Electronic Notes in Theoretical Computer Science 41 No. 3 (2000)
URL: http://www.elsevier.nl/locate/entcs/volume41.html pages 50–69

First–order Semantics for Higher–order Processes

Michael Baldamus

University of Karlsruhe
Institute for Computer Design and Fault Tolerance
baldamus@ira.uka.de

Abstract

Context bisimulation is a conceptually appealing behavioural equivalence for higher–order processes. Assuming that the scoping discipline for names or channels is static, that notion can be characterized in terms of so–called normal bisimulation and in terms of π–calculus bisimulation.

This paper provides a new characterization. Specifically, late context bisimulation in the static setting is characterized as ordinary delay bisimulation. The basis of that is a new operational semantics for higher–order processes, where higher–order communication is replaced by the coupling of trigger actions. These actions are freshly generated whenever, in the original semantics, an input or output would occur. To obtain the characterization, those actions must be normalized; to achieve this normalization, in turn, a novel scheme of shifting and unshifting actions is used.

1 Introduction

The mathematical study of systems of interconnected concurrent processes has a wide range of possible sub–topics. A particularly interesting one is that of *mobility*, where processes may be relocated and/or the communication topology may change dynamically. Within the field of process algebra, the π–calculus [13] was invented to describe and analyze mobility brought about by dynamically changing communication topologies; *higher–order processes* and *mobile ambients* [6] have been introduced to describe and analyze process relocation. Here, the focus is on higher–order processes. This approach can be traced back to [3]; it was firmly established by subsequent work starting with

[1] The material presented herein was developed while the author was a member of the FLP/KIT research group at the Berlin University of Technology. The author's current work is supported by the Deutsche Forschungsgemeinschaft (German Research Society) within the project Design and Design Methodology of Embedded Systems.

[19]. The idea is to endow processes with the ability to send and receive their own kind. This stipulation renders processes first–class citizens; it is therefore justified to speak of "higher–order" processes.

Standard process algebra methodology calls for studying behavioural preorders and equivalences that are derived from operational semantics. As for higher–order processes, that obligation has been met in various ways. For example, there is the distinction between a dynamic and a static scoping discipline for names or channels. Dynamic scoping means that process passing does not invoke α–conversion, so that free names or channels of a process passed may be bound by the receiving context; static scoping means that process passing does invoke α–conversion, so that no capture of free names or channels of a process being passed can occur.

As for behavioural preorders and equivalences over higher–order processes, *context bisimulation* [1, 15, 17] is a particularly appealing equivalence. This notion avoids over–discrimination as we see it in the case of *higher–order bisimulation* (cf. [17]), a notion studied in [19] and related works. A theoretical alternative is De Nicola/Hennessy–style testing [7] applied to higher–order processes. This notion, however, can be much more complicated to deal with than bisimulation.

Assuming that the scoping discipline is static, context bisimulation can be characterized in terms of *normal bisimulation* [17] and in terms of π–calculus bisimulation [18], where the latter characterization involves a syntactic translation from higher–order to π–calculus processes. These results are interesting since they essentially reduce higher–order communication in the static setting to first–order communication. The idea is that higher–order communication in that setting can be understood as a mechanism of generating replicas of processes to be sent, where these replicas do not change location conceptually because of the *very fact* that the scoping discipline is static. All what happens besides the generation of replicas is that they are triggered by receiving contexts.

The main purpose of the present paper, then, consists of characterizing context bisimulation in a new way, namely as delay bisimulation strictly in the sense of [9]. This result still adheres to that idea of reducing higher–order to first–order communication. It is distinguished from the above–mentioned ones chiefly in that delay bisimulation according to [9] is not located at the higher–order level, such as normal bisimulation, or at the π–calculus level. Besides that simplicity, the characterization should allow one to transfer many results and techniques from the world of CCS–like processes more or less directly to higher–order processes (cf. Section 5).

The technical basis of all of that is a new type of structural operational semantics (SOS) for higher–order processes with static scoping. Most notably, the idea of reducing higher–order to first–order communication is realized by dispensing with process passing from the beginning. To this end, the SOS puts processes to be sent into *replicator contexts* instead of sending them and

substitutes *trigger processes* for processes to be received; second, it does not contain a communication rule but a *coupling rule* for *trigger actions*; third, it employs a *shifting* and *unshifting* scheme for normalizing freshly generated trigger actions. Thus, part of what is technically new herein is the on–the–fly manner of generating replicas and triggers, which is built into the SOS. Those existing characterizations in terms of normal and π–calculus bisimulation bring replication and triggering into play via the notion of bisimulation [17] or via syntactic transformation [18].

It should also be noted that the normalization of fresh *names* is an ingredient of π–calculus models. Those models always consider processes relative to name contexts, since name normalization is affected by what free names of a π–calculus process can actually become active (see, for example, [8]). The shifting and unshifting scheme introduced herein, by contrast, is not affected by anything like that; for this reason, it allows us to go about in the usual way, that is, by considering processes on their own.

The paper is organized as follows:

- Section 2 provides background by introducing a language, a standard SOS and notions of context and normal bisimulation for higher–order processes.
- Section 3 introduces the new SOS mentioned above.
- Section 4 introduces the notion of delay bisimulation over the new SOS from Section 3, compares it with the notion of normal bisimulation, states the characterization theorem and outlines that theorem's proof.
- Section 5 concludes the paper and gives an outlook on applications of the SOS and the characterization theorem. Specifically, it briefly mentions a fully abstract final coalgebra semantics for higher–order processes with static scoping. This semantics is based on the SOS from Section 3; its full abstraction property is proved with the help of the characterization theorem.

Because of space considerations, most proofs are omitted. All of them can be found in [4].

Acknowledgment

The author would like to thank the anonymous referees of this paper for all their remarks and suggestions.

2 Background

This section introduces a language, an SOS with static scoping and notions of context and normal bisimulation for higher–order processes.

$$
\begin{array}{lll}
\mathrm{T}_{\mathsf{I}}(\mathit{Var}): & P & ::= \sum_{i\in I} \mathit{pre}_i.P_i \;\Big|\; P_1 \mid P_2 \;\Big|\; P\backslash a_o \;\Big|\; !P \;\Big|\; X_{\mathsf{I}} \;\Big|\; F\langle P\rangle \\
\mathrm{Pre}(\mathit{Var}): & \mathit{pre} ::= & a_{\mathsf{I}} \mid \overline{a_{\mathsf{I}}} \mid n \mid \overline{n} \quad ; \textit{first–order prefix} \\
 & & \mid a_{\mathsf{II}}?X_{\mathsf{I}} \mid a_{\mathsf{II}}!P \quad ; \textit{higher–order prefix} \\
\mathrm{T}_{\mathsf{II}}(\mathit{Var}): & F & ::= Y_{\mathsf{II}} \mid \lambda X_{\mathsf{I}}.P
\end{array}
$$

Table 1
BNF–like grammar for higher–order processes, where I ranges over finite index sets and $o \in \{\mathsf{I}, \mathsf{II}\}$

2.1 Language

Strictly speaking, the calculus to be introduced next is a second–order calculus, since there is no exchange of abstractions. It is, nevertheless, consistent with most of the relevant literature to speak of higher–order processes. The language, then, is given in a BNF–like manner by Table 1 combined with Table 2. Suffixes of I and II are used to designate first–order and higher–order entities, respectively. Specifically, I accompanies (ordinary) first–order channels, which are used for mere synchronization, and variables of type process; II accompanies second–order channels, which are used for process passing, and variables of type process $\longrightarrow$ process. These subscripts are sometimes omitted given that they are clear form the context.

Channels and variables or order I or II must be given via two mutually disjoint, countably infinite sets ch_{I} and $\mathrm{ch}_{\mathsf{II}}$, and via two countably infinite sets $\mathit{Var}_{\mathsf{I}}$ and $\mathit{Var}_{\mathsf{II}}$. — The language contains first–order channels since they are necessary for the new SOS introduced in Section 3. Another ingredient necessitated by that SOS is the possibility of using natural numbers in first–order channel position. In this role they serve as *normalized channels*. Accordingly, it is assumed that ch_{I} and N, the set of natural numbers including zero, are disjoint. It is intentional that normalized channels can not be restricted because of that assumption.

The constructors are basically as in Sangiorgi's higher–order π–calculus (cf. [16]), without any π–calculus ingredients and without the possibility of going beyond the second order. In particular, summation must be guarded, since weak behavioural equivalences over higher–order processes must be congruences right away; it is not feasible to admit unguarded summation while considering "observation congruences" of any kind, such as it can be done within first–order frameworks (cf. [12]). Moreover, the abstraction of X_{I} in P is denoted by $\lambda X_{\mathsf{I}}.P$, and $\_\langle\_\rangle$ is an application constructor of type $((\text{process} \longrightarrow \text{process}) \times \text{process}) \longrightarrow \text{process}$.

To finish this subsection, it should be noted that $P\backslash a_o$ binds a_o in P, $o \in \{\mathsf{I}, \mathsf{II}\}$, and that $a_{\mathsf{II}}?X_{\mathsf{I}}.P$ and $\lambda X_{\mathsf{I}}.P$ bind X_{I} in P. Substitution always invokes α–conversion so that any capture of free channels or variables is prevented. The set of free first–order channels of a (first–order) term P is denoted by $\mathrm{fch}_{\mathsf{I}}(P)$.

class	definition	terminology/description	typical variable(s)
Ch_{I}	(given)	*first–order channels*	$a_{\mathsf{I}}, b_{\mathsf{I}}, \ldots$
Ch_{II}	(given)	*higher–order channels*	$a_{\mathsf{II}}, b_{\mathsf{II}}, \ldots$
$\overline{Ch_{\mathsf{I}}}$	$\{\overline{a_{\mathsf{I}}} : a_{\mathsf{I}} \in Ch_{\mathsf{I}}\}$	*complementary first–order channels*	$\overline{a_{\mathsf{I}}}, \overline{b_{\mathsf{I}}}, \ldots$
N	(given)	*normalized channels*	$n, m, \ldots$
$\overline{N}$	$\{\overline{n} : n \in N\}$	*complementary normalised channels*	$\overline{n}, \overline{m}, \ldots$
Var_o	(given)	*variables of order o*	$X_o, Y_o, \ldots$
Var	$Var_{\mathsf{I}} \cup Var_{\mathsf{II}}$	*variables*	$V, W, \ldots$
$\mathrm{T}_{\mathsf{I}}(Var)$	see Table 1	*first–order terms*	$P, Q, \ldots$
$\mathrm{Pre}(Var)$	see Table 1	*prefix terms*	*pre*
$\mathrm{T}_{\mathsf{II}}(Var)$	see Table 1	*higher–order terms*	$F, G, \ldots$
$\mathrm{T}(Var)$	$\mathrm{T}_{\mathsf{I}}(Var) \cup \mathrm{T}_{\mathsf{II}}(Var)$	*terms*	$T, U, \ldots$
$\mathrm{Subj}_{\mathsf{I}}$	$N \cup Ch_{\mathsf{I}}$	*first–order subjects*	$s_{\mathsf{I}}, t_{\mathsf{I}}, \ldots$
$\mathrm{Subj}_{\mathsf{II}}$	Ch_{II}	*higher–order subjects*	$s_{\mathsf{II}}, t_{\mathsf{II}}, \ldots$

Table 2
Syntactic classes and their typical variables, $o \in \{\mathsf{I}, \mathsf{II}\}$.

2.2 SOS

An SOS for the language from the preceding subsection is given by the axioms and rules from Table 3. It is an SOS in the style of [2, 5] in that process transmission is achieved by turning the receiving context into an abstraction and substituting this abstraction into the sending context. To this end, the axiom for higher–order output prefixing, **(out–HO)**, and the rule for communication, **(com)**, differ from their counterparts within more conventional frameworks. Specifically, **(out–HO)** sets up the output context as a term of the form $(Y_{\mathsf{II}}\langle P_1\rangle) \mid P_2$, where Y_{II} is an arbitrary (higher–order) variable, P_1 is the process send and P_2 is the output residual; **(com)** does what was already mentioned. On a technical remark, the choice of Y_{II} is arbitrary; the reason is that we only consider transitions whose source term does not contain free variables. On another technical remark, static scoping is realized by the fact that **(com)** and **(app)** invoke ordinary substitution, which means that processes substituted into input contexts and input contexts substituted into output contexts never have their free names captured.

The SOS operates on various *action sets*. Their definitions and typical

(pre) $pre.P \xrightarrow{pre} P$ if pre is not a higher–order output prefix

(out) $a_{\mathsf{II}}!P_1.P_2 \xrightarrow{a_{\mathsf{II}}!Y_{\mathsf{II}}} (Y_{\mathsf{II}}\langle P_1\rangle) \mid P_2$

(choice) $pre_k.P_k \xrightarrow{\mu} P'_k$ implies $\sum_{i\in I} pre_i.P_i \xrightarrow{\mu} P'_k$ for each $k \in I$

(par) $P_1 \xrightarrow{\mu} P'_1$ implies $P_1 \mid P_2 \xrightarrow{\mu} P'_1 \mid P_2$

(sync) $P_1 \xrightarrow{a_{\mathsf{I}}} P'_1$ and $P_2 \xrightarrow{\overline{a_{\mathsf{I}}}} P'_2$ implies $P_1 \mid P_2 \xrightarrow{\tau} P'_1 \mid P'_2$

(com) $P_1 \xrightarrow{a_{\mathsf{II}}?X_{\mathsf{I}}} P'_1$ and $P_2 \xrightarrow{a_{\mathsf{II}}!Y_{\mathsf{II}}} P'_2$ implies $P_1 \mid P_2 \xrightarrow{\tau} P'_2[P'_1[X_{\mathsf{I}}]/Y_{\mathsf{II}}]$

(res) $P \xrightarrow{\mu} P'$ implies $P\backslash a_o \xrightarrow{\mu} P'\backslash a_o$ if $a_o \notin \mathrm{ch}_o(\mu)$, $o \in \{\mathsf{I},\mathsf{II}\}$

(rep) $P|!P \xrightarrow{\mu} P'$ implies $!P \xrightarrow{\mu} P'$

(app) $P_1[P_2/X_{\mathsf{I}}] \xrightarrow{\mu} P'$ implies $(\lambda X_{\mathsf{I}}.P_1)\langle P_2\rangle \xrightarrow{\mu} P'$

Table 3
Higher–order structural axioms and rules. The symmetric counterparts of **(par)**, **(sync)** and **(com)** are not shown.

$$Ch_{\mathsf{II}}?Var_{\mathsf{I}} =_{\mathrm{def}} \{a_{\mathsf{II}}?X_{\mathsf{I}} : a_{\mathsf{II}} \in Ch_{\mathsf{II}} \text{ and } X_{\mathsf{I}} \in Var_{\mathsf{I}}\}$$

$$Ch_{\mathsf{II}}!Var_{\mathsf{II}} =_{\mathrm{def}} \{a_{\mathsf{II}}!Y_{\mathsf{II}} : a_{\mathsf{II}} \in Ch_{\mathsf{II}} \text{ and } Y_{\mathsf{II}} \in Var_{\mathsf{II}}\}$$

class	definition	typical variables
$\mathrm{Act}_{\mathsf{I}}$	$Ch_{\mathsf{I}} \cup \overline{Ch_{\mathsf{I}}} \cup N \cup \overline{N}$	$\alpha_{\mathsf{I}}, \ldots$
$\mathrm{Act}_{\mathsf{II}}$	$Ch_{\mathsf{II}}?Var_{\mathsf{I}} \cup Ch_{\mathsf{II}}!Var_{\mathsf{II}}$	$\alpha_{\mathsf{II}}, \ldots$
Act	$\mathrm{Act}_{\mathsf{I}} \cup \mathrm{Act}_{\mathsf{II}}$	$\alpha, \ldots$
$\mathrm{Act}^{\mathsf{I}}_{\tau}$	$\mathrm{Act}_{\mathsf{I}} \cup \{\tau\}$	$\mu_{\mathsf{I}}, \eta_{\mathsf{I}}, \ldots$
Act_{τ}	$\mathrm{Act} \cup \{\tau\}$	$\mu, \eta, \ldots$

$action =$	$\mathrm{ch}_o(action) =_{\mathrm{def}}$
$a_{\mathsf{I}}/\overline{a_{\mathsf{I}}}$	$\begin{cases}\{a_{\mathsf{I}}\} & \text{if } o = \mathsf{I}\\ \emptyset & \text{if } o = \mathsf{II}\end{cases}$
$n/\overline{n}$	$\emptyset$
$a_{\mathsf{II}}?X_{\mathsf{I}}/a_{\mathsf{II}}!Y_{\mathsf{II}}$	$\begin{cases}\emptyset & \text{if } o = \mathsf{I}\\ \{a_{\mathsf{II}}\} & \text{if } o = \mathsf{II}\end{cases}$
τ/ϵ	$\emptyset$

Table 4
Action classes and their typical variables and the ch_o–operator on actions, $o \in \{\mathsf{I},\mathsf{II}\}$.

variables and the definition of the $\mathrm{ch}(\_)_o$–operator, $o \in \{\mathsf{I},\mathsf{II}\}$, can be found in Table 4. It is assumed that the silent action, τ, is not contained in Ch_{I} and Ch_{II}.

2.3 Context Bisimulation

To introduce the notion of context bisimulation, the following preliminaries are required:

- The reflexive and transitive closure of $\stackrel{\tau}{\longrightarrow}$ is denoted by $\stackrel{\epsilon}{\Longrightarrow}$. Moreover, the relational product of $\stackrel{\epsilon}{\Longrightarrow}$ and $\stackrel{\mu}{\longrightarrow}$ is denoted by $\stackrel{\mu}{\Longrightarrow}$. As an aside, $\stackrel{\mu}{\Longrightarrow}$ is not defined as $\stackrel{\epsilon}{\Longrightarrow}\stackrel{\mu}{\longrightarrow}\stackrel{\epsilon}{\Longrightarrow}$ since the SOS from the preceding subsection is asymmetric. Specifically, source terms are always *processes* in the sense that they do not have free variables; residual terms of input and output transitions, on the other hand, in general do have free variables. This situation is typical within the world of higher–order processes (cf. [17], for instance).
- $\widehat{\tau} =_{\text{def}} \epsilon$, $\widehat{\alpha} =_{\text{def}} \alpha$
- A *ground substitution* is a mapping from the set of variables into the set of terms without free variables, given that first–order variables are mapped to first–order terms, and given that second–order variables are mapped to second–order terms.
- Suppose T is a term and suppose σ is a ground substitution. Then the result of simultaneously substituting every free variable of T by its image under σ is denoted by $T\sigma$.
- Suppose $\mathcal{R}$ is a binary relation on terms without free variables. Then its *open extension* to terms (with or without free variables), $\mathcal{R}^\circ$, is given by

  $$P\,\mathcal{R}^\circ\,Q \text{ if and only if } P\sigma\,\mathcal{R}\,Q\sigma \text{ for every ground substitution } \sigma.$$

  One could demand that $\mathcal{R}$ be *order respecting* in the sense that $\mathcal{R}$ be a subset of the union $(T_{\mathsf{I}}(Var) \times T_{\mathsf{I}}(Var)) \cup (T_{\mathsf{II}}(Var) \times T_{\mathsf{II}}(Var))$. Technically, however, that side condition is not necessary.
- The set of first–order terms without free variables is denoted by T_{I}.

By virtue of that entire machinery, then, the following definition is remarkably concise. It looks *almost* like weak bisimulation over CCS–like processes. What is rather different, however, is the matching condition, since it involves open extension. In particular, P' and Q' must be compared with respect to all input contexts if μ is an output action. This aspect can be regarded as the essence of context bisimulation.

Definition 2.1 A binary relation $\mathcal{R}$ on T_{I} is a *context simulation* if $P\,\mathcal{R}\,Q$ implies:

$$\text{Whenever } P \stackrel{\mu}{\longrightarrow} P' \text{ then, for some } Q',\ Q \stackrel{\hat{\mu}}{\Longrightarrow} Q' \text{ and } P'\,\mathcal{R}^\circ\,Q'.$$

If both $\mathcal{R}$ and $\mathcal{R}^{-1}$ are such simulations, then $\mathcal{R}$ is a *context* bi*simulation*. The union of all such bisimulations is denoted by $\approx_{\mathrm{Ct}}$; we call it *context bisimilarity*.

It should be noted that $\approx_{\mathrm{Ct}}$, being the union of all context bisimulations, is itself a context bisimulation. Also, $\approx_{\mathrm{Ct}}$ can be proven to be an equivalence and a congruence [17, 4].

It should be noted too that Definition 2.1 adheres to what is known as the *late* style of introducing context bisimulation. There is also an early style but

both are equivalent [17].

2.4 Normal Bisimulation

To conclude this section, we recall the notion of normal bisimulation. To this end, we need to introduce *triggers* and *replicators* formally. A trigger is defined to be a process of the form $s_{\mathsf{I}}.\mathrm{Nil}$, where s_{I} is a subject, and where Nil is the constant for inaction, which is understood to be the empty sum; a replicator is defined to be an abstraction of the form $\lambda X_{\mathsf{I}}.(!\overline{s_{\mathsf{I}}}.X_{\mathsf{I}})$. Such constructs are denoted by Tr_s or Rep_s, respectively, calling s_{I} the subject of the trigger or replicator.

The idea of normal bisimulation consists of simplifying the matching conditions for input and output actions drastically. Instead of respectively comparing the residuals with respect to all inputs or input contexts, they are only compared with respect to one trigger as input or one replicator as input context.

Definition 2.2 A binary relation $\mathcal{R}$ on T_{I} is a *normal bisimulation* if $P\,\mathcal{R}\,Q$ implies:

i. Whenever $P \xrightarrow{\mu_{\mathsf{I}}} P'$ then, for some Q', $Q \xRightarrow{\widehat{\mu_{\mathsf{I}}}} Q'$ and $P'\,\mathcal{R}\,Q'$.

ii. Whenever $P \xrightarrow{a_{\mathsf{II}}?X_{\mathsf{I}}} P'$ then, for some Q' and some b_{I} with $b_{\mathsf{I}} \notin \mathrm{fch}_{\mathsf{I}}(P', Q')$, $Q \xRightarrow{a_{\mathsf{II}}?X_{\mathsf{I}}} Q'$ and $P'[\mathrm{Tr}_{b_{\mathsf{I}}}/X_{\mathsf{I}}]\,\mathcal{R}\,Q'[\mathrm{Tr}_{b_{\mathsf{I}}}/X_{\mathsf{I}}]$.

iii. Whenever $P \xrightarrow{a_{\mathsf{II}}!Y_{\mathsf{II}}} P'$ then, for some Q' and some b_{I} with $b_{\mathsf{I}} \notin \mathrm{fch}_{\mathsf{I}}(P', Q')$, $Q \xRightarrow{a_{\mathsf{II}}!Y_{\mathsf{II}}} Q'$ and $P'[\mathrm{Rep}_{b_{\mathsf{I}}}/Y_{\mathsf{II}}]\,\mathcal{R}\,Q'[\mathrm{Rep}_{b_{\mathsf{I}}}/Y_{\mathsf{II}}]$.

If both $\mathcal{R}$ and $\mathcal{R}^{-1}$ are such simulations, then $\mathcal{R}$ is a *normal* bi*simulation*. The union of all such bisimulations is denoted by $\approx_{\mathrm{Nr}}$; we call it *normal bisimilarity*.

The characterization is simply as follows (cf. [17]):

Theorem 2.3 *(Sangiorgi)* $\approx_{\mathrm{Ct}} \;=\; \approx_{\mathrm{Nr}}$

3 New SOS for Higher–Order Processes

In the previous section it has been recalled what a classic SOS for higher–order processes with static scoping may look like; this section introduces the new SOS mentioned in the introduction. This SOS is called the *alternative SOS*. On a preliminary note, a trigger or replicator is *connected* if its subject is a restricted channel.

As a first approximation to the alternative SOS, one might consider replacing the axiom for higher–order output — $a_{\mathsf{II}}!P_1.P_2 \xrightarrow{a_{\mathsf{II}}!Y_{\mathsf{II}}} (\,Y_{\mathsf{II}}\langle P_1\rangle) \mid P_2$ — by

$$a_{\mathsf{II}}!P_1.P_2 \xrightarrow{a_{\mathsf{II}}!Y_{\mathsf{II}}} (\,Y_{\mathsf{II}}\langle \mathrm{Tr}_b\rangle \mid \mathrm{Rep}_b\langle P_1\rangle)\backslash b \mid P_2,$$

where $b_{\mid}$ is fresh from $Ch_{\mid}$. This measure has already the effect of reducing higher–order to trigger communication: Instead of sending P_1, the trigger Tr_b is sent, P_1 is left behind attached to a replicator, and both the trigger and the replicator are properly connected by making their common subject private to them. Adopting this axiom in favor of **(out–HO)**, however, is not enough if one wants to characterize context bisimulation as done below; the reason is that the ensuing SOS still involves process passing, albeit on an extremely reduced scale. To solve this problem, process passing is abolished altogether. The SOS does so by moving the creation of triggers to the input axiom and supplanting the communication rule by a *connection rule* which, predictably, does nothing than to connect triggers and replicators.

Moreover, all subjects of freshly created and not yet connected triggers and replicators are normalized. Whenever such a subject is needed, then each natural contained in the process term in question is increased by one and zero is taken. The reverse operation takes place whenever the connection rule is applied. We call these operations *shift* and *unshift*.

Definition 3.1 The shift, $\gg$, and the unshift, $\ll$, of a term T are given as follows: $\gg$ replaces every n or $\overline{n}$ that occurs in T respectively by $n+1$ or $\overline{n+1}$; $\ll$ replaces every n or $\overline{n}$ that occurs in T respectively by $n-1$ or $\overline{n-1}$. As a side condition, $\ll$ can be only applied to T if it contains neither 0 nor $\overline{z}$.

We continue with going through the structural axioms and rules that make up the core of the alternative SOS. This group includes axioms for higher–order input **(in–At)** and output **(out–At)**, and a connection rule **(con–At)**. A rule for higher–order interleaving **(par2–At)** is its fourth member.

(in–At) $\boxed{a_{\mid\mid}?X_{\mid}.P \xrightarrow{a_{\mid\mid}?}_{\mathrm{At}} (\gg(P))[\mathrm{Tr}_0/X_{\mid}]}$

This axiom states that an input prefix is executed by shifting the receiving context and substituting the trigger Tr_0 for the process variable. In other words, the input of a process via $a_{\mid\mid}$ is explained in terms of a *preemptive input* of the trigger Tr_0. For example, we have

$$a?X.b?Y.(X \mid Y) \xrightarrow{a?}_{\mathrm{At}} b?Y.(\mathrm{Tr}_0 \mid Y) \xrightarrow{b?}_{\mathrm{At}} \mathrm{Tr}_1 \mid \mathrm{Tr}_0.$$

(out–At) $\boxed{a_{\mid\mid}!P_1.P_2 \xrightarrow{a_{\mid\mid}!}_{\mathrm{At}} \mathrm{Rep}_0\langle\gg(P_1)\rangle \mid \gg(P_2)}$

This axiom states that an output prefix is executed by shifting the process sent, applying the replicator Rep_0 to the result, shifting the residual and putting both in parallel. For example, we have

$$a!(c.\mathrm{Nil}).b!(\,\mathrm{d.Nil}) \xrightarrow{a!}_{\mathrm{At}} \mathrm{Rep}_0\langle c.\mathrm{Nil}\rangle|b!(\,\mathrm{d.Nil}) \xrightarrow{b!}_{\mathrm{At}} \mathrm{Rep}_1\langle c.\mathrm{Nil}\rangle|\mathrm{Rep}_0(\,\mathrm{d.Nil}),$$

using **(out–At)** and standard interleaving (rule **par2–At** from Table 5).

(par2–At)
$$\frac{P_1 \xrightarrow{\alpha_{\mathsf{II}}}_{\mathrm{At}} P_1'}{P_1 \mid P_2 \xrightarrow{\alpha_{\mathsf{II}}}_{\mathrm{At}} P_1' \mid \gg(P_2)}$$

This rule states that higher–order actions are propagated over parallel composition by shifting the inactive sub–process. As an example, one might consider what may happen if the processes from the previous two examples are put in parallel:

$$\begin{aligned}(1)\quad & a?X.b?\,Y.(X \mid Y) \mid a!(c.\mathrm{Nil}).b!(\,\mathrm{d.Nil}) \\ & \xrightarrow{a!}_{\mathrm{At}} a?X.b?\,Y.(X \mid Y) \mid (\mathrm{Rep}_0\langle c.\mathrm{Nil}\rangle \mid b!(\,\mathrm{d.Nil})) \\ & \xrightarrow{a?}_{\mathrm{At}} b?\,Y.(\mathrm{Tr}_0 \mid Y) \mid (\mathrm{Rep}_1\langle c.\mathrm{Nil}\rangle \mid b!(\,\mathrm{d.Nil}))\end{aligned}$$

The second transition is the interesting one, because its inference involves shifting the sub–process $\mathrm{Rep}_0\langle c.\mathrm{Nil}\rangle \mid b!(\,\mathrm{d.Nil})$.

(con–At)
$$\frac{P_1 \xrightarrow{a_{\mathsf{II}}?}_{\mathrm{At}} P_1' \quad P_2 \xrightarrow{a_{\mathsf{II}}!}_{\mathrm{At}} P_2'}{P_1 \mid P_2 \xrightarrow{\tau}_{\mathrm{At}} \ll\big((P_1'[b_{\mathsf{I}}/0] \mid P_2'[b_{\mathsf{I}}/0])\backslash b_{\mathsf{I}}\big)}$$

for $b_{\mathsf{I}} \notin \mathrm{fch}_{\mathsf{I}}(P_1', P_2')$. This rule states in which way the triggers and the replicator created by complementary input and output actions are connected: Their subject 0 is replaced by a fresh first–order channel b_{I}, this channel is made private to the triggers and the replicator, and the ensuing process is unshifted.

As an example, one might consider the following extension of (1):

$$\begin{aligned}& a?X.b?\,Y.(X \mid Y) \mid a!(c.\mathrm{Nil}).b!(\,\mathrm{d.Nil}) \\ & \xrightarrow{a!}_{\mathrm{At}} a?X.b?\,Y.(X \mid Y) \mid (\mathrm{Rep}_0\langle c.\mathrm{Nil}\rangle \mid b!(\,\mathrm{d.Nil})) \\ & \xrightarrow{a?}_{\mathrm{At}} b?\,Y.(\mathrm{Tr}_0 \mid Y) \mid (\mathrm{Rep}_1\langle c.\mathrm{Nil}\rangle \mid b!(\,\mathrm{d.Nil})) \\ & \xrightarrow{\tau}_{\mathrm{At}} \big((\mathrm{Tr}_0 \mid \mathrm{Tr}_e) \mid (\mathrm{Rep}_1\langle c.\mathrm{Nil}\rangle \mid \mathrm{Rep}_e\langle\,\mathrm{d.Nil}\rangle)\big)\backslash e,\end{aligned}$$

where e is fresh. The third transition is the interesting one, because it is inferred from

$$b?\,Y.(\mathrm{Tr}_0 \mid Y) \xrightarrow{b?}_{\mathrm{At}} \mathrm{Tr}_1 \mid \mathrm{Tr}_0 \text{ and}$$
$$\mathrm{Rep}_1\langle c.\mathrm{Nil}\rangle \mid b!(\,\mathrm{d.Nil}) \xrightarrow{b!}_{\mathrm{At}} \mathrm{Rep}_2\langle c.\mathrm{Nil}\rangle \mid \mathrm{Rep}_1\langle\,\mathrm{d.Nil}\rangle.$$

This remark concludes the description of the specific aspects of the alternative SOS. All other axioms and rules remain unchanged. The complete set can be found in Table 5.

(pre–At)	$\alpha_{\mathsf{I}}.P \xrightarrow{\alpha_{\mathsf{I}}}_{\mathrm{At}} P$
(in–At)	$a_{\mathsf{II}}?X_{\mathsf{I}}.P \xrightarrow{a_{\mathsf{II}}?X_{\mathsf{I}}}_{\mathrm{At}} (\gg(P))[\mathrm{Tr}_0/X_{\mathsf{I}}]$, where $\mathrm{Tr}_0 =_{(\mathrm{def})} 0.\mathrm{Nil}$
(out–At)	$a_{\mathsf{II}}!P_1.P_2 \xrightarrow{a_{\mathsf{II}}!Y_{\mathsf{II}}}_{\mathrm{At}} \mathrm{Rep}_0\langle\gg(P_1)\rangle \mid \gg(P_2)$, where $\mathrm{Rep}_0 =_{(\mathrm{def})} (!\overline{z}.X_{\mathsf{I}})[X_{\mathsf{I}}]$
(choice–At)	$\pi_k.P_k \xrightarrow{\mu}_{\mathrm{At}} P'_k$ implies $\sum_{i\in\mathsf{I}} \pi_i.P_i \xrightarrow{\mu}_{\mathrm{At}} P'_k$ for each $k \in I$
(par1–At)	$P_1 \xrightarrow{\mu_{\mathsf{I}}}_{\mathrm{At}} P'_1$ implies $P_1 \mid P_2 \xrightarrow{\mu_{\mathsf{I}}}_{\mathrm{At}} P'_1 \mid P_2$
(par2–At)	$P_1 \xrightarrow{\alpha_{\mathsf{II}}}_{\mathrm{At}} P'_1$ implies $P_1 \mid P_2 \xrightarrow{\alpha_{\mathsf{II}}}_{\mathrm{At}} P'_1 \mid \gg(P_2)$
(sync–At)	$P_1 \xrightarrow{a_{\mathsf{I}}}_{\mathrm{At}} P'_1$ implies $P_2 \xrightarrow{\overline{a_{\mathsf{I}}}}_{\mathrm{At}} P'_2$ and $P_1 \mid P_2 \xrightarrow{\tau}_{\mathrm{At}} P'_1 \mid P'_2$
(con–At)	$P_1 \xrightarrow{a_{\mathsf{II}}?X_{\mathsf{I}}}_{\mathrm{At}} P'_1$ and $P_2 \xrightarrow{a_{\mathsf{II}}!Y_{\mathsf{II}}}_{\mathrm{At}} P'_2$ implies $\ll\big((P'_1[b_{\mathsf{I}}/0] \mid P'_2[b_{\mathsf{I}}/0])\backslash b_{\mathsf{I}}\big)$ for $b_{\mathsf{I}} \notin \mathrm{fch}_{\mathsf{I}}(P'_1, P'_2)$
(res-At)	$P \xrightarrow{\mu}_{\mathrm{At}} P'$ implies $P\backslash a_o \xrightarrow{\mu}_{\mathrm{At}} P'\backslash a_o$, if $a_o \notin \mathrm{ch}_o(\mu)$, $o \in \{\mathsf{I}, \mathsf{II}\}$
(rep–At)	$P\mid !P \xrightarrow{\mu}_{\mathrm{At}} P'$ and $!P \xrightarrow{\mu}_{\mathrm{At}} P'$
(appl–At)	$P_1[P_2/X_{\mathsf{I}}] \xrightarrow{\mu}_{\mathrm{At}} P'$ implies $(\lambda X_{\mathsf{I}}.P_1)\langle P_2\rangle \xrightarrow{\mu}_{\mathrm{At}} P'$

Table 5
Alternative structural axioms and rules. The symmetric counterparts of **(par1–At)**, **(par2–At)**, **(sync–At)** and **(con–At)** not shown. Note that input and output actions are endowed with variables, just like input and output actions within the context of the SOS from Section 2. The axioms and rules would work just as well if this endowment was simply omitted. Keeping track of those variables is, however, necessary for proving that context bisimulation and delay bisimulation over the alternative semantics are equivalent. By using the same action format for both kinds of semantics this necessity is taken account of in the easiest way.

4 Charaterization of Late Context Bisimulation

The previous section has been concerned with introducing what we have called the alternative SOS for higher–order processes with static scoping; this section is initially concerned with stating the characterization of context bisimulation in terms of delay bisimulation over that new SOS. The main part of this section then gives an outline of how that characterization can be proven.

4.1 The Theorem

To begin with, the notion of delay bisimulation according to [9] is instantiated to the alternative transition semantics. This step is possible since that semantics can meaningfully be regarded as a labelled transition system, also in the sense of [9]. We call the ensuing equivalence *alternative bisimilarity*. As

for preliminaries, $\stackrel{\epsilon}{\Longrightarrow}_{\mathrm{At}}$ denotes the reflexive and transitive closure of $\stackrel{\tau}{\Longrightarrow}_{\mathrm{At}}$; $\stackrel{\mu}{\Longrightarrow}_{\mathrm{At}}$ denotes the relational product of $\stackrel{\epsilon}{\Longrightarrow}_{\mathrm{At}}$ and $\stackrel{\mu}{\longrightarrow}_{\mathrm{At}}$.

Definition 4.1 A binary relation $\mathcal{R}$ on T_{I} is an *alternative simulation* if $P\,\mathcal{R}\,Q$ implies:

$$\text{Whenever } P \stackrel{\mu}{\longrightarrow}_{\mathrm{At}} P' \text{ then, for some } Q',\ Q \stackrel{\hat{\mu}}{\Longrightarrow}_{\mathrm{At}} Q' \text{ and } P'\,\mathcal{R}\,Q'.$$

If both $\mathcal{R}$ and $\mathcal{R}^{-1}$ are such simulations, then $\mathcal{R}$ is an *alternative* bi*simulation*. We denote the union of all alternative bisimulations by $\approx_{\mathrm{At}}$ and call it *alternative bisimulation bisimilarity*.

Predictably, alternative bisimilarity is itself an alternative bisimulation. Also, the following lemma is a prerequisite for proving the characterization theorem:

Lemma 4.2 *Alternative bisimilarity is a congruence.*

This property can be proven directly, by adapting and extending what is known as Howe's method (see [4] for the proof and [11] for that method's original presentation). The congruence property of normal bisimulation does not have any known direct proof (cf. [17]), so the existence of a direct proof of Lemma 4.2 may be rather surprising. One has to bear in mind, however, that normal bisimulation is defined over the standard SOS for higher–order processes, which involves term substitution. The bisimulation game is, at the same time, very much restricted, and this situation seems to be the reason why a direct congruence proof for normal bisimulation is so difficult. The alternative SOS does not involve term substitution; this fact renders the direct proof of Lemma 4.2 possible.

Another important difference between normal and alternative bisimulation is that the normal bisimulation game involves choosing channels that are fresh with respect to residuals. This aspect may appear like a small technicality but in fact it becomes a significant issue as soon as questions such as fully abstract denotational semantics become interesting. Specifically, the naive idea would be to obtain "normal" process unfoldings on the basis of rendering every transition of the from $P \stackrel{a_{\mathsf{II}}?X_{\mathsf{I}}}{\longrightarrow} P'$ as $P \stackrel{a_{\mathsf{II}}?}{\longrightarrow} P'[\mathrm{Tr}_{b_{\mathsf{I}}}/X_{\mathsf{I}}]$, and on the basis of rendering every transition of the form $P \stackrel{a_{\mathsf{II}}!\,Y_{\mathsf{II}}}{\longrightarrow} P'$ as $P \stackrel{a_{\mathsf{II}}!}{\longrightarrow} P'[\mathrm{Rep}_{b_{\mathsf{I}}}/Y_{\mathsf{II}}]$, where $b_{\mathsf{I}} \notin \mathrm{fch}_{\mathsf{I}}(P')$ in both cases. The ensuing semantics would not be fully abstract, since bisimilar processes do not necessarily have identical sets of free channels. Analogous situations can be found within the framework of the π–calculus. They are one reason why fully abstract denotational models for the π–calculus took so much longer to develop and are rather different from models for CCS–like processes (see [8], for instance). The alternative behavioral semantics sidesteps the problem. On its basis, it is then possible to interpret higher–order processes so that they are reduced to first–order entities to what seems to be the greatest possible degree (cf. [4] and Section 5).

To conclude this subsection, what is left is to state the characterization of context in terms of alternative bisimilarity:

Theorem 4.3 $\approx_{\mathrm{Ct}} \;=\; \approx_{\mathrm{At}}$

4.2 Outline of the Proof of the Theorem

4.2.1 Prerequisites

Output Residuals

The first prerequisite of the proof of Theorem 4.3 consists of making explicit what terms can occur as output residuals of transitions generated by the SOS from Section 2. We denote individual output residuals by O and the entire set of output residuals over some variable Y by $\mathrm{OR}(Y)$. This set is given by

$$O ::= Y\langle P_1\rangle \mid P_2 \;\Big|\; O \mid P \;\Big|\; P \mid O \;\Big|\; O\backslash a_o \text{ for } P_1, P_2, P \in \mathrm{T}_{\mathsf{I}} \text{ and } o \in \{\mathsf{I}, \mathsf{II}\}.$$

The following lemma validates this definition: part 1 states that each residual of an output action of the form $a!Y$ is an element of $\mathrm{OR}(Y)$; part 2 states that each element of $\mathrm{OR}(Y)$, where Y is an arbitrary second–order variable, is a possible output residual.

Lemma 4.4

(i) $P \xrightarrow{a!Y} P'$ *implies* $P' \in \mathrm{OR}(Y)$

(ii) *For each* $a \in Ch_{\mathsf{II}}$, $O \in \mathrm{OR}(Y)$ *implies* $P \xrightarrow{a!Y} O$, *where* P *is the process that results from replacing the subterm* $Y\langle P_1\rangle \mid P_2$ *of* O *by* $a!P_1.P_2$.

Factorization

The idea of factorization, which is due to Sangiorgi [17], is the most decisive prerequisite. Here, it can be seen as basically stating that Q can be *factored out* of every context of the form $P[Q/X]$, up to context bisimulation, using triggers and a replicator. The theorem can also be shown if context bisimulation is replaced by alternative bisimulation. In this capacity, we call it the *alternative factorization theorem* (AFT).

Theorem 4.5 *(Factorization) Suppose* $\mathcal{R} \in \{\approx_{\mathrm{Ct}}, \approx_{\mathrm{At}}\}$.

1. *Let* $\lambda X.P \in \mathrm{T}_{\mathsf{II}}$ *and* $Q \in \mathrm{T}_{\mathsf{I}}$. *Then*

$$P[Q/X]\ \mathcal{R}\ \Big(P[\mathrm{Tr}_a/X] \mid \mathrm{Rep}_a\langle Q\rangle\Big)\backslash a \text{ whenever } a \in Ch_{\mathsf{I}} \setminus \mathrm{fch}_{\mathsf{I}}(P, Q).$$

2. *Let* $Y \in Var_{\mathsf{II}}$, $O \in \mathrm{OR}(Y)$, $X \in Var_{\mathsf{I}}$, *and* $P \in \mathrm{T}_{\mathsf{I}}(X)$, *the set of first–order terms with free variables in* $\{X\}$. *Then*

$$O[\lambda X.P/Y]\ \mathcal{R}\ \Big(O[\mathrm{Rep}_a/Y] \mid P[\mathrm{Tr}_a/X]\Big)\backslash a \text{ whenever } a \in Ch_{\mathsf{I}} \setminus \mathrm{fch}_{\mathsf{I}}(P, Q).$$

Bisimulation–Up–To

Another, rather more technical prerequisite is the up–to method of proving bisimilarity. Because this method is so well known, it suffices to state the relevant definitions and lemmas right away. There are only small adaptions to the framework of the present paper.

Definition 4.6 A binary relation $\mathcal{R}$ on T_{I} is a *context simulation up to* $\approx_{\mathrm{Ct}}$ if $P\,\mathcal{R}\,Q$ implies:

$$\text{Whenever } P \xrightarrow{\mu} P' \text{ then, for some } Q',\ Q \stackrel{\hat{\mu}}{\Longrightarrow} Q' \text{ and } P'\ (\mathcal{R}{\approx_{\mathrm{Ct}}})^{\circ}\ Q'.$$

If both $\mathcal{R}$ and $\mathcal{R}^{-1}$ are such simulations, then $\mathcal{R}$ is a *context* bi*simulation up to* $\approx_{\mathrm{Ct}}$.

Lemma 4.7 *For every binary relation* $\mathcal{R}$ *on* T_{I} *that is a context bisimulation up to* $\approx_{\mathrm{Ct}}$*, we have* $\mathcal{R} \subseteq \approx_{\mathrm{Ct}}$.

Definition 4.8 A binary relation $\mathcal{R}$ on T_{I} is an *alternative simulation up to* $\approx_{\mathrm{At}}$ if $P\,\mathcal{R}\,Q$ implies:

$$\text{Whenever } P \xrightarrow{\mu}_{\mathrm{At}} P' \text{ then, for some } Q',\ Q \stackrel{\hat{\mu}}{\Longrightarrow}_{\mathrm{At}} Q' \text{ and } P'\,\mathcal{R}{\approx_{\mathrm{At}}}\,Q'.$$

If both $\mathcal{R}$ and $\mathcal{R}^{-1}$ are such simulations, then $\mathcal{R}$ is an *alternative* bi*simulation up to* $\approx_{\mathrm{At}}$.

Lemma 4.9 *For every binary relation* $\mathcal{R}$ *on* T_{I} *that is an alternative bisimulation up to* $\approx_{\mathrm{At}}$*, we have* $\mathcal{R} \subseteq \approx_{\mathrm{At}}$.

4.2.2 Operational Correspondence

Lemma 4.9 concludes the prerequisites. Turning to the main part of the proof, we first show a correspondence result for the standard and alternative transition semantics, which is as follows:

- Every visible standard transition can be transformed into an alternative transition with the same label, and conversely.
- – For every invisible standard transition, there is an invisible alternative transition so that the residuals are alternatively bisimilar.
  – Conversely, for every invisible alternative transition, there is an invisible standard transition so that the residuals are context bisimilar.

Using this result and the factorization theorems for context and alternative bisimulation, we show, afterward, that alternative bisimulation is a context bisimulation up to $\approx_{\mathrm{Ct}}$ and, conversely, that context bisimulation is an alternative bisimulation up to $\approx_{\mathrm{At}}$. The equivalence theorem is an immediate consequence of that.

As a technical preliminary for the correspondence lemma, substitutions of the form $[X/\mathrm{Tr}_0]$ and of the form $[Y/\mathrm{Rep}_0]$ need to be introduced, where

$X \in Var_{\mathrm{I}}$ and $Y \in Var_{\mathrm{II}}$. It is assumed that α–conversion is invoked so that X and Y are not accidentally captured.

Lemma 4.10

1. *i.* $P \xrightarrow{\alpha_{\mathrm{I}}} P'$ *implies* $P \xrightarrow{\alpha_{\mathrm{I}}}_{\mathrm{At}} P'$
   ii. $P \xrightarrow{a?X} P'$ *implies* $P \xrightarrow{a?X}_{\mathrm{At}} (\gg(P'))[\mathrm{Tr}_0/X]$
   iii. $P \xrightarrow{a!Y} P'$ *implies* $P \xrightarrow{a!Y}_{\mathrm{At}} (\gg(P'))[\mathrm{Rep}_0/Y]$
   iv. $P \xrightarrow{\tau} P'$ *implies, for some* $P^\sharp$, $P \xrightarrow{\tau}_{\mathrm{At}} P^\sharp$ *and* $P' \approx_{\mathrm{At}} P^\sharp$
2. *i.* $P \xrightarrow{\alpha_{\mathrm{I}}}_{\mathrm{At}} P'$ *implies* $P \xrightarrow{\alpha_{\mathrm{I}}} P'$
   ii. $P \xrightarrow{a?X}_{\mathrm{At}} P'$ *implies* $P \xrightarrow{a?X} \ll(P'[X/\mathrm{Tr}_0])$
   iii. $P \xrightarrow{a!Y}_{\mathrm{At}} P'$ *implies* $P \xrightarrow{a!Y} \ll(P'[Y/\mathrm{Rep}_0])$
   iv. $P \xrightarrow{\tau}_{\mathrm{At}} P'$ *implies, for some* $P^\sharp$, $P \xrightarrow{\tau} P^\sharp$ *and* $P' \approx_{\mathrm{Ct}} P^\sharp$

Proof. (Idea) By transition induction. For details see [4]. □

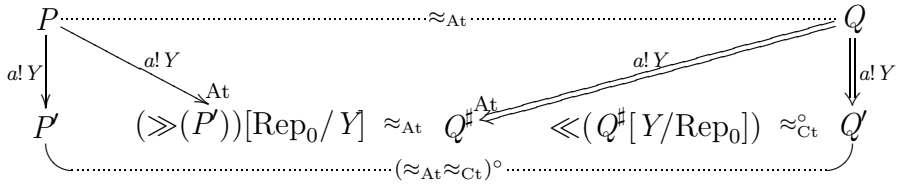
Note that transforming a standard transition with a higher–order label into an alternative transition requires to shift the residual. Conversely, transforming an alternative transition with a higher–order label requires to unshift the residual. Also, the correspondence for silent transitions is weaker than in the other cases, as such transitions may hide communication or connection steps. Specifically, the correspondence for silent steps requires to bring to bear the appropriate factorization theorem. This situation allows one to relate communication to connection only up to bisimilarity.

4.2.3 Concluding the Proof

Proof. (Theorem 4.3, $\approx_{\mathrm{Ct}} = \approx_{\mathrm{At}}$) "$\supseteq$": To prove this inclusion, we show that $\approx_{\mathrm{At}}$ is a simulation up to $\approx_{\mathrm{Ct}}$. Then, by symmetry, $\approx_{\mathrm{At}}$ is a *bi*simulation up to $\approx_{\mathrm{Ct}}$. The conclusion, $\approx_{\mathrm{Ct}} \supseteq \approx_{\mathrm{At}}$, follows by Lemma 4.7.

So suppose $P \approx_{\mathrm{At}} Q$ and $P \xrightarrow{\mu} P'$. We consider only the case of $\mu = a!Y$ because those of $\mu = \alpha_{\mathrm{I}}$, $\mu = a?X$, and $\mu = \tau$ are easier or very similar. In this case, by Lemma 4.10(1,iii), $P \xrightarrow{a!Y}_{\mathrm{At}} P^\sharp$ with $P^\sharp = (\gg(P'))[\mathrm{Rep}_0/Y]$ and further, by $P \approx_{\mathrm{At}} Q$, $Q \xRightarrow{a!Y}_{\mathrm{At}} Q^\sharp$ with $P^\sharp \approx_{\mathrm{At}} Q^\sharp$ for some $Q^\sharp$. The remaining steps are as follows:

1. We construct a transition $Q \xRightarrow{a!Y} Q'$ so that $\ll(Q^\sharp[Y/\mathrm{Rep}_0]) \approx^\circ_{\mathrm{Ct}} Q'$. (rightmost arrow in the diagram below)
2. We show $P' (\approx_{\mathrm{At}}\approx_{\mathrm{Ct}})^\circ Q'$, thereby concluding the proof. (dotted line in the diagram below)



1. As for this step, we distinguish whether $Q \stackrel{a!\,Y}{\Longrightarrow}_{\mathrm{At}} Q^\sharp$ is actually a strong transition of the form $Q \stackrel{a!\,Y}{\longrightarrow}_{\mathrm{At}} Q^\sharp$ or a truly weak one of the form

$$Q \stackrel{\tau}{\longrightarrow}_{\mathrm{At}} Q_1 \stackrel{\tau}{\longrightarrow}_{\mathrm{At}} \ldots \stackrel{\tau}{\longrightarrow}_{\mathrm{At}} Q_l \stackrel{a!\,Y}{\longrightarrow}_{\mathrm{At}} Q^\sharp, \text{ where } k \geq 1.$$

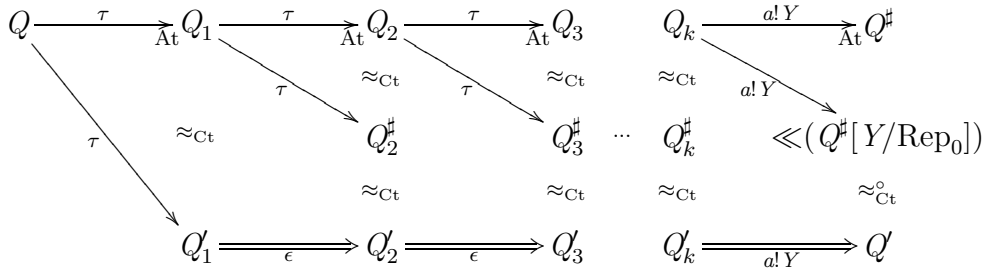
   In the first case the desired conclusion is immediate by Lemma 4.10(2,iii), in the second we show, first, that there exists a Q'_l so that $Q \stackrel{\epsilon}{\Longrightarrow} Q'_l$ and $Q_l \approx_{\mathrm{Ct}} Q'_l$ for each $l \in \{1, \ldots, k\}$. We do so by induction on l:

   $\underline{l = 1}$: Immediate by Lemma 4.10(2,vi). Note that $Q \stackrel{\epsilon}{\Longrightarrow} Q'_1$ is, therefore, actually a strong transition of the form $Q \stackrel{\tau}{\longrightarrow} Q'_1$.

   $\underline{l \to l+1}$: Suppose there exists a Q'_l so that $Q \stackrel{\epsilon}{\Longrightarrow} Q'_l$ and $Q_l \approx_{\mathrm{Ct}} Q'_l$. We have $Q_l \stackrel{\tau}{\longrightarrow}_{\mathrm{At}} Q_{l+1}$, whence Lemma 4.10(2,vi) implies that there exists a $Q^\sharp_{l+1}$ so that $Q_l \stackrel{\tau}{\longrightarrow} Q^\sharp_{l+1}$ and $Q_{l+1} \approx_{\mathrm{Ct}} Q^\sharp_{l+1}$. Then, because $\approx_{\mathrm{Ct}}$ is a bisimulation, there exists a Q'_{l+1} so that $Q'_l \stackrel{\epsilon}{\Longrightarrow} Q'_{l+1}$ and $Q^\sharp_{l+1} \approx_{\mathrm{Ct}} Q'_{l+1}$. This fact implies $Q \stackrel{\epsilon}{\Longrightarrow} Q'_{l+1}$ in combination with $Q_{l+1} \approx_{\mathrm{Ct}} Q'_{l+1}$.

   In sum, there exists a Q'_k so that $Q \stackrel{\epsilon}{\Longrightarrow} Q'_k$ and $Q_k \approx_{\mathrm{Ct}} Q'_k$. Obtaining a Q' so that $Q \stackrel{a!\,Y}{\Longrightarrow} Q'$ and $\ll(Q^\sharp[\,Y/\mathrm{Rep}_0]) \approx^\circ_{\mathrm{Ct}} Q'$ is now essentially a matter of repeating the induction step above, this time appealing to Lemma 4.10(2,iii).

   The following diagram depicts this argument for $k \geq 3$.



2. This step consists of closing up the bisimulation up to $\approx_{\mathrm{Ct}}$ or, in other words, of meeting the proof obligation $P'[\lambda X.R/\,Y] \approx_{\mathrm{At}} \approx_{\mathrm{Ct}} Q'[\lambda X.R/\,Y]$ for each $\lambda X.R \in \mathrm{T}_{\mathsf{II}}$. Our main tools for doing so are AFT and the (ordinary) factorization theorem. To apply them, we need the following technical prerequisites:
   a. Let b be fresh from Ch_{I}.
   b. We have $P'[\mathrm{Rep}_b/\,Y] = \ll\Big((\gg(P'))[\mathrm{Rep}_0/\,Y][b/0]\Big)$ and, as stated at the beginning of the proof, $(\gg(P'))[\mathrm{Rep}_0/\,Y] \approx_{\mathrm{At}} Q^\sharp$. Hence, $P'[\mathrm{Rep}_b/\,Y] \approx_{\mathrm{At}} \ll(Q^\sharp[b/0])$ since $\approx_{\mathrm{At}}$ is preserved by subject replacement.
   c. By the alternative SOS, 0 occurs in $Q^\sharp$ exactly once, namely in the replicator sub–term Rep_0. Hence, $\ll(Q^\sharp[b/0]) =$

$(\ll(Q^\sharp[\,Y/\mathrm{Rep}_0]))[\mathrm{Rep}_b/\,Y]$.

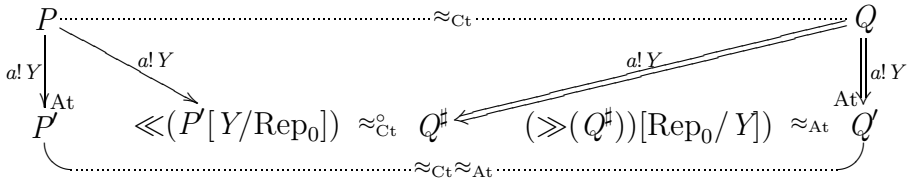
Then the final reasoning is as follows:

$$
\begin{aligned}
& P[\lambda X.R/\,Y] \\
\approx_{\mathrm{At}}\ & (P[\mathrm{Rep}_b/\,Y] \mid R[\mathrm{Tr}_b/X])\backslash b && \text{; AFT, using (a)} \\
\approx_{\mathrm{At}}\ & (\ll(Q^\sharp[b/0]) \mid R[\mathrm{Tr}_b/X])\backslash b && \text{; (b), Lemma 4.2} \\
=\ & \Big((\ll(Q^\sharp[\,Y/\mathrm{Rep}_0]))[\mathrm{Rep}_b/\,Y] \mid R[\mathrm{Tr}_b/X]\Big)\backslash b && \text{; (c)} \\
\approx_{\mathrm{Ct}}\ & (\ll(Q^\sharp[\,Y/\mathrm{Rep}_0]))[\lambda X.R/Y] && \text{; factorization theorem} \\
\approx_{\mathrm{Ct}}\ & Q'[\lambda X.R/Y] && \text{; (1)}
\end{aligned}
$$

"$\subseteq$": The proof of this inclusion is largely complementary to that of the other one or, in other words, we show $\approx_{\mathrm{Ct}}$ to be an alternative bisimulation up to $\approx_{\mathrm{At}}$. The conclusion, $\approx_{\mathrm{Ct}} \subseteq \approx_{\mathrm{At}}$, then follows by symmetry and Lemma 4.9.

So suppose $P \approx_{\mathrm{Ct}} Q$ and $P \xrightarrow{\mu}_{\mathrm{At}} P'$. Again we consider only the case of $\mu = a!\,Y$ since the other ones are easier or very similar. In this case, by Lemma 4.10(2,iii), $P \xrightarrow{a!\,Y} P^\sharp$ with $P^\sharp = \ll(P'[\,Y/\mathrm{Rep}_0])$ and further, by $P \approx_{\mathrm{Ct}} Q$, $Q \stackrel{a!\,Y}{\Longrightarrow} Q^\sharp$ with $P^\sharp \approx^\circ_{\mathrm{Ct}} Q^\sharp$ for some $Q^\sharp$. The remaining steps are analogous to (1) and (2) in the proof of "$\supseteq$":

1. We construct a transition $Q \xrightarrow{a!\,Y}_{\mathrm{At}} Q'$ so that $(\gg(Q^\sharp))[\mathrm{Rep}_0/\,Y] \approx_{\mathrm{At}} Q'$. (rightmost arrow in the diagram below)
2. We show $P' \approx_{\mathrm{At}}\!\approx_{\mathrm{Ct}} Q'$, thereby concluding the proof. (dotted line in the diagram below)



Step 1 is in fact entirely complementary to step 1 in the proof of "$\supseteq$" and may, therefore, be omitted. As for step 2, this step is significantly simpler than step 2 in the proof of "$\supseteq$" and, therefore, constitutes the only part where there is a real difference. Specifically, it does not require to bring to bear any factorization theorem. To see that, let us note the following prerequisites:

a. By the alternative SOS, 0 occurs in P' exactly once, namely in the replicator sub–term Rep_0. Hence, $\ll(P'[\,Y/\mathrm{Rep}_0])$ is well–defined and, in consequence, P' can be represented as $(\gg(\ll(P'[\,Y/\mathrm{Rep}_0])))[\mathrm{Rep}_0/\,Y]$.
b. By $\ll(P'[\,Y/\mathrm{Rep}_0]) \approx^\circ_{\mathrm{Ct}} Q^\sharp$ and the fact that $\approx^\circ_{\mathrm{Ct}}$ is preserved by subject replacement, we have $(\gg(\ll(P'[\,Y/\mathrm{Rep}_0])))[\mathrm{Rep}_0/\,Y] \approx_{\mathrm{Ct}} (\gg(Q^\sharp))[\mathrm{Rep}_0/\,Y]$.

Then the final reasoning is simply as follows:

$$\begin{aligned} P' &= (\gg(\ll(P'[Y/\mathrm{Rep}_0])))[\mathrm{Rep}_0/Y] && ; \text{(a)} \\ &\approx_{\mathrm{Ct}} (\gg(Q^\sharp))[\mathrm{Rep}_0/Y] && ; \text{(b)} \\ &\approx_{\mathrm{At}} Q' && ; (1) \end{aligned}$$

This concludes the proof of Theorem 4.3. □

5 Conclusion and Outlook

The preceding sections have been concerned with characterizing context bisimulation over higher–order processes with static scoping. The idea of using replication and triggering to reduce higher–order communication in that setting to first–order communication has been the basis of all of that. This idea has been implemented in a new way, namely as part of a new operational semantics for higher–order processes, which has been called the alternative operational semantics. It has then been possible to provide the characterization in terms of delay bisimulation strictly in the sense of [9].

In the introduction it was claimed that this result should allow one to transfer many results and techniques from the world of CCS–like processes more or less directly to higher–order processes. The remainder of this section briefly mentions work that has followed through on that claim. A detailed account can be found in [4].

The topic is the question of how to model higher–order processes with static scoping. This problem can be regarded as interesting since the only published model for higher–order processes deals with the dynamic setting [10].

The model rests in part on the observation that the alternative operational semantics gives rise to a labelled transition system. It also rests on the fact that labelled transition systems are in one–to–one correspondence with specific instantiations of the categorical notion of *F–coalgebra* (see, for example, [14]), where F is a functor on a category. What is suitable for the purpose of modeling higher–order processes is a category of classes and maps together with a specific functor F. This functor has a unique largest fixed point, which can serve as semantic domain. What is more, that fixed point is a final F–coalgebra. The ensuing (final) *coalgebra homomorphism* from the coalgebraic representation of the alternative transition system into the fixed point can naturally serve as semantic mapping. It is not difficult to prove that this semantics is fully abstract with respect to alternative bisimulation; thus, by Theorem 4.3, the model is fully abstract with respect to context bisimulation. In this sense it is the first model of its kind. What might be regarded as most remarkable is the method of building it and of proving full abstraction: That method is almost entirely borrowed from the final coalgebra semantics

of CCS–like processes (cf. [14] as well). Only Theorem 4.3 needs to be added to complete the full abstraction proof.

One might object that the model is really a model of first–order processes, as process interpretations are essentially obtained via unfoldings under the (first–order) alternative transition system. In a technical sense, this observation is certainly true; from a conceptual viewpoint, however, it must be pointed out that the whole point of the model consists of reflecting semantically what can be seen as the hidden first–order nature of higher–order processes with static scoping. Moreover, the model does not only contain semantic interpretations of higher–order process terms but also semantic interpretations of all term constructors contained in the language, namely as operations on the semantic domain. These interpretations are compositional in the sense that the interpretation of a process term is the same as the interpretation of its outermost constructor applied to the interpretations of its immediate sub–terms. The crucial point, then, consists of the fact that some of those constructor interpretations, by virtue of their compositionality within a higher–order framework, differ significantly from standard coalgebraic interpretations of the corresponding constructors of first–order process algebras. In sum, the model seen in isolation and as a whole, that is, the model including all constructor interpretations is not like a model for first–order processes, even if one disregards the conceptual intent behind it.

References

[1] R. Amadio. On the Reduction of CHOCS Bisimulation to π–Calculus Bisimulation. In *Concurrency Theory*, LNCS 715, pages 112–126. Springer–Verlag, 1993. Proceedings CONCUR conference.

[2] R. Amadio and M. Dam. Reasoning about Higher–order Processes. In *Theory and Practice of Software Development*, LNCS 915, pages 202–216. Springer–Verlag, 1995. Proceedings TAPSOFT '95 conference.

[3] E. Astesiano, A. Giovini, and G. Reggio. Generalized Bisimulation on Relational Specifications. In *Theoretical Aspects of Computer Science*, LNCS 295, pages 207–226. Springer–Verlag, 1988. Proceedings STACS symposium.

[4] M. Baldamus. *Semantics and Logic of Higher–Order Procsesses: Characterizing Late Context Bisimulation*. PhD thesis, computer science department, Berlin University of Technology, 1998.

[5] M. Baldamus and J. Dingel. Modal Characterization of Weak Bisimulation for Higher–order Processes (Extended Abstract). In *Theory and Practice of Software Development*, LNCS 1214, pages 285–296. Springer–Verlag, 1997. Proceedings TAPSOFT '97 conference.

[6] L. Cardelli and A. Gordon. Mobile Ambients. In *Foundations of Soft-*

ware Science and Computational Structures, LNCS 1378, pages 140–155. Springer–Verlag, 1998. Proceedings FoSSaCS '98.

[7] R. De Nicola and M. Hennessy. Testing Equivalences for Processes. *Theoretical Computer Science*, 34:83–133, 1983.

[8] M. Fiore, E. Moggi, and D. Sangiorgi. A Fully Abstract Model for the π–Calculus. In *Logic in Computer Science*, pages 43–54. IEEE Computer Society Press, 1996. Proceedings LICS symposium.

[9] R. van Glabbeek. The Linear Time – Branching Time Spectrum II. In *Concurrency Theory*, LNCS 715, pages 66–81. Springer–Verlag, 1993. Proceedings CONCUR conference.

[10] M. Hennessy. A Fully Abstract Denotational Model for Higher–Order Processes. *Information and Computation*, 112(1):55–95, 1994.

[11] D. Howe. Equality in Lazy Computation Systems. In *Logic in Computer Science*, pages 198–203. IEEE Computer Society Press, 1989. Proceedings LICS '89 symposium.

[12] R. Milner. *Communication and Concurency*. Prentice Hall International, 1989.

[13] R. Milner, J. Parrow, and D. Walker. A Calculus of Mobile Processes, Parts I/II. *Information and Computation*, 100:1–77, 1992. Journal version of two technical reports from 1989.

[14] J. Rutten and D. Turi. Initial Algebra and Final Coalgebra Semantics for Concurrency. In *A Decade of Concurrency — Reflections and Perspectives*, LNCS 803, pages 530–583. Springer–Verlag, 1994. Proceedings REX school/symposium.

[15] D. Sangiorgi. *Expressing Mobility in Process Algebras: First–Order and Higher–Order Paradigms*. PhD Thesis CST–99–93, Department of Computer Science, The University of Edinburgh, 1993.

[16] D. Sangiorgi. From π–Calculus to Higher–order π–Calculus — and Back. In *Theory and Practice of Software Development*, LNCS 668, pages 151–161. Springer–Verlag, 1993. Proceedings TAPSOFT conference.

[17] D. Sangiorgi. Bisimulation in Higher–order Calculi. *Information and Computation*, 131:141–178, 1996.

[18] D. Sangiorgi. π–Calculus, Internal Mobility, and Agent–Passing Calculi. *Theoretical Computer Science*, 167:235–274, 1996.

[19] B. Thomsen. A Calculus of Higher–Order Communicating Systems. In *Principles of Programming Languages*, pages 143–154. ACM, 1989. Proceedings POPL conference.