Bisimulation for higher-order process calculi[*]

Davide Sangiorgi
INRIA Sophia Antipolis
F-06902 Sophia Antipolis, France.
Email: davide@cma.cma.fr.

Abstract

A *higher-order process calculus* is a calculus for communicating systems which contains higher-order constructs like communication of terms. We analyse the notion of *bisimulation* in these calculi. We argue that both the standard definition of bisimulation (i.e., the one for CCS and related calculi), as well as *higher-order bisimulation* [AGR88, Bou89, Tho90] are in general unsatisfactory, because of their over-discrimination.

We propose and study a new form of bisimulation for such calculi, called *context bisimulation*, which yields a more satisfactory discriminanting power. A drawback of context bisimulation is the heavy use of universal quantification in its definition, which is hard to handle in practice. To resolve this difficulty we introduce *triggered bisimulation* and *normal bisimulation*, and we prove that they both coincide with context bisimulation. In the proof, we exploit the *factorisation theorem*: When comparing the behaviour of two processes, it allows us to "isolate" subcomponents which might give differences, so that the analysis can be concentrated on them.

---

[*] This is a revised and extended version of the homonymous short paper appeared in the Proceedings of the IFIP Working Conference on Programming Concepts, Methods and Calculi (PROCOMET'94, North Holland, 1994) and which was based on results in the author's PhD thesis. The main modifications have been made on the presentation of the technical material, with the purpose of having simpler definitions and proofs.

This work has been supported by the ESPRIT BRA Project 6454 CONFER.

1 Introduction

Recently, various process calculi have been proposed which allow us to describe mobile systems, i.e. concurrent systems whose communication topology may change dynamically. We can categorise these calculi into *first-order calculi* like π-calculus [MPW92], in which only *names* (i.e., ports, or channels) can be communicated, and *higher-order calculi* like CHOCS [Tho90], γ-calculus [Bou89], Higher-Order π-calculus [San92], in which *agents* (i.e., terms of the language) can be communicated. Higher-order calculi are formally closer to the λ-calculus, whose basic computational step — β-reduction — involves term instantiation. In this paper we analyse the notion of *bisimulation* in higher-order calculi, and propose a new form of bisimulation for them.

Bisimulation was originally introduced by Milner and Park [Mil80, Par81] for CCS-like process calculi, in which mobility is not explicitly present, and since then it has become a fundamental concept in the theory of concurrency. In mobility-free calculi, bisimulation is defined on top of a labeled transition system, which describes the operational behaviour of processes, by imposing the following circular requirement: Two processes are *bisimilar* if any action by one of them can be matched by an equal action from the other in such a way that the resulting derivatives are again bisimilar. Note that two matching actions must be syntactically *identical*. This condition is generally too strong in calculi with mobility. For instance, in calculi with name-passing it does not respect alpha-conversion on names [MPW92]. But in higher-order calculi the damage goes well beyond alpha-conversion. We illustrate the kind of problems which arise, using a simple process-passing calculus described by the following grammar (roughly, the language we shall use in the paper):

$$P ::= \overline{a}.\, \langle P_1 \rangle P_2 \quad \Big| \quad a.\,(X)\,P \quad \Big| \quad P_1 \mid P_2 \quad \Big| \quad \boldsymbol{\nu}\, a\, P \quad \Big| \quad X \quad \Big| \quad !\,P \quad \Big| \quad \mathbf{0}$$

This language is similar to Thomsen's Plain CHOCS [Tho93], and is a second-order fragment of the Higher-Order π-calculus [San92]. Informally, process $\overline{a}.\,\langle P_1 \rangle P_2$ can perform an output action at name a emitting process P_1 and then continues as a process P_2. Process $a.\,(X)\,P$ can receive a process at name a, say Q, and then continues as $P\{Q/X\}$. Symbol X represents a process variable, $P_1 \mid P_2$ is parallel composition of two processes P_1 and P_2, and $\mathbf{0}$ denotes inaction. A replication $!\,P$ stands, intuitively, for an infinite number of copies of P in parallel. Finally, $\boldsymbol{\nu}\, a\, P$ is the restriction operator, which makes name a local (i.e., private) to process P, and therefore different from all other names. Restriction is a static binder, as the 'λ' of the λ-calculus. We shall write $\overline{a}.P$ and $a.P$, abbreviating output and input prefixes, respectively, when the process received or emitted is not important.

In the above calculus, the definition of bisimulation used for CCS or π-calculus breaks basic algebraic laws, such as the commutativity of parallel composition. For instance, in general processes $\overline{a}.\,\langle P \mid Q \rangle \mathbf{0}$ and $\overline{a}.\,\langle Q \mid P \rangle \mathbf{0}$ are distinguishable, since the actions they perform may have syntactically different object parts, namely $P \mid Q$ and $Q \mid P$.

The approach taken by Thomsen [Tho90], following earlier ideas by Astesiano et al. [AGR88] and Boudol [Bou89], is to require *bisimilarity* rather than *identity* of the processes emitted in a higher-order output action. This form of bisimulation, called *higher-order bisimulation*, seems troublesome when restriction is a static binder, as in our setting. (By contrast, higher-order bisimulation appears to work well in calculi using dynamic binding as in [AGR88, AGR92, Bou89] or in the calculus CHOCS [Tho90]; we discuss the issue of dynamic and static binding for the restriction construct in the concluding section; in this paper we *only* deal with static binding.) For instance, take

$$P \overset{\mathrm{def}}{=} \overline{a}.\,\langle \mathbf{0} \rangle \mathbf{0}\,, \qquad Q \overset{\mathrm{def}}{=} \boldsymbol{\nu}\, m\, (\overline{a}.\,\langle m.\mathbf{0} \rangle \mathbf{0})\,. \tag{1}$$

Processes P and Q differ in the value carried by $\overline{a}$, which is $\mathbf{0}$ in the former, and $m.\mathbf{0}$ in the latter. Moreover, since $\boldsymbol{\nu}$ is a static binder, the output by Q causes the extrusion of name m, i.e., the scope of the restriction $\boldsymbol{\nu}\, m$ is enlarged to embrace the recipient of $m.\mathbf{0}$. To see an example, the interaction between Q and $a.\,(X)\,R$ is (by alpha-conversion we can assume that m does not occur in R):

$$\big(a.\,(X)\,R\big) \mid Q = \big(a.\,(X)\,R\big) \mid \boldsymbol{\nu}\, m\, (\overline{a}.\,\langle m.\mathbf{0} \rangle \mathbf{0}) \xrightarrow{\tau} \boldsymbol{\nu}\, m\, (R\{m.\mathbf{0}/X\} \mid \mathbf{0})\,.$$

In the derivative, since m is restricted and does not occur in R, process $m.\mathbf{0}$ will never find a partner to communicate with. It is therefore a deadlocked process, and as such semantically the *same* as $\mathbf{0}$. Hence the above derivative is undistinguishable from the derivative of an interaction between $a.\,(X)\,R$ and P, namely

$$\big(a.\,(X)\,R\big) \mid P = \big(a.\,(X)\,R\big) \mid \overline{a}.\,\langle \mathbf{0} \rangle \mathbf{0} \xrightarrow{\tau} R\{\mathbf{0}/X\} \mid \mathbf{0}\,.$$

Indeed, in any context P and Q give rise to the same interactions and, accordingly, should be considered equivalent. Unfortunately, they are *not* higher-order bisimilar. Higher-order bisimulation "forgets" restrictions which are extruded in an output, like the restriction on m in Q. For instance, P and Q are distinguished because the processes they transmit, namely $\mathbf{0}$ and $m.\mathbf{0}$, are not equivalent.

One could think of adjusting the previous example by imposing a different treatment of the extruded name m, thus comparing $\mathbf{0}$ with $\boldsymbol{\nu}\, m\, (m.\mathbf{0})$ rather than $m.\mathbf{0}$. But this approach is certainly wrong and can be disastrous in other situations. For instance, if T is a deadlocked process with m free in it, like $\boldsymbol{\nu}\, n\, (n.m.\mathbf{0})$, then this choice equates processes

$$\boldsymbol{\nu}\, m\, (\overline{a}.\,\langle m.R \rangle \overline{m}.\mathbf{0}) \qquad \text{and} \qquad \boldsymbol{\nu}\, m\, (\overline{a}.\,\langle T \rangle \overline{m}.\mathbf{0}) \tag{2}$$

which by contrast have completely different possibilities of interactions (the former can communicate at m with the recipient of $m.R$ and thus activate a copy of R). This treatment of extruded names in higher-order bisimulation would also yield the law

$$\boldsymbol{\nu}\, m\, (\overline{a}.\,\langle P \rangle Q) = \overline{a}.\,\langle \boldsymbol{\nu}\, m\, P \rangle Q$$

Yet in the first process all copies of P activated by its recipient share the name m, whereas in the second one, the name m is private to each copy.

Higher-order bisimulation appears over-discriminanting even if the restriction operator is omitted. Consider, for instance,

$$P_1 \overset{\text{def}}{=} \overline{a}.\,\langle \mathbf{0} \rangle \,!\, m.\mathbf{0} \qquad P_2 \overset{\text{def}}{=} \overline{a}.\,\langle m.\mathbf{0} \rangle \,!\, m.\mathbf{0}\,. \tag{3}$$

They are not equated according to higher-order bisimulation. This is unsatisfactory because the replication $!\,m.\mathbf{0}$ covers the difference between $\mathbf{0}$ and $m.\mathbf{0}$, regardless of how many copies of them are used.

The above counter-intuitive equalities of higher-order bisimulation are due to the fact that, in an output, the object part (i.e., the process emitted) and the continuation are examined separately. Above all, this separation prevents a satisfactory treatment of the channels private to the two. For the reader familiar with non-interleaving process algebras, this might recall the problems of *distributed bisimulation* [CH89] in presence of restriction. One might then try to follow the solution for the latter proposed by Boudol et al. in [BCHK94] and based on the introduction of *localities.* The idea is to keep the two components to analyse together but assign them different localities, 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 be private names and communications between them. An inconvenience of this approach is that it requires an extension of the syntax of the language.

Instead, our choice has been to avoid the separation between object part and continuation of an output action by explicitly taking into account the *context* in which the emitted process is supposed to go. The resulting bisimulation, called *context bisimulation*, can be given an elegant formulation using the syntactic constructs of *abstraction* and *concretion*, borrowed from [Mil91, Hen93]. Abstractions and concretions are expressions of the form $(X)\,Q$ and $\boldsymbol{\nu}\,\widetilde{z}\,\langle R \rangle Q$, respectively. With these, input and output transitions can be written in the form

$$P \xrightarrow{a} (X)\,Q \qquad \text{and} \qquad P \xrightarrow{\overline{a}} \boldsymbol{\nu}\,\widetilde{z}\,\langle R \rangle Q,$$

the former meaning "P can receive a process at a, say R, and continue as $Q\{R/X\}$", and the latter meaning "by extruding names $\widetilde{z}$, process P can emit R at a and evolve to Q". A pseudo application "$\bullet$" between an abstraction $(X)\,P$ and a concretion $\boldsymbol{\nu}\,\widetilde{z}\,\langle R \rangle Q$ can be defined thus:

$$((X)\,P) \bullet (\boldsymbol{\nu}\,\widetilde{z}\,\langle R \rangle Q) \overset{\text{def}}{=} \boldsymbol{\nu}\,\widetilde{z}\,(P\{R/X\}|Q)$$

(with possible renaming of $\widetilde{z}$ to avoid capture of names in P). Using abstractions and concretions, we introduce the notion of context bisimulation in the following way. Let C and D be concretions, F and G be abstractions, $\approx_{\mathrm{Ct}}$ be context bisimulation, and the weak arrow $P \overset{\overline{a}}{\Longrightarrow} D$ denote abstraction from silent steps; then the bisimilarity clause on the outputs of two processes P and Q is:

$$\begin{gathered} \text{whenever } P \xrightarrow{\overline{a}} C, \text{ there exists } D \text{ s.t. } Q \overset{\overline{a}}{\Longrightarrow} D \\ \text{and } F \bullet C \approx_{\mathrm{Ct}} F \bullet D, \text{ for all abstractions } F. \end{gathered} \tag{4}$$

(here, F plays the role of a possible recipient of the process emitted by P and Q). We impose the symmetric requirement on inputs:

$$\text{whenever } P \xrightarrow{a} F, \text{ there exists } G \text{ s.t. } Q \overset{a}{\Longrightarrow} G \qquad (5)$$
$$\text{and } C \bullet F \approx_{\mathrm{Ct}} C \bullet G, \text{ for all concretions } C.$$

This definition has similarities with Abramsky's definition of applicative bisimulation for the λ-calculus [Abr89]. Context bisimulation equates the processes in (1) and (3), and distinguishes those in (2), as we wished to do. A drawback of context bisimulation is the universal quantifications over abstractions in (4) and over concretions in (5), which can make it hard, in practice, to use this equivalence. We shall therefore look for simpler characterisations, which do not require universal quantifications. Our best candidate for this will be *normal bisimulation*: It shows that it is possible to reason fairly efficiently in a higher-order calculus, notwithstanding its sophisticated transitional semantics.

A crucial role in our study is played by *triggers* and by the *factorisation theorem.* A trigger is an elementary process whose only functionality is to activate a copy of another process. We shall use triggers to perform process transformations which make the treatment of the constructs of higher-order processes easier. The most important of such transformations is indeed the factorisation theorem, which states that, by means of triggers, a subprocess of a given process can be factorised out.

To prove that context bisimulation and normal bisimulation coincide, we shall go through an intermediate characterisation, namely *triggered bisimulation.* This is a bisimilarity relation with extremely simple clauses on input and output actions. However, it is only defined on the subclass of *triggered processes*, roughly, processes in which triggers only can be exchanged in communications. Triggered processes will represent a sort of "normal form" for processes. The factorisation theorem will be used to transform any process into a triggered agent.

Recently, a number of works which use bisimulations similar to context bisimulation have appeared. Amadio [Ama93] uses a similar notion to study the encoding of Plain CHOCS into π-calculus. Amadio and Dams [AD95] propose an extension of Hennessy and Milner's modal logics which characterises this behavioural equivalence (for the strong case). Hansen and Kleist [HK94] analyse a form of asynchronous higher-order calculus and show that *late, early* and *open* (this terminology is borrowed from the π-calculus literature) variants of (strong) context bisimulation coincide. Pitts and Ross [PR96] prove a characterisation of context bisimilarity as an *evaluation* bisimilarity; the latter is defined on top of an evaluation relation of processes to normal forms, that is input- or output-prefixed expressions, and hence does not talk of "τ-moves". Other related work is discussed in the concluding section.

Structure of the paper. In Section 2 we give the formal syntax and the transitional semantics of the higher-order calculus used as test bed in the paper. In Section 3 we put forward context bisimulation, as an adjustment of higher-order bisimulation for eliminating its drawbacks discussed above. In Section 4 we introduce triggers and we prove the factorisation theorem. In Section 5 we define a mapping $\mathcal{T}$ which transforms every process into a triggered agent. In Sections 6 and 7, by exploiting $\mathcal{T}$, we are able to prove simpler characterisations of context bisimulation, namely triggered bisimulation and normal bisimulation; the former is even simpler than the latter, but is only defined on the subclass of triggered processes. In Section 8, we discuss the extension of the theory to a richer calculus, namely the Higher-Order π-calculus [San92]; this is a ω-order calculus, whereas the calculus of Section 2 or Plain CHOCS are second-order calculi. Section 9 concludes the paper and mentions directions for future research.

2 The language and its transition semantics

The calculus we use is similar to Thomsen's Plain CHOCS [Tho93]; hence, processes can be passed around. The main differences from Plain CHOCS are: The use of abstractions and concretions to represent input and output prefixes; the presence of *first-order names*, i.e., names which carry nothing. These are opposed to *higher-order names*, i.e., names used to exchange processes. For the theory we shall develop, the presence of first-order names is not necessary, but makes the presentation of various results easier.

Thus, let $\mathcal{F}$ be the countable infinite set of first-order names, and $\mathcal{H}$ the countable infinite set of higher-order names. Then, $\overline{\mathcal{F}} \stackrel{\mathrm{def}}{=} \{\overline{m} \ : \ m \in \mathcal{F}\}$, $\overline{\mathcal{H}} \stackrel{\mathrm{def}}{=} \{\overline{a} \ : \ a \in \mathcal{H}\}$, $\mathcal{N} \stackrel{\mathrm{def}}{=} \mathcal{F} \cup \mathcal{H}$ and $\overline{\mathcal{N}} \stackrel{\mathrm{def}}{=} \overline{\mathcal{F}} \cup \overline{\mathcal{H}}$. The special symbol τ, which does not occur in $\mathcal{N}$ or $\overline{\mathcal{N}}$, denotes a silent step. We let μ range over $\mathcal{N} \cup \overline{\mathcal{N}} \cup \{\tau\}$, and ℓ range over $\mathcal{F} \cup \overline{\mathcal{F}} \cup \{\tau\}$ (these are the CCS-like actions). By convention, if $\ell \neq \tau$ then $\overline{\overline{\ell}} = \ell$. We use symbols x, y, z for names in $\mathcal{N}$; symbols m, n for names in $\mathcal{F}$; and symbols a, b, c for names in $\mathcal{H}$. We also assume a countable infinite set of *process variables*, ranged over by X, Y, Z.

Definition 2.1 *The syntactic categories of our language and their grammar are:*

$$
\begin{array}{rlll}
\textit{Processes} & P & ::= & a.\,F \;\big|\; \overline{a}.\,C \;\big|\; \ell.\,P \;\big|\; P_1 \,|\, P_2 \;\big|\; \boldsymbol{\nu}\, x\, P \;\big|\; X \;\big| \\
 & & & !\,P \;\big|\; \mathbf{0} \\
\textit{Abstractions} & F & ::= & (X)\,P \\
\textit{Concretions} & C & ::= & \boldsymbol{\nu}\, x\, C \;\big|\; \langle P_1 \rangle P_2 \\
\textit{Agents} & A & ::= & P \;\big|\; F \;\big|\; C
\end{array}
$$

P, Q, R and T will range over processes, F and G over abstractions, C and D over concretions, A and B over agents.

An abstraction $(X)\,P$ binds all free occurrences of X in P; similarly a restriction $\boldsymbol{\nu}\, x\, P$ binds the free occurrences of x in P. These binders give rise in the expected way to the definitions of alpha conversion, free variables and free names of an agent A, respectively $\mathrm{fv}(A)$ and $\mathrm{fn}(A)$; alpha conversion relates expressions which are syntactically identical modulo renaming of bound names and bound variables.

An agent is *closed* if it has no free variable; it is *open* if it may have free variables. $\mathcal{A}g$ is the set of all agents (i.e., the set of open agents); $\mathcal{A}g^{\mathrm{o}}$ is the set of all closed agents.

Similarly, $\mathcal{P}r$ and $\mathcal{P}r^{\mathrm{o}}$ are the sets of all processes and of all closed processes, respectively. $P\{\widetilde{Q}/\widetilde{X}\}$ denotes the componentwise and simultaneous substitution of variables $\widetilde{X}$ with processes $\widetilde{Q}$ (where it is assumed that the members of $\widetilde{X}$ are distinct). We often abbreviate $\boldsymbol{\nu}\, x_1 \ldots \boldsymbol{\nu}\, x_n\, A$ as $\boldsymbol{\nu}\, x_1, \ldots, x_n\, A$. In a statement, a name is called *fresh* if it is different from any other name occurring in agents of the statement. In a prefix $\mu.A$ we call μ the *subject*; thus if the prefix is an input $x.A$ then x is in *input subject* position, and similarly if it is an output $\overline{x}.A$ then x is in *output subject* position.

We shall only admit *standard concretions*, i.e., expressions $\boldsymbol{\nu}\, \widetilde{x}\, \langle P_1 \rangle P_2$ where names in $\widetilde{x}$ are pairwise distinct and $\{\widetilde{x}\} \subseteq \mathrm{fn}(P_1)$. Indeed, the remaining concretions have little significance: In $\boldsymbol{\nu}\, \widetilde{x}\, \langle Q \rangle P$, by alpha conversion, names $\widetilde{x}$ can be assumed to be distinct; and if $x \notin \mathrm{fn}(Q) \cup \{\widetilde{x}\}$ then in $\boldsymbol{\nu}\, x, \widetilde{x}\, \langle Q \rangle P$ name x can be pushed inwards, resulting in the standard concretion $\boldsymbol{\nu}\, \widetilde{x}\, \langle Q \rangle (\boldsymbol{\nu}\, x\, P)$. In the following, we therefore assume that if $x \notin \mathrm{fn}(Q) \cup \{\widetilde{x}\}$, then $\boldsymbol{\nu}\, x, \widetilde{x}\, \langle Q \rangle P$ denotes $\boldsymbol{\nu}\, \widetilde{x}\, \langle Q \rangle (\boldsymbol{\nu}\, x\, P)$.

We wish to extend restriction to operate on abstractions, and (a form of) parallel composition to operate on abstractions and concretions.

Let $F = (X)\, Q$:

— if $X \notin \mathrm{fv}(P)$, then $F \mid P$ denotes $(X)\, (Q \mid P)$
and $P \mid F$ denotes $(X)\, (P \mid Q)$,

— $\boldsymbol{\nu}\, x\, F$ denotes $(X)\, \boldsymbol{\nu}\, x\, Q$.

Let $C = \boldsymbol{\nu}\, \widetilde{x}\, \langle Q \rangle R$:

— if $\{\widetilde{x}\} \cap \mathrm{fn}(P) = \emptyset$, then $C \mid P$ denotes $\boldsymbol{\nu}\, \widetilde{x}\, \langle Q \rangle (R \mid P)$
and $P \mid C$ denotes $\boldsymbol{\nu}\, \widetilde{x}\, \langle Q \rangle (P \mid R)$.

We now present the operational semantics of the calculus. First, we define an operation "$\bullet$" of pseudo-application between an abstraction $F = (X)\, P$ and a concretion $C = \boldsymbol{\nu}\, \widetilde{x}\, \langle Q \rangle R$. By alpha conversion, we can assume that $\{\widetilde{x}\} \cap \mathrm{fn}(F) = \emptyset$ and then we set

$$C \bullet F \stackrel{\mathrm{def}}{=} \boldsymbol{\nu}\, \widetilde{x}\, (R \mid P\{Q/X\})$$

and, symmetrically,

$$F \bullet C \stackrel{\mathrm{def}}{=} \boldsymbol{\nu}\, \widetilde{x}\, (P\{Q/X\} \mid R)\, .$$

Similarly, we define an operation of application "$\circ$" between an abstraction $F = (X)\, P$ and a process Q thus:

$$F \circ Q \stackrel{\mathrm{def}}{=} P\{Q/X\}$$

The operational semantics of the calculus is reported in Table 1. We have omitted the symmetric of the parallelism and communication rules. There are these forms of judgements:

$$\begin{array}{lll} P \xrightarrow{a} F & & \text{(higher-order input transition at port } a\text{)} \\ P \xrightarrow{\overline{a}} C & & \text{(higher-order output transition at port } a\text{)} \\ P \xrightarrow{\ell} Q & & \text{(first-order transition)} \end{array}$$

Alpha	P and Q alpha convertible		
	$Q \xrightarrow{\mu} A$	implies	$P \xrightarrow{\mu} A$
Prefix	$\mu.\, A \xrightarrow{\mu} A$		
Parallelism	$P_1 \xrightarrow{\mu} A$	implies	$P_1 \mid P_2 \xrightarrow{\mu} A \mid P_2$
First-order communication	$P_1 \xrightarrow{m} P_1'$		
	$P_2 \xrightarrow{\overline{m}} P_2'$	implies	$P_1 \mid P_2 \xrightarrow{\tau} P_1' \mid P_2'$
Higher-order communication	$P_1 \xrightarrow{a} F$		
	$P_2 \xrightarrow{\overline{a}} C$	implies	$P_1 \mid P_2 \xrightarrow{\tau} F \bullet C$
Restriction	$P \xrightarrow{\mu} A,\ \mu \notin \{x, \overline{x}\}$	implies	$\boldsymbol{\nu}\, x\, P \xrightarrow{\mu} \boldsymbol{\nu}\, x\, A$
Replication	$P \mid {!P} \xrightarrow{\mu} A$	implies	${!P} \xrightarrow{\mu} A$

Table 1: The transition system

where $P, Q, F, C \in \mathcal{A}g$. In turn, a first-order transition can be a first-order input (if $\ell \in \mathcal{F}$), a first-order output (if $\ell \in \overline{\mathcal{F}}$), or an interaction (if $\ell = \tau$).

In the remainder of the paper we work up to alpha conversion; thus "=" denotes syntactic equality up to alpha conversion. We shall normally put enough brackets in the expressions so to avoid precedence ambiguities among the operators. However, to reduce the number of brackets, in a few places we shall assume the following syntactic rules: Substitutions and metanotations "$\bullet$" and "$\circ$" have the highest syntactic precedence; the abstraction and concretion constructs the lowest; parallel composition has weaker precedence than the other process constructs. For instance, $\langle P \rangle\, !\, m.R|Q$ stands for $\langle P \rangle((\, !\, m.R)|Q)$, and $F \bullet C | Q\{R/X\}$ stands for $(F \bullet C) \mid (Q\{R/X\})$.

If $\mathcal{R}$ is a relation on processes, we write $P \,\mathcal{R}\, Q$ to that $(P, Q) \in \mathcal{R}$. Moreover, $\mathcal{R}_1 \mathcal{R}_2$ is the composition of the two relations $\mathcal{R}_1$ and $\mathcal{R}_2$.

3 Context bisimulation

We shall study behavioural equivalences based on bisimulation for the language of the previous section. To overcome the counter-intuitive equalities of higher-order bisimulation examined in Section 1, we propose the *context bisimilarity* relation below.

Definition 3.1 (strong context bisimulation) *A relation* $\mathcal{R} \subseteq \mathcal{P}r^o \times \mathcal{P}r^o$ *is a* strong context simulation *if* $P \,\mathcal{R}\, Q$ *implies*

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

2. *whenever* $P \xrightarrow{a} F$, *there exists* G *s.t.* $Q \xrightarrow{a} G$ *and* $C \bullet F \mathrel{\mathcal{R}} C \bullet G$, *for all closed concretions* C.
3. *whenever* $P \xrightarrow{\overline{a}} C$, *there exists* D *s.t.* $Q \xrightarrow{\overline{a}} D$ *and* $F \bullet C \mathrel{\mathcal{R}} F \bullet D$, *for all closed abstractions* F.

A relation $\mathcal{R}$ *is a* strong context bisimulation, *in symbols* $\sim_{Ct}$*-bisimulation, if* $\mathcal{R}$ *and* $\mathcal{R}^{-1}$ *are strong context simulations. We say that* P, Q *are* strongly context bisimilar, *in symbols* $P \sim_{Ct} Q$, *if* $P \mathrel{\mathcal{R}} Q$, *for some* $\sim_{Ct}$*-bisimulation* $\mathcal{R}$.

Relation $\sim_{\mathrm{Ct}}$ is generalised to abstractions, concretions and open agents as expected.

Definition 3.2

1. *For closed abstractions* F_1 *and* F_2, *we set* $F_1 \sim_{Ct} F_2$ *if* $C \bullet F_1 \sim_{Ct} C \bullet F_2$ *for all closed concretions* C.
2. *For closed concretions* C_1 *and* C_2, *we set* $C_1 \sim_{Ct} C_2$ *if* $F \bullet C_1 \sim_{Ct} F \bullet C_2$ *for all closed abstractions* F.
3. *For open agents* A_1 *and* A_2 *with* $fv(A_1, A_2) \subseteq \{\widetilde{X}\}$, *set* $A_1 \sim_{Ct} A_2$ *if* $A_1\{\widetilde{P}/\widetilde{X}\} \sim_{Ct} A_2\{\widetilde{P}/\widetilde{X}\}$ *for all closed processes* $\widetilde{P}$.

Lemma 3.3 $\sim_{Ct}$ *is an equivalence relation.* ☐

Example 3.4 *Let*

$$P_1 \stackrel{def}{=} \overline{a}.\langle \mathbf{0} \rangle \mathbf{0}$$
$$P_2 \stackrel{def}{=} \boldsymbol{\nu}\, m\, (\overline{a}.\langle m.\mathbf{0} \rangle \mathbf{0})$$

We argued in Section 1 that P_1 *and* P_2 *should be equated. We show that, indeed,* $P_1 \sim_{Ct} P_2$. *The set* $\mathcal{R}$ *of all pairs of the form*

$$(\boldsymbol{\nu}\, m\, (R\{m.\mathbf{0}/X\}), R\{\mathbf{0}/X\}) \text{ with } m \notin fn(R)$$

contains the pair (P_1, P_2) *and is a* $\sim_{Ct}$*-bisimulation. We sketch the argument.*

Since m *is restricted, process* $\boldsymbol{\nu}\, m\, (R\{m.\mathbf{0}/X\})$ *cannot perform a visible action at* m; *moreover, since* m *occurs in* $R\{m.\mathbf{0}/X\}$ *only in input position, no interaction along* m *can occur. Therefore, any transition for* $\boldsymbol{\nu}\, m\, (R\{m.\mathbf{0}/X\})$ *is of the form*

$$\boldsymbol{\nu}\, m\, (R\{m.\mathbf{0}/X\}) \xrightarrow{\mu} \boldsymbol{\nu}\, m\, (A\{m.\mathbf{0}/X\})$$

and we also have

$$R\{\mathbf{0}/X\} \xrightarrow{\mu} A\{\mathbf{0}/X\}.$$

Suppose A *is a concretion, say* $\boldsymbol{\nu}\, \widetilde{x}\, \langle Q_1 \rangle Q_2$ *with* X *free in* Q_1 *and* Q_2. *Then, for all abstractions* $F \stackrel{def}{=} (Y)\, Q_3$ *we have:*

$$\begin{aligned} F \bullet \boldsymbol{\nu}\, m\, (A\{m.\mathbf{0}/X\}) &= \boldsymbol{\nu}\, m\, \big(Q_3\{Q_1/Y\}\{m.\mathbf{0}/X\} \mid Q_2\{m.\mathbf{0}/X\}\big) \\ &= \boldsymbol{\nu}\, m\, \big((Q_3\{Q_1/Y\} \mid Q_2)\{m.\mathbf{0}/X\}\big) \stackrel{def}{=} Q_4, \\ F \bullet A\{\mathbf{0}/X\} &= (Q_3\{Q_1/Y\} \mid Q_2)\{\mathbf{0}/X\} \stackrel{def}{=} Q_5 \end{aligned}$$

Thus (Q_4, Q_5) *is in* $\mathcal{R}$.

Some simple laws for $\sim_{\mathrm{Ct}}$:

Lemma 3.5

1. $P_1 \mid P_2 \sim_{Ct} P_2 \mid P_1$;
2. $P_1 \mid (P_2 \mid P_3) \sim_{Ct} (P_1 \mid P_2) \mid P_3$;
3. $P \mid \mathbf{0} \sim_{Ct} P$;
4. $\boldsymbol{\nu}\, x\, \boldsymbol{\nu}\, y\, A \sim_{Ct} \boldsymbol{\nu}\, y\, \boldsymbol{\nu}\, x\, A$;
5. *if* $x \notin fn(P_2)$*, then* $(\boldsymbol{\nu}\, x\, P_1) \mid P_2 \sim_{Ct} \boldsymbol{\nu}\, x\, (P_1 \mid P_2)$;
6. $!\, P \sim_{Ct} P \mid !\, P$. ☐

We shall use the up-to technique below for establishing bisimilarity results.

Definition 3.6 *A relation* $\mathcal{R} \subseteq \mathcal{P}r^o \times \mathcal{P}r^o$ *is a* strong context simulation up-to $\sim_{\mathrm{Ct}}$ *if* $P \mathcal{R} Q$ *implies*

1. *whenever* $P \xrightarrow{\ell} P'$*, there exists* Q' *s.t.* $Q \xrightarrow{\ell} Q'$ *and* $P' \sim_{Ct} \mathcal{R} \sim_{Ct} Q'$,
2. *whenever* $P \xrightarrow{a} F$*, there exists* G *s.t.* $Q \xrightarrow{a} G$ *and* $C \bullet F \sim_{Ct} \mathcal{R} \sim_{Ct} C \bullet G$*, for all concretions* C.
3. *whenever* $P \xrightarrow{\overline{a}} C$*, there exists* D *s.t.* $Q \xrightarrow{\overline{a}} D$ *and* $F \bullet C \sim_{Ct} \mathcal{R} \sim_{Ct} F \bullet D$*, for all abstractions* F.

Lemma 3.7 *If* $\mathcal{R}$ *is a strong context simulation up-to* $\sim_{Ct}$ *then* $\mathcal{R} \subseteq \sim_{Ct}$.

PROOF: The same argument of the analogous result for CCS bisimilarity [Mil80]: Use a diagram-chasing argument to show that $\sim_{\mathrm{Ct}} \mathcal{R} \sim_{\mathrm{Ct}}$ is a strong context bisimulation. ☐

3.1 Congruence properties of context bisimulation

Context bisimulation is preserved by all operators of the language. As far as proofs are concerned, the most difficult one is congruence for object constructor (i.e., $P \sim_{\mathrm{Ct}} Q$ implies $a.\langle P\rangle R \sim_{\mathrm{Ct}} a.\langle Q\rangle R$); this case is specific of the higher-order setting to which our language belongs. To derive this, we need to prove a congruence result w.r.t. substitutions:

Lemma 3.8 *Let* $P_1, P_2, A \in \mathcal{A}g$*; then* $P_1 \sim_{Ct} P_2$ *implies* $A\{P_1/X\} \sim_{Ct} A\{P_2/X\}$.

PROOF: See Appendix A. ☐

Theorem 3.9 (congruence of $\sim_{\mathbf{Ct}}$) *Let* $P_i, A_i \in \mathcal{A}g$.

1. $A_1 \sim_{Ct} A_2$ *implies:* $\boldsymbol{\nu}\, x\, A_1 \sim_{Ct} \boldsymbol{\nu}\, x\, A_2$,
   $\mu.\, A_1 \sim_{Ct} \mu.\, A_2$.

2. $P_1 \sim_{Ct} P_2$ *implies:* $P_1 \mid Q \sim_{Ct} P_2 \mid Q$,
$!\, P_1 \sim_{Ct} !\, P_2$,
$\langle P_1 \rangle Q \sim_{Ct} \langle P_2 \rangle Q$,
$\langle Q \rangle P_1 \sim_{Ct} \langle Q \rangle P_2$,
$(X)\, P_1 \sim_{Ct} (X)\, P_2$.

Proof: Each clause can either be derived from Lemma 3.8, or is straightforward on its own. $\square$

Corollary 3.10

1. *$F_1 \sim_{Ct} F_2$ and $C_1 \sim_{Ct} C_2$ imply $F_1 \bullet C_1 \sim_{Ct} F_2 \bullet C_2$;*
2. *$F_1 \sim_{Ct} F_2$ and $P_1 \sim_{Ct} P_2$ imply $F_1 \circ P_1 \sim_{Ct} F_2 \circ P_2$.* $\square$

3.2 Weak context bisimulation

To introduce weak context bisimulation, we first define weak transitions, where τ steps are absorbed. Thus $\Longrightarrow$ is the reflexive and transitive closure of $\xrightarrow{\tau}$, and $P \overset{\mu}{\Longrightarrow} A$ holds if there is P' s.t. $P \Longrightarrow P'$ and $P' \xrightarrow{\mu} A$. Finally, $P \overset{\widehat{\mu}}{\Longrightarrow} A$ is $P \Longrightarrow A$ if $\mu = \tau$ and $P \overset{\mu}{\Longrightarrow} A$ otherwise.

Definition 3.11 (weak context bisimulation) *A relation $\mathcal{R} \subseteq \mathcal{P}r^o \times \mathcal{P}r^o$ is a* weak context simulation *if $P \mathrel{\mathcal{R}} Q$ implies*

1. *whenever $P \xrightarrow{\ell} P'$, there exists Q' s.t. $Q \overset{\widehat{\ell}}{\Longrightarrow} Q'$ and $P' \mathrel{\mathcal{R}} Q'$;*
2. *whenever $P \xrightarrow{a} F$, there exists G s.t. $Q \overset{a}{\Longrightarrow} G$ and $C \bullet F \mathrel{\mathcal{R}} C \bullet G$, for all closed concretions C;*
3. *whenever $P \xrightarrow{\overline{a}} C$, there exists D s.t. $Q \overset{\overline{a}}{\Longrightarrow} D$ and $F \bullet C \mathrel{\mathcal{R}} F \bullet D$, for all closed abstractions F.*

A relation $\mathcal{R}$ is a weak context bisimulation, *in symbols $\approx_{Ct}$-bisimulation, if $\mathcal{R}$ and $\mathcal{R}^{-1}$ are weak context simulations. We say that P, Q are* weakly context bisimilar, *in symbols $P \approx_{Ct} Q$, if $P \mathrel{\mathcal{R}} Q$, for some $\approx_{Ct}$-bisimulation $\mathcal{R}$.*

Remark 3.12 (delay and late bisimulations) In the formulation of weak context bisimulation above, agents are immediately tested after a visible action (recall that $\overset{\mu}{\Longrightarrow}$ stands for $\Longrightarrow \xrightarrow{\mu}$). This treatment of weak transitions is characteristic of *delay bisimulation*, studied in CCS-like languages in [Wei89]. Moreover, in the input and output clauses of Definition 3.11 the existential quantifier precedes the universal one. This is characteristic of *late* bisimulation [MPW92], as opposed to *early* bisimulation, in which the order of the quantifiers is exchanged. We shall show in Section 7.1 that, for weak context bisimulation, the late and early versions coincide.

We think that the "late delay schema" fits well the machinery of abstractions and concretions adopted. First, it well describes the complementarity between abstractions and concretions. Secondly, it seems natural to require that abstractions and concretions do not evolve on their own, but only after meeting a complementary agent. Another compelling motivation for formulating a late bisimulation as a delay bisimulation is that otherwise the resulting relation might not be an equivalence relation; this, for instance, happens in the π-calculus [San96b]. ☐

Weak context bisimulation is extended to concretions, abstractions and open agents in the same way as the strong equivalence (Definition 3.2); thus, for concretions C_1 and C_2, we have $C_1 \approx_{\mathrm{Ct}} C_2$ if $C_1 \bullet F \approx_{\mathrm{Ct}} C_2 \bullet F$ for all closed abstractions F. The congruence results for $\sim_{\mathrm{Ct}}$ in Lemma 3.3 and Theorem 3.9 can be extended to $\approx_{\mathrm{Ct}}$, with a completely analogous proof (see also Remark A.5 in Appendix A).

Theorem 3.13 *$\approx_{Ct}$ is an equivalence relation and is preserved by all operators of the language.* ☐

We shall use the following up to technique to establish weak-context bisimilarity results.

Definition 3.14 *A relation $\mathcal{R} \subseteq \mathcal{P}r^o \times \mathcal{P}r^o$ is a* weak context simulation up-to $\approx_{\mathrm{Ct}}$ *if $P \,\mathcal{R}\, Q$ implies*

1. *whenever $P \overset{\ell}{\Longrightarrow} P'$, there exists Q' s.t. $Q \overset{\widehat{\ell}}{\Longrightarrow} Q'$ and $P' \approx_{Ct} \mathcal{R} \approx_{Ct} Q'$,*
2. *whenever $P \overset{a}{\Longrightarrow} F$, there exists G s.t. $Q \overset{a}{\Longrightarrow} G$ and $C \bullet F \approx_{Ct} \mathcal{R} \approx_{Ct} C \bullet G$, for all closed concretions C.*
3. *whenever $P \overset{\overline{a}}{\Longrightarrow} C$, there exists D s.t. $Q \overset{\overline{a}}{\Longrightarrow} D$ and $F \bullet C \approx_{Ct} \mathcal{R} \approx_{Ct} F \bullet D$, for all closed abstractions F.*

Lemma 3.15 *If $\mathcal{R}$ is a weak context simulation up-to $\approx_{Ct}$ then $\mathcal{R} \subseteq \approx_{Ct}$.*

PROOF: By showing that $\approx_{\mathrm{Ct}} \mathcal{R} \approx_{\mathrm{Ct}}$ is a $\approx_{\mathrm{Ct}}$-bisimulation. ☐

Weak context bisimulation is the relation we are mostly interested in, since it abstracts away from silent steps of processes. We shall look for characterisations of this behavioural equivalence which do not use the heavy universal quantifications in its input clause (quantification on concretions) and in its output clause (quantification on abstractions). We shall use strong context bisimulation as an auxiliary relation.

4 The factorisation theorem

The main result of this section is the *factorisation theorem.* It allows us to factorise out certain subagents of a given agent. Thus, a complex process can be decomposed into the parallel composition of simpler processes.

The assertion of the factorisation theorem uses a special kind of agents called *triggers* and the metanotation $A\,\{z := B\}$. We introduce this metanotation, and prove some algebraic properties about it, in Section 4.1, and we present triggers and the factorisation theorem in Section 4.2.

4.1 Distributivity properties of private replications

We write $A\,\{z := B\}$ as an abbreviation for $\boldsymbol{\nu}\, z\,(A \mid\, !\, z.\, B)$, under the assumption that z may occur free in A and B only in output subject position.

Intuitively, in $A\,\{z := B\}$, agent B represents a "local environment" for A and z is a "pointer" that allows A to access this local environment; alternatively, we can think of B as a resource with owner A and z as a trigger with which a copy of the resource may be activated.

Remember that z is restricted, and hence *not free*, in $A\,\{z := B\}$, 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 $\{z := B\}$ behaves just like a substitution in $A\{z := B\}$. For instance, if B is an abstraction and z a higher-order name, then $A\{z := B\}$ behaves as the agent obtained from A by substituting '$B \circ R \mid Q$' for any subexpressions '$\overline{z}.\, \langle R \rangle Q$'. This because, given the side condition on the use of z in A and B, the only possible effect of the prefix $\overline{z}.\, \langle R \rangle Q$ is to trigger a copy of B with argument R. For example, a little thinking should convince the reader that if a is not free in R, R', P, F, then the visible behaviour of $\big(\overline{a}.\, \langle R \rangle \overline{a}.\, \langle R' \rangle P\big)\,\{a := F\}$ is the same as that of $F \circ R \mid F \circ R' \mid P$.

The results in this and in the next section show that, indeed, the metanotation $\{z := B\}$ enjoys algebraic properties similar to those of substitutions. We shall sometimes call $\{z := B\}$ an *implicit substitution.* We give $\{z := B\}$ the same precedence as substitutions; thus $Q \mid P\,\{z := B\}$ stands for $Q \mid (P\,\{z := B\})$.

Theorem 4.2 shows that $\{z := B\}$ distributes over all operators of the language. To prove this, we first need to show that $\{z := B\}$ distributes over process substitutions (in the same way as in Section 3.1, to prove the congruence of $\sim_{\mathrm{Ct}}$, we first needed the result on substitutions).

Lemma 4.1 *Let $A, P, B \in \mathcal{A}g$. Suppose that $z \notin fn(B)$ and that z occurs free in A and P only in output subject position. Then*

$$A\{P/X\}\,\{z := B\} \sim_{Ct} A\,\{z := B\}\{P\,\{z := B\}/X\}.$$

Proof: See Appendix B. □

Theorem 4.2 (distributivity of $\{z := B\}$) *Suppose that $z \notin fn(B)$ and that z occurs free in A, P, Q and C only in output subject position. Then the following results on open agents hold:*

1. $(\boldsymbol{\nu}\, x\, A)\,\{z := B\} \sim_{Ct} \boldsymbol{\nu}\, x\,(A\,\{z := B\})$, *if* $x \notin fn(B) \cup \{z\}$.

2. $(P \mid Q)\ \{z := B\} \sim_{Ct} P\ \{z := B\} \mid Q\ \{z := B\}$.

3. $(\,!P)\ \{z := B\} \sim_{Ct}\ !\,(P\ \{z := B\})$.

4. $(\mu.\, A)\ \{z := B\} \sim_{Ct} \mu.\,(A\ \{z := B\})$, *if* $\mu \neq z$.

5. $(\overline{z}.\, C)\ \{z := B\} \sim_{Ct} (\tau.\, C \bullet B)\ \{z := B\}$ *(here z is a higher-order name and B is an abstraction).*

6. $(\overline{z}.\, P)\{z := B\} \sim_{Ct} (\tau.\,(P|B))\{z := B\}$ *(here z is a first-order name and B a process).*

7. $\mathbf{0}\ \{z := B\} \sim_{Ct} \mathbf{0}$.

8. $(\langle Q\rangle P)\ \{z := B\} \sim_{Ct} \langle Q\ \{z := B\}\rangle(P\ \{z := B\})$.

PROOF: Cases 2, 3 and 8 can be derived from Lemma 4.1, cases 1, 4, 5, 6 and 7 are straightforward. ☐

Remark 4.3 In the two results above, the requirement $z \notin \mathrm{fn}(B)$ could be weakened to "z free in B only in output subject position", at the price of some more work in the proofs. This extra power is not necessary for our purposes. ☐

4.2 Triggers

A trigger is a process of the form $\overline{m}.\mathbf{0}$; we write Tr_m to denote a trigger whose free name is m.

The assertion of the factorisation theorem, that we are going to give, reads as follows. Take an agent A, and suppose we can extract an expression Q from certain components of A, in form of an explicit substitution; i.e., for some agent A' and variable Y, it holds that $A = A'\{Q/Y\}$. Using a trigger Tr_m, for some fresh m, the explicit substitution can be transformed into an implicit one, obtaining $(A'\{\mathrm{Tr}_m/Y\})\ \{m := Q\}$. By contrast with $A\{Q/Y\}$, in $(A'\{\mathrm{Tr}_m/Y\})\ \{m := Q\}$ each copy of Q is activated only when it is needed using the trigger m.

Example 4.4 *If* $P \overset{def}{=} Q \mid \overline{a}.\,\langle Q\rangle R$, *then* $P = (X \mid \overline{a}.\,\langle X\rangle R)\{Q/X\}$ *and, applying the factorisation theorem (and assuming m fresh),*

$$\begin{array}{rcl} P & = & (X \mid \overline{a}.\,\langle X\rangle R)\{\mathit{Tr}_m/X\}\ \{m := Q\} \\ & = & (\mathit{Tr}_m \mid \overline{a}.\,\langle \mathit{Tr}_m\rangle R)\ \{m := Q\} \end{array}$$

Lemma 4.5 *For each $P \in \mathcal{P}r$, it holds that $\tau.P \approx_{Ct} P$.* ☐

We derive the factorisation theorem from Lemma 4.6, which shows us that the effect of using a trigger is precisely to add a τ-action on the head of the replaced expression.

Lemma 4.6 *For every $A, R \in \mathcal{A}g$ with $m \notin fn(A, R)$, it holds that*

$$A\{\tau.\, R/X\} \sim_{Ct} A\{\mathit{Tr}_m/X\}\ \{m := R\}.$$

PROOF: By induction on the structure of A. The basic case is when $A = Y$ and is immediate using Theorem 4.2. For the inductive cases, as an example we show the case of parallel composition, since the other cases are similar or simpler. We have:

$$\begin{array}{rcl} (P_1 \mid P_2)\{\tau.\, R/X\} & = & \\ P_1\{\tau.\, R/X\} \mid P_2\{\tau.\, R/X\} & \sim_{\mathrm{Ct}} & \text{(induction twice)} \\ P_1\{\mathrm{Tr}_m/X\}\,\{m := R\} \mid P_2\{\mathrm{Tr}_m/X\}\,\{m := R\} & \sim_{\mathrm{Ct}} & \text{(Theorem 4.2(2))} \\ \big(P_1\{\mathrm{Tr}_m/X\} \mid P_2\{\mathrm{Tr}_m/X\}\big)\,\{m := R\} & = & \\ (P_1 \mid P_2)\{\mathrm{Tr}_m/X\}\,\{m := R\} & & \end{array}$$

☐

Theorem 4.7 (factorisation theorem) *For every $A, Q \in \mathcal{A}g$ with $m \notin fn(A, Q)$, it holds that*

$$A\{Q/X\} \approx_{Ct} A\{Tr_m/X\}\,\{m := Q\}.$$

PROOF: From Lemma 4.6, $A\{\mathrm{Tr}_m/X\}\,\{m := Q\} \sim_{\mathrm{Ct}} A\{\tau.\, Q/X\}$. Since, by Lemma 4.5, $\tau.\, Q \approx_{\mathrm{Ct}} Q$ and $\approx_{\mathrm{Ct}}$ is a congruence relation, we can infer $A\{\tau.\, Q/X\} \approx_{\mathrm{Ct}} A\{Q/X\}$. ☐

In the remainder of the paper, for weak context bisimulation or other weak equivalences, the adjective "weak" might be omitted.

5 Triggered agents

In this section we introduce the class of *triggered agent.* They represent a sort of "normal form" for the agents of the calculus. Most importantly, there is a very simple characterisation of context bisimulation on triggered agents, called *triggered bisimulation.* We shall exploit the factorisation theorem to transform every agent into a triggered agent. The transformation allows us to use the simpler theory of triggered agents to reason about the set of all agents. The distinguishing feature of triggered agents is that every communication among them is the exchange of a trigger.

Definition 5.1 (triggered agents) *The grammar for triggered agents is obtained from that of ordinary agents in Definition 2.1 by replacing the productions for concretion with the production*

$$\textit{Concretions} \quad C \quad ::= \quad \boldsymbol{\nu}\,\widetilde{x}\,(\langle Tr_m\rangle P_1)\,\{m := P_2\} \qquad \textit{with } m \notin fn(P_1, P_2) \cup \{\widetilde{x}\}.$$

In other words, we place the additional requirement that all concretions should be in the above "triggered" form. Recall that $\boldsymbol{\nu}\,\widetilde{x}\,(\langle \mathrm{Tr}_m\rangle P_1)\,\{m := P_2\}$ is an abbreviation for $\boldsymbol{\nu}\,m\,\langle \mathrm{Tr}_m\rangle \boldsymbol{\nu}\,\widetilde{x}\,(P_1 \mid\, !m.\, P_2)$. We write $\mathcal{TA}g$ and $\mathcal{TA}g^{\mathrm{o}}$ for the classes of triggered agents and of closed triggered agents, respectively. $\mathcal{TP}r$ and $\mathcal{TP}r^{\mathrm{o}}$ are the subclasses of triggered processes and of closed triggered processes, respectively.

We give a mapping $\mathcal{T}$ which transforms every agent A into the triggered agent $\mathcal{T}[\![A]\!]$. The mapping is defined inductively on the structure of A; for concretions we have

$$\mathcal{T}[\![\langle Q\rangle P]\!] = ((\langle \mathrm{Tr}_m\rangle \mathcal{T}[\![P]\!])\,\{m := \mathcal{T}[\![Q]\!]\} \qquad \text{where } m \text{ is a fresh name}$$

$\mathcal{T}$ acts as a homomorphism on all remaining constructs; thus on parallel composition and higher-order output we have:

$$\mathcal{T}[\![Q \mid P]\!] = \mathcal{T}[\![Q]\!] \mid \mathcal{T}[\![P]\!] \qquad\qquad \mathcal{T}[\![\overline{a}.\langle C\rangle]\!] = \overline{a}.\langle \mathcal{T}[\![C]\!]\rangle$$

For instance, we have:

$$\begin{aligned}
\mathcal{T}[\![\big(a.\,(X)\,X\big) \mid \overline{a}.\,\boldsymbol{\nu}\,x\,\langle Q\rangle P]\!] &= \big(a.\,(X)\,X\big) \mid \overline{a}.\,\boldsymbol{\nu}\,x\,\big(((\langle \mathrm{Tr}_m\rangle \mathcal{T}[\![P]\!])\,\{m := \mathcal{T}[\![Q]\!]\}\big) \\
&= \big(a.\,(X)\,X\big) \mid \overline{a}.\,\boldsymbol{\nu}\,m\,\langle \mathrm{Tr}_m\rangle \boldsymbol{\nu}\,x\,\big(\mathcal{T}[\![P]\!] \mid\, !\,m.\,\mathcal{T}[\![Q]\!]\big)
\end{aligned}$$

Theorem 5.2 (correctness of $\mathcal{T}$) *For each $A \in \mathcal{A}g$:*

1. $\mathcal{T}[\![A]\!]$ *is a triggered agent;*
2. $\mathcal{T}[\![A]\!] \approx_{Ct} A$.

Proof: Assertion (1) is straightforward. Assertion (2) can be proved by induction on the structure of A. The only case in which $\mathcal{T}$ does not act as a homomorphism is when A is a concretion of the form $\langle R\rangle P$. If $m \notin \mathrm{fn}(A)$, then we have:

$$\begin{aligned}
\langle R\rangle P \quad & \\
\approx_{\mathrm{Ct}} \quad & \text{(Theorem 4.7)} \\
((\langle \mathrm{Tr}_m\rangle P)\ \{m := R\} \quad & \\
\approx_{\mathrm{Ct}} \quad & \text{(induction twice)} \\
((\langle \mathrm{Tr}_m\rangle \mathcal{T}[\![P]\!])\ \{m := \mathcal{T}[\![R]\!]\} \quad &= \quad \mathcal{T}[\![A]\!]
\end{aligned}$$

□

It is useful to see the operational correspondence between P and $\mathcal{T}[\![P]\!]$. Transformation $\mathcal{T}$ may expand the number of silent steps in a process. But the behaviour is otherwise the same. The expansion is due to the fact that if in P a process Q is transmitted and used n times then, in $\mathcal{T}[\![P]\!]$, n interactions are required to activate the copies of Q, as the following example shows.

Example 5.3 *Let* $P \overset{def}{=} \big(\overline{a}.\,\langle Q\rangle R\big) \mid a.\,(X)\,(X \mid X)$. *Then*

$$P \xrightarrow{\tau} R \mid Q \mid Q \overset{def}{=} P'$$

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

$$\begin{aligned}
\mathcal{T}[\![P]\!] \quad &= \quad \overline{a}.\,((\langle \mathit{Tr}_m\rangle \mathcal{T}[\![R]\!])\,\{m := \mathcal{T}[\![Q]\!]\} \mid a.\,(X)\,(X \mid X) \\
\xrightarrow{\tau} \sim_{Ct} \quad & \quad \big(\mathcal{T}[\![R]\!] \mid \mathit{Tr}_m \mid \mathit{Tr}_m\big)\,\{m := \mathcal{T}[\![Q]\!]\} \\
= \quad & \quad \big(\mathcal{T}[\![R]\!] \mid \overline{m}.\mathbf{0} \mid \overline{m}.\mathbf{0}\big)\,\{m := \mathcal{T}[\![Q]\!]\} \\
\xrightarrow{\tau} \xrightarrow{\tau} \sim_{Ct} \quad & \quad \big(\mathcal{T}[\![R]\!] \mid \mathcal{T}[\![Q]\!] \mid \mathcal{T}[\![Q]\!]\big)\,\{m := \mathcal{T}[\![Q]\!]\} \\
\sim_{Ct} \quad & \quad \mathcal{T}[\![R]\!] \mid \mathcal{T}[\![Q]\!] \mid \mathcal{T}[\![Q]\!] \\
= \quad & \quad \mathcal{T}[\![P']\!]
\end{aligned}$$

Lemma 5.4 *For all F and C, it holds that $\mathcal{T}[\![F \bullet C]\!] \approx_{Ct} \mathcal{T}[\![F]\!] \bullet \mathcal{T}[\![C]\!]$.*

Proof: Let $F \stackrel{\mathrm{def}}{=} (X)\,P$ and $C \stackrel{\mathrm{def}}{=} \boldsymbol{\nu}\,\widetilde{x}\,\langle Q\rangle R$. We have:

$$\begin{array}{rcl}
\mathcal{T}[\![F]\!] \bullet \mathcal{T}[\![C]\!] & = & \big((X)\,\mathcal{T}[\![P]\!]\big) \bullet \boldsymbol{\nu}\,\widetilde{x}\,\big((\langle \mathrm{Tr}_m\rangle \mathcal{T}[\![R]\!])\,\{m := \mathcal{T}[\![Q]\!]\}\big) \\
 & \sim_{\mathrm{Ct}} & \boldsymbol{\nu}\,\widetilde{x}\,\big((\mathcal{T}[\![P]\!]\{\mathrm{Tr}_m/X\} \mid \mathcal{T}[\![R]\!])\,\{m := \mathcal{T}[\![Q]\!]\}\big) \\
 & \approx_{\mathrm{Ct}} & \boldsymbol{\nu}\,\widetilde{x}\,\big(\mathcal{T}[\![P]\!]\{\mathcal{T}[\![Q]\!]/X\} \mid \mathcal{T}[\![R]\!]\big) \\
 & = & \mathcal{T}[\![\boldsymbol{\nu}\,\widetilde{x}\,(P\{Q/X\} \mid R)]\!] \\
 & = & \mathcal{T}[\![F \bullet C]\!]
\end{array}$$

where the use of $\approx_{\mathrm{Ct}}$ is due to Theorem 4.7. ☐

Lemma 5.5 (operational correspondence for $\mathcal{T}$ on strong transitions)

1. (a) *If $P \xrightarrow{\mu} A$ and $\mu \neq \tau$, then $\mathcal{T}[\![P]\!] \xrightarrow{\mu} \sim_{Ct} \mathcal{T}[\![A]\!]$;*

   (b) *if $P \xrightarrow{\tau} P'$, then $\mathcal{T}[\![P]\!] \xrightarrow{\tau} \approx_{Ct} \mathcal{T}[\![P']\!]$.*

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

   (a) *If $\mathcal{T}[\![P]\!] \xrightarrow{\mu} A'$ and $\mu \neq \tau$, then there is A s.t. $P \xrightarrow{\mu} A$ and $\mathcal{T}[\![A]\!] \sim_{Ct} A'$;*

   (b) *if $\mathcal{T}[\![P]\!] \xrightarrow{\tau} P''$, then there is P' s.t. $P \xrightarrow{\tau} P'$ and $P'' \approx_{Ct} \mathcal{T}[\![P']\!]$.*

Proof: By transition induction. We only consider the rule for parallel composition for assertion 1(a) in the case when μ is a higher-order output, and the higher-order communication rule for assertion 1(b).

Suppose $P_1 \mid P_2 \xrightarrow{\overline{a}} C \mid P_2$, for some C s.t. $P_1 \xrightarrow{\overline{a}} C$. By induction,

$$\mathcal{T}[\![P_1]\!] \xrightarrow{\overline{a}} \sim_{\mathrm{Ct}} \mathcal{T}[\![C]\!]\,.$$

Hence

$$\mathcal{T}[\![P_1 \mid P_2]\!] = \mathcal{T}[\![P_1]\!] \mid \mathcal{T}[\![P_2]\!] \xrightarrow{\overline{a}} \sim_{\mathrm{Ct}} \mathcal{T}[\![C]\!] \mid \mathcal{T}[\![P_2]\!]\,.$$

Let $C = \boldsymbol{\nu}\,\widetilde{x}\,\langle Q\rangle R$. Then $\mathcal{T}[\![C]\!] = \boldsymbol{\nu}\,m\,\langle \mathrm{Tr}_m\rangle \boldsymbol{\nu}\,\widetilde{x}\,(R \mid\, !\,m.Q)$ and

$$\mathcal{T}[\![C]\!] \mid \mathcal{T}[\![P_2]\!] = \boldsymbol{\nu}\,m\,\langle \mathrm{Tr}_m\rangle \boldsymbol{\nu}\,\widetilde{x}\,(R \mid\, !\,m.Q) \mid \mathcal{T}[\![P_2]\!]$$

and, assuming $\widetilde{x} \cap \mathrm{fn}(\mathcal{T}[\![P_2]\!]) = \emptyset$, by Lemma 3.5,

$$\begin{array}{rl}
\sim_{\mathrm{Ct}} & \boldsymbol{\nu}\,m\,\langle \mathrm{Tr}_m\rangle \boldsymbol{\nu}\,\widetilde{x}\,(R \mid \mathcal{T}[\![P_2]\!] \mid\, !\,m.Q) \\
= & \mathcal{T}[\![\boldsymbol{\nu}\,\widetilde{x}\,\langle Q\rangle (R \mid P_2)]\!] = \mathcal{T}[\![C \mid P_2]\!]
\end{array}$$

Now, the communication rule. Thus, suppose $P_1 \mid P_2 \xrightarrow{\tau} F \bullet C$, for some F and C s.t. $P_1 \xrightarrow{a} F$ and $P_2 \xrightarrow{\overline{a}} C$. By induction, $\mathcal{T}[\![P_1]\!] \xrightarrow{a} \sim_{\mathrm{Ct}} \mathcal{T}[\![F]\!]$ and $\mathcal{T}[\![P_2]\!] \xrightarrow{\overline{a}} \sim_{\mathrm{Ct}} \mathcal{T}[\![C]\!]$. Hence $\mathcal{T}[\![P_1 \mid P_2]\!] = \mathcal{T}[\![P_1]\!] \mid \mathcal{T}[\![P_2]\!] \xrightarrow{\tau} \sim_{\mathrm{Ct}} \mathcal{T}[\![F]\!] \bullet \mathcal{T}[\![C]\!]$ and, by Lemma 5.4, $\mathcal{T}[\![F]\!] \bullet \mathcal{T}[\![C]\!] \approx_{\mathrm{Ct}} \mathcal{T}[\![F \bullet C]\!]$. ☐

6 Triggered bisimulation

We show that the class $\mathcal{TP}r^{o}$ of triggered processes is amenable to an analysis in which only triggers are exchanged with an external observer. We start by showing that the class $\mathcal{TP}r^{o}$ is closed with respect to the production of such actions (Lemma 6.2) and then we define a bisimulation in which only this kind of actions is taken into account.

Lemma 6.1 *If* $A \in \mathcal{TA}g$*, then for every* Tr_m *we have* $A\{Tr_m/X\} \in \mathcal{TA}g$*.* ☐

Lemma 6.2 *Suppose* $P \in \mathcal{TP}r^{o}$*. It holds that:*

1. *If* $P \xrightarrow{\ell} P'$*, then* $P' \in \mathcal{TP}r^{o}$*;*
2. *if* $P \xrightarrow{a} F$*, then for all* m*,* $F \circ Tr_m \in \mathcal{TP}r^{o}$*;*
3. *if* $P \xrightarrow{\overline{a}} C$*, then then there is* $P' \in \mathcal{TP}r^{o}$ *s.t.* $C = \boldsymbol{\nu}\, m\, \langle Tr_m \rangle P'$ *and, moreover, for some* $\widetilde{x}, P''$ *and* R *with* $m \notin fn(P'', R) \cup \{\widetilde{x}\}$*, we have* $P' \sim_{Ct} \boldsymbol{\nu}\, \widetilde{x}\, (P'' \mid\, !m.\, R)$*.*

Proof: A simple transition induction; for assertion (2), use Lemma 6.1. ☐

Definition 6.3 (triggered bisimulation) *A relation* $\mathcal{R} \subseteq \mathcal{TP}r^{o} \times \mathcal{TP}r^{o}$ *is a* triggered simulation *if* $P\ \mathcal{R}\ Q$ *implies, for* $m \notin fn(P, Q)$*:*

1. *whenever* $P \xrightarrow{\ell} P'$*, there exists* Q' *s.t.* $Q \overset{\widehat{\ell}}{\Longrightarrow} Q'$ *and* $P'\ \mathcal{R}\ Q'$*,*
2. *whenever* $P \xrightarrow{a} F$*, there exists* G *s.t.* $Q \overset{a}{\Longrightarrow} G$ *and* $F \circ Tr_m\ \mathcal{R}\ G \circ Tr_m$*,*
3. *whenever* $P \xrightarrow{\overline{a}} \boldsymbol{\nu}\, m\, \langle Tr_m \rangle P'$*, there exists* Q' *s.t.* $Q \overset{\overline{a}}{\Longrightarrow} \boldsymbol{\nu}\, m\, \langle Tr_m \rangle Q'$ *and* $P'\ \mathcal{R}\ Q'$*.*

$\mathcal{R}$ *is a* triggered bisimulation*, in symbols* $\approx_{Tr}$*-bisimulation, if* $\mathcal{R}$ *and* $\mathcal{R}^{-1}$ *are triggered simulations. We say that* P *and* Q *are* triggered bisimilar*, in symbols* $P \approx_{Tr} Q$*, if* $P\ \mathcal{R}\ Q$*, for some* $\approx_{Tr}$*-bisimulation* $\mathcal{R}$ *.*

Definition 6.3 is consistent since $P', Q', F \circ \mathrm{Tr}_m$ and $G \circ \mathrm{Tr}_m$ are triggered processes by Lemma 6.2. Note that for clauses (2) and (3) it is enough to take *some* $m \notin \mathrm{fn}(P, Q)$. Any other choice of m — with $m \notin \mathrm{fn}(P, Q)$ — 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 6.3 compared to those in the definition of context bisimulation: In clause (3) of Definition 6.3, no universal quantification and no check whatsoever on the agents emitted in the output is necessary; similarly, in clause (2) a single (fresh) trigger is used, whereas in context bisimulation every agent which can possibly be received in the input is taken into account.

Relation $\approx_{\mathrm{Tr}}$ is extended to $\mathcal{TA}g$ accordingly; in particular, in the case of open agents free variables are instantiated with fresh triggers only.

Definition 6.4

1. *For closed triggered abstractions* F_1 *and* F_2*, we set* $F_1 \approx_{Tr} F_2$ *if* $F_1 \circ Tr_m \approx_{Tr} F_2 \circ Tr_m$*, for some fresh name* m*.*
2. *For closed triggered concretions* $\boldsymbol{\nu}\, m\, \langle Tr_m \rangle P$ *and* $\boldsymbol{\nu}\, m\, \langle Tr_m \rangle Q$*, we set* $\boldsymbol{\nu}\, m\, \langle Tr_m \rangle P \approx_{Tr} \boldsymbol{\nu}\, m\, \langle Tr_m \rangle Q$ *if* $P \approx_{Tr} Q$*.*
3. *For open triggered agents* A_1 *and* A_2 *with* $fv(A_1, A_2) = \{X_1, \ldots, X_n\}$*, if* $\{m_1, \ldots, m_n\}$ *is a set of distinct fresh names, then we set* $A_1 \approx_{Tr} A_2$ *if* $A_1\{Tr_{m_1}/X_1, \ldots, Tr_{m_n}/X_n\} \approx_{Tr} A_2\{Tr_{m_1}/X_1, \ldots, Tr_{m_n}/X_n\}$*.*

Definition 6.5 *A relation* $\mathcal{R} \subseteq \mathcal{TPr}^o \times \mathcal{TPr}^o$ *is a triggered bisimulation up-to* $\approx_{Tr}$ *if* $P \mathrel{\mathcal{R}} Q$ *implies, for* $m \notin fn(P, Q)$*:*

1. *whenever* $P \overset{\widehat{\ell}}{\Longrightarrow} P'$*, there exists* Q' *s.t.* $Q \overset{\widehat{\ell}}{\Longrightarrow} Q'$ *and* $P' \approx_{Tr} \mathcal{R} \approx_{Tr} Q'$*,*
2. *whenever* $P \overset{a}{\Longrightarrow} F$*, there exists* G *s.t.* $Q \overset{a}{\Longrightarrow} G$ *and* $F \circ Tr_m \approx_{Tr} \mathcal{R} \approx_{Tr} G \circ Tr_m$*,*
3. *whenever* $P \overset{\overline{a}}{\Longrightarrow} \boldsymbol{\nu}\, m\, \langle Tr_m \rangle P'$*, there exists* Q' *s.t.* $Q \overset{\overline{a}}{\Longrightarrow} \boldsymbol{\nu}\, m\, \langle Tr_m \rangle Q'$ *and* $P' \approx_{Tr} \mathcal{R} \approx_{Tr} Q'$*.*

Lemma 6.6 *If* $\mathcal{R}$ *is a triggered bisimulation up-to* $\approx_{Tr}$*, then* $\mathcal{R} \subseteq \approx_{Tr}$*.* ☐

6.1 Weak triggered and context bisimulations coincide

Next we prove that, on triggered processes, $\approx_{\mathrm{Tr}}$ and $\approx_{\mathrm{Ct}}$ coincide. First, we need some properties of $\approx_{\mathrm{Tr}}$.

Lemma 6.7 $P \approx_{Tr} Q$ *implies:*

1. $\boldsymbol{\nu}\, x\, P \approx_{Tr} \boldsymbol{\nu}\, x\, Q$*;*
2. $P \mid R \approx_{Tr} Q \mid R$*.*

Proof: Following Definition 6.4, it suffices to prove the result on closed processes. We only examine (2). For this, we prove that

$$\mathcal{R} = \{(\boldsymbol{\nu}\, \widetilde{y}\, (P_1 \mid R), \boldsymbol{\nu}\, \widetilde{y}\, (P_2 \mid R)) \;:\; P_1, P_2, R \in \mathcal{TPr}^o \text{ and } P_1 \approx_{\mathrm{Tr}} P_2\}$$

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

$$P_1 \xrightarrow{\overline{a}} \boldsymbol{\nu}\, m\, \langle \mathrm{Tr}_m \rangle P_1', \qquad R \xrightarrow{a} F, \qquad \mu = \tau\,, \qquad \text{and} \qquad Q_1 = \boldsymbol{\nu}\, \widetilde{y}, m\, (P_1' \mid F \circ \mathrm{Tr}_m).$$

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

$$P_2 \overset{\overline{a}}{\Longrightarrow} \boldsymbol{\nu}\, m\, \langle \mathrm{Tr}_m \rangle P_2' \qquad \text{with} \qquad P_1' \approx_{\mathrm{Tr}} P_2', \qquad \text{and} \qquad \boldsymbol{\nu}\, \widetilde{y}\, (P_2 \mid R) \overset{\tau}{\Longrightarrow} \boldsymbol{\nu}\, \widetilde{y}, m\, (P_2' \mid F \circ \mathrm{Tr}_m).$$

This is enough, because by Lemma 6.2 we have $P_1', P_2', F \circ \mathrm{Tr}_m \in \mathcal{TP}r^{\mathrm{o}}$. All other cases are similar. $\square$

Lemma 6.8 *Let $P, Q \in \mathcal{TP}r^o$; then $P \approx_{Ct} Q$ implies $P \approx_{Tr} Q$.*

Proof: We prove that

$$\mathcal{R} \ = \{(P_1, P_2) \ : \ P_1, P_2 \in \mathcal{TP}r^{\mathrm{o}} \text{ and } P_1 \approx_{\mathrm{Ct}} P_2\}$$

is a $\approx_{\mathrm{Tr}}$-bisimulation. Let $(P_1, P_2) \in \mathcal{R}$ and suppose that $P_1 \xrightarrow{\mu} A$. The only non-trivial case is when μ is a higher-order output, so we only consider this case. Thus suppose $\mu = \overline{a}$; by Lemma 6.2, then $A = \boldsymbol{\nu}\, m\, \langle \mathrm{Tr}_m \rangle P_1'$. By definition of $\approx_{\mathrm{Ct}}$ and Lemma 6.2, there exists P_2' s.t. $P_2 \overset{\overline{a}}{\Longrightarrow} \boldsymbol{\nu}\, m\, \langle \mathrm{Tr}_m \rangle P_2'$ and for every G (by alpha conversion we can assume that $m \notin \mathrm{fn}(G)$:

$$\boldsymbol{\nu}\, m\, (G \circ \mathrm{Tr}_m \mid P_1') \approx_{\mathrm{Ct}} \boldsymbol{\nu}\, m\, (G \circ \mathrm{Tr}_m \mid P_2')\,. \tag{6}$$

In order to close the bisimulation we have to show that $P_1' \approx_{\mathrm{Ct}} P_2'$. Lemma 6.2 tells us that there exist $\widetilde{y_1}, \widetilde{y_2}, R_1, R_2, P_1'', P_2''$ s.t.

$$P_1' \sim_{\mathrm{Ct}} \boldsymbol{\nu}\, \widetilde{y_1}\, (P_1'' \mid\, !\, m.\, R_1)\,, \quad P_2' \sim_{\mathrm{Ct}} \boldsymbol{\nu}\, \widetilde{y_2}\, (P_2'' \mid\, !\, m.\, R_2) \tag{7}$$

where $m \notin \mathrm{fn}(R_1, R_2, P_1'', P_2'') \cup \{\widetilde{y_1}\} \cup \{\widetilde{y_2}\}$. Suppose m' is a fresh name. Now, changing m for m' does not affect the equivalence among processes, i.e. we have

$$\begin{array}{lcll} \boldsymbol{\nu}\, \widetilde{y_1}\, (P_1'' \mid\, !\, m.\, R_1) & \approx_{\mathrm{Ct}} & \boldsymbol{\nu}\, \widetilde{y_2}\, (P_2'' \mid\, !\, m.\, R_2) & \text{iff} \\ \boldsymbol{\nu}\, \widetilde{y_1}\, (P_1'' \mid\, !\, m'.\, R_1) & \approx_{\mathrm{Ct}} & \boldsymbol{\nu}\, \widetilde{y_2}\, (P_2'' \mid\, !\, m'.\, R_2) & \end{array}$$

Therefore we get $P_1' \approx_{\mathrm{Ct}} P_2'$ if we can prove the latter equivalence. The advantage with this is that we shall be able to exploit (6). Since m is not free in $\boldsymbol{\nu}\, \widetilde{y_1}\, (P_1'' \mid\, !\, m'.\, R_1)$ and $\boldsymbol{\nu}\, \widetilde{y_2}\, (P_2'' \mid\, !\, m'.\, R_2)$, using the factorisation theorem (4.7) we have

$$\begin{array}{rcl} \boldsymbol{\nu}\, \widetilde{y_1}\, (P_1'' \mid\, !\, m'.\, R_1) & \approx_{\mathrm{Ct}} & \\ \boldsymbol{\nu}\, \widetilde{y_1}\, \big((P_1'' \mid\, !\, m'.\, \mathrm{Tr}_m)\, \{m := R_1\}\big) & \sim_{\mathrm{Ct}} & \\ \boldsymbol{\nu}\, m\, \big(\, !\, m'.\, \mathrm{Tr}_m \mid \boldsymbol{\nu}\, \widetilde{y_1}\, (P_1'' \mid\, !\, m.\, R_1)\big) & \sim_{\mathrm{Ct}} & \text{(by (7))} \\ \boldsymbol{\nu}\, m\, (\, !\, m'.\, \mathrm{Tr}_m \mid P_1') & \overset{\mathrm{def}}{=} & Q_1 \end{array}$$

and similarly,

$$\boldsymbol{\nu}\, \widetilde{y_2}\, \big(P_2'' \mid\, !\, m'.\, R_2\big) \quad \approx_{\mathrm{Ct}} \quad \boldsymbol{\nu}\, m\, (\, !\, m'.\, \mathrm{Tr}_m \mid P_2') \overset{\mathrm{def}}{=} Q_2$$

Now $Q_1 \approx_{\mathrm{Ct}} Q_2$ can be derived from (6), for $G \overset{\mathrm{def}}{=} (X)(\, !\, m'.\, X)$. $\square$

Lemma 6.9 *Let $P, Q \in \mathcal{TP}r^o$; then $P \approx_{Tr} Q$ implies $P \approx_{Ct} Q$.*

Proof: We show that

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

is a $\approx_{\mathrm{Ct}}$-bisimulation up-to $\approx_{\mathrm{Ct}}$. Let $P \mathcal{R} Q$. The most interesting case is to see how higher-order input actions of P are matched by Q, and we consider this case only. Thus, suppose $P_1 \overset{a}{\Longrightarrow} F_1$. Since $P_1 \approx_{\mathrm{Tr}} P_2$, there is F_2 s.t. $P_2 \overset{a}{\Longrightarrow} F_2$ and, if m is fresh,

$$F_1 \circ \mathrm{Tr}_m \approx_{\mathrm{Tr}} F_2 \circ \mathrm{Tr}_m \,. \tag{8}$$

To close the bisimulation, we have to show that for all C,

$$F_1 \bullet C \approx_{\mathrm{Ct}} \mathcal{R} \approx_{\mathrm{Ct}} F_2 \bullet C \tag{9}$$

Let $C = \boldsymbol{\nu}\, \widetilde{y}\, \langle Q \rangle R$. We have

$$\begin{array}{rcl} F_1 \bullet C = \boldsymbol{\nu}\, \widetilde{y}\, (F_1 \circ Q \mid R) & \approx_{\mathrm{Ct}} & \text{(by the factorisation theorem)} \\ \boldsymbol{\nu}\, \widetilde{y}\Big((F_1 \circ \mathrm{Tr}_m)\, \{m := Q\} \mid R\Big) & \approx_{\mathrm{Ct}} & \text{(Theorem 5.2(2))} \\ \boldsymbol{\nu}\, \widetilde{y}((F_1 \circ \mathrm{Tr}_m)\, \{m := \mathcal{T}[\![Q]\!]\} \mid \mathcal{T}[\![R]\!]) & \overset{\mathrm{def}}{=} & P_1' \end{array}$$

and, similarly, $F_2 \bullet C \approx_{\mathrm{Ct}} \boldsymbol{\nu}\, \widetilde{y}((F_2 \circ \mathrm{Tr}_m)\{m := \mathcal{T}[\![Q]\!]\} | \mathcal{T}[\![R]\!]) \overset{\mathrm{def}}{=} P_2'$. We have $P_1', P_2' \in \mathcal{TP}r^{\mathrm{o}}$. From (8), since $\approx_{\mathrm{Tr}}$ is preserved by parallel composition and restriction, we infer $P_1' \approx_{\mathrm{Tr}} P_2'$, thus concluding the proof of (9). ☐

The two previous lemmas give:

Theorem 6.10 *Relations $\approx_{Tr}$ and $\approx_{Ct}$ coincide on closed triggered processes.* ☐

Theorem 6.10 can be extended to open triggered agents. A direct proof is possible, using the factorisation theorem; instead, we shall derive it as a corollary of other results (Corollary 7.6).

7 Normal bisimulation

The mapping to triggered agents is a useful tool for reasoning with higher-order processes. (For instance, in [San92] we used the mapping as an intermediate step to define a compilation from the Higher-Order π-calculus to the π-calculus and to prove its full abstraction.) In this section, we exploit the mapping to derive a characterisation of context bisimulation, called *normal bisimulation*, which does not have universal quantifications in the clauses of its definition. Normal bisimulation is not as simple as triggered bisimulation, but the former is defined on the whole class of agents of the calculus, whereas the latter is only defined on triggered agents.

The name "normal bisimulation" is to indicate that it is obtained by "normalising" the clauses of context bisimulation. Let us present the definition first; then we shall comment on it. In the following, Ab_m denotes the abstraction $(X)\; !\, m. X$.

Definition 7.1 (normal bisimulation) *A relation $\mathcal{R} \subseteq \mathcal{P}r^o \times \mathcal{P}r^o$ is a* normal simulation *if $P \mathcal{R} Q$ implies, for $m \notin fn(P, Q)$:*

1. *whenever* $P \xrightarrow{\ell} P'$*, there exists* Q' *s.t.* $Q \overset{\widehat{\ell}}{\Longrightarrow} Q'$ *and* $P' \,\mathcal{R}\, Q'$*;*
2. *whenever* $P \xrightarrow{a} F$*, there exists* G *s.t.* $Q \overset{a}{\Longrightarrow} G$ *and* $F \circ Tr_m \,\mathcal{R}\, G \circ Tr_m$*;*
3. *whenever* $P \xrightarrow{\overline{a}} C$*, there exists* D *s.t.* $Q \overset{\overline{a}}{\Longrightarrow} D$ *and* $C \bullet Ab_m \,\mathcal{R}\, D \bullet Ab_m$*.*

A relation $\mathcal{R}$ *is a* normal bisimulation*, in symbols* $\approx_{Nr}$*-bisimulation, if* $\mathcal{R}$ *and* $\mathcal{R}^{-1}$ *are normal simulations. We say that* P *and* Q *are* normal bisimilar*, in symbols* $P \approx_{Nr} Q$*, if* $P \,\mathcal{R}\, Q$*, for some* $\approx_{Nr}$*-bisimulation* $\mathcal{R}$*.*

As for triggered bisimulation, we should stress that in clauses (2) and (3) of Definition 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 extension of $\approx_{\mathrm{Nr}}$ to closed abstractions and open agents is defined as for triggered bisimulation, therefore only employing fresh triggers. For closed concretions C and D, following Definition 7.1 we set $C \approx_{\mathrm{Nr}} D$ if $C \bullet \mathrm{Ab}_m \approx_{\mathrm{Nr}} D \bullet \mathrm{Ab}_m$, for some fresh m.

The idea of normal bisimulation comes from the results on the discriminanting power given by triggers and shown in Sections 4 and 6. To test the equivalence between $F \overset{\mathrm{def}}{=} a.\,(X)\,P$ and $G \overset{\mathrm{def}}{=} a.\,(X)\,Q$ we do not have to try *every* applications $F \bullet C$ and $G \bullet C$, but it is enough to verify that P and Q are equivalent when X is instantiated with a *single* (fresh) trigger Tr_m. In fact, by the congruence properties of $\approx_{\mathrm{Ct}}$,

$$F \circ \mathrm{Tr}_m = P\{\mathrm{Tr}_m/X\} \quad \approx_{\mathrm{Ct}} \quad Q\{\mathrm{Tr}_m/X\} = G \circ \mathrm{Tr}_m$$

implies that for any $\widetilde{y}, R$ and T

$$\boldsymbol{\nu}\,\widetilde{y}\,\big((P\{\mathrm{Tr}_m/X\})\,\{m := R\} \mid T\big) \;\approx_{\mathrm{Ct}}\; \boldsymbol{\nu}\,\widetilde{y}\,\big((Q\{\mathrm{Tr}_m/X\})\,\{m := F\} \mid T\big)\,; \tag{10}$$

then, since by the factorisation theorem

$$(P\{\mathrm{Tr}_m/X\})\,\{m := R\} \approx_{\mathrm{Ct}} P\{R/X\} \qquad \text{and} \qquad (Q\{\mathrm{Tr}_m/X\})\,\{m := R\} \approx_{\mathrm{Ct}} Q\{R/X\}\,,$$

from (10), if $C \overset{\mathrm{def}}{=} \boldsymbol{\nu}\,\widetilde{y}\,\langle R\rangle T$, we get

$$F \bullet C = \boldsymbol{\nu}\,\widetilde{y}\,(P\{R/X\} \mid T) \approx_{\mathrm{Ct}} \boldsymbol{\nu}\,\widetilde{y}\,(Q\{R/X\} \mid T) = G \bullet C\,.$$

Similar reasoning can be used to justify the clause of normal bisimulation on output actions, w.r.t. that of context bisimulation. It may be useful however to see why in the requirement of this clause, namely

$$C \bullet \mathrm{Ab}_m \approx_{\mathrm{Nr}} C \bullet \mathrm{Ab}_m$$

the abstraction Ab_m cannot be made simpler without losing discriminanting power. We recall that if $C \overset{\mathrm{def}}{=} \boldsymbol{\nu}\,\widetilde{x}\,\langle P_1\rangle P_2$ and $D \overset{\mathrm{def}}{=} \boldsymbol{\nu}\,\widetilde{y}\,\langle Q_1\rangle Q_2$, then $C \bullet \mathrm{Ab}_m \approx_{\mathrm{Nr}} D \bullet \mathrm{Ab}_m$ means that

$$\boldsymbol{\nu}\,\widetilde{x}\,(P_2 \mid\, !\,m.\,P_1) \approx_{\mathrm{Nr}} \boldsymbol{\nu}\,\widetilde{y}\,(Q_2 \mid\, !\,m.\,Q_1) \tag{11}$$

We show that both the guard on m and the replication are necessary. First of all, we need the guard m on P_1 and Q_1: For, otherwise, if R is a process like $!\,n.\mathbf{0}$, which can perform an unbounded number of identical visible actions, then the processes $\overline{a}.\langle R\rangle\mathbf{0}$ and $\overline{a}.\langle\mathbf{0}\rangle R$ 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 (11). Then let $R \stackrel{\mathrm{def}}{=} n.n.\overline{p}.\mathbf{0}|\overline{n}.\mathbf{0}$ and consider $P \stackrel{\mathrm{def}}{=} \boldsymbol{\nu}\, n\,(\overline{a}.\langle R\rangle\mathbf{0})$ and $Q \stackrel{\mathrm{def}}{=} \overline{a}.\langle\mathbf{0}\rangle\mathbf{0}$. Again, P and Q would become equivalent since $\boldsymbol{\nu}\, n\,(\mathbf{0}\,|\,m.R)$ and $\mathbf{0}\,|\,m.\mathbf{0}$ 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, since then the action c of R can be observed. In fact, only in the *linear* calculus, where in each expression a variable may occur free at most once, the replication in (11) can be avoided.

We now prove formally that $\approx_{\mathrm{Ct}}$ and $\approx_{\mathrm{Nr}}$ coincide. The inclusion $\approx_{\mathrm{Ct}} \subseteq \approx_{\mathrm{Nr}}$ is obvious, since the requirements in the definition of $\approx_{\mathrm{Nr}}$ are a subset of those in the definition of $\approx_{\mathrm{Ct}}$. To prove the opposite inclusion, we use Lemma 7.2 below, which relates weak transitions of P and $\mathcal{T}\llbracket P\rrbracket$. Its proof is obtained using the operational correspondence on strong transitions in Lemma 5.5.

Lemma 7.2

1. *if* $P \stackrel{\mu}{\Longrightarrow} A$, *then* $\mathcal{T}\llbracket P\rrbracket \stackrel{\mu}{\Longrightarrow} \approx_{Tr} \mathcal{T}\llbracket A\rrbracket$;
2. *The converse: If* $\mathcal{T}\llbracket P\rrbracket \stackrel{\mu}{\Longrightarrow} A'$, *then there is* A *s.t.* $P \stackrel{\mu}{\Longrightarrow} A$ *and* $A' \approx_{Tr} \mathcal{T}\llbracket A\rrbracket$. ☐

Lemma 7.3 *For all concretions* C, *if* m *is fresh, then* $\mathcal{T}\llbracket C\rrbracket = \boldsymbol{\nu}\, m\,\langle Tr_m\rangle\mathcal{T}\llbracket C \bullet Ab_m\rrbracket$.

PROOF: Follows from the definition of mapping $\mathcal{T}$, of the application "$\bullet$", and of the abstraction Ab_m. ☐

Theorem 7.4 *Relations* $\approx_{Nr}$ *and* $\approx_{Ct}$ *coincide on* $\mathcal{P}r^o \times \mathcal{P}r^o$.

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

$$\mathcal{R} \;=\{(\mathcal{T}\llbracket P\rrbracket, \mathcal{T}\llbracket Q\rrbracket)\;:\; P \approx_{\mathrm{Nr}} Q\}$$

is a $\approx_{\mathrm{Tr}}$-bisimulation up-to $\approx_{\mathrm{Tr}}$ (Definition 6.5). This is enough, because on triggered processes $\approx_{\mathrm{Tr}}$ and $\approx_{\mathrm{Ct}}$ coincide (Theorem 6.10), and $\mathcal{T}$ respects $\approx_{\mathrm{Ct}}$ (Theorem 5.2(2)): Hence from $\mathcal{T}\llbracket P\rrbracket \approx_{\mathrm{Tr}} \mathcal{T}\llbracket Q\rrbracket$, we can infer $P \approx_{\mathrm{Ct}} Q$. Let $(\mathcal{T}\llbracket P\rrbracket, \mathcal{T}\llbracket Q\rrbracket) \in \mathcal{R}$; the only non-trivial case is to show how higher-order output actions of $\mathcal{T}\llbracket P\rrbracket$ are matched by $\mathcal{T}\llbracket Q\rrbracket$.

Suppose $\mathcal{T}\llbracket P\rrbracket \stackrel{\overline{a}}{\Longrightarrow} C'$. By Lemma 7.2(2), C exists s.t.

$$P \stackrel{\overline{a}}{\Longrightarrow} C \text{ and } C' \approx_{\mathrm{Tr}} \mathcal{T}\llbracket C\rrbracket\,.$$

Since $P \approx_{\mathrm{Nr}} Q$, there exist D s.t. $Q \stackrel{\overline{a}}{\Longrightarrow} D$ and

$$D \bullet \mathrm{Ab}_m \approx_{\mathrm{Nr}} C \bullet \mathrm{Ab}_m \,. \tag{12}$$

Further, by Lemma 7.2(1),

$$\mathcal{T}[\![Q]\!] \stackrel{\overline{a}}{\Longrightarrow} D' \approx_{\mathrm{Tr}} \mathcal{T}[\![D]\!]\,.$$

By Lemma 7.3, $\mathcal{T}[\![C]\!] = \boldsymbol{\nu}\, m\, \langle \mathrm{Tr}_m \rangle \mathcal{T}[\![C \bullet \mathrm{Ab}_m]\!]$ and $\mathcal{T}[\![D]\!] = \boldsymbol{\nu}\, m\, \langle \mathrm{Tr}_m \rangle \mathcal{T}[\![D \bullet \mathrm{Ab}_m]\!]$. Summarising, we have:

$$\begin{array}{lll} \mathcal{T}[\![P]\!] & \stackrel{\overline{a}}{\Longrightarrow} \approx_{\mathrm{Tr}} & \boldsymbol{\nu}\, m\, \langle \mathrm{Tr}_m \rangle \mathcal{T}[\![C \bullet \mathrm{Ab}_m]\!] \\ \mathcal{T}[\![Q]\!] & \stackrel{\overline{a}}{\Longrightarrow} \approx_{\mathrm{Tr}} & \boldsymbol{\nu}\, m\, \langle \mathrm{Tr}_m \rangle \mathcal{T}[\![D \bullet \mathrm{Ab}_m]\!] \end{array}$$

and, since by (12), $C \bullet \mathrm{Ab}_m \approx_{\mathrm{Nr}} D \bullet \mathrm{Ab}_m$, we have $\mathcal{T}[\![C \bullet \mathrm{Ab}_m]\!] \mathrel{\mathcal{R}} \mathcal{T}[\![D \bullet \mathrm{Ab}_m]\!]$. This is enough, because $\mathcal{R}$ is a $\approx_{\mathrm{Tr}}$-bisimulation up-to $\approx_{\mathrm{Tr}}$. ☐

Theorem 7.4 can be extended to open agents:

Corollary 7.5 *Relations $\approx_{Nr}$ and $\approx_{Ct}$ coincide on $\mathcal{A}g \times \mathcal{A}g$.*

PROOF: Again, the inclusion $\approx_{\mathrm{Ct}} \subseteq \approx_{\mathrm{Nr}}$ is given by definition. For the opposite inclusion, one can follow essentially the same argument we used to explain the idea of normal bisimulation, right after its definition. Thus, in case of closed abstractions F and G, by definition $F \approx_{\mathrm{Nr}} G$ means $F \circ \mathrm{Tr}_m \approx_{\mathrm{Nr}} G \circ \mathrm{Tr}_m$. Since these are closed processes, by Theorem 7.4, $F \circ \mathrm{Tr}_m \approx_{\mathrm{Ct}} G \circ \mathrm{Tr}_m$; then, using the congruence properties of $\approx_{\mathrm{Ct}}$ and the factorisation theorem, we deduce that for all closed concretions C, it holds that $F \bullet C \approx_{\mathrm{Ct}} G \bullet C$, which means $F \approx_{\mathrm{Ct}} G$. The proof for closed concretion is similar. Having now proved the result for all closed agents, one can extend it to open agents using the same proof schema. ☐

Corollary 7.6 *Relations $\approx_{Tr}$, $\approx_{Nr}$ and $\approx_{Ct}$ coincide on $\mathcal{T}\mathcal{A}g \times \mathcal{T}\mathcal{A}g$.*

PROOF: Follows from Theorem 6.10 and Corollary 7.5. In the case of closed concretions $\boldsymbol{\nu}\, m\, \langle \mathrm{Tr}_m \rangle P$ and $\boldsymbol{\nu}\, m\, \langle \mathrm{Tr}_m \rangle Q$ in the relation $\approx_{\mathrm{Ct}}$, one shows $P \approx_{\mathrm{Tr}} Q$ proceeding as the proof of Lemma 6.8 (precisely the proof of $P'_1 \approx_{\mathrm{Ct}} P'_2$ from (6)). ☐

Remark 7.7 Now that we have proved that $\approx_{\mathrm{Ct}}$ and $\approx_{\mathrm{Nr}}$ coincide, one might wonder why we did not introduce $\approx_{\mathrm{Nr}}$ directly. The reason is that we would have needed $\approx_{\mathrm{Ct}}$ to prove the congruence of $\approx_{\mathrm{Nr}}$ over parallel composition. Moreover, it is easier to convince ourselves of the naturalness of context bisimulation; we can then accept normal bisimulation as a simpler characterisation of the former. ☐

7.1 Late and early equivalences

As pointed out in Remark 3.12, the formulation of weak context bisimulation in Definition 3.11 is in the *late* style. In the early style, the order of quantifiers in the input and output clauses is reversed. Thus, $\mathcal{R}$ is an *weak early context bisimulation* if $P \mathrel{\mathcal{R}} Q$ implies:

1. whenever $P \xrightarrow{\ell} P'$, there exists Q' s.t. $Q \overset{\widehat{\ell}}{\Longrightarrow} Q'$ and $P' \mathcal{R} Q'$;
2. whenever $P \xrightarrow{a} F$, then for all closed concretions C there exists G s.t. $Q \overset{a}{\Longrightarrow} G$ and $C \bullet F \mathcal{R} C \bullet G$;
3. whenever $P \xrightarrow{\overline{a}} C$, then for all closed abstractions F there exists D s.t. $Q \overset{\overline{a}}{\Longrightarrow} D$ and $F \bullet C \mathcal{R} F \bullet D$.

We denote the largest weak early context bisimulation by $\approx^{\mathrm{E}}_{\mathrm{Ct}}$. Hansen and Kleist [HK94] have proved that late and early strong context bisimulations coincide on an asynchronous variant of Plain CHOCS. The theory developed in this paper yields a straightforward proof of the result for the weak equivalences.

Corollary 7.8 *Relations $\approx_{Ct}$ and $\approx^{E}_{Ct}$ coincide.*

Proof: Clearly, $\approx_{\mathrm{Ct}} \subseteq \approx^{\mathrm{E}}_{\mathrm{Ct}}$: Every late bisimulation is an early bisimulation. On the other hand, an early bisimulation is also a normal bisimulation: Hence, by Corollary 7.5, $\approx^{\mathrm{E}}_{\mathrm{Ct}} \subseteq \approx_{\mathrm{Ct}}$. ☐

By contrast, late and early bisimulations are usually different in first-order calculi like the π-calculus [MPW92, PS95]. Note also that in normal and triggered bisimulation the late vs. early issue disappears, since the responsible quantifiers are absent in their definitions.

8 Extensions

In this section, we discuss the extension of the theory presented to a richer calculus, namely the Higher-Order π-calculus, briefly HOπ [San92]. We shall focus on the higher-order fragment of HOπ, thus ignoring first-order features like communication of names and abstraction over names. (This fragment of) HOπ can be thought of as an ω-order extension of the process-passing language in Section 2. Also, we shall stick to a monadic calculus (only one agent can be transmitted at a time), although polyadicity can be easily accommodated. The discussion in this section is brief and informal; details are found in [San92]. The reader not interested in the extension can safely move to the following section.

In the CHOCS-like language of Section 2, only processes can be passed around. In HOπ, besides processes, abstractions can be passed too. Moreover, abstractions can be of arbitrary high order. We explain what the order of an abstraction is. All abstractions seen so far are of the form $(X)\,P$, where X is a process variable which may appear in P. If $*$ is taken to be the type of all processes, then from a function-theoretic point of view, $(X)\,P$ has type $* \to *$, for $(X)\,P$ takes a process and returns a process. We shall say that $(X)\,P$ is a second-order abstraction (first-order abstractions being abstractions on names, as found in the full *HOπ*). The syntax of HOπ allows agent-variables, rather than just process-variables, and an application construct $A_1 \circ A_2$ (sometimes written $A_1\langle A_2\rangle$ in the literature); thus, one can write meaningful abstractions of order greater than two. An example is

$$G \stackrel{\text{def}}{=} (Y)\,(Q \mid Y \circ Q)$$

G takes an agent of type $* \to *$ and and yields back a process. Therefore G has type $(* \to *) \to *$. Abstraction G has order three, the order being determined by the level of arrow-nesting in the type. If F is $(X)\,(P \mid X)$, then $G \circ F$ yields $Q \mid F \circ Q = Q \mid P \mid Q$.

In the same way, we can construct fourth-order abstractions, fifth-order abstractions and so forth. In this sense HOπ is a ω-order calculus: There is no bound on the order of agents which can be written and communicated. Abstractions can also be passed around, like in

$$P \stackrel{\text{def}}{=} \Big(\overline{a}.\,\langle G\rangle \overline{b}.\,\langle F\rangle \mathbf{0}\Big) \mid a.\,(Z)\,b.\,(Y)\,(Z \circ Y)$$

where G and F are the abstractions above defined, and then we have (garbage-collecting $\mathbf{0}$ processes)

$$P \xrightarrow{\tau} \xrightarrow{\tau} G \circ F = Q \mid P \mid Q\,.$$

There is also a type discipline on names, so to avoid disagreement on what is carried or expected at a given name. For instance, in process P above name a would have type $(* \to *) \to *$ and name b type $* \to *$. Types are extended to concretions in the expected way: If G has type T, then $\langle G\rangle P$ has type $T \to *$. The pseudo-application $\bullet$ and the application $\circ$ are only allowed between agents of compatible types. Below, we write $A : T$ if A has type T.

The definition of context bisimulation can be easily extended to HOπ — one just have to take into account the type informations. More interesting is to see how to extend the definitions of trigger bisimulation and normal bisimulation. We shall briefly consider the latter. We first have to understand what is the appropriate notion of trigger in HOπ. A trigger of type $T = S \to *$ at m is

$$\mathrm{Tr}^T_m \stackrel{\text{def}}{=} (X)\,\overline{m}.\,\langle X\rangle \mathbf{0}$$

where X is a variable of type S. Thus a trigger takes an argument X and export it in the action at m. The appearance and the use of the argument X is the new ingredient w.r.t. the triggers of Section 4.2. Dually, for $T = (S \to *) \to *$, the abstraction Ab_m used in the output clause of normal bisimulation becomes

$$\mathrm{Ab}^T_m \stackrel{\text{def}}{=} (Y)\;!\,m.\,(X)\,(Y \circ X)$$

where Y has type $S \to *$ and X has type S. Note that $\mathrm{Tr}^T_m : T$ and $\mathrm{Ab}^T_m : T$.

Thus, the requirements of normal bisimulation over HOπ processes are the following: $P \approx_{\mathrm{Nr}} Q$ implies:

- Whenever $P \xrightarrow{\ell} P'$,there exists Q' s.t. $Q \stackrel{\widehat{\ell}}{\Longrightarrow} Q'$ and $P' \approx_{\mathrm{Nr}} Q'$;
- for all types $T = S \to *$, whenever $P \xrightarrow{a} F$ with $F : T$, there exists $G : T$ s.t. $Q \stackrel{a}{\Longrightarrow} G$ and $F \circ \mathrm{Tr}^S_m \approx_{\mathrm{Nr}} G \circ \mathrm{Tr}^S_m$, for some fresh m;

- for all T, whenever $P \xrightarrow{\overline{a}} C$ with $C : T$, there exists $D : T$ s.t. $Q \overset{\overline{a}}{\Longrightarrow} D$ and $C \bullet \mathrm{Ab}_m^T \approx_{\mathrm{Nr}} D \bullet \mathrm{Ab}_m^T$, for some fresh m.

In this way, normal bisimulation and context bisimulation coincide over HOπ processes.

9 Conclusions

In this paper we have considered a few bisimilarity equivalences for higher-order process calculi, in particular context bisimulation and normal bisimulation. Normal bisimulation specifies some minimum requirements on higher-order actions and hence provides us with a useful mathematical tool to verify agent equivalences; the characterisation in terms of context bisimulation gives us a measure of the power of such requirements and reinforces the naturalness of the equivalence. We have isolated a subclass of agents, called *triggered agents*, which represent some sort of "normal form" for agents. We have shown that on triggered agents context and normal bisimulations also coincide with triggered bisimulation, which is the simplest of the bisimilarities examined.

In [San92] we also compare context bisimulation and normal bisimulation with barbed congruence, introduced in [MS92] as a tool to *uniformly* define bisimulation-based equivalences in different calculi. It is shown in [San92] that under certain conditions on the syntax of the calculus, barbed congruence coincides with the "early non-delay" version of context bisimulation. In this paper, we have chosen the "late delay" schema because it seems more appropriate with an operational semantics based on the abstraction and concretion constructs.

At a first glance, it appears surprising that the bisimilarity clauses of normal bisimulation and triggered bisimulation contain no form of universal quantifications on matching actions. For instance, in normal bisimulation to see whether an input action $P \xrightarrow{a} F$ is matched by the input action $Q \xrightarrow{a} G$, it suffices to examine the derivative abstractions F and G on a *single* argument (a trigger): This guarantees that F and G are equivalent on *all* possible arguments. This simplicity contrasts with what happens in first-order calculi like the π-calculus, where the bisimilarity clauses require the examination of different arguments (as in late and early bisimulations [MPW92, PS95]) or, at least, impose multiple name instantiations (as in open bisimulation [San96b]). But the extra complexity of first-order calculi can be understood and justified with their greater expressiveness: See for instance [San96a], where it is shown that the expressiveness of a Plain CHOCS-like calculus is the same as that of a sublanguage of the π-calculus obeying some strong syntactic constraints (essentially, this is due to the ability of transmitting private names in the π-calculus).

It would be interesting to know at which extent the results presented here depend upon the choice of operators in our higher-order calculus. An important condition seems to be that context bisimulation be a congruence relation — the factorisation theorem would fail otherwise. Thus, for instance, adding summation in an unconstrained way would break the factorisation theorem, but guarded summation would not [San92].

We hope that the results presented can contribute to the development of a manageable and solid theory for higher-order process calculi. In [San92], we used these results to prove the full abstraction of a compilation from HOπ to π-calculus. As argued in [San92, Hen93], the possibility of encoding higher-order calculi into first-order calculi does not mean that the former are superfluous: Higher-order constructs arise in many applications, for instance operating systems, and it is advantageous to be able to use them and to reason with them *directly*, i.e. not via an encoding.

Appropriate extensions of the factorisation theorem and of normal bisimulation might also be useful on first-order or mobility-free calculi, to develop a theory of equality of open terms (or process contexts). Progress in this direction has been made by Sewell [Sew95], using techniques (seemingly) related to ours.

We have compared the weak delay versions of context bisimulation, normal bisimulation and triggered bisimulation. We believe that similar results can be obtained for the *branching* [GW89] versions of these bisimilarities. We do not know at present if they could be established also for the *strong* versions of the equivalences, where τ-actions have the same weight as visible actions. Indeed, some of our central technical results, like the factorisation theorem and the full abstraction of the mapping to triggered agents, are only true in the weak case.

We have carried out an operational study of higher-order process calculi. The denotational approach has been investigated by Hennessy [Hen93], who has given a model for (a slight variant of) Thomsen's CHOCS [Tho90]. The model is constructed from a domain equation and is proved to be fully abstract for a form of may testing. The language CHOCS differs from Plain CHOCS, and hence from the calculus used in this paper, in an important aspect: in CHOCS the restriction operator is a *dynamic* binder, whereas in Plain CHOCS it is a *static* binder. To see the difference, suppose b is a name which occurs free in P and Q. Having static binding, an interaction between $a.\,(X)\,X$ and $\boldsymbol{\nu}\, b\,(\overline{a}.\,\langle P\rangle Q)$ is

$$\big(a.\,(X)\,X\big) \mid \boldsymbol{\nu}\, b\,(\overline{a}.\,\langle P\rangle Q) \;\xrightarrow{\tau}\; \boldsymbol{\nu}\, b\,(P \mid Q)$$

and a scope extrusion of $\boldsymbol{\nu}\, b$ accompanies the movement of process P. By contrast, having dynamic binding we would get:

$$\big(a.\,(X)\,X\big) \mid \boldsymbol{\nu}\, b\,\big(\overline{a}.\,\langle P\rangle Q\big) \;\xrightarrow{\tau}\; P \mid \boldsymbol{\nu}\, b\, Q\,.$$

Note that the reduction has destroyed the privacy of the b-link between P and Q: The free occurrences of b in P have evaded the restriction which embraced them. The main advantage of dynamic binding is an easier semantics (operationally and denotationally); but static binding facilitates the analysis of a program from its text. Our and Hennessy's works are tailored to the specific discipline chosen for restriction. In both cases, it appears that the theory developed would not support a different discipline.

It would be interesting to understand whether our formulations of bisimulation for higher-order calculi can be expressed within the framework of *generalised algebraic specifications* of Astesiano et al. [AGR92], especially to shed lights on modal logics for the bisimulations that we have introduced in the paper.

Acknowledgements. The author would like to express his gratitude to Robin Milner, who supervised the thesis work on which this paper is based, for his encouragement and many insightful discussions. The author is also very grateful to Egidio Astesiano and Takayasu Ito, whose suggestions lead to several improvements of the technical presentation. Finally, thanks to Takayasu Ito and the two anonymous referees for pointing out a number of corrections on an earlier version of the paper.

References

[Abr89] S. Abramsky. The lazy lambda calculus. In D. Turner, editor, *Research Topics in Functional Programming*, pages 65–116. Addison-Wesley, 1989.

[AD95] R. Amadio and M. Dam. Reasoning about higher-order processes. In Mosses P. et al. editor, Proceedings of TAPSOFT'95, volume 915 of *Lecture Notes in Computer Science*, pages 202–216. Springer Verlag, 1995.

[AGR88] E. Astesiano, A. Giovini, and G. Reggio. Generalized bisimulation in relational specifications. In *STACS '88*, volume 294 of *Lecture Notes in Computer Science*, pages 207–226. Springer Verlag, 1988.

[AGR92] E. Astesiano, A. Giovini, and G. Reggio. Observational structures and their logic. *Theoretical Computer Science*, 96:249–283, 1992.

[Ama93] R. Amadio. On the reduction of CHOCS bisimulation to π-calculus bisimulation. In E. Best, editor, *Proceedings of CONCUR '93*, volume 715 of *Lecture Notes in Computer Science*, pages 112–126. Springer Verlag, 1993.

[BCHK94] G. Boudol, I. Castellani, M. Hennessy, and A. Kiehn. A theory of processes with localities. *Formal Aspects of Computing*, 6:165–200, 1994.

[Bou89] G. Boudol. Towards a lambda calculus for concurrent and communicating systems. In *TAPSOFT '89*, volume 351 of *Lecture Notes in Computer Science*, pages 149–161, 1989.

[CH89] I. Castellani and M. Hennessy. Distributed bisimulation. *JACM*, 10(4):887–911, 1989.

[GW89] R.J. van Glabbeek and W. P. Weijland. Branching time and abstraction in bisimulation semantics. In G.X. Ritter, editor, *Information Processing '89*, pages 613–618. Elsevier Science, 1989.

[Hen93] M. Hennessy. A fully abstract denotational model for higher-order processes. In *Proc. 8th LICS Conf.*, pages 397–408. IEEE Computer Society Press, 1993.

[HK94] M. Hansen and J. Kleist. Process calculi with asynchronous communication. Master thesis, Aalborg University (supervisor Hans Hüttel.), August 1994.

[Mil80] R. Milner. *A Calculus of Communicating Systems*, volume 92 of *Lecture Notes in Computer Science*. Springer Verlag, 1980.

[Mil91] R. Milner. The polyadic π-calculus: a tutorial. Technical Report ECS–LFCS–91–180, LFCS, Dept. of Comp. Sci., Edinburgh Univ., October 1991. Also in *Logic and Algebra of Specification*, ed. F.L. Bauer, W. Brauer and H. Schwichtenberg, Springer Verlag, 1993.

[MPW92] R. Milner, J. Parrow, and D. Walker. A calculus of mobile processes, (Parts I and II). *Information and Computation*, 100:1–77, 1992.

[MS92] R. Milner and D. Sangiorgi. Barbed bisimulation. In W. Kuich, editor, *19th ICALP*, volume 623 of *Lecture Notes in Computer Science*, pages 685–695. Springer Verlag, 1992.

[Par81] D.M. Park. Concurrency on automata and infinite sequences. In P. Deussen, editor, *Conf. on Theoretical Computer Science*, volume 104 of *Lecture Notes in Computer Science*. Springer Verlag, 1981.

[PR96] A.M. Pitts and J.R.X. Ross. Process calculus based upon evaluation to committed forms. In *Proceedings of CONCUR '96*, Lecture Notes in Computer Science, pages 18–33. Springer Verlag, 1996.

[PS95] J. Parrow and D. Sangiorgi. Algebraic theories for name-passing calculi. *Information and Computation*, 120(2):174–197, 1995.

[San92] D. Sangiorgi. *Expressing Mobility in Process Algebras: First-Order and Higher-Order Paradigms*. PhD thesis CST–99–93, Department of Computer Science, University of Edinburgh, 1992.

[San96a] D. Sangiorgi. π-calculus, internal mobility and agent-passing calculi. *Theoretical Computer Science*, 167(2):235–274, 1996.

[San96b] D. Sangiorgi. A theory of bisimulation for the π-calculus. *Acta Informatica*, 33:69–97, 1996. An extract appeared in *Proc. CONCUR '93*, Lecture Notes in Computer Science 715, Springer Verlag.

[Sew95] P.M. Sewell. *The Algebra of Finite State Processes*. PhD thesis CST–118-95, Department of Computer Science, University of Edinburgh, 1995.

[Tho90] B. Thomsen. *Calculi for Higher Order Communicating Systems*. PhD thesis, Department of Computing, Imperial College, 1990.

[Tho93] B. Thomsen. Plain CHOCS, a second generation calculus for higher-order processes. *Acta Informatica*, 30:1–59, 1993.

[Wei89] W.P. Weijland. *Synchrony and Asynchrony in Process Algebra.* PhD thesis, University of Amsterdam, 1989.

A Proof of Lemma 3.8

In this appendix we prove Lemma 3.8, that is that $\sim_{\mathrm{Ct}}$ is a congruence on substitutions. To prove the general result, we first prove some instances of it (Lemma A.2).

Lemma A.1 *If* $!P \xrightarrow{\mu} A$, *then* $A \sim_{Ct} A_1 \mid !P$, *for some* A_1 *s.t.* $P \mid P \xrightarrow{\mu} A_1$.

Proof: By transition induction. ☐

Lemma A.2 *For all* $P_1, P_2, R \in \mathcal{P}r$, $P_1 \sim_{Ct} P_2$ *implies*

1. $\boldsymbol{\nu}\, x\, P_1 \sim_{Ct} \boldsymbol{\nu}\, x\, P_2$;
2. $P_1 \mid R \sim_{Ct} P_2 \mid R$;
3. $!P_1 \sim_{Ct} !P_2$.

Proof: We only examine (2) and (3). It is enough to prove the result on closed expressions. For (2), we show that

$$\mathcal{R} = \{(\boldsymbol{\nu}\, \widetilde{x}\,(P_1 \mid R), \boldsymbol{\nu}\, \widetilde{x}\,(P_2 \mid R)) \;:\; P_1 \sim_{\mathrm{Ct}} P_2\} \cup \sim_{\mathrm{Ct}}$$

is a bisimulation up-to $\sim_{\mathrm{Ct}}$. Take $(\boldsymbol{\nu}\, \widetilde{x}\,(P_1 | R), \boldsymbol{\nu}\, \widetilde{x}\,(P_2 | R)) \in \mathcal{R}$ with $P_1 \sim_{\mathrm{Ct}} P_2$ and suppose that $\boldsymbol{\nu}\, \widetilde{x}\,(P_1 \mid R) \xrightarrow{\mu} A$. There are five cases to consider:

Case 1 $P_1 \xrightarrow{\mu} A_1$ and $A = \boldsymbol{\nu}\, \widetilde{x}\,(A_1 \mid R)$.

Use the definition of $\sim_{\mathrm{Ct}}$ and simple algebraic manipulations with the laws of Lemma 3.5.

Case 2 $R \xrightarrow{\mu} A_1$ and $A = \boldsymbol{\nu}\, \widetilde{x}\,(P_1 \mid A_1)$.

Similar to case 1.

Case 3 $P_1 \xrightarrow{\overline{a}} C_1$, $R \xrightarrow{a} F$, $\mu = \tau$ and $A = \boldsymbol{\nu}\, \widetilde{x}\,(C_1 \bullet F)$.

By definition of $\sim_{\mathrm{Ct}}$, $P_1 \sim_{\mathrm{Ct}} Q_1$ and $P_1 \xrightarrow{\overline{a}} C_1$ imply that $P_2 \xrightarrow{\overline{a}} C_2$ and $G \bullet C_1 \sim_{\mathrm{Ct}} G \bullet C_2$, for all G. Using this and assertion (1) of this lemma, we infer $\boldsymbol{\nu}\, \widetilde{x}\,(P_2 | R) \xrightarrow{\tau} \boldsymbol{\nu}\, \widetilde{x}\,(C_2 \bullet F) \sim_{\mathrm{Ct}} \boldsymbol{\nu}\, \widetilde{x}\,(C_1 \bullet F)$, which closes up the bisimulation.

Case 4 $P_1 \xrightarrow{a} F_1$, $R \xrightarrow{a} C$, $\mu = \tau$ and $A = \boldsymbol{\nu}\, \widetilde{x}\,(F_1 \bullet C)$.

Similar to case 3.

Case 5 Interaction between P_1 and R along a first-order name.

This case is simpler than cases (3) and (4).

Now, assertion (3) of the lemma. We show that

$$\mathcal{R} \overset{\mathrm{def}}{=} \{(\,!\,P \mid R, !\,Q \mid R) \;:\; P \sim_{\mathrm{Ct}} Q\}$$

is a $\sim_{\mathrm{Ct}}$-bisimulation up-to $\sim_{\mathrm{Ct}}$. We first notice that since $\sim_{\mathrm{Ct}}$ is transitive and preserved by parallel composition (assertion (2) of this lemma), $P \sim_{\mathrm{Ct}} Q$ implies $P \mid P \sim_{\mathrm{Ct}} Q \mid Q$.

We show that $!\,Q \mid R$ can match any action from $!\,P \mid R$, say $!\,P \mid R \xrightarrow{\mu} T_1$. We only examine the case in which μ is a silent action and originates from $!\,P$; the remaining cases are similar. Thus, assume $T_1 = P_1 \mid R$, for some P_1 s.t. $!\,P \xrightarrow{\tau} P_1$. By Lemma A.1, if $!\,P \xrightarrow{\tau} P_1$ then $P_1 \sim_{\mathrm{Ct}} P_1' \mid !\,P$, for some P_1' s.t. $P \mid P \xrightarrow{\tau} P_1'$. Since $Q \mid Q \sim_{\mathrm{Ct}} P \mid P$, there is Q_1' s.t. $Q \mid Q \xrightarrow{\tau} Q_1' \sim_{\mathrm{Ct}} P_1'$; therefore, $!\,Q \xrightarrow{\tau} \sim_{\mathrm{Ct}} !\,Q \mid Q_1'$. From these facts, and using the congruence of $\sim_{\mathrm{Ct}}$ for parallel composition, we have:

$$\begin{array}{lll} !\,P \mid R & \xrightarrow{\tau} \sim_{\mathrm{Ct}} & !\,P \mid P_1' \mid R \\ !\,Q \mid R & \xrightarrow{\tau} \sim_{\mathrm{Ct}} & !\,Q \mid P_1' \mid R\,. \end{array}$$

This is enough, since $(\,!\,P \mid P_1' \mid R, !\,Q \mid P_1' \mid R) \in \mathcal{R}$. ☐

We say that a variable X is *guarded* in an agent A if X only occurs in subexpressions of A of the form $\mu.B$.

Lemma A.3 *Suppose that $fv(P) = \{X\}$ and that X is guarded in P. Then, for all Q,*

1. *If $P \xrightarrow{\mu} A$, then $P\{Q/X\} \xrightarrow{\mu} A\{Q/X\}$;*
2. *If $P\{Q/X\} \xrightarrow{\mu} A'$, then there is A s.t. $P \xrightarrow{\mu} A$ and $A' = A\{Q/X\}$.*

Proof: By transition induction. ☐

Lemma A.4 *Suppose $X_1 \in fv(P_1)$ and that $X_2 \notin fv(P_1)$. Then there exists P_2 such that*

1. *X_1 is guarded in P_2;*
2. *$P_2\{X_1/X_2\} = P_1$;*
3. *if $Q_1 \sim_{Ct} Q_2$, then $P_1\{Q_1/X_1\} \sim_{Ct} P_2\{Q_1/X_1, Q_2/X_2\}$.*

Proof: Induction on the structure of P_1. The basis of the induction occurs when $P_1 = X$ or $P_1 = \mu.A$. If $P_1 = X$, then define $P_2 \overset{\mathrm{def}}{=} P_1$ if $X \neq X_1$ and $P_2 \overset{\mathrm{def}}{=} X_2$ if $X = X_1$; if $P_1 = \mu.A$, then take $P_2 \overset{\mathrm{def}}{=} P_1$.

When P_1 is a process of the form $R_1 \mid R_2$, $\boldsymbol{\nu}\, x\, R$, or $!\,R$, use the inductive hypothesis on the R or R_i's and (to prove assertion (3)) Lemma A.2. ☐

We are now ready to prove Lemma 3.8. We first recall its assertion.

Lemma 3.8 Let $P_1, P_2, A \in \mathcal{A}g$; then $P_1 \sim_{\mathrm{Ct}} P_2$ implies $A\{P_1/X\} \sim_{\mathrm{Ct}} A\{P_2/X\}$.

PROOF: It suffices to prove the result for closed processes. Consider the set $\mathcal{R}$ of all pairs of the form:

$$\big(P\{Q_2/X_1\}, P\{Q_1/X_1\}\big), \text{ with } Q_2 \sim_{\mathrm{Ct}} Q_1 \text{ and } X_1 \text{ guarded in } P.$$

We prove the following facts:

(a) $\mathcal{R}$ is a $\sim_{\mathrm{Ct}}$-bisimulation up-to $\sim_{\mathrm{Ct}}$;

(b) for all R with $\mathrm{fv}(R) \subseteq \{X\}$, if $Q_1 \sim_{\mathrm{Ct}} Q_2$, then

$$R\{Q_2/X\} \; \mathcal{R} \; \sim_{\mathrm{Ct}} R\{Q_1/X\}\,.$$

Then the assertion of the proposition follows from (a) and (b) by transitivity of $\sim_{\mathrm{Ct}}$.

We first prove (b). We can apply Lemma A.4 to R; let R' be the process returned. We have

$$\begin{array}{rcl} R\{Q_2/X_1\} & = & \text{(by Lemma A.4(2))} \\ R'\{Q_2/X_1, Q_2/X_2\} & \mathcal{R} & (X_1 \text{ is guarded in } R' \text{ by Lemma A.4(1))} \\ R'\{Q_1/X_1, Q_2/X_2\} & \sim_{\mathrm{Ct}} & \text{(by Lemma A.4(3))} \\ R\{Q_1/X_1\} & & \end{array} \tag{13}$$

which proves fact (b).

We now prove fact (a). Let $P\{Q_2/X_1\} \; \mathcal{R} \; P\{Q_1/X_1\}$. We show that the actions of $P\{Q_2/X_1\}$ can be matched by $P\{Q_1/X_1\}$. Since X_1 is guarded in P, by Lemma A.3 we can infer all transitions for $P\{Q_2/X_1\}$ and $P\{Q_1/X_1\}$ from those of P. There are three cases:

Case (a) $P \xrightarrow{\ell} P'$.

Then we have the transitions

$$P\{Q_2/X_1\} \xrightarrow{\ell} P'\{Q_2/X_1\}, \text{ and } P\{Q_1/X_1\} \xrightarrow{\ell} P'\{Q_1/X_1\}\,.$$

By fact (b), it holds that $P'\{Q_2/X_1\} \; \mathcal{R} \; \sim_{\mathrm{Ct}} P'\{Q_1/X_1\}$, which closes up the bisimulation.

Case (b) $P \xrightarrow{\overline{a}} C$.

Then we have the transitions

$$P\{Q_2/X_1\} \xrightarrow{\overline{a}} C\{Q_2/X_1\}, \text{ and } P\{Q_1/X_1\} \xrightarrow{\overline{a}} C\{Q_1/X_1\}$$

For every closed abstraction G, we have $G \bullet (C\{Q_i/X_1\}) = (G \bullet C)\{Q_i/X_1\}$, $i = 1, 2$. Let $R = G \bullet C$. Then $R\{Q_2/X_1\} \; \mathcal{R} \; \sim_{\mathrm{Ct}} R\{Q_1/X_1\}$ follows by fact (b).

Case (c) $P \xrightarrow{a} F$.

Similar to case (b). $\square$

Remark A.5 The weak version of Lemma 3.8 (i.e., $R \approx_{\mathrm{Ct}} Q$ implies $A\{R/X\} \approx_{\mathrm{Ct}} A\{Q/X\}$) can be proved along the same lines. We should mention, however, that in place of the "bisimulation up-to $\sim_{\mathrm{Ct}}$" technique (used in the proofs of Lemma A.2 and Lemma 3.8), one should use the following up-to technique for $\approx_{\mathrm{Ct}}$:

Given a symmetric relation $\mathcal{R} \subseteq \mathcal{P}r^{\mathrm{o}} \times \mathcal{P}r^{\mathrm{o}}$, suppose that $(P, Q) \in \mathcal{R}$ imply:

1. whenever $P \xrightarrow{\ell} P'$, there is Q' s.t. $Q \overset{\widehat{\ell}}{\Longrightarrow} Q'$ and $P' \sim_{\mathrm{Ct}} \mathcal{R} \approx_{\mathrm{Ct}} Q'$;
2. whenever $P \xrightarrow{a} F$, there exists G s.t. $Q \overset{a}{\Longrightarrow} G$ and $C \bullet F \sim_{\mathrm{Ct}} \mathcal{R} \approx_{\mathrm{Ct}} C \bullet G$, for all concretions C.
3. whenever $P \xrightarrow{\overline{a}} C$, there exists D s.t. $Q \overset{\overline{a}}{\Longrightarrow} D$ and $F \bullet C \sim_{\mathrm{Ct}} \mathcal{R} \approx_{\mathrm{Ct}} F \bullet D$, for all abstractions F.

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

The soundness of this technique can be established using a standard argument for up-to techniques. $\square$

B Proof of Lemma 4.1

In this appendix we prove Lemma 4.1, that is that $\{z := B\}$ distributes over substitution, on the hypothesis that z is used only in output subject position. The presentation of this appendix is similar to that of Appendix A, where we inferred congruence for $\sim_{\mathrm{Ct}}$ over substitution. In both cases, we have first to prove some instance (Lemma A.2 and B.3, respectively) of the general result (Lemma 3.8 and 4.1, respectively). Moreover, the proof of Lemma 4.1 closely follows the proof of Lemma 3.8.

We write "x nsp A" to mean that name x occurs free in agent A only in output subject position. Our first target is to show that $\{z := B\}$ distributes over parallel composition. We cannot quite follow the proof technique used by Milner to show the analogous result for the π-calculus [Mil91, Section 5.4], due to the higher-order setting in which we are working. For our proof, Lemmas B.1 and B.2 are needed. Lemma B.1 relates the actions of the processes Q and Q', where Q' is obtained from Q through a substitution of one of its names. In the lemma, assertion (2) — the vice versa of (1) — is only possible because of the condition x, z nsp Q which prevents new possible interactions from being generated as effect of the substitution $Q\{z/x\}$.

Lemma B.1 *Suppose* x, z nsp Q. *We have:*

1. *If* $Q \xrightarrow{\mu} A$, *then* $Q\{z/x\} \xrightarrow{\mu\{z/x\}} A\{z/x\}$;

2. if $Q\{z/x\} \xrightarrow{\mu'} A'$*, then there exists* A *s.t.*

$Q \xrightarrow{\mu} A$ *with* $\mu' = \mu\{z/x\}$ *and* $A' = A\{z/x\}$.

Moreover both in (1) and and in (2), it holds that x, z nsp A*.*

Proof: A simple transition induction, both for (1) and for (2). ☐

Lemma B.2 *Suppose that* x, z_1 nsp Q*, that* $x, z_1 \notin fn(B)$ *and that* $z_2 \notin fn(Q, B)$*. Then*

$$Q\{z_1/x\}\,\{z_1 := B\} \sim_{Ct} Q\{z_2/x\}\,\{z_1 := B\}\,\{z_2 := B\}.$$

Proof: Below, for a generic process P, we write $P\,\{\widetilde{z} := \widetilde{B}\}$ for $P\,\{z_1 := B\}\,\{z_2 := B\}$. Since z_2 does not occur free in $Q\{z_1/x\}\,\{z_1 := B\}$, we have

$$Q\{z_1/x\}\,\{z_1 := B\} \sim_{\mathrm{Ct}} Q\{z_1/x\}\,\{\widetilde{z} := \widetilde{B}\}\,;$$

therefore it suffices to prove that $Q\{z_1/x\}\,\{\widetilde{z} := \widetilde{B}\} \sim_{\mathrm{Ct}} Q\{z_2/x\}\,\{\widetilde{z} := \widetilde{B}\}$. For this, we show that the set $\mathcal{R}$ of all pairs of the form

$$(Q\{z_1/x\}\,\{\widetilde{z} := \widetilde{B}\}, Q\{z_2/x\}\,\{\widetilde{z} := \widetilde{B}\})$$

with x, z_1 nsp Q, $x, z_1 \notin \mathrm{fn}(B)$ and $z_2 \notin \mathrm{fn}(Q, B)$, is a $\sim_{\mathrm{Ct}}$-bisimulation up-to $\sim_{\mathrm{Ct}}$. Suppose $P_1 \mathcal{R} P_2$, for

$$P_1 \overset{\mathrm{def}}{=} Q\{z_1/x\}\,\{\widetilde{z} := \widetilde{B}\}, \qquad P_2 \overset{\mathrm{def}}{=} Q\{z_2/x\}\,\{\widetilde{z} := \widetilde{B}\}.$$

We have to consider the actions of $Q\{z_1/x\}$ and $Q\{z_2/x\}$ which can cause an action of either P_1 or P_2. In the following we make use of Lemma B.1 which tells us how to infer the actions for $Q\{z_1/x\}$ and $Q\{z_2/x\}$ from those of Q.

Case (a) $Q \xrightarrow{\tau} Q'$.

We get the actions

$$P_1 \xrightarrow{\tau} Q'\{z_1/x\}\,\{\widetilde{z} := \widetilde{B}\} \quad \text{and} \quad P_2 \xrightarrow{\tau} Q'\{z_2/x\}\,\{\widetilde{z} := \widetilde{B}\}.$$

Now we are back in $\mathcal{R}$ because Lemma B.1 insures that Q' meets the conditions of $\mathcal{R}$.

Case (b) $Q \xrightarrow{a} G$.

By the conditions on x and z_1 it must be $a \notin \{x, z_1\}$. We thus get the actions:

$$P_1 \xrightarrow{a} G\{z_1/x\}\,\{\widetilde{z} := \widetilde{B}\} \overset{\mathrm{def}}{=} G_1 \quad \text{and } P_2 \xrightarrow{a} G\{z_2/x\}\,\{\widetilde{z} := \widetilde{B}\} \overset{\mathrm{def}}{=} G_2.$$

We have to prove that $C \bullet G_1 \sim_{\mathrm{Ct}} \mathcal{R} \sim_{\mathrm{Ct}} C \bullet G_2$, for every concretion C. Assuming, by alpha conversion, that x, z_1, z_2 do not occur free in C, we have

$$C \bullet G_i \sim_{\mathrm{Ct}} (C \bullet G)\{z_i/x\}\,\{\widetilde{z} := \widetilde{B}\}\,, \text{ for } i = 1, 2$$

which shows that $C \bullet G_1 \sim_{\mathrm{Ct}} \mathcal{R} \sim_{\mathrm{Ct}} C \bullet G_2$ holds.

Case (c) $Q \xrightarrow{\overline{a}} C$, with $a \notin \{x, z_1\}$.

Similar to case (b).

Case (d) $Q \xrightarrow{\overline{a}} C$, with $a \in \{x, z_1\}$.

We assume that $a = x$, as the case $a = z_1$ is similar. The transition by Q determines an interaction between $Q\{z_1/x\}$ and $!\, z_1 . B$ in P_1, and an interaction between $Q\{z_2/x\}$ and $!\, z_2 . B$ in P_2:

$$P_1 \xrightarrow{\tau} \sim_{\mathrm{Ct}} (B \bullet C)\{z_1/x\}\,\{\widetilde{z} := \widetilde{B}\} \quad \text{and} \quad P_2 \xrightarrow{\tau} \sim_{\mathrm{Ct}} (B \bullet C)\{z_2/x\}\,\{\widetilde{z} := \widetilde{B}\}.$$

and we are back in $\mathcal{R}$.

Case (e) $Q \xrightarrow{\ell} Q'$, with $\ell \neq \tau$.

Similar to the previous cases. $\square$

Lemma B.3 *Suppose that $z \notin$ fn(B) and that z nsp P, Q. Then*

1. $(\boldsymbol{\nu}\, x\, P)\ \{z := B\} \sim_{Ct} \boldsymbol{\nu}\, x\, (P\ \{z := B\})\, ,\ x \neq z.$
2. $(P \mid Q)\ \{z := B\} \sim_{Ct} P\ \{z := B\} \mid Q\ \{z := B\}.$
3. $(\,!P)\,\{z := B\} \sim_{Ct}\ !\,(P\ \{z := B\}).$

Proof: The hardest cases are (2) and (3), and we only consider these. It is enough to show the result for closed agents. We start with (2).

Let $x \notin \mathrm{fn}(P, Q)$ and $Q' \overset{\mathrm{def}}{=} Q\{x/z\}$. Then $Q = Q'\{z/x\}$; moreover, if $z' \notin \mathrm{fn}(P, Q, B)$, then

$$\begin{array}{rcl}
(P \mid Q)\,\{z := B\} & = & \\
(P \mid Q')\{z/x\}\,\{z := B\} & \sim_{\mathrm{Ct}} & \text{(by Lemma B.2)} \\
(P \mid Q')\{z'/x\}\,\{z := B\}\,\{z' := B\} & = & \\
(P \mid Q'\{z'/x\})\,\{z := B\}\,\{z' := B\} & \sim_{\mathrm{Ct}} & \text{(since } z \notin \mathrm{fn}(Q'\{z'/x\}, B) \text{ and } z' \notin \mathrm{fn}(P, B)) \\
\big(P\,\{z := B\}\big) \mid \big(Q'\{z'/x\}\,\{z' := B\}\big) & = & \text{(using alpha conversion)} \\
P\,\{z := B\} \mid Q\,\{z := B\} & &
\end{array}$$

Now assertion (4) of the lemma. We show that the set $\mathcal{R}$ of all pairs of the form

$$\big(Q \mid (\,!P)\,\{z := B\},\ Q \mid\ !\,(P\,\{z := B\})\big) \quad \text{with } z \notin \mathrm{fn}(B) \text{ and } z \text{ nsp } P$$

is a $\sim_{\mathrm{Ct}}$-bisimulation up-to $\sim_{\mathrm{Ct}}$. An action of

$$Q \mid (\,!P)\,\{z := B\} \qquad \text{or} \qquad Q \mid\ !\,(P\,\{z := B\})$$

comes from an action of Q alone, an action of P alone, an interaction between Q and P, or an interaction between P and $\{z := B\}$. We analyse the cases of higher-order output action by P and of communication between P and $\{z := B\}$, supposing z is a higher-order name.

Suppose $P \xrightarrow{\overline{a}} C$, with $a \neq z$. Since z nsp P, also z nsp C. Assuming z does indeed occur free in C, we have:

$$Q \mid (\,!\,P)\,\{z := B\} \quad \xrightarrow{\overline{a}} \sim_{\mathrm{Ct}} \quad (C \mid Q \mid \,!\,P)\,\{z := B\} \quad \stackrel{\mathrm{def}}{=} C_1$$

and

$$Q \mid \,!\,(P\,\{z := B\}) \qquad \xrightarrow{\overline{a}} \sim_{\mathrm{Ct}} \quad \big(C \mid Q \mid \,!\,(P\,\{z := B\})\big)\,\{z := B\} \quad \stackrel{\mathrm{def}}{=} C_2$$

We show that, for all G, it holds that $G \bullet C_1 \;\sim_{\mathrm{Ct}} \mathcal{R} \sim_{\mathrm{Ct}}\; G \bullet C_2$. By alpha conversion, we can assume that $z \notin \mathrm{fn}(G)$ and, therefore, using assertion (2) of this lemma, we have

$$\begin{aligned} G \bullet C_1 \quad & \sim_{\mathrm{Ct}} \quad (G \bullet C \mid Q \mid \,!\,P)\,\{z := B\} \\ & \sim_{\mathrm{Ct}} \quad (G \bullet C \mid Q)\,\{z := B\} \mid (\,!\,P)\,\{z := B\} \stackrel{\mathrm{def}}{=} R_1 \end{aligned}$$

and

$$\begin{aligned} G \bullet C_2 \quad & \sim_{\mathrm{Ct}} \quad \big(G \bullet C \mid Q \mid \,!\,(P\,\{z := B\})\big)\,\{z := B\} \\ & \sim_{\mathrm{Ct}} \quad (G \bullet C \mid Q)\,\{z := B\} \mid \,!\,(P\,\{z := B\}) \quad \stackrel{\mathrm{def}}{=} R_2 \end{aligned}$$

which is enough, since $R_1 \;\mathcal{R}\; R_2$.

Now we look at the case in which z is a higher-order name and there is an interaction between P and $\{z := B\}$. Suppose $P \xrightarrow{\overline{z}} C$ is the action of P. We have

$$\begin{aligned} Q \mid (\,!\,P)\,\{z := B\} & \quad \xrightarrow{\tau} \sim_{\mathrm{Ct}} \\ Q \mid \big(C \bullet B \mid \,!\,P\big)\,\{z := B\} & \quad \stackrel{\mathrm{def}}{=} \quad R_1 \end{aligned}$$

and

$$\begin{aligned} Q \mid \,!\,(P\,\{z := B\}) & \quad \xrightarrow{\tau} \sim_{\mathrm{Ct}} \\ Q \mid (C \bullet B)\,\{z := B\} \mid \,!\,(P\,\{z := B\}) & \quad \stackrel{\mathrm{def}}{=} \quad R_2\,. \end{aligned}$$

It holds that z nsp $(C \bullet B \mid \,!\,P)$. Therefore, using assertion (2) of this lemma,

$$R_1 \sim_{\mathrm{Ct}} Q \mid (C \bullet F)\,\{z := B\} \mid (\,!\,P)\,\{z := B\} \stackrel{\mathrm{def}}{=} R_1'$$

which, since $R_1' \;\mathcal{R}\; R_2$, closes up the bisimulation. □

Recall that a variable X is *guarded* in an agent A if X only occurs in subexpressions of A of the form $\mu.B$.

Lemma B.4 *Suppose that $X_1 \in fv(A_1)$ and that $X_2 \notin fv(A_1)$. Then there exists A_2 such that*

1. *X_1 is guarded in A_2;*
2. *$A_2\{X_1/X_2\} = A_1$;*
3. *for all Q, we have $A_1\{Q/X_1\}\,\{z := B\} \sim_{Ct} A_2\{Q/X_1, Q\,\{z := B\}/X_2\}\,\{z := B\}$.*

Proof: By induction on the structure of A_1. The basis of the induction occurs when $A_1 = X$ or $A_1 = \mu.A$. If $A_1 = X$, then define $A_2 \stackrel{\mathrm{def}}{=} A_1$ if $X \neq X_1$, and $A_2 \stackrel{\mathrm{def}}{=} X_2$ if $X = X_1$; if $A_1 = \mu.A$, then take $A_2 \stackrel{\mathrm{def}}{=} A_1$.

When A_1 is a process of the form $P_1 \mid P_2$, $\boldsymbol{\nu}\, x\, P$, or $!\, P$, use the inductive hypothesis and Lemma B.3. As an example, we consider the case of parallel composition and prove assertion (3). Suppose $A_1 = P_1 \mid P_2$. By induction there exists P_i', $i = 1, 2$ such that P_i' and P_i satisfy the assertion of the lemma. Define $A_2 \stackrel{\mathrm{def}}{=} P_1' \mid P_2'$. Abbreviating the substitution $\{Q/X_1, Q\,\{z := B\}/X_2\}$ as $\{\widetilde{Q_F}/\widetilde{x}\}$, we have:

$$\begin{array}{rcl}
(P_1 \mid P_2)\{Q/X_1\}\,\{z := B\} & = & \text{(distributing the substitution)} \\
\big(P_1\{Q/X_1\} \mid P_2\{Q/X_1\}\big)\,\{z := B\} & \sim_{\mathrm{Ct}} & \text{(Lemma B.3(2))} \\
P_1\{Q/X_1\}\,\{z := B\} \mid P_2\{Q/X_1\}\,\{z := B\} & \sim_{\mathrm{Ct}} & \text{(induction)} \\
P_1'\{\widetilde{Q_F}/\widetilde{x}\}\,\{z := B\} \mid P_2'\{\widetilde{Q_F}/\widetilde{x}\}\,\{z := B\} & \sim_{\mathrm{Ct}} & \text{(reversing the steps)} \\
(P_1' \mid P_2')\{\widetilde{Q_F}/\widetilde{x}\}\,\{z := B\}. & &
\end{array}$$

☐

We are now ready to prove Lemma 4.1. We first recall its assertion:

Lemma 4.1 Let $A, P, B \in \mathcal{A}g$. Suppose that $z \notin \mathrm{fn}(B)$ and that z occurs free in A and P only in output subject position. Then

$$A\{P/X\}\,\{z := B\} \sim_{\mathrm{Ct}} A\,\{z := B\}\{P\,\{z := B\}/X\}.$$

Proof: It suffices to prove the assertion for closed processes. Consider the set $\mathcal{R}$ of all pairs of the form:

$$\big(P\{Q/X_1\}\,\{z := B\}, P\,\{z := B\}\big\{Q\,\{z := B\}/X_1\big\}\big), \text{ with } X_1 \text{ guarded in } P.$$

We prove the following facts:

(a) $\mathcal{R}$ is a $\sim_{\mathrm{Ct}}$-bisimulation up-to $\sim_{\mathrm{Ct}}$;

(b) for all R with $\mathrm{fv}(R) \subseteq \{X\}$,

$$R\{Q/X_1\}\,\{z := B\} \sim_{\mathrm{Ct}} \mathcal{R} \sim_{\mathrm{Ct}} R\,\{z := B\}\big\{Q\,\{z := B\}/X_1\big\}.$$

The assertion of the theorem follows from (a) and (b) by transitivity of $\sim_{\mathrm{Ct}}$.

We first prove (b). We can apply Lemma B.4 to R; let R' be the process returned. We have

$$\begin{array}{rcl}
R\{Q/X_1\}\,\{z := B\} & \sim_{\mathrm{Ct}} & \text{(by Lemma B.4(3))} \\
R'\{Q/X_1, Q\,\{z := B\}/X_2\}\,\{z := B\} & = & \\
\big(R'\{Q\,\{z := B\}/X_2\}\big)\{Q/X_1\}\,\{z := B\} & \mathcal{R} & \\
 & & \text{(by Lemma B.4(1), } X_1 \text{ is guarded in } R'\text{)} \\
\big(R'\{Q\,\{z := B\}/X_2\}\big)\,\{z := B\}\big\{Q\,\{z := B\}/X_1\big\} & = & (z \text{ is not free in } Q\,\{z := B\}) \\
R'\,\{z := B\}\big\{Q\,\{z := B\}/X_1, Q\,\{z := B\}/X_2\big\} & = & \text{(using Lemma B.4(2))} \\
R\,\{z := B\}\{Q\,\{z := B\}/X_1\} & &
\end{array}$$

i.e., summarising:

$$R\{Q/X_1\}\,\{z := B\} \sim_{\mathrm{Ct}} \mathcal{R}\ R\,\{z := B\}\big\{Q\,\{z := B\}/X_1\big\},$$

which proves fact (b).

We now prove fact (a). Let $Q_1 \mathcal{R} Q_2$, for

$$Q_1 \stackrel{\mathrm{def}}{=} P\{Q/X_1\}\,\{z := B\} \qquad\qquad Q_2 \stackrel{\mathrm{def}}{=} P\,\{z := B\}\big\{Q\,\{z := B\}/X_1\big\}$$

with X_1 guarded in P. Because X_1 is guarded in P, we can infer all transitions for Q_1 and Q_2 from those of P. There are four cases to consider:

Case 1 $P \xrightarrow{\tau} P'$.

Then we have the transitions

$$Q_1 \xrightarrow{\tau} P'\{Q/X_1\}\,\{z := B\} \stackrel{\mathrm{def}}{=} Q_1',\;\; Q_2 \xrightarrow{\tau} P'\,\{z := B\}\big\{Q\,\{z := B\}/X_1\big\} \stackrel{\mathrm{def}}{=} Q_2'.$$

By fact (b), we have $Q_1' \sim_{\mathrm{Ct}} \mathcal{R} \sim_{\mathrm{Ct}} Q_2'$, which is enough because $\mathcal{R}$ a bisimulation up-to $\sim_{\mathrm{Ct}}$.

Case 2 $P \xrightarrow{\overline{a}} C$.

First suppose $a \neq z$. Then we have the transitions

$$Q_1 \quad \xrightarrow{\overline{a}} \quad C\{Q/X_1\}\,\{z := B\} \qquad Q_2 \quad \xrightarrow{\overline{a}} \quad C\,\{z := B\}\big\{Q\,\{z := B\}/X_1\big\}\,.$$

For every closed abstraction G, we have, using simple algebraic manipulations,

$$\begin{array}{rcl} G \bullet \big(C\{Q/X_1\}\,\{z := B\}\big) & \sim_{\mathrm{Ct}} & (G \bullet C)\{Q/X_1\}\,\{z := B\} \stackrel{\mathrm{def}}{=} Q_1' \\ G \bullet \big(C\,\{z := B\}\big\{Q\,\{z := B\}/X_1\big\}\big) & \sim_{\mathrm{Ct}} & (G \bullet C)\,\{z := B\}\big\{Q\,\{z := B\}/X_1\big\} \stackrel{\mathrm{def}}{=} Q_2' \end{array}$$

Now $Q_1' \sim_{\mathrm{Ct}} \mathcal{R} \sim_{\mathrm{Ct}} Q_2'$ holds by fact (b), for $R = G \bullet C$.

Suppose now that $a = z$. In this case the transition $P \xrightarrow{\overline{z}} C$ determines an interaction with $!\,z.B$ as follows in Q_1 and Q_2:

$$\begin{array}{lcl} Q_1 & \xrightarrow{\tau} \sim_{\mathrm{Ct}} & (B \bullet C)\{Q/X_1\}\,\{z := B\} \stackrel{\mathrm{def}}{=} Q_1' \\ Q_2 & \xrightarrow{\tau} \sim_{\mathrm{Ct}} & (B \bullet C)\,\{z := B\}\big\{Q\,\{z := B\}/X_1\big\} \stackrel{\mathrm{def}}{=} Q_2' \end{array}$$

and $Q_1' \sim_{\mathrm{Ct}} \mathcal{R} \sim_{\mathrm{Ct}} Q_2'$ holds by fact (b).

Case 3 $P \xrightarrow{a} G$.

Since z nsp P, it must be $a \neq z$; then proceed as in case (2) for $a \neq z$.

Case 4 $P \xrightarrow{\ell} P'$ and $\ell \neq \tau$.

Similar to previous cases. □