Modal Characterization of Weak Bisimulation for Higher–order Processes

Extended Abstract

Michael Baldamus
Berlin University of Technology
michael@cs.tu-berlin.de

Jürgen Dingel
Carnegie Mellon University, Pittsburgh
jurgend@cs.cmu.edu

Abstract

Context bisimulation [12, 1] has become an important notion of behavioural equivalence for higher–order processes. Weak forms of context bisimulation are particularly interesting, because of their high level of abstraction. We present a modal logic for this setting and provide a characterization of a variant of weak context bisimulation on second–order processes. We show how the logic permits compositional reasoning. In comparison to previous work by Amadio and Dam [2] on the strong case, our modal logic supports derived operators through a complete duality and thus constitutes an appealing extension of Hennessy–Milner logic.

1 Introduction

First–order process calculi like CCS have long been known as a tractable tool for the description of concurrent processes. Modal (temporal) logic on the other hand has proved itself to be a powerful specification and verification device for such systems. Hennessy–Milner logic [7], for instance, provides an adequate logical match to CCS, and thus complements its algebraic nature very nicely. However, first–order process calculi are limited in the sense that they assume a *fixed* interconnection structure between the processes involved. Recently, name–passing and higher–order calculi have been proposed to remedy this obvious deficiency [10, 15]. They allow the communication of processes and functions, and thus support a powerful abstraction technique which is similar to the one found in higher–order programming languages and caters for systems with *changing* interconnection structure. Not surprisingly, this additional expressive power complicates the theory significantly.

Certain higher–order calculi have received continued attention. One of them is Thomsen's Plain CHOCS [15], which features a static treatment of the restriction operator and a bisimulation–based semantics. In [2], Amadio and Dam address the lack of specification formalisms for Plain CHOCS and propose a modal logic which extends Hennessy–Milner logic and characterizes *strong context bisimulation* (called *CHOCS bisimulation* in [2]). Moreover, they present a sound and complete infinitary proof system for the subcalculus without restriction.

Strong forms of bisimulation are often too fine–grained and, as well–known from the first–order setting, weak forms are more useful because they better capture the observable behaviour of processes. However, this higher level of abstraction in general also makes weak notions of bisimulation less tractable. The presence of higher–order processes aggravates this problem.

This paper picks up the thread initiated by [2], considers a variant of *weak context* *bisimulation* (*bisimulation* for short), and presents what constitutes, to our knowledge, the first logical characterization of weak context bisimulation. We propose a modal logic that, in contrast to the one used in [2], supports full negation. Another significant difference is of proof–technical nature. We introduce a new notion of context bisimulation, which we call *existential bisimulation*. The proof of the characterization result rests on the equivalence of bisimulation and existential bisimulation. Amadio and Dam on the other hand approximate strong context bisimulation and heavily rely on the congruence of these approximations. It is an open question how congruence of analogous approximations in the weak case could be established.

Acknowledgment. We would like to thank Rainer Glas for discussions which helped us to find the right formulation of Definition 3.1(1).

2 Preliminaries

We adopt the process calculus used in [2] and briefly review its syntax and operational semantics before we define weak transitions and our weak variant of bisimulation.

Syntax The following list introduces the syntactic sets and typical variables to be used in the sequel.

- $(c, \cdots \in)\mathbb{C}$: channel names
- $(u, \cdots \in)\mathbb{V}$: variables, where the set of finite subsets of $\mathbb{V}$ is ranged over by $U, \ldots$; $(x, \cdots \in)\mathbb{V}_{\mathbb{P}}$: process variables with $\mathbb{V}_{\mathbb{P}} \subseteq \mathbb{V}$; $(f, \cdots \in)\mathbb{V}_{\mathbb{F}}$: function variables with $\mathbb{V}_{\mathbb{F}} \subseteq \mathbb{V}$
- $(P, \cdots \in)\mathbb{P}^{\circ}$: process expressions, possibly containing free variables or channel names; $\mathbb{P}^{U}$: process expressions whose free variables are from U; $\mathbb{P}^{u}$: process expressions whose free variables are from $\{u\}$; $\mathbb{P}$: process expressions without free variables; elements in $\mathbb{P}$ will be referred to as *processes*.

We require $\mathbb{C}$, $\mathbb{V}_{\mathbb{P}}$, and $\mathbb{V}_{\mathbb{F}}$ to be countably infinite. Note that processes may contain free channel names, but no free variables. Process expressions are generated by the following grammar:

$$P ::= x \mid (fP) \mid (\Sigma_{i\in I} pre_i.P_i) \mid (P \mid P) \mid (\nu c.P) \qquad pre ::= c?x \mid c!P$$

where I ranges over finite index sets. The empty sum represents the process that cannot do anything and is denoted by 0. As in [15, 2], our calculus does not contain any data. Note, however, that they could easily be added. We use *λ–abstraction*, which we denote by $P[u]$ rather than $\lambda u.P$. Input prefixing, λ–abstraction and restriction bind the respective variable or channel name as follows:

operator	$c?x$	$[u]$	νc
bound name	x	u	c

The corresponding definition of α–conversion of bound variables and channel names is standard and omitted. Throughout this paper, we will not distinguish between α–convertible expressions. Then, given that $P[u]$ does not contain any free variable, $P[u]$ is a *function* if $u = x$ and a *functional* if $u = f$. Whenever we want to refer to processes, functions and functionals at the same time, we write $P(u)$. In other words, $P(u)$ has to be thought of as either P or as $P[u]$, depending on whether the statement is to be instantiated to processes, functions, or functionals.

Second–order Substitution Communication will be modeled by the substitution of expressions for free variables. More precisely, the communication rule (?!) below employs *second–order substitution*, where first the free function variable f in P_2' is replaced by a function $P_1'[x]$ and then the argument is replaced for x. The underlying definition of first–order substitution for free variables, denoted by $P[Q/x]$, is standard and omitted. Note that in each of these substitutions α–conversion of bound variables and channel names will avoid unwanted capture of free variables or names.

Operational Semantics The operational semantics is given by three (families of) *labeled transition relations*: For all c, x, f, we have $\xrightarrow{c?x} \subseteq \mathbb{P}\times\mathbb{P}^x$ (input), $\xrightarrow{c!f} \subseteq \mathbb{P}\times\mathbb{P}^f$ (output) and $\xrightarrow{\tau} \subseteq \mathbb{P}\times\mathbb{P}$ (silent move). Note that only processes, that is, closed processes expressions, can perform transitions. Each of the transition relations is defined as the smallest relation satisfying the following axioms and rules where μ ranges over labels.

(?) $pre.P \xrightarrow{pre} P$, for $pre = c?x, \tau$ (!) $c!P_1.P_2 \xrightarrow{c!f} (fP_1 \mid P_2)$

(Σ) $P_k \xrightarrow{\mu} P_k'$ for some $k \in I$ implies $\Sigma_{i\in I}P_i \xrightarrow{\mu} P_k'$

($|$) $P_1 \xrightarrow{\mu} P_1'$ implies $P_1 \mid P_2 \xrightarrow{\mu} P_1' \mid P_2$; and symmetrically

(?!) $P_1 \xrightarrow{c?x} P_1'$ and $P_2 \xrightarrow{c!f} P_2'$ implies $P_1 \mid P_2 \xrightarrow{\tau} P_2'[P_1'[x]/f]$; and symmetrically

(ν) $P \xrightarrow{\mu} P'$ implies $\nu c.P \xrightarrow{\mu} \nu c.P'$ provided that $\mu \neq c?u, c!u$

The communication rule (?!) deserves some explanation. Consider the following typical scenario. Process P_1 wants to transmit P_1' along channel c and then behave like P_1''. Process P_2 on the other hand is waiting for input along c and then turns into P_2'. More precisely,

$$P_1 \equiv c!.P_1'.P_1'' \xrightarrow{c!f} (fP_1' \mid P_1'') \text{ and } P_2 \equiv c?x.P_2' \xrightarrow{c?x} P_2'.$$

Informally, rule (?!) models communication by placing the receiving process inside the sending process. Second–order substitution then yields the expected result.

$$P_1 \mid P_2 \xrightarrow{\tau} (fP_1' \mid P_1'')[P_2[x]/f] = (P_2'[P_1'/x] \mid P_1'')$$

Note how this rule elegantly deals with two difficulties in the definition of higher–order operational semantics: The communication neither causes free channel names to become bound nor bound channel names to become unbound. In particular, the scope of a channel name c that is privately shared between a transmitted process and the continuation of the sender before the communication, is unaffected by the communication. That is, c will still be privately shared between the same processes. We have, therefore, what is called *static scoping* of channel names.

Weak Transitions The above transition relation is very low–level because it renders *all* transitions, including silent τ–transitions, observable. The following definition of weak transitions $\xRightarrow{\widehat{\mu}}$ remedies this. For all c, x, f, we have $\xRightarrow{c?x} \subseteq \mathbb{P} \times \mathbb{P}^x$ (input), $\xRightarrow{c!f} \subseteq \mathbb{P} \times \mathbb{P}^f$ (output) and $\xRightarrow{\epsilon} \subseteq \mathbb{P} \times \mathbb{P}$ (silent move). Each of these relations is defined as the smallest relation satisfying $P \xRightarrow{\epsilon} P'$ if $P \xrightarrow{\tau}^* P'$ and $P \xRightarrow{\mu} P'$ if $P \xRightarrow{\epsilon}\xrightarrow{\mu} P'$. Furthermore, let $\widehat{\mu} \equiv \mu$ if $\mu = c?x, c!f$, and $\widehat{\mu} \equiv \epsilon$ if $\mu = \tau$.

Open Extension Let $\mathcal{R}$ be a binary relation on $\mathbb{P}$. $\mathcal{R}$ can be extended to a binary relation on $\mathbb{P}^u$: For all x, f, binary relations $\mathcal{R}^x$, $\mathcal{R}^f$ on $\mathbb{P}^x$, $\mathbb{P}^f$ are given by $P \; \mathcal{R}^x \; Q$ if $P[R/x] \; \mathcal{R} \; Q[R/x]$ for all $R \in \mathbb{P}$ and $P \; \mathcal{R}^f \; Q$ if $P[R[y]/f] \; \mathcal{R} \; Q[R[y]/f]$ for all y, $R \in \mathbb{P}^y$. The *open extension* $\mathcal{R}^\circ$ of $\mathcal{R}$ then is defined by $\mathcal{R}^\circ \equiv \bigcup_u \mathcal{R}^u$. Note that $\mathcal{R}^\circ$ does not consider any expression with more than one free variable. Our definition of open extension is, therefore, not the usual one. It is, however, sufficient for our purposes.

Context Bisimulation We now define the notion of bisimulation we will strive to characterize by logical means in Section 4.

Definition 2.1. A binary relation $\mathcal{R}$ on $\mathbb{P}$ is a *(weak late context) bisimulation* if $P \; \mathcal{R} \; Q$ implies: Whenever $P \xrightarrow{\mu} P'$ then, for some Q', $Q \xRightarrow{\widehat{\mu}} Q'$ and $P' \; \mathcal{R}^\circ \; Q'$, and symmetrically. We denote by $\approx$ the union of all bisimulations.

This notion differs from the one used in [2] only in that the matching transition by Q may be weak. Moreover, this transition must not contain any trailing τ–move, so $\approx$ is what is called a *delay bisimulation* [6]. Also, considering the derivation of a communication transition, note that the receiver does not know the identity of the process transmitted until the (?!)–rule is applied. Such a communication scheme is called late which makes $\approx$ is a *late bisimulation*. For a conceptual discussion and practical application of this combination, see [13, 5]. We do not know whether the results of this paper could be obtained for classically weak and/or non–late, that is, early forms of bisimulation.

3 Existential Bisimulation

We now introduce a new notion of bisimulation on higher–order processes called existential bisimulation. It bridges the gap between bisimulation and logical equivalence, that is, our characterization proof falls into two parts: 1) Two processes are bisimilar if and only if they are existentially bisimilar. 2) Two processes are existentially bisimilar if and only if they satisfy the same modal formulas.

The intuition behind existential bisimulation is given by the following simple idea: An operational semantics is assigned to function(al)s by regarding function(al)–argument–result triples of the form $(P[u], Q(x), P[Q(x)/u])$ as labeled transitions. For example, the first three steps of the overall operational semantics of the process $c?x.d!x.0$ are

$$\begin{tikzcd}
 & & d!R_1.0 \arrow[r, "d!f"] & fR_1 \mid 0[f] \\
c?x.d!x.0 \arrow[r, "c?x"] & d!x.0[x] \arrow[ur, "R_1"] \arrow[r, "R_2"'] & d!R_2.0 \arrow[r, "d!f"] & fR_2 \mid 0[f] \\
 & & \vdots &
\end{tikzcd}$$

Each process has a transition for each action it can perform. Each function(al) has a transition for each of its potential arguments. (In the diagram this aspect is indicated by dot notation.) To obtain a notion of observational equivalence we now need to define how a transition of a process can be matched by another transition of another process. For process transitions we adopt the standard definition: A transition of the form $P \xrightarrow{\mu} P'$ has to be matched by a transition of the form $Q \overset{\hat{\mu}}{\Longrightarrow} Q'$. For function(al) transitions, however, we now have a choice: A transition of the form $P[u] \xrightarrow{R(x)} P[R(x)/u]$ could be matched by either

- a transition $Q[u] \xrightarrow{R(x)} Q[R(x)/u]$, where the argument is the same, or
- a transition $Q[u] \xrightarrow{S(x)} Q[S(x)/u]$, where $R(x)$ must be bisimilar to $S(x)$ and $P[R(x)/u]$ must be bisimilar to $Q[S(x)/u]$. More precisely, $P[u]$ and $Q[u]$ are bisimilar if, for every $R(x)$, there **exists** a $S(x)$ so that $R(x)$ is bisimilar to $S(x)$ and $P[R(x)/u]$ is bisimilar to $Q[S(x)/u]$, and conversely.

Adopting the first option leads us to bisimulation, adopting the second to existential bisimulation. The latter notion forms the basis of our modal characterization of $\approx$.

In the study of higher–order functional languages such as the typed λ–calculus it proved extremely useful to compare functions by means of *logical relations* [11]. In this setting, two functions are related if whenever they are applied to related arguments they yield related results. From this perspective, existential bisimulation appears to be a hybrid between the standard notion of bisimulation on the one hand and the concept of logical relations on the other. Given the apparent mix of functional and concurrent concepts in our process calculus this is not surprising.

Before we introduce existential bisimulation formally, we would like to point out that an entirely different route to modal logic for higher–order process may be possible by using Sangiorgi's results about the fully abstract translation of higher–order into π–calculus processes [12]. Satisfaction of modal formulas in such a framework would be defined wrt. π–calculus processes and their transitions. Note, however, that any characterization result obtained using this approach would, therefore, be significantly less direct than ours.

Existential Extension The formal definition of existential bisimulation is based on the notion of existential extension. This construction is similar to the open extension of a binary relation on $\mathbb{P}$. Lemma 3.2 states the central property of existential extension, which is appropriately called existential property.

Definition 3.1. Let $\mathcal{R}$ be a binary relation on $\mathbb{P}$.

1. For all U, V, a relation $\mathcal{R}^{\exists}_{U,V} \subseteq \mathbb{P}^U \times \mathbb{P}^V$ is given by $\mathcal{R}^{\exists}_{U,V} \equiv \mathcal{R}$ if $U = V = \emptyset$ and, otherwise, $P\ \mathcal{R}^{\exists}_{U,V}\ Q$ if, for $W \equiv U \cup V$:

I.a. $\forall x \in W, \quad R \in \mathbb{P} \,.\, \exists S_1 \in \mathbb{P} \,.\, R \quad \mathcal{R}^{\exists}_{\emptyset,\emptyset} \quad S_1 \wedge \quad P[R/x] \quad \mathcal{R}^{\exists}_{U-x,V-x} \quad Q[S_1/x]$
I.b. $\forall f \in W, x, R \in \mathbb{P}^x.\, \exists S_1 \in \mathbb{P}^x.\ R\ \mathcal{R}^{\exists}_{\{x\},\{x\}}\ S_1 \wedge P[R[x]/f]\ \mathcal{R}^{\exists}_{U-f,V-f}\ Q[S_1[x]/f]$
II.a. $\forall x \in W, \quad R \in \mathbb{P} \,.\, \exists S_2 \in \mathbb{P} \,.\, S_2 \quad \mathcal{R}^{\exists}_{\emptyset,\emptyset} \quad R \wedge \quad P[S_2/x] \quad \mathcal{R}^{\exists}_{U-x,V-x} \quad Q[R/x]$
II.b. $\forall f \in W, x, R \in \mathbb{P}^x.\, \exists S_2 \in \mathbb{P}^x.\ S_2\ \mathcal{R}^{\exists}_{\{x\},\{x\}}\ R \wedge P[S_2[x]/f]\ \mathcal{R}^{\exists}_{U-f,V-f}\ Q[R[x]/f]$

2. A binary relation $\mathcal{R}^{\exists}$ on $\mathbb{P}^{\circ}$ is given by $\mathcal{R}^{\exists} \equiv \bigcup_{U,V} \mathcal{R}^{\exists}_{U,V}$. We call $\mathcal{R}^{\exists}$ the *existential extension* of $\mathcal{R}$.

Lemma 3.2. *$P\ \mathcal{R}^{\exists}\ Q$ implies:*

i.a. $\forall x, \quad R \in \mathbb{P} \,.\, \exists S_1 \in \mathbb{P} \,.\, R\ \mathcal{R}^{\exists}\ S_1 \wedge \quad P[R/x] \quad \mathcal{R}^{\exists} \quad Q[S_1/x]$
i.b. $\forall f, x, R \in \mathbb{P}^x.\, \exists S_1 \in \mathbb{P}^x.\ R\ \mathcal{R}^{\exists}\ S_1 \wedge P[R[x]/f]\ \mathcal{R}^{\exists}\ Q[S_1[x]/f]$
ii.a. $\forall x, \quad R \in \mathbb{P} \,.\, \exists S_2 \in \mathbb{P} \,.\, S_2\ \mathcal{R}^{\exists}\ R \wedge \quad P[S_2/x] \quad \mathcal{R}^{\exists} \quad Q[R/x]$
ii.b. $\forall f, x, R \in \mathbb{P}^x.\, \exists S_2 \in \mathbb{P}^x.\ S_2\ \mathcal{R}^{\exists}\ R \wedge P[S_2[x]/f]\ \mathcal{R}^{\exists}\ Q[R[x]/f]$

Proof. Straightforward by induction on the generation of $\mathcal{R}^{\exists}$. □

Existential Bisimulation

Definition 3.3. A binary relation $\mathcal{R}$ on $\mathbb{P}$ is an *existential (weak late context) bisimulation* if $P\ \mathcal{R}\ Q$ implies: Whenever $P \overset{\widehat{\mu}}{\Longrightarrow} P'$ then, for some Q', $Q \overset{\widehat{\mu}}{\Longrightarrow} Q'$ and $P'\ \mathcal{R}^{\exists}\ Q'$, and symmetrically. We denote by $\approx_{\exists}$ the union of all existential bisimulations.

Lemma 3.4. *Existential bisimilarity is an equivalence and, at the same time, $P \approx_{\exists} Q$ if and only if: Whenever $P \overset{\widehat{\mu}}{\Longrightarrow} P'$ then, for some Q', $Q \overset{\widehat{\mu}}{\Longrightarrow} Q'$ and $P' \approx^{\exists}_{\exists} Q'$, and symmetrically. As a consequence, $\approx_{\exists}$ is the largest existential bisimulation.*

Theorem 3.5. $\approx_{\exists} \; = \; \approx$.

Proof. "$\supseteq$": By the fact that $\approx$ itself is a bisimulation together with the easily proved property $\mathcal{R}^{\circ} \subseteq \mathcal{R}^{\exists}$.
"$\subseteq$": (Outline) This direction requires far more extensive reasoning. The idea behind the proof is as follows: Consider Lemma 3.4: $P \approx_{\exists} Q$ iff whenever $P \overset{\widehat{\mu}}{\Longrightarrow} P'$ then, for some Q', $Q \overset{\widehat{\mu}}{\Longrightarrow} Q'$ and $P' \approx^{\exists}_{\exists} Q'$, and symmetrically. Assume, also, congruence of

$\approx_\exists$ and, for simplicity's sake, $P', Q' \in \mathbb{P}^x$ for some x. Then: $P' \approx_\exists^{\boxed{\circ}} Q'$ implies $\forall R \in \mathbb{P}.\ \exists S \in \mathbb{P}.\ R \approx_\exists S \wedge P'[R/x] \approx_\exists Q'[S/x]$. Further, $R \approx_\exists S$ plus congruence of $\approx_\exists$ implies $Q'[R/x] \approx_\exists Q'[S/x]$, so symmetry and transitivity of $\approx_\exists$ imply $\forall R \in \mathbb{P}.\ P'[R/x] \approx_\exists Q'[R/x]$, so $P' \approx_\exists^{\circ} Q'$, so $\approx_\exists$ is a bisimulation, so $\approx_\exists\ \subseteq\ \approx$. The problem is that proving congruence of weak bisimulation on higher–order processes is inherently difficult [12, 5, 4], and even more so in the case of existential bisimulation.

To solve this problem we use a variation of a method that was originally developed by Howe to prove congruence of applicative bisimulation in functional computational frameworks [8]. Howe's method has already been adapted to prove congruence of standard forms of weak bisimulation on ω–order processes [5, 4]. We need to give the following definition: A *constructor co* has the form f, $c?x$, $c!$, Σ, $|$ or νc. Constructors may be applied to families of expressions, $co(\widetilde{P})$, where the result is defined according to the grammar. The size of $\widetilde{P}$ must of course coincide with the arity of co and, in the case of Σ, every element of $\widetilde{P}$ must be of the form $pre.P$.

$$
\begin{array}{llllll}
f(P) & \equiv fP & c!(P_1, P_2) & \equiv c!P_1.P_2 & |(P_1, P_2) & \equiv P_1 \mid P_2 \\
c?x(P) & \equiv c?x.P & \Sigma(pre_i.P_i)_{i\in I} & \equiv \Sigma_{i\in I} pre_i.P_i & \nu c(P) & \equiv \nu c.P
\end{array}
$$

The centerpiece of the whole proof then consists of the *existential Howe closure* $\approx_\exists^{\boxed{\bullet}}$ of $\approx_\exists$. This relation is defined to be the smallest binary relation on $\mathbb{P}^\circ$ for which

$$
\frac{x \approx_\exists^{\boxed{\circ}} Q}{x \approx_\exists^{\boxed{\bullet}} Q} \quad \text{and} \quad \frac{\widetilde{P} \approx_\exists^{\boxed{\bullet}} \widetilde{Q} \quad co(\widetilde{Q}) \approx_\exists^{\boxed{\circ}} Q}{co(\widetilde{P}) \approx_\exists^{\boxed{\bullet}} Q},
$$

where $\widetilde{P}$ and $\widetilde{Q}$ must be of the same size, $\widetilde{P} \approx_\exists^{\boxed{\bullet}} \widetilde{Q}$ is understood component–wise and $co(\widetilde{P})$ and $co(\widetilde{Q})$ must be well–formed. For the *Howe closure* $\approx_\exists^{\bullet}$ of $\approx_\exists$, one would use the same definition with $\approx_\exists^{\circ}$ instead of $\approx_\exists^{\boxed{\circ}}$. A number of standard general properties can be shown for Howe closures provided that the underlying relation is an equivalence and preserved by the renaming of unbound variables, names and the like. Analogous properties can be shown in mostly identical or similar ways for $\approx_\exists^{\boxed{\bullet}}$: 1) $\approx_\exists^{\boxed{\bullet}}$ is reflexive, 2) $\approx_\exists^{\boxed{\circ}}\ \subseteq\ \approx_\exists^{\boxed{\bullet}}$, 3) $\approx_\exists^{\boxed{\bullet}}\approx_\exists^{\boxed{\circ}}\ \subseteq\ \approx_\exists^{\boxed{\bullet}}$, where $\approx_\exists^{\boxed{\bullet}}\approx_\exists^{\boxed{\circ}}$ is the relational composition of $\approx_\exists^{\boxed{\bullet}}$ and $\approx_\exists^{\boxed{\circ}}$, 4) $\approx_\exists^{\boxed{\bullet}}$ is a congruence, 5) $\approx_\exists^{\boxed{\bullet}\,*}$ is symmetric.

Next, we show that, for all $P, Q \in \mathbb{P}$, $P \approx_\exists^{\boxed{\bullet}} Q$ implies: Whenever $P \xrightarrow{\mu} P'$ then, for some Q', $Q \overset{\widehat{\mu}}{\Longrightarrow} Q'$ and $P' \approx_\exists^{\boxed{\bullet}} Q'$. This proof can be done by transition induction on $P \xrightarrow{\mu} P'$, using (1)–(4). Further, let $\approx_{\exists\,c}^{\boxed{\bullet}}\ \equiv\ \approx_\exists^{\boxed{\bullet}}\ \cap\ \mathbb{P} \times \mathbb{P}$. We can prove $(\approx_{\exists\,c}^{\boxed{\bullet}})^*$ to be a simulation, using the preceding property in combination with induction on the length of sequences of the form $P_0 \approx_\exists^{\boxed{\bullet}} \ldots \approx_\exists^{\boxed{\bullet}} P_k$. Then, by (5), $(\approx_{\exists\,c}^{\boxed{\bullet}})^*$ is a bisimulation. At the same time, an easily proved lemma is is $\approx_\exists\ \subseteq\ (\approx_{\exists\,c}^{\boxed{\bullet}})^*$, so we have indeed $\approx_\exists\ \subseteq\ \approx$. $\Box$

The full proof of Theorem 3.5 is presented in our technical report [3].

Approximating Existential Bisimulation For the proof of the characterization theorem in [2], Amadio and Dam show that strong bisimulation $\sim$ can be obtained as the limit of a descending chain of equivalence relations $\sim^k$ where k is a natural number.

We will transfer this idea to our setting and also generalize the approximation from natural numbers to ordinals. Let ORD denote the class of ordinals ranged over by κ and λ. A basic result of set theory says that ORD is a proper class, that is, not a set.

Definition 3.6. A ORD-indexed family of binary relations $\approx_{\exists}^{\kappa}$ on $\mathbb{P}$ is given as follows:

i. $P \approx_{\exists}^{0} Q$ always.

ii. $P \approx_{\exists}^{\kappa+1} Q$ if: Whenever $P \stackrel{\widehat{\mu}}{\Longrightarrow} P'$ then, for some Q', $Q \stackrel{\widehat{\mu}}{\Longrightarrow} Q'$ and $P' \approx_{\exists}^{\kappa} Q'$, and symmetrically.

iii. For every limit ordinal λ, $P \approx_{\exists}^{\lambda} Q$ if $P \approx_{\exists}^{\kappa} Q$ for every $\kappa < \lambda$.

Finally, $P \approx_{\exists}^{ORD} Q$ if $P \approx_{\exists}^{\kappa} Q$ for every $\kappa \in ORD$.

Proposition 3.7. $\approx_{\exists}^{ORD} \; = \; \approx_{\exists}$.

Proof. Similar to the proof of the corresponding approximation result for weak bisimulation on CCS in [9]. For details see [3]. □

Note that, by the proof of Theorem 3.5, $\approx_{\exists}$ is a congruence because $\approx$ is one [4]. However, it is still an open question whether $\approx_{\exists}^{\kappa}$ is a congruence for every κ.

4 Modal Characterization of Bisimilarity

Modal Formulas Properties of process expressions are expressed using modal formulae generated by the grammar below. An important syntactic feature of modal formulae is that, just like process expressions, they have one of three orders. Formulae of order 0, 1, or 2 describe processes, functions or functionals respectively. Let ϕ^i, ψ^i, χ^i range over modal formulas of order i, where $i \in \{0, 1, 2\}$. We omit these superscripts when the ranges of possible orders are determined by the context.

$$\begin{aligned}
\phi^i &::= \textstyle\bigwedge_{j \in I} \phi^i_j \mid \neg\phi^i \quad \text{for } i \in \{0,1,2\} \\
\phi^0 &::= \langle c?x\rangle\phi^1 \mid \langle c!f\rangle\phi^2 \mid \langle\epsilon\rangle\phi^0 \\
\phi^i &::= \langle\phi^{i-1}\rangle\psi^0 \quad \text{for } i \in \{1,2\}
\end{aligned}$$

where I is a *countable* index set, that is, conjunctions maybe infinite but countable. We use the standard abbreviations: $\top \equiv \bigwedge\emptyset$ and $\bot \equiv \neg\top$ and $\phi_1 \vee \phi_2 \equiv \neg(\neg\phi_1 \wedge \neg\phi_2)$.

In contrast to [2], the logic features the dualities familiar from Hennessy–Milner logic. We define $[\widehat{\mu}]\phi \;\equiv\; \neg\langle\widehat{\mu}\rangle\neg\phi$ and $[\phi]\psi \;\equiv\; \neg\langle\phi\rangle\neg\psi$.

Realization The logic allows two kinds of modal judgments. The meaning of $P \models \langle\widehat{\mu}\rangle\phi$ is familiar from Hennessy–Milner logic: P can make a weak transition labeled with $\widehat{\mu}$ and then behave as specified by ϕ. The intuition behind $P[u] \models \langle\phi\rangle\psi$ is that there is an argument satisfying ϕ so that the application of the function(al) $P[u]$ to it results in a process satisfying ψ. More precisely, $\langle\phi^1\rangle\psi^0$ describes a functional $P[f]$ for which there is a function $Q[x]$ satisfying ϕ^1 so that $P[Q[x]/f]$ satisfies ψ^0. $\langle\phi^0\rangle\psi^0$, on the other hand, specifies a function $P[x]$ for which there is a process Q satisfying ϕ^0 so that $P[Q/x]$ satisfies ψ^0. Formally,

$$\begin{array}{ll}
P(u) \models \bigwedge_{i\in I}\phi_i & \text{if, for every } i \in I,\ P(u) \models \phi_i \\
P(u) \models \neg\phi & \text{if not } P(u) \models \phi \\
P \models \langle\widehat{\mu}\rangle\phi & \text{if, for some } P',\ P \overset{\widehat{\mu}}{\Longrightarrow} P' \text{ and } P'(u) \models \phi, \\
 & \text{where } u \text{ is the variable that possibly occurs in } \mu \\
P[u] \models \langle\phi\rangle\psi & \text{if, for some } Q(x),\ Q(x) \models \phi \text{ and } P[Q(x)/u] \models \psi.
\end{array}$$

Note that $[\phi]\psi$ is equivalent to Amadio and Dam's implication operator $\phi \Rightarrow \psi$ in [2].

Example 4.1. Consider the following process expressions.

$$\begin{array}{lll}
P_1 \equiv (x \mid P_1')[x] & \text{where} & P_1' \equiv c!0.0 \mid d?y.0 \text{ and} \\
P_2 \equiv \nu e.(c?z.e!z.d!z.0 \mid e?w.P_2') & \text{where} & P_2' \in \mathbb{P}
\end{array}$$

P_1 can be thought of as a *client* which when given a *server* P_2 first provides input to the server along c and then expects the result on d. In this particular case, P_1 just outputs the 0 process and P_2 just passes its input along after having copied it to the parallel subprocess P_2'. In the overall system $P_1[P_2/x] = P_2 \mid P_1'$ both processes can communicate as expected and then halt. More precisely, they are allowed to engage in an arbitrary but finite number of internal actions and then reach a state from which no further action is possible: $P_2 \mid P_1' \models \langle\epsilon\rangle\phi_{hlt}$ where $\phi_{hlt} \equiv [\epsilon]\bot \wedge \bigwedge_{c,x,f}[c?x]\bot \wedge [c!f]\bot$.

Modal Depth $|\phi|$ denotes the modal depth of a formula and is defined as follows:

$$|\bigwedge_{i\in I}\phi_i| \equiv sup\,(|\phi_i|)_{i\in I} \quad |\neg\phi| \equiv |\phi| \quad |\langle\widehat{\mu}\rangle\phi| \equiv 1 + |\phi| \quad |\langle\phi\rangle\psi| \equiv max(|\phi|, |\psi|)$$

Definition 4.2. For every κ, we define an equivalence $\approx_L^\kappa$ on processes, functions and functionals by

$P(u) \approx_L^\kappa Q(u)$ if, for all ϕ with $|\phi| \leq \kappa$, $P(u) \models \phi$ if and only if $\models Q(u) \models \phi$.

$P(u) \approx_L Q(u)$ if, for every κ, $P(u) \approx_L^\kappa Q(u)$.

Characteristic Formulas For all processes, functions or functionals $P(u)$, $Q(u)$ and every κ:

- We choose some $\psi^\kappa_{P(u),Q(u)}$ with $P(u) \models \psi^\kappa_{P(u),Q(u)}$ and $Q(u) \not\models \psi^\kappa_{P(u),Q(u)}$, provided that $P(u) \not\approx_L^\kappa Q(u)$.
- $\phi^\kappa_{P(u),Q(u)} \equiv \begin{cases} \psi^\kappa_{P(u),Q(u)} & \text{if } P(u) \not\approx_L^\kappa Q(u) \\ \top & \text{if } P(u) \approx_L^\kappa Q(u) \end{cases}$
- $\phi^\kappa_{P(u)} \equiv \bigwedge_{Q(u)} \phi^\kappa_{P(u),Q(u)}$

Lemma 4.3. *$Q(u) \models \phi^\kappa_{P(u)}$ if and only if $Q(u) \approx_L^\kappa P(u)$.*

Proof. "$\Leftarrow$": Immediate since $|\phi^\kappa_{P(u)}| \leq \kappa$ and $P(u) \models \phi^\kappa_{P(u)}$. "$\Rightarrow$": By contraposition. □

The Characterization Theorem For the proof of the characterization of $\approx$ in terms of $\approx_L$, we need to extend $\approx_{\exists}^{\kappa}$ to function(al)s: $P[u] \approx_{\exists}^{\kappa} Q[u]$ if $P \approx_{\exists}^{\kappa\circ} Q$.

Proposition 4.4. $\approx_L^{\kappa} \;=\; \approx_{\exists}^{\kappa}$.

Proof. By transfinite induction on κ, following the lines of the proof of Proposition 10.6 in [9]. Only the cases where a function(al) meets some formula of the form $\langle\phi\rangle\psi$ are new. We restrict ourselves accordingly.

base case: Immediate since $P(u) \approx_{\exists}^{0\circ} Q(u)$ for all $P(u), Q(u)$.

successor case: "$\supseteq$"; Using structural induction on ϕ, where $|\phi| \leq \kappa + 1$, we show that $P(u) \approx_{\exists}^{\kappa+1} Q(u)$ implies: $P(u) \models \phi$ if and only if $Q(u) \models \phi$.

In the non–standard case we consider the sub–case where $P(u) = P[f]$, $Q(u) = Q[f]$ and $\phi = \langle\psi\rangle\chi$, proving that $P[f] \models \phi$ implies $Q[f] \models \phi$. The desired equivalence follows by symmetry. | By $P[f] \models \langle\psi\rangle\chi$: There exists a $R[x]$ so that $R[x] \models \psi$ and $P[R[x]/f] \models \chi$. | By $P[f] \approx_{\exists}^{\kappa+1} Q[f]$: There exists a $S_1[x]$ so that $R[x] \approx_{\exists}^{\kappa+1} S_1[x]$ and $P[R[x]/f] \approx_{\exists}^{\kappa+1} Q[S_1[x]/f]$. | By structural induction, taking into account that $|\langle\psi\rangle\phi| \leq \kappa + 1$ implies $|\psi|, |\chi| \leq \kappa + 1$: $S_1[x] \models \psi$ and $Q[S_1[x]/f] \models \chi$. | By definition: $Q[f] \models \langle\psi\rangle\chi$.

"$\subseteq$"; By contraposition, assuming $P(u) \not\approx_{\exists}^{\kappa+1} Q(u)$. In the non–standard case we again consider the sub–case where $P(u) = P[f]$ and $Q(u) = Q[f]$. | By assumption: (a) or (b), where:

a. $\exists R[x].\, \forall S_1[x].\, R[x] \not\approx_{\exists}^{\kappa+1} S_1[x] \vee P[R[x]/f] \not\approx_{\exists}^{\kappa+1} Q[S_1[x]/f]$

b. $\exists R[x].\, \forall S_2[x].\, S_2[x] \not\approx_{\exists}^{\kappa+1} R[x] \vee P[S_2[x]/f] \not\approx_{\exists}^{\kappa+1} Q[R[x]/f]$

Because (a) and (b) are practically the same, we consider only (b). | In this case, by Lemma 4.3: $Q[f] \models \langle\phi_{R[x]}^{\kappa+1}\rangle\phi_{Q[R[x]/f]}^{\kappa+1}$ where the argument $R[x]$ is as in (b). | Suppose now $P[f] \models \langle\phi_{R[x]}^{\kappa+1}\rangle\phi_{Q[R[x]/f]}^{\kappa+1}$. | By definition: There exists a $S_2[x]$ so that $S_2[x] \models \phi_{R[x]}^{\kappa+1}$ and $P[S_2[x]/f] \models \phi_{Q[R[x]/f]}^{\kappa+1}$. | By Lemma 4.3 and induction on the order: $S_2[x] \approx_{\exists}^{\kappa+1} R[x]$ and $P[S_2[x]/f] \approx_{\exists}^{\kappa+1} Q[R[x]/f]$, contradiction. | Thus, $P[f] \not\models \langle\phi_{R[x]}^{\kappa+1}\rangle\phi_{Q[R[x]/f]}^{\kappa+1}$. Note $|\langle\phi_{R[x]}^{\kappa+1}\rangle\phi_{Q[R[x]/f]}^{\kappa+1}| \leq \kappa + 1$.

limit case: By induction on the order. The non–standard cases are once again those situations where a function(al) meets some formula of the form $\langle\phi\rangle\psi$. They can be dealt with in practically the same way as we have done it in the successor case. □

The proof of the corresponding result in [2] for the strong case hinges on $\sim$ and $\sim^k$ being congruences. It is important to note that our proof does not rely on this kind of requirement.

Finally, we can state and prove the actual characterization. To this end, we need to extend $\approx$ to function(al)s, similarly as we did it with $\approx_{\exists}^{\kappa}$: $P[u] \approx Q[u]$ if $P \approx^{\circ} Q$.

Theorem 4.5. $\approx_L \;=\; \approx$.

Proof. We give the proof for the restriction of $\approx_L$ to processes. The proof of the full result would require us to elaborate somewhat on the third and fifth steps. This additional reasoning, however, is straightforward.

$$\begin{aligned}
\approx_L &= \textstyle\bigcap_\kappa \approx_L^\kappa && \text{; def.}\\
&= \textstyle\bigcap_\kappa \approx_\exists^\kappa && \text{; Proposition 4.4}\\
&= \approx_\exists^{ORD} && \text{; def.}\\
&= \approx_\exists && \text{; Proposition 3.7}\\
&= \approx && \text{; Theorem 3.5} \quad \square
\end{aligned}$$

5 Compositional Verification

We will now demonstrate how both the process calculus and the modal logic support compositional reasoning. Compositionality will be achieved by means of a well–known technique called *assumption–commitment reasoning*. In this approach, proofs are split into two parts: First we prove that a component of the overall system satisfies a certain property under the assumption that the environment behaves in a certain way. In a second step, we show that the environment does indeed behave as assumed. We will illustrate the application of this idea by means of the example of Section 4. The client P_1 makes certain assumptions about the server it will work with. If the server does meet those assumptions, then the overall system will behave as desired. More precisely,

$$P_1 \models [\phi_{srv}](\langle\epsilon\rangle\phi_{hlt}) \text{ where } \phi_{srv} \equiv \langle c?z\rangle([\top](\langle d!f\rangle\top)).$$

ϕ_{srv} is the environment assumption the client wants a server to satisfy. It specifies that the server should first be able to receive input along c and then, for all of those inputs, it should be able to offer output along d. If the input to P_1 satisfies ϕ_{srv} then P_1 can engage in an arbitrary but finite number of internal actions and then stop. We see that process P_2 does meet the requirement expected of the server: $P_2 \models \phi_{srv}$. The logic now allows us to conclude that the overall system $P_1 P_2 = P_2 \mid P_1'$ will work correctly: $P_2 \mid P_1' \models \langle\epsilon\rangle\phi_{hlt}$. In sum, instead of reasoning about the entire system as in Section 4, we can reason about each of its constituents separately and thus reap the benefits of compositionality. Note that both $P_1 P_2$ and P_2 exhibit invisible transitions and that this example consequently could not have been expressed in the strong setting used in [2].

6 Conclusion and Further work

We have given what constitutes, to our knowledge, the first logical characterization of a weak variant of context bisimulation on second–order processes. The characterization hinges on a novel notion of observable equivalence on higher–order processes called existential bisimulation. This notion, apart form its proof technical importance, also seems to be of conceptual value as it matches the combination of functional and concurrent features of the process calculus. The modal logic comprises negation and all dualities known from Hennessy–Milner logic. We have demonstrated that the process calculus on the one hand and the modal logic on the other hand mesh very well and open up a way towards modular verification of higher–order processes.

So far, we clearly lack a syntactic framework which permits the formal derivation of statements like $P \models \phi$. Unfortunately, our attempts to equip the presented combination of process calculus and logic with a proof system have failed. The fact that a μ–move by a parallel composition $P_1 \mid P_2$ may hide arbitrarily many communications between the two processes poses a substantial problem. Additionally, the rules for parallel composition seem to require a congruence property our setting does not offer. Alternatively, we tried to find a complete axiomatization along the lines of [14]. However, a straightforward adaption of the results in [14] is encumbered by the more complex modalities.

The results of this paper rest on the notion of existential bisimulation. There is some hope that this new notion may also be fruitfully applied to other higher–order calculi. The most promising candidates seem to be ω–order calculi like Sangiorgi's $HO\pi$ [12]. In this setting, context bisimulation also serves as the notion of observational equivalence.

References

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

[2] R.M. Amadio and M. Dam. Reasoning about Higher–order Processes. In P.D. Mosses, M. Nielsens, and M.I. Schwartzbach, editors, *Theory and Practice of Software Development*, LNCS 915, pages 202–216. Springer, 1995. Proceedings TAPSOFT.

[3] M. Baldamus and J. Dingel. Modal Characterization of Weak Bisimulation for Higher–order Processes. Report 96–27, Berlin University of Technology, Computer Science Department, 1996. Retrievable via the Hypatia electronic library.

[4] M. Baldamus and T. Frauenstein. Congruence Proofs for Weak Bisimulation Equivalences on Higher–order Process Calculi. Report 95–21, Berlin University of Technology, Computer Science Department, 1995.

[5] W. Ferreira, M. Hennessy, and A. Jeffrey. A Theory of Weak Bisimulation for Core CML. In *Functional Programming*, pages 201–212. ACM Press, 1996. Conference proceedings.

[6] R.J. van Glabeek. The Linear Time — Branching Time Spectrum II. In *CONCUR*, LNCS 715, pages 66–81. Springer, 1993. Proceedings.

[7] M. Hennessy and R. Milner. Algebraic laws for nondeterminism and concurrency. *Journal of the ACM*, 32:137–161, 1985.

[8] D. Howe. Equality in Lazy Computation Systems. In *Logic in Computer Science*, pages 198–203, 1989. Proceedings LICS.

[9] R. Milner. *Communication and Concurrency*. Prentice–Hall, 1989.

[10] R. Milner, J. Parrow, and D. Walker. A Calculus of Mobile Processes, (Parts I and II). *Information and Computation*, (100):1–77, 1992.

[11] J.C. Mitchell. Type Systems for Programming Languages. In J. van Leeuwen, editor, *Handbook of Theoretical Computer Science*, pages 365–458. North–Holland, 1990.

[12] D. Sangiorgi. *Expressing Mobility in Process Algebras: First–order and Higher–order Paradigms*. Cst–99–93, Department of Computer Science, University of Edinburgh, 1993.

[13] D. Sangiorgi. Bisimulation in Higher–order Calculi. Report RR–2508, INRIA–Sophia Antipolis, 1995. To appear in Information and Computation.

[14] C. Stirling. Modal Logics for Communicating Systems. *Theoretical Computer Science*, (49):311–347, 1987.

[15] B. Thomsen. Plain CHOCS — A Second Generation Calculus for Higher–order Processes. *Acta Informatica*, (30):1–59, 1993.