First-Order Reasoning for Higher-Order Concurrency[☆]

Vasileios Koutavas[*], Matthew Hennessy[*]

Trinity College Dublin

Abstract

We present a practical first-order theory of a higher-order π-calculus which is both sound and complete with respect to a standard semantic equivalence. The theory is a product of combining and simplifying two of the most prominent theories for HOπ of Sangiorgi et al. and Jeffrey and Rathke [10, 21], and a novel approach to *scope extrusion*. In this way we obtain an elementary labelled transition system where the standard theory of first-order weak bisimulation and its corresponding propositional Hennessy-Milner logic can be applied.

The usefulness of our theory is demonstrated by straightforward proofs of equivalences between compact but intricate higher-order processes using witness first-order bisimulations, and proofs of inequivalence using the propositional Hennessy-Milner logic. Finally we show that contextual equivalence in a higher-order setting is a conservative extension of the first-order π-calculus.

Keywords: higher-order concurrency, bisimulation, Hennessy-Milner logic

[☆]This research was supported by SFI project SFI 06 IN.1 1898.

[*]Corresponding author

Email addresses: Vasileios.Koutavas@scss.tcd.ie (Vasileios Koutavas), Matthew.Hennessy@scss.tcd.ie (Matthew Hennessy)

$$c?(X,Y).\nu t.\,(\; c!\langle(\mathsf{app}\,X \oplus \mathsf{app}\,Y) \mid \mathbf{app}\,\boldsymbol{X}\rangle.\mathtt{0} \oplus c!\langle t!.\mathbf{app}\,\boldsymbol{Y}\rangle.\mathtt{0}\;) \mid *(\; t?.(\mathsf{app}\,X \oplus \mathsf{app}\,Y)\;) \quad (\dagger)$$
$$\approx c?(X,Y).\nu t.\,(\; c!\langle(\mathsf{app}\,X \oplus \mathsf{app}\,Y) \mid \mathbf{app}\,\boldsymbol{Y}\rangle.\mathtt{0} \oplus c!\langle t!.\mathbf{app}\,\boldsymbol{X}\rangle.\mathtt{0}\;) \mid *(\; t?.(\mathsf{app}\,X \oplus \mathsf{app}\,Y)\;)$$

$$c?(X,Y).\nu t.\,(\; c!\langle(\mathsf{app}\,X \mid \mathsf{app}\,Y) \oplus \mathbf{app}\,\boldsymbol{X}\rangle.\mathtt{0} \oplus c!\langle t!.\mathbf{app}\,\boldsymbol{Y}\rangle.\mathtt{0}\;) \mid *(\; t?.(\mathsf{app}\,X \mid \mathsf{app}\,Y)\;) \quad (\ddagger)$$
$$\not\approx c?(X,Y).\nu t.\,(\; c!\langle(\mathsf{app}\,X \mid \mathsf{app}\,Y) \oplus \mathbf{app}\,\boldsymbol{Y}\rangle.\mathtt{0} \oplus c!\langle t!.\mathbf{app}\,\boldsymbol{X}\rangle.\mathtt{0}\;) \mid *(\; t?.(\mathsf{app}\,X \mid \mathsf{app}\,Y)\;)$$

Figure 1: An Equivalence and an inequivalence in higher-order concurrency

1. Introduction

Developing effective reasoning techniques for programming languages with higher-order constructs is a challenging problem, made even more challenging by the presence of concurrency, mobility, and distribution. The difficulties involved are exemplified by the search for reasonable proof techniques for establishing behavioural equivalences between processes written in higher-order versions of the π-calculus [10, 15, 17–23] in which, besides first-order values, processes can be communicated.

To illustrate the challenges of reasoning in higher-order concurrent languages let us consider the pairs of higher-order processes shown in Figure 1, where $\oplus$ is an internal choice operator and app causes the execution of a *suspended process*. The differences within each pair of processes is highlighted. All processes initially receive two suspended processes X and Y on channel c and dynamically create a local channel t. Then each one combines X and Y in a slightly different way into two possible replies on channel c, chosen non-deterministically. After a reply is sent, all that is left of the processes is the same replication (denoted by the $*$-operator) guarded by the private channel t. Up to this point the behaviour of the processes is indistinguishable by a potential interrogator. But from this point on the interrogator may use the values previously emitted from the processes to make further observations.

Let us consider the first process first. It non-deterministically chooses between replying to its interrogator with either (1) the suspended process $(\mathsf{app}\,X \oplus \mathsf{app}\,Y) \mid \mathsf{app}\,X$, or (2) the process $t!.\mathsf{app}\,Y$. When the interrogator runs the former, either two copies of X, or X in parallel with Y, are executed. When it runs the latter, Y is executed after triggering the replication, which in turn executes X or Y. Hence, when the interrogator runs the second of the possible responses of the first process, two copies of Y, or X in parallel with Y, are run. The second process in Figure 1 exhibits the same observable behaviour. The (unobservable) difference with the first process is that it sends the suspended "two copies of Y or X in parallel with Y" directly to the observer and uses the trigger to encode the "two copies of Y or X in parallel with Y" process.

The third process of Figure 1 also chooses between two replies to the interrogator. The first is $(\mathsf{app}\,X \mid \mathsf{app}\,Y) \oplus \mathsf{app}\,X$ which, when run, it will select between running X in parallel with Y, or just X. The second will run two copies of Y in parallel with X, using the trigger channel. The asymmetry with the final process of Figure 1 is clear: this chooses between replying to the interrogator with $(\mathsf{app}\,X \mid \mathsf{app}\,Y) \oplus \mathsf{app}\,Y$ (X in parallel with Y, or just Y) or a suspended process that when run will execute two copies of X in parallel with Y.

The aim of the current paper is to design an elementary bisimulation theory for

higher-order concurrency which, besides being fully-abstract with respect to a semantic equivalence, supports a practical and effective reasoning technique that can be used to prove subtle equivalences and inequivalences such as those in Figure 1. We will formally prove that our intuitive explanation of what an interrogator can observe about those processes is in fact correct. There is no strategy of the interrogator (possibly replicating the processes in the figure and combining their outputs into more complex processes) that can distinguish the first pair of processes, and there is no way of matching the the non-deterministic choices that can deem the second pair of processes equivalent.

Our methodology combines and simplifies ideas carefully selected from the literature [10, 21] and employs a novel treatment of names and extrusion. This results to a purely first-order theory of bisimulation in which actions take a particularly simple form, and in which the examples in Figure 1 can be handled in a straightforward manner. In this paper we study a process-passing π-calculus (pp-π) and bisimulation since this is a simple setting to develop our technique. However, this technique scales to other behavioural theories and languages; for instance we have recently applied this technique to develop a may-testing theory for a higher-order concurrent language with cryptographic primitives [12].

A number of different fully-abstract reasoning techniques have been developed for HOπ including the translation of higher-order communication to triggers in the first-order π-calculus [18, 19] and its improvement in [10], and *environmental bisimulations* [21].

Sangiorgi, Kobayashi, and Sumii [21], motivated by work on environmental bisimulations for functional languages [13, 27, 28], use a standard LTS and annotate bisimulations with an environment (relation) containing the knowledge currently known to the interrogator; this allows the interrogating actions to be meaningfully based on the interrogator's current knowledge. Their method simplifies the metatheory (e.g. showing that bisimilarity is a congruence), but leads to a definition for bisimulations with many and arguably complex conditions. As an example, for higher-order inputs of related processes one has to consider all possible input values constructed by identical contexts with related values in their holes. This strong proof obligation is sometimes mitigated by the use of up-to context techniques.

Jeffrey and Rathke [10] use a more restrictive approach of *formal triggers*, informed by Sangiorgi's translation to triggers [19]. A higher-order output of a process is transformed to a special trigger service holding the actual value and only a pointer for invoking the service is passed to the interrogator. Similarly, a higher-order input is fed with a trigger with which the process can intuitively run the actual value—but actually only an observable action is recorded in the LTS.

We believe that both methods have useful intuitions and that their combination has greater value than the sum of its parts. Our theory incorporates and simplifies their insights.

We use knowledge environments in the LTS, rather than on bisimulations, that record the values exposed to the context, and test related processes with symbolic higher-order inputs. We also take this one step further by including an explicit representation of the information known only to the process. Thus configurations take the form $\nu\tilde{a}\,\langle\Delta,\, P\rangle$ which consists of the (higher-order) process P under interrogation,

a representation Δ of the knowledge currently known to the interrogator about this process, and the information $\overline{a}$ known to the process but currently unknown to the interrogator.

This extension allows us to simplify considerably the actions on which our bisimulations are based; in particular it eliminates the need for explicitly *extruding* new information and communication actions are labelled simply $c!v$ and $c?v$ thereby relieving us of the need to manage the complications inherent in the use of extrusion. A significant consequence is that bisimulation in our theory is characterised by a *propositional* Hennessy-Milner Logic (HML) [8], which would not be possible with other LTS's.

The main contributions of the paper can be summarised as follows:

(i) We define a first-order, fully-abstract, theory of standard weak bisimulation equivalence for a higher-order π-calculus, called pp-π, that unifies two distinct techniques. The theory is compositional in the sense that the equivalence is preserved by arbitrary process contexts.

(ii) The associated coinductive reasoning technique for pp-π processes is effective: because the theory is first-order it is straightforward to demonstrate equivalences between processes by exhibiting witness bisimulations. In support of this we provide a series of compelling example process equivalences.

(iii) We give the first propositional HML characterisation of weak bisimulation for a higher-order π-calculus; this result easily transfers to the first-order π-calculus. We use this to give simple proofs of *in*equivalence between higher-order processes, which is difficult to achieve with existing theories.

(iv) We prove that contextual equivalence in a higher-order setting is a conservative extension of the first-order π-calculus, thus confirming that results and reasoning methods from first-order π-calculus transfer to a higher-order setting.

A direct consequence of our theory is that it brings the analysis of higher-order concurrent and distributed systems within the scope of existing first-order proof technologies.

The remainder of the paper is organised as follows: the next section defines the language pp-π, giving the syntax, a reduction semantics and a simple type system for ensuring that communicated values are appropriately typed. Section 3 details our first-order LTS for pp-π, and Section 4 defines strong and weak bisimulations and a characterisation of the latter in terms of a propositional Hennessy-Milner Logic. Section 5 is devoted to proving several interesting equivalences by using weak bisimulations, and an inequivalence by providing a discriminating HML formula. Sections 6 and 7 contain the proofs of soundness and completeness of our theory with respect to contextual equivalence that preserves only parallel contexts, and Section 8 proves that our theory is fully abstract with respect to the full contextual equivalence. Section 9 proves the conservativity theorem. The paper closes in Section 10 with conclusions and a discussion of related work.

$$
\begin{array}{rcll}
t & ::= & \mathtt{Nm} \mid \mathtt{Pr} & \text{Type} \\
x, y, z & & & \text{Variable} \\
a, b, c, n & & & \text{Name} \\
u, v & \in & \mathsf{Variable} \cup \mathsf{Name} & \text{Identifier} \\
P, Q & ::= & \mathtt{0} \mid u!\langle V{:}t\rangle.P \mid u?(x{:}t).P \mid P \mid P \mid \nu n.\, P & \text{Process} \\
 & \mid & \mathtt{app}\, V \mid *(P) \mid \mathtt{if}\, u = u\, \mathtt{then}\, P\, \mathtt{else}\, P & \\
U, V, W & ::= & u \mid \lambda P & \text{Value}
\end{array}
$$

$\boxed{\Gamma \vdash V : t}$

$$
\frac{}{\Gamma \vdash n : \mathtt{Nm}} \qquad \frac{\Gamma \vdash P : \mathtt{OK}}{\Gamma \vdash \lambda P : \mathtt{Pr}} \qquad \frac{x : t \in \Gamma}{\Gamma \vdash x : t}
$$

$\boxed{\Gamma \vdash P : \mathtt{OK}}$

$$
\frac{}{\Gamma \vdash \mathtt{0} : \mathtt{OK}} \qquad \frac{\Gamma \vdash u : \mathtt{Nm} \quad \Gamma \vdash V : t \quad \Gamma \vdash P : \mathtt{OK}}{\Gamma \vdash u!\langle V{:}t\rangle.P : \mathtt{OK}} \qquad \frac{\Gamma \vdash u : \mathtt{Nm} \quad \Gamma, x : t \vdash P : \mathtt{OK}}{\Gamma \vdash u?(x{:}t).P : \mathtt{OK}}
$$

$$
\frac{\Gamma \vdash P : \mathtt{OK} \quad \Gamma \vdash Q : \mathtt{OK}}{\Gamma \vdash P \mid Q : \mathtt{OK}} \qquad \frac{\Gamma, n : \mathtt{Nm} \vdash P : \mathtt{OK}}{\Gamma \vdash \nu n.\, P : \mathtt{OK}} \qquad \frac{\Gamma \vdash V : \mathtt{Pr}}{\Gamma \vdash \mathtt{app}\, V : \mathtt{OK}} \qquad \frac{\Gamma \vdash P : \mathtt{OK}}{\Gamma \vdash *(P) : \mathtt{OK}}
$$

$$
\frac{\Gamma \vdash u : \mathtt{Nm} \quad \Gamma \vdash v : \mathtt{Nm} \quad \Gamma \vdash P : \mathtt{OK} \quad \Gamma \vdash Q : \mathtt{OK}}{\Gamma \vdash \mathtt{if}\, u = v\, \mathtt{then}\, P\, \mathtt{else}\, Q : \mathtt{OK}}
$$

Figure 2: Syntax and typing of pp-π

2. The Language

2.1. Syntax

We study the language pp-π (process-passing-π), a higher-order version of the π-calculus, which allows processes to be communicated and is roughly equivalent to the language studied in [20].

We assume a set of channel names Name, ranged over by $a, b, \ldots$ and a separate set of variables Variable, ranged over by $x, y, \ldots$, and use $u, v, \ldots$ to denote identifiers, from (Variable $\cup$ Name).

The syntax of the language is given in Figure 2. The basic constructs in pp-π are the input and output of typed values along channels, $u?(x{:}t).P$ and $u!\langle V{:}t\rangle.P$. In the former a value of type t is received on channel u and bound to the variable x in P, while in the latter the value V of type t is output on channel u and the process continues with the execution of the code P. In addition we have the standard constructs of the π-calculus: replication $*(P)$, parallel execution $(P \mid Q)$, the generation of new names $\nu n.\, P$, and the testing of these names $\mathtt{if}\, u = v\, \mathtt{then}\, P\, \mathtt{else}\, Q$.

In the π-calculus the only values which can be transmitted along channels are names, but in pp-π *thunked* or *suspended* processes, of the form λP, are also allowed; when such a value is received by a process it can be executed, via the new construct $\mathtt{app}\, V$.

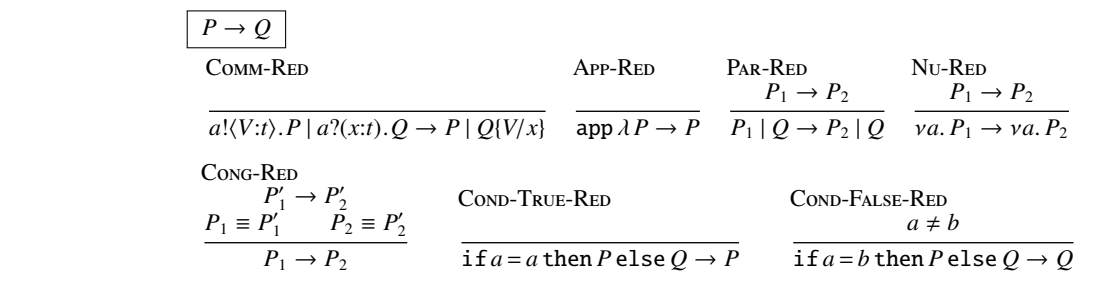


Figure 3: Reduction semantics for pp-π

Typing:. We have a very-lightweight notion of type whose purpose is simply to ensure, dynamically, that at any point in time when a value is received at a certain type it is subsequently only used at that type. Values can be one of two types, `Nm` for names and `Pr` for (suspended) processes. Type inference is with respect to *type environments*, consisting of finite sets of variable-type associations $x : t$. Then the typing judgements take the form

- $\Gamma \vdash V : t$, indicating that relative to Γ the value V has type t
- $\Gamma \vdash P : \mathtt{OK}$, indicating that the process term P is well-typed relative to Γ.

The rules for inferring the judgements are also given in Figure 2.

Typing is *dynamic*: inputs and outputs communicate only when they agree on the type of the communicated value. Therefore channels do not have a single static type. At different points in time they may be used to communicate values of different type. For example a client using the service of the following example at s will be expected to follow an implicit protocol, whereby first a name is sent on s and then a process.

Example. Consider the following process, which describes a service at s:

$$*(s?(x{:}\mathtt{Nm}).s?(y{:}\mathtt{Pr}).\nu f.\,\nu r.\,x!\langle f\rangle.x!\langle r\rangle.\\ *(f?(z{:}\mathtt{Nm}).z!\langle y{:}\mathtt{Pr}\rangle.\mathbb{0}) \mid\\ *(r?.\mathtt{app}\,y))$$

It first receives as input a reply channel name, bound to x, and then a (suspended) process bound to y. It generates two new names f and r which it returns on the reply channel, and then sets up two new servers at those names. The first, at f, receives a name and forwards the suspended process there; the second, at r, runs the suspended process on request.

2.2. Reduction semantics

The reduction semantics is expressed as a relation

$$P \to Q$$

where P and Q are assumed to be well-typed processes, that is process terms satisfying $\emptyset \vdash P : \mathtt{OK}$ and $\emptyset \vdash Q : \mathtt{OK}$. The rules for inferring these judgements are given in Figure 3 and are relatively standard. The main rule is for communication,

$$a!\langle V{:}t\rangle.P \mid a?(x{:}t).Q \to P \mid Q\{V/x\}$$

Note that this communication along a can only happen if the partners agree on the type of the value being transmitted. The other significant rule is for the initiation of a suspended process,

$$\mathsf{app}\,\lambda P \to P$$

The remaining rules are standard, borrowed from the π-calculus; in particular reductions are relative to a structural equivalence $P \equiv Q$ which we now define.

Definition 2.1 (Structural equivalences)**.** Limited structural equivalence ($\hat{=}$) *is defined to be the least equivalence relation on processes satisfying the axioms*

$$P = \mathtt{0} \mid P \qquad P \mid Q = Q \mid P \qquad (P_1 \mid P_2) \mid P_3 = P_1 \mid (P_2 \mid P_3)$$

and closed under the two operators $- \mid -$ *and* $\nu a.\,-$.

Structural equivalence ($\equiv$) *is obtained by adding the further axioms*

$$\nu a.\,\nu b.\,P = \nu b.\,\nu a.\,P \qquad *(P) = P \mid *(P)$$
$$\nu a.\,\mathtt{0} = \mathtt{0} \qquad \nu a.\,(P \mid Q) = (\nu a.\,P) \mid Q \quad (a \notin fn(Q))$$

As we have already stated, structural equivalence ($\equiv$) is used in the reduction semantics, but the more restrictive limited equivalence ($\hat{=}$) will be useful in proofs of equivalence. Note that these structural equivalences are not full congruences. They are only closed under *evaluation contexts*, that is under parallel and restriction, but not under $\lambda -$ and $c!\langle - \rangle.P$. This simplifies the extension of these relations to configurations in Section 4 and the proofs that ($\hat{=}$) and ($\equiv$) are bisimulations (Propositions 4.3 and 4.11, respectively).

Lemma 2.2 (Substitution)**.** *If* $\Gamma, x : t \vdash P : \mathtt{OK}$ *and* $\vdash V : t$ *then* $\Gamma \vdash P\{V/x\} : \mathtt{OK}$.

Proof. By rule induction. □

Lemma 2.3. *If* $\Gamma, x : t \vdash P : \mathtt{OK}$ *and* $x \notin fn(P)$ *then* $\Gamma \vdash P : \mathtt{OK}$.

Proof. By rule induction. □

Lemma 2.4. *If* $P \equiv Q$ *and* $\Gamma \vdash P : \mathtt{OK}$ *then* $\Gamma \vdash Q : \mathtt{OK}$.

Proof. By case analysis on Definition 2.1, using Lemma 2.3. □

Proposition 2.5 (Preservation)**.** *If* $\vdash P : \mathtt{OK}$ *and* $P \to Q$ *then* $\vdash Q : \mathtt{OK}$.

Proof. By rule induction on $P \to Q$, using Lemmas 2.2 and 2.4. □

2.3. A behavioural equivalence

We focus on reasoning about reduction-closed barbed congruence [10, 11, 18, 19, 22] of closed, well-typed processes, but in this section we content ourselves with a simplified version of it. We write $P\ R\ P'$ when R is a binary relation on closed, well-typed processes and $(P, P') \in R$.

We consider the basic observable of a process to be the ability to output on a given channel, called a barb.

Definition 2.6 (Barbs)**.** *We write $P \downarrow_b$ if and only if there exist $\overline{c}, V, t, P_1, P_2$, with $b \notin \{\overline{c}\}$, such that $P \equiv \nu\overline{c}.\,(b!\langle V{:}t\rangle.P_1 \mid P_2)$.*

We write $P \Downarrow_b$ if and only if there exists Q such that $P \rightarrow^* Q$ and $Q \downarrow_b$.

Definition 2.7 (Parallel Semantic Equivalence ($\cong_{\text{pcxt}}$))**.** *($\cong_{\text{pcxt}}$) is the largest relation on closed processes that preserves barbs, is reduction closed, and is preserved by parallel contexts; i.e. $P \cong_{\text{pcxt}} P'$ if and only if*

(i) Barb preserving: *for all b, $P \Downarrow_b$ iff $P' \Downarrow_b$,*

(ii) Reduction closed: *for all P_1 with $P \rightarrow P_1$ there exists P'_1 such that $P' \rightarrow^* P'_1$ and $P_1 \cong_{\text{pcxt}} P'_1$, and vice-versa, and*

(iii) Preserves parallel constructs: *for all well-typed processes Q, $P \mid Q \cong_{\text{pcxt}} P' \mid Q$.*

It is straightforward to show that $\cong_{\text{pcxt}}$ is an equivalence relation. On the other hand to give a direct proof that two processes are related is very difficult, especially in the higher-order π-calculus. In the following sections we define a labelled transition system (LTS) and show that ($\cong_{\text{pcxt}}$) coincides with weak bisimulation in the LTS. We also demonstrate the usefulness of bisimulation as a proof technique of equivalence via several examples. Finally we show that the equivalence remains unchanged if we extend the third requirement (iii) to demand that the relation be preserved by all contexts.

3. The Labelled Transition System

The idea behind an LTS-based semantics for a process language is to describe the interactions which an observer can have with processes; indeed semantic equivalences such as bisimulation equivalence can be expressed in terms of games, and strategies for such games, over these interactions [24].

We first give an informal account of the kinds of interactions we envision for pp-π and then consider their formalisation. For the standard (first-order) π-calculus observers interact with processes via inputs and outputs on channels. But these interactions are constrained by the knowledge which the observer has of the process being interrogated. For example if an observer has no knowledge of channel b then it can not distinguish between the two processes

$$a!.\mathbf{0} \mid b!.\mathbf{0} \qquad\qquad a!.\mathbf{0}$$

as the only possible known source of interaction is the channel a.

In general the observer's knowledge is accumulated by receiving values from the process under interrogation. In pp-π the observer also accumulates knowledge about

higher-order values, and may use these to further interrogate the process. This further interrogation can either take the form of transmitting these values along communication channels, or *executing them*. For example consider the two processes

$$P \stackrel{\text{def}}{=} \nu a.\, c!\langle \lambda a!.\mathtt{0}\rangle.a?.\mathtt{0} \qquad\qquad Q \stackrel{\text{def}}{=} \nu a.\, c!\langle \lambda a!.\mathtt{0}\rangle.a?.c!.\mathtt{0}$$

and an observer which only knows of the channel c. By inputting on c it gains knowledge of the (suspended) process $a!.\mathtt{0}$ although it does not gain any knowledge of the existence of the private channel a. Nevertheless by running this suspended process a difference can be detected between P and Q; in one case output can be detected on channel c after the execution of the suspended process.

However, even in the first-order case, it is necessary for the observer to independently generate new values with which to interrogate the process. For example consider a situation in which the observer is only aware of the channel name a. Then the only way for the observer to distinguish between the two processes

$$a?(x).\mathtt{0} \qquad\qquad a?(x).\mathtt{if}\, x = a\, \mathtt{then}\, \mathtt{0}\, \mathtt{else}\, a!.\mathtt{0}$$

is to generate an new channel name, say b, and send this as input along the known channel a.

In pp-π it is also necessary for the observer to generate new higher-order values with which to interrogate the process, by sending them as inputs. However in our LTS these new higher-order values are simply *abstract constants*, ranged over by α, taken from a countable set AConstant. On receiving such an abstract higher-order value α the processes under interrogation has very little it can do with it; α can only be transmitted as a value along other channels. However, as we will see, our LTS will also allow the process to apply α in a trivial manner. To accommodate these abstract values we need to extend the syntax in Figure 2 to allow them to be used as values. We let $\mathcal{P}$ and $\mathcal{V}$ range over the extended syntax of *abstract processes*, AProcess, and *abstract values*, AValue, respectively; $acon(\mathcal{P})$ denotes the set of abstract constants occurring in $\mathcal{P}$. Furthermore we extend the typing rules to apply to abstract processes and values by adding the following typing judgement for abstract constants:

$$\frac{}{\Gamma \vdash \alpha : \mathtt{Pr}}$$

In order to formalise these kinds of interactions our LTS needs to take into account both the process being interrogated and the current knowledge of the observer, or context. As we have indicated this knowledge is accumulated via interactions with the process, and consists either of (first-order) channel names or higher-order values. To tabulate the latter we use a countable set of *concrete* constants CConstant, disjoint from other kinds of constants, and ranged over by κ.

Definition 3.1 (Knowledge environments)**.** *A knowledge environment Δ is a finite set of the kind*

$$\mathsf{Name} \cup \mathsf{AConstant} \cup (\mathsf{CConstant} \rightarrow_{\text{fin}} \mathsf{AValue})$$

with the property that it maps concrete constants to abstract values of type Pr*:*

$$\Delta(\kappa) = \mathcal{V} \;\; \textit{implies} \;\; \vdash \mathcal{V} : \mathtt{Pr}$$

We write $\Delta(\kappa) = \mathcal{V}$ for $(\kappa, \mathcal{V}) \in \Delta$; we also write $names(\Delta)$, $acon(\Delta)$, and $ccon(\Delta)$ for the name component, the abstract constant component, and the domain of the functional component of Δ, respectively.

Our LTS will be defined between *configurations* of the form $\nu\overline{a}\,\langle\Delta, \mathcal{P}\rangle$, where $\overline{a}$ are names the scope of which extends to $\mathcal{P}$ and the processes indexed in Δ, $\mathcal{P}$ is an abstract process and Δ is a knowledge environment. Configurations are identified up to alpha-equivalence, are ranged over by C, and are subject to the following well-formedness constraints:

Definition 3.2 (Well-Formed Configuration)**.** *A* well-formed *configuration is any configuration $\nu\overline{a}\,\langle\Delta, \mathcal{P}\rangle$ with the properties:*

(i) $\overline{a}$ *are distinct bound names*

(ii) $\{\overline{a}\} \cap names(\Delta) = \emptyset$

(iii) $\vdash \mathcal{P} : \mathtt{OK}$ *and* $fn(\mathcal{P}) \subseteq \{\overline{a}\} \cup names(\Delta)$ *and* $acon(\mathcal{P}) \subseteq acon(\Delta)$

(iv) $\vdash \mathcal{V} : \mathtt{Pr}$ *and* $fn(\mathcal{V}) \subseteq \{\overline{a}\} \cup names(\Delta)$ *and* $acon(\mathcal{V}) \subseteq acon(\Delta)$ *for every $\mathcal{V}$ in the codomain of* Δ.

In a configuration $\nu\overline{a}\,\langle\Delta, \mathcal{P}\rangle$ the environment Δ represents the knowledge of the observer. The names $\overline{a}$ are those known to the process under investigation $\mathcal{P}$, which are not known to the observer, motivating condition (ii); note however that these private names are shared between the process and the abstract values indexed in Δ, values sent to the observer from the process. The remaining conditions guarantee that all processes and values must be well-typed and only use names which are in $\overline{a}$ or are known to the environment, and abstract constants in Δ. For the remainder of this paper we only consider well-formed configurations. When $C = \nu\overline{a}\,\langle\Delta, \mathcal{P}\rangle$ we will write $names(C)$, $acon(C)$, and $ccon(C)$ for $names(\Delta)$, $acon(\Delta)$, and $ccon(\Delta)$, respectively.

The judgements for the LTS take the form

$$\nu\overline{a}\,\langle\Delta, \mathcal{P}\rangle \xrightarrow{\eta} \nu\overline{b}\,\langle\Delta', Q\rangle$$

and the rules for generating them are given in Figure 4 and Figure 5. The label η can take one of the following forms:

(i) *Internal action, τ:* these are the unobservable actions of the process (e.g. internal communication) and weakly correspond to the semantics of Figure 3.

(ii) *First-order input, $c?n$:* input by the process along the channel c, known to the observer, of the name n; n is picked by the observer and (due to well-formedness conditions) it might already be recorded in the knowledge environment or is freshly generated—in both cases n is recorded in the knowledge of the observer after the transition.

(iii) *Higher-order input, $c?\alpha$:* input by the process of an abstract constant α, which is always taken to be fresh.

(iv) *First-order output, $c!n$:* output by the process along the known channel c of the name n. Here the name n may be private to the process or known to the observer. In both cases, n is known to the observer after the transition.

$\boxed{\nu\overline{a_1}\,\langle\Delta_1,\,\mathcal{P}_1\rangle \xrightarrow{\eta} \nu\overline{a_2}\,\langle\Delta_2,\,\mathcal{P}_2\rangle}$

NAME-IN-TRANS
$$\frac{c \in \mathit{names}(\Delta)}{\nu\overline{a}\,\langle\Delta,\, c?(x{:}\mathtt{Nm}).\mathcal{P}\rangle \xrightarrow{c?n} \nu\overline{a}\,\langle\Delta\cup\{n\},\, \mathcal{P}\{n/x\}\rangle}$$

PROC-IN-TRANS
$$\frac{c \in \mathit{names}(\Delta)}{\nu\overline{a}\,\langle\Delta,\, c?(x{:}\mathtt{Pr}).\mathcal{P}\rangle \xrightarrow{c?\alpha} \nu\overline{a}\,\langle\Delta\uplus\{\alpha\},\, \mathcal{P}\{\alpha/x\}\rangle}$$

NAME-OUT-TRANS
$$\frac{c \in \mathit{names}(\Delta) \qquad \{\overline{b}\} = \{\overline{a}\}\backslash\{n\}}{\nu\overline{a}\,\langle\Delta,\, c!\langle n{:}\mathtt{Nm}\rangle.\mathcal{P}\rangle \xrightarrow{c!n} \nu\overline{b}\,\langle\Delta\cup\{n\},\, \mathcal{P}\rangle}$$

PROC-OUT-TRANS
$$\frac{c \in \mathit{names}(\Delta) \qquad \kappa \notin \mathit{ccon}(\Delta)}{\nu\overline{a}\,\langle\Delta,\, c!\langle\mathcal{V}{:}\mathtt{Pr}\rangle.\mathcal{P}\rangle \xrightarrow{c!\kappa} \nu\overline{a}\,\langle\Delta\cup\{\kappa{\mapsto}\mathcal{V}\},\, \mathcal{P}\rangle}$$

COMM-NAME-TRANS
$$\frac{\langle\Delta\uplus\{\overline{a}\},\mathcal{P}_1\rangle \xrightarrow{c!n} \langle\Delta\uplus\{\overline{a}\},\mathcal{P}_2\rangle \qquad \langle\Delta\uplus\{\overline{a}\},\mathcal{Q}_1\rangle \xrightarrow{c?n} \langle\Delta\uplus\{\overline{a}\},\mathcal{Q}_2\rangle}{\nu\overline{a}\,\langle\Delta,\, \mathcal{P}_1 \mid \mathcal{Q}_1\rangle \xrightarrow{\tau} \nu\overline{a}\,\langle\Delta,\, \mathcal{P}_2 \mid \mathcal{Q}_2\rangle}$$

COMM-PROC-TRANS
$$\frac{\langle\Delta\uplus\{\overline{a}\},\mathcal{P}_1\rangle \xrightarrow{c!\kappa} \langle\Delta\uplus\{\overline{a},\kappa{\mapsto}\mathcal{V}\},\mathcal{P}_2\rangle \qquad \langle\Delta\uplus\{\overline{a}\},\mathcal{Q}_1\rangle \xrightarrow{c?\alpha} \langle\Delta\uplus\{\overline{a},\alpha\},\mathcal{Q}_2\rangle}{\nu\overline{a}\,\langle\Delta,\, \mathcal{P}_1 \mid \mathcal{Q}_1\rangle \xrightarrow{\tau} \nu\overline{a}\,\langle\Delta,\, \mathcal{P}_2 \mid \mathcal{Q}_2\{\mathcal{V}/\alpha\}\rangle}$$

ABS-APP-TRANS
$$\frac{}{\nu\overline{a}\,\langle\Delta,\, \mathtt{app}\,\alpha\rangle \xrightarrow{\mathtt{app}\,\alpha} \nu\overline{a}\,\langle\Delta,\, \mathtt{0}\rangle}$$

CONC-APP-TRANS
$$\frac{\Delta(\kappa) = \mathcal{V}}{\nu\overline{a}\,\langle\Delta,\, \mathcal{P}\rangle \xrightarrow{\mathtt{app}\,\kappa} \nu\overline{a}\,\langle\Delta,\, \mathcal{P} \mid \mathtt{app}\,\mathcal{V}\rangle}$$

Figure 4: The LTS: main rules (omitting symmetric rules)

(v) *Higher-order output, $c!\kappa$:* output by the process of some value along the channel c. The concrete constant κ is picked fresh and the actual value output by the process is indexed by κ in Δ.

(vi) *Abstract value application,* $\mathtt{app}\,\alpha$*:* signals the execution by the process of the abstract higher-order value α supplied by the observer. The computational effect of this transition is effectively a noop.

(vii) *Concrete value application,* $\mathtt{app}\,\kappa$*:* the execution of the higher-order value associated with κ in the knowledge environment in parallel with the process. This represents the effect of the observer executing a value originally supplied by the process.

An important aspect of our LTS is the handling of names and extrusion. A (sub-) process with a top-level ν-binder can only take a τ-transition, which lifts the binder to the level of the configuration (NU-TRANS). Since configurations are identified up to renaming bound names, this is essentially an extrusion step. When such a bound name is revealed to the observer via an output transition, the binder is removed and the name is added in knowledge environment (NAME-OUT-TRANS). In this way we greatly simplify the labels of the transitions in our LTS, which allows us to give a *propositional* characterisation of weak bisimilarity, as we will see in Section 4.3.

We require that communication with the observer occurs over known channels, and that the observer never provides a private channel as an input. Hence, a transition

$$\nu\overline{a}\,\langle\Delta,\, \mathcal{P}\rangle \xrightarrow{\eta} \nu\overline{b}\,\langle\Delta',\, \mathcal{Q}\rangle$$

$\boxed{\nu\overline{a_1}\,\langle\Delta_1,\,\mathcal{P}_1\rangle \xrightarrow{\eta} \nu\overline{a_2}\,\langle\Delta_2,\,\mathcal{P}_2\rangle}$

COND-TRUE-TRANS
$$\frac{n_1 = n_2}{\nu\overline{a}\,\langle\Delta,\,\mathtt{if}\,n_1 = n_2\,\mathtt{then}\,\mathcal{P}\,\mathtt{else}\,Q\rangle \xrightarrow{\tau} \nu\overline{a}\,\langle\Delta,\,\mathcal{P}\rangle}$$

REC-TRANS
$$\frac{}{\nu\overline{a}\,\langle\Delta,\,*(\mathcal{P})\rangle \xrightarrow{\tau} \nu\overline{a}\,\langle\Delta,\,\mathcal{P}\mid *(\mathcal{P})\rangle}$$

COND-FALSE-TRANS
$$\frac{n_1 \neq n_2}{\nu\overline{a}\,\langle\Delta,\,\mathtt{if}\,n_1 = n_2\,\mathtt{then}\,\mathcal{P}\,\mathtt{else}\,Q\rangle \xrightarrow{\tau} \nu\overline{a}\,\langle\Delta,\,Q\rangle}$$

NU-TRANS
$$\frac{b \notin \{\overline{a}\}}{\nu\overline{a}\,\langle\Delta,\,\nu b.\mathcal{P}\rangle \xrightarrow{\tau} \nu\overline{a},b\,\langle\Delta,\,\mathcal{P}\rangle}$$

PAR-L-TRANS
$$\frac{\langle\Delta_1 \uplus \{\overline{a}\},\mathcal{P}_1\rangle \xrightarrow{\eta} \nu\overline{b}\,\langle\Delta_2 \uplus \{\overline{a}\},\,\mathcal{P}_2\rangle \qquad \{\overline{c}\} = \{\overline{a}\}\backslash exp(\eta)}{\nu\overline{a}\,\langle\Delta_1,\,\mathcal{P}_1 \mid Q\rangle \xrightarrow{\eta} \nu\overline{b},\overline{c}\,\langle\Delta_2 \cup exp(\eta),\,\mathcal{P}_2 \mid Q\rangle}$$

APP-TRANS
$$\frac{}{\nu\overline{a}\,\langle\Delta,\,\mathtt{app}\,\lambda\mathcal{P}\rangle \xrightarrow{\tau} \nu\overline{a}\,\langle\Delta,\,\mathcal{P}\rangle}$$

Figure 5: The LTS: more rules (omitting symmetric rules)

happens only if

$$s(\eta) \cap \{\overline{a}\} = inp(\eta) \cap \{\overline{a}\} = \emptyset$$

where $s(-)$ and $inp(-)$ return a singleton set containing, respectively, the *subject* of a communication action, the *object* of an input action; in all other cases they return the empty set.

Internal communication for first-order values is captured by COMM-NAME-TRANS, and for higher-order values by COMM-PROC-TRANS. Such communication can take place over channels that are local to the process, hence the temporary addition of the local channels in Δ in the premises. Note that in COMM-PROC-TRANS, the concrete constant κ used to temporarily store the value being communicated is not included in the environment Δ in the conclusion.

In rule PAR-L-TRANS all the private names bound by the configuration are temporarily added in Δ to avoid their alpha renaming without the corresponding renaming of names in Q. The side condition ensures that names exported to the observer are properly added to the knowledge environment: $exp(-)$ returns the object of an output action and the empty set otherwise.

The LTS only increases the knowledge of the observer.

Proposition 3.3. *Suppose* $\nu\overline{a}\,\langle\Delta_1,\,\mathcal{P}\rangle \xrightarrow{\eta} \nu\overline{b}\,\langle\Delta_2,\,Q\rangle$*; then* $\Delta_1 \subseteq \Delta_2$.

Proof. By straightforward rule induction on the transition relation. □

For the rest of this subsection we analyse in considerable detail the structure of the actions in the LTS. First we give an exhaustive analysis of the structure of configurations which are produced by these actions.

Proposition 3.4. *The following properties are true.*

(i) *If* $\nu\overline{a}\,\langle\Delta_1, \mathcal{P}\rangle \xrightarrow{c!n} \nu\overline{b}\,\langle\Delta_2, \mathcal{Q}\rangle$ *then for some* $\mathcal{P}_1$ *and* $\mathcal{P}_2$

$$\mathcal{P} \triangleq c!\langle n\rangle.\mathcal{P}_1 \mid \mathcal{P}_2 \qquad \mathcal{Q} \triangleq \mathcal{P}_1 \mid \mathcal{P}_2 \qquad \Delta_2 = \Delta_1 \cup \{n\} \qquad \{\overline{b}\} = \{\overline{a}\} \setminus \{n\}$$

(ii) *If* $\nu\overline{a}\,\langle\Delta_1, \mathcal{P}\rangle \xrightarrow{c!\kappa} \nu\overline{b}\,\langle\Delta_2, \mathcal{Q}\rangle$ *then for some* $\mathcal{P}_1$, $\mathcal{P}_2$, *and* $\mathcal{V}$

$$\mathcal{P} \triangleq c!\langle\mathcal{V}\rangle.\mathcal{P}_1 \mid \mathcal{P}_2 \qquad \mathcal{Q} \triangleq \mathcal{P}_1 \mid \mathcal{P}_2 \qquad \Delta_2 = \Delta_1 \uplus \{\kappa \mapsto \mathcal{V}\} \qquad \{\overline{b}\} = \{\overline{a}\}$$

(iii) *If* $\nu\overline{a}\,\langle\Delta_1, \mathcal{P}\rangle \xrightarrow{c?n} \nu\overline{b}\,\langle\Delta_2, \mathcal{Q}\rangle$ *then for some* $\mathcal{P}_1$ *and* $\mathcal{P}_2$

$$\mathcal{P} \triangleq c?(x{:}\mathtt{Nm}).\mathcal{P}_1 \mid \mathcal{P}_2 \qquad \mathcal{Q} \triangleq \mathcal{P}_1\{n/x\} \mid \mathcal{P}_2 \qquad \Delta_2 = \Delta_1 \cup \{n\} \qquad \{\overline{b}\} = \{\overline{a}\}$$

(iv) *If* $\nu\overline{a}\,\langle\Delta_1, \mathcal{P}\rangle \xrightarrow{c?\alpha} \nu\overline{b}\,\langle\Delta_2, \mathcal{Q}\rangle$ *then for some* $\mathcal{P}_1$ *and* $\mathcal{P}_2$

$$\mathcal{P} \triangleq c?(x{:}\mathtt{Pr}).\mathcal{P}_1 \mid \mathcal{P}_2 \qquad \mathcal{Q} \triangleq \mathcal{P}_1\{\alpha/x\} \mid \mathcal{P}_2 \qquad \Delta_2 = \Delta_1 \uplus \{\alpha\} \qquad \{\overline{b}\} = \{\overline{a}\}$$

(v) *If* $\nu\overline{a}\,\langle\Delta_1, \mathcal{P}\rangle \xrightarrow{\mathtt{app}\,\alpha} \nu\overline{b}\,\langle\Delta_2, \mathcal{Q}\rangle$ *then for some* $\mathcal{P}_1$

$$\mathcal{P} \triangleq \mathtt{app}\,\alpha \mid \mathcal{P}_1 \qquad \mathcal{Q} \triangleq \mathcal{P}_1 \qquad \Delta_2 = \Delta_1 \qquad \{\overline{b}\} = \{\overline{a}\}$$

(vi) *If* $\nu\overline{a}\,\langle\Delta_1, \mathcal{P}\rangle \xrightarrow{\mathtt{app}\,\kappa} \nu\overline{b}\,\langle\Delta_2, \mathcal{Q}\rangle$ *then for some* $\mathcal{V} = \Delta_1(\kappa)$

$$\mathcal{Q} \triangleq \mathtt{app}\,\mathcal{V} \mid \mathcal{P} \qquad \Delta_2 = \Delta_1 \qquad \{\overline{b}\} = \{\overline{a}\}$$

(vii) *If* $\nu\overline{a}\,\langle\Delta_1, \mathcal{P}\rangle \xrightarrow{\tau} \nu\overline{b}\,\langle\Delta_2, \mathcal{Q}\rangle$ *then* $\Delta_2 = \Delta_1$ *and* $\{\overline{a}\} \subseteq \{\overline{b}\}$.

Proof. All properties are shown by rule induction on the transition relation. □

From the above analysis we conclude that $\mathtt{app}\,\alpha$ transitions can be delayed arbitrarily.

Corollary 3.5 (Delay $\mathtt{app}\,\alpha$)**.** *Let* $C_1 \xrightarrow{\mathtt{app}\,\alpha}\xrightarrow{\eta} C_2$. *Then also* $C_1 \xrightarrow{\eta}\xrightarrow{\mathtt{app}\,\alpha} C_2$.

In a configuration $\nu\overline{a}\,\langle\Delta, \mathcal{P}\rangle$ there are two sources of knowledge, the environment's knowledge in Δ and the internal knowledge of the process in $\overline{a}$. The next result shows that changes to this knowledge has no effect on many actions.

Proposition 3.6.

(i) Knowledge extension: *If* $\nu\overline{a}\,\langle\Delta_1, \mathcal{P}\rangle \xrightarrow{\eta} \nu\overline{b}\,\langle\Delta_2, \mathcal{Q}\rangle$ *and* $\nu\overline{a}, \overline{c}\,\langle\Delta_0 \uplus \Delta_1, \mathcal{P}\rangle$ *is well-formed, and* $\mathit{names}(\eta) \cap \mathit{names}(\Delta_0) = \mathit{acon}(\eta) \cap \mathit{acon}(\Delta_0) = \mathit{ccon}(\eta) \cap \mathit{ccon}(\Delta_0) = \emptyset$ *then*

$$\nu\overline{a}, \overline{c}\,\langle\Delta_0 \uplus \Delta_1, \mathcal{P}\rangle \xrightarrow{\eta} \nu\overline{b}, \overline{c}\,\langle\Delta_0 \uplus \Delta_2, \mathcal{Q}\rangle$$

(ii) Knowledge restriction: *If* $\nu\overline{a}, \overline{c}\,\langle \Delta_0 \uplus \Delta_1, \mathcal{P}\rangle \xrightarrow{\eta} \nu\overline{b}\,\langle \Delta, \mathcal{Q}\rangle$ *and* $\nu\overline{a}\,\langle \Delta_1, \mathcal{P}\rangle$ *is well-formed and* $\eta \neq \mathtt{app}\,\kappa$*, for any* $\kappa \in \Delta_0$*, then there exist* Δ_2 *and* b' *such that* $\Delta = \Delta_0 \cup \Delta_2$*,* $\{\overline{b'}\} = \{\overline{b}\}\backslash\{\overline{c}\}$*, and*

$$\nu\overline{a}\,\langle \Delta_1, \mathcal{P}\rangle \xrightarrow{\eta} \nu\overline{b'}\,\langle \Delta_2, \mathcal{Q}\rangle$$

Proof. In both cases we use rule induction on the transition relation. □

Information in $\nu\overline{a}\,\langle \Delta, \mathcal{P}\rangle$ can also be shifted between the observers knowledge Δ and the processes knowledge $\overline{a}$ without affecting actions, provided of course that information is not used in the actions.

Proposition 3.7 (Unused Information)**.**

(i) Hiding: *Suppose* $\nu\overline{a}\,\langle \Delta_1 \cup \{b\}, \mathcal{P}\rangle \xrightarrow{\eta} \nu\overline{d}\,\langle \Delta_2 \cup \{b\}, \mathcal{Q}\rangle$ *and* b *does not occur in* η*. Then*

$$\nu b, \overline{a}\,\langle \Delta_1, \mathcal{P}\rangle \xrightarrow{\eta} \nu b, \overline{d}\,\langle \Delta_2, \mathcal{Q}\rangle$$

(ii) Revealing: *Conversely, suppose* $\nu b, \overline{a}\,\langle \Delta_1, \mathcal{P}\rangle \xrightarrow{\eta} \nu b, \overline{d}\,\langle \Delta_2, \mathcal{Q}\rangle$ *where again* b *does not occur in* η*. Then*

$$\nu\overline{a}\,\langle \Delta_1 \cup \{b\}, \mathcal{P}\rangle \xrightarrow{\eta} \nu\overline{d}\,\langle \Delta_2 \cup \{b\}, \mathcal{Q}\rangle$$

Proof. Again, by rule induction. □

With reference to this proposition there are actually very limited ways in which an action η from the configuration $\nu b, \overline{a}\,\langle \Delta_1, \mathcal{P}\rangle$ can use the name b. Indeed the only possibility is an output action, which by Proposition 3.4 must have the form $\nu b, \overline{a}\,\langle \Delta, \mathcal{P}\rangle \xrightarrow{c!b} \nu\overline{d}\,\langle \Delta \cup \{b\}, \mathcal{Q}\rangle$; and this action can still be performed when the observer knows of the existence of b:

Proposition 3.8 (Extrusion)**.** *Provided* c *is different than* b*,*

$$\nu b, \overline{a}\,\langle \Delta, \mathcal{P}\rangle \xrightarrow{c!b} \nu\overline{d}\,\langle \Delta \cup \{b\}, \mathcal{Q}\rangle \quad \textit{iff} \quad \nu\overline{a}\,\langle \Delta \cup \{b\}, \mathcal{P}\rangle \xrightarrow{c!b} \nu\overline{d}\,\langle \Delta \cup \{b\}, \mathcal{Q}\rangle$$

Proof. By rule induction, in both directions. □

Our behavioural theory is based on *weak transitions* in the LTS. We write $\stackrel{\eta}{\Longrightarrow}$ to mean the reflexive, transitive closure of $\xrightarrow{\tau}$, when $\eta = \tau$, and $\stackrel{\tau}{\Longrightarrow}\xrightarrow{\eta}\stackrel{\tau}{\Longrightarrow}$, otherwise.

Traces are sequences of actions; *weak traces* are sequences of observable actions. We let t range over (weak) traces.

Definition 3.9 (Trace)**.** *A configuration* C *transitions to* C' *by a trace* t*, and we write* $C \xrightarrow{t} C'$*, according to the rules:*

(i) $C \xrightarrow{\epsilon} C$

(ii) $C \xrightarrow{\eta,t} C'$ *when there exists* C'' *such that* $C \xrightarrow{\eta} C'' \xrightarrow{t} C'$

Similarly, C *transitions to* C' *by a weak trace* t*, and we write* $C \stackrel{t}{\Longrightarrow} C'$*, according to the rules:*

(i) $C \stackrel{\epsilon}{\Longrightarrow} C'$ *when* $C \stackrel{\tau}{\Longrightarrow} C'$

(ii) $C \stackrel{\eta,t}{\Longrightarrow} C'$ *when* $\eta \neq \tau$ *and there exists* C'' *such that* $C \stackrel{\eta}{\Longrightarrow} C'' \stackrel{t}{\Longrightarrow} C'$

4. Bisimulations

In this section we give the definitions for strong and weak bisimulations. We prove that the limited structural equivalence ($\hat{=}$) is a strong bisimulation and the full structural equivalence ($\equiv$) is a weak bisimulation over configurations. We also prove several useful weak bisimulations that encode properties of local and global names. Finally we give a characterisation of weak bisimilarity in terms of a propositional Hennessy-Milner Logic.

4.1. Strong Bisimulations

We start with the definition of *strong bisimulation*, a rather strict equivalence on configurations which will be useful later for deriving technical results.

We write binary relations on well-formed configurations as $\mathbb{R}$, $\mathbb{X}$, etc.

Definition 4.1 (Strong Bisimulation)**.** $\mathbb{R}$ *is a strong bisimulation if and only if for all* $C\ \mathbb{R}\ C'$*:*

(i) If $C \xrightarrow{\eta} C_1$ *then there exists* C'_1 *such that* $C' \xrightarrow{\eta} C'_1$ *and* $C_1\ \mathbb{R}\ C'_1$.

(ii) The converse of (i)

Strong bisimulations are closed under unions. Thus the union of all strong bisimulations is the largest strong bisimulation; it is also easy to see that it is an equivalence relation.

Definition 4.2 (Strong Bisimilarity ($\sim$))**.** ($\sim$) *is the largest strong bisimulation.*

The limited structural equivalence from Definition 2.1 can be extended to configurations in the obvious manner. First it is extended to abstract processes by applying the axioms and rules in Definition 2.1. Then we let $\nu\overline{a}\,\langle\Delta, \mathcal{P}\rangle \mathrel{\hat{=}} \nu\overline{a'}\,\langle\Delta', \mathcal{P}'\rangle$ whenever $\mathcal{P} \mathrel{\hat{=}} \mathcal{P}'$, $\overline{a} = \overline{a'}$ and $\Delta = \Delta'$.

Proposition 4.3. ($\hat{=}$) *is a strong bisimulation over configurations.*

Proof. By using induction on the rules of ($\hat{=}$); i.e. the rules shown in Definition 2.1 and the standard rules for an equivalence, we can show that all moves from related configurations can be appropriately matched. Structural equivalence allows the reordering of *running* parallel processes but not suspended processes under $c!\langle - \rangle.P$ (or $\lambda -$). Therefore, the knowledge environments of configurations related by ($\hat{=}$) will remain equal after any transition (esp. any higher-order output transition). □

4.2. Weak Bisimulations

Our theory of behavioural equivalence is based on weak bisimulations, which use *weak* actions from the LTS of the previous section.

Definition 4.4 (Weak Bisimulation)**.** $\mathbb{R}$ *is a bisimulation if and only if for all* $C\ \mathbb{R}\ C'$*:*

(i) If $C \xrightarrow{\eta} C_1$ *then there exists* C'_1 *such that* $C' \overset{\eta}{\Longrightarrow} C'_1$ *and* $C_1\ \mathbb{R}\ C'_1$*.*

(ii) The converse of (i)

The collection of weak bisimulations is closed under unions, and thus the union of all weak bisimulations is the largest weak bisimulation; again it is straightforward to show that this is also an equivalence relation.

Definition 4.5 (Weak Bisimilarity ($\approx$))**.** ($\approx$) *is the largest weak bisimulation.*

The freshness condition for concrete constants in higher-order output transitions (rule Proc-Out-Trans in Figure 4) only considers names in the configuration that performs the transition. In bisimulation proofs we will never need to assume that this condition extends over a related configuration because environments of bisimilar configurations contain the same concrete constants. Therefore this condition does not pose any complication in such proofs.

Lemma 4.6. *If* $\nu\overline{a}\,\langle \Delta, \mathcal{P}\rangle \approx \nu\overline{a'}\,\langle \Delta', \mathcal{P}'\rangle$ *then* $ccon(\Delta) = ccon(\Delta')$*.*

Proof. Let $\kappa \in ccon(\Delta)$. Then $\nu\overline{a}\,\langle \Delta, \mathcal{P}\rangle$ has an $\mathsf{app}\,\kappa$ transition to a configuration C. By definition of bisimulation $\nu\overline{a'}\,\langle \Delta', \mathcal{P}'\rangle$ also has an $\mathsf{app}\,\kappa$ transition to some configuration C' (and $C \approx C'$), which is only possible if $\kappa \in ccon(\Delta')$. Similarly for the reverse direction. □

A consequence of the above lemma is that weak bisimilarity is equivariant for concrete constants.

Lemma 4.7 (Equivariance for concrete constants)**.** *If* $C \approx C'$ *then* $C\{\kappa_1/\kappa_2\} \approx C'\{\kappa_1/\kappa_2\}$*.*

Proof. By closing ($\approx$) under renaming of concrete constants and showing that the obtained relation is a weak bisimulation. □

In the meta-theory and applications we will confine our attention to relations over configurations which have the same *acon* components. These are relations that have been generated by source-level processes with no abstract constants, closed under the same transitions, including the same higher-order input transitions which introduce the abstract constants. Therefore, as with the freshness condition for concrete constants in output transitions, the condition for fresh abstract constants in input transitions (rule Proc-In-Trans in Figure 4) does not pose any difficulty in bisimulation proofs since it never needs to extend over names other than those in the configuration itself. Bisimilarity is equivariant for configurations with the same *acon* components.

Lemma 4.8 (Equivariance for abstract constants)**.** *If* $C \approx C'$ *and* $acon(C) = acon(C')$ *then* $C\{\alpha_1/\alpha_2\} \approx C'\{\alpha_1/\alpha_2\}$*.*

Proof. By showing that the following relation is a weak bisimulation.

$$\mathbb{X} \stackrel{\text{def}}{=} \{(C\{\alpha_1/\alpha_2\}, C'\{\alpha_1/\alpha_2\}) \mid \text{any } \alpha_1, \alpha_2,\ \ acon(C) = acon(C'),\ \ C \approx C'\} \qquad \square$$

Our primary concern is the ability for our bisimulations to support reasoning about process behaviour. To this end we extend weak bisimilarity to closed processes as follows:

Definition 4.9. *We write $P \simeq P'$ if and only if there exist $\overline{b}$ such that*

$$\langle\{\overline{b}\}, P\rangle \approx \langle\{\overline{b}\}, P'\rangle$$

Note that since $(\approx)$ is only defined between well-formed configurations the names $\overline{b}$ in the above definition include the free names of P and P'.

As with $(\hat{=})$, we extend the structural equivalence $(\equiv)$ to abstract processes in the usual way, and to LTS configurations as follows; note that this is extension is slightly more general than that used for the limited structural equivalence $(\hat{=})$.

Definition 4.10 ($(\equiv)$ on LTS configurations)**.** *We write $\nu\overline{a}\,\langle\Delta, \mathcal{P}\rangle \equiv \nu\overline{a'}\,\langle\Delta', \mathcal{P}'\rangle$ if and only if*

$$\nu\overline{a}.\mathcal{P} \equiv \nu\overline{a'}.\mathcal{P}' \qquad \Delta = \Delta'$$

Proposition 4.11. $(\equiv)$ *is a weak bisimulation over configurations.*

Proof (sketch). Suppose

$$\nu\overline{a}\,\langle\Delta_1, \mathcal{P}\rangle \xrightarrow{\eta} \nu\overline{b}\,\langle\Delta_2, \mathcal{Q}\rangle \qquad \text{and } \ \nu\overline{a}\,\langle\Delta_1, \mathcal{P}\rangle \equiv \nu\overline{a'}\,\langle\Delta'_1, \mathcal{P}'\rangle$$

We show that

$$\nu\overline{a'}\,\langle\Delta'_1, \mathcal{P}'\rangle \stackrel{\eta}{\Longrightarrow} \nu\overline{b'}\,\langle\Delta'_2, \mathcal{Q}'\rangle$$

for some $\nu\overline{b'}\,\langle\Delta'_2, \mathcal{Q}'\rangle \equiv \nu\overline{b}\,\langle\Delta_2, \mathcal{Q}\rangle$, and vice-versa.

We proceed by induction on the proof that $\nu\overline{a}.\mathcal{P} \equiv \nu\overline{a'}.\mathcal{P}'$. The base cases are provided by the axioms for $(\equiv)$ in Definition 2.1 and reflexivity. The only complication here involves any outermost ν-binders in the processes of the axioms: for each binder we distinguish the case where it is a binder at the level of the configuration and the case where it is a binder in the process part of the configuration. In all cases, the behaviour of the related processes are identical, modulo the τ-transitions that extrude a binder to the level of the configuration.

The cases for symmetry and transitivity are shown by straightforward applications of the induction hypothesis.

For the case of closure under $(- \mid -)$ we have $\nu\overline{a}.\mathcal{P} = \mathcal{P}_1 \mid \mathcal{P}_2$ and $\nu\overline{a'}.\mathcal{P}' = \mathcal{P}'_1 \mid \mathcal{P}'_2$, for some $\mathcal{P}_1, \mathcal{P}_2, \mathcal{P}'_1$, and $\mathcal{P}'_2$, with $\mathcal{P}_1 \equiv \mathcal{P}'_1$ and $\mathcal{P}_2 \equiv \mathcal{P}'_2$. Hence $\{\overline{a}\} = \{\overline{a'}\} = \emptyset$. We proceed by cases on the η-transition. The only applicable cases are Comm-Name-Trans, Comm-Proc-Trans, and Par-L-Trans, which are all proved by straightforward applications of the induction hypothesis and Proposition 3.4.

The case for closure under $(\nu n.\,-)$ follows by the induction hypothesis and Propositions 3.7 and 3.8.

As with the proof of Proposition 4.3, the environments of configurations related by $(\equiv)$ remain equal after higher-order output transitions because $(\equiv)$ is not closed under $c!\langle - \rangle.P$ (or $\lambda -$). □

Corollary 4.12. $(\equiv) \subseteq (\simeq)$.

Extension of knowledge environments with identical names preserves weak bisimilarity.

Lemma 4.13. *If* $\nu\overline{a}\,\langle \Delta,\, \mathcal{P}\rangle \approx \nu\overline{a'}\,\langle \Delta',\, \mathcal{P}'\rangle$ *and* $n \notin \{\overline{a}, \overline{a'}\}$ *then*

$$\nu\overline{a}\,\langle \Delta \cup \{n\},\, \mathcal{P}\rangle \approx \nu\overline{a'}\,\langle \Delta' \cup \{n\},\, \mathcal{P}'\rangle$$

Proof. Let

$$\mathbb{X} = \{(\nu\overline{a}\,\langle \Delta \cup \{\overline{n}\},\, \mathcal{P}\rangle, \nu\overline{a'}\,\langle \Delta' \cup \{\overline{n}\},\, \mathcal{P}'\rangle) \mid \nu\overline{a}\,\langle \Delta,\, \mathcal{P}\rangle \approx \nu\overline{a'}\,\langle \Delta',\, \mathcal{P}'\rangle \quad \{\overline{n}\} \cap \{\overline{a}, \overline{a'}\} = \emptyset\}$$

It is easy to show that $\mathbb{X}$ is a weak bisimulation using Proposition 3.6. □

Hiding names also preserves weak bisimilarity.

Lemma 4.14. *If* $\nu\overline{a}\,\langle \Delta \uplus \{n\},\, \mathcal{P}\rangle \approx \nu\overline{a'}\,\langle \Delta' \uplus \{n\},\, \mathcal{P}'\rangle$ *then*

$$\nu\overline{a}, n\,\langle \Delta,\, \mathcal{P}\rangle \approx \nu\overline{a'}, n\,\langle \Delta',\, \mathcal{P}'\rangle$$

Proof. Similar to the above proof. □

Furthermore, extrusion of private names to the level of the configuration is indistinguishable by weak bisimilarity.

Lemma 4.15. $\nu a, \overline{b}\,\langle \Delta,\, \mathcal{P}\rangle \approx \nu\overline{b}\,\langle \Delta,\, \nu a.\mathcal{P}\rangle$

Proof. Trivial. □

Lemma 4.16.

$$\nu a, \overline{b}\,\langle \Delta,\, \mathcal{P}\rangle \approx \nu a', \overline{b'}\,\langle \Delta',\, \mathcal{P}'\rangle \quad \textit{iff} \quad \nu\overline{b}\,\langle \Delta,\, \nu a.\mathcal{P}\rangle \approx \nu\overline{b'}\,\langle \Delta',\, \nu a'.\mathcal{P}'\rangle$$

Proof. By Lemma 4.15 and transitivity of $(\approx)$. □

4.3. Logical Characterisation

Weak bisimilarity is characterised by a propositional Hennessy-Milner Logic with the following syntax.

$$F ::= \neg F \mid \bigwedge_{i \in I} F_i \mid \langle \eta \rangle F$$

where I is a (possibly infinite) indexing set.

These formulas define a set of basic properties satisfied by configurations of our LTS. The construct $\neg F$ encodes negation and $\bigwedge_{i \in I} F_i$ encodes (possibly infinite) propositional conjuction. The modal construct $\langle \eta \rangle F$ encodes the property that there is a weak η-transition to a configuration that satisfies F.

The semantics of this logic is given by a satisfaction relation $C \models F$ between a configuration C and a formula F.

Definition 4.17 (Satisfaction Relation ($C \models F$))**.**

$$\begin{array}{lll} C \models \neg F & \textit{iff} & C \not\models F \\ C \models \bigwedge_{i \in I} F_i & \textit{iff} & \forall i \in I.\ C \models F_i \\ C \models \langle \eta \rangle F & \textit{iff} & \exists C'.\ C \stackrel{\eta}{\Longrightarrow} C' \textit{ and } C' \models F \end{array}$$

As usual, more predicates are derivable; e.g.:

$$\begin{array}{lcl} C \models \mathtt{tt} & \stackrel{\text{def}}{=} & C \models \bigwedge_{i \in \emptyset} F_i \\ C \models \mathtt{ff} & \stackrel{\text{def}}{=} & C \models \neg \mathtt{tt} \\ C \models [\eta] F & \stackrel{\text{def}}{=} & C \models \neg \langle \eta \rangle \neg F \\ C \models \bigvee_{i \in I} F_i & \stackrel{\text{def}}{=} & C \models \neg \bigwedge_{i \in I} \neg F_i \\ C \models F_1 \wedge F_2 & \stackrel{\text{def}}{=} & C \models \bigwedge_{i \in \{1,2\}} F_i \\ C \models F_1 \vee F_2 & \stackrel{\text{def}}{=} & C \models \bigvee_{i \in \{1,2\}} F_i \end{array}$$

As the transition labels η in our LTS contain actual (not extruded) names, i.e. constants, the above logic is similar to that of the CCS ([16], Chapter 10). Hence, we avoid the complications of extrusion and generation of fresh names in the logic.

The main theorem in this section is the characterisation of weak bisimilarity by the logic.

Theorem 4.18. *$C \approx C'$ if and only if for all F*

$$C \models F \textit{ iff } C' \models F$$

Proof. For the forward direction we proceed by structural induction, taking cases on the formula F:

Case $F = \neg F_0$: By the induction hypothesis,

$$C \models F_0 \textit{ iff } C' \models F_0$$

hence, by Definition 4.17, $C \models \neg F_0$ *iff* $C' \models \neg F_0$.

♦ $F = \bigwedge_{i \in I} F_i$: by Definition 4.17,

$$C \models \bigwedge_{i \in I} F_i \;\; \textit{iff} \;\; (\forall i \in I.\, C \models F_i)$$
$$C' \models \bigwedge_{i \in I} F_i \;\; \textit{iff} \;\; (\forall i \in I.\, C' \models F_i)$$

By the induction hypothesis,

$$\forall i \in I.\, C \models F_i \textit{ iff } C' \models F_i$$

and by the definition of the (possibly infinite) conjuction $C \models \bigwedge_{i \in I} F_i$ *iff* $C' \models \bigwedge_{i \in I} F_i$.

♦ $F = \langle \eta \rangle F_0$: If $C \models \langle \eta \rangle F_0$ then, by Definition 4.17, there exists C_0 such that

$$C \stackrel{\eta}{\Longrightarrow} C_0 \qquad C_0 \models F_0$$

Because $C \approx C'$, there exists C'_0 such that

$$C' \stackrel{\eta}{\Longrightarrow} C'_0 \qquad C_0 \approx C'_0$$

Hence, by the induction hypothesis, it must be that $C'_0 \models F_0$, and, by Definition 4.17, $C' \models \langle \eta \rangle F_0$. Similarly if $C' \models \langle \eta \rangle F_0$.

For the converse direction of the theorem we define the following relation.

$$\mathbb{R} = \{(C, C') \mid \forall F.\, C \models F \textit{ iff } C' \models F\}$$

We show *by contradiction* that $\mathbb{R}$ is a weak bisimulation:

We assume that $\mathbb{R}$ is not a bisimulation. Because $\mathbb{R}$ is obviously symmetric, w.l.o.g., this means that for some $(C, C') \in \mathbb{R}$ there exists C_1 such that

$$C \stackrel{\eta}{\longrightarrow} C_1 \qquad \forall C'_i \in S.\, (C_1, C'_i) \notin \mathbb{R}$$

where $S = \{C'_i \mid C' \stackrel{\eta}{\Longrightarrow} C'_i\}$. By the definition of $\mathbb{R}$, for every $C'_i \in S$ there exists F_i such that

$$C_1 \models F_i \qquad C'_i \not\models F_i$$

or vice-versa, but in this case we consider $\neg F_i$. Hence, if I contains the indices of exactly these formulas,

$$C_1 \models \bigwedge_{i \in I} F_i$$

and therefore

$$C \models \langle \eta \rangle (\bigwedge_{i \in I} F_i) \qquad C' \not\models \langle \eta \rangle (\bigwedge_{i \in I} F_i)$$

which contradicts the fact that $(C, C') \in \mathbb{R}$. □

An immediate consequence of this theorem is that the logic is particularly useful in giving simple proofs of inequivalence. In Section 5.5 we prove such an inequivalence by providing an HML formula that is satisfied by one of the processes and not the other.

5. Examples

Here we illustrate the effectiveness of our theory by giving simple proofs of equivalence using first-order weak bisimulation, and of inequivalence using the Hennessy-Milner Logic. Many of our examples involve ping servers and triggers, which in our opinion get to the heart of the challenges of reasoning about higher-order concurrent processes.

All equivalences can be proved using the standard weak bisimulation. However, to improve presentation, we develop a lightweight up-to β and the limited structural equivalence ($\hat{=}$) technique similar to that in [7], Chapter 6. β-moves are τ-transitions that are confluent with all other transitions.

Definition 5.1 (β-move ($\xrightarrow{\tau_\beta}$))**.** *A τ-transition $C_1 \xrightarrow{\tau} C_2$ is a β-move and we write $C_1 \xrightarrow{\tau_\beta} C_2$ if and only if for all transitions $C_1 \xrightarrow{\eta} C_3$ one of the following is true:*

(i) $\eta = \tau$ and $C_2 = C_3$, or

(ii) there exists C_4 such that $C_2 \xrightarrow{\eta} C_4$ and $C_3 \xrightarrow{\tau_\beta} C_4$.

Definition 5.2 (Weak Bisimulation up-to β and ($\hat{=}$))**.** *A relation $\mathbb{R}$ on configurations is a weak bisimulation up-to β and ($\hat{=}$) if and only if for all $C\ \mathbb{R}\ C'$,*

(i) if $C \xrightarrow{\eta} C_1$ then, for some C_1',

$$C' \stackrel{\eta}{\Longrightarrow} C_1' \qquad C_1 \xrightarrow{\tau_\beta}{}^* \hat{=}\ \mathbb{R}\ \hat{=}\ C_1'$$

(ii) the converse of (i)

Proposition 5.3. *The relation ($\xrightarrow{\tau_\beta}{}^* \hat{=}$) is transitive.*

Proof. By induction on the rules of ($\hat{=}$). □

It is easy to verify that ($\approx$) is a weak bisimulation up-to β and ($\hat{=}$). Any weak bisimulation up-to β and ($\hat{=}$) is included in ($\approx$):

Proposition 5.4. *If $\mathbb{R}$ is a weak bisimulation up-to β and ($\hat{=}$), then $\mathbb{R} \subseteq (\xrightarrow{\tau_\beta}{}^* \hat{=}\ \mathbb{R}\ \hat{=} \xleftarrow{\tau_\beta}{}^*) \subseteq (\approx)$.*

Proof. Because ($\xrightarrow{\tau_\beta}{}^*$) and ($\hat{=}$) contain the identity, $\mathbb{R} \subseteq (\xrightarrow{\tau_\beta}{}^* \hat{=}\ \mathbb{R}\ \hat{=} \xleftarrow{\tau_\beta}{}^*)$. Thus, it suffices to show that ($\xrightarrow{\tau_\beta}{}^* \hat{=}\ \mathbb{R}\ \hat{=} \xleftarrow{\tau_\beta}{}^*$) is a weak bisimulation.

Let

$$C_1 \xrightarrow{\tau_\beta}{}^* C_2 \hat{=} C_3\ \mathbb{R}\ C_3' \hat{=} C_2' \xleftarrow{\tau_\beta}{}^* C_1'$$

then

$$
\begin{array}{lll}
 & C_1 \xrightarrow{\eta} C_4 & \\
\textit{implies} & C_2 \xrightarrow{\eta} C_5 \ \wedge\ C_4 \xrightarrow{\tau_\beta}{}^{*}\ C_5 & \text{(for some } C_5\text{, by definition of } (\xrightarrow{\tau_\beta})) \\
\textit{implies} & C_3 \xrightarrow{\eta} C_6 \ \wedge\ C_5 \mathrel{\hat{=}} C_6 & \text{(for some } C_6\text{, because } (\hat{=}) \text{ is a strong bisimulation)} \\
\textit{implies} & C_3' \xRightarrow{\eta} C_6' \ \wedge\ C_6 \xrightarrow{\tau_\beta}{}^{*} \mathrel{\hat{=}} \mathbb{R} \mathrel{\hat{=}} C_6' & \text{(for some } C_6'\text{, because } \mathbb{R} \text{ is a weak bisimulation up-to } \beta \text{ and } (\hat{=})) \\
\textit{implies} & C_2' \xRightarrow{\eta} C_5' \ \wedge\ C_6' \mathrel{\hat{=}} C_5' & \text{(for some } C_5'\text{, because } (\hat{=}) \text{ is a strong bisimulation)} \\
\textit{implies} & C_1' \xRightarrow{\eta} C_5' & \text{(because } \tau_\beta\text{-moves are } \tau\text{-steps)}
\end{array}
$$

Hence, we have that for some C_5', $C_1' \xRightarrow{\eta} C_5'$ and

$$C_4 \xrightarrow{\tau_\beta}{}^{*} \mathrel{\hat{=}} \xrightarrow{\tau_\beta}{}^{*} \mathrel{\hat{=}} \mathbb{R} \mathrel{\hat{=}} C_5'$$

and by Proposition 5.3 and the fact that $(\xleftarrow{\tau_\beta}{}^{*})$ contains the identity we get

$$C_4 \xrightarrow{\tau_\beta}{}^{*} \mathrel{\hat{=}} \mathbb{R} \mathrel{\hat{=}} \xleftarrow{\tau_\beta}{}^{*} C_5'$$

Similarly we prove the converse condition of Definition 4.4. □

Using weak bisimulation up-to β and $(\hat{=})$ we prove several interesting equivalences in the following sections.

5.1. Implementation of Replication

For our first example we consider an encoding of replication via higher-order communication. The following process receives a suspended process on channel p which then replicates and runs.

$$
\begin{aligned}
Rec &\stackrel{\text{def}}{=} p?(X).\nu a.\,(R \mid a!\langle \lambda\, \mathtt{app}\, X \mid R\rangle.\mathbb{0}) \\
R &\stackrel{\text{def}}{=} a?(X).(\mathtt{app}\, X \mid a!\langle X\rangle.\mathbb{0})
\end{aligned}
$$

We show that this is weakly bisimilar to

$$Rec' \stackrel{\text{def}}{=} p?(X).*(\mathtt{app}\, X)$$

Namely, we prove that $Rec \simeq Rec'$, which by definition amounts to proving

$$\langle \{p\}, Rec\rangle \approx \langle \{p\}, Rec'\rangle$$

To prove this we will provide a relation $\mathbb{R}$ on configurations that relates $\langle \{p\}, Rec\rangle$ and $\langle \{p\}, Rec'\rangle$ and show that it is a bisimulation up-to $(\hat{=})$.

Let us first consider the configurations reachable from $\langle\{p\}, Rec\rangle$ that are relevant to our proof. These can be partitioned to the following families of configurations.

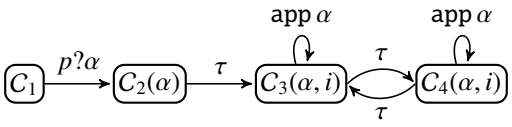
$$\begin{aligned}
C_1 &\triangleq \langle\{p\}, Rec\rangle \\
C_2(\alpha) &\triangleq \langle\{p, \alpha\}, \nu a.\ R \mid a!\langle\lambda(\mathtt{app}\,\alpha \mid R)\rangle.\mathtt{0}\rangle \\
C_3(\alpha, i) &\triangleq \nu a\,\langle\{p, \alpha\},\ R \mid a!\langle\lambda(\mathtt{app}\,\alpha \mid R)\rangle.\mathtt{0} \mid \textstyle\prod_i \mathtt{app}\,\alpha\rangle \\
C_4(\alpha, i) &\triangleq \nu a\,\langle\{p, \alpha\},\ \mathtt{app}\,\lambda(\mathtt{app}\,\alpha \mid R) \mid a!\langle\lambda(\mathtt{app}\,\alpha \mid R)\rangle.\mathtt{0} \mid \textstyle\prod_i \mathtt{app}\,\alpha\rangle
\end{aligned}$$

Similarly, all configurations reachable from $\langle\{p\}, Rec'\rangle$ are members of one of the two families of configurations

$$\begin{aligned}
C'_1 &\triangleq \langle\{p\}, Rec'\rangle \\
C'_2(\alpha, i) &\triangleq \langle\{p, \alpha\}, *(\mathtt{app}\,\alpha) \mid \textstyle\prod_i \mathtt{app}\,\alpha\rangle
\end{aligned}$$

In this and following sections we visualise the structure of each LTS involved in a bisimulation proof by a more abstract Kripke-like structure that uses families of configurations. Each node in the structure represents a family of configurations; the parameters of each family are quantified at each state. A labelled arrow between families of configurations exists if there is a configuration belonging to the originating family that has an LTS transition with the same label to a configuration in the target family. We sometimes identify transitions with the same originating and target families using metavariables.

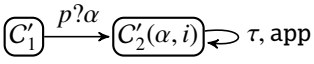
Here the possible-worlds structure that corresponds to *Rec* is



This picture has an arrow labelled $p?\alpha$ from C_1 to C_2 because of the LTS transition $C_1 \xrightarrow{p?\alpha} C_2(\alpha)$ that inputs an abstract constant α on channel p. The τ-labelled arrow from C_2 to C_3 is because of the the transition $C_2(\alpha) \xrightarrow{\tau} C_3(\alpha, 0)$ that extrudes the private name a to the level of the configuration. The τ-labelled arrow from C_3 to C_4 is due to the transitions $C_3(\alpha, i) \xrightarrow{\tau} C_4(\alpha, i)$ that communicate the value $\lambda(\mathtt{app}\,\alpha \mid R)$ over the channel a. The remaining τ-arrow is a result of the application of $\lambda(\mathtt{app}\,\alpha \mid R)$: $C_4(\alpha, i) \xrightarrow{\tau} C_3(\alpha, i+1)$ that produces one more process $\mathtt{app}\,\alpha$. The self-loops on the configurations C_3 and C_4, labelled $\mathtt{app}\,\alpha$, are because of the transitions that apply a process $\mathtt{app}\,\alpha$:

$$C_3(\alpha, i+1) \xrightarrow{\mathtt{app}\,\alpha} C_3(\alpha, i) \qquad C_4(\alpha, i+1) \xrightarrow{\mathtt{app}\,\alpha} C_4(\alpha, i)$$

Similarly the LTS that corresponds to Rec' can be abstracted by the following picture.



Here we have the input $p?\alpha$ due to the transition $C'_1 \xrightarrow{p?\alpha} C'_2(\alpha, 0)$, and the τ-labelled loop on C'_2 due to an unfolding of the replication

$$C'_2(\alpha, i) \xrightarrow{\tau} C'_2(\alpha, i+1)$$

The $\mathtt{app}\,\alpha$-labelled loop is because of applications of process $\mathtt{app}\,\alpha$:

$$C'_2(\alpha, i+1) \xrightarrow{\mathtt{app}\,\alpha} C'_2(\alpha, i)$$

We validate that the following relation is a weak bisimulation up-to ($\hat{=}$).

$$\begin{aligned}\mathbb{R} = \{\ & (C_1, C'_1),\ (C_2(\alpha), C'_2(\alpha, 0)),\ (C_3(\alpha, i), C'_2(\alpha, i)),\\ & (C_4(\alpha, i), C'_2(\alpha, i))\ \mid\ \alpha, i\ \}\end{aligned}$$

Indeed, the transition $C_1 \xrightarrow{p?\alpha} C_2(\alpha)$ is matched by $C'_1 \xrightarrow{p?\alpha} C'_2(\alpha, 0)$ The τ-transitions $C_2(\alpha) \xrightarrow{\tau} C_3(\alpha, 0)$ and $C_3(\alpha, i) \xrightarrow{\tau} C_4(\alpha, i)$ are matched by zero τ-transitions from $C'_2(\alpha, 0)$ and $C'_2(\alpha, i)$, respectively. The transition $C_4(\alpha, i) \xrightarrow{\tau} C_3(\alpha, i+1)$ is matched by $C'_2(\alpha, i) \xrightarrow{\tau} C'_2(\alpha, i+1)$. Finally both the transitions $C_3(\alpha, i+1) \xrightarrow{\mathtt{app}\,\alpha} C_3(\alpha, i)$ and $C_4(\alpha, i+1) \xrightarrow{\mathtt{app}\,\alpha} C_4(\alpha, i)$ are matched by $C'_2(\alpha, i+1) \xrightarrow{\mathtt{app}\,\alpha} C'_2(\alpha, i)$. All resulting configurations of matching transitions are related in $\mathbb{R}$.

5.2. A Trigger-Installing Ping Service

We now consider a ping service that receives a suspended process on channel *png* and sends back on the same channel a trigger. When the context applies the trigger a copy of the suspended process is run.

$$\begin{aligned}Ping_1 &\stackrel{\text{def}}{=} *(\nu tr.\, P_1(tr))\\ P_1(tr) &\stackrel{\text{def}}{=} png?(X{:}\mathtt{Pr}).png!\langle \lambda tr!.\mathtt{0}\rangle.*(tr?.\mathtt{app}\,X)\end{aligned}$$

We show that this service is weakly bisimilar to the trivial ping service

$$Ping_2 \stackrel{\text{def}}{=} *(png?(X{:}\mathtt{Pr}).png!\langle X\rangle.\mathtt{0})$$

Because ($\simeq$) is a full congruence (Theorem 8.5), it suffices to show that for the processes under the replication

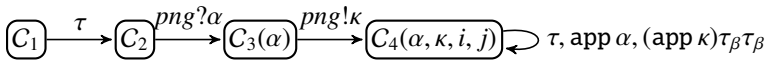
$$M_1 \stackrel{\text{def}}{=} \nu tr.\, P_1(tr) \qquad M_2 \stackrel{\text{def}}{=} png?(X{:}\mathtt{Pr}).png!\langle X\rangle.\mathtt{0}$$

it is the case that $M_1 \simeq M_2$ or, by definition, $\langle\{png\}, M_1\rangle \approx \langle\{png\}, M_2\rangle$. We prove this by providing a relation $\mathbb{R}$ that contains M_1 and M_2 and show that it is a weak bisimulation up-to β and ($\hat{=}$).

We identify the following families of configurations reachable from $\langle\{png\}, M_1\rangle$.

$$\begin{aligned}C_1 &\hat{=} \langle\{png\}, M_1\rangle\\ C_2 &\hat{=} \nu tr\,\langle\{png\},\ P_1(tr)\rangle\\ C_3(\alpha) &\hat{=} \nu tr\,\langle\{png, \alpha\},\ png!\langle\lambda tr!.\mathtt{0}\rangle.*(tr?.\mathtt{app}\,\alpha)\rangle\\ C_4(\alpha, \kappa, i, j) &\hat{=} \nu tr\,\langle\{png, \alpha, \kappa{\mapsto}\lambda tr!.\mathtt{0}\},\ *(tr?.\mathtt{app}\,\alpha)\ \mid\\ &\qquad\qquad \textstyle\prod_i tr?.\mathtt{app}\,\alpha\ \mid\ \prod_k \mathtt{app}\,\alpha\rangle\end{aligned}$$

An abstraction of the LTS for M_1 is the following.



Transition $C_1 \xrightarrow{\tau} C_2$ is the extrusion of the local name tr to the level of the configuration. Transition $C_2 \xrightarrow{png?\alpha} C_3(\alpha)$ is the input on channel png of an abstract constant α and transition $C_3(\alpha) \xrightarrow{png!\kappa} C_4(\alpha, \kappa, 0, 0)$ is the subsequent output of the concrete constant κ on the same channel. The remaining transitions are between configurations in the family $C_4(\alpha, \kappa, i, j)$ that only affect the value of the parameters:

(i) $C_4(\alpha, \kappa, i, j) \xrightarrow{\tau} C_4(\alpha, \kappa, i + 1, j)$ is an unfolding of the replication.

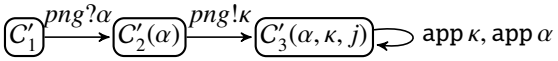
(ii) $C_4(\alpha, \kappa, i + 1, j) \xrightarrow{\mathtt{app}\,\kappa} \xrightarrow{\tau_\beta} \xrightarrow{\tau_\beta} C_4(\alpha, \kappa, i, j + 1)$ is an $\mathtt{app}\,\kappa$ transition, which puts $\mathtt{app}\,\lambda tr!.\mathtt{0}$ in parallel with the current process, followed by two β-moves. The first β-move is the application of $\mathtt{app}\,\lambda tr!.\mathtt{0}$ that produces the process $tr!.\mathtt{0}$, and the second is the communication over the channel tr that releases one more process $\mathtt{app}\,\alpha$.

(iii) $C_4(\alpha, \kappa, i, j) \xrightarrow{\mathtt{app}\,\alpha} C_4(\alpha, \kappa, i, j - 1)$ is an observable application of the abstract constant α.

Similarly we find the families of configurations reachable from $\langle\{png\}, M_2\rangle$,

$$\begin{aligned} C'_1 &\triangleq \langle\{png\}, M_2\rangle \\ C'_2(\alpha) &\triangleq \langle\{png, \alpha\}, png!\langle\alpha\rangle.\mathtt{0}\rangle \\ C'_3(\alpha, \kappa, j) &\triangleq \langle\{png, \alpha, \kappa{\mapsto}\alpha\}, \textstyle\prod_j \mathtt{app}\,\alpha\rangle \end{aligned}$$

and the corresponding abstraction of the LTS



Here we have no τ-transitions, only the input and output on channel png and the transitions $\mathtt{app}\,\kappa$ and $\mathtt{app}\,\alpha$, which increase and decrease, respectively, the number of $\mathtt{app}\,\alpha$ in the process. The following relation is a weak bisimulation up-to β and ($\hat{=}$).

$$\begin{aligned} \mathbb{R} = \{\ &(C_1, C'_1),\ (C_2, C'_1),\ (C_3(\alpha), C'_2(\alpha)), \\ &(C_4(\alpha, \kappa, i, j), C'_3(\alpha, \kappa, j))\ \mid\ \alpha, \kappa, i, j\ \} \end{aligned}$$

The proof is straightforward. All τ-transitions on the LHS are matched by zero transitions on the RHS; the transitions $\mathtt{app}\,\kappa$ and $\mathtt{app}\,\alpha$ on the LHS are matched with the same transitions on the RHS. Conversely, any transition on the RHS is matched with a corresponding weak transition on the LHS. Furthermore, all configurations resulting from matching moves are related by ($\xrightarrow{\tau_\beta}{}^* \hat{=} \mathbb{R} \hat{=}$).

5.3. A Trigger-Promoting Ping Service

We now consider a ping service that, instead of locally installing a trigger service for each suspended process it receives, wraps this trigger service in the response sent to the client. The trigger service is installed in only one of the clients, after an application

of the response. If $\mathcal{V}$ is the suspended process received by the service then the response will be

$$\mathcal{U}(\mathcal{V}, inst, tr) \stackrel{\text{def}}{=} \lambda(tr!.\mathbf{0} \mid inst?.*(tr?.\mathtt{app}\,\mathcal{V}))$$

where *inst* is a private channel controlling the installation of the trigger service and *tr* is a private channel that invokes it. The ping service is encoded as

$$\begin{aligned} Ping_3 &\stackrel{\text{def}}{=} *(\nu inst.\, \nu tr.\, P_3(inst, tr)) \\ P_3(inst, tr) &\stackrel{\text{def}}{=} png?(X{:}\mathtt{Pr}).png!\langle\mathcal{U}(X, inst, tr)\rangle.inst!.\mathbf{0} \end{aligned}$$

We prove that $Ping_3$ is weakly bisimilar to the trivial ping service $Ping_2$, defined in the previous section. As before we will use the property of congruence for $(\simeq)$ to simplify the proof. Hence we only have to prove that

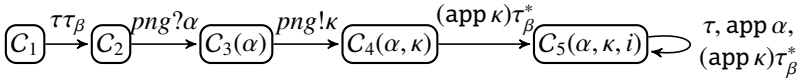
$$M_3 \stackrel{\text{def}}{=} \nu inst.\, \nu tr.\, png?(X{:}\mathtt{Pr}).png!\langle\mathcal{U}(X, inst, tr)\rangle.inst!.\mathbf{0}$$

is weakly bisimilar to M_2, also defined in the previous section. We show this by providing a relation $\mathbb{R}$ that relates the two processes and showing that it is a weak bisimulation up-to β and $(\hat{=})$.

First, we identify the following families of configurations reachable from configuration $\langle\{png\}, M_3\rangle$. Here we omit subscripts to parallel products that do not affect the equivalence between configurations.

$$\begin{aligned} C_1 &\hat{=} \langle\{png\}, M_3\rangle \\ C_2 &\hat{=} \nu inst, tr\, \langle\{png\},\ P_3(inst, tr)\rangle \\ C_3(\alpha) &\hat{=} \nu inst, tr\, \langle\{png, \alpha\},\ png!\langle\mathcal{U}(\alpha, inst, tr)\rangle.inst!.\mathbf{0}\rangle \\ C_4(\alpha, \kappa) &\hat{=} \nu inst, tr\, \langle\{png, \alpha, \kappa{\mapsto}\mathcal{U}(\alpha, inst, tr)\},\ inst!.\mathbf{0}\rangle \\ C_5(\alpha, \kappa, i) &\hat{=} \nu inst, tr\, \langle\{png, \alpha, \kappa{\mapsto}\mathcal{U}(\alpha, inst, tr)\}, \\ &\qquad *(tr?.\mathtt{app}\,\alpha) \mid \textstyle\prod tr?.\mathtt{app}\,\alpha \mid \\ &\qquad \textstyle\prod_i \mathtt{app}\,\alpha \mid \textstyle\prod inst?.*(tr?.\mathtt{app}\,\alpha)\rangle \end{aligned}$$

The corresponding abstraction of the LTS is:



The first $\tau\tau_\beta$ transition extrudes the private names *inst* and *tr*. The transitions $png?\alpha$ and $png!\kappa$ are the input and output of the trigger, respectively. The transition $C_4(\alpha, \kappa) \xrightarrow{\mathtt{app}\,\kappa}\xrightarrow{\tau_\beta}{}^* C_5(\alpha, \kappa, 1)$ is the application of the concrete constant κ by the context, followed by a communication on channel *inst* that will install the service $*(tr?.\mathtt{app}\,\alpha)$, at least one unfolding of the replication, and a communication on channel *tr* that will produce the process $\mathtt{app}\,\alpha$. $C_5(\alpha, \kappa, i)$ has a τ-loop that unfolds the replication, as well as the transitions $C_5(\alpha, \kappa, i) \xrightarrow{\mathtt{app}\,\kappa}\xrightarrow{\tau_\beta}{}^* C_5(\alpha, \kappa, i+1)$ and $C_5(\alpha, \kappa, i+1) \xrightarrow{\mathtt{app}\,\alpha} C_5(\alpha, \kappa, i)$.

The families of configurations and the abstraction of the LTS for M_2 are given in the previous section.

We can easily show that the following relation on configurations[1] is a weak bisimulation up-to β and ($\hat{=}$).

$$\begin{aligned}\mathbb{R} = \{\ & (C_1, C_1'),\ (C_2, C_1'),\ (C_3(\alpha), C_2'(\alpha)),\\ & (C_4(\alpha,\kappa), C_3'(\alpha,\kappa,0)),\ (C_5(\alpha,\kappa,i), C_3'(\alpha,\kappa,i)) \mid \alpha,\kappa,i\ \}\end{aligned}$$

5.4. Composition of Triggers with Replication

In previous examples we used the fact that ($\simeq$) is a congruence to factor out the common contexts and simplify the proofs of equivalence. This is not always possible. To illustrate this we prove the equivalence

$$Ping_3 \simeq Ping_4$$

where $Ping_3$ is the ping service defined in Section 5.3, and

$$\begin{aligned} Ping_4 &\stackrel{\text{def}}{=} rec(M_4) & \qquad M_4 &\stackrel{\text{def}}{=} png?(X{:}\mathtt{Pr}).png!\langle X\rangle.\mathbf{0}\\ rec(P) &\stackrel{\text{def}}{=} \nu a.\,(R \mid a!\langle \lambda P \mid R\rangle.\mathbf{0}) & \qquad R &\stackrel{\text{def}}{=} a?(X).(\mathtt{app}\,X \mid a!\langle X\rangle.\mathbf{0})\end{aligned}$$

Because of the use of different replication constructs, there is no common context between $Ping_3$ and $Ping_4$ that we can factor out to reduce the proof obligation. Hence, we have to provide a relation $\mathbb{R}$ such that $\langle\{png\}, Ping_3\rangle\ \mathbb{R}\ \langle\{png\}, Ping_4\rangle$ and show that it is a weak bisimulation up-to β and ($\hat{=}$).

We devise the following family of configurations that describes the configurations that are reachable from $\langle\{png\}, Ping_3\rangle$ and relevant to the bisimulation proof.

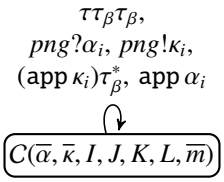
$$\begin{aligned} C(\overline{\alpha},\overline{\kappa},I,J,K,L,\overline{m}) \mathrel{\hat{=}} \nu\overline{tr}\,\langle\ & \{png, \overline{\alpha}, \overline{\kappa\mapsto \mathcal{U}(\alpha,inst,tr)}^{K\cup L}\},\ *(M_3)\\ & \mid \textstyle\prod_{i\in I} P_3(inst_i, tr_i)\\ & \mid \textstyle\prod_{j\in J} png!\langle \mathcal{U}(\alpha_j, inst_j, tr_j)\rangle.inst_j!.\mathbf{0}\\ & \mid \textstyle\prod_{k\in K} inst_k!.\mathbf{0}\\ & \mid \textstyle\prod_{l\in L} (*(tr_l?.\mathtt{app}\,\alpha_l) \mid \prod tr_l?.\mathtt{app}\,\alpha_l\\ & \qquad \mid \textstyle\prod_{m_l} \mathtt{app}\,\alpha_l \mid \prod inst_l?.*(tr_l?.\mathtt{app}\,\alpha_l))\rangle \end{aligned}$$

Here I, J, K, and L are finite sets of natural numbers. We also use the notation $\overline{A}^S$ to mean that the length of the sequence is the cardinality of the set S, and the subscripts of the metavariables in A are drawn from the elements of S.

The reader may observe that C contains the parallel composition of the process $*(M_3)$ with an arbitrary number of the processes in configurations C_2, C_3, C_4, and C_5 of Section 5.3. This is because the ping service may be invoked multiple times by the context, and each invocation will create a separate set of states C_2 to C_5. The sets I, J, K, L contain the indices of local trigger channels, abstract constants, and concrete constants that correspond to the different instances of C_2 to C_5, respectively. Moreover, the transitions that in Section 5.3 are between configurations in C_i and C_j now only change the parameters of C.

The abstraction of the LTS in this case has only one state.

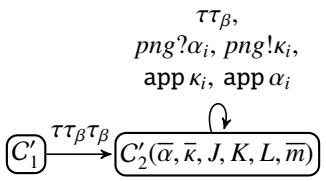
[1] Because of omitted indices in the definition of C_6, the expression $C_5(\overline{n},\alpha,\kappa,i)$ is a set of configurations. Here we abuse notation to mean any configuration in that set.



The configurations reachable from $\langle\{png\}, Ping_4\rangle$ and relevant to this bisimulation proof belong to the following families of configurations.

$$\begin{aligned} C_1' &\triangleq \langle\{png\}, rec(M_4)\rangle \\ C_2'(\overline{\alpha},\overline{\kappa},J,K,L,\overline{m}) &\triangleq \nu a\langle\, \{png,\overline{\alpha},\overline{\kappa\mapsto\alpha}^{K\cup L}\},\ R \mid a!\langle\lambda\, M_4 \mid R\rangle.\mathtt{0} \\ &\qquad \mid \textstyle\prod M_4 \mid \prod_{j\in J} png!\langle\alpha_j\rangle.\mathtt{0} \mid \prod_{l\in L}(\prod_{m_l}\mathtt{app}\,\alpha_l)\rangle \end{aligned}$$

Notice that C_2' is similar to configuration C_3 of Section 5.1. C_4 is not necessary here because of the use of the up-to β technique. These are also the two states of the abstract LTS.



It is straightforward to verify that the following relation is a weak bisimulation up-to β and $(\hat{=})$ and $\langle\{png\}, Ping_3\rangle\ \mathbb{R}\ \langle\{png\}, Ping_4\rangle$.

$$\begin{aligned} \mathbb{R} = \{\ & (C(\cdot,\cdot,\emptyset,\emptyset,\emptyset,\emptyset,\cdot), C_1'),\ (C(\overline{\alpha},\overline{\kappa},I,J,K,L,\overline{m}), C_2'(\overline{\alpha},\overline{\kappa},J,K,L,\overline{m})), \\ & \mid\ \overline{\alpha},\overline{\kappa},\overline{m},\ (\forall k\in K.\ i_k + j_k + l_k = m_k),\ \textit{and } I,J,K,L \text{ are pairwise disjoint}\ \} \end{aligned}$$

5.5. The Processes in Figure 1

For our last two examples we consider the two pairs of processes in Figure 1, discussed in the introduction. We prove that the processes in (†) are indeed weakly bisimilar, while the processes in (‡) are not.

We extend the language with internal choice by adding the following reduction and transition rules (and their symmetric ones).

$$P\oplus Q \to P \qquad \nu\overline{a}\langle\Delta,\ P\oplus Q\rangle \xrightarrow{\tau} \nu\overline{a}\langle\Delta,\ P\rangle$$

Adding these rules does not change our theory since internal choice can be encoded using communication:

$$P\oplus Q \stackrel{\text{def}}{=} \nu a.\ a!.\mathtt{0} \mid a?.P \mid a?.Q \qquad a\notin fn(P,Q)$$

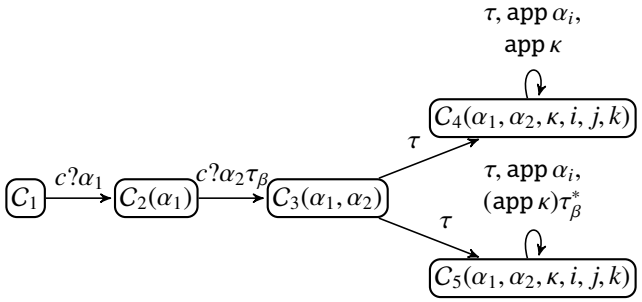
The equivalence.. First we consider the equivalence (†) in Figure 1. We will show that $P \simeq P'$; i.e. we will show $\langle\{c\}, P\rangle \approx \langle\{c\}, P'\rangle$, where

$$\begin{aligned}
P &\stackrel{\text{def}}{=} c?(X).c?(Y).\nu t.\ (c!\langle \mathcal{V}_1(X,Y)\rangle.\mathbb{0} \oplus c!\langle \mathcal{V}_2(X)\rangle.\mathbb{0}) \\
&\qquad\qquad \mid *(t?.(\mathsf{app}\, X \oplus \mathsf{app}\, Y)) \\
P' &\stackrel{\text{def}}{=} c?(X).c?(Y).\nu t.\ (c!\langle \mathcal{V}_1(Y,X)\rangle.\mathbb{0} \oplus c!\langle \mathcal{V}_2(Y)\rangle.\mathbb{0}) \\
&\qquad\qquad \mid *(t?.(\mathsf{app}\, X \oplus \mathsf{app}\, Y)) \\
\mathcal{V}_1(X,Y) &\stackrel{\text{def}}{=} \lambda((\mathsf{app}\, X \oplus \mathsf{app}\, Y) \mid \mathsf{app}\, X) \\
\mathcal{V}_2(X) &\stackrel{\text{def}}{=} \lambda(t!.\mathsf{app}\, Y)
\end{aligned}$$

The relevant configurations reachable from $\langle\{c\}, P\rangle$ can be described by the following families of configurations.

$$\begin{aligned}
C_1 &\triangleq \langle\{c\}, \boldsymbol{P}\rangle \\
C_2(\alpha_1) &\triangleq \langle\{c, \alpha_1\}, c?(Y).\nu t.\ (c!\langle \boldsymbol{\mathcal{V}_1(\alpha_1, Y)}\rangle.\mathbb{0} \oplus c!\langle \boldsymbol{\mathcal{V}_2(\alpha_1)}\rangle.\mathbb{0}) \\
&\qquad\qquad \mid *(t?.(\mathsf{app}\, \alpha_1 \oplus \mathsf{app}\, Y)) \\
&\qquad\qquad \mid \textstyle\prod t?.(\mathsf{app}\, \alpha_1 \oplus \mathsf{app}\, \alpha_2)\rangle \\
C_3(\alpha_1, \alpha_2) &\triangleq \nu t\, \langle\{c, \alpha_1, \alpha_2\},\ (c!\langle \boldsymbol{\mathcal{V}_1(\alpha_1, \alpha_2)}\rangle.\mathbb{0} \oplus c!\langle \boldsymbol{\mathcal{V}_2(\alpha_1)}\rangle.\mathbb{0}) \\
&\qquad\qquad \mid *(t?.(\mathsf{app}\, \alpha_1 \oplus \mathsf{app}\, \alpha_2)) \\
&\qquad\qquad \mid \textstyle\prod t?.(\mathsf{app}\, \alpha_1 \oplus \mathsf{app}\, \alpha_2)\rangle \\
C_4(\alpha_1, \alpha_2, \kappa, i, j, k) & \\
&\triangleq \nu t\, \langle\{c, \alpha_1, \alpha_2, \kappa \mapsto \boldsymbol{\mathcal{V}_1(\alpha_1, \alpha_2)}\}, \\
&\qquad *(t?.(\mathsf{app}\, \alpha_1 \oplus \mathsf{app}\, \alpha_2)) \\
&\qquad \mid \textstyle\prod t?.(\mathsf{app}\, \alpha_1 \oplus \mathsf{app}\, \alpha_2) \\
&\qquad \mid \textstyle\prod_i(\mathsf{app}\, \alpha_1 \oplus \mathsf{app}\, \alpha_2) \mid \prod_j \mathsf{app}\, \alpha_1 \mid \prod_k \mathsf{app}\, \alpha_2\rangle \\
C_5(\alpha_1, \alpha_2, \kappa, i, j, k) & \\
&\triangleq \nu t\, \langle\{c, \alpha_1, \alpha_2, \kappa \mapsto \boldsymbol{\mathcal{V}_2(\alpha_1)}\}, \\
&\qquad *(t?.(\mathsf{app}\, \alpha_1 \oplus \mathsf{app}\, \alpha_2)) \\
&\qquad \mid \textstyle\prod t?.(\mathsf{app}\, \alpha_1 \oplus \mathsf{app}\, \alpha_2) \\
&\qquad \mid \textstyle\prod_i(\mathsf{app}\, \alpha_1 \oplus \mathsf{app}\, \alpha_2) \mid \prod_j \mathsf{app}\, \alpha_1 \mid \prod_k \mathsf{app}\, \alpha_2\rangle
\end{aligned}$$

The corresponding abstraction of the LTS is:



We obtain the families C_1' to C_5' of configurations reachable from $\langle\{c\}, P'\rangle$ by performing the following replacements of the boldfaced parts in C_1 to C_5:

(i) in C_1 we replace P with P',

(ii) in C_2 we replace the $\mathcal{V}_1(\alpha_1, Y)$ and $\mathcal{V}_2(\alpha_1)$ with $\mathcal{V}_1(Y, \alpha_1)$ and $\mathcal{V}_2(Y)$, respectively,

(iii) in C_3 to C_5 we replace $\mathcal{V}_1(\alpha_1, \alpha_2)$ and $\mathcal{V}_2(\alpha_1)$ with $\mathcal{V}_1(\alpha_2, \alpha_1)$ and $\mathcal{V}_2(\alpha_2)$, respectively.

The corresponding abstraction of the LTS is the same as the one shown above.

The intuition of this equivalence is that the τ-transition

$$C_3(\alpha_1, \alpha_2) \xrightarrow{\tau} C_4(\alpha_1, \alpha_2, \kappa, 0, 0, 0)$$

on the left-hand side is matched by the τ-transition

$$C'_3(\alpha_1, \alpha_2) \xrightarrow{\tau} C'_5(\alpha_1, \alpha_2, \kappa, 0, 0, 0)$$

on the right-hand side, and vice-versa. Hence, from that point onward, a transition

$$C_4(\alpha_1, \alpha_2, \kappa, i, j, k) \xrightarrow{\mathsf{app}\,\kappa} C_4(\alpha_1, \alpha_2, \kappa, i+1, j+1, k)$$

is matched by the transitions

$$C_5(\alpha_1, \alpha_2, \kappa, i, j, k) \xrightarrow{\mathsf{app}\,\kappa} \xrightarrow{\tau_\beta} \xrightarrow{\tau_\beta} C_5(\alpha_1, \alpha_2, \kappa, i+1, j+1, k)$$

where the first τ-step unfolds the replication once, and the second is the internal communication on channel t.

The proof of this equivalence concludes by verifying that the following relation is a weak bisimulation up-to β and ($\hat{=}$).

$$\begin{aligned}\mathbb{R} = \{\ & (C_1, C'_1),\ (C_2(\alpha_1), C'_2(\alpha_1)),\\ & (C_3(\alpha_1, \alpha_2), C'_3(\alpha_1, \alpha_2)),\\ & (C_4(\alpha_1, \alpha_2, \kappa, i, j, k), C'_5(\alpha_1, \alpha_2, \kappa, i, j, k)),\\ & (C_5(\alpha_1, \alpha_2, \kappa, i, j, k), C'_4(\alpha_1, \alpha_2, \kappa, i, j, k)),\\ & \mid \alpha_1, \alpha_2, \kappa, i, j, k\ \}\end{aligned}$$

The inequivalence.. We now consider the processes (‡) in Figure 1 and prove they are not equivalent. These processes, written in pp-π syntax, are:

$$\begin{aligned} Q &\stackrel{\text{def}}{=} c?(X).c?(Y).\nu t.\ (\ c!\langle\lambda((\mathsf{app}\,X \mid \mathsf{app}\,Y) \oplus \mathsf{app}\,X)\rangle.\mathbf{0}\\ &\qquad\qquad \oplus c!\langle\lambda(t!.\mathsf{app}\,Y)\rangle.\mathbf{0}\ )\\ &\qquad\qquad \mid *(\ t?.(\mathsf{app}\,X \mid \mathsf{app}\,Y)\ )\\ Q' &\stackrel{\text{def}}{=} c?(X).c?(Y).\nu t.\ (\ c!\langle\lambda((\mathsf{app}\,X \mid \mathsf{app}\,Y) \oplus \mathsf{app}\,Y)\rangle.\mathbf{0}\\ &\qquad\qquad \oplus c!\langle\lambda(t!.\mathsf{app}\,X)\rangle.\mathbf{0}\ )\\ &\qquad\qquad \mid *(\ t?.(\mathsf{app}\,X \mid \mathsf{app}\,Y)\ ) \end{aligned}$$

Let us assume that we match the output

$$c!\langle\lambda((\mathsf{app}\,X \mid \mathsf{app}\,Y) \oplus \mathsf{app}\,X)\rangle.\mathbf{0}$$

on the left-hand side with the output

$$c!\langle\lambda(t!.\mathtt{app}\,X)\rangle.\mathbb{0}$$

on the right-hand side. Then, after an $\mathtt{app}\,\kappa$ transition, we could eventually arrive at related configurations where the one on the left-hand side would be able to apply the value bound to X, *but not* the value bound to Y; the configuration on the right-hand side would always be able to apply both the values bound to X and Y, therefore, these configurations would not be weakly bisimilar (and would be distinguishable by an observer).

Of course there are more choices of relating configurations on the left- and right-hand side that might lead to a bisimulation. The Hennessy-Milner Logic that we used to characterise weak bisimilarity is useful in proving that *none* of these choices would be successful. It suffices to find an HML formula that is satisfied by the configuration $\langle\!\langle\{c\}, Q\rangle\!\rangle$ but not by $\langle\!\langle\{c\}, Q'\rangle\!\rangle$. This formula F is

$$\langle c?\alpha_1\rangle\langle c?\alpha_2\rangle\langle c!\kappa\rangle\langle\mathtt{app}\,\kappa\rangle(\langle\mathtt{app}\,\alpha_1\rangle\,\mathtt{tt}\ \wedge\ [\mathtt{app}\,\alpha_2]\,\mathtt{ff}\,)$$

It is the case that $\langle\!\langle\{c\}, Q\rangle\!\rangle \models F$, because after the inputs $c?\alpha_1$ and $c?\alpha_2$ the process can pick the output

$$c!\langle\lambda((\mathtt{app}\,\alpha_1 \mid \mathtt{app}\,\alpha_2)\oplus\mathtt{app}\,\alpha_1)\rangle.\mathbb{0}$$

to perform an $c!\kappa$ transition. A subsequent $\mathtt{app}\,\kappa$ transition followed by a τ-transition of the internal choice releases the process $\mathtt{app}\,\alpha_1$, which can perform an $\mathtt{app}\,\alpha_1$ transition but not an $\mathtt{app}\,\alpha_2$ transition.

On the other hand, $\langle\!\langle\{c\}, Q'\rangle\!\rangle \not\models F$ because none of the outputs

$$c!\langle\lambda((\mathtt{app}\,\alpha_1 \mid \mathtt{app}\,\alpha_2)\oplus\mathtt{app}\,\alpha_1)\rangle.\mathbb{0} \qquad c!\langle\lambda(t!.\mathtt{app}\,\alpha_1)\rangle.\mathbb{0}$$

leads to a configuration satisfying $\langle\mathtt{app}\,\alpha_1\rangle\mathtt{tt} \wedge [\mathtt{app}\,\alpha_2]\mathtt{ff}$.

6. Soundness of Weak Bisimilarity

In this section we prove that weak bisimulation equivalence ($\simeq$) satisfies the defining properties of parallel contextual equivalence ($\cong_{\text{pcxt}}$) and therefore is included in it. For convenience it is divided into four sub-sections. The first establishes a close relationship between the reduction semantics of Section 2 and the τ-moves in the LTS semantics of Section 3. This is used in the second sub-section which proves that ($\simeq$) is reduction-closed and preserves barbs. The third subsection shows that ($\approx$) is closed under substitutions of abstract constants with values indexed by concrete constants. This result is central in proving the most difficult property, preservation of ($\approx$) by parallel contexts.

6.1. Reductions versus τ-steps

To prove that ($\simeq$) is reduction-closed we first need to show that τ-transitions correspond to reduction steps.

Proposition 6.1. *If* $\nu\overline{a}\,\langle\{\overline{c}\},\, P\rangle \xrightarrow{\tau} \nu\overline{b}\,\langle\{\overline{c}\},\, Q\rangle$ *then* $\nu\overline{a}.\, P \to^* \nu\overline{b}.\, Q$

Proof. By induction on the transition $\nu\overline{a}\,\langle\{\overline{c}\},\, P\rangle \xrightarrow{\tau} \nu\overline{b}\,\langle\{\overline{c}\},\, Q\rangle$. The cases Cond-True-Trans, Cond-False-Trans, Rec-Trans, Nu-Trans, and App-Trans are trivial.

Case Par-L-Trans: we have $\{\overline{a}\} = \{\overline{a}\}\backslash exp(\tau)$ and

$$\langle\Delta_1 \uplus \{\overline{a}\}, \mathcal{P}_1\rangle \xrightarrow{\tau} \nu\overline{b}\,\langle\Delta_2 \uplus \{\overline{a}\},\, \mathcal{P}_2\rangle \qquad \nu\overline{a}\,\langle\Delta_1,\, \mathcal{P}_1 \mid \mathcal{Q}\rangle \xrightarrow{\tau} \nu\overline{b}, \overline{a}\,\langle\Delta_2,\, \mathcal{P}_2 \mid \mathcal{Q}\rangle$$

and want to show that $\nu\overline{a}.\,(P_1 \mid Q) \to^* \nu\overline{b}.\,(P_2 \mid Q)$. By the induction hypothesis $P_1 \to^* \nu\overline{b}.\, P_2$, and, by the reduction rule Par-L-Red in Figure 3, $P_1 \mid Q \to^* (\nu\overline{b}.\, P_2) \mid Q$. By the properties of well-formed configurations and Proposition 3.3, $\overline{b} \notin fn(\mathcal{Q})$. Hence, by Par-Cong-Red and Nu-Red, $\nu\overline{a}.\,(P_1 \mid Q) \to^* \nu\overline{b}, \overline{a}.\,(P_2 \mid Q)$.

Case Comm-Name-Trans: we have

$$\langle\{\overline{c}, \overline{a}\}, P_1\rangle \xrightarrow{c!n} \langle\Delta', P_2\rangle \qquad \langle\{\overline{c}, \overline{a}\}, Q_1\rangle \xrightarrow{c?n} \langle\Delta'', Q_2\rangle$$
$$\nu\overline{a}\,\langle\{\overline{c}\},\, P_1 \mid Q_1\rangle \xrightarrow{\tau} \nu\overline{a}\,\langle\{\overline{c}\},\, P_2 \mid Q_2\rangle$$

and want to show that $\nu\overline{a}.\, P_1 \mid Q_1 \to^* \nu\overline{a}.\, P_2 \mid Q_2$. By Proposition 3.4 (i) and (iii) we get that

$$P_1 = c!\langle n{:}\mathtt{Nm}\rangle.P_{11} \mid P_{12} \qquad Q_1 = c?(x{:}\mathtt{Nm}).Q_{11} \mid Q_{12}$$
$$P_2 = P_{11} \mid P_{12} \qquad Q_2 = Q_{12}\{n/x\} \mid Q_{22}$$

Thus, by Comm-Red, Cong-Red, Par-Red, and Nu-Red, $\nu\overline{a}.\,(P_1 \mid Q_1) \to^* \nu\overline{a}.\,(P_2 \mid Q_2)$.

Similarly for the case Comm-Proc-Trans. The rest of the cases are vacuously true. □

Proposition 6.2. *If* $P \to Q$, *and* $fn(P) \subseteq \{\overline{c}\}$ *then there exist* $\overline{a}$ *and* Q_0 *such that*

$$\langle\{\overline{c}\}, P\rangle \xRightarrow{\tau} \nu\overline{a}\,\langle\{\overline{c}\},\, Q_0\rangle \qquad \nu\overline{a}.\, Q_0 \equiv Q$$

Proof. By induction on $P \to Q$. Cases Comm-Red, App-Red, and Cond-Red are straightforward.

Case Par-Red: we have

$$P_1 \to P_2 \qquad P_1 \mid Q \to P_2 \mid Q$$

and want to show that there exist $\overline{a}$ and Q_0 such that $\langle\{\overline{c}\}, P_1 \mid Q\rangle \xRightarrow{\tau} \nu\overline{a}\,\langle\{\overline{c}\},\, Q_0\rangle$ and $\nu\overline{a}.\, Q_0 \equiv P_2 \mid Q$. By the induction hypothesis there exist $\overline{a}$ and P_{20} such that $\langle\{\overline{c}\}, P_1\rangle \xRightarrow{\tau} \nu\overline{a}\,\langle\{\overline{c}\},\, P_{20}\rangle$ and $\nu\overline{a}.\, P_{20} \equiv P_2$. By Par-L-Trans and well-formedness of configurations

$$\langle\{\overline{c}\}, P_1 \mid Q\rangle \xRightarrow{\tau} \nu\overline{a}\,\langle\{\overline{c}\},\, P_{20} \mid Q\rangle \qquad \nu\overline{a}.\,(P_{20} \mid Q) \equiv (\nu\overline{a}.\, P_{20}) \mid Q \equiv P_2 \mid Q$$

Case Nu-Red: we have

$$P \to Q \qquad \nu b.\, P \to \nu b.\, Q$$

and want to show that there exist $\overline{a}$ and Q_0 such that $\langle \{\overline{c}\}, \nu b.\, P\rangle \stackrel{\tau}{\Longrightarrow} \nu \overline{a}\, \langle \{\overline{c}\},\, Q_0\rangle$ and $\nu \overline{a}.\, Q_0 \equiv \nu b.\, Q$. By the induction hypothesis there exist $\overline{a}$ and Q_1 such that $\langle \{\overline{c}, b\}, P\rangle \stackrel{\tau}{\Longrightarrow} \nu \overline{a}\, \langle \{\overline{c}, b\},\, Q_1\rangle$ and $\nu \overline{a}.\, Q_1 \equiv Q$. By Nu-Trans and Proposition 3.7 (Hiding) we have

$$\langle \{\overline{c}\}, \nu b.\, P\rangle \stackrel{\tau}{\longrightarrow} \nu b\, \langle \{\overline{c}\},\, P\rangle \stackrel{\tau}{\Longrightarrow} \nu \overline{a}, b\, \langle \{\overline{c}\},\, Q_1\rangle$$
$$\nu \overline{a}.\, \nu b.\, Q_1 \equiv \nu b.\, \nu \overline{a}.\, Q_1 \equiv \nu b.\, Q$$

Case Cong-Red: we have

$$P' \to Q' \qquad P \equiv P' \qquad Q \equiv Q' \qquad P \to Q$$

and want to show that there exist $\overline{a}$ and Q_0 such that $\langle \{\overline{c}\}, P\rangle \stackrel{\tau}{\Longrightarrow} \nu \overline{a}\, \langle \{\overline{c}\},\, Q_0\rangle$ and $\nu \overline{a}.\, Q_0 \equiv Q$. By the induction hypothesis there exist $\overline{a'}$ and Q'_0 such that $\langle \{\overline{c}\}, P'\rangle \stackrel{\tau}{\Longrightarrow} \nu \overline{a'}\, \langle \{\overline{c}\},\, Q'_0\rangle$ and $\nu \overline{a'}.\, Q'_0 \equiv Q'$. By Proposition 4.11 and because $\langle \{\overline{c}\}, P\rangle \equiv \langle \{\overline{c}\}, P'\rangle$ there exist $\overline{a}$ and Q_0 such that

$$\langle \{\overline{c}\}, P\rangle \stackrel{\tau}{\Longrightarrow} \nu \overline{a}\, \langle \{\overline{c}\},\, Q_0\rangle \qquad \nu \overline{a}\, \langle \{\overline{c}\},\, Q_0\rangle \equiv \nu \overline{a'}\, \langle \{\overline{c}\},\, Q'_0\rangle$$

and by Definition 2.1 $\nu \overline{a}.\, Q_0 \equiv \nu \overline{a'}.\, Q'_0 \equiv Q' \equiv Q$. □

6.2. Reduction-closure and preservation of barbs

We can now prove that ($\simeq$) is reduction-closed and preserves barbs.

Proposition 6.3 (Reduction Closure of ($\simeq$))**.** *If $P \simeq P'$ and $P \to Q$ then there exists Q' such that:*

$$P' \to^* Q' \qquad Q \simeq Q'$$

and vice-versa.

Proof. We prove only the forward direction, the converse is symmetric. By the first premise and Definition 4.9, there exist $\overline{b}$ (with $fn(P, P') \subseteq \{\overline{b}\}$) such that

$$\langle \{\overline{b}\}, P\rangle \approx \langle \{\overline{b}\}, P'\rangle \tag{1}$$

By the second premise and Proposition 6.2 there exist $\overline{a}$ and Q_0 such that

$$\langle \{\overline{b}\}, P\rangle \stackrel{\tau}{\Longrightarrow} \nu \overline{a}\, \langle \{\overline{b}\},\, Q_0\rangle \qquad \nu \overline{a}.\, Q_0 \equiv Q$$

Thus, by Definition 4.4 and (1), there exist $\overline{a'}$, Δ', and Q'_0 such that

$$\langle \{\overline{b}\}, P'\rangle \stackrel{\tau}{\Longrightarrow} \nu \overline{a'}\, \langle \Delta',\, Q'_0\rangle \tag{2}$$
$$\nu \overline{a}\, \langle \{\overline{b}\},\, Q_0\rangle \approx \nu \overline{a'}\, \langle \Delta',\, Q'_0\rangle \tag{3}$$

and by Proposition 3.4 (vii) $\Delta' = \{\overline{b}\}$.

By (2) and Proposition 6.1, $P' \to^* \nu\overline{a'}.\, Q'_0$.

By (3) and Lemma 4.16 $\langle\{\overline{b}\}, \nu\overline{a}.\, Q_0\rangle \approx \langle\{\overline{b}\}, \nu\overline{a'}.\, Q'_0\rangle$ and therefore $\nu\overline{a}.\, Q_0 \simeq \nu\overline{a'}.\, Q'_0$. Hence $Q \equiv\simeq \nu\overline{a'}.\, Q'_0$, and by transitivity of $(\simeq)$ and Corollary 4.12 we get $Q \simeq \nu\overline{a'}.\, Q'_0$. □

Proposition 6.4 (Preservation of Barbs of $(\simeq)$)**.** *If $P \simeq P'$ then $P \Downarrow_n$ iff $P' \Downarrow_n$.*

Proof. We prove only the forward direction, the converse is symmetric. By the second premise and Definition 2.6 we get that there exists Q such that $P \to^* Q$, $Q \equiv \nu\overline{a}.\, n!\langle V{:}t\rangle.\, Q_1 \mid Q_2$, and $n \notin \{\overline{a}\}$. By Proposition 6.3 there exists Q' such that $P' \to^* Q'$ and $Q \simeq Q'$.

By the first premise, transitivity of $(\simeq)$, and Corollary 4.12, $\nu\overline{a}.\, n!\langle V{:}t\rangle.\, Q_1 \mid Q_2 \simeq Q'$. Thus, by Definition 4.9, there exist $\overline{b}$ such that

$$\langle\{\overline{b}, n\}, \nu\overline{a}.\, n!\langle V{:}t\rangle.\, Q_1 \mid Q_2\rangle \approx \langle\{\overline{b}, n\}, Q'\rangle \tag{4}$$

and by the transition rules of the LTS we get

$$\langle\{\overline{b}, n\}, \nu\overline{a}.\, n!\langle V{:}t\rangle.\, Q_1 \mid Q_2\rangle \overset{n!V}{\Longrightarrow} \nu\overline{a}\,\langle\{\overline{b}, n\} \cup \{V\},\ Q_1 \mid Q_2\rangle$$

if $t = \mathtt{Nm}$ or

$$\langle\{\overline{b}, n\}, \nu\overline{a}.\, n!\langle V{:}t\rangle.\, Q_1 \mid Q_2\rangle \overset{n!\kappa}{\Longrightarrow} \nu\overline{a}\,\langle\{\overline{b}, n, \kappa{\mapsto}V\},\ Q_1 \mid Q_2\rangle$$

if $t = \mathtt{Pr}$. By Definition 4.4 and (4), there exist $\overline{a'}$, Δ', and Q'_1 such that one of the following is true:

$$\langle\{\overline{b}, n\}, Q'\rangle \overset{n!V}{\Longrightarrow} \nu\overline{a'}\,\langle\Delta',\, Q'_1\rangle$$

or

$$\langle\{\overline{b}, n\}, Q'\rangle \overset{n!\kappa}{\Longrightarrow} \nu\overline{a'}\,\langle\Delta',\, Q'_1\rangle$$

Therefore, by Proposition 3.4 (i) or (ii), and (vii), there exist Q'_1 and Q'_2 such that

$$\langle\{\overline{b}, n\}, Q'\rangle \overset{\tau}{\Longrightarrow} \nu\overline{a'}\,\langle\{\overline{b}, n\},\ n!\langle V'{:}t\rangle.\, Q'_1 \mid Q'_2\rangle$$

and by Proposition 6.1 $Q' \to^* \nu\overline{a'}.\, n!\langle V'{:}t\rangle.\, Q'_1 \mid Q'_2$ with $n \notin \{\overline{a'}\}$. Hence $P' \Downarrow_n$. □

6.3. Substitution of Abstract Constants

To prove preservation of bisimilarity by parallel contexts we need to relate the transitions of two configurations C_1 and C_2 with the transitions of their parallel composition. The main challenge is higher-order communication between C_1 and C_2. If

$$C_1 \xrightarrow{a?\alpha} C'_1 \quad \text{and} \quad C_2 \xrightarrow{a!\kappa} C'_2$$

then we would like to show that the composition of C_1 and C_2 takes a τ-transition to the composition of C'_1 and C'_2. However this is not the case: the composition of C'_1

and C_2' contains the abstract constant α in the place of the actual value communicated, which is indexed by κ in the knowledge environment. Hence we need to relate configurations containing such an "indirection" of values through the environment with the configurations that do not have the indirection.

We define the indexed relation ($\succeq_{\alpha\mapsto\kappa}$), which relates any configuration with an environment containing κ and α, to the configuration we get by substituting α with the value pointed to by κ in the environment.

Definition 6.5. *(Substitution Relation)*

$$\nu\overline{a}\,\langle\Delta \uplus \{\kappa\mapsto\mathcal{V}\} \uplus \{\alpha\},\, \mathcal{P}\rangle \succeq_{\alpha\mapsto\kappa} \nu\overline{a}\,\langle\Delta\{\mathcal{V}/\alpha\},\, \mathcal{P}\{\mathcal{V}/\alpha\}\rangle \quad \textit{if } \alpha \notin \textit{acon}(\mathcal{V})$$

The substitution relation commutes with limited structural equivalence.

Lemma 6.6. *If $C_1 \mathrel{\hat{=}} C_2$ then*

(i) for all C_1', if $C_1 \succeq_{\alpha\mapsto\kappa} C_1'$ then there exists C_2' such that $C_2 \succeq_{\alpha\mapsto\kappa} C_2' \mathrel{\hat{=}} C_1'$,
(ii) for all C_1', if $C_1' \succeq_{\alpha\mapsto\kappa} C_1$ then there exists C_2' such that $C_1' \mathrel{\hat{=}} C_2' \succeq_{\alpha\mapsto\kappa} C_2$,

Proof. We strengthen the lemma by proving (i) and (ii) and their symmetric (obtained by exchanging the subscripts 1 and 2). We proceed by induction on $C_1 \mathrel{\hat{=}} C_2$. The case for reflexivity is immediate; the cases for symmetry and transitivity follow by the induction hypothesis. The remaining cases reorder the processes running in parallel in the configuration; these follow by distribution of substitution over parallel. □

Corollary 6.7. $C_1 \mathrel{\hat{=}} \succeq_{\alpha\mapsto\kappa} C_2$ *iff* $C_1 \succeq_{\alpha\mapsto\kappa} \mathrel{\hat{=}} C_2$.

The following three propositions show that the substitution relation ($\succeq_{\alpha\mapsto\kappa}$) preserves transitions up to pairs of actions $\mathtt{app}\,\alpha, \mathtt{app}\,\kappa$.

Proposition 6.8. *If $C_1 \succeq_{\alpha\mapsto\kappa} C_2 \xrightarrow{\eta} C_2'$ and*

$$\eta \in \{c?n, c!n, c?\alpha', c!\kappa', \mathtt{app}\,\kappa'' \mid \textit{ any } c, n, \alpha', \kappa', \kappa'' \textit{ s.t. } \alpha' \neq \alpha, \kappa' \neq \kappa\}$$

then there exists C_1' such that $C_1 \xrightarrow{\eta} C_1' \succeq_{\alpha\mapsto\kappa} C_2'$, and if $\eta = \mathtt{app}\,\kappa''$ then $\kappa'' \neq \kappa$.

Proof. By induction on the transition $C_2 \xrightarrow{\eta} C_2'$. □

Proposition 6.9. *If $C_1 \succeq_{\alpha\mapsto\kappa} C_2 \xrightarrow{\mathtt{app}\,\alpha'} C_2'$ then $\alpha \neq \alpha'$ and there exists C_1' such that one of the following is true:*

(i) $C_1 \xrightarrow{\mathtt{app}\,\alpha'} C_1' \succeq_{\alpha\mapsto\kappa} C_2'$, *or*
(ii) $C_1 \xrightarrow{\mathtt{app}\,\alpha}\xrightarrow{\mathtt{app}\,\kappa}\xrightarrow{\mathtt{app}\,\alpha'} C_1' \succeq_{\alpha\mapsto\kappa} C_2'$.

Proof. Since α has been substituted in C_1, it is necessary that $\alpha \neq \alpha'$.

The rest follows by induction on the transition $C_2 \xrightarrow{\eta} C_2'$, where $\eta = \mathtt{app}\,\alpha'$. The only cases we need to consider are Par-L-Trans (and its symmetric) and Abs-App-Trans. The former follows by the induction hypothesis; for the latter we have to consider two cases:

(i) $C_1 = \nu\overline{a}\,\langle \Delta \uplus \{\kappa \mapsto \mathcal{V}\} \uplus \{\alpha\},\ \mathtt{app}\,\alpha' \rangle$ and $C_2 = \nu\overline{a}\,\langle \Delta\{\mathcal{V}/\alpha\},\ \mathtt{app}\,\alpha'\{\mathcal{V}/\alpha\} \rangle$ and $C_2' = \nu\overline{a}\,\langle \Delta\{\mathcal{V}/\alpha\},\ \mathtt{0} \rangle$. Then

$$C_1 \xrightarrow{\mathtt{app}\,\alpha'} \nu\overline{a}\,\langle \Delta \uplus \{\kappa \mapsto \mathcal{V}\} \uplus \{\alpha\},\ \mathtt{0} \rangle \geq_{\alpha \mapsto \kappa} C_2'$$

(ii) $C_1 = \nu\overline{a}\,\langle \Delta \uplus \{\kappa \mapsto \alpha'\} \uplus \{\alpha\},\ \mathtt{app}\,\alpha \rangle$ and $C_2 = \nu\overline{a}\,\langle \Delta\{\alpha'/\alpha\},\ \mathtt{app}\,\alpha\{\alpha'/\alpha\} \rangle$ and $C_2' = \nu\overline{a}\,\langle \Delta\{\alpha'/\alpha\},\ \mathtt{0} \rangle$. Then

$$\begin{aligned} C_1 &\xrightarrow{\mathtt{app}\,\alpha} \nu\overline{a}\,\langle \Delta \uplus \{\kappa \mapsto \alpha'\} \uplus \{\alpha\},\ \mathtt{0} \rangle \\ &\xrightarrow{\mathtt{app}\,\kappa} \nu\overline{a}\,\langle \Delta \uplus \{\kappa \mapsto \alpha'\} \uplus \{\alpha\},\ \mathtt{app}\,\alpha' \rangle \\ &\xrightarrow{\mathtt{app}\,\alpha'} \nu\overline{a}\,\langle \Delta \uplus \{\kappa \mapsto \alpha'\} \uplus \{\alpha\},\ \mathtt{0} \rangle \geq_{\alpha \mapsto \kappa} C_2' \end{aligned}$$ □

Proposition 6.10. *If $C_1 \geq_{\alpha \mapsto \kappa} C_2 \xrightarrow{\tau} C_2'$ then there exists C_1' such that one of the following is true:*

(i) $C_1 \xrightarrow{\tau} C_1' \geq_{\alpha \mapsto \kappa} C_2'$, *or*

(ii) $C_1 \xrightarrow{\mathtt{app}\,\alpha\ \mathtt{app}\,\kappa} \xrightarrow{\tau} C_1' \geq_{\alpha \mapsto \kappa} C_2'$.

Proof. By induction on the transition $C_2 \xrightarrow{\eta} C_2'$, where $\eta = \tau$. Cases Comm-Name-Trans and Comm-Proc-Trans follow by Proposition 6.8; cases Cond-True-Trans, Cond-False-Trans, Rec-Trans, and Nu-Trans follow easily by the distribution of substitution over language constructors; case Par-L-Trans (and its symmetric) follow by the induction hypothesis. For case App-Trans we need to consider two cases:

(i) $C_1 = \nu\overline{a}\,\langle \Delta \uplus \{\kappa \mapsto \mathcal{V}\} \uplus \{\alpha\},\ \mathtt{app}\,\lambda\mathcal{P} \rangle$ and $C_2 = \nu\overline{a}\,\langle \Delta\{\mathcal{V}/\alpha\},\ \mathtt{app}\,\lambda\mathcal{P}\{\mathcal{V}/\alpha\} \rangle$ and $C_2' = \nu\overline{a}\,\langle \Delta\{\mathcal{V}/\alpha\},\ \mathcal{P}\{\mathcal{V}/\alpha\} \rangle$. Then

$$C_1 \xrightarrow{\tau} \nu\overline{a}\,\langle \Delta \uplus \{\kappa \mapsto \mathcal{V}\} \uplus \{\alpha\},\ \mathcal{P} \rangle \geq_{\alpha \mapsto \kappa} C_2'$$

(ii) $C_1 = \nu\overline{a}\,\langle \Delta \uplus \{\kappa \mapsto \lambda\mathcal{P}\} \uplus \{\alpha\},\ \mathtt{app}\,\alpha \rangle$ and $C_2 = \nu\overline{a}\,\langle \Delta\{\lambda\mathcal{P}/\alpha\},\ \mathtt{app}\,\alpha\{\lambda\mathcal{P}/\alpha\} \rangle$ and $C_2' = \nu\overline{a}\,\langle \Delta\{\lambda\mathcal{P}/\alpha\},\ \mathcal{P} \rangle = \nu\overline{a}\,\langle \Delta\{\lambda\mathcal{P}/\alpha\},\ \mathcal{P}\{\lambda\mathcal{P}/\alpha\} \rangle$ (because by definition of $(\leq_{\alpha \mapsto \kappa})$, $\alpha \notin \mathit{acon}(\mathcal{P})$). Then

$$\begin{aligned} C_1' &\xrightarrow{\mathtt{app}\,\alpha} \nu\overline{a}\,\langle \Delta \uplus \{\kappa \mapsto \lambda\mathcal{P}\} \uplus \{\alpha\},\ \mathtt{0} \rangle \\ &\xrightarrow{\mathtt{app}\,\kappa} \nu\overline{a}\,\langle \Delta \uplus \{\kappa \mapsto \lambda\mathcal{P}\} \uplus \{\alpha\},\ \mathtt{app}\,\lambda\mathcal{P} \rangle \\ &\xrightarrow{\tau} \nu\overline{a}\,\langle \Delta \uplus \{\kappa \mapsto \lambda\mathcal{P}\} \uplus \{\alpha\},\ \mathcal{P} \rangle \geq_{\alpha \mapsto \kappa} C_2 \end{aligned}$$ □

The following proposition shows that the inverse substitution relation $(\leq_{\alpha \mapsto \kappa})$ preserves transitions, but converts pairs of $\mathtt{app}\,\alpha, \mathtt{app}\,\kappa$ to τ-transitions.

Proposition 6.11. *Let $C_1 \leq_{\alpha \mapsto \kappa} C_2$.*

(i) If $C_2 \xrightarrow{\eta} C_2'$, $\eta \neq \mathtt{app}\,\alpha$, and $\eta \neq \mathtt{app}\,\kappa$ then there exists C_1' such that $C_1 \xrightarrow{\eta} C_1' \leq_{\alpha \mapsto \kappa} C_2'$.

(ii) *If* $C_2 \xrightarrow{\mathsf{app}\,\alpha\,\mathsf{app}\,\kappa} C_2'$ *then* $C_1 \preceq_{\alpha\mapsto\kappa} \hat{=} C_2'$

(iii) *If* $C_2 \overset{\eta}{\Longrightarrow} C_2'$, $\eta \neq \mathsf{app}\,\alpha$, *and* $\eta \neq \mathsf{app}\,\kappa$ *then there exists* C_1' *such that* $C_1 \overset{\eta}{\Longrightarrow} C_1' \preceq_{\alpha\mapsto\kappa} C_2'$.

(iv) *If* $C_2 \overset{\mathsf{app}\,\alpha\,\mathsf{app}\,\kappa}{\Longrightarrow\Longrightarrow} C_2'$ *then there exists* C_1' *such that then* $C_1 \overset{\tau}{\Longrightarrow} C_1' \preceq_{\alpha\mapsto\kappa} \hat{=} C_2'$

Proof. The first property can be shown by induction on the transition $C_2 \xrightarrow{\eta} C_2'$. The second property follows by Propositions 3.4 (v) and (vi). The third is a consequence of (i) and the fourth a consequence of (i), (ii), Corollary 3.5 and Proposition 4.3. □

The preceding propositions in this section show that the substitution relation ($\succeq_{\alpha\mapsto\kappa}$) preserves transitions, up to pairs of $\mathsf{app}\,\alpha$ and $\mathsf{app}\,\kappa$ transitions.

We extend this result to an arbitrary number of substitutions. To do that we use δ to range over ordered *substitution environments* that map abstract to concrete constants and are bijections; i.e. for all $\alpha, \alpha' \in dom(\delta)$, $\alpha = \alpha'$ iff $\delta(\alpha) = \delta(\alpha')$. We then define the relation ($\succeq_\delta$) over configurations:

$$\frac{}{C \succeq_\epsilon C}\ \textsc{Subst-}\epsilon \qquad \frac{C \succeq_\delta C'}{C \succeq_{\alpha\mapsto\kappa,\delta} C'}\ \textsc{Subst-id} \qquad \frac{C \succeq_{\alpha\mapsto\kappa} C'' \qquad C'' \succeq_\delta C'}{C \succeq_{\alpha\mapsto\kappa,\delta} C'}\ \textsc{Subst-}\alpha\kappa$$

We also extend this relation to relate traces of the form

$$(\mathsf{app}\,\alpha_1, \mathsf{app}\,\kappa_1, \ldots \mathsf{app}\,\alpha_n, \mathsf{app}\,\kappa_n, \eta)$$

with the single action η:

$$\frac{}{\eta \succeq_\epsilon \eta}\ \textsc{SubstTr-}\epsilon\eta \qquad \frac{t \succeq_\delta \eta \qquad \alpha \notin acon(\eta), \kappa \notin ccon(\eta)}{t \succeq_{\alpha\mapsto\kappa,\delta} \eta}\ \textsc{SubstTr-id}$$

$$\frac{t \succeq_\delta \eta \qquad \alpha \notin acon(\eta), \kappa \notin ccon(\eta)}{\mathsf{app}\,\alpha, \mathsf{app}\,\kappa, t \succeq_{\alpha\mapsto\kappa,\delta} \eta}\ \textsc{SubstTr-}\alpha\kappa$$

Proposition 6.12. *If* $C_1 \succeq_\delta C_2 \xrightarrow{\eta} C_2'$ *and* $dom(\delta) \cap acon(\eta) = codom(\delta) \cap ccon(\eta) = \emptyset$ *then there exists* C_1', t *such that* $t \succeq_\delta \eta$ *and* $C_1 \xrightarrow{t} C_1' \succeq_\delta C_2'$.

Proof. By induction on $C_1 \succeq_\delta C_2$, using Corollary 6.7 and Propositions 6.8 – 6.10. □

Proposition 6.13. *If* $C_1 \preceq_\delta C_2 \xrightarrow{t} C_2'$ *and* $t \succeq_\delta \eta$ *then there exists* C_1' *such that* $C_1 \xrightarrow{\eta} C_1' \hat{=} \preceq_\delta C_2'$.

Proof. By induction on $C_1 \preceq_\delta C_2$. Case Subst-ϵ is trivial and Subst-id follows by the induction hypothesis. For Subst-$\alpha\kappa$ we proceed by cases on $t \succeq_{\alpha\mapsto\kappa,\delta} \eta$: case SubstTr-$\epsilon\eta$ is vacuously true; case SubstTr-id follows by Proposition 6.11 (i) and the induction hypothesis; case SubstTr-$\alpha\kappa$ follows by Proposition 6.11 (i) and (ii), Corollary 6.7, and the induction hypothesis. □

To handle weak transitions of weak bisimilarity we need to state the last proposition for weak traces. The proof is similar to the one above, using Proposition 6.11 (iii) and (iv).

Proposition 6.14.

(i) If $C_1 \preceq_\delta C_2 \stackrel{t}{\Longrightarrow} C_2'$ *and* $t \succeq_\delta \eta \neq \tau$ *then there exists* C_1' *such that*

$$C_1 \stackrel{\eta}{\Longrightarrow} C_1' \mathrel{\hat{=}} \preceq_\delta C_2'$$

(ii) If $C_1 \preceq_\delta C_2 \stackrel{t}{\Longrightarrow} C_2'$ *and* $(t, \tau) \succeq_\delta \tau$ *then there exists* C_1' *such that*

$$C_1 \stackrel{\tau}{\Longrightarrow} C_1' \mathrel{\hat{=}} \preceq_\delta C_2'$$

Although we will not use this result directly, it is easy to show that weak bisimilarity is preserved by substitution of abstract constants with values indexed by concrete constants.

Proposition 6.15. $(\preceq_\delta \approx \succeq_\delta) \subseteq (\approx)$.

Proof. We have that $(\preceq_\delta \approx \succeq_\delta) \subseteq (\hat{=} \preceq_\delta \approx \succeq_\delta \hat{=})$ and show that $(\hat{=} \preceq_\delta \approx \succeq_\delta \hat{=})$ satisfies the conditions for a weak bisimulation using Propositions 4.3, 6.12, and 6.14, and the fact that $(\approx)$ is a weak bisimulation. □

6.4. Parallel Contexts

Here our intention is to show that

$$P \simeq P' \;\; \textit{implies} \;\; P \mid Q \simeq P' \mid Q \qquad (*)$$

for any Q. That is $(\simeq)$ satisfies property (iii) of Definition 2.7. However, the proof requires a generalisation of (*) to arbitrary configurations.

We define the partial function $\cdot \parallel^{\overline{c}} \cdot$ that composes two well-formed *component configurations* into one well-formed *composite configuration*. The names $\overline{c}$ are names shared in the knowledge environments of the component configurations but are locally bound in the composite configuration.

Definition 6.16 (Configuration Composition)**.** *If the configurations* $\nu\overline{a}\,\langle \Delta_1 \uplus \{\overline{c}\}, \mathcal{P}\rangle$, $\nu\overline{b}\,\langle \Delta_2 \uplus \{\overline{c}\}, \mathcal{Q}\rangle$ *and* $\nu\overline{a}, \overline{b}, \overline{c}\,\langle \Delta_1 \cup \Delta_2, \mathcal{P} \mid \mathcal{Q}\rangle$ *are well-formed then*

$$\nu\overline{a}\,\langle \Delta_1 \uplus \{\overline{c}\}, \mathcal{P}\rangle \parallel^{\overline{c}} \nu\overline{b}\,\langle \Delta_2 \uplus \{\overline{c}\}, \mathcal{Q}\rangle \stackrel{\text{def}}{=} \nu\overline{a}, \overline{b}, \overline{c}\,\langle \Delta_1 \cup \Delta_2, \mathcal{P} \mid \mathcal{Q}\rangle$$

The vector $\overline{c}$ records the names that were sent via first-order communication from one component configuration to the other but are not in the knowledge of the observer of the composite configuration.

Parallel composition of configurations is preserved by $(\approx)$, provided the bisimilar configurations have the same names in their knowledge environment.

Proposition 6.17. *Let $C_1 \parallel^{\overline{c}} C_2$ be defined and for $i = 1, 2$, $C_i \approx C_i'$ and $names(C_i) = names(C_i')$. Then $C_1' \parallel^{\overline{c}} C_2'$ is defined.*

Proof. By Definition 6.16 and because $names(C_i) = names(C_i')$, for $i = 1, 2$, the names $\overline{c}$ are in knowledge environments of C_1' and C_2'. It remains to show that $C_1' \parallel^{\overline{c}} C_2'$ is a well-formed configuration. The union of the knowledge environments of C_1' and C_2' is a valid knowledge environment because by Lemma 4.6 $ccon(C_i) = ccon(C_i')$, for $i = 1, 2$, and the union of the knowledge environments of C_1 and C_2 is a valid knowledge environment. The rest of the conditions for $C_1' \parallel^{\overline{c}} C_2'$ being a well-formed configuration are easily established. □

We will compose two configurations in parallel and then replace abstract variables in one with values indexed by concrete variables in the other through the relation $(\geq_\delta)$, for a *compatible substitution environment* δ.

Definition 6.18. *(Compatible Substitution Environment) A substitution environment δ is compatible to a configuration composition $\nu\overline{a}\,\langle\Delta_1, \mathcal{P}\rangle \parallel^{\overline{c}} \nu\overline{b}\,\langle\Delta_2, Q\rangle$ when $\delta(\alpha) = \kappa$ implies either*

(i) $\alpha \in \Delta_1$, $\alpha \notin \Delta_2$, $\kappa \in dom(\Delta_2)$, $\kappa \notin dom(\Delta_1)$, *or*

(ii) $\alpha \in \Delta_2$, $\alpha \notin \Delta_1$, $\kappa \in dom(\Delta_1)$, $\kappa \notin dom(\Delta_2)$.

The substitution environment records the higher-order communications between the component configurations, hence the α and κ in each binding of δ will be part of different component configurations. As the composite configuration progresses with τ-transitions due to higher-order communication, the necessary substitution environment may grow. The properties of substitution from the previous section will help us to map these transitions to transitions of the component configurations and back.

We now define the closure operator $(\cdot)^{\mathsf{par}}$ of a relation under parallel contexts and substitution of abstract variables. We aim to show that $(\approx)^{\mathsf{par}}$ is contained in $(\approx)$, from which (*) will follow.

Definition 6.19 (Parallel Context Closure of a Relation)**.** *If $\mathbb{R}$ is a relation on well-formed configurations of the LTS then $\mathbb{R}^{\mathsf{par}}$ is the relation on well-formed configurations defined by:*

$$\begin{aligned}\mathbb{R}^{\mathsf{par}} \stackrel{\text{def}}{=} \{(C, C') \mid \exists C_1, C_2, C_1', C_2', \delta, \overline{c}.\ \ & C_1\ \mathbb{R}\ C_1',\ \ C_2\ \mathbb{R}\ C_2' \\ & C_1 \parallel^{\overline{c}} C_2 \geq_\delta C,\ \ C_1' \parallel^{\overline{c}} C_2' \geq_\delta C' \\ & acon(C_1) = acon(C_1'),\ \ names(C_1) = names(C_1') \\ & acon(C_2) = acon(C_2'),\ \ names(C_2) = names(C_2') \\ & \delta \text{ is compatible to } C_1 \parallel^{\overline{c}} C_2\}\end{aligned}$$

The important part of this definition is closure under parallel composition. As we discussed earlier, the closure under the substitution relation $(\geq_\delta)$ is necessary to deal with higher-order communication between the component configurations of the composition.

Before showing that $(\approx)^{\mathsf{par}} \subseteq (\approx)$, we reason about the transitions of parallel composition. We first show that observable transitions of the composite configuration are the result of the interleaving of transitions of the component configurations.

Proposition 6.20. *If $C_1 \parallel^{\overline{c}} C_2 \xrightarrow{\eta} C_3$, with $\eta \neq \tau$, then for $\{\overline{d}\} = \{\overline{c}\} \setminus exp(\eta)$, either*

(i) there exists C_1' such that $C_3 = C_1' \parallel^{\overline{d}} C_2$ and $C_1 \xrightarrow{\eta} C_1'$ and $C_1' \parallel^{\overline{d}} C_2'$, or

(ii) there exists C_2' such that $C_3 = C_1 \parallel^{\overline{d}} C_2'$ and $C_2 \xrightarrow{\eta} C_2'$ and $C_1' \parallel^{\overline{d}} C_2'$.

Proof. We start by observing that $C_1 \parallel^{\overline{c}} C_2 \xrightarrow{\eta} C_3$ can only be derived by rule Par-L-Trans of Figure 5 or its symmetric. We then take cases on η and complete the proof using Propositions 3.4, 3.6 (ii), 3.7 (i), and 3.8. □

Internal transitions of the composite configuration may be the result of interleaving of internal transitions of, or communication between, the components.

Proposition 6.21. *If $C_1 \parallel^{\overline{c}} C_2 \xrightarrow{\tau} C_3$ then either*

(i) there exists C_1' such that $C_3 = C_1' \parallel^{\overline{c}} C_2$ and $C_1 \xrightarrow{\tau} C_1'$,

(ii) there exist C_1', C_2', and $\overline{d}$ such that $C_3 = C_1' \parallel^{\overline{d}} C_2'$ and $C_1 \xrightarrow{a!b} C_1'$ and $C_2 \xrightarrow{a?b} C_2'$ and if $C_1 = \nu\overline{b}, b\,\langle\Delta, \mathcal{P}\rangle$ then $\{\overline{d}\} = \{\overline{c}\} \cup \{b\}$ else $\overline{d} = \overline{c}$,

(iii) there exist $\alpha, \kappa \in fresh(C_1, C_2)$, C_1', and C_2' such that $C_1' \parallel^{\overline{c}} C_2' \geq_{\alpha\mapsto\kappa} C_3$ and $C_1 \xrightarrow{a!\kappa} C_1'$ and $C_2 \xrightarrow{a?\alpha} C_2'$,

or the symmetric of one of the above conditions.

Proof. By cases on $C_1 \parallel^{\overline{c}} C_2 \xrightarrow{\tau} C_3$ using Propositions 3.4, 3.6 (ii), 3.7 (i), and 3.8. For case Par-L-Trans we show (i); for cases Comm-Name-Trans and Comm-Proc-Trans we show (ii) and (iii), respectively. Similarly we prove the symmetric cases. □

We now prove that all interleavings of component transitions correspond to a transition of the composite configuration. Similarly, communication between the components corresponds to a τ-transition of the composite configuration. The next three propositions follow by Propositions 3.4, 3.6 (i), 3.7 (ii), and 3.8.

Proposition 6.22. *If $C_1 \parallel^{\overline{c}} C_2$ and $C_1 \xrightarrow{\eta} C_1'$ and $C_1' \parallel^{\overline{d}} C_2$ then $C_1 \parallel^{\overline{c}} C_2 \xrightarrow{\eta} C_1' \parallel^{\overline{d}} C_2$, with $\{\overline{d}\} = \{\overline{c}\} \setminus exp(\eta)$.*

Proposition 6.23. *Let $C_1 \parallel^{\overline{c}} C_2$ and $C_1 \xrightarrow{a!b} C_1'$ and $C_2 \xrightarrow{a?b} C_2'$. Then $C_1 \parallel^{\overline{c}} C_2 \xrightarrow{\tau} C_1' \parallel^{\overline{d}} C_2'$ and if $C_1 = \nu\overline{b}, b\,\langle\Delta, \mathcal{P}\rangle$ then $\{\overline{d}\} = \{\overline{c}\} \cup \{b\}$ else $\overline{d} = \overline{c}$.*

Proposition 6.24. *Let $C_1 \parallel^{\overline{c}} C_2$ and $C_1 \xrightarrow{a!\kappa} C_1'$ and $C_2 \xrightarrow{a?\alpha} C_2'$ and $\alpha, \kappa \in fresh(C_1, C_2)$. Then $C_1 \parallel^{\overline{c}} C_2 \xrightarrow{\tau} \leq_{\alpha\mapsto\kappa} C_1' \parallel^{\overline{c}} C_2'$.*

By the preceding propositions we can derive similar ones for weak transitions.

Proposition 6.25. *If* $C_1 \parallel^{\overline{c}} C_2$ *and* $C_1 \stackrel{\eta}{\Longrightarrow} C_1'$ *and* $C_1' \parallel^{\overline{c}} C_2$ *then* $C_1 \parallel^{\overline{c}} C_2 \stackrel{\eta}{\Longrightarrow} C_1' \parallel^{\overline{d}} C_2$, *with* $\{\overline{d}\} = \{\overline{c}\} \setminus exp(\eta)$.

Proposition 6.26. *Let* $C_1 \parallel^{\overline{c}} C_2$ *and* $C_1 \stackrel{a!b}{\Longrightarrow} C_1'$ *and* $C_2 \stackrel{a?b}{\Longrightarrow} C_2'$. *Then* $C_1 \parallel^{\overline{c}} C_2 \stackrel{\tau}{\Longrightarrow} C_1' \parallel^{\overline{d}} C_2'$ *and if* $C_1 = \nu \overline{b}, b\,\langle \Delta, \mathcal{P} \rangle$ *then* $\{\overline{d}\} = \{\overline{c}\} \cup \{b\}$ *else* $\overline{d} = \overline{c}$.

Proposition 6.27. *Let* $C_1 \parallel^{\overline{c}} C_2$ *and* $C_1 \stackrel{a!\kappa}{\Longrightarrow} C_1'$ *and* $C_2 \stackrel{a?\alpha}{\Longrightarrow} C_2'$ *and* $\alpha, \kappa \in fresh(C_1, C_2)$. *Then* $C_1 \parallel^{\overline{c}} C_2 \stackrel{\tau}{\Longrightarrow} \preceq_{\alpha \mapsto \kappa} C_1' \parallel^{\overline{c}} C_2'$.

We can now prove that $(\approx)^{\mathsf{par}} \subseteq (\approx)$ and derive the soundness of our technique.[2]

Theorem 6.28 (Parallel Context Closure of $(\approx)$)**.** $(\approx)^{\mathsf{par}} \subseteq (\approx)$.

Proof. We will show that $(\approx)^{\mathsf{par}}$ is a weak bisimulation. Let $C_1 \approx^{\mathsf{par}} C_6$ and $C_1 \stackrel{\eta}{\longrightarrow} C_1'$. By Definition 6.19, there exist C_2, C_3, C_4, C_5, δ and $\overline{c}$ such that δ is compatible to $C_2 \parallel^{\overline{c}} C_3$, $acon(C_2) = acon(C_4)$, $names(C_2) = names(C_4)$, $acon(C_3) = acon(C_5)$, $names(C_3) = names(C_5)$, and

$$C_1 \preceq_\delta C_2 \parallel^{\overline{c}} C_3 \qquad C_2 \approx C_4 \qquad C_3 \approx C_5 \qquad C_4 \parallel^{\overline{c}} C_5 \succeq_\delta C_6$$

The constants in δ do not appear in C_1 and C_1'. Moreover, $(\approx)$ is equivariant for concrete constants (Lemma 4.7) and, because of the definition of $(-)^{\mathsf{par}}$, here we consider only a fragment of $(\approx)$ which is equivariant also for abstract constants; that is the fragment that relates configurations with the same *acon* components (Lemma 4.8). Therefore we can always pick a δ such that $dom(\delta) \cap acon(\eta) = \emptyset$, $codom(\delta) \cap ccon(\eta) = \emptyset$. Hence we can apply Proposition 6.12 and derive that there exist C' and $t \succeq_\delta \eta$ such that $C_2 \parallel^{\overline{c}} C_3 \stackrel{t}{\longrightarrow} C' \succeq_\delta C_1'$.

[2] An alternative method for proving preservation of weak bisimulation by parallel contexts is by defining the following weak bisimulation up to substitution, prove its soundness w.r.t. weak bisimulation, and show that $\{(C_1 \parallel^{\overline{c}} C_2,\ C_1' \parallel^{\overline{c}} C_2') \mid C_1 \approx C_1',\ C_2 \approx C_2',\ acon(C_i) = acon(C_i'),\ names(C_i) = names(C_i')\}$ is a weak bisimulation up to substitution. This proof strategy would not require the definition of the relation $(\succeq_\delta)$ and Propositions 6.12 – 6.14 but it would require a proof of soundness of the up-to technique.

Definition. $\mathbb{R}$ is a weak bisimulation up to substitution if for all $C\ \mathbb{R}\ C'$:

1. $C\{\alpha_1/\alpha_2\}\ \mathbb{R}\ C'\{\alpha_1/\alpha_2\}$ and $C\{\kappa_1/\kappa_2\}\ \mathbb{R}\ C'\{\kappa_1/\kappa_2\}$
2. If $\eta \neq \tau$ and $C \stackrel{\eta}{\longrightarrow} C_1$ then there exists C_1' such that $C' \stackrel{\eta}{\Longrightarrow} C_1'$ and $C_1\ \mathbb{R}\ C_1'$.
3. If $C \stackrel{\tau}{\longrightarrow} C_1$ then there exist C_1' such that $C' \stackrel{\tau}{\Longrightarrow} C_1'$ and $C_1\ \mathbb{R}\ C_1'$ or $C_1 \preceq_{\alpha \mapsto \kappa} \mathbb{R} \succeq_{\alpha \mapsto \kappa} C_1'$, for some α and κ.
4. The converse of (2) and (3).

By Proposition 6.14, it suffices to show that there exist $C'_2, C'_3, C'_4, C'_5, C'', \delta'$, and d such that δ' is compatible to $C'_2 \parallel^{\overline{d}} C'_3$, $acon(C'_2) = acon(C'_4)$, $names(C'_2) = names(C'_4)$, $acon(C'_3) = acon(C'_5)$, $names(C'_3) = names(C'_5)$, and

$$C' \preceq_{\delta'} C'_2 \parallel^{\overline{d}} C'_3 \qquad C'_2 \approx C'_4 \qquad C'_3 \approx C'_5 \qquad C'_4 \parallel^{\overline{d}} C'_5 \succeq_{\delta'} C'' \qquad C_4 \parallel^{\overline{c}} C_5 \stackrel{t}{\Longrightarrow} C''$$

We proceed by induction on $t \succeq_\delta \eta$.

Case SubstTr-$\epsilon\eta$: $\eta \succeq_\epsilon \eta$. We proceed by cases on η.

♦ $\eta \neq \tau$: we take $\delta' = \epsilon$ and $\{\overline{d}\} = \{\overline{c}\} \setminus exp(\eta)$ and apply Proposition 6.20 which gives us that there exists C'_2 such that

$$C' \preceq_\epsilon C'_2 \parallel^{\overline{d}} C_3 \qquad C_2 \stackrel{\eta}{\longrightarrow} C'_2 \qquad C'_2 \parallel^{\overline{d}} C'_3$$

or the symmetric property. We prove only the first case; the proof of the symmetric case is similar. By $C_2 \approx C_4$, there exists C'_4 such that $C_4 \stackrel{\eta}{\Longrightarrow} C'_4$ and $C'_2 \approx C'_4$. By Proposition 6.17 $C'_4 \parallel^{\overline{d}} C_5$ and by Proposition 6.25 we get C'' such that

$$C'_4 \parallel^{\overline{d}} C_5 \succeq_\epsilon C'' \qquad C_4 \parallel^{\overline{c}} C_5 \stackrel{\eta}{\Longrightarrow} C''$$

Moreover, we can derive $acon(C'_2) = acon(C'_4)$ and $names(C'_2) = names(C'_4)$ by cases on the transition and using Proposition 3.4.

♦ $\eta = \tau$: by Proposition 6.21 we get the following cases (and their symmetric):

(i) There exists C'_2 such that

$$C' \preceq_\epsilon C'_2 \parallel^{\overline{c}} C_3 \qquad C_2 \stackrel{\tau}{\longrightarrow} C'_2$$

This case is completed similar to the case for $\eta \neq \tau$, taking $\delta' = \epsilon$, $\overline{d} = \overline{c}$, using Propositions 6.17 and 6.25.

(ii) There exist C'_2, C'_3, and $\overline{d}$ such that

$$C' \preceq_\epsilon C'_2 \parallel^{\overline{d}} C'_3 \qquad C_2 \stackrel{a!b}{\longrightarrow} C'_2 \qquad C_3 \stackrel{a?b}{\longrightarrow} C'_3$$

and if $C_2 = \nu\overline{b}, b\langle\Delta, \mathcal{P}\rangle$ then $\{\overline{d}\} = \{\overline{c}\} \cup \{b\}$ else $\overline{d} = \overline{c}$. By $C_2 \approx C_4$ and $C_3 \approx C_5$, there exist C'_4 and C'_5 such that $C_4 \stackrel{a!b}{\Longrightarrow} C'_4$, $C_5 \stackrel{a?b}{\Longrightarrow} C'_5$, $C'_2 \approx C'_4$, and $C'_3 \approx C'_5$. By Proposition 6.26 we get C'' such that

$$C'_4 \parallel^{\overline{d}} C'_5 \succeq_\epsilon C'' \qquad C_4 \parallel^{\overline{c}} C_5 \stackrel{\tau}{\Longrightarrow} C''$$

Moreover, by Proposition 3.4, we can derive $acon(C'_2) = acon(C'_4)$, $names(C'_2) = names(C'_4)$, $acon(C'_3) = acon(C'_5)$, and $names(C'_3) = names(C'_5)$.

(iii) There exist $\alpha, \kappa \in \mathit{fresh}(C_2, C_3)$, C_2', and C_3' such that

$$C' \preceq_{\alpha \mapsto \kappa} C_2' \parallel^{\overline{c}} C_3' \qquad C_2 \xrightarrow{a!\kappa} C_2' \qquad C_3 \xrightarrow{a?\alpha} C_3'$$

By $C_2 \approx C_4$ and $C_3 \approx C_5$, there exist C_4' and C_5' such that $C_4 \xRightarrow{a!\kappa} C_4'$, $C_5 \xRightarrow{a?\alpha} C_5'$, $C_2' \approx C_4'$, and $C_3' \approx C_5'$. Because $\mathit{acon}(C_2) = \mathit{acon}(C_4)$, $\mathit{acon}(C_3) = \mathit{acon}(C_5)$, and $\mathit{acon}(C_2) = \mathit{acon}(C_4)$ $\mathit{acon}(C_3) = \mathit{acon}(C_5)$ (by Proposition 4.6) we have $\alpha, \kappa \in \mathit{fresh}(C_4, C_5)$. Thus, by Proposition 6.27 we get C'' such that

$$C_4' \parallel^{\overline{c}} C_5' \succeq_{\alpha \mapsto \kappa} C'' \qquad C_4 \parallel^{\overline{c}} C_5 \xRightarrow{\tau} C''$$

Moreover, by Proposition 3.4, we can derive $\mathit{acon}(C_2') = \mathit{acon}(C_4')$, $\mathit{names}(C_2') = \mathit{names}(C_4')$, $\mathit{acon}(C_3') = \mathit{acon}(C_5')$, and $\mathit{names}(C_3') = \mathit{names}(C_5')$.

Case SubstTr-id: $\delta = \alpha \mapsto \kappa, \delta_0$, $t \succeq_{\delta_0} \eta$, $\alpha \notin \mathit{acon}(\eta)$, $\kappa \notin \mathit{ccon}(\eta)$. This case follows directly by the induction hypothesis on $t \succeq_{\delta_0} \eta$.

Case SubstTr-$\alpha\kappa$: $\delta = \alpha \mapsto \kappa, \delta_0$, $t = \mathtt{app}\,\alpha, \mathtt{app}\,\kappa, t_0$, $t_0 \succeq_{\delta_0} \eta$, $\alpha \notin \mathit{acon}(\eta)$, $\kappa \notin \mathit{ccon}(\eta)$. By definition of compatible substitution environment (Definition 6.18) we have that the transitions $\mathtt{app}\,\alpha$ and $\mathtt{app}\,\kappa$ will be performed by the configurations C_2 and C_3, respectively, or vice-versa. Hence, by Proposition 6.20, we have

$$C_2 \parallel^{\overline{c}} C_3 \xrightarrow{\mathtt{app}\,\alpha} C_2 \parallel^{\overline{c}} C_3' \xrightarrow{\mathtt{app}\,\kappa} C_2' \parallel^{\overline{c}} C_3' \xrightarrow{t} C' \qquad C_2 \xrightarrow{\mathtt{app}\,\alpha} C_2' \qquad C_3 \xrightarrow{\mathtt{app}\,\kappa} C_3'$$

By Propositions 6.25, $C_2 \approx C_4$, and $C_3 \approx C_5$, there exist C_4', and C_5' such that

$$C_4 \xRightarrow{\mathtt{app}\,\alpha} C_4' \parallel^{\overline{c}} C_5 \xRightarrow{\mathtt{app}\,\kappa} C_4' \parallel^{\overline{c}} C_5' \qquad C_2' \approx C_4' \qquad C_3' \approx C_5'$$

The proof is completed by applying the induction hypothesis at $t_0 \succeq_{\delta_0} \eta$. □

Corollary 6.29 (Parallel Context Closure of $(\simeq)$)**.** *If $P \simeq P'$ then for any process Q, $Q \mid P \simeq Q \mid P'$.*

Proof. By the premise and Definition 4.9, there exist $\overline{b}$ such that $\langle \{\overline{b}\}, P \rangle \approx \langle \{\overline{b}\}, P' \rangle$.

Let Q be a process with $\mathit{fn}(Q) \subseteq \{\overline{n}\}$. By Lemma 4.13, $\langle \{\overline{b}\} \cup \{\overline{n}\}, P \rangle \approx \langle \{\overline{b}\} \cup \{\overline{n}\}, P' \rangle$ and $\langle \{\overline{b}\} \cup \{\overline{n}\}, Q \rangle \approx \langle \{\overline{b}\} \cup \{\overline{n}\}, Q \rangle$. By Definition 6.19, $\langle \{\overline{b}\} \cup \{\overline{n}\}, Q \mid P \rangle$ $(\approx)^{\mathsf{par}}$ $\langle \{\overline{b}\} \cup \{\overline{n}\}, Q \mid P' \rangle$, and by Theorem 6.28 $\langle \{\overline{b}\} \cup \{\overline{n}\}, Q \mid P \rangle \approx \langle \{\overline{b}\} \cup \{\overline{n}\}, Q \mid P' \rangle$, Hence, by Definition 4.9, $Q \mid P \simeq Q \mid P'$. □

Theorem 6.30 (Soundness)**.** $(\simeq) \subseteq (\cong_{\mathrm{pcxt}})$.

Proof. In Propositions 6.4 and 6.3 and Corollary 6.29 we have shown that $(\simeq)$ preserves barbs, is reduction-closed, and preserves parallel contexts. Thus $(\simeq)$ is included in the largest relation with these properties, namely $(\cong_{\mathrm{pcxt}})$. □

7. Completeness of Weak Bisimilarity

Here we prove that $(\cong_{\mathrm{pcxt}})$ is included in $(\simeq)$. To do this we give a translation of LTS configurations into concrete processes.

7.1. Concretion of Configurations

We start with the definition of the translation of LTS configurations to concrete processes.

Definition 7.1 (Concretion)**.** *Let $\nu\overline{a}\,\langle\Delta, \mathcal{P}\rangle$ be a well-formed configuration, and f bijection that assigns fresh names (w.r.t. names(Δ) and $\overline{a}$) to the abstract and concrete constants in Δ. Then the concretion of $\mathcal{P}$, Δ, and a configuration are defined as follows:*

$$\mathcal{P}/f \stackrel{\text{def}}{=} \mathcal{P}\{\overline{(\lambda c?.\mathtt{0})/\alpha}\} \qquad \text{where } f(\alpha_i) = c_i, \text{ for all } i$$

$$\Delta/f \stackrel{\text{def}}{=} \prod_{\Delta(\kappa)=\mathcal{V}, f(\kappa)=c} *(c?.\mathtt{app}\,\mathcal{V}/f)$$

$$\nu\overline{a}\,\langle\Delta, \mathcal{P}\rangle/f \stackrel{\text{def}}{=} \nu\overline{a}\,(\Delta/f \mid \mathcal{P}/f)$$

The purpose of the concretion of a configuration is to simulate the handling of higher-order inputs and outputs in the LTS. When the abstract process applies a higher-order value that has been provided by the context the LTS simply raises a signal to the observer. The corresponding concrete process signals the observer via a communication on a unique global channel. Similarly, at any point in the execution, the LTS allows the observer to run a value that has been provided by the process (and is indexed in the environment of the configuration). The corresponding concrete process allows the same behaviour by exposing a service listening on a global channel; communication on the channel runs the value.

Because abstract and concrete constants are mapped to unique global channels by f, there is no possibility of interaction between the signal due to the application of an abstract constant and a service corresponding to a concrete constant. We emphasize this by using only inputs prefixes in the concretion of a configuration.

We now show that the reductions of translated LTS configurations are simulated by τ-transitions of the configurations.

Lemma 7.2. *If $\mathcal{P}/f \equiv Q$ then there exists $\mathcal{Q}_0$ such that $Q = \mathcal{Q}_0/f$ and that $\mathcal{P} \equiv \mathcal{Q}_0$.*

Proof. By induction on the height of the derivation tree $\mathcal{P}/f \equiv Q$. □

Lemma 7.3. *If $\Gamma, x : t \vdash \mathcal{P}/f : \mathtt{OK}$ and $\vdash \mathcal{V}/f : t$ then $\Gamma \vdash \mathcal{P}/f\{(\mathcal{V}/f)/x\} = \mathcal{P}\{\mathcal{V}/x\}/f : \mathtt{OK}$.*

Proof. By induction on the height of the derivation tree $\Gamma, x : t \vdash \mathcal{P}/f : \mathtt{OK}$. □

Lemma 7.4. *If $\nu\overline{a}.\mathcal{P}/f \;\rightarrow\; Q$ and $\nu\overline{a}\,\langle\Delta, \mathcal{P}\rangle$ is well-formed then exactly one of the following is true:*

(i) there exist $\overline{b}$ and $\mathcal{Q}_0$ such that

$$Q \equiv \nu\overline{b}.\,\mathcal{Q}_0/f \qquad \nu\overline{a}\,\langle\Delta, \mathcal{P}\rangle \stackrel{\tau}{\Longrightarrow} \nu\overline{b}\,\langle\Delta, \mathcal{Q}_0\rangle$$

(ii) there exist $\overline{b}$, $\mathcal{Q}_0$, and c such that

$$Q \equiv \nu\overline{b}.\,\mathcal{Q}_0/f \mid c?.\mathtt{0} \qquad \nu\overline{a}\,\langle\Delta, \mathcal{P}\rangle \stackrel{\tau}{\Longrightarrow} \nu\overline{b}\,\langle\Delta, \mathcal{Q}_0 \mid \mathtt{app}\,\alpha\rangle \qquad f(\alpha) = c$$

Proof. By induction on the height of the derivation tree $\mathcal{P}/f \to Q$, using Proposition 4.11 and Lemmas 7.2 and 7.3. □

Proposition 7.5. *If $\nu\overline{a}\,\langle\Delta, \mathcal{P}\rangle/f \to Q$ then exactly one of the following is true:*

(i) there exist $\overline{b}$ and Q_0 such that

$$Q \equiv \nu\overline{b}\,\langle\Delta, Q_0\rangle/f \qquad \nu\overline{a}\,\langle\Delta, \mathcal{P}\rangle \overset{\tau}{\Longrightarrow} \nu\overline{b}\,\langle\Delta, Q_0\rangle$$

(ii) there exist $\overline{b}$, Q_0, and c such that

$$Q \equiv \nu\overline{b}\,\langle\Delta, Q_0\rangle/f \mid c?.\mathbb{0} \qquad \nu\overline{a}\,\langle\Delta, \mathcal{P}\rangle \overset{\tau}{\Longrightarrow} \nu\overline{b}\,\langle\Delta, Q_0 \mid \mathtt{app}\,\alpha\rangle \qquad f(\alpha) = c$$

Proof. Because Δ/f can not take any steps or communicate with $\mathcal{P}/f$, it must be that for some Q_1

$$\nu\overline{a}\,\langle\Delta, \mathcal{P}\rangle/f \to \nu\overline{a}.\,(\Delta/f \mid Q_1) \equiv Q \qquad \nu\overline{a}.\,\mathcal{P}/f \to \nu\overline{a}.\,Q_1$$

and by Lemma 7.4 there exist $\overline{b}$ and Q_0 such that

$$\nu\overline{a}.\,Q_1 \equiv \nu\overline{b}.\,Q_0/f \qquad \nu\overline{a}\,\langle\Delta, \mathcal{P}\rangle \overset{\tau}{\Longrightarrow} \nu\overline{b}\,\langle\Delta, Q_0\rangle$$

or there exist $\overline{b}$, Q_0, and c such that

$$\nu\overline{a}.\,Q_1 \equiv \nu\overline{b}.\,Q_0/f \mid c?.\mathbb{0} \qquad \nu\overline{a}\,\langle\Delta, \mathcal{P}\rangle \overset{\tau}{\Longrightarrow} \nu\overline{b}\,\langle\Delta, Q_0 \mid \mathtt{app}\,\alpha\rangle \qquad f(\alpha) = c$$

By Proposition 3.4 (v) and (vii) we have that $\{\overline{a}\} \subseteq \{\overline{b}\}$ in both cases. Hence, either

$$Q \equiv \nu\overline{b}.\,(\Delta/f \mid Q_0/f) \equiv \nu\overline{b}\,\langle\Delta, Q_0\rangle/f \qquad \nu\overline{a}\,\langle\Delta, \mathcal{P}\rangle \overset{\tau}{\Longrightarrow} \nu\overline{b}\,\langle\Delta, Q_0\rangle$$

or there exist $\overline{b}$, Q_0, and c such that

$$Q \equiv \nu\overline{b}.\,(\Delta/f \mid Q_0/f) \mid c?.\mathbb{0} \equiv \nu\overline{b}\,\langle\Delta, Q_0\rangle/f \mid c?.\mathbb{0}$$
$$\nu\overline{a}\,\langle\Delta, \mathcal{P}\rangle \overset{\tau}{\Longrightarrow} \nu\overline{b}\,\langle\Delta, Q_0 \mid \mathtt{app}\,\alpha\rangle \qquad f(\alpha) = c$$ □

From the above we conclude that reductions of translated configurations correspond to τ-transitions of the original configurations, possibly accumulating several $\mathtt{app}\,\alpha$ processes.

Corollary 7.6. *If $\nu\overline{a}\,\langle\Delta, \mathcal{P}\rangle/f \to^* Q$ then there exist $\overline{b}$, $\overline{c,\alpha}$, Q_0 such that*

$$Q \equiv \nu\overline{b}\,\langle\Delta, Q_0\rangle/f \mid \prod_{c_i \in \{\overline{c}\}} c_i?.\mathbb{0} \quad \nu\overline{a}\,\langle\Delta, \mathcal{P}\rangle \overset{\tau}{\Longrightarrow} \nu\overline{b}\,\langle\Delta, Q_0 \mid \prod_{\alpha_i \in \{\overline{\alpha}\}} \mathtt{app}\,\alpha_i\rangle \quad f(\alpha_i) = c_i$$

Conversely, τ-transitions of configurations correspond to (zero or one) reductions of their corresponding translations.

Proposition 7.7. *If* $\nu\overline{a}\,\langle \Delta, \mathcal{P}\rangle \xrightarrow{\tau} \nu\overline{b}\,\langle \Delta, Q\rangle$ *then one of the following holds.*

(i) $\nu\overline{a}\,\langle \Delta, \mathcal{P}\rangle/f \equiv \nu\overline{b}\,\langle \Delta, Q\rangle/f$, *or*

(ii) $\nu\overline{a}\,\langle \Delta, \mathcal{P}\rangle/f \to \nu\overline{b}\,\langle \Delta, Q\rangle/f$.

Proof. By rule induction. □

In what follows we will use an eta-expansion lemma:

Lemma 7.8. $P \cong_{\text{pcxt}} \mathtt{app}\,\lambda P$

Proof. By considering the smallest relation on configurations containing the identity and satisfying the axiom $P = \mathtt{app}\,\lambda P$, and by showing that it is a weak bisimulation. □

7.2. Completeness

The completeness proof is based on the fact that the following relation on configurations is a weak bisimulation.

Definition 7.9 ($\mathbb{X}$)**.**

$$\mathbb{X} \stackrel{\text{def}}{=} \{(\nu\overline{a}\,\langle \Delta, \mathcal{P}\rangle, \nu\overline{a'}\,\langle \Delta', \mathcal{P}'\rangle) \mid \exists f.\ \nu\overline{a}\,\langle \Delta, \mathcal{P}\rangle/f \cong_{\text{pcxt}} \nu\overline{a'}\,\langle \Delta', \mathcal{P}'\rangle/f\}$$

First we show that $\mathbb{X}$ is closed under τ-transitions.

Proposition 7.10. *If* $\nu\overline{a}\,\langle \Delta, \mathcal{P}\rangle\ \mathbb{X}\ \nu\overline{a'}\,\langle \Delta', \mathcal{P}'\rangle$ *and* $\nu\overline{a}\,\langle \Delta, \mathcal{P}\rangle \xrightarrow{\tau} \nu\overline{b}\,\langle \Delta, Q\rangle$ *then there exist* $\overline{b'}$ *and* Q' *such that*

$$\nu\overline{a'}\,\langle \Delta', \mathcal{P}'\rangle \stackrel{\tau}{\Longrightarrow} \nu\overline{b'}\,\langle \Delta', Q'\rangle \qquad \nu\overline{a}\,\langle \Delta, Q\rangle\ \mathbb{X}\ \nu\overline{a'}\,\langle \Delta', Q'\rangle$$

Proof. By the first premise and Definition 7.9 we get that there exists f such that

$$\nu\overline{a}\,\langle \Delta, \mathcal{P}\rangle/f \cong_{\text{pcxt}} \nu\overline{a'}\,\langle \Delta', \mathcal{P}'\rangle/f \tag{5}$$

By the second premise and Proposition 7.7, either $\nu\overline{a}\,\langle \Delta, \mathcal{P}\rangle/f \equiv \nu\overline{b}\,\langle \Delta, Q\rangle/f$, or $\nu\overline{a}\,\langle \Delta, \mathcal{P}\rangle/f \to \nu\overline{b}\,\langle \Delta, Q\rangle/f$. In the former case the proof is completed because, by Proposition 4.11 and Theorem 6.30, $(\equiv) \subseteq (\simeq) \subset (\cong_{\text{pcxt}})$, hence $\nu\overline{b}\,\langle \Delta, Q\rangle/f \cong_{\text{pcxt}} \nu\overline{a'}\,\langle \Delta', \mathcal{P}'\rangle/f$ and $\nu\overline{b}\,\langle \Delta, Q\rangle\ \mathbb{X}\ \nu\overline{a'}\,\langle \Delta', \mathcal{P}'\rangle$. In the latter case, by (5) and Definition 2.7, we have that there exists Q' such that

$$\nu\overline{a'}\,\langle \Delta, \mathcal{P}'\rangle/f \to^{*} Q' \tag{6}$$

$$\nu\overline{b}\,\langle \Delta, Q\rangle/f \cong_{\text{pcxt}} Q' \tag{7}$$

By (6) and Corollary 7.6, there exist $\overline{b'}$, $\overline{c, \alpha}$, and Q'_0 such that $\overline{f(\alpha) = c}$ and

$$Q' \equiv \nu\overline{b'}\,\langle \Delta', Q'_0\rangle/f \mid \prod_{c \in \{\overline{c}\}} c?.\mathtt{0} \tag{8}$$

$$\nu\overline{a'}\,\langle \Delta', \mathcal{P}'\rangle \stackrel{\tau}{\Longrightarrow} \nu\overline{b'}\,\langle \Delta', Q'_0 \mid \prod_{\alpha \in \{\overline{\alpha}\}} \mathtt{app}\,\alpha\rangle$$

By (7), (8), the fact that $(\equiv) \subseteq (\simeq) \subset (\cong_{\text{pcxt}})$, and Lemma 7.8 we have

$$\begin{aligned} \nu\overline{b}\,\langle \Delta,\, Q\rangle/f &\cong_{\text{pcxt}} \nu\overline{b'}\,\langle \Delta',\, Q'_0\rangle/f \mid \textstyle\prod_{c\in\{\overline{c}\}} c?.\mathtt{0} \\ &\cong_{\text{pcxt}} \nu\overline{b'}\,\langle \Delta',\, Q'_0\rangle/f \mid \textstyle\prod_{c\in\{\overline{c}\}} \mathsf{app}\,\lambda c?.\mathtt{0} \\ &= \quad \nu\overline{b'}\,\langle \Delta',\, Q'_0 \mid \textstyle\prod_{\alpha\in\{\overline{\alpha}\}} \mathsf{app}\,\alpha\rangle/f \end{aligned}$$

thus

$$\nu\overline{b}\,\langle \Delta,\, Q\rangle \;\mathbb{X}\; \nu\overline{b'}\,\langle \Delta',\, Q'_0 \mid \prod_{\alpha\in\{\overline{\alpha}\}} \mathsf{app}\,\alpha\rangle$$

□

Proposition 7.11. *If* $\nu\overline{a}\,\langle \Delta_1,\, \mathcal{P}\rangle \;\mathbb{X}\; \nu\overline{a'}\,\langle \Delta'_1,\, \mathcal{P}'\rangle$ *and for some* $\eta \neq \tau$, $\nu\overline{a}\,\langle \Delta_1,\, \mathcal{P}\rangle \xrightarrow{\eta} \nu\overline{b}\,\langle \Delta_2,\, Q\rangle$ *then there exist* $\overline{b'}$, Δ'_2, *and* Q' *such that*

$$\nu\overline{a'}\,\langle \Delta'_1,\, \mathcal{P}'\rangle \overset{\eta}{\Longrightarrow} \nu\overline{b'}\,\langle \Delta'_2,\, Q'\rangle \qquad \nu\overline{b}\,\langle \Delta_2,\, Q\rangle \;\mathbb{X}\; \nu\overline{b'}\,\langle \Delta'_2,\, Q'\rangle$$

Proof. We proceed by cases on η.

Case $\eta = \mathsf{app}\,\alpha$: By the second premise and Proposition 3.4 (v),

$$\Delta_1 = \Delta_2 \qquad \mathcal{P} \mathrel{\hat{=}} Q \mid \mathsf{app}\,\alpha \qquad \{\overline{a}\} = \{\overline{b}\}$$

Thus by the first premise and Definition 7.9 we get that there exists f such that

$$\nu\overline{a}\,\langle \Delta_1,\, Q \mid \mathsf{app}\,\alpha\rangle/f \cong_{\text{pcxt}} \nu\overline{a'}\,\langle \Delta'_1,\, \mathcal{P}'\rangle/f$$

Because $(\cong_{\text{pcxt}})$ preserves parallel contexts we pick the context $C = [\cdot] \mid c!.\mathtt{0}$, where $f(\alpha) = c$. We have

$$\begin{aligned} \nu\overline{a}\,\langle \Delta_1,\, Q \mid \mathsf{app}\,\alpha\rangle/f \mid c!.\mathtt{0} = \quad & \nu\overline{a}\,\langle \Delta_1,\, Q\rangle/f \mid \mathsf{app}\,\lambda c?.\mathtt{0} \mid c!.\mathtt{0} \\ \cong_{\text{pcxt}} \; & \nu\overline{a'}\,\langle \Delta'_1,\, \mathcal{P}'\rangle/f \mid c!.\mathtt{0} \end{aligned}$$

Thus, by Definition 2.7 and because

$$\nu\overline{a}\,\langle \Delta_1,\, Q\rangle/f \mid \mathsf{app}\,\lambda c?.\mathtt{0} \mid c!.\mathtt{0} \to^* \nu\overline{a}\,\langle \Delta_1,\, Q\rangle/f \qquad \nu\overline{a}\,\langle \Delta_1,\, Q\rangle/f \not\downarrow_c$$

there must be Q' such that

$$\begin{aligned} \nu\overline{a'}\,\langle \Delta'_1,\, \mathcal{P}'\rangle/f \mid c!.\mathtt{0} \to^* Q' \qquad Q' \not\downarrow_c \\ \nu\overline{a}\,\langle \Delta_1,\, Q\rangle/f \cong_{\text{pcxt}} Q' \end{aligned} \tag{9}$$

Therefore, there exists Q'_1 such that

$$\begin{aligned} \nu\overline{a'}\,\langle \Delta'_1,\, \mathcal{P}'\rangle/f \to^* Q'_1 \mid c?.\mathtt{0} \\ \nu\overline{a'}\,\langle \Delta'_1,\, \mathcal{P}'\rangle/f \mid c!.\mathtt{0} \to^* Q'_1 \mid c?.\mathtt{0} \mid c!.\mathtt{0} \to^* Q'_1 \to^* Q' \end{aligned} \tag{10}$$

and by Corollary 7.6 there exist $\overline{b'}$, $\overline{c, \alpha}$, Q'_0 such that $f(\alpha_i) = c_i$ and

$$Q'_1 \mid c?.\mathbb{0} \equiv \nu \overline{b'} \langle \Delta'_1, Q'_0 \rangle / f \mid c?.\mathbb{0} \mid \prod_{c_i \in \{\overline{c}\}} c_i?.\mathbb{0} \tag{11}$$

$$\nu \overline{a'} \langle \Delta'_1, \mathcal{P}' \rangle \stackrel{\tau}{\Longrightarrow} \nu \overline{b'} \langle \Delta'_1, Q'_0 \mid \mathtt{app}\, \alpha \mid \prod_{\alpha_i \in \{\overline{\alpha}\}} \mathtt{app}\, \alpha_i \rangle \tag{12}$$

By (10) and (11),

$$\nu \overline{b'} \langle \Delta'_1, Q'_0 \rangle / f \mid \prod_{c_i \in \{\overline{c}\}} c_i?.\mathbb{0} \to^* Q'$$

and thus

$$\nu \overline{b'} \langle \Delta'_1, Q'_0 \mid \prod_{\alpha_i \in \{\overline{\alpha}\}} \mathtt{app}\, \alpha_i \rangle / f \to^* \nu \overline{b'} \langle \Delta'_1, Q'_0 \rangle / f \mid \prod_{c_i \in \{\overline{c}\}} c_i?.\mathbb{0} \to^* Q'$$

By Corollary 7.6 there exist $\overline{d'}$, $\overline{c', \alpha'}$, Q'_2 such that $f(\alpha'_i) = c'_i$ and

$$Q' \equiv \nu \overline{d'} \langle \Delta'_1, Q'_2 \rangle / f \mid \prod_{c'_i \in \{\overline{c'}\}} c'_i?.\mathbb{0} \tag{13}$$

$$\nu \overline{b'} \langle \Delta'_1, Q'_0 \mid \prod_{\alpha_i \in \{\overline{\alpha}\}} \mathtt{app}\, \alpha_i \rangle \stackrel{\tau}{\Longrightarrow} \nu \overline{d'} \langle \Delta'_1, Q'_2 \mid \prod_{\alpha'_i \in \{\overline{\alpha'}\}} \mathtt{app}\, \alpha'_i \rangle \tag{14}$$

Hence by (12) and (14)

$$\begin{aligned} \nu \overline{a'} \langle \Delta'_1, \mathcal{P}' \rangle &\stackrel{\tau}{\Longrightarrow} \nu \overline{b'} \langle \Delta'_1, Q'_0 \mid \mathtt{app}\, \alpha \mid \textstyle\prod_{\alpha_i \in \{\overline{\alpha}\}} \mathtt{app}\, \alpha_i \rangle \\ &\stackrel{\mathtt{app}\, \alpha}{\longrightarrow} \nu \overline{b'} \langle \Delta'_1, Q'_0 \mid \mathbb{0} \mid \textstyle\prod_{\alpha_i \in \{\overline{\alpha}\}} \mathtt{app}\, \alpha_i \rangle \\ &\stackrel{\tau}{\Longrightarrow} \nu \overline{d'} \langle \Delta'_1, Q'_2 \mid \mathbb{0} \mid \textstyle\prod_{\alpha'_i \in \{\overline{\alpha'}\}} \mathtt{app}\, \alpha'_i \rangle \end{aligned}$$

Furthermore, by (9) and (13), the fact that $(\equiv) \subseteq (\simeq) \subset (\cong_{\mathrm{pcxt}})$, and Lemma 7.8 we have

$$\begin{aligned} \nu \overline{a} \langle \Delta_1, Q \rangle / f &\cong_{\mathrm{pcxt}} \nu \overline{d'} \langle \Delta'_1, Q'_2 \rangle / f \mid \textstyle\prod_{c'_i \in \{\overline{c'}\}} c'_i?.\mathbb{0} \\ &\cong_{\mathrm{pcxt}} \nu \overline{d'} \langle \Delta'_1, Q'_2 \rangle / f \mid \textstyle\prod_{c'_i \in \{\overline{c'}\}} \mathtt{app}\, \lambda c'_i?.\mathbb{0} \\ &= \quad \nu \overline{d'} \langle \Delta'_1, Q'_2 \mid \mathbb{0} \mid \textstyle\prod_{\alpha'_i \in \{\overline{\alpha'}\}} \mathtt{app}\, \alpha'_i \rangle / f \end{aligned}$$

Hence,

$$\nu \overline{a} \langle \Delta_1, Q \rangle \mathbb{X} \nu \overline{d'} \langle \Delta'_1, Q'_2 \mid \mathbb{0} \mid \prod_{\alpha'_i \in \{\overline{\alpha'}\}} \mathtt{app}\, \alpha'_i \rangle$$

The rest of the cases are proved similarly using the following contexts (in which r is a fresh channel):

- For $\eta = c!n$ we use the context

$$c?(x).\texttt{if}\, x = n\, \texttt{then}\, (r!.\mathbb{0} \mid r?.\mathbb{0})\, \texttt{else}\, \mathbb{0}$$

  when $n \in \mathit{names}(\Delta_1)$, and

$$c?(x).\texttt{if}\, x \in \mathit{names}(\Delta_1)\, \texttt{then}\, \mathbb{0}\, \texttt{else}\, (r!.\mathbb{0} \mid r?.\mathbb{0})$$

  otherwise. Here $\texttt{if}\, x \in \mathit{names}(\Delta_1)\, \texttt{then}\, P\, \texttt{else}\, Q$ is expressible in terms of $\texttt{if}\, x = n_i\, \texttt{then}\, P\, \texttt{else}\, Q$ because $\mathit{names}(\Delta_1)$ is a finite set of names. Moreover, r is a barb of the context in parallel with the process. After the communication between the process and the context on c and the communication on r it is no longer a barb—in this way we "force" the communication on channel c.

- For $\eta = c?n$ we use the context

$$c!\langle n\rangle.(r!.\mathbb{0} \mid r?.\mathbb{0})$$

- For $\eta = c!\kappa$ we use the context

$$c?(X).(*(c_{fr}?.\texttt{app}\, X) \mid r!.\mathbb{0} \mid r?.\mathbb{0})$$

  where c_{fr} is a fresh name.

- For $\eta = c?\alpha$ we use the context

$$c!\langle \lambda\, c_{fr}?.\mathbb{0}\rangle.(r!.\mathbb{0} \mid r?.\mathbb{0})$$

  where c_{fr} is a fresh name.

- For $\eta = \texttt{app}\, \kappa$ we use the context

$$c!.\mathbb{0}$$

  and a concretion function with $f(\kappa) = c$. □

Proposition 7.12. *$\mathbb{X}$ is a weak bisimulation.*

Proof. By Propositions 7.10 and 7.11 and by symmetry, $\mathbb{X}$ satisfies the conditions of Definition 4.4 for a weak bisimulation. □

Theorem 7.13 (Completeness)**.** $(\cong_{\text{pcxt}}) \subseteq (\simeq)$.

Proof. If $P \cong_{\text{pcxt}} P'$ then for names $\overline{n} \subseteq \mathit{fn}(P, P')$ we have

$$P = \langle\{\overline{n}\}, P\rangle/\emptyset \cong_{\text{pcxt}} \langle\{\overline{n}\}, P'\rangle/\emptyset = P'$$

By Definition 7.9 $\langle\{\overline{n}\}, P\rangle\ \mathbb{X}\ \langle\{\overline{n}\}, P'\rangle$ and by Proposition 7.12, $\langle\{\overline{n}\}, P\rangle \approx \langle\{\overline{n}\}, P'\rangle$. Thus, by Definition 4.9, $P \simeq P'$. □

8. Full Barbed Congruence

In this section we study reduction-closed barbed congruence ($\cong_{\text{cxt}}$) with arbitrary contexts. By the definitions of ($\cong_{\text{pcxt}}$) and ($\cong_{\text{cxt}}$) (Definitions 2.7 and 8.2) and Theorem 7.13 we have that

$$(\cong_{\text{cxt}}) \subseteq (\cong_{\text{pcxt}}) \subseteq (\simeq)$$

We show that weak bisimilarity implies reduction-closed barbed congruence ($(\simeq) \subseteq (\cong_{\text{cxt}})$) and therefore, as with the first-order π-calculus, parallel contextual equivalence coincides with reduction-closed barbed congruence for pp-π.

This property of bisimilarity is particularly useful because it means that it is a full congruence. In Sections 5.2 and 5.3 we used this fact to factor out the common replication context $*(-)$ of related processes, and therefore reduce the complexity of the bisimulation proofs.

First we give the definition for contexts.

Definition 8.1 (Contexts)**.** *A context $\mathcal{K}$ is derived from the following grammar:*

$$\begin{aligned}\mathcal{K} ::= {} & [\cdot] \mid \mathtt{0} \mid \mathcal{V}_K!\langle \mathcal{V}_K{:}t\rangle.\mathcal{K} \mid \mathcal{V}_K?(x{:}t).\mathcal{K} \mid \mathcal{K} \mid \mathcal{K} \\ & \mid \ \nu n.\mathcal{K} \mid \mathtt{app}\,\mathcal{V}_K \mid *(\mathcal{K}) \mid \mathtt{if}\,\mathcal{V}_K = \mathcal{V}_K\,\mathtt{then}\,\mathcal{K}\,\mathtt{else}\,\mathcal{K} \\ \mathcal{V}_K ::= {} & x \mid \lambda\mathcal{K} \mid n \mid \alpha\end{aligned}$$

We write $\mathcal{K}[P]$ (resp. $\mathcal{V}_K[P]$) to mean the replacement of all holes in K (resp. V_K) with P. We write K and V_K when contexts are derived by a core syntax that does not contain abstract constants α.

Definition 8.2 (Reduction-Closed Barbed Congruence ($\cong_{\text{cxt}}$))**.** ($\cong_{\text{cxt}}$) *is the largest congruence on closed processes that preserves barbs and is reduction closed; i.e. $P \cong_{\text{cxt}} P'$ if and only if*

(i) Barb preserving: *for all b, $P \Downarrow_b$ iff $P' \Downarrow_b$,*

(ii) Reduction closed: *for all P_1 with $P \to P_1$ there exists P'_1 such that $P' \to^* P'_1$ and $P_1 \cong_{\text{cxt}} P'_1$, and vice-versa, and*

(iii) Preserves contexts: *for all K, $K[P] \cong_{\text{cxt}} K[P']$.*

The conditions of ($\cong_{\text{cxt}}$) are stronger than those of ($\cong_{\text{pcxt}}$), hence the following Proposition.

Proposition 8.3. ($\cong_{\text{cxt}}$) $\subseteq$ ($\cong_{\text{pcxt}}$).

We will show that ($\simeq$) $\subseteq$ ($\cong_{\text{cxt}}$). As we proved in Section 6.2, ($\simeq$) is reduction-closed and barb-preserving. Therefore it suffices to show that ($\simeq$) preserves contexts.

Proposition 8.4. *Let $\langle\{\overline{\alpha},\overline{n}\}, \mathcal{P}\rangle \approx \langle\{\overline{\alpha},\overline{n}\}, \mathcal{P}'\rangle$ and V and Q are closed well-typed terms using names from $\overline{n}$. Then:*

(i) $\langle\{\overline{\alpha},\overline{n}\}, \mathcal{P} \mid Q\rangle \approx \langle\{\overline{\alpha},\overline{n}\}, \mathcal{P}' \mid Q\rangle$

(ii) $\langle\{\overline{\alpha},\overline{n}\}, \nu n_i.\mathcal{P}\rangle \approx \langle\{\overline{\alpha},\overline{n}\}, \nu n_i.\mathcal{P}'\rangle$

(iii) $\langle\{\overline{\alpha},\overline{n}\},\ \mathtt{app}\,\lambda\mathcal{P}\rangle \approx \langle\{\overline{\alpha},\overline{n}\},\ \mathtt{app}\,\lambda\mathcal{P}'\rangle$

(iv) $\langle\{\overline{\alpha},\overline{n}\},\ \mathtt{if}\,n_i = n_j\,\mathtt{then}\,\mathcal{P}\,\mathtt{else}\,Q\rangle \approx \langle\{\overline{\alpha},\overline{n}\},\ \mathtt{if}\,n_i = n_j\,\mathtt{then}\,\mathcal{P}'\,\mathtt{else}\,Q\rangle$

(v) $\langle\{\overline{\alpha},\overline{n}\},\ \mathtt{if}\,n_i = n_j\,\mathtt{then}\,Q\,\mathtt{else}\,\mathcal{P}\rangle \approx \langle\{\overline{\alpha},\overline{n}\},\ \mathtt{if}\,n_i = n_j\,\mathtt{then}\,Q\,\mathtt{else}\,\mathcal{P}'\rangle$

(vi) $\langle\{\overline{\alpha},\overline{n}\},\ c?(x{:}\mathtt{Nm}).\mathcal{P}\{x/b\}\rangle \approx \langle\{\overline{\alpha},\overline{n}\},\ c?(x{:}\mathtt{Nm}).\mathcal{P}'\{x/b\}\rangle$

(vii) $\langle\{\overline{\alpha},\overline{n}\},\ c?(x{:}\mathtt{Pr}).\mathcal{P}\{x/\alpha\}\rangle \approx \langle\{\overline{\alpha},\overline{n}\},\ c?(x{:}\mathtt{Pr}).\mathcal{P}'\{x/\alpha\}\rangle$

(viii) $\langle\{\overline{\alpha},\overline{n}\},\ c!\langle\lambda\mathcal{P}{:}\mathtt{Pr}\rangle.Q\rangle \approx \langle\{\overline{\alpha},\overline{n}\},\ c!\langle\lambda\mathcal{P}'{:}\mathtt{Pr}\rangle.Q\rangle$

(ix) $\langle\{\overline{\alpha},\overline{n}\},\ c!\langle V{:}t\rangle.\mathcal{P}\rangle \approx \langle\{\overline{\alpha},\overline{n}\},\ c!\langle V{:}t\rangle.\mathcal{P}'\rangle$

(x) $\langle\{\overline{\alpha},\overline{n}\},\ *(\mathcal{P})\rangle \approx \langle\{\overline{\alpha},\overline{n}\},\ *(\mathcal{P}')\rangle$

Proof. We prove each property separately:

(i) By Theorem 6.28, using that $\langle\{\overline{n}\},\ Q\rangle \approx \langle\{\overline{n}\},\ Q\rangle$.

(ii) By Lemma 4.14, $\nu n_i\,\langle\{\overline{\alpha},\overline{n}\}\setminus\{n_i\},\ \mathcal{P}\rangle \approx \nu n_i\,\langle\{\overline{\alpha},\overline{n}\}\setminus\{n_i\},\ \mathcal{P}'\rangle$. This congruence property follows by Lemmas 4.16 and 4.13.

(iii) By proving that the relation

$$(\approx) \cup \{(\langle\{\overline{\alpha},\overline{n}\},\ \mathtt{app}\,\lambda\mathcal{P}\rangle,\ \ \langle\{\overline{\alpha},\overline{n}\},\ \mathtt{app}\,\lambda\mathcal{P}'\rangle) \mid \langle\{\overline{\alpha},\overline{n}\},\ \mathcal{P}\rangle \approx \langle\{\overline{\alpha},\overline{n}\},\ \mathcal{P}'\rangle\}$$

is a weak bisimulation.

(iv) Similar to (iii)

(v) Similar to (iii)

(vi) We first prove that renaming channels in the observer's knowledge preserves weak bisimulation by showing that the relation

$$\{(\nu\overline{a}\,\langle(\Delta\{m/b\}) \uplus \{m\},\ \mathcal{P}\{m/b\}\rangle,\ \ \nu\overline{a'}\,\langle(\Delta'\{m/b\}) \uplus \{m\},\ \mathcal{P}'\{m/b\}\rangle) \mid b \notin \{\overline{a},\overline{a'}\} \text{ and } \nu\overline{a}\,\langle\Delta,\ \mathcal{P}\rangle\ \mathbb{R}\ \nu\overline{a'}\,\langle\Delta',\ \mathcal{P}'\rangle\}$$

is a weak bisimulation when $\mathbb{R}$ is a weak bisimulation. The property follows by showing that the relation

$$(\approx) \cup \{(\langle\{\overline{\alpha},\overline{n}\},\ c?(x{:}\mathtt{Nm}).\mathcal{P}\{x/b\}\rangle,\ \ \langle\{\overline{\alpha},\overline{n}\},\ c?(x{:}\mathtt{Nm}).\mathcal{P}'\{x/b\}\rangle) \mid c \in \{\overline{n}\} \text{ and } \langle\{\overline{\alpha},\overline{n}\},\ \mathcal{P}\rangle \approx \langle\{\overline{\alpha},\overline{n}\},\ \mathcal{P}'\rangle\}$$

is a weak bisimulation.

(vii) Similar to (vi)

(viii) We consider the relation

$$\mathbb{J} \stackrel{\text{def}}{=} \{(\nu\overline{a}\,\langle\Delta \cup \{\overline{\alpha},\overline{n},\kappa{\mapsto}\lambda\mathcal{P}\},\ Q\rangle,\ \ \nu\overline{a'}\,\langle\Delta' \cup \{\overline{\alpha},\overline{n},\kappa{\mapsto}\lambda\mathcal{P}'\},\ Q'\rangle) \mid \kappa \text{ fresh and } \nu\overline{a}\,\langle\Delta,\ Q\rangle \approx \nu\overline{a'}\,\langle\Delta',\ Q'\rangle\}$$

where $\mathcal{P}$ and $\mathcal{P}'$ are given by the premise of the proposition. We show that it is a weak bisimulation by cases on η:

♦ $\eta \neq \mathsf{app}\,\kappa$: We have:

$$\nu\overline{a}\,\langle \Delta \cup \{\overline{\alpha}, \overline{n}, \kappa \mapsto \lambda\mathcal{P}\},\, Q\rangle \xrightarrow{\eta} \nu\overline{b}\,\langle \Delta_1 \cup \{\overline{\alpha}, \overline{n}, \kappa \mapsto \lambda\mathcal{P}'\},\, Q_1\rangle$$

By Proposition 3.6 (ii),

$$\nu\overline{a}\,\langle \Delta,\, Q\rangle \xrightarrow{\eta} \nu\overline{b}\,\langle \Delta_1,\, Q_1\rangle$$

Because $(\approx)$ is a weak bisimulation, there exist $\overline{b'}$, Δ_1', Q_1' such that

$$\nu\overline{a'}\,\langle \Delta',\, Q'\rangle \stackrel{\eta}{\Longrightarrow} \nu\overline{b'}\,\langle \Delta_1',\, Q_1'\rangle \qquad \nu\overline{b}\,\langle \Delta_1,\, Q_1\rangle \approx \nu\overline{b'}\,\langle \Delta_1',\, Q_1'\rangle$$

By Proposition 3.6 (i),

$$\nu\overline{a'}\,\langle \Delta' \cup \{\overline{\alpha}, \overline{n}, \kappa \mapsto \lambda\mathcal{P}\},\, Q'\rangle \stackrel{\eta}{\Longrightarrow} \nu\overline{b'}\,\langle \Delta_1' \cup \{\overline{\alpha}, \overline{n}, \kappa \mapsto \lambda\mathcal{P}'\},\, Q_1'\rangle$$

and by definition of $\mathbb{J}$,

$$\nu\overline{b}\,\langle \Delta_1 \cup \{\overline{\alpha}, \overline{n}, \kappa \mapsto \lambda\mathcal{P}\},\, Q_1\rangle\ \mathbb{J}\ \nu\overline{b'}\,\langle \Delta_1' \cup \{\overline{\alpha}, \overline{n}, \kappa \mapsto \lambda\mathcal{P}'\},\, Q_1'\rangle$$

♦ $\eta = \mathsf{app}\,\kappa$: We have:

$$\nu\overline{a}\,\langle \Delta \cup \{\overline{\alpha}, \overline{n}, \kappa \mapsto \lambda\mathcal{P}\},\, Q\rangle \xrightarrow{\mathsf{app}\,\kappa} \nu\overline{a}\,\langle \Delta \cup \{\overline{\alpha}, \overline{n}, \kappa \mapsto \lambda\mathcal{P}\},\, \mathcal{P} \mid Q\rangle$$
$$\nu\overline{a'}\,\langle \Delta' \cup \{\overline{\alpha}, \overline{n}, \kappa \mapsto \lambda\mathcal{P}'\},\, Q'\rangle \xrightarrow{\mathsf{app}\,\kappa} \nu\overline{a'}\,\langle \Delta' \cup \{\overline{\alpha}, \overline{n}, \kappa \mapsto \lambda\mathcal{P}'\},\, \mathcal{P}' \mid Q'\rangle$$

By Theorem 6.28:

$$\nu\overline{a}\,\langle \Delta,\, \mathcal{P} \mid Q\rangle \approx \nu\overline{a'}\,\langle \Delta',\, \mathcal{P}' \mid Q'\rangle$$

and by definition of $\mathbb{J}$:

$$\nu\overline{a}\,\langle \Delta \uplus \{\kappa \mapsto \lambda\mathcal{P}\},\, \mathcal{P} \mid Q\rangle\ \mathbb{J}\ \nu\overline{a'}\,\langle \Delta' \uplus \{\kappa \mapsto \lambda\mathcal{P}'\},\, \mathcal{P}' \mid Q'\rangle$$

(ix) Similar to (viii)

(x) Similar to (viii) considering the relation:

$$\mathbb{J} \stackrel{\text{def}}{=} \{(\nu\overline{a}\,\langle \Delta,\, Q \mid *(\mathcal{P})\rangle,\ \ \nu\overline{a'}\,\langle \Delta',\, Q' \mid *(\mathcal{P}')\rangle) \mid \overline{\alpha}, \overline{n} \in \Delta \text{ and } \nu\overline{a}\,\langle \Delta,\, Q\rangle \approx \nu\overline{a'}\,\langle \Delta',\, Q'\rangle\}$$

□

From the above theorem we conclude that $(\simeq)$ preserves arbitrary contexts.

Theorem 8.5 (Compositionality)**.** *If $P \simeq P'$ then for any context K, $K[P] \simeq K[P']$.*

Proof. We prove that for all $\mathcal{K}$, $\langle \{\overline{\alpha}, \overline{n}\},\, \mathcal{K}[\mathcal{P}]\rangle \approx \langle \{\overline{\alpha}, \overline{n}\},\, \mathcal{K}[\mathcal{P}']\rangle$ by induction on the size of $\mathcal{K}$, using Theorem 8.4. □

We can now show that reduction-closed barbed congruence ($\cong_{\text{cxt}}$) coincides with parallel contextual equivalence ($\cong_{\text{pcxt}}$) and weak bisimilarity ($\simeq$).

Theorem 8.6 (Coincidence of process equivalences)**.** $(\cong_{\text{cxt}}) = (\cong_{\text{pcxt}}) = (\simeq)$.

Proof. By the definitions of $(\cong_{\text{cxt}})$ and $(\cong_{\text{pcxt}})$ and Theorem 7.13 we have

$$(\cong_{\text{cxt}}) \subseteq (\cong_{\text{pcxt}}) \subseteq (\simeq)$$

It remains to show $(\simeq) \subseteq (\cong_{\text{cxt}})$.

In Propositions 6.4 and 6.3 and Theorem 8.5 we have shown that $(\simeq)$ preserves barbs, is reduction-closed, and preserves arbitrary contexts. Thus, $(\simeq)$ is included in the largest relation with these properties, namely $(\cong_{\text{cxt}})$. □

This latter result states that the observational power of arbitrary contexts can be adequately captured by the very restricted class of parallel contexts. It also states that our proof technique is both sound and complete with respect to the touchstone behavioural equivalence ($\cong_{\text{cxt}}$).

9. First-Order Processes

The first-order π-calculus, from Chapter 1 of [22], can be considered to be a sub-language of pp-π; let us refer to this sub-language as fo-π and use p, q to range over closed processes from fo-π. The more standard theory for this sub-language is given in terms of the *standard* LTS in which the nodes are processes and the actions have labels of the form $c?n$ – *input*, $c!n$ – *free output*, $(vn)\,c!n$ – *bound output*, or τ for internal activity; note in particular the use of *extrusion* in the bound outputs. For these first-order (closed) processes we have the following equivalences:

(i) $p \cong_{\text{cxt}} p'$ from Definition 8.2: intuitively this means that the first-order processes p and p' can not be distinguished by any higher-order context.

(ii) $p \simeq p'$ from Definition 4.9: this means that processes p and p' are weakly bisimilar when viewed as (degenerate) configurations in the LTS described in Section 3. Notice that the LTS generated by such first-order configurations only contains actions whose labels take the form $c?n, c!n$, or τ.

(iii) $p \simeq_{fo} p'$: meaning that p and p' are weakly bisimilar in the standard LTS alluded to above, as given in [9, 22].

For the purpose of analysis let us now introduce a fourth [9].

Definition 9.1 (First-order p-contextual equivalence $(\cong_{fo})$)**.** $(\cong_{fo})$ *is the largest relation on closed fo-*π *processes that preserves barbs, is reduction closed, and is preserved by first-order parallel contexts.*

It is known from the literature that $(\simeq_{fo})$ coincides with $(\cong_{fo})$ ([22], Chapter 2); we show that $(\simeq)$ also coincides with $(\cong_{fo})$:

Theorem 9.2. *In fo-*π $p \simeq p'$ *if and only if* $p \cong_{fo} p'$

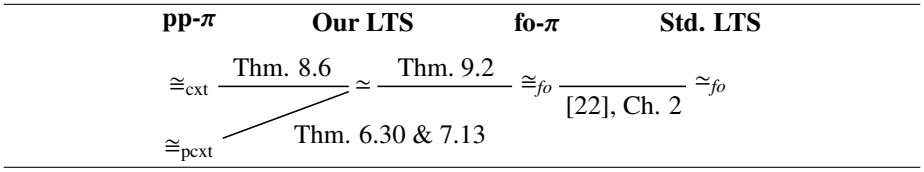


Figure 6: The big picture

Proof. (Outline) A very easy adaptation of Theorems 6.30 and 7.13. Note that for the right-to-left direction we reuse the fo-π contexts for first-order transitions in Theorem 7.13. □

The above theorem completes the link between contextual equivalence in pp-π and weak bisimilarity in the standard LTS for fo-π as shown in Figure 6. We can now derive the following interesting consequences:

Corollary 9.3. *In fo-π,*

(i) $p \simeq p'$ *iff* $p \simeq_{fo} p'$

(ii) $p \simeq_{fo} p'$ *iff* $p \cong_{\text{cxt}} p'$

Result (i) means that the standard bisimulation equivalence ($\simeq_{fo}$) which uses extrusion in output actions, is captured precisely by our extrusion-free bisimulation equivalence ($\simeq$). Result (ii) on the other hand states that our higher-order contextual equivalence ($\cong_{\text{cxt}}$) is a *conservative extension* over the standard bisimulation equivalence ($\simeq_{fo}$) for first-order processes. This latter result has significant implications for verification; if we prove an equivalence between two first-order processes using the first-order theory, this equivalence remains true even when these first-order processes are used in a higher-order setting.

10. Conclusions and Related Work

The main achievement of this paper is a simple and effective proof technique for equivalence for higher-order concurrency. Our technique extends the standard theory of weak bisimulations, and its corresponding Hennessy-Milner Logic to this setting. Previous work on logics for higher-order concurrency [1, 2] aims at characterisations of higher-order bisimilarity and relies on the use of constructive implication.

As discussed in the introduction, our proof technique combines and improves existing theories, particularly those in [10] and [21], and employs a novel treatment of extrusion. Compared to [21, 23], our bisimulations have significantly weaker conditions, do not quantify over input contexts, and do not rely on an up-to context to effectively reason about higher-order processes. This technique is robust with respect to other behavioural theories and languages; for instance we have recently applied this technique to develop a may-testing theory for a higher-order concurrent language with cryptographic primitives [12].

Sangiorgi has given a translation of HOπ with finite types to the π-calculus [19] based on *triggers*, and a full-abstraction proof [18]. This translation, generating a

fresh trigger channel at every output, is possible in pp-π. However it does not scale to languages with more complex types, or to languages where the retransmission of messages is observable, as in cryptographic calculi. An adaptation of the translation where a trigger is generated at every function definition [22, Sec. 13.2] would be incomplete for pp-π because first-order contexts can make more observations than the images of higher-order contexts through the translation [22, pg. 402]. A complex type system for "receptiveness" in the target language has been proposed to prune the problematic contexts, but to the extent of our knowledge the details of such a translation have not been published. Encoding the intuitions of the translation directly in an LTS, as we do in our work, avoids the issues with full-abstraction and scales to more complex types and languages (e.g. [12]).

Symbolic techniques that reduce the quantification over first-order messages have been developed for the spi- and applied π-calculus [3, 5, 6]. Our symbolic treatment addresses the quantification over higher-order, rather than first-order, values.

Laird [14] has developed a games semantic model for HOπ that is fully abstract with respect to may-testing, but is not sound with respect to reduction closed barbed congruence. However we intend to investigate possible connections between our LTS and this game semantics. It is also our intent to use our approach for other behavioural equivalences, such as must testing [4], and higher-order languages with distribution and passivation [15, 17].

Acknowledgements:. We are grateful to Edsko de Vries and Dimitrios Vytiniotis for useful discussions on this work, and to anonymous reviewers for their suggestions for improvement.

[1] Amadio, R.M., Dam, M., 1995. Reasoning about higher-order processes, in: Mosses, P.D., Nielsen, M., Schwartzbach, M.I. (Eds.), Proc. 6th International Joint Conference CAAP/FASE on the Theory and Practice of Software Development, (TAPSOFT'95), Springer. pp. 202–216.

[2] Baldamus, M., Dingel, J., 1997. Modal characterization of weak bisimulation for higher-order processes, in: Bidoit, M., Dauchet, M. (Eds.), Proc. 7th International Joint Conference CAAP/FASE on the Theory and Practice of Software Development (TAPSOFT'97), Springer. pp. 285–296.

[3] Borgström, J., Briais, S., Nestmann, U., 2004. Symbolic bisimulation in the spi calculus, in: Gardner, P., Yoshida, N. (Eds.), Proc. 15th International Conference on Concurrency Theory (CONCUR 2004), Springer. pp. 161–176.

[4] De Nicola, R., Hennessy, M.C.B., 1984. Testing equivalences for processes. Theoretical Computer Science 34, 83–133.

[5] Delaune, S., Kremer, S., Ryan, M.D., 2010. Symbolic bisimulation for the applied pi calculus. J. of Comp. Security 18, 317–377.

[6] Durante, L., Sisto, R., Valenzano, A., 2003. Automatic testing equivalence verification of spi calculus specifications. ACM Trans. Softw. Eng. Methodol. 12, 222–284.

[7] Hennessy, M., 2007. A Distributed Picalculus. Cambridge Univ. Press.

[8] Hennessy, M., Milner, R., 1985. Algebraic laws for nondeterminism and concurrency. J. ACM 32, 137–161.

[9] Honda, K., Yoshida, N., 1995. On reduction-based process semantics. Theor. Comput. Sci. 151, 437 – 486.

[10] Jeffrey, A., Rathke, J., 2005a. Contextual equivalence for higher-order pi-calculus revisited. Logical Methods in Computer Science 1(1:4).

[11] Jeffrey, A., Rathke, J., 2005b. Full abstraction for polymorphic pi-calculus, in: Sassone, V. (Ed.), Proc. 8th International Conference on Foundations of Software Science and Computational Structures, (FOSSACS 2005), ETAPS, Springer. pp. 266–281.

[12] Koutavas, V., Hennessy, M., 2011. A testing theory for a higher-order cryptographic language (extended abstract), in: Barthe, G. (Ed.), Proc. 20th European Symposium on Programming (ESOP 2011), Programming Languages and Systems, Springer. pp. 358–377.

[13] Koutavas, V., Wand, M., 2006. Small bisimulations for reasoning about higher-order imperative programs, in: Morrisett, J.G., Jones, S.L.P. (Eds.), Proc. 33rd ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, POPL 2006, ACM. pp. 141–152.

[14] Laird, J., 2006. Game semantics for higher-order concurrency, in: A.-K., S., Garg, N. (Eds.), Proc. Foundations of Software Technology and Theoretical Computer Science (FSTTCS 2006), Springer. pp. 417–428.

[15] Lenglet, S., Schmitt, A., Stefani, J.B., 2011. Characterizing contextual equivalence in calculi with passivation. Information and Computation 209, 1390 – 1433.

[16] Milner, R., 1989. Communication and Concurrency. Prentice-Hall.

[17] Piérard, A., Sumii, E., 2011. Sound bisimulations for higher-order distributed process calculus, in: Hofmann, M. (Ed.), Proc. 14th International Conference on Foundations of Software Science and Computational Structures (FOSSACS 2011), ETAPS. Springer. volume 6604 of *Lecture Notes in Computer Science*, pp. 123–137.

[18] Sangiorgi, D., 1992. Expressing mobility in process algebras: first-order and higher-order paradigms. PhD thesis CST–99–93. Dept. of Computer Science, Univ. of Edinburgh.

[19] Sangiorgi, D., 1993. From pi-calculus to higher-order pi-calculus – and back, in: Gaudel, M.C., Jouannaud, J.P. (Eds.), Proc. Theory and Practice of Software Development (TAPSOFT 1993), International Joint Conference CAAP/FASE, Springer. pp. 151–166.

[20] Sangiorgi, D., 1996. Bisimulation for higher-order process calculi. Information and Computation 131, 141–178.

[21] Sangiorgi, D., Kobayashi, N., Sumii, E., 2007. Environmental bisimulations for higher-order languages, in: Proc. 22nd IEEE Symposium on Logic in Computer Science (LICS 2007), IEEE Computer Society. pp. 293–302.

[22] Sangiorgi, D., Walker, D., 2001. The π-calculus: a theory of mobile processes. Cambridge Univ. Press.

[23] Sato, N., Sumii, E., 2009. The higher-order, call-by-value applied pi-calculus, in: Hu, Z. (Ed.), Proc. 7th Asian Symposium (APLAS 2009), Programming Languages and Systems, Springer. pp. 311–326.

[24] Stirling, C., 1999. Bisimulation, modal logic and model checking games. Logic Journal of the IGPL 7, 103–124.

[25] Sumii, E., Pierce, B.C., 2004. A bisimulation for dynamic sealing, in: Proc. 31st ACM SIGPLAN-SIGACT symposium on Principles of programming languages (POPL 2004), ACM. pp. 161–172.

[26] Sumii, E., Pierce, B.C., 2005. A bisimulation for type abstraction and recursion, in: Proc. 32nd ACM SIGPLAN-SIGACT symposium on Principles of programming languages (POPL 2005), ACM. pp. 63–74.

[27] Sumii, E., Pierce, B.C., 2007a. A bisimulation for dynamic sealing. TCS 375, 169–192. (extended abstract appeared in [25]).

[28] Sumii, E., Pierce, B.C., 2007b. A bisimulation for type abstraction and recursion. J. ACM 54, 1–43. (extended abstract appeared in [26]).