \forall i, \Gamma(a_i) = Ch^{M_1^i, M_2^i}(T_i) \\ \forall i, Lvl(T_i) <_{mul} M_2^i \quad \biguplus M_1^i >_{mul} (\biguplus o(M_2^i, x_i, P)) \uplus N\end{array}}{\Gamma \vdash\, !a_1(x_1)\ldots a_k(x_k).P : \emptyset}$$

Table 6 presents the rules that define the type system for $\mathrm{HOpi}^{!}_{\omega}$– the typing judgement is written $\Gamma \vdash P : N$.

The most complex rules are $(\mathbf{In}_{\mathrm{T}})$ and $(\mathbf{Rep}_{\mathrm{T}})$, where receiving processes are typed by analysing sequences of inputs. More precisely, in the former we compare the total weight associated to the channels involved in input sequences with their capacities. And in the latter, two potential sources of divergence are controlled, we compare the total weight associated to the channels involved in input sequences with the sum of two entities: the capacities on one side, to prevent self-emission, and the weight of the continuation on the other side, to prevent loops due to recursive calls between replications.

It can be remarked that to handle polyadic communications, we associate the same capacity to all arguments in an input: for instance, to type-check a process of the form $a(x_1, x_2, x_3).P'$, rule $(\mathbf{In}_{\mathrm{T}})$ assumes the capacity associated to a is strictly greater than the level of the types to variables x_1, x_2 and x_3 in the premise. This of course is rather rough – it would be easy to define a refinement assigning a specific capacity to each component of a tuple, at the cost of more complex types.

The type system of Table 6 enjoys the following standard properties.

Lemma 4.3. *If x does not occur free in P then $\Gamma, x : T \vdash P : N$ iff $\Gamma \vdash P : N$. If a does not occur free in P then $\Gamma, a : Ch^{M_1,M_2}(T) \vdash P : N$ iff $\Gamma \vdash P : N$.*

Proof. By induction on the derivation of $\Gamma \vdash P : N$. □

Lemma 4.4. *If $P \equiv Q$ then $\Gamma \vdash P : N$ iff $\Gamma \vdash Q : N$.*

Proof. By induction on the derivation of $P \equiv Q$, using the fact that $\uplus$ is associative and commutative. □

$\mathrm{HOpi}^{!,+}_{\omega}$, an auxiliary calculus to establish soundness. In order to prove that typable $\mathrm{HOpi}^{!}_{\omega}$ processes terminate, we rely, as above, on a measure which we define on typing derivations. However, due to our treatment of sequences of input prefixes, the situation is more complex here than in the calculi of the previous sections. Indeed, a typable $\mathrm{HOpi}^{!}_{\omega}$ term does not necessarily reduce to a process whose subterms are all typable: typically, $\Gamma \vdash a(x).b(y).P : N$ does not necessarily imply $\Gamma \vdash b(y).P[v/x] : N$. Intuitively, this is the case when the input prefixes $a(x)$ and $b(y)$ have to be treated together (using only one instance of rule $(\mathbf{In}_{\mathrm{T}})$) in order to type-check $a(x).b(y).P$.

We nevertheless want to be able to reason over all possible evolvings of a typable process, and in particular to define a measure that decreases along computations. To achieve this, we introduce a variant of $\mathrm{HOpi}^{!}_{\omega}$, called $\mathrm{HOpi}^{!,+}_{\omega}$, which is a kind of *"$\mathrm{HOpi}^{!}_{\omega}$ with delayed substitutions"*. The syntax of $\mathrm{HOpi}^{!,+}_{\omega}$ is as follows:

$$P ::= \mathbf{0} \mid (\nu c)P \mid P|P \mid v\lfloor v \rfloor \mid \overline{a}\langle v\rangle.P \mid \texttt{if } v \texttt{ then } P \texttt{ else } P$$
$$\mid a_1(x_1)^{\mathbf{id}_1}\ldots a_k(x_k)^{\mathbf{id}_k}.P \mid !a_1(x_1)^{\mathbf{id}_1}\ldots a_k(x_k)^{\mathbf{id}_k}.P,$$

where $(\mathbf{id}_i)_{1\leq i\leq k}$ is a sequence of *annotations*. An annotation is either $\mathbf{1}$ or a $\text{HOpi}^{!,+}_{\omega}$ value. We furthermore introduce a *well-formedness condition* to all $\text{HOpi}^{!}_{\omega}$ processes we manipulate: we impose that in $a_1(x_1)^{\mathbf{id}_1}\ldots a_k(x_k)^{\mathbf{id}_k}.P$ and $!a_1(x_1)^{\mathbf{id}_1}\ldots a_k(x_k)^{\mathbf{id}_k}.P$, $\mathbf{id}_i = \mathbf{1}$ must imply $\mathbf{id}_{i+1} = \mathbf{1}$ for $i < k$, and that every input prefix appearing inside a process in object position or annotation position is annotated with $\mathbf{1}$.

The intuition is that, for instance, $a_1(x_1)^{v_1}.a_2(x_2)^{v_2}.a_3(x_3)^{\mathbf{1}}.P$ will evolve, after reception of value v_3 along channel a_3, into $((P[v_1/x_1])[v_2/x_2])[v_3/x_3]$: as long as the last prefix of a sequence of inputs has not been consumed, the substitutions involving the variables of the previous prefixes are not applied. One can remark that the last prefix $a_k(x_k)$ is always labelled with $\mathbf{1}$, because when the corresponding substitution $[v_k/x_k]$ is applied, the whole sequence of prefixes is consumed.

This idea is formalised by the following operation of 'triggering', that maps $\text{HOpi}^{!,+}_{\omega}$ process to their $\text{HOpi}^{!}_{\omega}$ counterpart (in the following definition, we write $Q[v/x][w/y]$ for $(Q[v/x])[w/y]$).

Definition 4.5 (*From $\text{HOpi}^{!,+}_{\omega}$ to $\text{HOpi}^{!}_{\omega}$, and back*)**.** We introduce an operator $\mathbf{trig}(\cdot)$, mapping $\text{HOpi}^{!,+}_{\omega}$ processes (resp. values) to $\text{HOpi}^{!}_{\omega}$ processes (resp. values), defined by:

$$\mathbf{trig}(\mathbf{0}) = \mathbf{0} \qquad \mathbf{trig}((\nu c)\,P) = (\nu c)\,\mathbf{trig}(P) \qquad \mathbf{trig}(\overline{a}\langle v\rangle.P) = \overline{a}\langle v\rangle.\mathbf{trig}(P)$$
$$\mathbf{trig}(v_1\lfloor v_2\rfloor) = \mathbf{trig}(v_1)\lfloor\mathbf{trig}(v_2)\rfloor \qquad \mathbf{trig}(x \mapsto P) = x \mapsto \mathbf{trig}(P) \qquad \mathbf{trig}(x) = x$$
$$\mathbf{trig}(P_1 \mid P_2) = \mathbf{trig}(P_1) \mid \mathbf{trig}(P_2) \qquad \mathbf{trig}(!a_1(x_1)^{\mathbf{1}}\ldots a_k(x_k)^{\mathbf{1}}.P) = !a_1(x_1)\ldots a_k(x_k).\mathbf{trig}(P)$$
$$\mathbf{trig}(\texttt{if } v \texttt{ then } P \texttt{ else } Q) = \texttt{if } v \texttt{ then } \mathbf{trig}(P) \texttt{ else } \mathbf{trig}(Q)$$
$$\mathbf{trig}(!a_1(x_1)^{v_1}\ldots a_{i-1}(x_{i-1})^{v_{i-1}}.a_i(x_i)^{\mathbf{1}}\ldots a_k(x_k)^{\mathbf{1}}.P) = a_i(x_i)\ldots a_k(x_k).(\mathbf{trig}(P)[v_1/x_1]\ldots[v_{i-1}/x_{i-1}]) \quad \text{with } 1 < i \leq k$$
$$\mathbf{trig}(a_1(x_1)^{v_1}\ldots a_{i-1}(x_{i-1})^{v_{i-1}}.a_i(x_i)^{\mathbf{1}}\ldots a_k(x_k)^{\mathbf{1}}.P) = a_i(x_i)\ldots a_k(x_k).(\mathbf{trig}(P)[v_1/x_1]\ldots[v_{i-1}/x_{i-1}]) \quad \text{with } 1 \leq i \leq k.$$

If P is a $\text{HOpi}^{!}_{\omega}$ process, we write $\mathbf{comp}(P)$ for the $\text{HOpi}^{!,+}_{\omega}$ process obtained from P by decorating all input prefixes with annotation $\mathbf{1}$.

Note that $\mathbf{trig}(\mathbf{comp}(P)) = P$, and $\mathbf{trig}(Q)[v/x] = \mathbf{trig}(Q[v/x])$.

To define the operational semantics of $\text{HOpi}^{!,+}_{\omega}$, we keep rules ($\mathbf{Beta}_{\mathrm{T}}$), ($\mathbf{Spect}_{\mathrm{T}}$), ($\mathbf{Scop}_{\mathrm{T}}$), ($\mathbf{Cong}_{\mathrm{T}}$), ($\mathbf{CondT}_{\mathrm{T}}$), ($\mathbf{CondF}_{\mathrm{T}}$) unchanged, and introduce the rules of Table 7 (the reduction relation on $\text{HOpi}^{!,+}_{\omega}$ is written $\rightarrow^{!}$). According to the explanations above, these rules enforce that substitutions are delayed until the last prefix in a sequence of input prefixes is consumed. More precisely, rules ($\mathbf{PrUnr}_{\mathrm{T}}$) and ($\mathbf{PrRep}_{\mathrm{T}}$) accumulate substitutions along sequences of prefixes, while rules ($\mathbf{EndUnr}_{\mathrm{T}}$) and ($\mathbf{EndRep}_{\mathrm{T}}$) are used to trigger the last prefix of a sequence of inputs.

Note that we treat differently replicated and non-replicated sequences of input prefixes, as the condition associated to typability is different in the typing rules ($\mathbf{In}\kappa_{\mathrm{Pa}}$) and ($\mathbf{Rep}\kappa_{\mathrm{Pa}}$) (given below).

We immediately have that if $P \rightarrow^{!} P'$ and P satisfies the well-formedness condition introduced above, then so does P'.

Lemma 4.6. *Let $\mathcal{R}$ be the relation defined on $\text{HOpi}^{!}_{\omega} \times \text{HOpi}^{!,+}_{\omega}$ by: $(P, Q) \in \mathcal{R}$ iff $P = \mathbf{trig}(Q)$. Then $\mathcal{R}$ is a simulation, that is, for any $(P, Q) \in \mathcal{R}$, whenever $P \rightarrow P'$, there exists Q' s.t. $Q \rightarrow^{!} Q'$ and $(P', Q') \in \mathcal{R}$.*

$\mathcal{R}$ is actually a (strong) *bisimulation* [10]. We however prove only this simulation result, as it is sufficient to deduce that if $P = \mathbf{trig}(Q)$ and P diverges, then so does Q, which is what we shall need.

Proof. We reason by induction on the derivation of $P \rightarrow P'$.

The cases corresponding to ($\mathbf{Spect}_{\mathrm{T}}$), ($\mathbf{Beta}_{\mathrm{T}}$), ($\mathbf{Scop}_{\mathrm{T}}$), ($\mathbf{CondT}_{\mathrm{T}}$) and ($\mathbf{CondF}_{\mathrm{T}}$) are easily treated using Definition 4.5. The remaining cases are more interesting:

- Case ($\mathbf{Com}_{\mathrm{T}}$). We have $P = \mathbf{trig}(Q) = \overline{a}\langle v\rangle.P_1 \mid a(x).P_2$ and $P' = P_1 \mid P_2[v/x]$. By definition of $\mathbf{trig}()$, we deduce that $Q = \overline{a}\langle v\rangle.Q_1 \mid \mathbf{Q}$ where $P_1 = \mathbf{trig}(Q_1)$. We discuss on the form of $\mathbf{Q}$:
  Case $\mathbf{Q} = !a_1(x_1)^{v_1}\ldots a_{i-1}(x_{i-1})^{v_{i-1}}.a(x)^{\mathbf{1}}.a_{i+1}(x_{i+1})^{\mathbf{1}}\ldots a_k(x_k)^{\mathbf{1}}.Q_2$, with $1 < i < k$. We have

$$P_2 = a_{i+1}(x_{i+1})\ldots a_k(x_k).(\mathbf{trig}(Q_2)[v_1/x_1]\ldots[v_{i-1}/x_{i-1}]).$$

Table 7
Communication rules for $\mathrm{HOpi}_{\omega}^{!,+}$.

$$(\mathbf{TrRep}_{\mathrm{T}})\ \frac{}{\overline{a_1}\langle v_1\rangle.Q \mid !a_1(x_1)^{\mathbf{1}}.a_2(x_2)^{\mathbf{1}}..a_k(x_k)^{\mathbf{1}}.P \rightarrow^{!} Q \mid !a_1(x_1)^{v_1}.a_2(x_2)^{\mathbf{1}}..a_k(x_k)^{\mathbf{1}}.P \mid !a_1(x_1)^{\mathbf{1}}.a_2(x_2)^{\mathbf{1}}..a_k(x_k)^{\mathbf{1}}.P}$$

$$(\mathbf{PrUnr}_{\mathrm{T}})\ \frac{1 \le i < k}{\overline{a_i}\langle v_i\rangle.Q \mid a_1(x_1)^{v_1}\ldots a_{i-1}(x_{i-1})^{v_{i-1}}.a_i(x_i)^{\mathbf{1}}.a_{i+1}(x_{i+1})^{\mathbf{1}}\ldots a_k(x_k)^{\mathbf{1}}.P \rightarrow^{!} Q \mid a_1(x_1)^{v_1}\ldots a_{i-1}(x_{i-1})^{v_{i-1}}.a_i(x_i)^{v_i}.a_{i+1}(x_{i+1})^{\mathbf{1}}\ldots a_k(x_k)^{\mathbf{1}}.P}$$

$$(\mathbf{PrRep}_{\mathrm{T}})\ \frac{1 < i < k}{\overline{a_i}\langle v_i\rangle.Q \mid !a_1(x_1)^{v_1}\ldots a_{i-1}(x_{i-1})^{v_{i-1}}.a_i(x_i)^{\mathbf{1}}.a_{i+1}(x_{i+1})^{\mathbf{1}}\ldots a_k(x_k)^{\mathbf{1}}.P \rightarrow^{!} Q \mid !a_1(x_1)^{v_1}\ldots a_{i-1}(x_{i-1})^{v_{i-1}}.a_i(x_i)^{v_i}.a_{i+1}(x_{i+1})^{\mathbf{1}}\ldots a_k(x_k)^{\mathbf{1}}.P}$$

$$(\mathbf{EndUnr}_{\mathrm{T}})\ \frac{}{\overline{a_k}\langle v_k\rangle.Q \mid a_1(x_1)^{v_1}\ldots a_{k-1}(x_{k-1})^{v_{k-1}}.a_k(x_k)^{\mathbf{1}}.P \rightarrow^{!} Q \mid P[v_1/x_1]\ldots[v_{k-1}/x_{k-1}][v_k/x_k]}$$

$$(\mathbf{EndRep}_{\mathrm{T}})\ \frac{}{\overline{a_k}\langle v_k\rangle.Q \mid !a_1(x_1)^{v_1}\ldots a_{k-1}(x_{k-1})^{v_{k-1}}.a_k(x_k)^{\mathbf{1}}.P \rightarrow^{!} Q \mid P[v_1/x_1]\ldots[v_{k-1}/x_{k-1}][v_k/x_k]}$$

Process Q can perform a reduction, using rule ($\mathbf{PrRep}_{\mathrm{T}}$), to

$$Q' = Q_1 \mid !a_1(x_1)^{v_1}\ldots a_{i-1}(x_{i-1})^{v_{i-1}}.a(x)^{v}.a_{i+1}(x_{i+1})^{\mathbf{1}}\ldots a_k(x_k)^{\mathbf{1}}.Q_2.$$

We have

$$\mathbf{trig}(Q') \;=\; \mathbf{trig}(Q_1) \mid a_{i+1}(x_{i+1})\ldots a_k(x_k).(\mathbf{trig}(Q_2)[v_1/x_1]\ldots[v_{i-1}/x_{i-1}][v/x]) \;=\; P'$$

(note that we have $(a_j(x_j).S)[v/x] = a_j(x_j).(S[v/x])$).

Case $\mathbf{Q} = a_1(x_1)^{v_1}\ldots a_{i-1}(x_{i-1})^{v_{i-1}}.a(x)^{\mathbf{1}}.a_{i+1}(x_{i+1})^{\mathbf{1}}\ldots a_k(x_k)^{\mathbf{1}}.Q_2$, with $1 \le i < k$. We reason similarly using rule ($\mathbf{PrUnr}_{\mathrm{T}}$).

Case $\mathbf{Q} = !a_1(x_1)^{v_1}\ldots a_{k-1}(x_{k-1})^{v_{k-1}}.a(x)^{\mathbf{1}}.Q_2$. We have

$$P_2 = \mathbf{trig}(Q_2)[v_1/x_1]\ldots[v_{k-1}/x_{k-1}].$$

Process Q can reduce, using rule ($\mathbf{EndRep}_{\mathrm{T}}$), to

$$Q' = Q_1 \mid Q_2[v_1/x_1]\ldots[v_{k-1}/x_{k-1}][v/x],$$

and $\mathbf{trig}(Q') = P'$.

Case $\mathbf{Q} = a_1(x_1)^{0}\ldots a_{k-1}(x_{k-1})^{0}.a(x)^{\mathbf{1}}.Q_2$. We reason similarly using rule ($\mathbf{EndRep}_{\mathrm{T}}$).

- Case ($\mathbf{Trig}_{\mathrm{T}}$). We have $P = \mathbf{trig}(Q) = \overline{a}\langle v\rangle.P_1 \mid !a(x).P_2$ and $P' = P_1 \mid P_2[v/x] \mid !a(x).P_2$. Using the definition of $\mathbf{trig}()$, we deduce that $Q = \overline{a}\langle v\rangle.Q_1 \mid !a(x)^{\mathbf{1}}.a_2(x_2)^{\mathbf{1}}\ldots a_k(x_k)^{\mathbf{1}}.Q_2$, $P_1 = \mathbf{trig}(Q_1)$ and $P_2 = a_2(x_2)\ldots a_k(x_k).\mathbf{trig}(Q_2)$. Process Q can perform a reduction, using rule ($\mathbf{TrRep}_{\mathrm{T}}$), to

$$Q' = Q_1 \mid !a(x)^{v}.a_2(x_2)^{\mathbf{1}}\ldots a_k(x_k)^{\mathbf{1}}.Q_2 \mid !a(x)^{\mathbf{1}}.a_2(x_2)^{\mathbf{1}}\ldots a_k(x_k)^{\mathbf{1}}.Q_2.$$

We then have

$$\begin{aligned}\mathbf{trig}(Q') \;=\; &\mathbf{trig}(Q_1) \mid a_2(x_2)\ldots a_k(x_k).\mathbf{trig}(Q_2)[v/x]\\ &\mid !a(x).a_2(x_2)\ldots a_k(x_k).\mathbf{trig}(Q_2) \;=\; P'. \quad \square\end{aligned}$$

After having defined reduction in $\mathrm{HOpi}_{\omega}^{!,+}$, we now turn to typing. The basic idea is to start with a typable $\mathrm{HOpi}_{\omega}^{!}$ process, and execute it 'as a $\mathrm{HOpi}_{\omega}^{!,+}$ term'. In doing so, we keep a representation of the whole sequence of prefixes before it is totally consumed, and this allows us to reconstruct the original typing derivation along reductions. The type system for $\mathrm{HOpi}_{\omega}^{!,+}$ is thus very close to the system for $\mathrm{HOpi}_{\omega}^{!}$.

Typing judgements for $\mathrm{HOpi}_{\omega}^{!,+}$ processes (written $\Gamma \vdash_{+} P : N$) are derived using the rules of Table 6, where rules ($\mathbf{In}_{\mathrm{T}}$) and ($\mathbf{Rep}_{\mathrm{T}}$) are replaced respectively with the rules presented on Table 8, in order to handle annotations.

Table 8
Dedicated typing rules for $\mathrm{HOpi}_{\omega}^{!,+}$.

$$(\mathbf{In}\kappa_{\mathrm{T}})\ \frac{\Gamma \vdash_+ v_1 : T_1 \quad \dots \quad \Gamma \vdash_+ v_i : T_i \qquad \Gamma, x_1 : T_1, \dots, x_k : T_k \vdash_+ P : N \qquad \biguplus_{1\le j\le k} M_1^j >_{mul} \biguplus_{1\le j\le k} o(M_2^j, x_j, P) \qquad \forall 1 \le j \le k, Lvl(T_j) <_{mul} M_2^j \quad \Gamma(a_j) = Ch^{M_1^j, M_2^j}(T_j)}{\Gamma \vdash_+ a_1(x_1)^{v_1} \dots a_i(x_i)^{v_i}.a_{i+1}(x_{i+1})^{\mathbf{1}} \dots a_k(x_k)^{\mathbf{1}}.P : N}$$

$$(\mathbf{Rep}\kappa_{\mathrm{T}})\ \frac{\Gamma \vdash_+ v_1 : T_1 \quad \dots \quad \Gamma \vdash_+ v_i : T_i \qquad \Gamma, x_1 : T_1, \dots, x_k : T_k \vdash_+ P : N \qquad \biguplus_{1\le j\le k} M_1^j >_{mul} \biguplus_{1\le j\le k} o(M_2^j, x_j, P) \uplus N \qquad \forall 1 \le j \le k, Lvl(T_j) <_{mul} M_2^j \quad \Gamma(a_j) = Ch^{M_1^j, M_2^j}(T_j)}{\Gamma \vdash_+ !a_1(x_1)^{v_1} \dots a_i(x_i)^{v_i}.a_{i+1}(x_{i+1})^{\mathbf{1}} \dots a_k(x_k)^{\mathbf{1}}.P : \emptyset}$$

Accordingly, the contribution $o(M, P, x)$, for P a $\mathrm{HOpi}_{\omega}^{!,+}$ term, is defined as in Definition 4.2 – in particular, $o(M, a^{\mathbf{id}}(y).P, x) = o(M, P, x)$.

The careful reader may have noticed that different typing derivations for a $\mathrm{HOpi}_{\omega}^{!}$ term P can be mapped to the same typing derivation in $\mathrm{HOpi}_{\omega}^{!,+}$, for $\mathbf{comp}(P)$. For instance, if $P = a_1(x_1).a_2(x_2).a_3(x_3).P'$, we can choose to apply rule $(\mathbf{In}_{\mathrm{T}})$ once, to the sequence of prefixes $a_1(x_1).a_2(x_2).a_3(x_3)$ (with continuation process P'), but we can also, alternatively, apply $(\mathbf{In}_{\mathrm{T}})$ first with $a_1(x_1).a_2(x_2)$, the continuation process $a_3(x_3).P'$ being typed using a second application of $(\mathbf{In}_{\mathrm{T}})$. Both these derivations are mapped to the same 'maximal' typing derivation in $\mathrm{HOpi}_{\omega}^{!,+}$, where rule $(\mathbf{In}\kappa_{\mathrm{T}})$ is used only once. This has no important consequence on our reasonings, since, intuitively, a typing derivation that relies on several applications of rule $(\mathbf{In}_{\mathrm{T}})$ for a given sequence of prefixes can always be replaced by the 'maximal' derivation, where $(\mathbf{In}_{\mathrm{T}})$ is applied only once. The following easy result formalises this idea.

Lemma 4.7. *If P is a $\mathrm{HOpi}_{\omega}^{!}$ process, then $\mathcal{D} : (\Gamma \vdash P : N)$ if and only if $\mathcal{D} : (\Gamma \vdash_+ \mathbf{comp}(P) : N)$.*

We are now ready to prove soundness of our type system for $\mathrm{HOpi}_{\omega}^{!,+}$. As above, we introduce for this two measures on $\mathrm{HOpi}_{\omega}^{!,+}$ processes. These measures are the counterpart of the measures presented in Definition 3.2.

Definition 4.8. If $\mathcal{D} : (\Gamma \vdash_+ Q : N)$, we define $\mathcal{M}^1_{\mathcal{D}}(P)$ by induction over the structure of $\mathcal{D}$ as follows:

- $\mathcal{M}^1_{\mathcal{D}}(\mathbf{0}) = \emptyset$.
- $\mathcal{M}^1_{\mathcal{D}}(Q_1 \mid Q_2) = \mathcal{M}^1_{\mathcal{D}^1}(Q_1) \uplus \mathcal{M}^1_{\mathcal{D}^2}(Q_2)$ where $\mathcal{D}$ is obtained from premises $\mathcal{D}^1 : (\Gamma \vdash_+ Q_1 : N_1)$ and $\mathcal{D}^2 : (\Gamma \vdash_+ Q_2 : N_2)$ for some N_1, N_2.
- $\mathcal{M}^1_{\mathcal{D}}((\nu a)\, Q_1) = \mathcal{M}^1_{\mathcal{D}^1}(Q_1)$ where $\mathcal{D}$ is obtained from premise $\mathcal{D}^1 : (\Gamma, a : Ch^k(T) \vdash_+ Q_1 : N)$.
- $\mathcal{M}^1_{\mathcal{D}}(\overline{a}\langle v\rangle.Q_1) = \mathcal{M}^1_{\mathcal{D}^1}(Q_1) \uplus \{M_1\}$ where $\mathcal{D}$ is obtained from premises $\mathcal{D}^v : (\Gamma \vdash_+ v : T)$, $\mathcal{D}^1 : (\Gamma \vdash Q_1 : N_1)$ (for some T, N_1) and $\Gamma(a) = Ch^{M_1, M_2}(T)$.
- $\mathcal{M}^1_{\mathcal{D}}(x\lfloor v_2 \rfloor) = \emptyset$, for any variable x.
- $\mathcal{M}^1_{\mathcal{D}}(v_1\lfloor v_2 \rfloor) = \{M\}$ when v_1 is not a variable and $\mathcal{D}$ is obtained from premises $\mathcal{D}^1 : (\Gamma \vdash_+ v_1 : T \to^M \diamond)$ and $\mathcal{D}^2 : (\Gamma \vdash_+ v_2 : T)$.
- $\mathcal{M}^1_{\mathcal{D}}(a_1(x_1)^{v_1} \dots a_{i-1}(x_{i-1})^{v_{i-1}}.a_i(x_i)^{\mathbf{1}} \dots a_k(x_k)^{\mathbf{1}}.Q_1) = \mathcal{M}^1_{\mathcal{D}^1}(Q_1) \uplus \{M_1^1\} \uplus \dots \uplus \{M_1^{i-1}\}$ where $\mathcal{D}$ is obtained from premise $\mathcal{D}^1 : (\Gamma, x_1 : T_1, \dots, x_k : T_k \vdash_+ Q_1 : N)$ and $\forall j \le k$, $\Gamma(a_j) = Ch^{M_1^j, M_2^j}(T_j)$ and $\forall j < i$, $\mathcal{D}_j : (\Gamma \vdash_+ v_j : T_j)$.
- $\mathcal{M}^1_{\mathcal{D}}(!a_1(x_1)^{v_1} \dots a_{i-1}(x_{i-1})^{v_{i-1}}.a_i(x_i)^{\mathbf{1}} \dots a_k(x_k)^{\mathbf{id}_k}.Q_1) = \{M_1^1\} \uplus \dots \uplus \{M_1^{i-1}\}$, where for all j, $\Gamma(a_j) = Ch^{M_1^j, M_2^j}(T_j)$.

In order to handle delayed substitutions, we introduce another measure, noted $\mathcal{M}^2_{\mathcal{D}}(Q)$, and defined like $\mathcal{M}^1_{\mathcal{D}}(Q)$ except for the following cases:

- $\mathcal{M}^2_{\mathcal{D}}(a_1(x_1)^{v_1} \dots a_{i-1}(x_{i-1})^{v_{i-1}}.a_i(x_i)^{\mathbf{1}} \dots a_k(x_k)^{\mathbf{1}}.Q_1) = \mathcal{M}^2_{\mathcal{D}^1}(Q_1)$ where $\mathcal{D}$ is obtained from premise $\mathcal{D}^1 : (\Gamma, x_1 : T_1, \dots, x_k : T_k \vdash_+ Q_1 : N)$.
- $\mathcal{M}^2_{\mathcal{D}}(!a_1(x_1)^{v_1} \dots a_{i-1}(x_{i-1})^{v_{i-1}}.a_i(x_i)^{\mathbf{1}} \dots a_k(x_k)^{\mathbf{id}_k}.Q_1) = \emptyset$.

Note that $\mathcal{M}^1_{\cdot}(\cdot)$ and $\mathcal{M}^2_{\cdot}(\cdot)$ coincide on processes of the form $\mathbf{comp}(P)$ (for some $\mathrm{HOpi}_{\omega}^{!}$ process P).

Lemma 4.9. *If $Q \equiv Q'$, then*

- *there exists $\mathcal{D} : (\Gamma \vdash_+ Q : N)$ s.t. $\mathcal{M}^1_{\mathcal{D}}(Q) = M$ iff there exists $\mathcal{D}' : (\Gamma' \vdash_+ Q' : N')$ s.t. $\mathcal{M}^1_{\mathcal{D}'}(Q') = M$, and*
- *there exists $\mathcal{D} : (\Gamma \vdash_+ Q : N)$ s.t. $\mathcal{M}^2_{\mathcal{D}}(Q) = M$ iff there exists $\mathcal{D}' : (\Gamma' \vdash_+ Q' : N')$ s.t. $\mathcal{M}^2_{\mathcal{D}'}(Q') = M$.*

Proof. By induction on the derivation of $Q \equiv Q'$. □

Lemma 4.10. *Suppose that $\mathcal{D}^w : (\Gamma \vdash_+ w : T)$, where $T = T' \to^M \diamond$.*

If $\mathcal{D} : (\Gamma, x : T \vdash_+ P : N)$, then there exists $\mathcal{D}'$ s.t. $\mathcal{D}' : (\Gamma \vdash_+ P[w/x] : N)$, $\mathcal{M}^1_{\mathcal{D}'}(P[w/x]) = \mathcal{M}^1_{\mathcal{D}}(P) \uplus o(M, P, x)$ and $\mathcal{M}^2_{\mathcal{D}'}(P[w/x]) = \mathcal{M}^2_{\mathcal{D}}(P) \uplus o(M, P, x)$.

If $\mathcal{D} : (\Gamma, x : T \vdash_+ v : T_0)$, then there exists $\mathcal{D}'$ s.t. $\mathcal{D}' : (\Gamma \vdash_+ v[w/x] : T_0)$.

Proof. By induction on the typing derivation.

- Cases ($\mathbf{Nil}_{\mathrm{T}}$), ($\mathbf{Res}_{\mathrm{T}}$), ($\mathbf{Par}_{\mathrm{T}}$), ($\mathbf{If}_{\mathrm{T}}$), ($\mathbf{Uni}_{\mathrm{T}}$) and ($\mathbf{Bool}_{\mathrm{T}}$) are easily done using the induction hypotheses when needed and Definition 4.8.
- Case ($\mathbf{App}_{\mathrm{T}}$). We have $P = v_1\lfloor v_2 \rfloor$. Derivation $\mathcal{D}$ is build using $\mathcal{D}^1, \mathcal{D}^2$ as premises s.t. $\mathcal{D}^1 : (\Gamma, x : T \vdash_+ v_1 : T_1 \to^N \diamond)$ and $\mathcal{D}^2 : (\Gamma, x : T \vdash_+ v_2 : T_1)$ for some T_1. We use the induction hypothesis and we get $\mathcal{D}^{(1)}, \mathcal{D}^{(2)}$ s.t. $\mathcal{D}^{(1)} : (\Gamma \vdash_+ v_1[w/x] : T_1 \to^N \diamond)$ and $\mathcal{D}^{(2)} : (\Gamma \vdash_+ v_2[w/x] : T_1)$. Using the rule ($\mathbf{App}_{\mathrm{T}}$) and the fact that $(v_1\lfloor v_2 \rfloor)[w/x] = (v_1[w/x])\lfloor (v_2[w/x]) \rfloor$, we construct

  $$\mathcal{D} = (\mathbf{App}_{\mathrm{T}}) \frac{\mathcal{D}^{(1)} \quad \mathcal{D}^{(2)}}{\Gamma \vdash_+ (v_1\lfloor v_2 \rfloor)[w/x] : N}.$$

  We distinguish three cases to compute the measures according to Definition 4.8:
  - if $v_1 = x$ then $N = M$. Definition 4.2 gives $o(M, x\lfloor v_2 \rfloor, x) = M$ and we have $\mathcal{M}^1_{\mathcal{D}}(x\lfloor v_2 \rfloor) = \mathcal{M}^2_{\mathcal{D}}(x\lfloor v_2 \rfloor) = \emptyset$ and $\mathcal{M}^1_{\mathcal{D}'}(w\lfloor v_2[w/x] \rfloor) = \mathcal{M}^2_{\mathcal{D}'}(w\lfloor v_2[w/x] \rfloor) = M$.
  - if $v_1 = y$ and $y \neq x$, then Definition 4.2 gives $o(M, x\lfloor v_2 \rfloor, x) = M$ and $\mathcal{M}^1_{\mathcal{D}'}(y\lfloor v_2[w/x] \rfloor) = \mathcal{M}^2_{\mathcal{D}'}(y\lfloor v_2[w/x] \rfloor) = \mathcal{M}^1_{\mathcal{D}}(y\lfloor v_2 \rfloor) = \mathcal{M}^2_{\mathcal{D}}(y\lfloor v_2 \rfloor)$, all these quantities being equal to $\emptyset$.
  - if v_1 is not a variable then Definition 4.2 gives $o(M, x\lfloor v_2 \rfloor, x) = \emptyset$ and Definition 2.3 gives $\mathcal{M}^1_{\mathcal{D}'}(v_1\lfloor v_2[w/x] \rfloor) = \mathcal{M}^2_{\mathcal{D}'}(v_1\lfloor v_2[w/x] \rfloor) = \mathcal{M}^1_{\mathcal{D}}(v_1\lfloor v_2 \rfloor) = \mathcal{M}^2_{\mathcal{D}}(v_1\lfloor v_2 \rfloor) = N$.
- Case ($\mathbf{In}\kappa_{\mathrm{T}}$). We have $P = a_1(x_1)^{v_1} \ldots a_i(x_i)^{v_i}.a_{i+1}(x_{i+1})^{\mathbf{1}} \ldots a_k(x_k)^{\mathbf{1}}.P_1$. Derivation $\mathcal{D}$ is obtained using rule ($\mathbf{In}\kappa_{\mathrm{T}}$) from premises $\mathcal{D}^1 : (\Gamma, x : T, x_1 : T_1, \ldots, x_k : T_k \vdash_+ P_1 : N)$ and $\mathcal{D}_j : (\Gamma, x : T \vdash_+ v_j : T_j)$, for $1 \leq j \leq i$, with $\Gamma(a_j) = Ch^{M^j_1, M^j_2}(T_j)$, $M^j_2 > Lvl_\Gamma(v_j)$ and $\uplus M^j_1 >_{mul} \uplus\, o(M^j_2, x_j, P_1)$. Since $x \neq x_j$ and $x_j \notin w$, we have $o(M^j_2, x_j, P_1) = o(M^j_2, x_j, P_1[w/x])$ (no occurrence of x_j is added or removed by replacing the occurrences of x by w in P_1).

  The induction hypothesis gives $\mathcal{D}^{(1)}, \mathcal{D}_{(1)}, \ldots, \mathcal{D}_{(i)}$ s.t. $\mathcal{D}^{(1)} : (\Gamma, x_1 : T_1, \ldots, x_k : T_k \vdash_+ P_1[w/x] : N)$, for all $1 \leq j \leq i$, $\mathcal{D}_{(j)} : (\Gamma \vdash_+ v_j[w/x] : T_j)$, $\mathcal{M}^1_{\mathcal{D}^{(1)}}(P_1) = \mathcal{M}^1_{\mathcal{D}^1}(P_1) \uplus o(M, P_1, x)$ and $\mathcal{M}^2_{\mathcal{D}^{(1)}}(P_1) = \mathcal{M}^2_{\mathcal{D}^1}(P_1) \uplus o(M, P_1, x)$. All necessary side conditions are satisfied, and we can construct the following derivation $\mathcal{D}'$, by application of rule ($\mathbf{In}\kappa_{\mathrm{T}}$):

  $$\frac{\mathcal{D}^{(1)} \quad \mathcal{D}_{(1)} \quad \ldots \quad \mathcal{D}_{(i)}}{\Gamma \vdash_+ a_1(x_1)^{v_1[w/x]} \ldots a_i(x_i)^{v_i[w/x]}.a_{i+1}(x_{i+1})^{\mathbf{1}} \ldots a_k(x_k)^{\mathbf{1}}.P_1[w/x] : N}.$$

  We can then use Definitions 4.8 and 4.2 to conclude that $\mathcal{M}^1_{\mathcal{D}'}(P') = \mathcal{M}^1_{\mathcal{D}}(P) \uplus o(M, P, x)$, and $\mathcal{M}^2_{\mathcal{D}'}(P') = \mathcal{M}^2_{\mathcal{D}}(P) \uplus o(M, P, x)$.
- Case ($\mathbf{Rep}_{\mathrm{T}}$) is treated like case ($\mathbf{In}\kappa_{\mathrm{T}}$).
- Case ($\mathbf{Out}_{\mathrm{T}}$). We have $P = \overline{a}\langle v \rangle.P_1$. There exists T' such that $\Gamma(a) = Ch^{M_1, M_2}(T')$, and $\mathcal{D}$ is obtained using rule ($\mathbf{Out}_{\mathrm{T}}$) from premises $\mathcal{D}^v : (\Gamma, x : T \vdash_+ v : T')$, $M_2 >_{mul} Lvl(T')$ and $\mathcal{D}^1 : (\Gamma, x : T \vdash_+ P_1 : N)$. The induction hypothesis gives $\mathcal{D}^{(v)}, \mathcal{D}^{(1)}$ s.t. $\mathcal{D}^{(1)} : (\Gamma \vdash_+ P_1[w/x] : N)$, $\mathcal{D}^{(v)} : (\Gamma \vdash_+ v[w/x] : T')$, $\mathcal{M}^1_{\mathcal{D}^{(1)}}(P_1) = \mathcal{M}^1_{\mathcal{D}^1}(P_1) \uplus o(M, P_1, x)$ and $\mathcal{M}^2_{\mathcal{D}^{(1)}}(P_1) = \mathcal{M}^2_{\mathcal{D}^1}(P_1) \uplus o(M, P_1, x)$.

  As $(\overline{a}\langle v \rangle.P_1)[w/x] = \overline{a}\langle v[w/x] \rangle.(P_1[w/x])$, we can construct

  $$\mathcal{D}' = (\mathbf{Out}_{\mathrm{T}}) \frac{\mathcal{D}^{(1)} \quad \mathcal{D}^{(v)}}{\Gamma(\overline{a}\langle v[w/x] \rangle.P_1)[w/x]N}.$$

  Finally, we use Definitions 4.8 and 4.2 to conclude.
- Case ($\mathbf{Fun}_{\mathrm{T}}$). We have $v = (y \mapsto P_1)$. There exists T' s.t. $T_0 = T' \to^{succ(M)} \diamond$ and $\mathcal{D}$ is obtained using rule ($\mathbf{Fun}_{\mathrm{T}}$) from a premise of the form $\mathcal{D}^1 : (\Gamma, x : T, y : T' \vdash_+ P_1 : M)$. The induction hypothesis gives $\mathcal{D}^{(1)}$ s.t. $\mathcal{D}^{(1)} : (\Gamma, y : T' \vdash_+ P_1[w/x] : M)$.

We can construct

$$\mathcal{D}' = (\mathbf{Fun}_{\mathrm{T}})\ \frac{\mathcal{D}^{(1)}}{\Gamma \vdash_+ y \mapsto P_1[w/x] : T_0}.$$

- Case ($\mathbf{Var}_{\mathrm{T}}$) with $v = y$. We have $\mathcal{D} : (\Gamma, x : T \vdash_+ y : T_0)$ with $\Gamma(y) = T_0$. We distinguish two cases:
  - $x \neq y$. Then $y[w/x] = y$, and the result follows.
  - $x = y$. Then $T = T_0$, and we can exhibit $\mathcal{D}^w : (\Gamma \vdash_+ x[w/x] : T_0)$. □

Lemma 4.11. *If $\mathcal{D} : (\Gamma \vdash_+ Q : N)$ then $\mathcal{M}^2_{\mathcal{D}}(Q) \leq_{mul} N$.*

Proof. By induction on the typing derivation:

- Cases ($\mathbf{Nil}_{\mathrm{T}}$), ($\mathbf{Res}_{\mathrm{T}}$), ($\mathbf{Par}_{\mathrm{T}}$), ($\mathbf{If}_{\mathrm{T}}$), ($\mathbf{Out}_{\mathrm{T}}$), ($\mathbf{Rep}\kappa_{\mathrm{T}}$) are treated easily, using the induction hypotheses when needed, Definition 4.8, as well as some simple properties of multisets to do the calculations.
- Case ($\mathbf{App}_{\mathrm{T}}$). We have $P = v_1\lfloor v_2 \rfloor$. There exists T_2 such that $\mathcal{D}$ is obtained by applying ($\mathbf{App}_{\mathrm{T}}$) to premises $\mathcal{D}^1 : (\Gamma \vdash_+ v_1 : T_2 \to^N \diamond)$ and $\mathcal{D}^2 : (\Gamma \vdash_+ v_2 : T_2)$. Either $v_1 = x$ for some x, and, by Definition 4.8, $\mathcal{M}^2_{\mathcal{D}}(v_1\lfloor v_2 \rfloor) = \emptyset$, or v_1 is not a variable, and, by Definition 4.8, $\mathcal{M}^2_{\mathcal{D}}(v_1\lfloor v_2 \rfloor) = N$: we can conclude in both cases.
- Case ($\mathbf{In}\kappa_{\mathrm{T}}$). We have $P = a_1(x_1)^{v_1} \ldots a_i(x_i)^{v_i}.a_{i+1}(x_{i+1})^{\mathbf{1}} \ldots a_k(x_k)^{\mathbf{1}}.P_1$. Derivation $\mathcal{D}$ is build by applying rule ($\mathbf{In}\kappa_{\mathrm{T}}$) with $\mathcal{D}^1 : (\Gamma \vdash_+ P_1 : N)$, $\mathcal{D}_1 : (\Gamma \vdash_+ v_1 : T_1), \ldots, \mathcal{D}_i : (\Gamma \vdash_+ v_i : T_i)$ as premises. The induction hypothesis gives $\mathcal{M}^2_{\mathcal{D}^1}(P_1) \leq N$. As Definition 4.8 gives $\mathcal{M}^2_{\mathcal{D}}(P) = N$, we can conclude. □

Lemma 4.7, and the fact that **comp**() is compatible with congruence, ensures that the results of Lemmas 4.3 and 4.4 still hold for $\mathrm{HOpi}^{!,+}_{\omega}$ processes. We moreover have:

Lemma 4.12. *If $\mathcal{D} : (\Gamma \vdash_+ Q : N)$ and $Q \to^! Q'$ then there exist N' and $\mathcal{D}'$, s.t. $\mathcal{D}' : (\Gamma \vdash_+ Q' : N')$ and*

- *either $\mathcal{M}^1_{\mathcal{D}}(Q) >_{mul} \mathcal{M}^1_{\mathcal{D}'}(Q')$,*
- *or $\mathcal{M}^1_{\mathcal{D}}(Q) = \mathcal{M}^1_{\mathcal{D}'}(Q')$ and $\mathcal{M}^2_{\mathcal{D}}(Q) >_{mul} \mathcal{M}^2_{\mathcal{D}}(Q)$.*

Proof. We reason by induction on the derivation of $Q \to^! Q'$.

- Cases ($\mathbf{Cong}_{\mathrm{T}}$), ($\mathbf{Spect}_{\mathrm{T}}$), ($\mathbf{Scop}_{\mathrm{T}}$), ($\mathbf{CondT}_{\mathrm{T}}$) and ($\mathbf{CondF}_{\mathrm{T}}$) can be treated using the induction hypothesis, Lemmas 4.4 and 4.9, and Definition 4.8.
- Case ($\mathbf{Beta}_{\mathrm{T}}$). We have $Q = (x \mapsto Q_1)\lfloor v_2 \rfloor$, $Q \to^! Q_1[v_2/x]$ and $\mathcal{D} : (\Gamma \vdash_+ (x \mapsto Q_1)\lfloor v_2 \rfloor : N)$. This means that $\mathcal{D}$ is of the form

$$\frac{\dfrac{\mathcal{D}^1}{\Gamma \vdash_+ (x \mapsto Q_1) : T \to^{succ(N_1)} \diamond} \quad \mathcal{D}^2}{\Gamma \vdash_+ (x \mapsto Q_1)\lfloor v_2 \rfloor : N}$$

  with $\mathcal{D}^1 : (\Gamma, x : T \vdash_+ Q_1 : N_1)$ and $\mathcal{D}^2 : (\Gamma \vdash_+ v_2 : T)$. From Definition 4.8, we deduce $\mathcal{M}^1_{\mathcal{D}}(Q) = succ(N_1)$. We apply Lemma 4.10 to $\mathcal{D}^1$, yielding $\mathcal{D}' : (\Gamma \vdash_+ Q_1[v_2/x] : N_1)$. From Lemma 4.11, we get $\mathcal{M}^2_{\mathcal{D}'}(Q_1[v_2/x]) \leq_{mul} N_1$. As Q_1 appears in a message position in Q, because of the well-formedeness condition, every input prefix in Q_1 and v_2 is annotated with $\mathbf{1}$. This allows us to deduce $\mathcal{M}^1_{\mathcal{D}'}(Q_1[v_2/x]) \leq_{mul} N_1$. Thus $\mathcal{M}^1_{\mathcal{D}}(Q) >_{mul} \mathcal{M}^1_{\mathcal{D}'}(Q')$.
- Case ($\mathbf{PrUnr}_{\mathrm{T}}$). We have

$$(\overline{a_i}\langle v_i \rangle.Q_1 \mid a_1(x_1)^{v_1} \ldots a_{i-1}(x_{i-1})^{v_{i-1}}.a_i(x_i)^{\mathbf{1}}.a_{i+1}(x_{i+1})^{\mathbf{1}} \ldots a_k(x_k)^{\mathbf{1}}.Q_2) \to^! (Q_1 \mid a_1(x_1)^{v_1} \ldots a_{i-1}(x_{i-1})^{v_{i-1}}.a_i(x_i)^{v_i}.a_{i+1}(x_{i+1})^{\mathbf{1}} \ldots a_k(x_k)^{\mathbf{1}}.Q_2).$$

  Thus $\mathcal{D}$ is of the form $\dfrac{\dfrac{\mathcal{D}^1 \quad \mathcal{D}_i}{\Gamma \vdash_+ \overline{a_i}\langle v_i \rangle.Q_1 : N_1} \quad \mathcal{D}''}{\Gamma \vdash_+ Q : N}$, $\mathcal{D}''$ being given by the following application of the ($\mathbf{In}\kappa_{\mathrm{T}}$) rule,

$$\frac{\mathcal{D}^2 \quad \mathcal{D}_1 \quad .. \quad \mathcal{D}_{i-1}}{\Gamma \vdash_+ a_1(x_1)^{v_1} \ldots a_{i-1}(x_{i-1})^{v_{i-1}}.a_i(x_i)^{\mathbf{1}}.a_{i+1}(x_{i+1})^{\mathbf{1}} \ldots a_k(x_k)^{\mathbf{1}}.Q_2 : N_2},$$

and with where $\mathcal{D}_i : (\Gamma \vdash_+ v_i : T_i)$, $\mathcal{D}^1 : (\Gamma \vdash_+ Q_1 : N_1')$, $N_1 = N_1' \uplus M_1^i$, for all $1 \leq j \leq k$, $\Gamma(a_j) = Ch^{M_1^j, M_2^j}(T_j)$, $M_2^j > Lvl_\Gamma(T_j)$, $\mathcal{D}_1 : (\Gamma \vdash_+ v_1 : T_1), \ldots, \mathcal{D}_{i-1} : (\Gamma \vdash_+ v_{i-1} : T_{i-1})$ and $\mathcal{D}^2 : (\Gamma, , x_1 : T_1, \ldots, x_k : T_k \vdash_+ Q_2 : N_2)$. We can construct, using rule ($\mathbf{Par}_\mathrm{T}$), the following derivation, called $\mathcal{D}'$:

$$\mathcal{D}' = (\mathbf{Par}_\mathrm{T}) \dfrac{\mathcal{D}^1 \quad (\mathbf{In}\kappa_\mathrm{T}) \dfrac{\mathcal{D}^2 \quad \mathcal{D}_1 \quad \ldots \quad \mathcal{D}_{i-1} \quad \mathcal{D}_i}{a_1(x_1)^{v_1} \ldots a_{i-1}(x_{i-1})^{v_{i-1}}.a_i(x_i)^{v_i}.a_{i+1}(x_{i+1})^{\mathbf{1}} \ldots a_k(x_k)^{\mathbf{1}}.Q_2}}{\Gamma \vdash_+ Q' : N'}$$

with $N' = N_1' \uplus N_2$. This is possible as the side conditions holding in $\mathcal{D}$ still hold, and $\Gamma \vdash_+ v_i : T_i$. By definition, we have $\mathcal{M}^1_{\mathcal{D}}(Q) = (\mathcal{M}^1_{\mathcal{D}^1}(Q_1) \uplus M_1^i) \uplus (\mathcal{M}^1_{\mathcal{D}^2}(Q_2) \uplus M_1^1 \uplus \cdots \uplus M_1^{i-1})$ and $\mathcal{M}^1_{\mathcal{D}}(Q') = (\mathcal{M}^1_{\mathcal{D}^1}(Q_1)) \uplus (\mathcal{M}^1_{\mathcal{D}^2}(Q_2) \uplus M_1^1 \uplus \cdots \uplus M_1^{i-1} \uplus M_1^i)$ which is $\mathcal{M}^1_{\mathcal{D}}(Q) = \mathcal{M}^1_{\mathcal{D}}(Q')$. Using the construction rules for the second measure, we deduce: $\mathcal{M}^2_{\mathcal{D}}(Q) = (\mathcal{M}^2_{\mathcal{D}^1}(Q_1) \uplus M_1^i) \uplus (\mathcal{M}^2_{\mathcal{D}^2}(Q_2))$ and $\mathcal{M}^2_{\mathcal{D}}(Q') = (\mathcal{M}^2_{\mathcal{D}^1}(Q_1)) \uplus (\mathcal{M}^1_{\mathcal{D}^2}(Q_2))$ which implies $\mathcal{M}^2_{\mathcal{D}}(Q) >_{mul} \mathcal{M}^2_{\mathcal{D}}(Q')$

- A similar reasoning is used to treat cases ($\mathbf{PrRep}_\mathrm{T}$) and ($\mathbf{Trig}_\mathrm{T}$).
- Case ($\mathbf{EndUnr}_\mathrm{T}$). We have

$$(\overline{a_k}\langle v_k\rangle.Q_1 \mid a_1(x_1)^{v_1} \ldots a_{k-1}(x_{k-1})^{v_{k-1}}.a_k^{\mathbf{1}}(x_k).Q_2) \;\to^!\; (Q_1 \mid Q_2[v_1/x_1] \ldots [v_{k-1}/x_{k-1}][v_k/x_k]).$$

Thus $\mathcal{D}$ is of the form

$$\dfrac{\dfrac{\mathcal{D}_k \quad \mathcal{D}^1}{\Gamma \vdash_+ \overline{a_k}\langle v_k\rangle.Q_1 : N_1} \quad \dfrac{\mathcal{D}^2 \quad \mathcal{D}_1 \ldots \mathcal{D}_{k-1}}{\Gamma \vdash_+ a_1(x_1)^{v_1} \ldots a_k(x_k)^{\mathbf{1}}.Q_2 : N_2}}{\Gamma \vdash_+ Q : N},$$

where $\mathcal{D}^1 : (\Gamma \vdash_+ Q_1 : N_1')$, $\mathcal{D}_1 : (\Gamma \vdash_+ v_1 : T_1), \ldots, \mathcal{D}_k : (\Gamma \vdash_+ v_k : T_k)$, $N_1 = N_1' \uplus M_1^i$, $\forall 1 \leq j \leq k$, $\Gamma(a_j) = Ch^{M_1^j, M_2^j}(T_j)$, $M_2^j > Lvl_\Gamma(T_j)$ (notice that $M_2^k > Lvl_\Gamma(T_k)$ is given by $\Gamma \vdash_+ \overline{a_k}\langle v_k\rangle.Q_1 : N_1$) and $\mathcal{D}^2 : (\Gamma, x_1 : T_1, \ldots, x_k : T_k \vdash_+ Q_2 : N_2)$.

We use k times Lemma 4.10 to obtain $\mathcal{D}^{(2)} : (\Gamma \vdash_+ Q_2[v_1/x_1] \ldots [v_k/x_k] : N_2)$ and

$$\mathcal{M}^1_{\mathcal{D}^{(2)}}(Q_2[v_1/x_1] \ldots [v_k/x_k]) \;=\; \mathcal{M}^1_{\mathcal{D}^2}(Q_2) \uplus o(Lvl_\Gamma(v_1), Q_2, x_1) \uplus \cdots \uplus o(Lvl_\Gamma(v_k), Q_2, x_k).$$

We then construct

$$\mathcal{D}' = (\mathbf{Par}_\mathrm{T}) \dfrac{\mathcal{D}^1 \quad \mathcal{D}^{(2)}}{\Gamma \vdash_+ Q_1 \mid Q_2[v_1/x_1] \ldots [v_k/x_k] : N'}$$

with $N' = N_1' \uplus N_2$. Using Definition 4.8, we have $\mathcal{M}^1_{\mathcal{D}}(Q) = \mathcal{M}^1_{\mathcal{D}^1}(Q_1) \uplus \mathcal{M}^1_{\mathcal{D}^2}(Q_2) \uplus M_1^1 \uplus \cdots \uplus M_1^k$ and $\mathcal{M}^1_{\mathcal{D}'}(Q') = \mathcal{M}^1_{\mathcal{D}^1}(Q_1) \uplus \mathcal{M}^1_{\mathcal{D}^2}(Q_2) \uplus o(Lvl_\Gamma(v_1), Q_2, x_1) \uplus \cdots \uplus o(Lvl_\Gamma(v_k), Q_2, x_k)$.

The side conditions in the usage of rule ($\mathbf{In}\kappa_{\mathrm{Pa}}$) in $\mathcal{D}$ give for all $j < k$, $M_2^j >_{mul} Lvl_\Gamma(T_j) = Lvl_\Gamma(v_j)$. Similarly, the usage of rule ($\mathbf{Out}\kappa_{\mathrm{Pa}}$) in $\mathcal{D}$ gives $M_2^k >_{mul} Lvl_\Gamma(T_k) = Lvl_\Gamma(v_k)$ and the rule ($\mathbf{In}\kappa_{\mathrm{Pa}}$) in $\mathcal{D}$ gives $\uplus_{1\leq j\leq k} M_1^j >_{mul} \uplus_{1\leq j\leq k} o(M_2^j, Q_2, x_j)$.

Thus, finally, we obtain $\uplus_{1\leq j\leq k} M_1^j >_{mul} \uplus_{1\leq j\leq k} o(Lvl_\Gamma(v_j), Q_2, x_j)$ and $\mathcal{M}^1_{\mathcal{D}'}(Q') <_{mul} \mathcal{M}^1_{\mathcal{D}}(Q)$.

- Case ($\mathbf{EndRep}_\mathrm{T}$) is treated similarly. □

Theorem 4.13 (Termination)**.** If $\mathcal{D} : (\Gamma \vdash P : N)$, then P terminates.

Proof. Suppose, towards a contradiction, that process P diverges. Then so does $\mathbf{comp}(P)$. We thus have an infinite sequence $(Q_i)_{i\geq 0}$ such that $Q_0 = \mathbf{comp}(P)$ and $\forall i$, $Q_i \to^! Q_{i+1}$. We can apply Lemma 4.12 to each Q_i to obtain an infinite sequence $(\mathcal{D}^i)_{i\geq 0}$ s.t. $\forall i$, $\mathcal{D}^i : (\Gamma \vdash_+ Q_i : N_i)$ and an infinite sequence $(\mathcal{M}^1_{\mathcal{D}^i}(Q_i), \mathcal{M}^2_{\mathcal{D}^i}(Q_i))$ such that $\forall i$, $\mathcal{M}^1_{\mathcal{D}^i}(Q_i) >_{mul} \mathcal{M}^1_{\mathcal{D}^{i+1}}(Q_{i+1})$ or $\mathcal{M}^1_{\mathcal{D}^i}(Q_i) = \mathcal{M}^1_{\mathcal{D}^{i+1}}(Q_{i+1})$ and $\mathcal{M}^2_{\mathcal{D}^i}(Q_i) >_{mul} \mathcal{M}^2_{\mathcal{D}^{i+1}}(Q_{i+1})$. This contradicts the well-foundedness of $>_{mul}$. □

4.3. Encoding separate choice

To illustrate the expressiveness of our type system for $\mathrm{HOpi}^!_\omega$, we present the encoding of the separate choice operator. Separate choice here means that operator $+$ is applied only to inputs, or only to outputs.

Table 9
Separate choice in $\mathrm{HOpi}^{!}_{\omega}$.

Outputs:

$$\begin{aligned}
&\left[\textstyle\sum_{i=1}^{n} \overline{x_i}\langle d_i\rangle.P_i\right] = \\
&(\nu s)\,(\,s(f_v).(f_v\lfloor \texttt{true}\rfloor \mid !s(f_v).f_v\lfloor \texttt{false}\rfloor) \\
&\mid (\nu a)\textstyle\prod_{i=1}^{n} \overline{x_i}\langle d_i, x \mapsto \overline{s}\langle x\rangle, y \mapsto \overline{a}\langle y\rangle\rangle.a(x').\texttt{if } x' \texttt{ then } [P_i] \\
&\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \texttt{else } \mathbf{0}\,)
\end{aligned}$$

Inputs:

$$\begin{aligned}
&\left[\textstyle\sum_{i=1}^{m} y_i(z).Q_i\right] = \\
&(\nu r)\,(\,\overline{r}\langle \texttt{true}\rangle \\
&\quad \mid \textstyle\prod_{i=1}^{m}(\nu g)\,(\,\overline{g}\langle\bigstar\rangle \\
&\quad\quad \mid !g(t).y_i(z, f_s, f_a).r(x).\texttt{if } x \\
&\quad\quad\quad \texttt{then } (\nu u)\,(\,f_s\lfloor x \mapsto \overline{u}\langle x\rangle\rfloor \mid u(y).\texttt{if } y \\
&\quad\quad\quad\quad\quad \texttt{then } \overline{r}\langle \texttt{false}\rangle \mid f_a\lfloor \texttt{true}\rfloor \mid [Q_i] \\
&\quad\quad\quad\quad\quad \texttt{else } \overline{r}\langle \texttt{true}\rangle \mid f_a\lfloor \texttt{false}\rfloor \mid \overline{g}\langle\bigstar\rangle\,) \\
&\quad\quad\quad \texttt{else } \overline{r}\langle \texttt{false}\rangle \mid \overline{y_i}\langle z, f_s, f_a\rangle\,))
\end{aligned}$$

The protocol in $\mathrm{HOpi}^{!}_{\omega}$ is presented in Table 9; we make use of notation $\prod_{i=1}^{n} P_i$ to represent $P_1 | \ldots | P_n$, and we write $[P]$ for the encoding of process P. The protocol is designed to let sums of output processes (the emitters) synchronise with sums of input processes (the receivers), whenever matching actions can be found. It works as follows. Any output action of an emitter may proceed. Whenever a matching input action exists, a mechanism of locks is used to ensure that at most one branch has been chosen on the emitter's side, and the same on the receiver's side. If this is not the case, the protocol backtracks, and the initial output action that has started executing is cancelled. Channels s and r are used to implement two locks, that are tested on the receiver's side to decide whether the corresponding branch in the sum of inputs is allowed to proceed. When this is not the case, the protocol backtracks. The reader is referred to [6] for a more detailed description of the protocol: ours closely follows the steps of Nestmann's original proposal.

Because of backtracking, and of the inherent complexity of the processes being manipulated, the analysis of the protocol in terms of termination is non-trivial. Also, when rewritten in the higher-order paradigm (more precisely, in $\mathrm{HOpi}^{!}_{\omega}$), the protocol makes use of some patterns or combinations of operators that are delicate for termination (in particular, a pattern similar to $a(X).b(Y).X$, as discussed in Section 4). The proof that the original protocol does not add divergence is given in [6], while [3] uses a type system to derive the same result.

Several details have to be changed or adapted w.r.t. the protocol in [6] to program separate choice in our setting. For instance, in [6], there may be a sequence of requests on channel s, the first of which receives answer `true`, and answer `false` is given to all following requests. In our protocol, this behaviour has been "hardwired" in the definition of the emitters, to clarify the encoding.

Names a and r are given the simple type $Ch(\texttt{bool})$ and g is given the simple type $Ch(\texttt{unit})$, like in the original process. Instead of sending channel a, which we cannot do since our calculus does not feature name passing, we send a function $f_a : \texttt{bool} \to \diamond$ which allows the process that encodes a sum of inputs to output boolean values on a. The case of s is more complex, because the *input capability* on s is transmitted in the protocol of [6]: the process encoding the sum of inputs performs an input on s after receiving s. The protocol of Table 9 exploits an encoding of the π-calculus into $\mathrm{HOpi}^{!}_{\omega}$, more precisely, of the *localised* π-calculus [10, Section 5.6]. Accordingly, a function of higher-order is transmitted in place of s in our encoding: upon reception of this function $f_s : (\texttt{bool} \to \diamond) \to \diamond$, it is applied to a function $f_u : \texttt{bool} \to \diamond$ (which intuitively represents the input capability), and this finally allows the process which sent f_s (the process encoding a sum of outputs) to transmit boolean values on channel $u : Ch(\texttt{bool})$ to the process encoding a sum of inputs. The latter protocol, in which functions are transmitted and applied, illustrates the higher-order nature of $\mathrm{HOpi}^{!}_{\omega}$.

Typing the processes. As stated above, the protocol does not add divergence. We rely on the type system of Section 4.2 to show that our encoding of this protocol does not add non-typability; that is, if the processes Q_i and P_i are typable, the whole process is typable. Indeed, typing the processes given in Table 9 is possible provided the continuation processes P_i, Q_i can be typed. When this is the case, we must use, to type our protocol, levels that are strictly greater than those used to type the P_i, Q_i, which is always possible. In what follows, we ignore this point, and assume the context $\emptyset$ is sufficient to type the P_i, Q_i with the global weight $\emptyset$. Adapting the typing to a situation where P_i, Q_i have non-$\emptyset$ weights is not conceptually difficult. For the same reasons, we ignore the level of the values d_i sent by the emitters. Instead, we just assume they have a well-formed type T of level $\emptyset$. It is easy to adapt the typing to a situation where the level of T is a given multiset M.

Proposition 4.14 (Typing the encoding of separate choice). *Consider two sets of processes* $(P_i)_{i=1,\dots,n}$ *and* $(Q_i)_{i=1,\dots,m}$ *such that* $\emptyset \vdash P_i : \emptyset$, *and* $z : T \vdash Q_i : \emptyset$ *for some name* z *and type* T.

Then $\left[\sum_{i=1}^{n} \overline{x_i}\langle d_i\rangle.P_i\right] \mid \left[\sum_{i=1}^{m} y_i(z).Q_i\right]$ *does not exhibit a divergence.*

Proof. We establish this by applying Theorem 4.13. For this, we construct a typing derivation in which we assign types to the names used in Table 9. The type assignement is as follows (we introduce $T_0 = \mathtt{bool}^{\emptyset} \to^{\{0,0\}} \diamond$):

$$
\begin{array}{ll}
t : \mathtt{unit}^{\emptyset} & g : Ch^{\{4\},\{0\}}(\mathtt{unit}^{\emptyset}) \\
x_i, y_i : Ch^{\{4\},\{3\}}(T, T_0 \to^{\{2,0\}} \diamond, T_0) & s : Ch^{\{2\},\{1\}}(T_0) \\
r : Ch^{\{0\},\{0\}}(\mathtt{bool}^{\emptyset}) & a : Ch^{\{0\},\{0\}}(\mathtt{bool}^{\emptyset}) \\
u : Ch^{\{0\},\{0\}}(\mathtt{bool}^{\emptyset}) & f_a : \mathtt{bool}^{\emptyset} \to^{\{0,0\}} \diamond \\
f_u, f_v : \mathtt{bool}^{\emptyset} \to^{\{0,0\}} \diamond & f_s : T_0 \to^{\{2,0\}} \diamond \\
z, d_i : T & x, y : \mathtt{bool}^{\emptyset}.
\end{array}
$$

We do not give explictly the construction of the whole derivation, but explain why every subprocess is typable. We introduce some notations that we use in the following calculations: we write $\mathsf{W}(a)$ for the weight of a (we overload notations and write $\mathsf{W}(P)$ for the weight of a process P), $\mathsf{C}(a)$ for the capacity of a, and $\mathsf{L}(v)$ for the level of v.

- $s(f_v).(f_v\lfloor\mathtt{true}\rfloor \mid !s(f_v).f_v\lfloor\mathtt{false}\rfloor)$. To use rule ($\mathbf{In}_{\mathrm{T}}$) here, we need to check that $\mathsf{C}(s) = \{1\}$ is greater than $\mathsf{L}(f_v) = \{0, 0\}$, and that $\mathsf{W}(s) = \{3\}$ is greater than $o(\{2\}, (f_v\lfloor\mathtt{true}\rfloor \mid !s(f_v).f_v\lfloor\mathtt{false}\rfloor), f_v) = \{2\}$ (Remember that, by Definition 4.2, $o(M, !a(y).P, x) = \emptyset$).
- $!s(f_v).f_v\lfloor\mathtt{false}\rfloor$. To use rule ($\mathbf{Rep}_{\mathrm{T}}$) here, we need to check that $\mathsf{C}(s) = \{1\}$ is greater than $\mathsf{L}(f_v) = \{0, 0\}$, and that $\mathsf{W}(s) = \{3\}$ is greater than $o(\{2\}, f_v\lfloor\mathtt{false}\rfloor, f_v) \uplus \mathsf{W}(f_v\lfloor\mathtt{false}\rfloor) = \{2, 2\}$.
- $\overline{x_i}\langle d_i, x \mapsto \overline{s}\langle x\rangle, y \mapsto \overline{a}\langle y\rangle\rangle$; this output is well-typed as $\mathsf{W}(s) = 2$, hence $\mathsf{L}(x \mapsto \overline{s}\langle x\rangle) = \{2, 0\}$, which is smaller than $\{3\} = \mathsf{C}(x_i)$.
  A similar reasoning holds for $y \mapsto \overline{a}\langle y\rangle$, which has level $\{0, 0\}$. Moreover, $\mathsf{L}(d_i) = \emptyset$ is smaller than $\{3\} = \mathsf{C}(x_i)$.
- The inputs $a(x')$, $u(y)$ are typed easily, because the types we assume for a and u impose level $\emptyset$ for the boolean variables x, y.
- $f_s\lfloor x \mapsto \overline{u}\langle x\rangle\rfloor$. The application is well-typed, because $\mathsf{W}(u) = \{0\}$, which gives a type compatible with the type we have assumed for f_s.
- The crux of this proof is the type-checking of the replicated subterm $!g(t).y_i(z, f_s, f_a).r(x).C$ (C is the continuation process, which can be deduced from the definition of the process in Table 9).
  In order to apply rule ($\mathbf{Rep}_{\mathrm{T}}$), we have to check that the two domination conditions hold. The condition $Lvl(T_i) <_{mul} M_2^i$ is fulfilled because:
  - $\mathsf{C}(g) = \{0\}$ is greater than $\mathsf{L}(y) = \emptyset$,
  - $\mathsf{C}(y_i) = \{3\}$ is greater than $\mathsf{L}(z) = \emptyset$, $\mathsf{L}(f_s) = \{2, 0\}$ and $\mathsf{L}(f_a) = \{0, 0\}$,
  - $\mathsf{C}(r) = \{0\}$ is greater than $\mathsf{L}(x) = \emptyset$.

  As far as the other condition is concerned, we have to compute $\mathsf{W}(C)$ and the contributions $o(\{3\}, C, f_s)$, $o(\{3\}, C, f_a)$, $o(\{3\}, C, z)$ and $o(\{0\}, C, x)$. Because of rule ($\mathbf{If}_{\mathrm{Pa}}$), and by Definition 4.2, we have to compute these values for each branch in the nested conditional tests (we call these C_j, for $j = 1, 2, 3$), and compute the maximum.
  - We have $\mathsf{W}(C_1) = \mathsf{L}(f_s) \uplus \mathsf{L}(f_a) \uplus \mathsf{W}(r) = \{2, 0, 0, 0\}$ (remember Q_i has weight $\emptyset$). If we suppose that z appears c_1 times in Q_i, not in object position nor inside the continuation of a replication, we have $o(\{3\}, z, C_1) = c_1.\{3\}$, $o(\{3\}, C_1, f_s) = \{3\}$, $o(\{3\}, C_1, f_a) = \{3\}$, $o(\{0\}, C_1, x) = \{0\}$.
  - We have $\mathsf{W}(C_2) = \mathsf{L}(f_s) \uplus \mathsf{L}(f_a) \uplus \mathsf{W}(r) \uplus \mathsf{W}(g) = \{4, 2, 0, 0, 0\}$. We have $o(\{3\}, z, C_2) = \emptyset$, $o(\{3\}, C_2, f_s) = \{3\}$, $o(\{3\}, C_2, f_a) = \{3\}$, $o(\{0\}, C_2, x) = \{0\}$.
  - We have $\mathsf{W}(C_3) = \mathsf{W}(r) \uplus \mathsf{W}(y_i) = \{4, 0\}$. We have $o(\{3\}, z, C_3) = \emptyset$, $o(\{3\}, C_3, f_s) = \emptyset$, $o(\{3\}, C_3, f_a) = \emptyset$, $o(\{0\}, C_3, x) = \{0\}$.

  Rule ($\mathbf{If}_{\mathrm{T}}$) allows us to compute $\mathsf{W}(C) = \{4, 2, 0, 0, 0\}$. Definition 4.2 gives $o(\emptyset, C, t) = \emptyset$, $o(\{3\}, C, f_s) = \{3\}$, $o(\{3\}, C, f_a) = \{3\}$, $o(\{3\}, C, z) = c_1.\{3\}$ and $o(\{0\}, C, x) = \{0\}$. We can thus apply rule ($\mathbf{Rep}_{\mathrm{T}}$) as $\{4, 4, 0\}$, which is the multiset sum of the weights of g, y_i, r, is strictly greater than $\{4, 3, 3, 2, 0, 0, 0, 0\} \uplus c_1.\{3\}$, the multiset sum of the global weight of C and the contribution of the capacities of g, y_i, r in C. □

The last item above illustrates the usefulness of the treatment of sequences of input prefixes: indeed we need to apply rule ($\mathbf{Rep}_{\mathrm{T}}$) in a non-trivial way (more precisely, by treating together names g and y_i) in order to type-check this part of the process.

5. Controlling communication and passivation

5.1. Paπ: a calculus with locations and passivation

The objective of this section is to study termination in presence of further constructs that are known to be challenging in the semantics of higher-order concurrent languages, notably constructs of locations (i.e., explicit spatial distribution) and of passivation. We consider a calculus, which we refer to as Paπ (for '*Pa*ssivation *Pi*-calculus'), that combines such constructs with the higher-order features of HOpi_2 and the name-passing capabilities of the π-calculus.

We start by defining Paπ and its operational semantics. As in the previous sections, lowercase letters, $a, b, c, \ldots, l, \ldots, p, q, \ldots x, y, \ldots$, will be used to range over names. We adopt some conventions to distinguish several usages of names: we write a, b, c for names used as channels, l for names used as locations, and x, y for names used in input variable position. The distinction will be ensured by the type system, presented below. The grammar of Paπ processes is as follows:

$$P ::= \mathbf{0} \mid P|P \mid (\nu p)\, P \mid \overline{p}\langle q\rangle.P \mid p(x).P \mid !p(x).P \\ \mid p(\!|P|\!) \mid \overline{p}\langle P\rangle.P \mid p(X).P \mid p(X) \triangleright P \mid X.$$

As in Section 4, replication is allowed only on name-passing input prefixes. $l(\!|P|\!)$ stands for the process P running at location l (locations can be nested). The construct $l(X) \triangleright P$ corresponds to passivation: such a process is willing to freeze a computation running at location l, call it X and proceed according to P. For instance, in the process

$$l(\!|T|\!) \mid l(X) \triangleright (\nu l')\, (\overline{a}\langle X\rangle \mid l'(\!|X|\!)),$$

T can execute, until at some point location l is passivated. When this happens, a copy of the process at l is sent along channel a, and computation resumes at a new location named l'. Passivation can be found in calculi like Kells [5,11] or Homer [4].

The definition of structural congruence in Paπ inherits the laws of $\equiv$ in HOpi_2. No specific law is introduced for locations – in particular, restrictions are not allowed to cross-location boundaries.

Reduction in Paπ is defined by the following inference rules:

$$(\mathbf{ComN_P})\ \frac{}{\overline{p}\langle q\rangle.P_1 \mid p(x).P_2 \to P_1 \mid P_2[q/x]} \qquad (\mathbf{Trig_P})\ \frac{}{\overline{p}\langle q\rangle.P_1 \mid !p(x).P_2 \to P_1 \mid P_2[q/x] \mid !p(x).P_2}$$

$$(\mathbf{ComP_P})\ \frac{}{\overline{p}\langle Q\rangle.P_1 \mid p(X).P_2 \to P_1 \mid P_2[Q/X]} \qquad (\mathbf{Pass_P})\ \frac{}{l(\!|Q|\!) \mid l(X) \triangleright P \to P[Q/X]} \qquad (\mathbf{Local_P})\ \frac{P \to P'}{l(\!|P|\!) \to l(\!|P'|\!)}$$

$$(\mathbf{Spect_P})\ \frac{P \to P'}{P|Q \to P'|Q} \qquad (\mathbf{Cong_P})\ \frac{P \equiv P' \quad P' \to Q' \quad Q' \equiv Q}{P \to Q} \qquad (\mathbf{Scop_P})\ \frac{P \to P'}{(\nu p)\, P \to (\nu p)\, P'}$$

It has to be noted that we do not claim here that the combination of primitives provided in Paπ (essentially, first and higher-order message passing, localised interaction, passivation) makes this calculus a proposal for a model for distributed or component-based programming, as is the case for the process calculi mentioned above [4,5,11]. Indeed, important interaction mechanisms such as communication between distant locations, subjective mobility, or dynamic binding of names, are not available in Paπ.

Our primary goal is instead to study how the constructs of Paπ, which have the advantage of being presented in a rather simple way, can be taken into account in our termination analysis. We believe that the way we handle these can be smoothly adapted to small variations: for instance, typing distant communication in kπ [5] should be feasible in pretty much the same way we type local communication in Paπ.

Example 5.1. We now provide a few examples of Paπ processes to illustrate typical idioms that can be programmed using passivation.

$$\mathit{Dup}\quad c(r).l(X) \triangleright \left(l(\!|X|\!) \mid (\nu l')\, (\overline{r}\langle l'\rangle \mid l'(\!|X|\!)) \right)$$

$$\mathit{Res}\quad c(l).l(X) \triangleright l(\!|P_0|\!) \qquad \mathit{DynUpd}\quad c(l).d(X).(\, l(Y) \triangleright l(\!|X|\!)\,)$$

$$\mathit{Coloc}\quad l_1(X) \triangleright \left(l_2(Y) \triangleright (l_1(\!|X|Y|\!) \mid l_2(\!|\mathbf{0}|\!))\right)$$

We briefly explain these definitions.

Dup performs code duplication: when a message is received on channel c, the computation running at location l is duplicated, and the location of the new copy is sent back on r, the channel transmitted along c.

Table 10
Typing rules for Paπ.

$$(\mathbf{Nil}_{\mathrm{Pa}})\ \frac{}{\Gamma \vdash \mathbf{0} : 0} \qquad (\mathbf{Var}_{\mathrm{Pa}})\ \frac{\Gamma(X) = n}{\Gamma \vdash X : n} \qquad (\mathbf{Res}_{\mathrm{Pa}})\ \frac{\Gamma, p : T_V \vdash P : n}{\Gamma \vdash (\nu p)\, P : n}$$

$$(\mathbf{Par}_{\mathrm{Pa}})\ \frac{\Gamma \vdash P_1 : n_1 \quad \Gamma \vdash P_2 : n_2}{\Gamma \vdash P_1 \mid P_2 : \max(n_1, n_2)} \qquad (\mathbf{Pas}_{\mathrm{Pa}})\ \frac{\Gamma, X : k-1 \vdash P : n \quad \Gamma(p) = \mathbf{loc}^k}{\Gamma \vdash p(X) \triangleright P : n}$$

$$(\mathbf{Loc}_{\mathrm{Pa}})\ \frac{\Gamma(p) = \mathbf{loc}^k \quad \Gamma \vdash Q : n \quad k > n}{\Gamma \vdash p(\!|Q|\!) : k} \qquad (\mathbf{InP}_{\mathrm{Pa}})\ \frac{\Gamma, X : k-1 \vdash P : n \quad \Gamma(p) = Ch^k(\diamond)}{\Gamma \vdash p(X).P : n}$$

$$(\mathbf{OutP}_{\mathrm{Pa}})\ \frac{\Gamma \vdash P : n \quad \Gamma \vdash Q : n' \quad \Gamma(p) = Ch^k(\diamond) \quad k > n'}{\Gamma \vdash \overline{p}\langle Q\rangle.P : \max(k, n)}$$

$$(\mathbf{InN}_{\mathrm{Pa}})\ \frac{\Gamma, x : T_V \vdash P : n \quad \Gamma(p) = Ch^k(T_V)}{\Gamma \vdash p(x).P : n}$$

$$(\mathbf{OutN}_{\mathrm{Pa}})\ \frac{\Gamma \vdash P : n \quad \Gamma(q) = T_V \quad \Gamma(p) = Ch^k(T_V)}{\Gamma \vdash \overline{p}\langle q\rangle.P : \max(k, n)}$$

$$(\mathbf{Rep}_{\mathrm{Pa}})\ \frac{\Gamma, x : T_V \vdash P : n \quad \Gamma(p) = Ch^k(T_V) \quad k > n}{\Gamma \vdash !p(x).P : 0}$$

Process *Res* (reset): upon reception of a location name l along c, the computation taking place at l is replaced with P_0, that can be considered as a start state. Essentially the same "program" can be used when we want to replace the code running at l with a new version, that is transmitted along some channel d: this is a form of dynamic update (process *DynUpd*).

"Co-localisation": processes running at locations l_1 and l_2 are put together, and computation proceeds within location l_1. This might trigger new interactions between formerly separated processes. This is a form of *objective mobility* (running computations are being moved around).

5.2. Controlling termination in Paπ

In Paπ, divergences arise both from recursion in usages of the passivation and process-passing mechanisms, and from recursive calls in the continuation of replicated (name passing) inputs. We control the latter source of divergences by resorting to the type discipline of [3], while the former is controlled by associating levels to locations and to process-carrying channels, along the lines of the type systems we have studied in the previous sections.

However, the mere superposition of these two systems (of Section 2.2 and of [3]) does not work, as the two mechanisms can cooperate to produce divergences. This can be illustrated by looking at process

$$S_5 \;=\; l(X) \triangleright !a(y).X \;\mid\; l(\!|\overline{a}\langle p\rangle|\!) \;\mid\; \overline{a}\langle p\rangle.$$

This process is divergent, but, unfortunately, the usages of passivation (which can be treated as a form of process passing) and name passing in S_5 are compliant with the principles of the aforementioned type systems. In this particular case, we must take into account the fact that X can be instantiated by a process containing an output on a channel having the same level as a. More generally, we must understand how the two type systems can interact, in order to avoid diverging behaviours.

The following grammar introduces the types for processes, location names, channels and values (to be transmitted along channels):

$$T_P = m \qquad T_L = \mathbf{loc}^m \qquad T_C = Ch^m(T_V) \qquad T_V = T_L \mid T_C \mid \diamond$$

In Paπ, every entity (process, location, name-passing channel and process-passing channel) is given a level which is used to control the two sources of divergences discussed above. The level of a name-passing channel a corresponds to the maximum level allowed for the continuation P in a replicated input of the form $!a(x).P$. The level of a process-passing channel a corresponds to the maximum level of a process sent on a. Similarly, the level of a location l corresponds to the maximum level a process executing at l can have. In turn, the level of a process P corresponds to the maximum level of messages and locations that occur in P neither within a higher-order output nor under a replication.

The rules defining the type system for termination in Paπ are given in Table 10. As far as typing termination is concerned, we treat higher-order inputs (resp. outputs) like passivations (resp. located processes).

Example 5.2 (*Typing examples*)**.** Process S_5 seen above cannot be typed. The typing rule for locations forces the level of location l to be strictly greater than $lvl(a)$ when typing $l(\!|\overline{a}\langle p\rangle|\!)$. The typing rule for passivation forces the level of l to be equal to $1 + lvl(X)$. Thus $lvl(X) \leq lvl(a)$ and the typing rule for replicated inputs cannot be applied to $!a(y).X$.

For process *Coloc* to be typable, $lvl(l_1)$, the level assigned to l_1, should be greater than $lvl(l_2)$. In this case, we can observe that, thanks to typing, we know it is safe to take two processes running in separate locations and let them run in parallel, as *Coloc* does: while this might trigger new interactions (inter-locations communication is forbidden in Paπ), this is of no harm for termination.

Remark 5.3 (*An extension of two type systems*)**.** We can remark that the sub-set consisting of rules ($\mathbf{Nil}_{\mathrm{Pa}}$), ($\mathbf{Var}_{\mathrm{Pa}}$), ($\mathbf{Res}_{\mathrm{Pa}}$), ($\mathbf{Par}_{\mathrm{Pa}}$), ($\mathbf{InP}_{\mathrm{Pa}}$),($\mathbf{OutP}_{\mathrm{Pa}}$) corresponds exactly to the type system for HOpi$_2$ introduced in Section 2. Hence, every HOpi$_2$ process that is typable according to the rules of Table 3 is typable as a Paπ process using rules of Table 10.

Moreover, the type system of Table 10 subsumes the type system of [3] for the π-calculus: if a π-calculus process P is typable according to [3], then it is typable as a Paπ process.

Remark 5.4. It has to be noted that the type system we present can be made more expressive by exploiting ideas from Section 4. Indeed, we associate a *unique* level to names, and we could instead use three natural numbers to type a name: one would be its weight, and the other two would be interpreted as capacities, used to control the two sources of recursion: the weight of name passing outputs on one side, and the weight of process passing outputs and located processes on the other side. In what we have presented, these three components of the type of a name are merged into a single one. Additionally, sequences of inputs could be analysed according to the ideas of Section 4.

5.3. Correctness of the type system

The soundness proof for our type system essentially follows the same strategy as in the previous sections. At its core is the definition of a measure on processes, that takes into account the contribution of locations and first- and higher-order outputs that do not occur within a message.

The type system for Paπ enjoys as usual the properties of weakening and strengthening (omitted), as well as the preservation of typability under $\equiv$:

Lemma 5.5. *If $P \equiv P'$ then $\Gamma \vdash P : n$ iff $\Gamma \vdash P' : n$.*

Proof. By induction on the derivation of $P \equiv P'$. □

Definition 5.6. Given a Paπ process P, we associate to a typing derivation $\mathcal{D} : (\Gamma \vdash P : n)$ a multiset, noted $\mathcal{M}_{\mathcal{D}}(P)$, and defined as follows:

- $\mathcal{M}_{\mathcal{D}}(\mathbf{0}) = \mathcal{M}_{\mathcal{D}}(X) = \emptyset$;
- $\mathcal{M}_{\mathcal{D}}(P_1 \mid P_2) = \mathcal{M}_{\mathcal{D}^1}(P_1) \uplus \mathcal{M}_{\mathcal{D}^2}(P_2)$ where $\mathcal{D}$ is obtained from premises $\mathcal{D}^1 : (\Gamma \vdash P_1 : n_1)$ and $\mathcal{D}^2 : (\Gamma \vdash P_2 : n_2)$ for some n_1, n_2;
- $\mathcal{M}_{\mathcal{D}}((\nu p)\, P_1) = \mathcal{M}_{\mathcal{D}^1}(P_1)$ where $\mathcal{D}$ is obtained from premise $\mathcal{D}^1 : (\Gamma, p : T_V \vdash P_1 : n)$ for some T_V;
- $\mathcal{M}_{\mathcal{D}}(l(X) \triangleright P_1) = \mathcal{M}_{\mathcal{D}^1}(P_1)$ where $\mathcal{D}$ is obtained from premise $\mathcal{D}^1 : (\Gamma, X : k-1 \vdash P_1 : n)$;
- $\mathcal{M}_{\mathcal{D}}(l(\!|Q|\!)) = \mathcal{M}_{\mathcal{D}^1}(Q) \uplus \{n\}$ where $\mathcal{D}$ is obtained from premise $\mathcal{D}^1 : (\Gamma \vdash Q : n')$, for some n' and $\Gamma(l) = \mathbf{loc}^n$.
- $\mathcal{M}_{\mathcal{D}}(p(X).P_1) = \mathcal{M}_{\mathcal{D}^1}(P_1)$ where $\mathcal{D}$ is obtained from premise $\mathcal{D}^1 : (\Gamma, X : k-1 \vdash P_1 : n)$;
- $\mathcal{M}_{\mathcal{D}}(\overline{p}\langle Q\rangle.P_1) = \mathcal{M}_{\mathcal{D}^1}(P_1) \uplus \{k\}$ where $\mathcal{D}$ is obtained from premises $\mathcal{D}^1 : (\Gamma \vdash Q : n_1)$, $\mathcal{D}^2 : (\Gamma \vdash P_1 : n_2)$ for some n_1, n_2 and $lvl_\Gamma(p) = k$;
- $\mathcal{M}_{\mathcal{D}}(\overline{p}\langle q\rangle.P_1) = \mathcal{M}_{\mathcal{D}^1}(P_1) \uplus \{k\}$ where $\mathcal{D}$ is obtained from premise $\mathcal{D}^1 : (\Gamma \vdash P_1 : n_1)$ for some n_1, $\Gamma(a) = Ch^k(T)$ for some k, T s.t. $n = \max(k, n_1)$;
- $\mathcal{M}_{\mathcal{D}}(p(x).P_1) = \mathcal{M}_{\mathcal{D}^1}(P_1)$ where $\mathcal{D}$ is obtained from premise $\mathcal{D}^1 : (\Gamma, x : T_V \vdash P_1 : n)$ for some T_V;
- $\mathcal{M}_{\mathcal{D}}(!p(x).P_1) = \emptyset$;

Lemma 5.7. *Let Γ be a typing context, N a multiset of natural numbers and n a natural number. If $P \equiv P'$ then there exists $\mathcal{D} : (\Gamma \vdash P : n)$ with $\mathcal{M}_{\mathcal{D}}(P) = N$ iff there exists $\mathcal{D}' : (\Gamma' \vdash P' : n)$ with $\mathcal{M}_{\mathcal{D}'}(P') = N$.*

Proof. Easily proved by induction on the derivation of $P \equiv P'$. □

As reduction in Paπ may involve two kinds of substitutions (for name variables or process variables) the 'subject substitution' lemma is decomposed into two properties, which we prove below.

Lemma 5.8. *If $\mathcal{D} : (\Gamma, x : T_V \vdash P : n)$ and $\Gamma(q) = T_V$, then there exists $\mathcal{D}'$ s.t. $\mathcal{D}' : (\Gamma \vdash P[q/x] : n')$ for some $n' \leq n$ and $\mathcal{M}_{\mathcal{D}'}(P[q/x]) = \mathcal{M}_{\mathcal{D}}(P)$.*

Proof. We reason by induction on the typing derivation:

- Cases ($\mathbf{Nil}_{\mathrm{Pa}}$), ($\mathbf{Var}_{\mathrm{Pa}}$), ($\mathbf{Par}_{\mathrm{Pa}}$) and ($\mathbf{Res}_{\mathrm{Pa}}$) are treated easily using the induction hypotheses (where relevant) as well as Definition 5.6.
- Case ($\mathbf{Loc}_{\mathrm{Pa}}$). Suppose $P = x(\!|Q_1|\!)$ (the case $P = l(\!|Q_1|\!)$ with $l \neq x$ can be deduced from the following). Then $T_V = \mathbf{loc}^n$, and $\mathcal{D}$ is derived using ($\mathbf{Loc}_{\mathrm{Pa}}$) from premise $\mathcal{D}^1 : (\Gamma, x : T_V \vdash Q_1 : n_1)$, for some n_1 s.t. $n > n_1$.

  The induction hypothesis gives $\mathcal{D}^{(1)} : (\Gamma \vdash Q_1[q/x] : n_1')$ for some $n_1' \leq n_1$ and $\mathcal{M}_{\mathcal{D}^{(1)}}(Q_1[q/x]) = \mathcal{M}_{\mathcal{D}^1}(Q_1)$. As $\Gamma(q) = T_V = \mathbf{loc}^n$ and $n > n_1 \geq n_1'$, we can construct

$$\mathcal{D}' = (\mathbf{Loc}_{\mathrm{Pa}})\frac{\mathcal{D}^{(1)}}{\Gamma \vdash q(\!|Q_1[q/x]|\!) : n}.$$

  As q and x have the same type, T_V, we can use use Definition 5.6 to conclude.
- Case ($\mathbf{Pas}_{\mathrm{Pa}}$). Suppose $x(X) \triangleright P_1$ (the case $P = l(X) \triangleright P_1$ and $l \neq x$ can easily be deduced from the following). There exists k s.t. $T_V = \mathbf{loc}^k$ and $\mathcal{D}$ is obtained by applying rule ($\mathbf{Pas}_{\mathrm{Pa}}$), with premise $\mathcal{D}^1 : (\Gamma, X : k-1, x : T_V \vdash P_1 : n)$. Induction gives $\mathcal{D}^{(1)} : (\Gamma, X : k-1 \vdash P_1[q/x] : n')$, for some $n' \leq n$. As $\Gamma(q) = \mathbf{loc}^k$ we can construct

$$\mathcal{D}' = (\mathbf{Pas}_{\mathrm{Pa}})\frac{\mathcal{D}^{(1)}}{\Gamma \vdash q(X) \triangleright P_1[q/x] : n'}.$$

  As q and x have same type T_V, we conclude using Definition 5.6.
- Case ($\mathbf{InP}_{\mathrm{Pa}}$) is treated like case ($\mathbf{Pas}_{\mathrm{Pa}}$).
- Case ($\mathbf{OutP}_{\mathrm{Pa}}$). Suppose $P = \overline{x}\langle Q_2\rangle.P_1$ (the case $P = \overline{p}\langle Q\rangle.P_1$ and $p \neq x$ can easily be deduced from the following). There exists k s.t. $T = Ch^k(\diamond)$ and $\mathcal{D}$ is built using rule ($\mathbf{OutP}_{\mathrm{Pa}}$), from premises $\mathcal{D}^1 : (\Gamma, x : T_V \vdash P_1 : n_1)$ and $\mathcal{D}^2 : (\Gamma, x : T_V \vdash Q_2 : n_2)$, with $k > n_2$ and $n = \max(k, n_1)$ for some n_1, n_2. The induction hypothesis gives $\mathcal{D}^{(1)}, \mathcal{D}^{(2)}, n_1', n_2'$ s.t. $\mathcal{D}^{(1)} : (\Gamma \vdash P_1[q/x] : n_1')$, $\mathcal{D}^{(2)} : (\Gamma \vdash Q_2[q/x] : n_2')$, $n_1' \leq n_1$, $n_2' \leq n_2$, $\mathcal{M}_{\mathcal{D}^1}(P_1) = \mathcal{M}_{\mathcal{D}^{(1)}}(P_1[q/x])$ and $\mathcal{M}_{\mathcal{D}^2}(Q_2) = \mathcal{M}_{\mathcal{D}^{(2)}}(Q_2[q/x])$.

  As $k > n_2 \geq n_2'$ and $\Gamma(q) = Ch^k(\diamond)$, we construct

$$\mathcal{D}' = (\mathbf{OutP}_{\mathrm{Pa}})\frac{\mathcal{D}^{(1)} \quad \mathcal{D}^{(2)}}{\Gamma \vdash \overline{q}\langle Q_2[q/x]\rangle.(P_1[q/x]) : \max(k, n_1')}.$$

  We conclude by stating that $\max(k, n_1') \leq \max(k, n_1)$ and relying on Definition 5.6 to obtain the result on the measure.
- Case ($\mathbf{InN}_{\mathrm{Pa}}$). Suppose $P = x(y).P_1$ (the case $P = p(y).P_1$ and $p \neq x$ can be deduced from the following). As our processes abide the Barendregt Convention, $y \neq x$. There exists k, T_V' s.t. $T_V = Ch^k(T_V')$ and $\mathcal{D}$ is obtained using ($\mathbf{InN}_{\mathrm{Pa}}$) from premise $\mathcal{D}^1 : (\Gamma, x : T_V, y : T_V' \vdash P_1 : n)$. The induction hypothesis gives $\mathcal{D}^{(1)} : (\Gamma, y : T_V' \vdash P_1[q/x] : n')$ for some $n' \leq n$. As $\Gamma(q) = Ch^k(T_V')$ we can construct

$$\mathcal{D}' = (\mathbf{InN}_{\mathrm{Pa}})\frac{\mathcal{D}^{(1)}}{\Gamma \vdash q(y).P_1[q/x] : n'}.$$

  As x and q have the same type, we can conclude using Definition 5.6.
- Case ($\mathbf{OutN}_{\mathrm{Pa}}$). Suppose $P = \overline{x}\langle q'\rangle.P_1$ (the case $P = \overline{p}\langle q'\rangle.P_1$ and $p \neq x$ can easily be deduced from the following). There exists k, T_V' s.t. $T = Ch^k(T_V')$, $\Gamma(q') = T_V'$ and $\mathcal{D}$ is obtained using rule ($\mathbf{OutN}_{\mathrm{Pa}}$) from premise $\mathcal{D}^1 : (\Gamma, x : T_V \vdash P_1 : n_1)$ and $n = \max(k, n_1)$. The induction hypothesis gives $\mathcal{D}^{(1)} : (\Gamma \vdash P_1[q/x] : n_1')$ with $n_1' \leq n_1$. As, $\Gamma(q) = Ch^k(T_V')$, we construct

$$\mathcal{D}' = (\mathbf{OutN}_{\mathrm{Pa}})\frac{\mathcal{D}^{(1)}}{\Gamma \vdash \overline{q}\langle q'\rangle.P_1[q/x] : \max(k, n_1')}.$$

  We conclude by stating that $\max(k, n_1') \leq \max(k, n_1)$, and using Definition 5.6.
- Case ($\mathbf{Rep}_{\mathrm{Pa}}$) can be deduced from cases($\mathbf{InN}_{\mathrm{Pa}}$) and ($\mathbf{Loc}_{\mathrm{Pa}}$). □

As announced, we then prove a similar property, about substitution of processes instead of substitutions of names.

Lemma 5.9. *Suppose $\mathcal{D} : (\Gamma, X : m \vdash P : n)$ and $\mathcal{D}^Q : (\Gamma \vdash Q : m')$ with $m' \leq m$. Then there exists $\mathcal{D}'$, c s.t. $\mathcal{D}' : (\Gamma \vdash P[Q/X] : n')$ for some $n' \leq n$ and $\mathcal{M}_{\mathcal{D}'}(P[q/x]) = \mathcal{M}_{\mathcal{D}}(P) + c.\mathcal{M}_{\mathcal{D}^Q}(Q)$.*

Proof. By induction on the typing derivation:

- Cases ($\mathbf{Var}_{\mathrm{Pa}}$) when $P = Y \neq X$, ($\mathbf{Nil}_{\mathrm{Pa}}$), ($\mathbf{Res}_{\mathrm{Pa}}$), ($\mathbf{Par}_{\mathrm{Pa}}$), ($\mathbf{Pas}_{\mathrm{Pa}}$), ($\mathbf{InP}_{\mathrm{Pa}}$), ($\mathbf{OutN}_{\mathrm{Pa}}$) and ($\mathbf{InN}_{\mathrm{Pa}}$) and are easily treated using the induction hypotheses (where relevant), as well as Definition 5.6.
- Case ($\mathbf{Rep}_{\mathrm{T}}$) is treated using the induction hypothesis and Definition 5.6 as we impose the condition $n' \leq n$ in the statement of the lemma.
- Case ($\mathbf{Var}_{\mathrm{Pa}}$). Suppose $P = X$. Derivation $\mathcal{D}$ is built using rule ($\mathbf{Var}_{\mathrm{Pa}}$). We set $\mathcal{D}' = \mathcal{D}^Q$ and we have $\mathcal{D}' : (\Gamma \vdash Q : m')$ with $m' \leq m$. We conclude using Definition 5.6, with $c = 1$.
- Case ($\mathbf{Loc}_{\mathrm{T}}$). Suppose $P = l(\!|Q_1|\!)$. We have $\Gamma(l) = \mathbf{loc}^n$ and $\mathcal{D}$ is obtained using rule ($\mathbf{Loc}_{\mathrm{Pa}}$) from premise $\mathcal{D}^1 : (\Gamma, X : m \vdash Q_1 : n_1)$, for some $n_1 < k$. The induction hypothesis gives $\mathcal{D}^{(1)}, c_1$ s.t. $\mathcal{D}^{(1)} : (\Gamma \vdash Q_1[Q/X] : n_1')$ for some $n_1' \leq n_1$ and $\mathcal{M}_{\mathcal{D}^{(1)}}(Q_1[Q/X]) = \mathcal{M}_{\mathcal{D}^1}(Q_1) \uplus c_1.\mathcal{M}_{\mathcal{D}^Q}(Q)$. As $k > n_1 \geq n_1'$, we can construct

$$\mathcal{D}' = (\mathbf{Loc}_{\mathrm{Pa}})\frac{\mathcal{D}^{(1)}}{\Gamma \vdash l(\!|Q_1[Q/X]|\!) : n}.$$

  As $\mathcal{M}_{\mathcal{D}}(X) = \emptyset$, we conclude using Definition 5.6, with $c = c_1$.
- Case ($\mathbf{OutP}_{\mathrm{Pa}}$). Suppose $P = \overline{p}\langle Q_2\rangle.P_1$. There exists k s.t. $\Gamma(p) = Ch^k(\diamond)$ and $\mathcal{D}$ is obtained using rule ($\mathbf{OutP}_{\mathrm{Pa}}$) from premises $\mathcal{D}^2 : (\Gamma, X : m \vdash Q_2 : n_2)$ and $\mathcal{D}^1 : (\Gamma, X : m \vdash P_1 : n_1)$ with $n_2 < k$ and $n = \max(k, n_1)$ for some n_1, n_2. By the induction hypothesis, we deduce the existence of $\mathcal{D}^{(2)}, \mathcal{D}^{(1)}, c_1, c_2$ s.t. $\mathcal{D}^{(2)} : (\Gamma \vdash Q_2[Q/X] : n_2')$ and $\mathcal{D}^{(1)} : (\Gamma \vdash P_1[Q/X] : n_1')$ for some n_1', n_2' s.t. $n_2' \leq n_2$ and $n_1' \leq n_1$, $\mathcal{M}_{\mathcal{D}^{(1)}}(P_1[Q/X]) = \mathcal{M}_{\mathcal{D}^1}(P_1) \uplus c_1.\mathcal{M}_{\mathcal{D}^Q}(Q)$ and $\mathcal{M}_{\mathcal{D}^{(2)}}(Q_2[Q/X]) = \mathcal{M}_{\mathcal{D}^2}(Q_2) \uplus c_2.\mathcal{M}_{\mathcal{D}^Q}(Q)$. As $k > n_2 \geq n_2'$, we can construct

$$\mathcal{D}' = (\mathbf{OutP}_{\mathrm{Pa}})\frac{\mathcal{D}^{(1)} \qquad \mathcal{D}^{(2)}}{\Gamma \vdash \overline{p}\langle Q_2[Q/X]\rangle.P_1[Q/X] : \max(k, n_1')}.$$

  We conclude using Definition 5.6, with $c = c_1$. □

We now establish an upper bound property about $\mathcal{M}_{\mathcal{D}}(P)$.

Lemma 5.10. *If $\mathcal{D} : (\Gamma \vdash P : n)$ then $\mathcal{M}_{\mathcal{D}}(P) <_{mul} \{n+1\}$.*

Proof. By induction on the typing derivation:

- Cases ($\mathbf{Nil}_{\mathrm{T}}$), ($\mathbf{Var}_{\mathrm{Pa}}$), ($\mathbf{Res}_{\mathrm{Pa}}$), ($\mathbf{Par}_{\mathrm{Pa}}$), ($\mathbf{Pas}_{\mathrm{Pa}}$), ($\mathbf{Rep}_{\mathrm{Pa}}$), ($\mathbf{InN}_{\mathrm{Pa}}$) and ($\mathbf{InP}_{\mathrm{Pa}}$) are easily treated, using the induction hypotheses when needed and Definition 5.6.
- Case ($\mathbf{Loc}_{\mathrm{Pa}}$). Suppose $P = l(\!|Q_1|\!)$. Derivation $\mathcal{D}$ is obtained using rule ($\mathbf{Loc}_{\mathrm{Pa}}$) from premises $\Gamma(l) = \mathbf{loc}^n$ and $\mathcal{D}^1 : (\Gamma \vdash Q_1 : n_1)$ for some $n_1 < n$. The induction hypothesis gives $\mathcal{M}_{\mathcal{D}^1}(Q_1) <_{mul} n_1 + 1$. By definition, $\mathcal{M}_{\mathcal{D}}(P) = \mathcal{M}_{\mathcal{D}^1}(Q_1) \uplus \{n\}$. We have $\{n+1\} >_{mul} \mathcal{M}_{\mathcal{D}}(P)$ as $\mathcal{M}_{\mathcal{D}'}(Q_1) <_{mul} \{n_1+1\} <_{mul} \{n+1\}$ and $\{n\} <_{mul} \{n+1\}$.
- Case ($\mathbf{OutP}_{\mathrm{Pa}}$). Suppose $P = \overline{p}\langle Q_2\rangle.P_1$. There exists k s.t. $\Gamma(p) = Ch^k(\diamond)$ and $\mathcal{D}$ is built by applying rule ($\mathbf{OutP}_{\mathrm{Pa}}$) with premises $\mathcal{D}^2 : (\Gamma \vdash Q_2 : n_2)$ and $\mathcal{D}^1 : (\Gamma \vdash P_1 : n_1)$, for some n_1, n_2 s.t. $n_2 < k$, and $n = \max(k, n_1)$. By definition, $\mathcal{M}_{\mathcal{D}}(P) = \mathcal{M}_{\mathcal{D}^1}(P_1) \uplus \{k\}$. The induction hypothesis gives $\{n_1+1\} >_{mul} \mathcal{M}_{\mathcal{D}^1}(P_1)$. Thus, as we have $\{\max(k, n_1)+1\} >_{mul} \{k\}$, we deduce $\{\max(k, n_1)+1\} >_{mul} \mathcal{M}_{\mathcal{D}}(P)$.
- Case ($\mathbf{OutN}_{\mathrm{Pa}}$). Suppose $P = \overline{p}\langle q\rangle.P_1$. There exists k, T_V s.t. $\Gamma(p) = Ch^k(T_V)$ and $\mathcal{D}$ is obtained using rule ($\mathbf{OutN}_{\mathrm{Pa}}$) from premise $\mathcal{D}^1 : (\Gamma \vdash P_1 : n_1)$ for some n_1 s.t. $n = \max(k, n_1)$. By definition, $\mathcal{M}_{\mathcal{D}}(P) = \mathcal{M}_{\mathcal{D}^1}(P_1) \uplus \{k\}$. The induction hypothesis gives $\{n_1+1\} >_{mul} \mathcal{M}_{\mathcal{D}^1}(P_1)$. Thus, as we have $\{\max(k, n_1)+1\} >_{mul} \{k\}$, we get $\{\max(k, n_1)+1\} >_{mul} \mathcal{M}_{\mathcal{D}}(P)$. □

Finally, we establish the main property of our type system, that relates typability and reduction.

Lemma 5.11. *If $\mathcal{D} : (\Gamma \vdash P : n)$ and $P \to P'$, then there exists $\mathcal{D}'$, n' s.t. $n \geq n'$, $\mathcal{D}' : (\Gamma \vdash P' : n')$ and $\mathcal{M}_{\mathcal{D}}(P) > \mathcal{M}_{\mathcal{D}'}(P')$.*

Proof. We reason by induction on the derivation of $P \to P'$.

- Cases ($\mathbf{Cong_P}$), ($\mathbf{Scop_P}$) and ($\mathbf{Spect_P}$) are treated easily using the induction hypotheses (when relevant), Lemmas 5.5 and 5.7, Definition 5.6, the compatibility of the multiset ordering with $\uplus$, and the compatibility of $\leq$ with max.
- Case ($\mathbf{Loc_P}$). We have $P = l(\!|Q_1|\!)$, $P' = l(\!|Q_1'|\!)$ and $Q_1 \to Q_1'$. The derivation $\mathcal{D}$ is obtained, using rule ($\mathbf{Loc}_{\mathrm{Pa}}$), from premise $\mathcal{D}^1 : (\Gamma \vdash Q_1 : n_1)$ with $\Gamma(l) = \mathbf{loc}^n$, for some $n_1 < n$. We have $\mathcal{M}_{\mathcal{D}}(P) = \mathcal{M}_{\mathcal{D}^1}(Q_1) \uplus \{n\}$. The induction hypothesis gives $\mathcal{D}^{(1)} : (\Gamma \vdash Q_1' : n_1')$ with $n_1' \leq n_1$ and $\mathcal{M}_{\mathcal{D}^{(1)}}(Q_1') <_{mul} \mathcal{M}_{\mathcal{D}^1}(Q_1)$. As $n > n_1 \geq n_1'$, we can construct

$$\mathcal{D}' = (\mathbf{Loc}_{\mathrm{Pa}})\,\frac{\mathcal{D}^{(1)}}{\Gamma \vdash l(\!|Q_1'|\!) : n}$$

and $\mathcal{M}_{\mathcal{D}'}(P') = \mathcal{M}_{\mathcal{D}^{(1)}}(Q_1') \uplus \{n\}$. We use the compatibility of $<_{mul}$ with $\uplus$ to conclude $\mathcal{M}_{\mathcal{D}'}(P') <_{mul} \mathcal{M}_{\mathcal{D}}(P)$.

- Case ($\mathbf{ComP_P}$). We have in this case $P = \overline{p}\langle Q_1\rangle.P_3 \mid p(X).P_2$ and $P' = P_3 \mid P_2[Q_1/X]$. The derivation $\mathcal{D}$ is of the form

$$(\mathbf{Par}_{\mathrm{Pa}})\,\frac{(\mathbf{OutP}_{\mathrm{Pa}})\,\dfrac{\mathcal{D}^1 \qquad \mathcal{D}^3}{\Gamma \vdash \overline{p}\langle Q_1\rangle.P_3 : \max(k, n_3)} \qquad (\mathbf{InP}_{\mathrm{Pa}})\,\dfrac{\mathcal{D}^2}{\Gamma \vdash p(X).P_2 : n_2}}{\Gamma \vdash P : n}$$

for some k, n_2, n_3 s.t. $\Gamma(p) = Ch^k(\diamond)$, $\mathcal{D}^3 : (\Gamma \vdash P_1 : n_3)$, $\mathcal{D}^1 : (\Gamma \vdash Q_1 : n_1)$ for some $n_1 < k$, $\mathcal{D}^2 : (\Gamma, X : k-1 \vdash P_2 : n_2)$ with $n = \max(k, n_3, n_2)$. As $k - 1 \geq n_1$, we construct a derivation $\mathcal{D}^{(2)}$ by using Lemma 5.9 with $\mathcal{D}^2$ and $\mathcal{D}^1$, and we get $\mathcal{D}^{(2)} : (\Gamma \vdash P_2[Q_1/X] : n_2')$ for some $n_2' \leq n_2$ and $\mathcal{M}_{\mathcal{D}^{(2)}}(P_2[Q_1/X]) = \mathcal{M}_{\mathcal{D}^2}(P_2) + c.\mathcal{M}_{\mathcal{D}^1}(Q_1)$ for some c. We can construct

$$\mathcal{D}' = (\mathbf{Par}_{\mathrm{Pa}})\,\frac{\mathcal{D}^3 \qquad \mathcal{D}^{(2)}}{\Gamma \vdash P' : \max(n_3, n_2')}.$$

Clearly $\max(n_3, n_2') \leq \max(k, n_3, n_2)$. By Definition 5.6 $\mathcal{M}_{\mathcal{D}}(P) = \{k\} \uplus \mathcal{M}_{\mathcal{D}^2}(P_2) \uplus \mathcal{M}_{\mathcal{D}^3}(P_3)$ and $\mathcal{M}_{\mathcal{D}'}(P') = \mathcal{M}_{\mathcal{D}^{(2)}}(P_2[Q/X]) \uplus \mathcal{M}_{\mathcal{D}^3}(P_3)$. From Lemma 5.10, we know that $\mathcal{M}_{\mathcal{D}_1}(Q_1) <_{mul} \{n_1+1\}$. As $n_1 < k$, we get $c.\mathcal{M}_{\mathcal{D}_1}(Q_1) <_{mul} \{k\}$. This allows us to conclude $\mathcal{M}_{\mathcal{D}}(P) <_{mul} \mathcal{M}_{\mathcal{D}'}(P')$.

- The case ($\mathbf{Pass_P}$), with $P = l(\!|Q_1|\!) \mid l(X) \triangleright P_2$ and $\Gamma(l) = \mathbf{loc}^k$ for some k, is treated like case ($\mathbf{ComP_P}$) with $P = \overline{a}\langle Q_1\rangle.\mathbf{0} \mid a(X).P_2$ for some a s.t. $\Gamma(a) = Ch^k(\diamond)$.
- ($\mathbf{Trig}_{\mathrm{Pa}}$). We have $P = \overline{p}\langle q\rangle.P_1 \mid !a(x).P_2$ and $P' = P_1 \mid P_2[q/x] \mid !p(x).P_2$. The derivation $\mathcal{D}$ is of the form

$$(\mathbf{Par}_{\mathrm{Pa}})\,\frac{(\mathbf{Out}_{\mathrm{Pa}})\,\dfrac{\mathcal{D}^1}{\Gamma \vdash \overline{p}\langle q\rangle.P_1 : \max(k, n_1)} \qquad (\mathbf{Rep}_{\mathrm{Pa}})\,\dfrac{\mathcal{D}^2}{\Gamma \vdash !p(x).P_2 : 0}}{\Gamma \vdash P : n}$$

for some k, n_1 s.t. $\Gamma(p) = Ch^k(T)$, $\Gamma(q) = T$, $\mathcal{D}^1 : (\Gamma \vdash P_1 : n_1)$, $\mathcal{D}^2 : (\Gamma, x : T \vdash P_2 : n_2)$ for some $n_2 < k$ and $n = \max(k, n_1)$. Applying Lemma 5.8 to $\mathcal{D}^2$ allows us to construct $\mathcal{D}^{(2)} : (\Gamma \vdash P_2[q/x] : n_2')$ with $n_2' \leq n_2$ and $\mathcal{M}_{\mathcal{D}^{(2)}}(P_2[q/x]) = \mathcal{M}_{\mathcal{D}^2}(P_2)$. We can then construct

$$\mathcal{D}' = (\mathbf{Par}_{\mathrm{Pa}})\,\frac{(\mathbf{Par}_{\mathrm{Pa}})\,\dfrac{\mathcal{D}^1 \qquad (\mathbf{Rep}_{\mathrm{Pa}})\,\dfrac{\mathcal{D}^2}{\Gamma \vdash !p(x).P_2 : 0}}{\Gamma \vdash P_1 \mid !p(x).P_2 : n_1} \qquad \mathcal{D}^{(2)}}{\Gamma \vdash P' : \max(k, n_1, n_2)}.$$

As $k > n_2$, we get $\max(n_1, n_2') < \max(k, n_1, n_2)$. By definition $\mathcal{M}_{\mathcal{D}}(P) = \{k\} \uplus \mathcal{M}_{\mathcal{D}^1}(P_1)$ and $\mathcal{M}_{\mathcal{D}'}(P') = \mathcal{M}_{\mathcal{D}^{(2)}}(P_2[q/x]) \uplus \mathcal{M}_{\mathcal{D}^1}(P_1)$. As $\Gamma, x : T \vdash P_2 : n_2$, we can use Lemma 5.10 to deduce $\mathcal{M}_{\mathcal{D}^{(2)}}(P_2[Q_1/X]) <_{mul} \{n_2 + 1\}$. As $k \geq (n_2 + 1)$, this allows us to conclude $\mathcal{M}_{\mathcal{D}}(P) <_{mul} \mathcal{M}_{\mathcal{D}'}(P')$.

- The proof for case ($\mathbf{ComN_P}$) is deduced from the proof for case ($\mathbf{Trig_P}$). □

Theorem 5.12. If $\Gamma \vdash P : n$, then P terminates.

Proof. We suppose by contradiction that P diverges, which means that we have an infinite sequence $(P_i)_{0 \leq i}$ s.t. $P_0 = P$ and for each i, $P_i \to P_{i+1}$.

By applying Lemma 5.11 to each P_i, we obtain an infinite sequence of typing derivations $(\mathcal{D}_i)_{0 \leq i}$ such that, for each i, $\mathcal{D}^i : (\Gamma \vdash P_i : n_i)$ and $\mathcal{M}_{\mathcal{D}_{i+1}}(P_{i+1}) <_{mul} \mathcal{M}_{\mathcal{D}_i}(P_i)$. The multiset extension of the standard ordering over natural number being well-founded, we obtain a contradiction. □

6. Concluding remarks

In this paper, we have analysed termination in higher-order concurrent languages, using the higher-order π-calculus as a core formalism to build the basis of our type systems. For future work, we plan to examine how the type systems we have

presented can be adapted to existing process calculi in which processes can be exchanged in communications or can move among locations such as, e.g., Ambients [1], Homer [4] and Kells [5,11].

Another question we would like to address is type inference; for this, [2] could serve as a starting point. We believe that the technique described in the first part of [2] can be used to establish that the problem of inferring types for the type system of Section 2 is polynomial. Intuitively, checking the typability of a HOpi_2 process P boils down to checking the presence of cycles in a graph where nodes are names of P and where there is an edge between a and b if b appears in output subject position inside a message sent on a. We should be able to reason analogously to establish a polynomial bound for type inference for the type system for Paπ (Section 5). On the other hand, we show in [2] that the analysis of sequences of prefixes introduces a combinatorial blow up, and this should be the case for the type system of Section 4. As in [2], reduction to the 3SAT problem should allow us to show that the type inference problem is NP-complete in this case.

References

[1] L. Cardelli, A.D. Gordon, Mobile ambients, in: M. Nivat (Ed.), Proceeedings of the First International Conference on Foundations of Software Science and Computation Structures, FoSSaCS'98 (Lisbon, March/April 1998), Lecture Notes in Computer Science, vol. 1378, Springer, 1998, pp. 140–155.
[2] R. Demangeon, D. Hirschkoff, N. Kobayashi, D. Sangiorgi, On the complexity of termination inference for processes, in: G. Barthe, C. Fournet (Eds.), Revised Selected Papers from Third Symposium on Trustworth Global Computing, TGC 2007 (Sophia-Antipolis, November 2007), Lecture Notes in Computer Science, vol. 4912, Springer, 2008, pp. 140–155.
[3] Y. Deng, D. Sangiorgi, Ensuring termination by typability, Inform. and Comput. 204 (7) (2006) 1045–1082.
[4] T. Hildebrandt, J.C Godskesen, M. Bundgaard, Bisimulation Congruences for Homer – a Calculus of Higher Order Mobile Embedded Resources, Technical Report TR-2004-52, University of Copenhagen, 2004.
[5] D. Hirschkoff, A. Pardon, T. Hirschowitz, S. Hym, A. Pardon, D. Pous, Encapsulation and dynamic modularity in the pi-calculus, in: V.T. Vasconcelos, N. Yoshida, in: Proceedings of First Workshop on Programming Language Approaches to Concurrency and Communication-Centric Software, PLACES 2008 (Oslo, June 2008), Electronic Notes in Theoretical Computer Science, vol. 249, Elsevier, 2009, pp. 85–100.
[6] U. Nestmann, What is a "good" encoding of guarded choice?, Inform. and Comput. 156 (1–2) (2000) 287–319.
[7] R. Demangeon, D. Hirschkoff, D. Sangiorgi, Termination in higher-order concurrent calculi, in: F. Arbab, M. Sirjani (Eds.), Revised Selected Papers from Third IPM International Conference on Foundations of Software Engineering, FSEN 2009 (Kish Island, April 2009), Lecture Notes in Computer Science, Springer, vol. 5961, 2010, pp. 81–96.
[8] D. Sangiorgi, Expressing Mobility in Process algebras: First-Order and Higher-Order Paradigms, Ph.D. Thesis, University of Edinburgh, 1992.
[9] D. Sangiorgi, Termination of processes, Math. Structures Comput. Sci. 16 (1) (2006) 1–39.
[10] D. Sangiorgi, D. Walker, The π-Calculus: A Theory of Mobile Processes, Cambridge University Press, 2001.
[11] A. Schmitt, J.-B. Stefani, The Kell calculus: a family of higher-order distributed process calculi, in: R. De Nicola, D. Sangiorgi (Eds.), Revised Selected Papers from First International Symposium in Trustworthy Global Computing, TGC 2005 (Edinburgh, April 2005), Lecture Notes in Computer Science, vol. 3267, Springer, 2005, pp. 146–178.
[12] P. Terese, Term rewriting systems, in: Cambridge Tracts in Theoretical Computer Science, vol. 55, Cambridge University Press, 2003.
[13] B. Thomsen, Calculi for Higher Order Communication Systems, Ph.D. Thesis, University of London, 1996.
[14] N. Yoshida, M. Berger, K. Honda, Strong normalisation in the pi-calculus, Inform. and Comput. 191 (2) (2004) 145–202.