The corresponding of $\sim_{\text{ccs}}$ in the weak case, called *weak bisimulation*, is denoted by $\approx_{\text{ccs}}$. Both $\sim_{\text{ccs}}$ and $\approx_{\text{ccs}}$ are also congruences. For the latter, $\approx_{\text{ccs}}$ is preserved by $+$ because we only allow *guarded* sums (however Theorem 3.3.2 remains valid if this restriction on sum is dropped, since the proof only uses static contexts).

We maintain from the π-calculus the hypothesis that each process leaves an infinity of unused names. This will allow us to define, for each given pair of processes P and Q, contexts which have names which do not occur in P or Q. The definition of these contexts will employ a countable infinite number of constants. This prevents us from extending the proof to a language where constants have been replaced by replication, since, as mentioned in Section 2.1.1, replication can only encode a finite number of constants. Moreover, if the number of free names in P or Q is infinite, our contexts will also incorporate infinite sums.

Theorem 3.3.2

1. *$\sim$ and $\sim_{\text{ccs}}$ coincide;*

2. *$\approx$ and $\approx_{\text{ccs}}$ coincide.*

We only show the proof for the weak case because it is the hardest and because in the thesis we shall be working in the weak case. The proof in the strong case is substantially simpler, due to the tighter control we have on the production of silent actions from the processes (see also for this the proof of Theorem 1 in [MS92]).

The inclusion $\approx_{\text{ccs}} \subseteq \approx$ is immediate: Clearly $\approx_{\text{ccs}} \subseteq \dot{\approx}$ (every weak bisimulation is a weak barbed bisimulation); since $\approx_{\text{ccs}}$ is preserved by the static operators, $P \approx_{\text{ccs}} Q$ implies $C[P] \approx_{\text{ccs}} C[Q]$, which in turn, implies $C[P] \dot{\approx} C[Q]$, for every static context $C[\cdot]$.

The remainder of the section is devoted to the proof of the inclusion in the opposite direction. The basic idea is the same as the related proofs in [MS92]: Build a class of contexts $\mathcal{C}xt$ powerful enough to guarantee that barbed bisimulation on these contexts implies bisimulation, i.e. prove that

$$\mathcal{R} = \{(P, Q) \ : \ \text{for some } C[\cdot] \in \mathcal{C}xt \ \ C[P] \dot{\approx} C[Q]\}$$

is a weak bisimulation. The key property of the contexts in $\mathcal{C}xt$ is the following. Suppose that $P \xrightarrow{\mu} P'$ and that $P \mathcal{R} Q$ because $C[P] \stackrel{\bullet}{\approx} C[Q]$. Then we can find a context $C'[\cdot] \in \mathcal{C}xt$ s.t. $C[P] \Longrightarrow C'[P']$. Since $C[P] \stackrel{\bullet}{\approx} C[Q]$, the process $C[Q]$ must be able to match this move. But the only way it can do so is by employing a μ-derivative Q' of Q (i.e. $Q \stackrel{\mu}{\Longrightarrow} Q'$) and evolve to $C'[Q'] \stackrel{\bullet}{\approx} C'[P']$. This closes up the bisimulation.

The definition of the contexts in $\mathcal{C}xt$ depends upon the set of free names in the processes P, Q under consideration; more precisely, such contexts must have "more names" than P and Q. Before defining them, let us introduce some notation.

Let H be a set of pairs of names. We denote by H_i the projection of H on the i-th element of the pairs, for $i = 1, 2$. For instance, if $H = \{(a, a'), (b, b')\}$, then $H_1 = \{a, b\}$ and $H_2 = \{a', b'\}$. We require that $H_1 \cap H_2 = \emptyset$ and that H_2 is made of all distinct names (so for instance, in the previous example for H, it must be that $a' \neq b'$). Moreover, we assume that c, in, ou and d_n, for n positive integer, are names which do not appear in H.

We follow our convention on the use of constants of omitting from the parameters of a constant those names which are supposed not to be instantiated. We make an exception for the constant V, which is given H as parameter although H does not change in the recursive calls of V. This is for uniformity with the corresponding proof for π-calculus (Theorem 3.3.3), in which H does change. For notational simplicity, we use H, which is a set of *pairs* of names, as parameter for V, rather than the tuple of names occurring in H, as the syntax of the language would require. The definition of V depends upon the cardinality of H; therefore, it is supposed that there is a different constant for each cardinality (including the case in which the cardinality is ω).

The term $V\langle H\rangle$ is the central process of a context in $\mathcal{C}xt$, which is of the form

$$C_n^H[\cdot] \stackrel{def}{=} [\cdot] \mid V\langle H\rangle \mid Count_n$$

where

$$Count_n \stackrel{def}{=} d_n + c.Count_{n+1} \qquad n > 1$$

$$\begin{array}{rlr} V \stackrel{def}{=} (H)\Big(& \sum_{(a,a')\in H} \overline{a}.\overline{c}.\overline{c}.\left(\tau.(a' + in) + \tau.V\langle H\rangle\right) & (a1) \\ & + \ \sum_{(a,a')\in H} a.\overline{c}.\overline{c}.\left(\tau.(a' + ou) + \tau.V\langle H\rangle\right) & (a2) \\ & + \ \tau.d_0 \ + \ \tau.d_1 \quad \Big) & (a3) \end{array}$$

The formal definition of $\mathcal{R}$ is

$$\mathcal{R} = \{(P, Q) \ : \ n, H \text{ exist s.t. } \ fn(P, Q) \subseteq H_1 \text{ and } \ C_n^H[P] \stackrel{\bullet}{\approx} C_n^H[Q]\}$$

For simplicity, in the following the parameter H will be omitted, and we simply write V in place of $V\langle H\rangle$. Let us now describe the rôles played by $Count_n$ and V to guarantee that $\mathcal{R}$ is weak bisimulation. The task of $Count_n$ is to control and discipline the production of visible actions by P and Q. The process $Count_n$ acts as a counter. In $C_n[P]$ the name d_n is observable, but no name $d_{n'}$ with $n' < n$ is. Now, if P can perform a communication, there is a reduction $C_n[P] \Longrightarrow R$ in which the counter $Count_n$ has evolved to $Count_{n+2}$, and hence the observability threshold for the d_i's increases to d_{n+2}. In consequence, a reduction $C_n[Q] \Longrightarrow T$ can match the move of $C_n[P]$ only if in so doing, Q has effected exactly one communication — otherwise, different sets of names d_i would be observable in R and T.

Now V. Summand $(a1)$ in V is used to verify the bisimilarity of P, Q on inputs, whereas $(a2)$ is used for outputs (the two summands have the same structure). Names in and ou are used to signal whether the summand $(a1)$ or $(a2)$ has been selected and therefore to know the kind of action — input or output — which P and Q have produced. Further, the label of P and Q's actions can be deduced from the name a'. In the definition of V, the presence of the two consecutive outputs $\overline{c}.\overline{c}$ is not strictly necessary, for one only output would have been enough; having two of them makes the proofs below clearer.

The recursive definition of V (and $Count_n$) allows us to iterate the above analysis on the derivatives of P and Q. Finally, the summand $(a3)$ of V is needed for technical reasons which will become clear later.

As it emerges from the explanation above, the proof that $\mathcal{R}$ is a weak bisimulation is based on the check of observability of the names in $K = \{in, ou\} \cup H_2 \cup \{d_n\}_n$. Let us write then $R \Downarrow_{K'}$, for $K' \subseteq K$, to mean that K' is the maximum subset of names from K which are observable in R. We shall abbreviate $R \Longrightarrow R_1 \Downarrow_{K'}$, for some R_1, as $R \Longrightarrow\Downarrow_{K'}$. Note the difference between this notation and the notation $R \Downarrow_{d_n}$, in which the subscript is a name rather than a set: The latter means that d_n can be observed in R, but without excluding other names from K from being observable too.

We can now embark in the proof that $\mathcal{R}$ is a weak bisimulation. We have to show that whenever $P \stackrel{\mu}{\longrightarrow} P'$, some Q' exists s.t.

$$Q \stackrel{\mu}{\Longrightarrow} Q' \quad \text{and} \quad P' \, \mathcal{R} \, Q'. \tag{3.1}$$

We consider only the case when μ is an input, say $\mu = a$. The cases when μ is an output can be worked out in a quite similar way; and the case when μ is a silent action is simpler. From the summand $(a1)$ of V, if V_1 is its derivative $\tau.(a' + in) + \tau.V$, we get

$$\begin{array}{rclcl} C_n[P] & \longrightarrow\!\longrightarrow & P' \mid \overline{c}.V_1 \mid Count_{n+1} & \stackrel{def}{=} & R_1 \\ & \longrightarrow & P' \mid V_1 \mid Count_{n+2} & \stackrel{def}{=} & R_2 \\ & \longrightarrow & P' \mid V \mid Count_{n+2} & \stackrel{def}{=} & R_3 \end{array}$$

Let us analyse how $C_n[Q]$ can match each of these steps. We need processes T_1, T_2, T_3, s.t.

$$\begin{array}{rcl} C_n[Q] & \Longrightarrow & T_1 \stackrel{\bullet}{\approx} R_1 \\ T_1 & \Longrightarrow & T_2 \stackrel{\bullet}{\approx} R_2 \\ T_2 & \Longrightarrow & T_3 \stackrel{\bullet}{\approx} R_3 \end{array}$$

We shall show that T_i's structure, $i = 1, 2, 3$, has strictly to mirror R_i's.

<u>First step</u>. ($C_n[P] \longrightarrow\!\longrightarrow R_1$.)
Process $C_n[Q]$ has to perform some move, because $C_n[P] \Downarrow_{d_n}$, but $R_1 \not\Downarrow_{d_n}$. However, since $R_1 \Downarrow_{d_{n+1}}$, process $Count_n$ cannot participate in more than one interaction. Therefore T_1 must be of the form $Q_1 \mid \overline{c}.V_1' \mid Count_{n+1}$ for some Q_1 and V_1' s.t.

$Q \stackrel{\mu'}{\Longrightarrow} Q_1$ and $V \stackrel{\overline{\mu'}}{\longrightarrow}\stackrel{\overline{c}}{\longrightarrow} \overline{c}.V_1'$. But $R_1 \Longrightarrow\Downarrow_{\{a',in,d_{n+2}\}}$ and T_1 must be able to do the same. We deduce that $\mu' = a$ and hence also $V_1' = V_1$. The conclusion is that $T_1 = Q_1 \,|\, \overline{c}.V_1 \,|\, Count_{n+1}$, for some Q_1 s.t. $Q \stackrel{a}{\Longrightarrow} Q_1$.

Second step ($R_1 \longrightarrow R_2$.)

We reason similarly as in the first step. Since $R_2 \not\Downarrow_{d_{n+1}}$ but $R_2 \Downarrow_{d_{n+2}}$, in $T_1 \Longrightarrow T_2$ there must be exactly one interaction between $c.V_1$ and $Count_{n+1}$. Moreover, because of the actions $V_1 \longrightarrow a'+in$, and $V_1 \longrightarrow\longrightarrow d_0$, we have $R_2 \Longrightarrow\Downarrow_{\{a',in,d_{n+2}\}}$ and $R_2 \Longrightarrow\Downarrow_{\{d_0,d_{n+2}\}}$. One can easily verify that in $T_1 \Longrightarrow T_2$, this precludes $\overline{c}.V_1$, once evolved to V_1, from performing other transitions. Thus $T_2 = Q_2|V_1|Count_{n+2}$, for some Q_2 s.t. $Q_2 \Longrightarrow Q_1$.

Third step ($R_2 \longrightarrow R_3$.)

First we make some consideration on R_2 and R_3. The process $\tau.V$ is a summand of V_1 and the transition $R_2 \longrightarrow R_3$ must have been caused by $V_1 \longrightarrow V$. Therefore:

1. Because of the transition $V_1 \longrightarrow a'+in$, we have $R_2 \Longrightarrow\Downarrow_{\{a',in,d_{n+2}\}}$, whereas R_3 cannot.

2. Because of the transitions $V \longrightarrow d_0$ and $V \longrightarrow d_1$, we have $R_3 \Longrightarrow\Downarrow_{\{d_0,d_{n+2}\}}$ and $R_3 \Longrightarrow\Downarrow_{\{d_1,d_{n+2}\}}$.

Now, let us turn to $T_2 \Longrightarrow T_3$. In this reduction:

- To match (1), process V_1 has to perform *some* action.

- To match (2), process V_1 cannot interact with Q_2 or $Count_{n+2}$, nor can it evolve to some derivative of $\tau.(a'+in)$ or $\tau.d_0$ or $\tau.d_1$. (The use of $\tau.d_1$, in addition to $\tau.d_0$, prevents V_1 from evolving to the derivative d_0 of $\tau.d_0$, in the case in which the subcomponent P' of R_3 does not have any observable behaviour).

Hence V_1 can only perform $V_1 \longrightarrow V$. We conclude that $T_3 = Q_3 \,|\, V \,|\, Count_{n+2}$, for some Q_3 s.t. $Q_2 \Longrightarrow Q_3$.

Combining what we have obtained in the three steps above, we have $Q \stackrel{a}{\Longrightarrow} Q_3$. Moreover, from $C_{n+2}[P'] = R_3 \stackrel{\bullet}{\approx} T_3 = C_{n+2}[Q_3]$, we get $P' \mathcal{R} Q_3$. Hence, for $Q' \stackrel{def}{=} Q_3$ in (3.1), this concludes the analysis.

3.3.2 The π-calculus case

The main theorem to prove is

Theorem 3.3.3

1. *$\sim$ and $\sim_{\mathrm{e}}$ coincide;*
2. *$\approx$ and $\approx_{\mathrm{e}}$ coincide.*

PROOF: Again, we only look at the weak case because it is much harder. The proof is quite similar to that of Theorem 3.3.2. The main difference is in the definition of the process V, whose summands have to be made more sophisticated since the π-calculus actions are more complex than the CCS actions. The details can be found in Appendix A. $\square$

From Theorem 3.3.3 we can infer similar equalities for the corresponding congruences:

Corollary 3.3.4

1. *$\sim^{\mathrm{c}}$ and $\sim^{\mathrm{c}}_{\mathrm{e}}$ coincide;*
2. *$\approx^{\mathrm{c}}$ and $\approx^{\mathrm{c}}_{\mathrm{e}}$ coincide.* $\square$

Chapter 4

Bisimulation in HOπ

In Section 1.3 we pointed out some counterintuitive equalities on higher-order processes determined by higher-order bisimulation. The adoption of barbed bisimulation allows us to overcome these problems. In this chapter we deepen the analysis of higher-order bisimulation in HOπ. As for π-calculus, we focus on the congruence induced by (weak) barbed bisimulation on static contexts and on all contexts, respectively called barbed equivalence ($\approx$) and barbed congruence ($\approx^c$). The target is to derive direct characterisations for $\approx$ and $\approx^c$, as simple as possible, somehow repeating what we did in the previous chapter for CCS and π-calculus.

We had to face some serious obstacle. In a higher-order process calculus, the possibility of transmitting an agent not only makes it difficult to understand which conditions a bisimulation should require on higher-order actions. Even more, it makes potentially very burdensome the verification of the congruence properties of the resulting relation (above all, the congruence over parallel composition) and the construction and proof of bisimulations. However, we think that our study does show that a simple treatment of the higher-order constructs is feasible. The mathematical tools we provide (factorisation theorem, triggered and normal bisimulation) are derived through non-trivial proofs, but once we have them, the reasoning on the calculus becomes manageable.

We start from *context bisimulation*, probably the most intuitive adjustment of higher-order bisimulation for eliminating its drawbacks discussed in Section 1.3. Context bisimulation is however heavy to handle, due to the appearance of universal quantifications in its definition. But we need it as a supporting relation to derive simpler characterisations. To this end, a central rôle is played by a special kind of agents called *triggers*. They are elementary agents whose only functionality is to activate a copy of another agent, providing it with the possible arguments. Using triggers, any subagent of a given agent can be factorised out: This is the content of the *factorisation theorem*. Building on this, we introduce the subclass of *triggered agents*, in which every agent emitted in an output or "expected" in an input is a trigger. Thus higher-order communications have become homogeneous and have lost all their potential richness and variety. This greatly simplifies the reasoning over agents. In particular, on triggered agents context bisimulation coincides with *triggered bisimulation*. The advantage of the latter is that the clause for higher-order outputs just requires *identity* (modulo α-conversion) on the labels of two matching actions and the clause for higher-order inputs only contemplates the input of a fresh trigger. We then define a mapping $\mathcal{T}$ which transforms every agent into a triggered agent. By exploiting $\mathcal{T}$, first we are able to prove a simple characterisation of context bisimulation on the whole class of HOπ agents (not necessarily in triggered form), called *normal bisimulation*. This is not as simple as triggered bisimulation, but at least all universal quantifications in the definition of context bisimulation have disappeared. Secondly, we show (for the weak case) that all these bisimulations and their congruences coincide with barbed equivalence and congruence. We shall use $\mathcal{T}$ also in Chapter 5, to define the encoding from HOπ to π-calculus, and in Chapter 7, to study the effect of modifications of the sorting on the equivalence of processes.

To sum up, the equivalences we shall consider on $HO\pi$ agents and which we shall prove to represent the same relation, are barbed equivalence, context bisimulation, and normal bisimulation (and for triggered agents, also triggered bisimulation). The same relationship holds on the corresponding congruences.

Some of the proofs we shall deal with are rather involved. In a few cases, in order to make our ideas more intelligible, we have chosen to present the proof of a simplified version of the result, under the condition that the extension to the full result is entirely smooth. The most important example is the limitation to unary arities for the sorts of the HOπ agents. The only exception is the finiteness requirement on sums and on the number of constant with which an agent is defined: These simplifications, discussed in the coming section, are meaningful because they are used in a non-trivial way in some proofs.

4.1 The reduced HOπ

We present our result on the subclass of the HOπ agents which satisfy the following two restrictions:

- All arities are unary, that is:
  - Only *unary* abstractions are permitted;
  - *One* only value can be exchanged in a communication. The value can be a name or a (unary) abstraction.
- Each agent A is *finitely* describable. That is, each sum in A is finite and the set $\mathcal{D}_A$ of constants from which the definition of A depends, is finite. This in turns implies finiteness on the number of free names of A and on the number of parameters of the constants in $\mathcal{D}_A$. These assumptions allow us to have replication in place of constants since, as mentioned in Section 2.1.1, a finite number of constants which have finite parameters can be coded up with replication — and vice versa.

It is important to stress the different significance of the two simplifications: The former is purely to make our results more readable — the generalisation to the calculus with arbitrary arities does not introduce semantic complications (see also Section 4.7.3). By contrast, the latter does intend to exclude some agents, namely those defined with an infinite number of constants or with infinite summation. In particular, the possibility of adopting replication is useful because it is precisely the recursive operator which is needed in the definition of our encodings and because it facilitates the reasoning by structural induction in the proofs. We think that our results do generalise to the case of infinite constants and sums; but we shall leave the issue for future investigations (we refer to Section 2.1.1 for more discussion on the choice between constants and replication).

Thus, the subclass of agents we shall be working with contains the well-sorted agents which are described by the following grammar:

$$\begin{array}{lcl} P & :: & \sum_{i \in I} \alpha_i . P_i \quad | \quad P_1 \mid P_2 \quad | \quad \boldsymbol{\nu}\, x\, P \quad | \quad [x = y]P \quad | \quad !\, P \quad | \\ & & X\langle F \rangle \quad | \quad X\langle x \rangle \end{array}$$

where I is a finite indexing set and prefixes and abstractions are of the form

$$\begin{array}{lcl} \alpha & :: & \overline{x}\langle F \rangle \quad | \quad \overline{x}\langle y \rangle \quad | \quad x(X) \quad | \quad x(y) \\ F & :: & (X)P \quad | \quad (x)P \quad | \quad X \end{array}$$

In the remainder of the chapter, anything which has to do with HOπ should be referred to this subclass of agents. We keep however '$\stackrel{def}{=}$' as abbreviation mechanism, to assign a name to expressions to which we want to refer later. We remind the reader that given an abstraction $G = (X)P$, we use $G\langle F \rangle$ as an abbreviation for $P\{F/X\}$.

Remark 4.1.1 In this reduced HOπ perhaps replication could have been avoided. Thomsen [Tho90] has shown that in a higher-order process calculus, the effect of recursion and replication in some cases can be simulated using restriction and parallel composition (similarly to the way in which one defines the fixpoint operator Y in the λ-calculus). It remains to see whether a general result of this form is possible. Anyway, it is convenient for us to have replication explicitly; moreover its presence does not complicate our theory. $\square$

We call $(x)P$ a *first-order abstraction*, and $(X)P$ a *higher-order abstraction*. Similarly, we distinguish *first-order applications* $X\langle x \rangle$, from *higher-order applications* $X\langle F \rangle$; and *first-order prefixings/actions* in which the object part is a name, like in $\overline{a}\langle b \rangle.P$ and $Q \xrightarrow{\overline{a}\langle b \rangle} P$, from *higher-order prefixings/actions* where the object part represents an agent, like in $\overline{a}\langle F \rangle.P$ and $Q \xrightarrow{\overline{a}\langle F \rangle} P$.

The simplifications in the syntax of HOπ also induce simplifications in the language of sorts. As object sort, all we need is the sort $()$ for processes and the unary object sorts El_u, which are described by the grammar

$$El_u :: s \mid (El_u) \qquad \text{for } \ s \in \text{subject sorts}$$

We shall omit the sorting information unless necessary or important. For instance, when writing $X\langle F\rangle$, it is intended that for some S, it holds that $X : (S)$ and $F : S$, and similarly in

$$\text{"for all } F, \text{ ... } G\{F/X\}\text{..."}$$

it is intended that

$$\text{"for every } F \text{ of the same sort as } X, \text{ ... } G\{F/X\}\text{..."}$$

Because of the clear separation between first-order and higher-order actions that we have in this reduced HOπ, when defining bisimulations both of them must be considered. Since the clauses for first-order actions will be always the same, it is convenient to factorise them out. They are the ordinary clauses of early bisimulation for π-calculus:

Definition 4.1.2 (first-order simulation) *A relation $\mathcal{R}$ on HOπ processes is a* strong first-order simulation *if $P \mathrel{\mathcal{R}} Q$ implies that whenever $P \xrightarrow{\mu} P'$ and μ is a first-order action with $bn(\mu) \cap fn(P,Q) = \emptyset$, then $Q \xrightarrow{\mu} Q'$ and $P' \mathrel{\mathcal{R}} Q'$.* □

Its corresponding in the weak case is called *weak first-order simulation.* The following is a useful lemma to reason on the input actions of HOπ processes. We state it only for higher-order actions because this is all we shall need but in fact the result holds for first-order actions too.

Lemma 4.1.3 *Suppose $P \xrightarrow{a\langle F\rangle} P'$; then there exists G s.t. $P' = G\langle F\rangle$ and for every E, it holds that $P \xrightarrow{a\langle E\rangle} G\langle E\rangle$.*

PROOF: Induction on the length of the inference of $P \xrightarrow{a\langle F\rangle} P'$. □

For instance, if $P = a(X).(\overline{b} \mid X\langle c\rangle)$, we have $P \xrightarrow{a\langle F\rangle} \overline{b} \mid F\langle c\rangle$ and then the G of the lemma is $(X)\big(\overline{b} \mid X\langle c\rangle\big)$.

4.2 Strong context bisimulation

The non-naturalness problems of higher-order bisimulation, examined in Section 1.3, were due to the fact that in a higher-order output the object part and the continuation are examined separately, thus preventing a satisfactory treatment of the channels private to the two. This resembles the problems of *distributed bisimulation* [CH89] in presence of restriction. We might then try to follow the solution for the latter proposed by Boudol et al. in [BCHK91] and based on the introduction of *locations*. The idea is to keep the two components to analyse together but assign them different locations, which can be detected when an observable action is produced. Thus the observable actions of the two components can be distinguished and yet, there can still be private names and communications between them. The inconvenient of this approach is that it forces an extension of the syntax of the language. Instead, we have chosen to avoid the separation between object part and continuation by explicitly taking into account the *context* in which the emitted agent is supposed to be used. We have called the resulting bisimulation *context bisimulation*. We shall see later that its weak version and the corresponding congruence coincide with barbed equivalence and congruence.

4.2.1 Definition and preliminaries

Definition 4.2.1 (context bisimulation) *A relation $\mathcal{R}$ over HOπ processes is a* strong context simulation *if $\mathcal{R}$ is a strong first-order simulation and $P \ \mathcal{R}\ Q$ implies, for every F,*

1. *whenever $P \xrightarrow{a\langle F\rangle} P'$, there exists Q' s.t. $Q \xrightarrow{a\langle F\rangle} Q'$ and $P' \ \mathcal{R}\ Q'$,*
2. *whenever $P \xrightarrow{(\boldsymbol{\nu}\,\widetilde{b})\overline{a}\langle F\rangle} P'$ there exist $Q', E, \widetilde{c}$, s.t. $Q \xrightarrow{(\boldsymbol{\nu}\,\widetilde{c})\overline{a}\langle E\rangle} Q'$ and for all G with $fn(G) \cap (\widetilde{b} \cup \widetilde{c}) = \emptyset$,*
$$\boldsymbol{\nu}\,\widetilde{b}(G\langle F\rangle \mid P') \ \ \mathcal{R} \ \ \boldsymbol{\nu}\,\widetilde{c}(G\langle E\rangle \mid Q').$$

A relation $\mathcal{R}$ is a strong context bisimulation*, briefly $\simeq_{\mathrm{Ct}}$-bisimulation, if $\mathcal{R}$ and $\mathcal{R}^{-1}$ are strong context simulations. We say that P, Q are* strong context bisimilar*, briefly $P \simeq_{\mathrm{Ct}} Q$, if $P \ \mathcal{R}\ Q$, for some $\simeq_{\mathrm{Ct}}$-bisimulation $\mathcal{R}$.* $\square$

Remark 4.2.2 There is an appealing formulation for context bisimulation following Milner's idea of *concretion* in [Mil91]. A higher-order concretion would be an expression of the form $\boldsymbol{\nu}\,\widetilde{b}\,\langle F, Q\rangle$, using which, a transition $P \xrightarrow{(\boldsymbol{\nu}\,\widetilde{b}\,)\overline{a}\langle F\rangle}$ can be rewritten as $P \xrightarrow{\overline{a}} \boldsymbol{\nu}\,\widetilde{b}\,\langle F, Q\rangle$. Also Milner's definition of pseudoapplication between concretion and abstraction (of the same arity) can be extended by setting

$$G \bullet \boldsymbol{\nu}\,\widetilde{b}\,\langle F, Q\rangle \stackrel{def}{=} \boldsymbol{\nu}\,\widetilde{b}\,\big(G\langle F\rangle \mid Q\big)$$

(with possible renaming of names in $\boldsymbol{\nu}\,\widetilde{b}$ to avoid the capture of names in G). Now, using C, D to denote concretions, clause (2) in the definition of context bisimulation can be rephrased as follows:

> whenever $P \xrightarrow{\overline{a}} C$ there exist D, s.t. $Q \xrightarrow{\overline{a}} D$ and for all G, it holds that $G \bullet C \;\mathcal{R}\; G \bullet D$.

As mentioned in Section 2.1.1, we chose to avoid concretions in order not to burden the proofs in the calculus. $\square$

It is easy to convince ourselves that $\simeq_{\mathrm{Ct}}$ is an equivalence relation and that it includes the structural congruence relation; that is

Proposition 4.2.3 *It holds that:*

1. *$\simeq_{\mathrm{Ct}}$ is an equivalence relation;*
2. *$P \equiv Q$ implies $P \simeq_{\mathrm{Ct}} Q$.* $\square$

The following proposition shows the soundness of some bisimulation up-to techniques discussed in Section 2.5.3, which will be useful when reasoning with context bisimulation.

Proposition 4.2.4 *Suppose $\mathcal{R}$ is one of the following:*

- *a $\simeq_{\mathrm{Ct}}$-bisimulation up-to restriction,*
- *a $\simeq_{\mathrm{Ct}}$-bisimulation up-to $\simeq_{\mathrm{Ct}}$,*

- *a $\simeq_{\mathrm{Ct}}$-bisimulation up-to $\simeq_{\mathrm{Ct}}$ and restriction.*

Then $\mathcal{R} \subseteq \simeq_{\mathrm{Ct}}$.

PROOF: Standard proof-techniques for bisimulations up-to ([Mil89,MPW92]). The higher-order setting in which we are working does not introduce any additional complication. □

The relation $\simeq_{\mathrm{Ct}}$ is generalised to abstractions and open agents in the obvious way. In the next definition, K stands for a name if F_1 and F_2 are first-order abstractions, for an agent otherwise.

Definition 4.2.5

1. ***($\simeq_{\mathrm{Ct}}$ on abstractions)*** *Let F_1 and F_2 in $HO\pi$. We set $F_1 \simeq_{\mathrm{Ct}} F_2$ when for all K it holds that $F_1\langle K\rangle \simeq_{\mathrm{Ct}} F_2\langle K\rangle$.*

2. ***($\simeq_{\mathrm{Ct}}$ on open agents)*** *Let A_1 and A_2 in $HO\pi^o$, with $fv(A_1, A_2) \subseteq \{\widetilde{X}\}$; we set $A_1 \simeq_{\mathrm{Ct}} A_2$ if for all $\widetilde{F} : \widetilde{X}$, it holds that $A_1\{\widetilde{F}/\widetilde{X}\} \simeq_{\mathrm{Ct}} A_2\{\widetilde{F}/\widetilde{X}\}$.* □

4.2.2 Congruence properties of context bisimulation

It is common in calculi which can describe mobile processes that the strong bisimilarity relations are preserved by all operators but first-order input prefixing. The same is true for $\simeq_{\mathrm{Ct}}$ (use the same counterexample 2.6.2 as for π-calculus's early bisimulation).

The proof of congruence over the remaining operators is presented in Theorem 4.2.7. The most difficult results are, by far, congruence for object constructor and for application on the right (Theorem 4.2.7(6-7)). They are exclusive of the higher-order setting to which HOπ belongs. To derive them, we prove congruence for $\simeq_{\mathrm{Ct}}$ over higher-order substitution. This is stated in Proposition 4.2.6; we refer to appendix B for the proof, which is rather delicate.

Proposition 4.2.6 *Let $F_1, F_2, A \in HO\pi^o$; then $F_1 \simeq_{\mathrm{Ct}} F_2$ implies $A\{F_1/X\} \simeq_{\mathrm{Ct}} A\{F_2/X\}$.*

PROOF: See Appendix B. □

Theorem 4.2.7 (congruence properties of $\simeq_{\mathrm{Ct}}$) *Let $P, Q, F_1, F_2, R \in HO\pi^{\mathrm{o}}$; if $P \simeq_{\mathrm{Ct}} Q$ and $F_1 \simeq_{\mathrm{Ct}} F_2$ then*

1. $\boldsymbol{\nu}\, a\, P \simeq_{\mathrm{Ct}} \boldsymbol{\nu}\, a\, Q$;
2. $P \mid R \simeq_{\mathrm{Ct}} Q \mid R$;
3. $!\, P \simeq_{\mathrm{Ct}} !\, Q$;
4. $[a = b]P \simeq_{\mathrm{Ct}} [a = b]Q$;
5. $\alpha.P + R \simeq_{\mathrm{Ct}} \alpha.Q + R$, *if α is not a first-order input;*
6. $\overline{a}\langle F_1\rangle.P + R \simeq_{\mathrm{Ct}} \overline{a}\langle F_2\rangle.P + R$;
7. $X\langle F_1\rangle \simeq_{\mathrm{Ct}} X\langle F_2\rangle$;

where, in clauses (5) and (6), the subterm $+R$ can also be missing from the left and right members.

PROOF: Clauses (2), (3), (4), (6), (7) and (5) when α is an output can be easily derived using Proposition 4.2.6. This does not hold for clauses (1), and (5) when α is a higher-order input, since a and α may bind some free occurrences of names or variables in P and Q. For these cases, one has to give the bisimulation explicitly. We omit the details, which are simple. □

From Proposition 4.2.6 we can also immediately infer the congruence of $\simeq_{\mathrm{Ct}}$ w.r.t. the metanotation $G\langle F\rangle$; that is, if $F_1 \simeq_{\mathrm{Ct}} F_2$ then

$$G\langle F_1\rangle \simeq_{\mathrm{Ct}} G\langle F_2\rangle \quad \text{and} \quad F_1\langle E\rangle \simeq_{\mathrm{Ct}} F_2\langle E\rangle$$

4.3 Strong context congruence

Let $\simeq^{\mathrm{c}}_{\mathrm{Ct}}$ be the congruence defined from $\simeq_{\mathrm{Ct}}$ in the standard way by means of first-order substitutions (see Section 2.5.2).

Theorem 4.3.1 $\simeq^{c}_{Ct}$ *is a congruence over all the operators.*

PROOF: It is straightforward to check that $\simeq^{c}_{Ct}$ is a congruence over first-order input prefixing. For the remaining operators, the result follows from the corresponding one for $\simeq_{Ct}$ (Theorem 4.2.7). $\Box$

Similarly, using Proposition 4.2.6 we can infer that $\simeq^{c}_{Ct}$ is preserved by higher-order substitutions and by the metanotation $G\langle F\rangle$. That is, on open agents, $F_1 \simeq^{c}_{Ct} F_2$ implies:

- $A\{F_1/X\} \simeq^{c}_{Ct} A\{F_2/X\}$;
- $G\langle F_1\rangle \simeq^{c}_{Ct} G\langle F_2\rangle$;
- $F_1\langle E\rangle \simeq^{c}_{Ct} F_2\langle E\rangle$.

4.3.1 Distributivity properties for replication

NOTATION. We shall write $m \star F$ as an abbreviation for

$$m(x).F\langle x\rangle \qquad\qquad \text{or} \qquad\qquad m(X).F\langle X\rangle$$

depending upon the sorts of the expressions involved, and $P\,\{m := F\}$ as abbreviation for

$$\boldsymbol{\nu}\, m\,(P \mid\, !\, m \star F)$$

For notational convenience, we define $\{m := F\}$ on abstractions as well. By η-conversion we can assume that the abstraction is of the form $(X)P$ or$(x)P$, and we define $\big((X)P\big)\,\{m := F\}$ as $(X)\big(P\,\{m := F\}\big)$, and $\big((a)P\big)\,\{m := F\}$ as $(a)\big(P\,\{m := F\}\big)$, for $a \neq m$. We let $\{m := F\}$ have the same precedence as substitution.

Remember that m is restricted, and hence *not free*, in $P\,\{m := F\}$, in the same way as x is not free in the λ-expression $M\{y/x\}$. Indeed, we chose curly brackets for the above abbreviation because, under a certain condition on the use of m in P and F, $\{m := F\}$ behaves just like a substitution in $P\,\{m := F\}$, roughly the substitution of '$F\langle E\rangle \mid Q$' (resp. '$F\langle b\rangle \mid Q$') for the prefixes '$\overline{m}\langle E\rangle.Q$' (resp. '$\overline{m}\langle b\rangle.Q$'). All results in the present and following section can be interpreted as formalisation of this fact. Intuitively, the condition on the use of m in P and F prevents m from being exported; hence, since m remains restricted, the only possible effect of $\overline{m}\langle E\rangle.Q$ is to trigger a copy of F with argument E. For example, a little thinking should convince the reader that if m is not free in E, E', P, F, then the visible behaviour of $\left(\overline{m}\langle E\rangle.\overline{m}\langle E'\rangle.P\right)\{m := F\}$ is the same as $F\langle E\rangle \mid F\langle E'\rangle \mid P$.

<u>CONVENTION</u>. To simplify the uses of the abbreviation $\{m := F\}$, it is convenient to allow a free use of expressions of the form

$$\left(\overline{a}\langle Tr_m\rangle.P\right)\{m := F\} \qquad (*)$$

Formally, since only guarded sums are admitted in the language, it is not legal to use such an expression in a context like $[\cdot] + Q$. Instead, we should write

$$\left(\overline{a}\langle Tr_m\rangle.P + Q\right)\{m := F\} \qquad (**)$$

provided m is not free in Q. However there are simple and obvious transformations which convert any misuse of $(*)$ into a process of the form $(**)$ and vice versa, which isolate from $(**)$ a subexpression of the form $(*)$. Thus we leave these transformations implicit and we work with $(*)$. □

We first show with Theorem 4.3.3 that $\{m := F\}$ distributes over all operators. For this, Proposition 4.3.2 below is crucial to guarantee that $\{m := F\}$ distributes over higher-order substitutions. What happens here is therefore similar to what we did in Section 4.2.2 to prove the congruences of $\simeq_{\mathrm{Ct}}$. In both cases the results for higher-order substitution, namely Propositions 4.2.6 and 4.3.2, are very delicate; but because the technicalities on which they are based will not be needed later, their proofs have been placed in an appendix.

Proposition 4.3.2 *Let $A, E, F \in HO\pi^{\circ}$. Suppose $m \notin fn(F)$, and that m appears free only in negative subject position in A, E. Then*

$$A\{E/X\}\{m := F\} \simeq^{c}_{\mathrm{Ct}} A\{m := F\}\{E\{m := F\}/X\}.$$

PROOF: See Appendix C. □

Theorem 4.3.3 (distributivity of $\{m := F\}$) *Suppose that $m \notin fn(F)$ and that m appears free only in negative subject position in α, P_i, P, Q, E. Then the following results on open agents hold:*

1. $\big(\sum_{i\in I} P_i\big)\{m := F\} \simeq^{c}_{\mathrm{Ct}} \sum_{i\in I}\big(P_i\{m := F\}\big)$.
2. $\big(\boldsymbol{\nu}\, a\, P\big)\{m := F\} \simeq^{c}_{\mathrm{Ct}} \boldsymbol{\nu}\, a\,\big(P\{m := F\}\big)$, *if* $a \notin fn(F) \cup m$.
3. $\big(P \mid Q\big)\{m := F\} \simeq^{c}_{\mathrm{Ct}} P\{m := F\} \mid Q\{m := F\}$.
4. $(!\,P)\{m := F\} \simeq^{c}_{\mathrm{Ct}} !\,(P\{m := F\})$.
5. $([a = b]P)\{m := F\} \simeq^{c}_{\mathrm{Ct}} [a = b](P\{m := F\})$, *if* $a, b \neq m$.
6. $X\langle E\rangle\{m := F\} \simeq^{c}_{\mathrm{Ct}} X\langle E\{m := F\}\rangle$.
7. *Suppose α does not bind m or free names or variables of F. Then*
$$\big(\alpha.P\big)\{m := F\} \simeq^{c}_{\mathrm{Ct}} \begin{cases} \big(\alpha.(P\{m := F\})\big)\{m := F\} & \text{if } m \text{ is free in } \alpha \\ \alpha.(P\{m := F\}) & \text{if } m \text{ is not free in } \alpha \end{cases}$$
8. $(\overline{a}\langle E\rangle.P)\{m := F\} \simeq^{c}_{\mathrm{Ct}} \big(\overline{a}\langle E\{m := F\}\rangle.P\big)\{m := F\}$.

Proof: Each case either can be easily derived using Proposition 4.3.2, or is straightforward on its own. $\square$

4.4 Weak context bisimulation and congruence

The corresponding of $\simeq_{\mathrm{Ct}}$ and $\simeq^{c}_{\mathrm{Ct}}$ in the weak case are called *weak context bisimulation* and *weak context congruence*. They are denoted by $\approxeq_{\mathrm{Ct}}$ and $\approxeq^{c}_{\mathrm{Ct}}$, respectively. All congruence results shown for $\simeq_{\mathrm{Ct}}$ and $\simeq^{c}_{\mathrm{Ct}}$ can be extended to $\approxeq_{\mathrm{Ct}}$ and $\approxeq^{c}_{\mathrm{Ct}}$, with a completely analogous proof. In particular, we have:

Theorem 4.4.1

1. $\approxeq_{\mathrm{Ct}}$ *is a congruence over all operators but first-order prefixing.*
2. $\approxeq^{c}_{\mathrm{Ct}}$ *is a congruence over all operators.* $\square$

A simple and useful lemma is the following:

Lemma 4.4.2 *For each $P \in HO\pi^{\circ}$, it holds that $\tau.P \approxeq^{c}_{\mathrm{Ct}} P$.* □

Notice that if the language for HOπ did not require only *guarded* sums, then it would not be true that $\approxeq_{\mathrm{Ct}}$ is a congruence over the sum operator; also, in Lemma 4.4.2, the congruence $\approxeq^{c}_{\mathrm{Ct}}$ should then be replaced by the bisimulation $\approxeq_{\mathrm{Ct}}$. In fact, in languages where sum can be used freely, the congruence between $\tau.P$ and P breaks down in contexts of the form $[\cdot] + Q$.

4.4.1 Triggers

A *trigger* is an agent whose only functionality is to activate (to trigger) a copy of another agent, providing it with its arguments. In the reduced HOπ we are dealing with, a trigger has only the form (depending upon the sort)

$$(x)\overline{m}\langle x\rangle.\mathbf{0}, \qquad \text{or} \qquad (X)\overline{m}\langle X\rangle.\mathbf{0}.$$

We write Tr_m to represent a trigger whose free name is m. We show now that using triggers, any subagent can be factorised out from a given agent. This is extremely useful for reasoning with higher-order processes. For instance, when comparing the behaviour of two processes it allows us to "isolate" subcomponents which might cause differences, so that the analysis can be concentrated on them.

Example 4.4.3 Consider the process

$$P \stackrel{def}{=} \big(\overline{a}\langle b\rangle.F\langle b\rangle + \alpha.R\big) \mid F\langle b'\rangle$$

We can factorise F out using triggers, thus obtaining

$$\begin{aligned} Q \ &\stackrel{def}{=}\ \Big(\big(\overline{a}\langle b\rangle.Tr_m\langle b\rangle + \alpha.R\big) \mid Tr_m\langle b'\rangle\Big)\,\{m := F\} \\ &=\ \Big(\big(\overline{a}\langle b\rangle.\overline{m}\langle b\rangle. + \alpha.R\big) \mid \overline{m}\langle b'\rangle\Big)\,\{m := F\} \end{aligned}$$

Theorem 4.4.7 will ensure that $P \approxeq^{c}_{\mathrm{Ct}} Q$. The difference in Q is that F is not activated immediately upon availability of its argument, which instead, is received via m. We can think of m as acting as a *pointer* which adds a level of indirection in the path leading to F. □

For the proof of the factorisation theorem we need some auxiliary lemmas.

Lemma 4.4.4 *In HOπ the following equalities hold:*

1. $\boldsymbol{\nu}\, a\,(\overline{a}\langle b\rangle.P \mid a(x).Q) \simeq^{c}_{\mathrm{Ct}} \tau.\boldsymbol{\nu}\, a\,(P \mid Q\{b/x\})$ *and, similarly,*
   $\boldsymbol{\nu}\, a\,(\overline{a}\langle E\rangle.P \mid a(X).Q) \simeq^{c}_{\mathrm{Ct}} \tau.\boldsymbol{\nu}\, a\,(P \mid Q\{E/X\})$.
2. $P\,\{m := F\} \simeq^{c}_{\mathrm{Ct}} P$ *if* $m \notin fn(P)$. □

Lemma 4.4.5 *Let* $F, E \in HO\pi^{\mathrm{o}}$ *and* $m \notin fn(F, E)$. *Then* $(\overline{m}\langle b\rangle.\mathbf{0})\,\{m := F\} \simeq^{c}_{\mathrm{Ct}} \tau.F\langle b\rangle$ *and* $(\overline{m}\langle E\rangle.\mathbf{0})\,\{m := F\} \simeq^{c}_{\mathrm{Ct}} \tau.F\langle E\rangle$

PROOF: Immediate using Lemma 4.4.4. □

In the version of the factorisation theorem which we present, only abstractions can be factorised out; it is however straightforward to generalise it to any agents. We derive the factorisation theorem from Lemma 4.4.6 which shows us precisely that the effect of using a trigger is to add a τ-action on the head of the replaced expression. We use $\tau \star F$ as abbreviation for $(a)\tau.F\langle a\rangle$ or $(E)\tau.F\langle E\rangle$, depending on whether F is a first-order or a higher-order abstraction.

Lemma 4.4.6 *For every* $A, F \in HO\pi^{\mathrm{o}}$ *with* $m \notin fn(A, F)$, *it holds that*

$$A\{\tau \star F/X\} \simeq^{c}_{\mathrm{Ct}} A\{Tr_m/X\}\,\{m := F\}$$

PROOF: By induction on the structure of A. The basic case is when $A = Y$. If $Y = X$ use Lemma 4.4.5; otherwise it is immediate. Now the inductive cases.

Case (a) $A = P_1 \mid P_2$

We have:

$$\begin{array}{rcl} (P_1 \mid P_2)\{\tau \star F/X\} & = & \\ P_1\{\tau \star F/X\} \mid P_2\{\tau \star F/X\} & \simeq^{c}_{\mathrm{Ct}} & \text{(induction twice)} \\ P_1\{Tr_m/X\}\,\{m := F\} \mid P_2\{Tr_m/X\}\,\{m := F\} & \simeq^{c}_{\mathrm{Ct}} & \text{(Theorem 4.3.3(3))} \\ \big(P_1\{Tr_m/X\} \mid P_2\{Tr_m/X\}\big)\,\{m := F\} & = & \\ (P_1 \mid P_2)\{Tr_m/X\}\,\{m := F\} & & \end{array}$$

Case (b) $A = \sum_{i\in I} P_i$, $A = \boldsymbol{\nu}\, a\, P$, $A = [a = b]P$, or $A = !\,P$

Similar to above.

Case (c) $A = (a)P_1$ or $A = (Y)P_1$

Easy.

Case (d) $A = \overline{a}\langle E\rangle.P$

We have:

$$\begin{array}{rcl}
(\overline{a}\langle E\rangle.P)\{\tau\star F/X\} & = & \\
\overline{a}\Big\langle E\{\tau\star F/X\}\Big\rangle.\Big(P\{\tau\star F/X\}\Big) & \simeq^{\mathrm{c}}_{\mathrm{Ct}} & \text{(induction twice)} \\
\overline{a}\Big\langle E\{Tr_m/X\}\ \{m := F\}\Big\rangle.\Big(P\{Tr_m/X\}\ \{m := F\}\Big) & \simeq^{\mathrm{c}}_{\mathrm{Ct}} & \text{(Theorem 4.3.3(7))} \\
\Big(\overline{a}\Big\langle E\{Tr_m/X\}\ \{m := F\}\Big\rangle.P\{Tr_m/X\}\Big)\ \{m := F\} & \simeq^{\mathrm{c}}_{\mathrm{Ct}} & \text{(Theorem 4.3.3(8))} \\
\Big(\overline{a}\Big\langle E\{Tr_m/X\}\Big\rangle.P\{Tr_m/X\}\Big)\ \{m := F\} & = & \\
(\overline{a}\langle E\rangle.P)\{Tr_m/X\}\ \{m := F\} & &
\end{array}$$

Case (e) $A = \alpha.P$ and α is not a higher-order output.

Use the induction hypothesis and Theorem 4.3.3(7).

Case (f) $A = Y\langle E\rangle$

Let us suppose $X = Y$ (the case $X \neq Y$ is simpler and requires Theorem 4.3.3(6)). We have:

$$\begin{array}{rcl}
\Big(X\langle E\rangle\Big)\{\tau\star F/X\} & = & \\
X\{\tau\star F/X\}\Big\langle E\{\tau\star F/X\}\Big\rangle & \simeq^{\mathrm{c}}_{\mathrm{Ct}} & \text{(induction twice)} \\
\Big(X\{Tr_m/X\}\ \{m := F\}\Big)\Big\langle E\{Tr_m/X\}\ \{m := F\}\Big\rangle & = & \\
\Big(Tr_m\ \{m := F\}\Big)\Big\langle E\{Tr_m/X\}\ \{m := F\}\Big\rangle & = & \\
\Big(\overline{m}\Big\langle E\{Tr_m/X\}\ \{m := F\}\Big\rangle.\mathbf{0}\Big)\ \{m := F\} & \simeq^{\mathrm{c}}_{\mathrm{Ct}} & \text{Theorem 4.3.3(8)} \\
\Big(\overline{m}\Big\langle E\{Tr_m/X\}\Big\rangle.\mathbf{0}\Big)\ \{m := F\} & = & \\
Tr_m\langle E\{Tr_m/X\}\rangle\ \{m := F\} & = & \\
X\langle E\rangle\{Tr_m/X\}\ \{m := F\} & & \square
\end{array}$$

Theorem 4.4.7 (factorisation theorem) *For every $A, F \in HO\pi^{\mathrm{o}}$ with $m \notin fn(A, F)$, it holds that*

$$A\{F/X\} \approxeq^{\mathrm{c}}_{\mathrm{Ct}} A\{Tr_m/X\}\ \{m := F\}.$$

PROOF: From Lemma 4.4.6, $A\{Tr_m/X\}\,\{m := F\} \simeq^{c}_{\mathrm{Ct}} A\{\tau \star F/X\}$. From Lemma 4.4.2, we infer $\tau \star F \approxeq^{c}_{\mathrm{Ct}} F$, hence $A\{\tau \star F/X\} \approxeq^{c}_{\mathrm{Ct}} A\{F/X\}$. Since $\simeq^{c}_{\mathrm{Ct}} \subseteq \approxeq^{c}_{\mathrm{Ct}}$, this proves the theorem. $\square$

Before leaving this section, let us present a useful corollary of the factorisation theorem, which relates higher-order input actions and "triggered" input actions (i.e. inputs of triggers). We shall need this result when proving a simpler alternative characterisation of context bisimulation, based on the use of triggers.

Corollary 4.4.8 *If $m \notin fn(P, F)$,*

1. *whenever $P \xrightarrow{a\langle F\rangle} P'$, there exists P'' s.t.*
   $P \xrightarrow{a\langle Tr_m\rangle} P''$ *and* $P' \approxeq^{c}_{\mathrm{Ct}} P''\,\{m := F\}$*;*
2. *whenever $P \xrightarrow{a\langle Tr_m\rangle} P''$ there exists P' s.t. $P \xrightarrow{a\langle F\rangle} P'$ and*
   $P' \approxeq^{c}_{\mathrm{Ct}} P''\,\{m := F\}$.

PROOF: Use the factorisation theorem and Lemma 4.1.3. $\square$

4.5 Triggered agents

We present in this section the subclass of *triggered agents* and the mapping $\mathcal{T}$ from agents to triggered agents s.t. $A \approxeq^{c}_{\mathrm{Ct}} \mathcal{T}[\![A]\!]$.

Preliminaries

In the following we suppose that Ob is a *downward-closed* sorting, that is if $s \mapsto (S) \in Ob$, then for some s', we have $s' \mapsto S \in Ob$. If Ob is not already downward-closed, then it can easily be extended to make it so.

The downward-closure property is necessary to be able to use triggers in the object part of higher-order actions. For instance, if $a : s \mapsto (S)$, then for the expression $\overline{a}\langle Tr_m\rangle.P$ to be legal, it must be $m : s' \mapsto S$, for some s'.

To define triggered agents, we first have to choose the subsorting SOb from which to select the free names of the triggers used. We introduce SOb to guarantee that if two triggers Tr_m and $Tr_{m'}$ have the same sort, then the names m, and m' belong to the same subject sort. This condition will be very useful in Chapter 5, to ensure the well-sortness of the encoding from triggered to π-calculus agents. It is also necessary for the definition of triggered bisimulation in Section 4.6, where it allows us to consider Tr_m and $Tr_{m'}$ α-convertible if m and m' are bound. We simply require that in SOb there is one and only subject sort for each object sort which appears in Ob; to put it formally, SOb must satisfy the two following conditions:

1. An object sort must appear at most once in SOb; i.e. if $s_1 \mapsto S_1 \in SOb$ and $s_2 \mapsto S_2 \in SOb$, then $S_1 \neq S_2$.
2. Each object sort which appears in Ob must be represented in SOb; i.e. if $s \mapsto S \in Ob$, then there exists s' s.t. $s' \mapsto S \in SOb$.

We denote the class of closed triggered agents by $THO\pi$ and the class of open triggered agents by $THO\pi^{\mathrm{o}}$. $THO\pi$ is parametric on SOb, since the choice of SOb in general is not unique and different choices give rise to different classes of triggered agents. We omit the reference to SOb for notational simplicity. In the following we assume that m, m' range over names with sort in SOb (i.e. SOb is defined on the sorts of m and m').

4.5.1 Definition of the class

The technique for the treatment of higher-order communications in triggered agents is the following. Instead of communicating an agent F explicitly, like in $P_1 = \overline{a}\langle F\rangle.Q$, what is transmitted is a trigger to F, that is the capability of activating an arbitrary number of copies of F, like in $P_2 = (\overline{a}\langle Tr_m\rangle.Q)\,\{m := F\}$, for $m \notin fn(Q, F)$. It is important to understand that from the point of view of an agent willing to perform an input at a, it does not make any difference whether it interacts with P_1 or P_2, although in one case the agent received is F, in the other case is Tr_m. To see this, consider what happens when the input is used with, say, argument E. In the first case the process $F\langle E\rangle$ is activated, in the second one

$\overline{m}\langle E\rangle.\mathbf{0}$; but since m is restricted, the only effect of the latter is to produce an interaction with P_2 which activates $F\langle E\rangle$. Therefore, the only difference w.r.t. the case of communication with P_1 is an additional internal interaction, which is not observable.

We could define a process P *triggered* if the following condition holds:

- If only triggers are received as inputs, then whenever a descendant of P can perform an output, this must have been caused by a subprocess of the form

$$\left(\overline{a}\langle Tr_m\rangle.P\right)\{m := F\}. \tag{†}$$

Similarly, we could say that abstractions and open agents are triggered if they are transformed into triggered processes by the instantiation of their variables with triggers.

Now, this is a *semantic* condition, and as such, in general difficult to verify. Let us see if we can translate it into a *syntactic* condition. We certainly want all higher-order outputs to be in the above triggered form; but is this enough? Consider the process

$$P \stackrel{def}{=} a(X).X\langle F\rangle$$

When P receives a trigger Tr_m at a, it becomes $\overline{m}\langle F\rangle.\mathbf{0}$, which is not of the form (†). This shows that we also need a condition on applications; they must be in triggered form too. If we adjust P to conform to this, we get

$$P' \stackrel{def}{=} a(X).\left(X\langle Tr_m\rangle\{m := F\}\right)$$

These two conditions, on outputs and applications, only fail for variables representing higher-order abstractions. Suppose X is one of such variables and let us instantiate it with the trigger $Tr_m = (Y)\overline{m}\langle Y\rangle.\mathbf{0}$; if we now use the trigger $Tr_{m'}$ as actual parameter for Y, we get $\overline{m}\langle Tr_{m'}\rangle.\mathbf{0}$, which is not of the form (†). Another way of showing the problem is by applying an η rule to X: We get $(Y)X\langle Y\rangle$, in which the application $X\langle Y\rangle$ is not triggered. This guides us also to a solution for the problem: we require that each non-binding occurrence of a variable is followed by its argument, so to make explicit all potential applications. For uniformity (and also to make smoother the mapping in Chapter 5 from triggered to first-order agents), we impose this condition also on variables representing first-order abstractions.

Summarising, an agent is triggered if each higher-order output and applications is in triggered form, and each non-binding occurrence of a variable — except those inside a trigger — is followed by its arguments. The formal definition of the class is given below. For this definition, the reader should keep in his/her mind the reduce syntax for HOπ agents, described in Section 4.1, which we are using in this chapter, and the convention on the use of the abbreviation $\{m := F\}$ in presence of sums, given in Section 4.3.1.

Definition 4.5.1 (triggered agents) *The class of* triggered agents *is the subclass of HOπ agents derived from the following grammar*

$$\begin{array}{lll} P & :: & \sum_{i\in I} N_i \quad | \quad P_1 \mid P_2 \quad | \quad \boldsymbol{\nu}\, x\, P \quad | \quad [x=y]P \quad | \quad !\,P \quad | \\ & & X\langle Tr_m\rangle \,\{m := F\} \quad | \quad X\langle x\rangle \end{array}$$

where I is a finite indexing set and N and F are of the form

$$N \;::\; (\overline{x}\langle Tr_m\rangle.P)\,\{m := F\} \quad | \quad \overline{x}\langle y\rangle.P \quad | \quad x(X).P \quad | \quad x(y).P$$

$$F \;::\; (X)P \quad | \quad (x)P \qquad \Box$$

The class $THO\pi^o$ is closed under instantiation of free variables with triggers and under the metanotation $\{m := F\}$.

Lemma 4.5.2 *If $A \in THO\pi^{\circ}$, then for every Tr_m we have $A\{Tr_m/X\} \in THO\pi^{\circ}$.*

Proof: By induction on the structure of A. We only look at the case in which $A = X\langle Tr_{m'}\rangle\,\{m' := F\}$; all the others are trivial. The trigger Tr_m must be of the form $(Y)\overline{m}\langle Y\rangle.\mathbf{0}$ and then we have

$$A\{Tr_m/X\} = Tr_m\langle Tr_{m'}\rangle\,\{m' := F\} = \left(\overline{m}\langle Tr_{m'}\rangle.\mathbf{0}\right)\{m' := F\} \in THO\pi^{\circ}. \quad \Box$$

Lemma 4.5.3 *If $P, F \in THO\pi^{\circ}$, then $P\,\{m := F\} \in THO\pi^{\circ}$.*

PROOF: Suppose F is of the form $(X)Q$, for $Q \in THO\pi^o$; the case $F = (x)P$ is analogous. Then $P\{m := F\}$ abbreviates

$$\boldsymbol{\nu}\, m\left(P \mid !\, m(Y).F\langle Y\rangle\right) = \boldsymbol{\nu}\, m\left(P \mid !\, m(Y).Q\{Y/X\}\right)$$

where by hypothesis, $P \in THO\pi^o$ and, since $Q \in THO\pi^o$, also $Q\{Y/X\} \in THO\pi^o$. Therefore, from the grammar of Definition 4.5.1, we get that the process $\boldsymbol{\nu}\, m\left(P \mid !\, m(Y).Q\{Y/X\}\right)$ is in $THO\pi^o$. □

4.5.2 The transformation to triggered agents

The transformation $\mathcal{T}$ transforms every well-sorted agent into triggered form. It is formally defined in Table 4–1. It is assumed that m is fresh, i.e. it does not occur in the source agent. The rules for higher-order application and output are the core of $\mathcal{T}$, and are motivated by the discussion in the introduction of the previous section. In the rule for variable, an η-rule is employed; elsewhere $\mathcal{T}$ acts as an homomorphism. In the rule for variable, when X is a higher-order abstraction, a new variable is introduced. The termination of $\mathcal{T}$ is guaranteed by the fact that the sort of Y is smaller than the one of X (in the sense that if the sort of X is (S), then the sort of Y is S). Note that this introduces a dependency from the sorting in the encoding.

Remark 4.5.4 Consider the rule for higher-order application $\mathcal{T}[\![X\langle F\rangle]\!]$. If $X : (S)$ and S is not among the object sorts appearing in the sorting Ob, then its translation $X\langle Tr_m\rangle\{m := \mathcal{T}[\![F]\!]\}$ does not make sense because m has to belong to a sort s with $s \mapsto S$. The problem is not serious. One could just extend the sorting to include a sort $s \mapsto S$. Alternatively, one could exclude agents using such variables from the domain of $\mathcal{T}$. In fact, these are rather uninteresting agents; in particular there is no closed process (in a closed process the only way to introduce a variable $X : (S)$ is by an input action $a(X)$; but then, for some s, it must be $a : s \mapsto ((S))$ and by the downward-closure property of Ob, some s' exists s.t. $s' \mapsto S \in Ob$). □

$$
\begin{array}{rcl}
\mathcal{T}[\![X]\!] & \stackrel{def}{=} & \begin{cases} (Y)\mathcal{T}[\![X\langle Y\rangle]\!] & \text{if } X \text{ is a higher-order abstraction} \\ (b)X\langle b\rangle & \text{otherwise} \end{cases} \\
\mathcal{T}[\![\alpha.P]\!] & \stackrel{def}{=} & \begin{cases} (\overline{a}\langle Tr_m\rangle.\mathcal{T}[\![P]\!])\,\{m := \mathcal{T}[\![F]\!]\} & \text{if } \alpha = \overline{a}\langle F\rangle \\ \alpha.\mathcal{T}[\![P]\!] & \text{otherwise} \end{cases} \\
\mathcal{T}[\![X\langle F\rangle]\!] & \stackrel{def}{=} & X\langle Tr_m\rangle\,\{m := \mathcal{T}[\![F]\!]\}
\end{array}
$$

$$
\begin{array}{rclrcl}
\mathcal{T}[\![X\langle b\rangle]\!] & \stackrel{def}{=} & X\langle b\rangle & \mathcal{T}[\![\,!\,P]\!] & \stackrel{def}{=} & !\,\mathcal{T}[\![P]\!] \\
\mathcal{T}[\![P \mid Q]\!] & \stackrel{def}{=} & \mathcal{T}[\![P]\!] \mid \mathcal{T}[\![Q]\!] & \mathcal{T}[\![\sum_{i\in I} P_i]\!] & \stackrel{def}{=} & \sum_{i\in I} \mathcal{T}[\![P_i]\!] \\
\mathcal{T}[\![\boldsymbol{\nu}\, a\, P]\!] & \stackrel{def}{=} & \boldsymbol{\nu}\, a\, \mathcal{T}[\![P]\!] & \mathcal{T}[\![[a=b]P]\!] & \stackrel{def}{=} & [a=b]\mathcal{T}[\![P]\!] \\
\mathcal{T}[\![(X)P]\!] & \stackrel{def}{=} & (X)\mathcal{T}[\![P]\!] & \mathcal{T}[\![(a)P]\!] & \stackrel{def}{=} & (a)\mathcal{T}[\![P]\!]
\end{array}
$$

Table 4–1: The transformation $\mathcal{T}$.

Example 4.5.5 *Let* $F \stackrel{def}{=} (b)\mathbf{0} : S$ *and* $X : (S)$*:*

1. *If* $P_1 = X\langle F\rangle \mid \overline{b}\langle F\rangle,$ *then*
$$\mathcal{T}[\![P_1]\!] = X\langle Tr_m\rangle\,\{m := F\} \mid \overline{b}\langle Tr_m\rangle\,\{m := F\}$$

2. *If* $P_2 = a(X).Z\langle X\rangle,$ *then*
$$\mathcal{T}[\![P_2]\!] = a(X).\Big(Z\langle Tr_m\rangle\,\Big\{m := (Y)\Big(X\langle Tr_{m'}\rangle\,\{m' := (b)Y\langle b\rangle\}\Big)\Big\}\Big) \quad \square$$

Let us also illustrate how the translation works on reductions. In the two examples we give, we use the algebraic law of Lemma 4.4.4, namely

$$P\,\{m := F\} \simeq^{c}_{\mathrm{Ct}} P \qquad \text{if } m \text{ not free in } P$$

There are two dimensions at which $\mathcal{T}$ expands the number of reductions. One is *horizontal*. If a transmitted agent F is used by its recipient n times, n interactions are required in the triggered agent to activate the copies of F.

Example 4.5.6 *Let* $P \stackrel{def}{=} \overline{a}\langle F\rangle.Q \mid a(Y).(Y\langle b\rangle \mid Y\langle c\rangle)$*. Then*

$$P \longrightarrow P' \stackrel{def}{=} Q \mid F\langle b\rangle \mid F\langle c\rangle.$$

In $\mathcal{T}[\![P]\!]$ this is simulated using two additional reductions:

$$\begin{array}{rcl}
\mathcal{T}[\![P]\!] & \stackrel{def}{=} & (\overline{a}\langle Tr_m\rangle.\mathcal{T}[\![Q]\!])\,\{m := \mathcal{T}[\![F]\!]\} \mid a(Y).(Y\langle b\rangle \mid Y\langle c\rangle) \\
& \equiv & \big(\overline{a}\langle Tr_m\rangle.\mathcal{T}[\![Q]\!] \mid a(Y).(Y\langle b\rangle \mid Y\langle c\rangle)\big)\,\{m := \mathcal{T}[\![F]\!]\} \\
& \longrightarrow & \big(\mathcal{T}[\![Q]\!] \mid Tr_m\langle b\rangle \mid Tr_m\langle c\rangle\big)\,\{m := \mathcal{T}[\![F]\!]\} \\
& = & \big(\mathcal{T}[\![Q]\!] \mid \overline{m}\langle b\rangle \mid \overline{m}\langle c\rangle\big)\,\{m := \mathcal{T}[\![F]\!]\} \\
& \longrightarrow\longrightarrow\equiv & \big(\mathcal{T}[\![Q]\!] \mid \mathcal{T}[\![F]\!]\langle b\rangle \mid \mathcal{T}[\![F]\!]\langle c\rangle\big)\,\{m := \mathcal{T}[\![F]\!]\} \\
& \simeq^{\mathrm{c}}_{\mathrm{Ct}} & \mathcal{T}[\![Q]\!] \mid \mathcal{T}[\![F]\!]\langle b\rangle \mid \mathcal{T}[\![F]\!]\langle c\rangle \;=\; \mathcal{T}[\![P']\!]
\end{array}$$

□

The other way to add interactions is *vertical* and takes its significance from the ω-order nature of HOπ. It arises with higher-order abstractions when, after the abstraction itself, one also has to trigger its arguments. This may give rise to interesting chains of activations. Below we show a simple case, in which the abstraction is of order two.

Example 4.5.7 Suppose $F \stackrel{def}{=} (b)\overline{b}\langle d\rangle : (s)$ and $G \stackrel{def}{=} (X)\big(Q \mid X\langle c\rangle\big) : ((s))$. Let $P \stackrel{def}{=} \overline{a}\langle G\rangle \mid a(Y).Y\langle F\rangle$; then

$$P \longrightarrow G\langle F\rangle = Q \mid F\langle c\rangle = Q \mid \overline{c}\langle d\rangle.$$

On the other hand, we have

$$\begin{array}{rcl}
\mathcal{T}[\![P]\!] & \stackrel{def}{=} & \overline{a}\langle Tr_{m_G}\rangle\,\{m_G := \mathcal{T}[\![G]\!]\} \mid a(Y).\big(Y\langle Tr_{m_F}\rangle\,\{m_F := \mathcal{T}[\![F]\!]\}\big) \\
& \longrightarrow\equiv & \big(\overline{m_G}\langle Tr_{m_F}\rangle\,\{m_F := \mathcal{T}[\![F]\!]\}\big)\,\{m_G := \mathcal{T}[\![G]\!]\} \\
& \longrightarrow\equiv & \big(\mathcal{T}[\![G]\!]\langle Tr_{m_F}\rangle\,\{m_F := \mathcal{T}[\![F]\!]\}\big)\,\{m_G := \mathcal{T}[\![G]\!]\} \\
& & \qquad\qquad\text{(a copy of } G \text{ is activated)} \\
& \simeq^{\mathrm{c}}_{\mathrm{Ct}} & (\mathcal{T}[\![Q]\!] \mid \overline{m_F}\langle c\rangle)\,\{m_F := \mathcal{T}[\![F]\!]\} \\
& \longrightarrow\simeq^{\mathrm{c}}_{\mathrm{Ct}} & \mathcal{T}[\![Q]\!] \mid \overline{c}\langle d\rangle = \mathcal{T}[\![Q \mid \overline{c}\langle d\rangle]\!] \qquad \text{(a copy of } F \text{ is activated)}
\end{array}$$

□

Now we have to prove the correctness of $\mathcal{T}$. For this, we have to show that $\mathcal{T}$ returns a $\approxeq^{\mathrm{c}}_{\mathrm{Ct}}$ agent which is triggered and respects Ob.

Theorem 4.5.8 (correctness of $\mathcal{T}$) *For each $A \in HO\pi^{\mathrm{o}}$ with $A :: Ob$, we have*

1. $\mathcal{T}[\![A]\!]$ *is triggered;*

2. $\mathcal{T}[\![A]\!] :: Ob$;

3. $\mathcal{T}[\![A]\!] \approx^{\mathrm{c}}_{\mathrm{Ct}} A$.

PROOF:

(1,2) : Immediate from the definition of $\mathcal{T}$.

(3) Induction on the structure of A. Using some η-conversion we can assume that each non-binding occurrence of a variable is followed by its arguments. Then there are only two cases in which $\mathcal{T}$ does not act as an homomorphism.

- $A = \overline{a}\langle F\rangle.P$

  We have, for $m \notin fn(A)$:

$$\begin{array}{rcl} \overline{a}\langle F\rangle.P & \approx^{\mathrm{c}}_{\mathrm{Ct}} & \text{(Theorem 4.4.7)} \\ \big(\overline{a}\langle Tr_m\rangle.P\big)\,\{m := F\} & \approx^{\mathrm{c}}_{\mathrm{Ct}} & \text{(induction twice)} \\ \big(\overline{a}\langle Tr_m\rangle.\mathcal{T}[\![P]\!]\big)\,\{m := \mathcal{T}[\![F]\!]\} & = & \mathcal{T}[\![A]\!] \end{array}$$

- $A = X\langle F\rangle$

  Similar to the previous case. $\square$

It is helpful to see precisely the operational correspondence between P and $\mathcal{T}[\![P]\!]$. This is important to understand better the relationship between them, but even more, because it gives us the intuitions for the definition of a simpler characterisation of $\approx_{\mathrm{Ct}}$, by evidencing some minimum test for higher-order actions (we shall develop this further in Section 4.7.1). An action $P \xrightarrow{\tau} P'$ whose derivation uses the rule COM with higher-order actions as premises (i.e., an agent is transmitted) is called a *higher-order reduction*; if the premises in the rule COM are first-order actions, then $P \xrightarrow{\tau} P'$ is called a *first-order reduction*. The result of the Lemma 4.5.9 is organised in the four clauses (a), (b), (c) and (d), which treat respectively, higher-order outputs, higher-order inputs, higher-order reductions and, finally, first-order inputs/outputs and reductions. Actually, for higher-order inputs we only deal with inputs of triggers, because — as we shall see better in Section 4.7.1 — this is the interesting case. In clause (c), F represents the abstraction exchanged in the interaction which has taken place in P. By contrast, in the corresponding triggered agent $\mathcal{T}[\![P]\!]$, the agent transmitted is a trigger Tr_m to $\mathcal{T}[\![F]\!]$. This is the reason for the peculiar structure of the derivative of $\mathcal{T}[\![P]\!]$. First we need a lemma:

Lemma 4.5.9 *Let $A \in HO\pi^{\circ}$ and $m \notin fn(A)$. Then* $\mathcal{T}[\![A\{Tr_m/X\}]\!] = \mathcal{T}[\![A]\!]\{Tr_m/X\}$.

PROOF: Induction on the structure of A. $\square$

Lemma 4.5.10 (operational correspondence for $\mathcal{T}$ on strong transitions) *Let $m \notin fn(P, \mathcal{T}[\![P]\!])$.*

1. (a) *If $P \stackrel{(\boldsymbol{\nu}\,\widetilde{b})\overline{a}\langle F\rangle}{\longrightarrow} P'$, then $\mathcal{T}[\![P]\!] \stackrel{(\boldsymbol{\nu}\,m)\overline{a}\langle Tr_m\rangle}{\longrightarrow} \equiv \mathcal{T}[\![\boldsymbol{\nu}\,\widetilde{b}\,(P' \mid\, !\,m \star F)]\!]$;*

   (b) *if $P \stackrel{a\langle Tr_m\rangle}{\longrightarrow} P'$, then $\mathcal{T}[\![P]\!] \stackrel{a\langle Tr_m\rangle}{\longrightarrow} \mathcal{T}[\![P']\!]$;*

   (c) *if $P \stackrel{\tau}{\longrightarrow} P'$ is a higher-order reduction, then $\widetilde{b}, G, F$ exist s.t.* $P' \equiv \boldsymbol{\nu}\,\widetilde{b}\,G\langle F\rangle$, *and* $\mathcal{T}[\![P]\!] \stackrel{\tau}{\longrightarrow} \equiv \mathcal{T}[\![\boldsymbol{\nu}\,\widetilde{b}\,\big(G\langle Tr_m\rangle\,\{m := F\}\big)]\!]$;

   (d) *if $P \stackrel{\mu}{\longrightarrow} P'$, and μ is a first-order input or output or a first-order reduction, then $\mathcal{T}[\![P]\!] \stackrel{\mu}{\longrightarrow} \mathcal{T}[\![P']\!]$.*

2. *The converse of (1), i.e. :*

   (a) *If $\mathcal{T}[\![P]\!] \stackrel{(\boldsymbol{\nu}\,m)\overline{a}\langle Tr_m\rangle}{\longrightarrow} P''$, then P' exists s.t.* $P \stackrel{(\boldsymbol{\nu}\,\widetilde{b})\overline{a}\langle F\rangle}{\longrightarrow} P'$, *and* $P'' \equiv \mathcal{T}[\![\boldsymbol{\nu}\,\widetilde{b}\,(P' \mid\, !\,m \star F)]\!]$;

   (b) *if $\mathcal{T}[\![P]\!] \stackrel{a\langle Tr_m\rangle}{\longrightarrow} P''$ then P' exists s.t.* $P \stackrel{a\langle Tr_m\rangle}{\longrightarrow} P'$, *and* $P'' = \mathcal{T}[\![P']\!]$;

   (c) *if $\mathcal{T}[\![P]\!] \stackrel{\tau}{\longrightarrow} P''$ is a higher-order reduction, then $\widetilde{b}, G, F$ exist s.t.* $P \stackrel{\tau}{\longrightarrow} \equiv \boldsymbol{\nu}\,\widetilde{b}\,(G\langle F\rangle)$ *and* $P'' \equiv \mathcal{T}[\![\boldsymbol{\nu}\,\widetilde{b}\,\big(G\langle Tr_m\rangle\,\{m := F\}\big)]\!]$;

   (d) *if $\mathcal{T}[\![P]\!] \stackrel{\mu}{\longrightarrow} P''$ and μ is a first-order input or output or a first-order reduction, then P' exists s.t. $P \stackrel{\mu}{\longrightarrow} P'$, and $P'' = \mathcal{T}[\![P']\!]$.*

PROOF: By induction on the structure of P. We consider the three main cases.

- P is of the form $\overline{a}\langle F\rangle.Q$.

  Then $\mathcal{T}[\![P]\!] = \left(\overline{a}\langle Tr_m\rangle.\mathcal{T}[\![Q]\!]\right)\{m := \mathcal{T}[\![F]\!]\}$. We have:

$$P \xrightarrow{\overline{a}\langle F\rangle} Q \quad \text{and}$$
$$\mathcal{T}[\![P]\!] \xrightarrow{(\boldsymbol{\nu}\, m)\overline{a}\langle Tr_m\rangle} \mathcal{T}[\![Q]\!] \mid !\, m \star \mathcal{T}[\![F]\!] = \mathcal{T}[\![Q \mid !\, m \star F]\!]$$

- P is of the form $a(X).Q$.

  Then $\mathcal{T}[\![P]\!] = a(X).\mathcal{T}[\![Q]\!]$. We have:

$$P \xrightarrow{a\langle Tr_m\rangle} Q\{Tr_m/X\} \quad \text{and}$$
$$\mathcal{T}[\![P]\!] \xrightarrow{a\langle Tr_m\rangle} \mathcal{T}[\![Q]\!]\{Tr_m/X\} = \mathcal{T}[\![Q\{Tr_m/X\}]\!]$$

  where the last equality holds by Lemma 4.5.9.

- P is of the form $P_1 \mid P_2$.

  Then $\mathcal{T}[\![P]\!] = \mathcal{T}[\![P_1]\!] \mid \mathcal{T}[\![P_2]\!]$. The interesting case is (c). We consider only (1.c), as (2.c) is analogous. Suppose that $P_1 \xrightarrow{a\langle F\rangle} P_1'$, and $P_2 \xrightarrow{(\boldsymbol{\nu}\,\widetilde{b})\overline{a}\langle F\rangle} P_2'$ with $\widetilde{b} \cap fn(P_1) = \emptyset$. Then from the communication law, we infer $P \xrightarrow{\tau} \boldsymbol{\nu}\,\widetilde{b}\,(P_1' \mid P_2') \stackrel{def}{=} P'$. By the induction hypothesis, we have

$$\mathcal{T}[\![P_2]\!] \xrightarrow{(\boldsymbol{\nu}\, m)\overline{a}\langle Tr_m\rangle} P_2'' \equiv \mathcal{T}[\![\boldsymbol{\nu}\,\widetilde{b}\,(P_2' \mid !\, m \star F)]\!]. \tag{4.1}$$

  Let us try to use the inductive hypothesis on P_1 too. By Lemma 4.1.3, G_1 exists s.t. $P_1' = G_1\langle F\rangle$ and $P_1 \xrightarrow{a\langle Tr_m\rangle} G_1\langle Tr_m\rangle$; therefore by induction

$$\mathcal{T}[\![P_1]\!] \xrightarrow{a\langle Tr_m\rangle} \mathcal{T}[\![G_1\langle Tr_m\rangle]\!] \stackrel{def}{=} P_1'' \tag{4.2}$$

  From (4.1) and (4.2) we infer

$$\begin{aligned}
\mathcal{T}[\![P]\!] \xrightarrow{\tau} P'' &= \boldsymbol{\nu}\, m\,(P_1'' \mid P_2'') \\
&\equiv \boldsymbol{\nu}\, m\left(\mathcal{T}[\![G_1\langle Tr_m\rangle]\!] \mid \mathcal{T}[\![\boldsymbol{\nu}\,\widetilde{b}\,(P_2' \mid !\, m \star F)]\!]\right) \\
&= \mathcal{T}[\![\boldsymbol{\nu}\, m\left(G_1\langle Tr_m\rangle \mid \boldsymbol{\nu}\,\widetilde{b}\,(P_2' \mid !\, m \star F)\right)]\!] \\
&\equiv \mathcal{T}[\![\boldsymbol{\nu}\,\widetilde{b}\left((G_1\langle Tr_m\rangle \mid P_2')\,\{m := F\}\right)]\!]
\end{aligned} \tag{4.3}$$

Finally, we are ready to prove that (1.c) holds. Set $G \stackrel{def}{=} (X)(G_1\langle X\rangle \mid P_2')$: We have $P' = \boldsymbol{\nu}\,\widetilde{b}(P_1' \mid P_2') = \boldsymbol{\nu}\,\widetilde{b}(G_1\langle F\rangle \mid P_2') = \boldsymbol{\nu}\,\widetilde{b}\,G\langle F\rangle$, and by (4.3), $P'' \equiv \mathcal{T}[\![\boldsymbol{\nu}\,\widetilde{b}\big(G\langle Tr_m\rangle\,\{m := F\}\big)]\!]$. $\square$

4.6 Triggered bisimulation

We show that the class $THO\pi$ of triggered agents is amenable to an analysis in which only first-order actions and higher-order actions in triggered form — that is in the form $a\langle Tr_m\rangle$ or $(\boldsymbol{\nu}\, m\,)\overline{a}\langle Tr_m\rangle$ — are employed. We call these actions *basic actions.*

We start by showing that the class $THO\pi$ is closed with respect to the production of basic actions and that every higher-order output that a process in $THO\pi$ can perform is indeed a basic action.

Lemma 4.6.1 *Let $P \in THO\pi$ and $P \xrightarrow{\mu} P'$. Then it holds that*

1. *If μ is a basic action, then $P' \in THO\pi$;*

2. *If μ is a higher-order output, then $\mu = (\boldsymbol{\nu}\, m)\overline{a}\langle Tr_m\rangle$ and for some $P'', F, \widetilde{b}$ with $m \notin fn(P'', F) \cup \widetilde{b}$ we have $P' \equiv \boldsymbol{\nu}\,\widetilde{b}(P'' \mid\, !\, m \star F)$; moreover, by (1) $P' \in THO\pi$.*

Proof: By induction on the structure of P. We consider the two main cases:

Case (a) $P = a(X).Q$

If $\mu = a\langle Tr_m\rangle$, then $P \xrightarrow{\mu} P' = Q\{Tr_m/X\}$. By Lemma 4.5.2, $P' \in THO\pi$.

Case (b) $P = P_1 \mid P_2$ and $P \xrightarrow{\mu} P'$.

There are four cases to examine, according to the rule which was used to infer the action for P.

Case (b.1) $P_1 \xrightarrow{\mu} P_1'$ and $P' = P_1' \mid P_2$.

Let us look at (2), as (1) is similar (actually simpler). By induction, $\mu = (\boldsymbol{\nu}\, m)\overline{a}\langle Tr_m\rangle$, and $P_1' \equiv \boldsymbol{\nu}\,\widetilde{b}\,(P_1'' \mid\, !\, m \star F)$ with $m \notin fn(P_1'', F) \cup \widetilde{b}$. Using α-conversion we can assume $(\widetilde{b} \cup m) \cap fn(P_2) = \emptyset$. Therefore

$$P' = (\boldsymbol{\nu}\,\widetilde{b}\,(P_1'' \mid !\,m \star F)) \mid P_2 \equiv \boldsymbol{\nu}\,\widetilde{b}\,((P_1'' \mid P_2) \mid !\,m \star F).$$

Case (b.2) $P_2 \xrightarrow{\mu} P_2'$ and $P' = P_1 \mid P_2'$.

Similar to (b.1).

Case (b.3) $P_1 \xrightarrow{(\boldsymbol{\nu}\,\widetilde{b})\overline{a}\langle F\rangle} P_1'$, $P_2 \xrightarrow{a\langle F\rangle} P_2'$ and $\mu = \tau$, $P' = \boldsymbol{\nu}\,\widetilde{b}\,(P_1' \mid P_2')$.

By induction, $P_1' \in THO\pi$ and the output emitted by P_1 is a trigger, i.e. we have $(\boldsymbol{\nu}\,\widetilde{b})\overline{a}\langle F\rangle = (\boldsymbol{\nu}\,m)\overline{a}\langle Tr_m\rangle$. Therefore we can exploit the induction assumption also on P_2 and infer $P_2' \in THO\pi$.

Case (b.4) $P_1 \xrightarrow{a\langle F\rangle} P_1'$, $P_2 \xrightarrow{(\boldsymbol{\nu}\,\widetilde{b})\overline{a}\langle F\rangle} P_2'$ and $\mu = \tau$, $P' = \boldsymbol{\nu}\,\widetilde{b}\,(P_1' \mid P_2')$.

Similar to (b.3). $\square$

Having proved that $THO\pi$ is closed w.r.t. the production of basic actions, it makes sense to define a bisimulation in which *only* basic actions are involved.

Definition 4.6.2 (triggered bisimulation) *A relation $\mathcal{R}$ on $THO\pi$ processes is a* strong triggered simulation *if $\mathcal{R}$ is a strong first-order simulation and $P\ \mathcal{R}\ Q$ implies, for $m \notin fn(P,Q)$:*

1. *whenever $P \xrightarrow{a\langle Tr_m\rangle} P'$, there exists Q' s.t. $Q \xrightarrow{a\langle Tr_m\rangle} Q'$ and $P'\ \mathcal{R}\ Q'$,*

2. *whenever $P \xrightarrow{(\boldsymbol{\nu}\,m)\overline{a}\langle Tr_m\rangle} P'$, there exists Q' s.t. $Q \xrightarrow{(\boldsymbol{\nu}\,m)\overline{a}\langle Tr_m\rangle} Q'$ and $P'\ \mathcal{R}\ Q'$.*

A relation $\mathcal{R}$ on $THO\pi$ processes is a strong triggered bisimulation, *briefly $\simeq_{\mathrm{Tr}}$-bisimulation, if $\mathcal{R}$ and $\mathcal{R}^{-1}$ are strong triggered simulations. We say that P and Q are* strong triggered bisimilar, *briefly $P \simeq_{\mathrm{Tr}} Q$, if $P\ \mathcal{R}\ Q$, for some $\simeq_{\mathrm{Tr}}$-bisimulation $\mathcal{R}$.* $\square$

Notice that in the clauses (1) and (2) it is enough to take *some* $m \notin fn(P,Q)$. Any other choice of m — with $m \notin fn(P,Q)$ — represents the same kind of test on the processes since, intuitively, the difference between the two choices is expressed by an injective mapping on names. This observation is to evidence the simplicity of the clauses of Definition 4.6.2 compared to those in the definition of context bisimulation: In clause (2) no universal quantification and no check whatsoever on the agents emitted in the output is necessary; similarly, in clause (1) a single fresh trigger is used, whereas in context bisimulation every agent which can possibly be received is taken into account.

In line with Definition 4.6.2, $\simeq_{\mathrm{Tr}}$ is extended to higher-order abstractions and open agents employing "fresh" triggers. Thus, if F, E are higher-order abstractions and $m \notin fn(F, E)$, then $F \simeq_{\mathrm{Tr}} E$ holds if $F\langle Tr_m \rangle \simeq_{\mathrm{Tr}} E\langle Tr_m \rangle$. On open agents A_1, A_2 with $fv(A_1, A_2) = \{X_1, \ldots, X_n\}$, if $\{m_1, \ldots, m_n\}$ is a set of distinct names not occurring free in A_1, A_2, then $A_1 \simeq_{\mathrm{Tr}} A_2$ holds if $A_1\{Tr_{m_1}, \ldots, Tr_{m_n}/X_1, \ldots, X_n\} \simeq_{\mathrm{Tr}} A_2\{Tr_{m_1}, \ldots, Tr_{m_n}/X_1, \ldots, X_n\}$. The definition of $\simeq_{\mathrm{Tr}}$ on first-order abstractions, the definition of *strong triggered congruence* ($\simeq^{\mathrm{c}}_{\mathrm{Tr}}$), *weak triggered bisimulation* and *congruence* (resp. $\approxeq_{\mathrm{Tr}}$ and $\approxeq^{\mathrm{c}}_{\mathrm{Tr}}$) are standard. Finally, for $\simeq_{\mathrm{Tr}}$ and $\approxeq_{\mathrm{Tr}}$ too, the up-to techniques for defining bisimulation described in Section 2.5.3 work fine.

4.6.1 Weak triggered and context bisimulations coincide

Lemma 4.6.3 *$P \approxeq_{\mathrm{Tr}} Q$ implies:*

1. $\boldsymbol{\nu}\, a\, P \approxeq_{\mathrm{Tr}} \boldsymbol{\nu}\, a\, Q$;

2. $P \mid R \approxeq_{\mathrm{Tr}} Q \mid R$.

PROOF: By Lemma 4.6.1 and definition of $\approxeq_{\mathrm{Tr}}$, the only actions we need to consider are communications of triggers. Then the proofs become similar to the corresponding ones for weak early bisimulation for π-calculus.

We only examine (2). For this, we prove that

$$\mathcal{R} = \{(P_1 \mid R, P_2 \mid R) \; : \; P_1, P_2, R \in THO\pi \textit{ and } P_1 \approxeq_{\mathrm{Tr}} P_2\}$$

is a $\approxeq_{\mathrm{Tr}}$-bisimulation up-to restriction. Suppose $(P_1 \mid R, P_2 \mid R) \in \mathcal{R}$ and $P_1 \mid R \xrightarrow{\mu} Q_1$. We proceed by case analysis on the rule used to infer this action. If COM has been used with P_1 performing the output, then for some m, we have

$$P_1 \xrightarrow{(\boldsymbol{\nu}\, m)\overline{a}\langle Tr_m \rangle} P_1',\ R \xrightarrow{a\langle Tr_m \rangle} R',\ \mu = \tau \text{ and } Q_1 = \boldsymbol{\nu}\, m\, (P_1' \mid R').$$

Since $P_1 \approxeq_{\mathrm{Tr}} P_2$, and assuming m not free in P_2, we have

$$P_2 \overset{(\boldsymbol{\nu}\, m)\overline{a}\langle Tr_m\rangle}{\Longrightarrow} P_2' \approxeq_{\mathrm{Tr}} P_1', \text{ and } P_2 \mid R \overset{\tau}{\Longrightarrow} \boldsymbol{\nu}\, m\,(P_2' \mid R').$$

This is enough because $\mathcal{R}$ is a bisimulation up-to restriction and by Lemma 4.6.1, we have $P_1', P_2', R' \in THO\pi$. All remaining cases are similar. □

Proposition 4.6.4 *Let $P, Q \in THO\pi$; then $P \approxeq_{\mathrm{Ct}} Q$ implies $P \approxeq_{\mathrm{Tr}} Q$.*

Proof: We proof that

$$\mathcal{R} = \{(P_1, P_2) \ :\ P_1, P_2 \in THO\pi \text{ and } P_1 \approxeq_{\mathrm{Ct}} P_2\}$$

is a $\approxeq_{\mathrm{Tr}}$-bisimulation. Let $(P_1, P_2) \in \mathcal{R}$ and suppose that $P_1 \xrightarrow{\mu} P_1'$. There are two cases:

Case (a) $\mu = (\boldsymbol{\nu}\, m)\overline{a}\langle Tr_m\rangle$.

Remember that by Lemma 4.6.1, this is the only possible form of higher-order output for P_1 and P_2. This and $P_1 \approxeq_{\mathrm{Ct}} P_2$ imply that there exists P_2' s.t. $P_2 \overset{(\boldsymbol{\nu}\, m)\overline{a}\langle Tr_m\rangle}{\Longrightarrow} P_2'$ and for every G, with $m \notin fn(G)$:

$$\boldsymbol{\nu}\, m\,(G\langle Tr_m\rangle \mid P_1') \approxeq_{\mathrm{Ct}} \boldsymbol{\nu}\, m\,(G\langle Tr_m\rangle \mid P_2') \tag{4.4}$$

In order to close the bisimulation we have to show that $P_1' \approxeq_{\mathrm{Ct}} P_2'$. Lemma 4.6.1 tells us that there exist $\tilde{b}_1, \tilde{b}_2, F_1, F_2, P_1'', P_2''$ s.t.

$$P_1' \equiv \boldsymbol{\nu}\, \tilde{b}_1 \left(P_1'' \mid !\, m \star F_1\right), \qquad P_2' \equiv \boldsymbol{\nu}\, \tilde{b}_2 \left(P_2'' \mid !\, m \star F_2\right)$$

where $m \notin fn(F_1, F_2, P_1'', P_2'') \cup \tilde{b}_1 \cup \tilde{b}_2$. Suppose m' is a fresh name. Now, changing m for m' does not affect the equivalence among processes, i.e. we have

$$\begin{array}{lcl} \boldsymbol{\nu}\, \tilde{b}_1 \left(P_1'' \mid !\, m \star F_1\right) & \approxeq_{\mathrm{Ct}} & \boldsymbol{\nu}\, \tilde{b}_2 \left(P_2'' \mid !\, m \star F_2\right) \quad \text{iff} \\ \boldsymbol{\nu}\, \tilde{b}_1 \left(P_1'' \mid !\, m' \star F_1\right) & \approxeq_{\mathrm{Ct}} & \boldsymbol{\nu}\, \tilde{b}_2 \left(P_2'' \mid !\, m' \star F_2\right) \end{array}$$

Therefore if we can prove the latter equivalence we get $P_1' \approxeq_{\mathrm{Ct}} P_2'$. The advantage with this is that we shall be able to exploit (4.4). Since m is not free in $\boldsymbol{\nu}\, \tilde{b}_1 \left(P_1'' \mid !\, m' \star F_1\right)$ and $\boldsymbol{\nu}\, \tilde{b}_2 \left(P_2'' \mid !\, m' \star F_2\right)$, and using the factorisation theorem 4.4.7 we have

$$\begin{aligned}
\boldsymbol{\nu}\,\widetilde{b}_1\left(P''_1 \mid !\,m' \star F_1\right) &\approx_{\mathrm{Ct}} \\
\boldsymbol{\nu}\,\widetilde{b}_1\left(\left(P''_1 \mid !\,m' \star Tr_m\right)\{m := F_1\}\right) &\equiv \\
\boldsymbol{\nu}\,m\left(!\,m' \star Tr_m \mid \boldsymbol{\nu}\,\widetilde{b}_1\,(P''_1 \mid !\,m \star F_1)\right) &= \\
\boldsymbol{\nu}\,m\,(!\,m' \star Tr_m \mid P'_1) &\stackrel{def}{=} \quad Q_1
\end{aligned}$$

and similarly,

$$\boldsymbol{\nu}\,\widetilde{b}_2\left(P''_2 \mid !\,m' \star F_2\right) \;\approx_{\mathrm{Ct}}\; \boldsymbol{\nu}\,m\,(!\,m' \star Tr_m \mid P'_2) \stackrel{def}{=} Q_2$$

Now $Q_1 \approx_{\mathrm{Ct}} Q_2$ can be derived from (4.4), for $G = (X)\,!\,m' \star X$.

Case (b) μ is an input or a first-order action.

In this case from $P_1 \approx_{\mathrm{Ct}} P_2$, we derive that there exists P'_2 s.t. $P_2 \stackrel{\hat{\mu}}{\Longrightarrow} P'_2$ and $P'_1 \approx_{\mathrm{Ct}} P'_2$. Thus $(P_1, P_2) \in \mathcal{R}$. $\square$

Proposition 4.6.5 *Let $P, Q \in THO\pi$; then $P \approx_{\mathrm{Tr}} Q$ implies $P \approx_{\mathrm{Ct}} Q$.*

Proof: We show that

$$\mathcal{R} = \{(P, Q) \;:\; P \approx_{\mathrm{Tr}} Q\}$$

is a $\approx_{\mathrm{Ct}}$-bisimulation up-to $\approx_{\mathrm{Ct}}$. For $P\,\mathcal{R}\,Q$, we have to examine the higher-order actions of P. The input is more delicate and we consider it first. Suppose that $P \stackrel{a\langle F\rangle}{\Longrightarrow} P'$. We have to find Q', Q'' and P'' s.t.

$$Q \stackrel{a\langle F\rangle}{\Longrightarrow} Q' \text{ and } P' \approx_{\mathrm{Ct}} P''\,\mathcal{R}\,Q'' \approx_{\mathrm{Ct}} Q'. \tag{4.5}$$

If $P \stackrel{a\langle F\rangle}{\Longrightarrow} P'$, then there exist P_1, P_2 s.t.

$$P \Longrightarrow P_1 \stackrel{a\langle F\rangle}{\longrightarrow} P_2 \Longrightarrow P'$$

Using Corollary 4.4.8(1), there exists P_3 s.t. if $m \notin fn(P, Q)$, then

$$P_1 \stackrel{a\langle Tr_m\rangle}{\longrightarrow} P_3 \qquad \text{and} \qquad P_3\,\{m := F\} \approx^{c}_{\mathrm{Ct}} P_2 \tag{4.6}$$

Moreover, $P_3 \in THO\pi$ by Lemma 4.6.1. In general, F is not triggered. We can "trigger" F using the mapping $\mathcal{T}$. Let $F' = \mathcal{T}[\![F]\!]$: By the correctness of $\mathcal{T}$ (Theorem 4.5.8(3)) we have $F' \approx^{c}_{\mathrm{Ct}} F$. Hence

$$P_3\{m := F'\} \approxeq^{\mathrm{c}}_{\mathrm{Ct}} P_3\{m := F\}$$

From this and (4.6), by transitivity of $\approxeq^{\mathrm{c}}_{\mathrm{Ct}}$, also $P_3\{m := F'\} \approxeq^{\mathrm{c}}_{\mathrm{Ct}} P_2$. Therefore, since $P_2 \Longrightarrow P'$, for some P_4,

$$P_3\{m := F'\} \Longrightarrow P_4 \approxeq_{\mathrm{Ct}} P' \tag{4.7}$$

It holds that $P_4 \in THO\pi$ because from $P_3 \in THO\pi$ and Lemma 4.5.3, we have $P_3\{m := F'\} \in THO\pi$ and, by Lemma 4.6.1, $THO\pi$ is closed w.r.t. basic actions. Summarising what we have obtained so far:

$$P \stackrel{a\langle F\rangle}{\Longrightarrow} P' \approxeq_{\mathrm{Ct}} P_4 \in THO\pi$$

Let $P'' \stackrel{def}{=} P_4$ in (4.5); it remains to find Q' and Q''. From $Q \approxeq_{\mathrm{Tr}} P$ and $P \stackrel{a\langle Tr_m\rangle}{\Longrightarrow} P_3$ we get that there exists Q_1 s.t.

$$Q \stackrel{a\langle Tr_m\rangle}{\Longrightarrow} Q_1 \quad \text{and} \quad Q_1 \approxeq_{\mathrm{Tr}} P_3$$

By the congruence properties of $\approxeq_{\mathrm{Tr}}$, we also have

$$Q_1\{m := F'\} \approxeq_{\mathrm{Tr}} P_3\{m := F'\} \tag{4.8}$$

Reversing the reasoning used to find P_4 and prove $P' \approxeq_{\mathrm{Ct}} P_4$ we find Q_2 s.t.

$$Q \stackrel{a\langle F\rangle}{\Longrightarrow} Q_2 \qquad \text{and} \qquad Q_2 \approxeq_{\mathrm{Ct}} Q_1\{m := F'\} \tag{4.9}$$

Now, from (4.7), (4.8), (4.9) and the definitions of $\approxeq_{\mathrm{Tr}}$, $\approxeq_{\mathrm{Ct}}$, we can complete the following diagram:

$$\begin{array}{ccccc}
P_3\{m := F'\} & \approxeq_{\mathrm{Tr}} & Q_1\{m := F'\} & \approxeq_{\mathrm{Ct}} & Q_2 \\
\Downarrow & & \Downarrow & & \Downarrow \\
P_4 & \approxeq_{\mathrm{Tr}} & Q_3 & \approxeq_{\mathrm{Ct}} & Q_4
\end{array}$$

For $Q'' \stackrel{def}{=} Q_3$, $Q' \stackrel{def}{=} Q_4$ in (4.5), this concludes the analysis.

Now the case of higher-order output. Because $P, Q \in THO\pi$, they can only output triggers (Lemma 4.6.1). Suppose that $P \stackrel{(\boldsymbol{\nu}\, m)\overline{a}\langle Tr_m\rangle}{\Longrightarrow} P'$. We can assume $m \notin fn(P, Q)$ and then, since $P \approxeq_{\mathrm{Tr}} Q$, for some Q' we have $Q \stackrel{(\boldsymbol{\nu}\, m)\overline{a}\langle Tr_m\rangle}{\Longrightarrow} Q' \approxeq_{\mathrm{Tr}} P'$. By definition of $\approxeq_{\mathrm{Ct}}$ and $\mathcal{R}$, we have to check that for every G with $m \notin fn(G)$,

$$\boldsymbol{\nu}\, m\, (G\langle Tr_m\rangle \mid P') \approxeq_{\mathrm{Ct}} \ \mathcal{R} \ \approxeq_{\mathrm{Ct}} \boldsymbol{\nu}\, m\, (G\langle Tr_m\rangle \mid Q').$$

Appealing to the correctness of $\mathcal{T}$ (Theorem 4.5.8(3)) and the definition of $\mathcal{R}$, this holds if

$$R_1 \stackrel{def}{=} \boldsymbol{\nu}\, m\, (\mathcal{T}[\![G\langle Tr_m\rangle]\!] \mid P') \approxeq_{\mathrm{Tr}} \boldsymbol{\nu}\, m\, (\mathcal{T}[\![G\langle Tr_m\rangle]\!] \mid Q') \stackrel{def}{=} R_2.$$

Since $R_1, R_2 \in THO\pi$ and $P' \approxeq_{\mathrm{Tr}} Q'$, we can infer $R_1 \approxeq_{\mathrm{Tr}} R_2$ from congruence of $\approxeq_{\mathrm{Tr}}$ over parallel composition and restriction proved in Theorem 4.6.3. $\square$

The previous two propositions give:

Corollary 4.6.6 *For all closed triggered processes P, Q, it holds that $P \approxeq_{\mathrm{Tr}} Q$ iff $P \approxeq_{\mathrm{Ct}} Q$.* $\square$

4.7 Uses of triggered agents

Triggered agents are a powerful tool for reasoning with higher-order process calculi. A striking example is presented in the next chapter, where their use is essential to obtain full abstraction for a compilation from higher-order to first-order agents. We believe that triggered agents would also be helpful for the development of an algebraic theory for HOπ. In this section, they are exploited to derive two alternative characterisations of context bisimulation on the whole class of HOπ processes, i.e. not necessarily in triggered form. The first is a more efficient characterisation, that is, easier to verify. The second is a characterisation in terms of barbed equivalence.

4.7.1 Normal bisimulation

The appellation "normal bisimulation" is to indicate that it is obtained by "normalising" the clauses of context bisimulation. Let us first present the definition; then we shall comment on it.

Definition 4.7.1 (normal bisimulation) *A relation $\mathcal{R}$ on HOπ processes is a* strong normal simulation *if $\mathcal{R}$ is a strong first-order simulation and $P\ \mathcal{R}\ Q$ implies, for $m \notin fn(P, Q)$:*

1. *whenever* $P \xrightarrow{a\langle Tr_m\rangle} P'$, *then* Q' *exists s.t.* $Q \xrightarrow{a\langle Tr_m\rangle} Q'$ *and* $P' \mathcal{R} Q'$,

2. *whenever* $P \xrightarrow{(\boldsymbol{\nu}\widetilde{b})\overline{a}\langle F\rangle} P'$, *then* $Q', \widetilde{c}, E$ *exist s.t. if* m *is a fresh name,* $Q \xrightarrow{(\boldsymbol{\nu}\widetilde{c})\overline{a}\langle E\rangle} Q'$ *and* $\boldsymbol{\nu}\,\widetilde{b}\,(P' \mid\, !\,m \star F)\;\mathcal{R}\;\boldsymbol{\nu}\,\widetilde{c}\,(Q' \mid\, !\,m \star E)$.

A relation $\mathcal{R}$ *on HOπ processes is a* strong normal bisimulation, *briefly* $\simeq_{\mathrm{Nr}}$-*bisimulation, if* $\mathcal{R}$ *and* $\mathcal{R}^{-1}$ *are strong normal simulations. We say that* P *and* Q *are* strong normal bisimilar, *briefly* $P \simeq_{\mathrm{Nr}} Q$, *if* $P \mathcal{R} Q$, *for some* $\simeq_{\mathrm{Nr}}$-*bisimulation* $\mathcal{R}$. □

The extension of $\simeq_{\mathrm{Nr}}$ to abstractions and open agents, its congruence $\simeq^{\mathrm{c}}_{\mathrm{Nr}}$, and their corresponding weak versions $\approxeq_{\mathrm{Nr}}$ and $\approxeq^{\mathrm{c}}_{\mathrm{Nr}}$ are defined similarly as for triggered bisimulation. In particular, the definitions on higher-order abstractions and open agents are obtained employing only fresh triggers. Both the strong and the weak version of normal bisimulation are equivalence and the up-to techniques described in Section 2.5.3 apply to them.

As for trigger bisimulation, we want to stress that in the clauses (1) and (2) of Definition 4.7.1, it is enough to pick *some* fresh name m, since the specific choice of the fresh name does not affect the equivalence of the resulting processes.

The idea of normal bisimulation comes from the results on the discriminanting power of triggers shown in the previous sections. To test the equivalence between $a(X).P$ and $a(X).Q$ we do not have to try every possible instantiation of X in P and Q, but it is enough to verify it with a trigger Tr_m, as long as m is fresh. In fact, by the congruence properties of $\approxeq_{\mathrm{Ct}}$, for any F,

$$P\{Tr_m/X\} \approxeq_{\mathrm{Ct}} Q\{Tr_m/X\} \qquad \text{implies}$$
$$P\{Tr_m/X\}\,\{m := F\} \approxeq_{\mathrm{Ct}} Q\{Tr_m/X\}\,\{m := F\}$$

and then, by Theorem 4.4.7 (the factorisation theorem),

$$P\{Tr_m/X\}\,\{m := F\} \approxeq_{\mathrm{Ct}} P\{F/X\},$$
$$Q\{Tr_m/X\}\,\{m := F\} \approxeq_{\mathrm{Ct}} Q\{F/X\}$$

which yields

$$P\{F/X\} \approx_{\mathrm{Ct}} Q\{F/X\}$$

Similar reasoning can be used to justify the clause on higher-order outputs in Definition 4.7.1; but to understand it, it is also enlighting to look back at the operational correspondence between P and $\mathcal{T}[\![P]\!]$ shown in Lemma 4.5.10, and compare clause (1.a) of that lemma with clause (2) in Definition 4.7.1. It may be useful however to see why in the latter the requirement

$$\boldsymbol{\nu}\,\widetilde{b}\,(P' \mid !\,m \star F)\;\mathcal{R}\;\boldsymbol{\nu}\,\widetilde{c}\,(Q' \mid !\,m \star E)$$

cannot be made simpler without losing discriminanting power. The counter-examples we give below are in the weak case – the interesting one – and for simplicity, they employ names which can carry either a process or the empty tuple.

First of all, the guard m on F and E is necessary: For, otherwise, if R is a process like $!\,d$, which can perform an unbounded number of identical visible actions, then the processes

$$\overline{a}\langle R\rangle.\mathbf{0} \qquad\qquad \text{and} \qquad\qquad \overline{a}\langle \mathbf{0}\rangle.R$$

would be made equivalent. But they are not context bisimilar; for instance they can be distinguished when interacting with a process like $a(X).\mathbf{0}$, which discharges what it receives at a.

Now, the use of replication. Suppose we eliminate it from the clause (2) of Definition 4.7.1. Then let $R = b.b.\overline{c} \mid \overline{b}$ and consider

$$P \stackrel{def}{=} (\boldsymbol{\nu}\,b)\overline{a}\langle R\rangle.\mathbf{0} \qquad\qquad\qquad Q \stackrel{def}{=} \overline{a}\langle \mathbf{0}\rangle.\mathbf{0}$$

Again, P and Q would become equivalent since $\boldsymbol{\nu}\,b\,(m.R)$ and $m.\mathbf{0}$ are indistinguishable. However P and Q can be differentiated when interacting with a process like $a(X).(X \mid X)$ which makes two copies of the input run in parallel so that the action c in R can be observed. (In fact, only in the *linear* HOπ, where in each expression, a variable may occur free only once, the replication in question can be avoided.)

We now prove formally that $\approxeq_{\mathrm{Ct}}$ and $\approxeq_{\mathrm{Nr}}$ coincide. The inclusion $\approxeq_{\mathrm{Ct}} \subseteq \approxeq_{\mathrm{Nr}}$ is obvious, since the requirements in the definition of $\approxeq_{\mathrm{Nr}}$ are a subset of those in the definition of $\approxeq_{\mathrm{Ct}}$. To prove the opposite inclusion, we use the Lemma 4.7.3 below, which relates weak transitions of P and $\mathcal{T}[\![P]\!]$. Its proof is obtained from the operational correspondence on strong transitions in Lemma 4.5.10 and the following auxiliary lemma:

Lemma 4.7.2 *For any G, F with $m \notin fn(G, F)$, it holds that*

$$\mathcal{T}[\![\boldsymbol{\nu}\,\widetilde{b}\,G\langle F\rangle]\!] \approxeq_{\mathrm{Tr}} \mathcal{T}[\![\boldsymbol{\nu}\,\widetilde{b}\,\big(G\langle Tr_m\rangle\,\{m := F\}\big)]\!]$$

Proof: Using Theorem 4.4.7 (factorisation theorem) and Theorem 4.5.8 (correctness of $\mathcal{T}$), we derive that the two processes in the assertion of the lemma are in the relation $\approxeq_{\mathrm{Ct}}$. Since $\approxeq_{\mathrm{Ct}}$ and $\approxeq_{\mathrm{Tr}}$ coincide on triggered agents, the two processes are also in the relation $\approxeq_{\mathrm{Tr}}$. $\square$

Lemma 4.7.3 *Let $m \notin fn(P, \mathcal{T}[\![P]\!])$:*

1. *(a) If $P \overset{(\boldsymbol{\nu}\,\widetilde{b})\overline{a}\langle F\rangle}{\Longrightarrow} P'$, then P'' exists, s.t.*
   $\mathcal{T}[\![P]\!] \overset{(\boldsymbol{\nu}\,m)\overline{a}\langle Tr_m\rangle}{\Longrightarrow} P'' \approxeq_{\mathrm{Tr}} \mathcal{T}[\![\boldsymbol{\nu}\,\widetilde{b}\,(P' \mid\, !\,m \star F)]\!]$

   (b) Suppose μ is a basic input or a silent action and $bn(\mu) \cap fn(P, Q) = \emptyset$; if $P \overset{\mu}{\Longrightarrow} P'$, then P'' exists, s.t. $\mathcal{T}[\![P]\!] \overset{\mu}{\Longrightarrow} P'' \approxeq_{\mathrm{Tr}} \mathcal{T}[\![P']\!]$.

2. *The converse of (1), i.e.*

   (a) If $\mathcal{T}[\![P]\!] \overset{(\boldsymbol{\nu}\,m)\overline{a}\langle Tr_m\rangle}{\Longrightarrow} P''$, then $P', F, \widetilde{b}$ exist, s.t.
   $P \overset{(\boldsymbol{\nu}\,\widetilde{b})\overline{a}\langle F\rangle}{\Longrightarrow} P'$ and $P'' \approxeq_{\mathrm{Tr}} \mathcal{T}[\![\boldsymbol{\nu}\,\widetilde{b}\,(P' \mid\, !\,m \star F)]\!]$,

   (b) Suppose μ is a basic input or a silent action and $bn(\mu) \cap fn(P, Q) = \emptyset$; if $\mathcal{T}[\![P]\!] \overset{\mu}{\Longrightarrow} P''$ then P' exists s.t. $P \overset{\mu}{\Longrightarrow} P'$ and $P'' \approxeq_{\mathrm{Tr}} \mathcal{T}[\![P']\!]$.

Proof: We only look at (1.a), since the other cases are similar, and we proceed by induction on the length of the transition for $P \overset{(\boldsymbol{\nu}\,\widetilde{b})\overline{a}\langle F\rangle}{\Longrightarrow} P'$. If the length is one, then the result follows from Lemma 4.5.10(1.a). If the length is greater than one, then, for some P_1, either

$$P \overset{\tau}{\longrightarrow} P_1 \overset{(\boldsymbol{\nu}\,\widetilde{b})\overline{a}\langle F\rangle}{\Longrightarrow} P' \quad \text{or} \quad P \overset{(\boldsymbol{\nu}\,\widetilde{b})\overline{a}\langle F\rangle}{\longrightarrow} P_1 \overset{\tau}{\Longrightarrow} P'$$

holds. Again, since the two cases can be handled in a similar way, we only consider the first. By the induction hypothesis, we have

$$\mathcal{T}[\![P_1]\!] \stackrel{(\boldsymbol{\nu}\, m)\overline{a}\langle Tr_m \rangle}{\Longrightarrow} P''' \approxeq_{\mathrm{Tr}} \mathcal{T}[\![\boldsymbol{\nu}\, \widetilde{b}\,(P' \mid\, !\, m \star F)]\!] \tag{4.10}$$

On the other hand, we have for some P_1',

$$\mathcal{T}[\![P]\!] \stackrel{\tau}{\longrightarrow} P_1' \approxeq_{\mathrm{Tr}} \mathcal{T}[\![P_1]\!] \tag{4.11}$$

as can be inferred from Lemma 4.5.10(1.d) or from Lemma 4.5.10(1.c) together with Lemma 4.7.2, depending on whether $\mathcal{T}[\![P]\!] \stackrel{\tau}{\longrightarrow} P''$ is a first-order or a higher-order reduction. Now, from (4.11) and (4.10),

$$P_1' \stackrel{(\boldsymbol{\nu}\, m)\overline{a}\langle Tr_m \rangle}{\Longrightarrow} P'' \approxeq_{\mathrm{Tr}} P'''$$

Summarising, we have $\mathcal{T}[\![P]\!] \stackrel{(\boldsymbol{\nu}\, m)\overline{a}\langle Tr_m \rangle}{\Longrightarrow} P'' \approxeq_{\mathrm{Tr}} P''' \approxeq_{\mathrm{Tr}} \mathcal{T}[\![\boldsymbol{\nu}\, b\,(P' \mid\, !\, m \star F)]\!]$, as required. □

Theorem 4.7.4 *On HOπ processes, $\approxeq_{\mathrm{Nr}}$ and $\approxeq_{\mathrm{Ct}}$ coincide.*

Proof: We have only to prove the inclusion $\approxeq_{\mathrm{Nr}} \subseteq \approxeq_{\mathrm{Ct}}$. For this, we show that

$$\mathcal{R} \;=\{(\mathcal{T}[\![P]\!], \mathcal{T}[\![Q]\!]) \;:\; P \approxeq_{\mathrm{Nr}} Q\}$$

is a $\approxeq_{\mathrm{Tr}}$-bisimulation up-to $\approxeq_{\mathrm{Tr}}$. This is enough, because on triggered processes $\approxeq_{\mathrm{Tr}}$ and $\approxeq_{\mathrm{Ct}}$ coincide (Corollary 4.6.6), and $\mathcal{T}$ respects $\approxeq_{\mathrm{Ct}}$ (Theorem 4.5.8(3)): Hence from $\mathcal{T}[\![P]\!] \approxeq_{\mathrm{Tr}} \mathcal{T}[\![Q]\!]$, we can infer $P \approxeq_{\mathrm{Ct}} Q$.

Suppose $\mathcal{T}[\![P]\!] \stackrel{(\boldsymbol{\nu}\, m)\overline{a}\langle Tr_m \rangle}{\Longrightarrow} P''$, for $m \notin fn(P, Q)$. By Lemma 4.7.3(2.a), P', F, and $\widetilde{b}$ exist s.t.

$$P \stackrel{(\boldsymbol{\nu}\, \widetilde{b})\overline{a}\langle F \rangle}{\Longrightarrow} P' \text{ and } P'' \approxeq_{\mathrm{Tr}} \mathcal{T}[\![\boldsymbol{\nu}\, \widetilde{b}\,(P' \mid\, !\, m \star F)]\!]$$

Since $P \approxeq_{\mathrm{Nr}} Q$, there exist Q', E, and $\widetilde{c}$, s.t. $Q \stackrel{(\boldsymbol{\nu}\, \widetilde{c})\overline{a}\langle E \rangle}{\Longrightarrow} Q'$ and

$$\boldsymbol{\nu}\, \widetilde{b}\,(P' \mid\, !\, m \star F) \approxeq_{\mathrm{Nr}} \boldsymbol{\nu}\, \widetilde{c}\,(Q' \mid\, !\, m \star E)$$

Further, by Lemma 4.7.3(1.a),

$$\mathcal{T}[\![Q]\!] \stackrel{(\boldsymbol{\nu}\, m)\overline{a}\langle Tr_m \rangle}{\Longrightarrow} Q'' \approxeq_{\mathrm{Tr}} \mathcal{T}[\![\boldsymbol{\nu}\, \widetilde{c}\,(Q' \mid\, !\, m \star E)]\!]$$

Summarising, we have

$$\mathcal{T}[\![P]\!] \stackrel{(\boldsymbol{\nu}\, m)\overline{a}\langle Tm\rangle}{\Longrightarrow} \cong_{\mathrm{Tr}} \mathcal{T}[\![\boldsymbol{\nu}\, \widetilde{b}\,(P' \mid !\, m \star F)]\!] \;\mathcal{R}\; \mathcal{T}[\![\boldsymbol{\nu}\, \widetilde{c}\,(Q' \mid !\, m \star E)]\!] \cong_{\mathrm{Tr}} \stackrel{(\boldsymbol{\nu}\, m)\overline{a}\langle Tm\rangle}{\Longleftarrow} \mathcal{T}[\![Q]\!]$$

which is enough, because $\mathcal{R}$ is a $\cong_{\mathrm{Tr}}$-bisimulation up-to $\cong_{\mathrm{Tr}}$. □

Remark 4.7.5 Now that we have proved that $\cong_{\mathrm{Ct}}$ and $\cong_{\mathrm{Nr}}$ coincide, one might wonder why we did not introduce $\cong_{\mathrm{Nr}}$ directly. The reason is that we would have needed $\cong_{\mathrm{Ct}}$ to prove the congruence of $\cong_{\mathrm{Nr}}$ over parallel composition (the skeptical reader might want to try the proof himself/herself). Moreover, it is easier to convince ourselves of the naturalness of context bisimulation; we then can accept normal bisimulation as a simpler characterisation of the former. □

4.7.2 Characterisation in terms of barbed bisimulation

We have already seen the correspondence between $\cong_{\mathrm{Ct}}, \cong_{\mathrm{Nr}}$, and $\cong_{\mathrm{Tr}}$. It remains to relate these with $\approx$ (weak barbed equivalence). For this, it is advantageous to work with $\cong_{\mathrm{Nr}}$.

Theorem 4.7.6 *On HOπ processes, $\approx$ and $\cong_{\mathrm{Nr}}$ coincide.* □

PROOF: The proof is obtained with a few adjustments from the proof of Theorem 3.3.3(2), where it is shown that in the π-calculus, $\approx$ coincides weak early bisimulation. See Appendix D for the details.

Corollary 4.7.7 *The relations $\cong_{\mathrm{Ct}}$, $\cong_{\mathrm{Nr}}$, and $\approx$ coincide on HOπ processes.*

PROOF: Follows from Theorems 4.7.4 and 4.7.6. □

In Chapter 3 we defined barbed bisimulation and its congruences only for closed processes. The most reasonable way to extend it to open agents is to do as for context bisimulation, requiring the instantiation of a variable with each agent of the right sort. Corollary 4.7.7 can then be generalised to open agents. To show that $\cong_{\mathrm{Ct}}$ and $\cong_{\mathrm{Nr}}$ coincide on open agents, we need also the factorisation theorem (since $\cong_{\mathrm{Nr}}$ is defined on open agents by requiring only instantiation of variables with fresh triggers).

Corollary 4.7.8 *The relations $\approxeq_{\mathrm{Ct}}$, $\approxeq_{\mathrm{Nr}}$, $\approx$ coincide on HOπ^{o} agents.* $\square$

The result can then be extended to the congruences:

Corollary 4.7.9 *The relations $\approxeq^{c}_{\mathrm{Ct}}$, $\approxeq^{c}_{\mathrm{Nr}}$, $\approx^{c}$ coincide on HOπ^{o} agents.* $\square$

Remark 4.7.10 We think that Corollaries 4.6.6, 4.7.8 and 4.7.9 hold in the strong case too. We have not studied it though, since we are mainly interested in weak equivalences (and mapping $\mathcal{T}$ only respects these). We leave the problem for future investigations. $\square$

4.7.3 Some comments on the bisimulations considered

We hope not to have confused the reader by considering too many equivalences on HOπ agents, namely context bisimulation, normal bisimulation and barbed equivalence, plus the corresponding congruences. But we feel that this variety was somehow necessary. Having shown the correspondence among them (Corollaries 4.7.8 and 4.7.9) we can now see how each of them has its own rôle in strengthening the importance of the resulting behavioural equivalence:

- the characterisation in terms of normal bisimulation specifies some minimum requirement, or test, for higher-order actions and as such provides us with an efficient mathematical tool to verify agent equivalences;

- the characterisation in terms of context bisimulation gives us a measure of the power of the above tests;

- finally, the characterisation in terms of barbed equivalence demonstrates the naturalness of the equivalence.

In the following, whenever possible we shall try to use barbed equivalence and congruence. But the other relations will often be invoked in the proofs.

EXTENSION TO AGENTS WITH ARBITRARY ARITIES. We have presented the definitions of the bisimulations on agents which may only use unary tuples in communications and abstractions. The extension of the definitions to agents with arbitrary arities is fairly intuitive: It is just a question of combining together the clauses for first-order and higher-order actions. As an example, we look at the output clause of normal bisimulation. Take the actions:

$$P \xrightarrow{(\boldsymbol{\nu}\,\widetilde{b}\,)\overline{a}\langle F,d\rangle} P' \qquad\qquad \text{and} \qquad\qquad Q \xrightarrow{(\boldsymbol{\nu}\,\widetilde{c}\,)\overline{a}\langle G,d'\rangle} Q'$$

with $(\widetilde{b} \cup \widetilde{c}) \cap fn(P, Q) = \emptyset$. When do they "match" each other? First of all, the usual condition on first-order actions demands that the names emitted are the same, thus $d = d'$ (up-to α-conversion). Moreover, if d represents a bound name, it becomes visible at completion of the actions. Therefore, if $\widetilde{b'} = \widetilde{b} - d$, $\widetilde{c'} = \widetilde{c} - d$, and m is a fresh name, the other condition to check is

$$(\boldsymbol{\nu}\,\widetilde{b'}\,)(\,!\,m \star F \mid P') \simeq_{\mathrm{Nr}} (\boldsymbol{\nu}\,\widetilde{c'}\,)(\,!\,m \star G \mid Q')$$

Chapter 5

The Representability of HOπ in π-calculus

In this chapter we show how the higher-order constructs of HOπ can be effectively represented in π-calculus. This is obtained via a compilation $\mathcal{C}$ which intuitively, replaces the communication of an agent with the communication of the *access* to that agent. Formally, $\mathcal{C}$ is derived in two steps. The first one is the mapping $\mathcal{T}$ presented in the previous chapter which transforms every agent into a triggered agent; the second one is a mapping $\mathcal{F}$ which takes from triggered agents down to first-order agents (and first-order sortings).

We continue to work in the reduced HOπ presented in Section 4.1, where: One only value – a name or an abstraction — can be exchanged in a communication; only unary abstractions are allowed; constants are replaced by replication and all sums are finite. We remind that only the latter restriction is meaningful — it intentionally limits the applicability of our results to those HOπ agents defined from a finite number of constants and with finite sums. The arity restrictions are adopted merely to make the presentation of the results clearer. We maintain the notation introduced in Section 4.1.

We study $\mathcal{F}$ and $\mathcal{C}$ in Sections 5.1 and 5.2, respectively. In Section 5.3 we analyse Bent Thomsen's work, which first attempted the translation of a higher-order calculus, namely Plain CHOCS, into the π-calculus. Thomsen's approach employs higher-order bisimulation; we take advantage of the opportunity to compare it with barbed bisimulation.

5.1 The encoding of triggered agents

In this section we present the encoding $\mathcal{F}$ which transforms triggered HOπ agents and their sorting into π-calculus agents and first-order sortings.

In triggered agents every higher-order communication is the communication of an agent, a trigger, which is univocally determined by the unique name free in it. Consequently, it is possible to simulate in a simple way the behaviour of triggered agents using only name-passing: Instead of the trigger Tr_m, one transmits the name m. We call m a *name-trigger*, since has the same functionality as a trigger. This transformation modifies the object sort of the name at which the communication has taken place and, hence, the transformation has to be defined on *sorts* too. Moreover, since its domain and codomain are different, some care is necessary when presenting results on its faithfulness. For this, it is elegant to exploit barbed bisimulation, whose versatility allows a uniform definition in the two calculi.

Definition of the encoding

NOTATION. In the definition of triggered agents, the choice of the subsorting SOb from which the free names of the triggers are drawn is important (see preliminaries of Section 4.5). For an object sort S, we denote by SOb_S the unique subject sort s s.t. $s \mapsto S \in SOb$. As in Chapter 4, whenever possible we leave SOb implicit; for instance, we simply write $A \in THO\pi$ to say that A is a triggered agent, without specifying SOb. We shall always use m, m' as names whose sort is in SOb.

On the sorting, the encoding $\mathcal{F}$ has to replace all uses of a higher-order sort S with SOb_S. Therefore we get

$$\mathcal{F}[\![Ob]\!] = \{s \mapsto (s') \in Ob\} \bigcup \{s \mapsto (SOb_S) \ : \ s \mapsto (S) \in Ob\}.$$

The definition of $\mathcal{F}$ on agents is presented in Table 5–1. In the table, we have assumes that if $X : S$ is a variable, its lower case letter x is an unused name whose

$$\mathcal{F}[\![(a)P]\!] \stackrel{def}{=} (a)\mathcal{F}[\![P]\!] \quad \mathcal{F}[\![(X)P]\!] \stackrel{def}{=} (x)\mathcal{F}[\![P]\!] \quad \mathcal{F}[\![[a=b]P]\!] \stackrel{def}{=} [a=b]\mathcal{F}[\![P]\!]$$

$$\mathcal{F}[\![\boldsymbol{\nu}\, a\, P]\!] \stackrel{def}{=} \boldsymbol{\nu}\, a\, \mathcal{F}[\![P]\!] \quad \mathcal{F}[\![\,!\,P]\!] \stackrel{def}{=} \,!\,\mathcal{F}[\![P]\!] \quad \mathcal{F}[\![\textstyle\sum_{i\in I} P_i]\!] \stackrel{def}{=} \textstyle\sum_{i\in I}\mathcal{F}[\![P_i]\!]$$

$$\mathcal{F}[\![X\langle Tr_m\rangle]\!] \stackrel{def}{=} \overline{x}\langle m\rangle.\mathbf{0} \quad \mathcal{F}[\![X\langle b\rangle]\!] \stackrel{def}{=} \overline{x}\langle b\rangle.\mathbf{0} \quad \mathcal{F}[\![P \mid Q]\!] \stackrel{def}{=} \mathcal{F}[\![P]\!] \mid \mathcal{F}[\![Q]\!]$$

$$\mathcal{F}[\![\alpha.P]\!] \stackrel{def}{=} \mathcal{F}[\![\alpha]\!].\mathcal{F}[\![P]\!] \quad \text{where } \mathcal{F}[\![\alpha]\!] \text{ is the abbreviation:}$$

$$\mathcal{F}[\![\alpha]\!] \stackrel{def}{=} \begin{cases} \overline{a}\langle m\rangle & \text{if } \alpha = \overline{a}\langle Tr_m\rangle \\ a(x) & \text{if } \alpha = a(X) \\ \alpha & \text{otherwise} \end{cases}$$

Table 5–1: Transformation $\mathcal{F}$

sort is SOb_S. We thus ensure that distinct variables are mapped onto distinct and fresh names by mapping each variable X to its corresponding lower case letter x. The encoding $\mathcal{F}$ acts inductively on the structure of an agent. The interesting cases are those for higher-order application and prefixing. $\mathcal{F}$ acts as a homomorphism elsewhere. Since in triggered agents every non-binding occurrence of a variable is followed by its argument, we do not need a rule for variables.

Let us add a comment on the rule for higher-order application $X\langle Tr_m\rangle$. When X is instantiated to a trigger, say $Tr_{m'}$, it becomes $Tr_{m'}\langle Tr_m\rangle = \overline{m'}\langle Tr_m\rangle.\mathbf{0}$. Translating this, we expect to send m instead of Tr_m, that is, to have $\overline{m'}\langle m\rangle.\mathbf{0}$ instead of $\overline{m'}\langle Tr_m\rangle.\mathbf{0}$. With this example in mind, our rule for higher-order application should become clear. Note that from the definitions of $\mathcal{F}$ and of the abbreviation $\{m := F\}$, we have

$$\mathcal{F}[\![P\,\{m := F\}]\!] = \mathcal{F}[\![P]\!]\,\{m := \mathcal{F}[\![F]\!]\}$$

We show how $\mathcal{F}$ acts with two examples. They continue Examples 4.5.5 and 4.5.6; for readability we have abbreviated $\mathcal{F}[\![\mathcal{T}[\![P]\!]]\!]$ as $\mathcal{FT}[\![P]\!]$.

Example 5.1.1 Let $\mathcal{T}[\![P_i]\!]$, $i = 1,2$ as in Example 4.5.5, i.e.

$$\mathcal{T}[\![P_1]\!] = X\langle Tr_m\rangle\,\{m := F\} \mid (\overline{b}\langle Tr_m\rangle.\mathbf{0})\,\{m := F\}, \text{ for } F \stackrel{def}{=} (b)\mathbf{0}$$

$$\mathcal{T}[\![P_2]\!] = a(X).\big(Z\langle Tr_m\rangle\,\big\{m := (Y)\big(X\langle Tr_{m'}\rangle\,\{m' := (b)Y\langle b\rangle\}\big)\big\}\big)$$

Then we have:

$$\begin{aligned}\mathcal{FT}[\![P_1]\!] &= (\overline{x}\langle m\rangle.\mathbf{0})\,\{m := F\} \mid (\overline{b}\langle m\rangle.\mathbf{0})\,\{m := F\}\\ \mathcal{FT}[\![P_2]\!] &= a(x).\Big((\overline{z}\langle m\rangle.\mathbf{0})\,\big\{m := (y)(\overline{x}\langle m'\rangle.\mathbf{0})\,\{m' := (b)\overline{y}\langle b\rangle.\mathbf{0}\}\big\}\Big)\end{aligned}$$ □

Example 5.1.2 Let us examine how the translation works on reductions, using the processes P and $\mathcal{T}[\![P]\!]$ of Example 4.5.6, namely $P = \overline{a}\langle F\rangle.Q|a(Y).(Y\langle b\rangle|Y\langle c\rangle)$, and $\mathcal{T}[\![P]\!] = (\overline{a}\langle Tr_m\rangle.\mathcal{T}[\![Q]\!])\,\{m := \mathcal{T}[\![F]\!]\} \mid a(Y).(Y\langle b\rangle \mid Y\langle c\rangle)$. We saw that P reduces to $Q \mid F\langle b\rangle \mid F\langle c\rangle$ in one step, and that $\mathcal{T}[\![P]\!]$ needs three steps to to simulate this and reduce to a process which is $\simeq^{c}_{\mathrm{Ct}}$ with $\mathcal{T}[\![Q \mid F\langle b\rangle \mid F\langle c\rangle]\!]$. Let us show how closely $\mathcal{FT}[\![P]\!]$ follows the behaviour of $\mathcal{T}[\![P]\!]$.

$$\begin{aligned}\mathcal{FT}[\![P]\!] \quad &= \quad (\overline{a}\langle m\rangle.\mathcal{FT}[\![Q]\!])\,\{m := \mathcal{FT}[\![F]\!]\} \mid a(y).(\overline{y}\langle b\rangle.\mathbf{0} \mid \overline{y}\langle c\rangle.\mathbf{0})\\ &\equiv \quad \big(\overline{a}\langle m\rangle.\mathcal{FT}[\![Q]\!] \mid a(y).(\overline{y}\langle b\rangle.\mathbf{0} \mid \overline{y}\langle c\rangle.\mathbf{0})\big)\,\{m := \mathcal{FT}[\![F]\!]\}\\ &\longrightarrow \quad \big(\mathcal{FT}[\![Q]\!] \mid \overline{m}\langle b\rangle.\mathbf{0} \mid \overline{m}\langle c\rangle.\mathbf{0}\big)\,\{m := \mathcal{FT}[\![F]\!]\}\\ &\longrightarrow\longrightarrow\equiv \quad \big(\mathcal{FT}[\![Q]\!] \mid \mathcal{FT}[\![F]\!]\langle b\rangle \mid \mathcal{FT}[\![F]\!]\langle c\rangle\big)\,\{m := \mathcal{FT}[\![F]\!]\}\\ &\sim^{c} \quad \mathcal{FT}[\![Q]\!] \mid \mathcal{FT}[\![F]\!]\langle b\rangle \mid \mathcal{FT}[\![F]\!]\langle c\rangle = \mathcal{FT}[\![Q \mid F\langle b\rangle \mid F\langle c\rangle]\!]\end{aligned}$$

where the last $\sim^{c}$ holds because m is not free in the body. □

Syntactic correctness

Before tackling the question of the semantic correctness of $\mathcal{F}$, we have to check that its definition is syntactically meaningful by ensuring that it returns first-order agents and that there is agreement between the definition of $\mathcal{F}$ on sorts and on agents. The former is immediate. The latter holds too because all new names which are introduced (in the rule for application) or whose object part is modified (in the rule for prefixing), respect the sorting $\mathcal{F}[\![Ob]\!]$. Note that for the well-definiteness of the translation of higher-order outputs and applications, we exploit the property of triggered agents that if two triggers Tr_m and $Tr_{m'}$ have the same sort S, then the names m and m' belong to the same sort SOb_S

Theorem 5.1.3 *Suppose that $A \in THO\pi$; we have:*

1. *$\mathcal{F}[\![A]\!]$ is first-order agent;*

2. *$\mathcal{F}[\![A]\!] :: \mathcal{F}[\![Ob]\!]$.* □

Operational correspondence

The following lemma is analogous to Lemma 4.5.9 for $\mathcal{T}$.

Lemma 5.1.4 *Let $A \in THO\pi^{\circ}$. Then $\mathcal{F}[\![A\{Tr_m/X\}]\!] = \mathcal{F}[\![A]\!]\{m/x\}$.*

PROOF: By induction on the structure of A. We consider only one case; the remaining ones are easier and can be treated in a similar way. Suppose that $A = X\langle Tr_{m'}\rangle \{m' := F\}$. Then Tr_m must be of the form $(Y)\overline{m}\langle Y\rangle.\mathbf{0}$ and, for $F' \stackrel{def}{=} F\{Tr_m/X\}$, we have

$$A\{Tr_m/X\} = Tr_m\langle Tr_{m'}\rangle \{m' := F'\} = \left(\overline{m}\langle Tr_{m'}\rangle.\mathbf{0}\right) \{m' := F'\}.$$

From here we get

$$\mathcal{F}[\![A\{Tr_m/X\}]\!] = \left(\overline{m}\langle m'\rangle.\mathbf{0}\right) \{m := \mathcal{F}[\![F']\!]\}.$$

This is the same expression denoted by $\mathcal{F}[\![A]\!]\{m/x\}$; in fact we have:

$$\begin{aligned}
\mathcal{F}[\![A]\!]\{m/x\} &= \\
\mathcal{F}[\![X\langle Tr_{m'}\rangle \{m' := F\}]\!]\{m/x\} &= \\
\left(\overline{m}\langle m'\rangle.\mathbf{0}\right) \{m' := \mathcal{F}[\![F]\!]\{m/x\}\} &= \\
\left(\overline{m}\langle m'\rangle.\mathbf{0}\right) \{m := \mathcal{F}[\![F']\!]\}
\end{aligned}$$

where the last equality holds because $\mathcal{F}[\![F']\!] = \mathcal{F}[\![F\{Tr_m/X\}]\!] =$ (by induction) $\mathcal{F}[\![F]\!]\{m/x\}$. $\square$

In Table 5–1, we extended $\mathcal{F}$ to prefixes; now we extend it also to actions to better evidence the operational correspondence yielded by $\mathcal{F}$. The definition of $\mathcal{F}[\![\mu]\!]$ is obvious from the one of $\mathcal{F}[\![\alpha]\!]$: Since actions use free inputs rather than bound inputs as prefixes, we have just to replace in the definition of $\mathcal{F}[\![\alpha]\!]$ the clause for higher-order inputs with the clause $\mathcal{F}[\![a\langle Tr_m\rangle]\!] = a\langle m\rangle$.

Lemma 5.1.5 (operational correspondence for $\mathcal{F}$ on strong transitions) *Let $P \in THO\pi$ and μ a basic action. Then*

1. *$P \xrightarrow{\mu} P'$ implies $\mathcal{F}[\![P]\!] \xrightarrow{\mathcal{F}[\![\mu]\!]} \mathcal{F}[\![P']\!]$.*