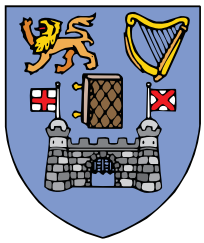


TRINITY COLLEGE DUBLIN
COLÁISTE NA TRÍONÓIDE, BAILE ÁTHA CLIATH

First-Order Reasoning for Higher-Order Concurrency

Vasileios Koutavas *Matthew Hennessy*

Computer Science Department Technical Report **TCS-CS-YYYY-NN**
Foundations and Methods Research Group **July 23, 2009**

First-Order Reasoning for Higher-Order Concurrency*

Vasileios Koutavas Matthew Hennessy
{Vasileios.Koutavas, Matthew.Hennessy}@cs.tcd.ie

Trinity College Dublin
July 23, 2009

Abstract

By combining and simplifying two of the most prominent theories for HOπ of Sangiorgi et al. and Jeffrey and Rathke [15, 4], we present an effective first-order theory for a higher-order *picalculus*.

There are two significant aspects to our theory. The first is that higher-order inputs are treated in a first-order manner, hence eliminating the need to reason about arbitrarily complicated higher-order contexts, or to use up-to context techniques, when establishing equivalences between processes. The second is that we use augmented processes to record directly the knowledge of the observer. This has the benefit of making ordinary first-order weak bisimulation fully abstract w.r.t. contextual equivalence. It also simplifies the handling of names, giving rise to a truly propositional Hennessy-Milner characterisation of higher-order contextual equivalence.

Furthermore, we illustrate the simplicity of our approach in proving several interesting equivalences by exhibiting first-order witness weak bisimulations, and inequivalences by 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 *picalculus*.

Contents

*The financial support of SFI is gratefully acknowledged

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

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

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

1 Introduction

Developing effective reasoning techniques for languages with higher-order constructs is a challenging problem, made even more challenging with the presence of concurrency and mobility. 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 *picalculus*, [11, 12, 13, 15, 16]. The *picalculus* is an abstract process description language in which first-order values are exchanged between sub-processes using communication channels. Much of its power derives from the fact that these values include the set of communication channels, which can be generated dynamically and shared privately among sub-processes. Higher-order versions also allow processes, or some form of abstractions over processes, to be communicated; that is the values communicated may be higher-order.

Intuitively two processes are deemed to be behaviourally equivalent if no user in any context can distinguish between them [10]; this has been formalised for a wide range of languages, including the higher-order *picalculus*, called HOπ, and is often referred to as *contextual equivalence* [3, 4, 5]; a slight variation is called *barbed congruence* [16, 12, 11]. The challenge is to develop useful reasoning techniques for this contextual equivalence, in particular techniques which do not involve reasoning about all possible programming contexts.

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 the X and Y in a slightly different way into two possible replies on channel c, chosen nondeterministically. After a reply is sent, all it 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. These include replication and execution of the values, as well as their embedding in more complex values fed into other instances of the same processes.

Our aim is to develop a practical and effective reasoning methodology for such processes. Our methodology will only employ first-order reasoning and we will see that the examples in Figure 1 can be handled in a straightforward manner.

The theory of *bisimulations* [9, 14] has proved to be a powerful operational technique for establishing contextual equivalences in a variety of settings. Briefly the bisimulation-based proof technique involves interrogating processes in a game-theoretic manner [17] using *actions* which represent the different ways in which a user can interact with a process. For

the *picalculus* and related languages the relevant actions take the form

$$\text{output: } P \xrightarrow{(\nu\overline{n})\, c!V} P' \qquad \text{input : } P \xrightarrow{(\nu\overline{n})\, c?V} P' \tag{1}$$

Here P is the process under investigation, P' the result of the investigation, c a communication channel, V an acceptable value which may be higher-order, and $\overline{n}$ represents the new information *extruded* by the process under investigation to the interrogator. These actions are defined using a Labelled Transition System (LTS), and then roughly speaking P and Q are deemed equivalent whenever they can support the same possible interrogations.

A number of different formulations of bisimulations have been developed for HOπ including *normal bisimulation* [11, 12], and its abstract version in [4], and *environmental bisimulations* [15]. All have been shown to coincide with *contextual bisimulation* [13] (often referred to as being *fully-abstract*) but we believe that none have provided a satisfactory applicable proof technique for program equivalence; as evidence we can cite the near complete lack of examples in the research literature. The aim of the current paper is to design an elementary bisimulation theory which is not only *fully-abstract* with respect to contextual equivalence but also supports a reasonable proof methodology, one in which program equivalence can be readily exhibited. By a careful selection of ideas from [4] and [15], together with the combination and simplification of their formalisms, we will end up with a purely first-order theory of bisimulations in which actions take a particularly simple form, and in which witness bisimulations underlying equivalences between higher-order processes are very easy to describe.

Sangiorgi et al. [15], motivated by work in sequential languages [19, 18, 7, 6, 8], use an LTS with the actions in (1) above 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 *related*, and not just identical, input values.

Let us briefly explain this point. Knowledge about two potentially bisimilar processes is accumulated by the interrogator by storing all (higher-order) values output by the processes in the knowledge environment. This knowledge can subsequently be used to construct arbitrarily complicated new values with which to further interrogate the processes. It is this generation of arbitrary new values, using knowledge previously gleaned from the process, which accounts for the requirement to quantify over all related contexts. This strong proof obligation is sometimes mitigated by the use of up-to context techniques.

Jeffrey and Rathke [4] use a more restrictive approach of *formal triggers*. 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. Hence our theory combines and simplifies their insights. We use knowledge environments in the LTS that record the values exposed to the context, and test related processes with symbolic higher-order inputs. In this way we recover a representation of triggers similar to [4], together with an explicit record of the names that are known to the context.

We also take this one step further by including an explicit representation of the information known only to the process. Thus here configurations take the form $\nu\overline{a}\,\langle\Delta,\, P\rangle$ which consists of

- the (higher-order) process under interrogation P,
- a representation Δ of the knowledge currently known to the interrogator about this process,
- 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. Thus the actions in (1) above, now applying to configurations, are labelled simply $c!v$ and $c?v$ respectively, 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) [2], that would not be possible by using Jeffrey and Rathke's LTS.

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 *picalculus*, 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, which the literature of higher-order concurrency currently lacks.

(iii) We give the first propositional HML characterisation of weak bisimulation for a higher-order *picalculus*; this result easily transfers to the first-order *picalculus*. 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 *picalculus*, thus confirming that results and reasoning methods from first-order *picalculus* transfer to a higher-order setting.

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. Sections 5 and 6 contain the proofs of soundness and completeness of our theory with respect to contextual equivalence that preserves only parallel contexts, and Section 7 proves that our theory is fully abstract with respect to the full contextual equivalence. Section 8 is devoted to proving several interesting equivalences by using weak bisimulations, and an inequivalence by providing a discriminating HML formula. Section 9 proves the conservativity theorem; the paper closes in Section 10 with conclusions and a discussion of related work.

2 The Language

2.1 Syntax

We study the language pp-π (process-passing-π), a higher-order version of the *picalculus* whose syntax is given in Figure 2. This is an extension of the *picalculus* which allows processes to be communicated and is roughly equivalent to the language studied in [13]. We

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

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

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

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

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

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

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

Figure 2: Syntax and typing of pp-π

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 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 *picalculus*: 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 *picalculus* 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 $\mathtt{app}\, V$.

We employ the usual abbreviations and notations from the *picalculus*. We identify alpha-equivalent processes and we write $P \mid \ldots \mid P$ (n parallel processes P) as $\prod_n P$. Moreover, $fv(P)$ denotes the set of variables occurring free in P, while $fn(P)$ denotes its set of free names. We are primarily interested in *closed* process terms, those satisfying $fv(P) = \emptyset$; these we refer to as *processes*. The standard notions of substitution are employed, denoted by $P\{a/b\}$ and $P\{V/x\}$; the latter may give rise to ill-formed terms but this is avoided by the use of types. We write $a?.P$ instead of $a?(x{:}\mathtt{Nm}).P$, when $x \notin fv(P)$, and $a!.P$ instead of $a!\langle a{:}\mathtt{Nm}\rangle.P$. Moreover, we omit type annotations on input and output when they are uniquely determined by the context.

$\boxed{P \to Q}$

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

APP-RED
$$\frac{}{\mathtt{app}\,\lambda P \to P}$$

PAR-RED
$$\frac{P_1 \to P_2}{P_1 \mid Q \to P_2 \mid Q}$$

NU-RED
$$\frac{P_1 \to P_2}{\nu a.\, P_1 \to \nu a.\, P_2}$$

CONG-RED
$$\frac{P_1' \to P_2' \qquad P_1 \equiv P_1' \qquad P_2 \equiv P_2'}{P_1 \to P_2}$$

COND-TRUE-RED
$$\frac{}{\mathtt{if}\,a = a\,\mathtt{then}\,P\,\mathtt{else}\,Q \to P}$$

COND-FALSE-RED
$$\frac{a \neq b}{\mathtt{if}\,a = b\,\mathtt{then}\,P\,\mathtt{else}\,Q \to Q}$$

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 types 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.

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

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

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.

Notice that we do not assume any static typing for channel names. At different points in time they may be used to communicate values of different type. So, for example, a client using this service at s will be expected to follow an implicit protocol, whereby first a name is sent on s and then a process.

2.2 Reduction semantics

This 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,

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

The remaining rules are standard, borrowed from the *picalculus*; 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.

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 contextual equivalence [4, 5, 16, 12, 11] 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* $P \to^* Q$ *and* $Q \downarrow_b$.

Definition 2.7 (Parallel Contextual Equivalence ($\cong_{\mathrm{pcxt}}$))**.** ($\cong_{\mathrm{pcxt}}$) *is the largest relation on closed processes that preserves barbs, is reduction closed, and is preserved by parallel contexts; i.e.* $P \cong_{\mathrm{pcxt}} P'$ *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* $P \to P_1$ *there exists* P'_1 *such that* $P' \to^* P'_1$ *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 (3) 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 [17].

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) *picalculus* 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

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

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

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

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

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

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

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

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

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

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

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

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)

higher-order values are simply *abstract* variables, ranged over by α, taken from a countable set AVariables, which is assumed to be disjoint from the ordinary program variables Variables. 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; $\mathit{fav}(\mathcal{P})$ denotes the set of abstract variables 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 variables:

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

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* variables CVariable, disjoint from other kinds of variables, and ranged over by κ.

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

$$\mathsf{Name} \cup \mathsf{AVariable} \cup (\mathsf{CVariable} \rightarrow_{\mathrm{fin}} \mathsf{AValue})$$

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

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

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

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

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

Par-Fst-Trans
$$\frac{\nu\overline{a}\,\langle\Delta_1,\,\mathcal{P}_1\rangle \xrightarrow{\eta} \nu\overline{b}\,\langle\Delta_2,\,\mathcal{P}_2\rangle}{\nu\overline{a}\,\langle\Delta_1,\,\mathcal{P}_1 \mid Q\rangle \xrightarrow{\eta} \nu\overline{b}\,\langle\Delta_2,\,\mathcal{P}_2 \mid Q\rangle}$$

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

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

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)

We write $\Delta[e]$ for $\Delta \cup \{e\}$, and $\Delta(\kappa) = \mathcal{V}$ for $(\kappa, \mathcal{V}) \in \Delta$; we also write *names*$(\Delta)$, *avars*$(\Delta)$, and *cvars*$(\Delta)$ for the name component, the abstract variable 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*(Δ) *and* *fav*$(\mathcal{P}) \subseteq$ *avars*(Δ)

(iv) $\vdash \mathcal{V} : \mathtt{Pr}$ *and* *fn*$(\mathcal{V}) \subseteq \{\overline{a}\} \cup$ *names*(Δ) *and* *fav*$(\mathcal{V}) \subseteq$ *avars*(Δ) *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 (i); 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 variables in Δ.

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). They weakly correspond to the reduction 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 might already be known to the observer or is freshly generated—in both cases n is in the knowledge of the observer after the transition.

(iii) *Higher-order input, $c?\alpha$:* input by the process of an abstract higher-order name α. For these actions, α is always taken to be *fresh*; that is a new abstract name, implicitly freshly generated by the observer.

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

(v) *Higher-order output, $c!\kappa$:* output by the process of some value along the channel c. The channel c must be known to the observer, but κ is fresh. Here the actual value output by the process is not represented directly in the label. Instead it is stored in the the knowledge environment Δ under the fresh key κ.

(vi) *Abstract value application, $\mathtt{app}\,\alpha$:* the execution by the process of the abstract higher-order value α supplied by the observer. But since this is an abstract value, this execution 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. This value was originally supplied by the process.

Rules Name-In-Trans and Proc-In-Trans in Figure 4 capture first-order and higher-order input of the process, respectively. In the former the observer provides to the process a known or fresh name over a known channel. In the latter the observer provides a new abstract variable, representing an arbitrary higher-order value, over a known channel.

Similarly, Name-In-Trans and Proc-In-Trans capture first-order and higher-order output of the process. In the latter, the higher-order value is not sent to the observer. Instead, it is indexed by a new concrete variable, and that variable is sent to the observer.

Rules Proc-In-Trans and Proc-Out-Trans include a freshness condition for the involved abstract and concrete variable, respectively, to ensure that abstract variables from distinct input transitions and concrete variables from distinct output transitions are not confused.

Internal communication is captured by Comm-Name-Trans, for first-order values, and Comm-Proc-Trans for higher-order values. 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 variable κ used to store the value being communicated is not included in the environment Δ in the conclusion.

Application of higher-order values is encoded in the rules Conc-App-Trans and Abs-App-Trans. The former says that for the observer to run a value originally supplied by the process, it simply executes it in parallel with the process currently under observation. In contrast, rule Abs-App-Trans says that the effect of the process executing an abstract value originally supplied by the observer is simply a signal to the observer (via the label of the transition) and the generation of the empty process $\mathtt{0}$.

The rest of the rules in Figure 5 are mostly house-keeping in nature.

The LTS preserves well-formedness of configurations.

Proposition 3.3. *Suppose $\nu\overline{a}\,\langle\Delta_1, \mathcal{P}\rangle$ is well-formed and $\nu\overline{a}\,\langle\Delta_1, \mathcal{P}\rangle \xrightarrow{\eta} \nu\overline{b}\,\langle\Delta_2, \mathcal{Q}\rangle$. Then*

(i) $\Delta_1 \subseteq \Delta_2$

(ii) $\nu\overline{b}\,\langle \Delta_2, \mathcal{Q}\rangle$ *is well-formed.*

Proof. By induction on the transition $\nu\overline{a}\,\langle \Delta_1, \mathcal{P}\rangle \xrightarrow{\eta} \nu\overline{b}\,\langle \Delta_2, \mathcal{Q}\rangle$. □

For the remainder of this paper we consider only well-formed configurations.

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* $\mathcal{P}_2$

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

(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* $\mathcal{P}_1$ *and* $\mathcal{P}_2$

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

(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* $\mathcal{P}_2$

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

(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* $\mathcal{P}_2$

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

(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* $\mathcal{P}_1$

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

(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* $\mathcal{V} = \Delta_1(\kappa)$

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

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

Proof. All properties are shown by induction on the rules of the LTS. □

In a configuration $\nu\overline{a}\,\langle \Delta, \mathcal{P}\rangle$ there are two sources of knowledge, the environments 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 3.5.

(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* $names(\eta) \cap names(\Delta_0) = avars(\eta) \cap avars(\Delta_0) = cvars(\eta) \cap cvars(\Delta_0) = \emptyset$ *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* $\nu\overline{a}, \overline{c}\,\langle \Delta_0 \uplus \Delta_1, \mathcal{P}\rangle \xrightarrow{\eta} \nu\overline{b}\,\langle \Delta_0 \uplus \Delta_2, \mathcal{Q}\rangle$ *and* $\nu\overline{a}\,\langle \Delta_1, \mathcal{P}\rangle$ *is well-formed, and* $\eta \notin \{\mathsf{app}\,\kappa \mid \kappa \in \Delta_0\} \cup \{\, c?n \mid n \in \Delta_0\}$ *then*

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

where $\{\overline{b'}\} = \{\overline{b}\}\backslash\{\overline{c}\}$.

Proof. In both cases we use rule induction on the inference of the actions. □

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 3.6 (Unused Information)**.**

(i) Hiding: *Suppose* $\nu\overline{a}\,\langle \Delta_1[b], \mathcal{P}\rangle \xrightarrow{\eta} \nu\overline{d}\,\langle \Delta_2[b], \mathcal{Q}\rangle$ *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*

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

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 $\nu b, \overline{a}\,\langle \Delta, \mathcal{P}\rangle \xrightarrow{c!b} \nu\overline{d}\,\langle \Delta[b], \mathcal{Q}\rangle$; and this action can still be performed when the observer knows of the existence of b:

Proposition 3.7 (Extrusion)**.** *Provided c is different than b,*

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

Proof. By rule induction, in both directions. □

Abstract variables are significant only in application and communication actions that mention them—substituting a value for an abstract variable leaves all other actions unaffected. Similarly, values are significant only in application steps—abstracting away values leaves other actions unaffected.

Proposition 3.8.

(i) Substitution: *Suppose* $\nu\overline{a}\,\langle \Delta_1, \mathcal{P}\rangle \xrightarrow{\eta} \nu\overline{b}\,\langle \Delta_2, \mathcal{Q}\rangle$ *and* $\nu\overline{a}\,\langle \Delta_1\{\mathcal{V}/\alpha\}, \mathcal{P}\{\mathcal{V}/\alpha\}\rangle$ *is well-formed and* α *does not occur in* η*. Then*

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

(ii) Abstraction: *Let* $\nu\overline{a}\,\langle \Delta_1\{\mathcal{V}/\alpha\}, \mathcal{P}\{\mathcal{V}/\alpha\}\rangle \xrightarrow{\eta} \nu\overline{b}\,\langle \Delta_2\{\mathcal{V}/\alpha\}, \mathcal{Q}\{\mathcal{V}/\alpha\}\rangle$ *is well-formed and* η *is not a* τ *action involving the rule* App-Trans *or an* $\mathsf{app}\,\alpha$ *action. Then*

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

Proof. Both properties are shown by rule induction. □

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*

$$C' \xrightarrow{\eta} C'_1 \qquad C_1\ \mathbb{R}\ C'_1$$

(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. □

4.2 Weak Bisimulations

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

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*

$$C' \stackrel{\eta}{\Longrightarrow} C'_1 \qquad C_1\ \mathbb{R}\ C'_1$$

(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.*

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

Proof (by contradiction). Let $\kappa \in cvars(\Delta)$ and $\kappa \notin cvars(\Delta')$; then $\nu\overline{a}\,\langle\Delta, \mathcal{P}\rangle$ has an $\mathsf{app}\,\kappa$-transition to another configuration but $\nu\overline{a'}\,\langle\Delta', \mathcal{P}'\rangle$ does not, which contradicts the premise. □

We extend weak bisimilarity to closed processes by the following definition.

Definition 4.7. *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 4.8 (($\equiv$) 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 4.9. ($\equiv$) *is a weak bisimulation over configurations.*

Proof (sketch). Suppose

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

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 $\nu\overline{b'}\,\langle\Delta_2', \mathcal{Q}'\rangle \equiv \nu\overline{b}\,\langle\Delta_2, \mathcal{Q}\rangle$.

We proceed by induction on the proof that $\nu\overline{a}.\mathcal{P} \equiv \nu\overline{a'}.\mathcal{P}'$. There are three cases;

(i) The vector $\overline{a'}$ is empty, so that $\nu\overline{a}.\mathcal{P} \equiv \mathcal{P}'$ Here we proceed by induction on the size of the vector $\overline{a}$, with this base case being when it is empty and thus we have $\mathcal{P} \equiv \mathcal{P}'$. We continue here by induction on the proof of this equivalence.

With this inner induction the base case is provided by the axioms for ($\equiv$) in Figure 3, with as usual the extrusion axiom $\nu a.\,(P \mid Q) \equiv (\nu a.\,P) \mid Q$ whenever $(a \notin fn(Q))$, being somewhat complex. There are two inductive cases within this inner induction, when the structural equivalent terms are composed parallel operator $\mid$ and $\nu a.-$ respectively are used. The former makes extensive use of Proposition 3.6 and is non-trivial. The latter is straightforward since the only moves from the configuration $\langle\Delta, \nu a.\,\mathcal{P}\rangle$ are those generated by the rule Nu-Trans.

Having finished the base case for the outer induction, we have one inductive case, when $\overline{a}$ has the form $d\overline{c}$, the process $\mathcal{P}'$ has the form $\nu d.\,\mathcal{P}_1'$ and by induction we know $\nu\overline{c}.\mathcal{P} \equiv \mathcal{P}_1'$. Here the proof proceeds by case analysis on η; if it involves d then Proposition 3.7 is used and otherwise Proposition 3.6 is employed.

(ii) $\mathcal{P} \equiv \nu\overline{a'}.\mathcal{P}'$ This case is similar to (i) and omitted.

(iii) $\nu d\overline{a}.\mathcal{P} \equiv \nu d\overline{a'}.\mathcal{P}'$ because, by induction $\nu\overline{a}.\mathcal{P} \equiv \nu\overline{a'}.\mathcal{P}'$ Here the proof is similar to the outer inductive step of (i), with a case analysis on whether or not η uses the name d.

□

Corollary 4.10. $(\equiv) \subseteq (\simeq)$.

Extending bisimilar configurations with identical names produces bisimilar configurations.

Lemma 4.11. *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*

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

Proof. Let

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

It is easy to show that $\mathbb{X}$ is a weak bisimulation using Proposition 3.5. □

Lemma 4.12. *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. □

Lemma 4.13. $\nu a, \overline{b}\,\langle \Delta, \mathcal{P}\rangle \approx \nu\overline{b}\,\langle \Delta, \nu a.\mathcal{P}\rangle$

Proof. Trivial. □

Lemma 4.14.

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

Proof. By Lemmas 4.13 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 4.15 (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.:

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

As the transition labels η in our LTS contain actual (not extruded) names, the above logic is similar to that of the CCS ([9], 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 4.16. *$C \approx C'$ if and only if for all F*

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

Proof. For the forward direction we first define the ordinal size of HML formulas:

$$\begin{array}{rcl} \textit{size}(\neg F) & \stackrel{\text{def}}{=} & 1 + \textit{size}(F) \\ \textit{size}(\bigwedge_{i\in I} F_i) & \stackrel{\text{def}}{=} & 1 + \Sigma_{i\in I}\textit{size}(F_i) \\ \textit{size}(\langle\eta\rangle F) & \stackrel{\text{def}}{=} & 1 + \textit{size}(F) \end{array}$$

We proceed by ordinal induction, using the induction hypothesis

$$IH(k) = \forall C, C', F.\ (C \approx C' \wedge \textit{size}(F) \leq k) \ \textit{ implies } \ (C \models F \textit{ iff } C' \models F)$$

Case $k = 0$: vacuously true because all formulas have size greater than 0.

Case $k > 0$: we assume that $IH(k-1)$ holds, and consider C, C', and F with $C \approx C'$ and $\textit{size}(F) = k$. We will show that

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

We proceed by cases on F:

♦ $F = \neg F'$: it must be that $\textit{size}(F') = k - 1$. By $IH(k-1)$,

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

and by Definition 4.15 we get

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

♦ $F = \bigwedge_{i\in I} F_i$: We proceed by cases on I:

• $I = \emptyset$: trivially, by Definition 4.15, $C \models F$ and $C' \models F$.

• $I = I' \uplus \{j\}$: by Definition 4.15,

$$C \models \bigwedge_{i \in I} F_i \;\; \textit{iff} \;\; (C \models F_j \wedge C \models \bigwedge_{i \in I'} F_i)$$
$$C' \models \bigwedge_{i \in I} F_i \;\; \textit{iff} \;\; (C' \models F_j \wedge C' \models \bigwedge_{i \in I'} F_i)$$

By the definition of *size*, it must be that for all $i \in I$, $0 < size(F_i) < k$, and therefore $size(F_j) \leq k-1$ and $size(\bigwedge_{i \in I'} F_i) \leq k-1$. Thus, by $IH(k-1)$, we get

$$C \models F_j \textit{ iff } C' \models F_j$$
$$C \models \bigwedge_{i \in I'} F_i \textit{ iff } C' \models \bigwedge_{i \in I'} F_i$$

and by easy propositional reasoning $C \models F$ *iff* $C' \models F$.

♦ $F = \langle \eta \rangle F'$: it must be that $size(F') = k-1$.

If $C \models \langle \eta \rangle F'$ then, by Definition 4.15, there exists C_1 such that

$$C \stackrel{\eta}{\Longrightarrow} C_1 \qquad C_1 \models F'$$

Because $C \approx C'$, there exists C'_1 such that

$$C' \stackrel{\eta}{\Longrightarrow} C'_1 \qquad C_1 \approx C'_1$$

By $IH(k-1)$ it must be that $C'_1 \models F'$, and by Definition 4.15 $C' \models \langle \eta \rangle F'$. Similarly if $C' \models \langle \eta \rangle F'$.

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

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

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 8.5 we prove such an inequivalence by providing an HML formula that is satisfied by one of the processes and not the other.

5 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 three 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. The second sub-section proves that ($\simeq$) is reduction-closed and preserves barbs, while in the final sub-section is devoted to the most difficult property, preservation by parallel contexts.

5.1 Reductions versus τ-steps

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

Lemma 5.1. *If* $\nu\overline{a}\,\langle\{\overline{c}\},\, P\rangle \xrightarrow{\tau} \nu\overline{b}\,\langle\{\overline{c}\},\, Q\rangle$ *then* $\nu\overline{a}.\,P \rightarrow^* \nu\overline{b}.\,Q$

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-Fst-Trans: we have

$$\frac{\nu\overline{a}\,\langle\{\overline{c}\},\, P_1\rangle \xrightarrow{\tau} \nu\overline{b}\,\langle\{\overline{c}\},\, P_2\rangle}{\nu\overline{a}\,\langle\{\overline{c}\},\, P_1 \mid Q\rangle \xrightarrow{\tau} \nu\overline{b}\,\langle\{\overline{c}\},\, P_2 \mid Q\rangle}$$

and want to show that $\nu\overline{a}.\,P_1 \mid Q \rightarrow^* \nu\overline{b}.\,P_2 \mid Q$. By the induction hypothesis $\nu\overline{a}.\,P_1 \rightarrow^* \nu\overline{b}.\,P_2$, and by Proposition 3.4 (vii) $\{\overline{a}\} \subseteq\{\ \overline{b}\}$. Hence, by the properties of reduction, $P_1 \rightarrow^* \nu\overline{a'}.\,P_2$, where $\overline{b} = \overline{a}, \overline{a'}$. By Par-Red, $P_1 \mid Q \rightarrow^* (\nu\overline{a'}.\,P_2) \mid Q$. Because $\nu\overline{a}\,\langle\{\overline{c}\},\, P_1 \mid Q\rangle$ is well-formed, $fn(Q) \subseteq \{\overline{a}, \overline{c}\}$ and $\{\overline{a'}\} \cap\{\ \overline{a}, \overline{c}\} = \emptyset$, and hence, by Cong-Red and Nu-Red, $\nu\overline{a}.\,(P_1 \mid Q) \rightarrow^* \nu\overline{b}.\,(P_2 \mid Q)$.

Case Comm-Name-Trans: we have

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

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

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

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

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

Lemma 5.2. *If* $P \rightarrow Q$, *and* $fn(P) \subseteq \{\overline{c}\}$ *then there exist* $\overline{a}$ *and* Q_0 *such that*

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

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

Case Par-Red: we have

$$\frac{P_1 \to P_2}{P_1 \mid Q \to P_2 \mid Q}$$

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

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

Case Nu-Red: we have

$$\frac{P \to Q}{\nu b.\, P \to \nu b.\, Q}$$

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 3.6 (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

$$\frac{P' \to Q' \qquad P \equiv P' \qquad Q \equiv Q'}{P \to Q}$$

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 4.9 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 $\nu\overline{a}.\, Q_0 \equiv \nu\overline{a'}.\, Q_0' \equiv Q' \equiv Q$. □

5.2 Reduction-closure and preservation of barbs

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

Proposition 5.3 (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 4.7, 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 Lemma 5.2 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 Lemma 5.1, $P' \rightarrow^* \nu\overline{a'}.\, Q'_0$.

By (3) and Lemma 4.14 $\langle\{\overline{b}\}, \nu\overline{a}.\, Q_0\rangle \approx \langle\{\overline{b}\}, \nu\overline{a'}.\, Q'_0\rangle$ 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 4.10 we get $Q \simeq \nu\overline{a'}.\, Q'_0$. □

Proposition 5.4 (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 $P \rightarrow^* Q$, $Q \equiv \nu\overline{a}.\, n!\langle V{:}t\rangle.\, Q_1 \mid Q_2$, and $n \notin \{\overline{a}\}$. By Proposition 5.3 there exists Q' such that $P' \rightarrow^* Q'$ and $Q \simeq Q'$.

By the first premise, transitivity of $(\simeq)$, and Corollary 4.10 we get that $\nu\overline{a}.\, n!\langle V{:}t\rangle.\, Q_1 \mid Q_2 \simeq Q'$. Thus, by Definition 4.7, there exist $\overline{b}$ such that

$$\langle\{\overline{b}, n\}, \nu\overline{a}.\, n!\langle V{:}t\rangle.\, Q_1 \mid Q_2\rangle \approx \langle\{\overline{b}, n\}, Q'\rangle \tag{1}$$

and by the transition rules of the LTS we get

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

if $t = \mathtt{Nm}$ or

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

if $t = \mathtt{Pr}$. By Definition 4.4 and (1), there exist $\overline{a'}$, Δ', and Q'_1 such that one of the following is true:

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

or

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

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

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

and by Lemma 5.1 $Q' \rightarrow^* \nu\overline{a'}.\, n!\langle V'{:}t\rangle.\, Q'_1 \mid Q'_2$ with $n \notin \{\overline{a'}\}$. Hence $P' \Downarrow_n$. □

5.3 Parallel Contexts

Here our intention is to show that

$$P \simeq P' \ \ \textit{implies} \ \ P \mid Q \simeq P' \mid Q \tag{*}$$

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

$$\textsc{Cxt-R}\quad \frac{\nu\overline{a}\,\langle \Delta, \mathcal{P}\rangle\ \mathbb{R}\ \nu\overline{a'}\,\langle \Delta', \mathcal{P}'\rangle \quad avars(\Delta) = avars(\Delta') \quad names(\Delta) = names(\Delta')}{\nu\overline{a}\,\langle \Delta, \mathcal{P}\rangle\ \mathbb{R}^{\mathsf{cxt}}\ \nu\overline{a'}\,\langle \Delta', \mathcal{P}'\rangle}$$

$$\textsc{Cxt-Par}\quad \frac{\nu\overline{a}\,\langle \Delta_1, \mathcal{P}_1\rangle\ \mathbb{R}^{\mathsf{cxt}}\ \nu\overline{a'}\,\langle \Delta'_1, \mathcal{P}'_1\rangle \quad \nu\overline{b}\,\langle \Delta_2, \mathcal{P}_2\rangle\ \mathbb{R}^{\mathsf{cxt}}\ \nu\overline{b'}\,\langle \Delta'_2, \mathcal{P}'_2\rangle}{\nu\overline{a}, \overline{b}\,\langle \Delta_1 \cup \Delta_2, \mathcal{P}_1 \mid \mathcal{P}_2\rangle\ \mathbb{R}^{\mathsf{cxt}}\ \nu\overline{a'}, \overline{b'}\,\langle \Delta'_1 \cup \Delta'_2, \mathcal{P}'_1 \mid \mathcal{P}'_2\rangle}$$

$$\textsc{Cxt-Hide}\quad \frac{\nu\overline{a}\,\langle \Delta \uplus \{n\}, \mathcal{P}\rangle\ \mathbb{R}^{\mathsf{cxt}}\ \nu\overline{a'}\,\langle \Delta' \uplus \{n\}, \mathcal{P}'\rangle}{\nu\overline{a}, n\,\langle \Delta, \mathcal{P}\rangle\ \mathbb{R}^{\mathsf{cxt}}\ \nu\overline{a'}, n\,\langle \Delta', \mathcal{P}'\rangle}$$

$$\textsc{Cxt-Subst}\quad \frac{\nu\overline{a}\,\langle \Delta \uplus \{\alpha, k{\mapsto}\mathcal{V}\}, \mathcal{P}\rangle\ \mathbb{R}^{\mathsf{cxt}}\ \nu\overline{a'}\,\langle \Delta' \uplus \{\alpha, k{\mapsto}\mathcal{V}'\}, \mathcal{P}'\rangle \quad \alpha \notin fav(\mathcal{V}, \mathcal{V}')}{\nu\overline{a}\,\langle \Delta\{\mathcal{V}/\alpha\}, \mathcal{P}\{\mathcal{V}/\alpha\}\rangle\ \mathbb{R}^{\mathsf{cxt}}\ \nu\overline{a'}\,\langle \Delta'\{\mathcal{V}'/\alpha\}, \mathcal{P}'\{\mathcal{V}'/\alpha\}\rangle}$$

Figure 6: Parallel Context Closure.

Definition 5.5 (Parallel Context Closure of a Relation)**.** *If $\mathbb{R}$ is a relation on well-formed configurations of the LTS then $\mathbb{R}^{\mathsf{cxt}}$ is the smallest relation on well-formed configurations satisfying the rules of Figure 6.*

Here Cxt-Par is the most significant closure property. We aim to show that $(\approx)^{\mathsf{cxt}}$ is contained in $(\approx)$, from which (*) will follow.

Lemma 5.6. *If $\nu\overline{a}\,\langle \Delta, \mathcal{P}\rangle\ \mathbb{R}^{\mathsf{cxt}}\ \nu\overline{a'}\,\langle \Delta', \mathcal{P}'\rangle$ then*

$$avars(\Delta) = avars(\Delta') \qquad names(\Delta) = names(\Delta')$$

Proof. By a straightforward induction on the rules of Figure 6. □

Theorem 5.7 (Parallel Context Closure of $(\approx)$)**.** $(\approx)^{\mathsf{cxt}} \subseteq (\approx)$.

Proof. By induction on the rules of Fig. 6.

Case Cxt-R: immediate.

Case Cxt-Par: the induction hypothesis gives

$$\nu\overline{a}\,\langle \Delta_1, \mathcal{P}_1\rangle \approx \nu\overline{a'}\,\langle \Delta'_1, \mathcal{P}'_1\rangle \tag{1}$$

$$\nu\overline{b}\,\langle \Delta_2, \mathcal{P}_2\rangle \approx \nu\overline{b'}\,\langle \Delta'_2, \mathcal{P}'_2\rangle \tag{2}$$

Therefore, by Lemmas 4.6 and 5.6 $avars(\Delta_1 \cup \Delta_2) = avars(\Delta'_1 \cup \Delta'_2)$, $cvars(\Delta_1 \cup \Delta_2) = cvars(\Delta'_1 \cup \Delta'_2)$, and $names(\Delta_1 \cup \Delta_2) = names(\Delta'_1 \cup \Delta'_2)$. Moreover, by well-formedness of $\nu\overline{a}, \overline{b}\,\langle \Delta_1 \cup \Delta_2, \mathcal{P}_1 \mid \mathcal{P}_2\rangle$ and $\nu\overline{a'}, \overline{b'}\,\langle \Delta'_1 \cup \Delta'_2, \mathcal{P}'_1 \mid \mathcal{P}'_2\rangle$, there exist $\Delta_{10} \subseteq \Delta_1$, $\Delta_{20} \subseteq \Delta_2$, $\Delta'_{10} \subseteq \Delta'_1$, and $\Delta'_{20} \subseteq \Delta'_2$ such that $\Delta_1 \cup \Delta_2 = \Delta_1 \uplus \Delta_{20} = \Delta_{10} \uplus \Delta_2$ and $\Delta'_1 \cup \Delta'_2 = \Delta'_1 \uplus \Delta'_{20} = \Delta'_{10} \uplus \Delta'_2$.

Here Δ_{10} and Δ_{20} encode the knowledge not in Δ_2 and Δ_1 respectively. It remains to show that if

$$\nu\overline{a}, \overline{b}\,\langle \Delta_1 \cup \Delta_2,\, \mathcal{P}_1 \mid \mathcal{P}_2\rangle \xrightarrow{\eta} \nu\overline{c}\,\langle \Delta_3,\, \mathcal{Q}\rangle$$

then there exist $\overline{c'}$, Δ'_3, $\mathcal{Q}'$ such that

$$\nu\overline{a'}, \overline{b'}\,\langle \Delta'_1 \cup \Delta'_2,\, \mathcal{P}'_1 \mid \mathcal{P}'_2\rangle \overset{\eta}{\Longrightarrow} \nu\overline{c'}\,\langle \Delta'_3,\, \mathcal{Q}'\rangle \qquad \nu\overline{c}\,\langle \Delta_3,\, \mathcal{Q}\rangle \approx \nu\overline{c'}\,\langle \Delta'_3,\, \mathcal{Q}'\rangle$$

and the symmetric.

Let $\nu\overline{a}, \overline{b}\,\langle \Delta_1 \cup \Delta_2,\, \mathcal{P}_1 \mid \mathcal{P}_2\rangle \xrightarrow{\eta} \nu\overline{c}\,\langle \Delta_3,\, \mathcal{Q}\rangle$. We consider the cases of this transition.

♦ Conc-App-Trans: we have

$$\frac{(\Delta_1 \cup \Delta_2)(\kappa) = \mathcal{V}}{\nu\overline{a}, \overline{b}\,\langle \Delta_1 \cup \Delta_2,\, \mathcal{P}_1 \mid \mathcal{P}_2\rangle \xrightarrow{\mathsf{app}\,\kappa} \nu\overline{a}, \overline{b}\,\langle \Delta_1 \cup \Delta_2,\, \mathcal{P}_1 \mid \mathcal{P}_2 \mid \mathsf{app}\,\mathcal{V}\rangle}$$

W.l.o.g. let $\kappa \in \mathit{cvars}(\Delta_1)$ and, thus, $\kappa \in \mathit{cvars}(\Delta'_1)$. Then

$$\nu\overline{a}\,\langle \Delta_1,\, \mathcal{P}_1\rangle \xrightarrow{\mathsf{app}\,\kappa} \nu\overline{a}\,\langle \Delta_1,\, \mathcal{P}_1 \mid \mathsf{app}\,\mathcal{V}\rangle$$

and by (1), Definition 4.4, and Proposition 3.4 (vii), there exist $\overline{c'}$ and $\mathcal{Q}'_1$ such that

$$\nu\overline{a'}\,\langle \Delta'_1,\, \mathcal{P}'_1\rangle \overset{\mathsf{app}\,\kappa}{\Longrightarrow} \nu\overline{c'}\,\langle \Delta'_1,\, \mathcal{Q}'_1\rangle \qquad \nu\overline{a}\,\langle \Delta_1,\, \mathcal{P}_1 \mid \mathsf{app}\,\mathcal{V}\rangle \approx \nu\overline{c'}\,\langle \Delta'_1,\, \mathcal{Q}'_1\rangle$$

Thus, by Proposition 3.5 (i), and rule Par-Fst-Trans

$$\nu\overline{a'}, \overline{b'}\,\langle \Delta'_1 \cup \Delta'_2,\, \mathcal{P}'_1 \mid \mathcal{P}'_2\rangle \overset{\mathsf{app}\,\kappa}{\Longrightarrow} \nu\overline{c'}, \overline{b'}\,\langle \Delta'_1 \cup \Delta'_2,\, \mathcal{Q}'_1 \mid \mathcal{P}'_2\rangle$$

and by (2), rules Cxt-R and Cxt-Par, and because $(\hat{=}\approx) \subseteq (\sim\approx) \subseteq (\approx)$

$$\nu\overline{a}, \overline{b}\,\langle \Delta_1 \cup \Delta_2,\, \mathcal{P}_1 \mid \mathcal{P}_2 \mid \mathsf{app}\,\mathcal{V}\rangle\ (\approx)^{\mathsf{cxt}}\ \nu\overline{c'}, \overline{b'}\,\langle \Delta'_1 \cup \Delta'_2,\, \mathcal{Q}'_1 \mid \mathcal{P}'_2\rangle$$

♦ Par-Fst-Trans: using Proposition 3.3 we have

$$\frac{\nu\overline{a}, \overline{b}\,\langle \Delta_1 \cup \Delta_2,\, \mathcal{P}_1\rangle \xrightarrow{\eta} \nu\overline{c}\,\langle \Delta_3 \cup \Delta_2,\, \mathcal{Q}_1\rangle}{\nu\overline{a}, \overline{b}\,\langle \Delta_1 \cup \Delta_2,\, \mathcal{P}_1 \mid \mathcal{P}_2\rangle \xrightarrow{\eta} \nu\overline{c}\,\langle \Delta_3 \cup \Delta_2,\, \mathcal{Q}_1 \mid \mathcal{P}_2\rangle}$$

We distinguish three cases for η:

• $\eta = \mathsf{app}\,\kappa$ with $\kappa \in \mathit{cvars}(\Delta_{20})$: for some $\mathcal{V}$, $\Delta_{20}(\kappa) = \mathcal{V}$, $\Delta_3 = \Delta_1$, $\mathcal{Q}_1 = \mathcal{P}_1 \mid \mathsf{app}\,\mathcal{V}$, and $\overline{c} = \overline{a}, \overline{b}$. Moreover,

$$\nu\overline{b}\,\langle \Delta_2,\, \mathcal{P}_2\rangle \xrightarrow{\mathsf{app}\,\kappa} \nu\overline{b}\,\langle \Delta_2,\, \mathcal{P}_2 \mid \mathsf{app}\,\mathcal{V}\rangle$$

By (2), Definition 4.4, and Proposition 3.4 (vii) there exist $\overline{c'}$ and $\mathcal{Q}'_2$ such that

$$\nu\overline{b'}\,\langle \Delta'_2,\, \mathcal{P}'_2\rangle \overset{\mathsf{app}\,\kappa}{\Longrightarrow} \nu\overline{c'}\,\langle \Delta'_2,\, \mathcal{Q}'_2\rangle \qquad \nu\overline{b}\,\langle \Delta_2,\, \mathcal{P}_2 \mid \mathsf{app}\,\mathcal{V}\rangle \approx \nu\overline{c'}\,\langle \Delta'_2,\, \mathcal{Q}'_2\rangle$$

Thus, by Proposition 3.5 (i), and rule Par-Fst-Trans

$$\nu\overline{a'}, \overline{b'}\,\langle \Delta'_1 \cup \Delta'_2,\, \mathcal{P}'_1 \mid \mathcal{P}'_2\rangle \overset{\mathsf{app}\,\kappa}{\Longrightarrow} \nu\overline{c'}\,\langle \Delta'_1 \cup \Delta'_2,\, \mathcal{P}'_1 \mid \mathcal{Q}'_2\rangle$$

and by (1), rules Cxt-R and Cxt-Par, and because $(\hat{=}\approx) \subseteq (\sim\approx) \subseteq (\approx)$

$$\nu\overline{a}, \overline{b}\,\langle \Delta_1 \cup \Delta_2,\, \mathcal{P}_1 \mid \mathsf{app}\,\mathcal{V} \mid \mathcal{P}_2\rangle\ (\approx)^{\mathsf{cxt}}\ \nu\overline{a'}, \overline{c'}\,\langle \Delta'_1 \cup \Delta'_2,\, \mathcal{P}'_1 \mid \mathcal{Q}'_2\rangle$$

• $\eta = c?n$ and $n \in \mathit{names}(\Delta_{20})$: by Proposition 3.4 (iii) we have $\Delta_3 \cup \Delta_2 = \Delta_1[n] \uplus (\Delta_{20}\backslash\{n\}) = \Delta_1 \cup \Delta_2$, $\{\overline{c}\} = \{\overline{a}, \overline{b}\}$. By well-formedness of $\nu\overline{a}\,\langle \Delta_1, \mathcal{P}_1\rangle$ we get $c \in \mathit{names}(\Delta_1)$ and, using Proposition 3.5 (ii),

$$\nu\overline{a}\,\langle \Delta_1[n],\, \mathcal{P}_1\rangle \xrightarrow{c?n} \nu\overline{a}\,\langle \Delta_1[n],\, \mathcal{Q}_1\rangle$$

By (1) and Lemma 4.11

$$\nu\overline{a}\,\langle \Delta_1[n],\, \mathcal{P}_1\rangle \approx \nu\overline{a'}\,\langle \Delta'_1[n],\, \mathcal{P}'_1\rangle$$

Thus, by Definition 4.4 and Proposition 3.4 (iii) there exist $\overline{c'}$ and $\mathcal{Q}'_1$ such that

$$\nu\overline{a'}\,\langle \Delta'_1[n],\, \mathcal{P}'_1\rangle \overset{c?n}{\Longrightarrow} \nu\overline{c'}\,\langle \Delta'_1[n],\, \mathcal{Q}'_1\rangle \qquad \nu\overline{a}\,\langle \Delta_1[n],\, \mathcal{Q}_1\rangle \approx \nu\overline{c'}\,\langle \Delta'_1[n],\, \mathcal{Q}'_1\rangle$$

By Lemma 5.6, $\Delta'_1[n] \uplus (\Delta'_{20}\backslash\{n\}) = \Delta'_1 \cup \Delta'_2$. Thus, by Proposition 3.5 (i) and rule Par-Fst-Trans,

$$\nu\overline{a'}, \overline{b'}\,\langle \Delta'_1 \cup \Delta'_2,\, \mathcal{P}'_1 \mid \mathcal{P}'_2\rangle \overset{c?n}{\Longrightarrow} \nu\overline{c'}, \overline{b'}\,\langle \Delta'_1 \cup \Delta'_2,\, \mathcal{Q}'_1 \mid \mathcal{P}'_2\rangle$$

and by (2), rules Cxt-R and Cxt-Par, and because $(\hat{=}\approx) \subseteq (\sim\approx) \subseteq (\approx)$

$$\nu\overline{a}, \overline{b}\,\langle \Delta_1 \cup \Delta_2,\, \mathcal{Q}_1 \mid \mathcal{P}_2\rangle\ (\approx)^{\mathsf{cxt}}\ \nu\overline{c'}, \overline{b'}\,\langle \Delta'_1 \cup \Delta'_2,\, \mathcal{Q}'_1 \mid \mathcal{P}'_2\rangle$$

• $\eta \notin \{\mathsf{app}\,\kappa \mid \kappa \in \mathit{cvars}(\Delta_{20})\} \cup \{c?n \mid n \in \mathit{names}(\Delta_{20})\}$: In this case, using the well-formedness of $\nu\overline{a}\,\langle \Delta_1, \mathcal{P}_1\rangle$ and Proposition 3.4 on the transition $\nu\overline{a}, \overline{b}\,\langle \Delta_1 \cup \Delta_2, \mathcal{P}_1\rangle \xrightarrow{\eta} \nu\overline{c}\,\langle \Delta_3 \cup \Delta_2, \mathcal{Q}_1\rangle$, we have

$$\mathit{names}(\eta) \cap \mathit{names}(\Delta_{20}) = \mathit{avars}(\eta) \cap \mathit{avars}(\Delta_{20}) = \mathit{cvars}(\eta) \cap \mathit{cvars}(\Delta_{20}) = \emptyset$$

and by Lemmas 4.6 and 5.6

$$\mathit{names}(\eta) \cap \mathit{names}(\Delta'_{20}) = \mathit{avars}(\eta) \cap \mathit{avars}(\Delta'_{20}) = \mathit{cvars}(\eta) \cap \mathit{cvars}(\Delta'_{20}) = \emptyset$$

Furthermore, by Proposition 3.5 (ii), for $\{\overline{c_0}\} = \{\overline{c}\}\backslash\{\overline{b}\}$,

$$\nu\overline{a}\,\langle \Delta_1, \mathcal{P}_1\rangle \xrightarrow{\eta} \nu\overline{c_0}\,\langle \Delta_3, \mathcal{Q}_1\rangle$$

By (1) and Definition 4.4 there exist $\overline{c'}$, $\mathcal{Q}'_1$ such that

$$\nu\overline{a'}\,\langle \Delta'_1, \mathcal{P}'_1\rangle \overset{\eta}{\Longrightarrow} \nu\overline{c'}\,\langle \Delta'_3, \mathcal{Q}'_1\rangle \qquad \nu\overline{c_0}\,\langle \Delta_3, \mathcal{Q}_1\rangle \approx \nu\overline{c'}\,\langle \Delta'_3, \mathcal{Q}'_1\rangle$$

Thus, by Proposition 3.5 (i) and rule Par-Fst-Trans

$$\nu\overline{a'}, \overline{b'}\,\langle \Delta'_1 \cup \Delta'_2,\, \mathcal{P}'_1 \mid \mathcal{P}'_2\rangle \overset{\eta}{\Longrightarrow} \nu\overline{c'}, \overline{b'}\,\langle \Delta'_3 \cup \Delta'_2,\, \mathcal{Q}'_1 \mid \mathcal{P}'_2\rangle$$

and by (2) and rules Cxt-R and Cxt-Par

$$\nu\overline{c}\,\langle \Delta_3 \cup \Delta_2,\, \mathcal{Q}_1 \mid \mathcal{P}_2\rangle\ (\approx)^{\mathsf{cxt}}\ \nu\overline{c'}, \overline{b'}\,\langle \Delta'_3 \cup \Delta_2,\, \mathcal{Q}'_1 \mid \mathcal{P}'_2\rangle$$

♦ Comm-Name-Trans: using Proposition 3.4 (i) and (iii) we have

$$\frac{\langle \Delta_1[\overline{a}] \cup \Delta_2[\overline{b}], \mathcal{P}_1\rangle \xrightarrow{c!n} \langle \Delta_1[\overline{a}] \cup \Delta_2[\overline{b}], \mathcal{Q}_1\rangle \quad \langle \Delta_1[\overline{a}] \cup \Delta_2[\overline{b}], \mathcal{P}_2\rangle \xrightarrow{c?n} \langle \Delta_1[\overline{a}] \cup \Delta_2[\overline{b}], \mathcal{Q}_2\rangle}{\nu\overline{a}, \overline{b}\,\langle \Delta_1 \cup \Delta_2,\, \mathcal{P}_1 \mid \mathcal{P}_2\rangle \xrightarrow{\tau} \nu\overline{a}, \overline{b}\,\langle \Delta_1 \cup \Delta_2,\, \mathcal{Q}_1 \mid \mathcal{Q}_2\rangle}$$

By well-formedness of $\nu\overline{a}\,\langle \Delta_1, \mathcal{P}_1\rangle$ and $\nu\overline{b}\,\langle \Delta_2, \mathcal{P}_2\rangle$, and Lemma 5.6, it must be that $c \in names(\Delta_1) \cap names(\Delta_2) = names(\Delta'_1) \cap names(\Delta'_2)$ and $n \in names(\Delta_1) = names(\Delta'_1)$, and, using Proposition 3.5 (ii),

$$\langle \Delta_1[\overline{a}], \mathcal{P}_1\rangle \xrightarrow{c!n} \langle \Delta_1[\overline{a}], \mathcal{Q}_1\rangle \qquad \langle \Delta_2[\overline{b}, n], \mathcal{P}_2\rangle \xrightarrow{c?n} \langle \Delta_2[\overline{b}, n], \mathcal{Q}_2\rangle$$

and by Proposition 3.6 (i)

$$\nu\overline{a}\,\langle \Delta_1, \mathcal{P}_1\rangle \xrightarrow{c!n} \nu\overline{c}\,\langle \Delta_1[n], \mathcal{Q}_1\rangle \qquad \nu\overline{b}\,\langle \Delta_2[n], \mathcal{P}_2\rangle \xrightarrow{c?n} \nu\overline{b}\,\langle \Delta_2[n], \mathcal{Q}_2\rangle$$

for $\{\overline{c}\} = \{\overline{a}\}\backslash\{n\}$. Therefore, by (1), (2), and Lemma 4.11, there exist $\overline{c'_1}$, $\overline{c'_2}$, $\mathcal{Q}'_1$, and $\mathcal{Q}'_2$ such that

$$\begin{aligned} \nu\overline{a'}\,\langle \Delta'_1, \mathcal{P}'_1\rangle &\overset{c!n}{\Longrightarrow} \nu\overline{c'_1}\,\langle \Delta'_1[n], \mathcal{Q}'_1\rangle \qquad \nu\overline{c}\,\langle \Delta_1[n], \mathcal{Q}_1\rangle \approx \nu\overline{c'_1}\,\langle \Delta'_1[n], \mathcal{Q}'_1\rangle \\ \nu\overline{b'}\,\langle \Delta'_2[n], \mathcal{P}'_2\rangle &\overset{c?n}{\Longrightarrow} \nu\overline{c'_2}\,\langle \Delta'_2[n], \mathcal{Q}'_2\rangle \qquad \nu\overline{b}\,\langle \Delta_2[n], \mathcal{Q}_2\rangle \approx \nu\overline{c'_2}\,\langle \Delta'_2[n], \mathcal{Q}'_2\rangle \end{aligned}$$

By Proposition 3.4 (i) and (iii), there exist $\overline{a'_1}$, $\overline{a'_2}$, $\overline{b'_1}$, $\mathcal{P}'_3$, $\mathcal{P}'_4$, $\mathcal{P}'_5$, and $\mathcal{P}'_6$, with $\{\overline{a'_2}\} = \{\overline{a'_1}\}\backslash\{n\}$ such that

$$\begin{aligned} \nu\overline{a'}\,\langle \Delta'_1, \mathcal{P}'_1\rangle &\overset{\tau}{\Longrightarrow} \nu\overline{a'_1}\,\langle \Delta'_1,\, c!\langle n\rangle.\mathcal{P}'_3 \mid \mathcal{P}'_4\rangle \\ &\xrightarrow{c!n} \nu\overline{a'_2}\,\langle \Delta'_1[n],\, \mathcal{P}'_3 \mid \mathcal{P}'_4\rangle \overset{\tau}{\Longrightarrow} \nu\overline{c'_1}\,\langle \Delta'_1[n],\, \mathcal{Q}'_1\rangle \\ \nu\overline{b'}\,\langle \Delta'_2[n], \mathcal{P}'_2\rangle &\overset{\tau}{\Longrightarrow} \nu\overline{b'_1}\,\langle \Delta'_2[n],\, c?(x).\mathcal{P}'_5 \mid \mathcal{P}'_6\rangle \\ &\xrightarrow{c?n} \nu\overline{b'_1}\,\langle \Delta'_2[n],\, \mathcal{P}'_5\{n/x\} \mid \mathcal{P}'_6\rangle \overset{\tau}{\Longrightarrow} \nu\overline{c'_2}\,\langle \Delta'_2[n],\, \mathcal{Q}'_2\rangle \end{aligned}$$

By Propositions 3.6 and 3.5 (i), and by well-formedness of $\nu\overline{a'}, \overline{b'}\,\langle \Delta'_1 \cup \Delta'_2, \mathcal{P}'_1 \mid \mathcal{P}'_2\rangle$, using the rules Par-Fst-Trans and its symmetric, and Comm-Name-Trans, we get

$$\begin{aligned} \nu\overline{a'}, \overline{b'}\,\langle \Delta'_1 \cup \Delta'_2, \mathcal{P}'_1 \mid \mathcal{P}'_2\rangle &\overset{\tau}{\Longrightarrow} \nu\overline{a'_1}, \overline{b'_1}\,\langle \Delta'_1 \cup \Delta'_2,\, c!\langle n\rangle.\mathcal{P}'_3 \mid \mathcal{P}'_4 \mid c?(x).\mathcal{P}'_5 \mid \mathcal{P}'_6\rangle \\ &\xrightarrow{\tau} \nu\overline{a'_1}, \overline{b'_1}\,\langle \Delta'_1 \cup \Delta'_2,\, \mathcal{P}'_3 \mid \mathcal{P}'_4 \mid \mathcal{P}'_5\{n/x\} \mid \mathcal{P}'_6\rangle \\ &\overset{\tau}{\Longrightarrow} \nu\overline{c'_3}, \overline{c'_2}\,\langle \Delta'_1 \cup \Delta'_2,\, \mathcal{Q}'_1 \mid \mathcal{Q}'_2\rangle \end{aligned}$$

for some $\overline{c'_3}$ with $\{\overline{c'_1}\} = \{\overline{c'_3}\}\backslash\{n\}$. Moreover, by rules Cxt-R and Cxt-Par,

$$\nu\overline{c}, \overline{b}\,\langle (\Delta_1 \cup \Delta_2)[n],\, \mathcal{Q}_1 \mid \mathcal{Q}_2\rangle\ (\approx)^{\mathsf{cxt}}\ \nu\overline{c'_1}, \overline{c'_2}\,\langle (\Delta'_1 \cup \Delta'_2)[n],\, \mathcal{Q}'_1 \mid \mathcal{Q}'_2\rangle$$

and we consider the following two cases:

• $n \in names(\Delta_1) = names(\Delta'_1)$: We have $n \notin \{\overline{a}\}$ and $n \notin \{\overline{c'_3}\}$, and thus, $\{\overline{c}\} = \{\overline{a}\}\backslash\{n\} = \{\overline{a}\}$ and $\{\overline{c'_1}\} = \{\overline{c'_3}\}\backslash\{n\} = \{\overline{c'_3}\}$, and the above can be written as

$$\nu\overline{a}, \overline{b}\,\langle \Delta_1 \cup \Delta_2,\, \mathcal{Q}_1 \mid \mathcal{Q}_2\rangle\ (\approx)^{\mathsf{cxt}}\ \nu\overline{c'_3}, \overline{c'_2}\,\langle \Delta'_1 \cup \Delta'_2,\, \mathcal{Q}'_1 \mid \mathcal{Q}'_2\rangle$$

• $n \notin names(\Delta_1) = names(\Delta_1')$: we have $\{\overline{a}\} = \{\overline{c}\} \cup \{n\}$ and $\{\overline{c_3'}\} = \{\overline{c_1'}\} \cup \{n\}$. Thus, by Lemma 4.12

$$\nu\overline{a}, \overline{b}\,\langle \Delta_1 \cup \Delta_2,\, Q_1 \mid Q_2\rangle \,(\approx)^{\mathsf{cxt}}\, \nu\overline{c_3'}, \overline{c_2'}\,\langle \Delta_1' \cup \Delta_2',\, Q_1' \mid Q_2'\rangle$$

♦ Comm-Proc-Trans: using Proposition 3.3 we have

$$\frac{\langle \Delta_1[\overline{a}] \cup \Delta_2[\overline{b}], \mathcal{P}_1\rangle \xrightarrow{c!\kappa} \langle (\Delta_1[\overline{a}] \cup \Delta_2[\overline{b}])[\kappa\mapsto\mathcal{V}], Q_1\rangle \quad \langle \Delta_1[\overline{a}] \cup \Delta_2[\overline{b}], \mathcal{P}_2\rangle \xrightarrow{c?\alpha} \langle (\Delta_1[\overline{a}] \cup \Delta_2[\overline{b}])[\alpha], Q_2\rangle}{\nu\overline{a}, \overline{b}\,\langle \Delta_1 \cup \Delta_2,\, \mathcal{P}_1 \mid \mathcal{P}_2\rangle \xrightarrow{\tau} \nu\overline{a}, \overline{b}\,\langle \Delta_1 \cup \Delta_2,\, Q_1 \mid Q_2\{\mathcal{V}/\alpha\}\rangle}$$

By well-formedness of $\nu\overline{a}\,\langle \Delta_1,\, \mathcal{P}_1\rangle$ and $\nu\overline{b}\,\langle \Delta_2,\, \mathcal{P}_2\rangle$, and Lemma 5.6, it must be that $c \in names(\Delta_1) \cap names(\Delta_2) = names(\Delta_1') \cap names(\Delta_2')$. Moreover, $\alpha, \kappa \notin \Delta_1 \cup \Delta_2 = \Delta_1' \cup \Delta_2'$. Using Proposition 3.5 (ii),

$$\langle \Delta_1[\overline{a}], \mathcal{P}_1\rangle \xrightarrow{c!\kappa} \langle \Delta_1[\overline{a}][k\mapsto\mathcal{V}], Q_1\rangle \qquad \langle \Delta_2[\overline{b}], \mathcal{P}_2\rangle \xrightarrow{c?\alpha} \langle \Delta_2[\overline{b}][\alpha], Q_2\rangle$$

and by Proposition 3.6 (i)

$$\nu\overline{a}\,\langle \Delta_1,\, \mathcal{P}_1\rangle \xrightarrow{c!\kappa} \nu\overline{a}\,\langle \Delta_1[\kappa\mapsto\mathcal{V}],\, Q_1\rangle \qquad \nu\overline{b}\,\langle \Delta_2,\, \mathcal{P}_2\rangle \xrightarrow{c?\alpha} \nu\overline{b}\,\langle \Delta_2[\alpha],\, Q_2\rangle$$

Therefore, by (1), (2), Definition 4.4, and Proposition 3.4 there exist $\overline{c_1'}$, $\overline{c_2'}$, $\mathcal{V}'$, Q_1', and Q_2' such that

$$\begin{aligned}
\nu\overline{a'}\,\langle \Delta_1',\, \mathcal{P}_1'\rangle &\overset{c!\kappa}{\Longrightarrow} \nu\overline{c_1'}\,\langle \Delta_1'[\kappa\mapsto\mathcal{V}'],\, Q_1'\rangle\\
\nu\overline{a}\,\langle \Delta_1[\kappa\mapsto\mathcal{V}],\, Q_1\rangle &\approx \nu\overline{c_1'}\,\langle \Delta_1'[\kappa\mapsto\mathcal{V}'],\, Q_1'\rangle\\
\nu\overline{b'}\,\langle \Delta_2',\, \mathcal{P}_2'\rangle &\overset{c?\alpha}{\Longrightarrow} \nu\overline{c_2'}\,\langle \Delta_2'[\alpha],\, Q_2'\rangle\\
\nu\overline{b}\,\langle \Delta_2[\alpha],\, Q_2\rangle &\approx \nu\overline{c_2'}\,\langle \Delta_2'[\alpha],\, Q_2'\rangle
\end{aligned}$$

Moreover, there exist $\overline{a_1'}$, $\overline{a_2'}$, $\overline{b_1'}$, $\mathcal{P}_3'$, $\mathcal{P}_4'$, $\mathcal{P}_5'$, and $\mathcal{P}_6'$, with $\{\overline{a_2'}\} = \{\overline{a_1'}\}\backslash\{n\}$ such that

$$\begin{aligned}
\nu\overline{a'}\,\langle \Delta_1',\, \mathcal{P}_1'\rangle &\overset{\tau}{\Longrightarrow} \nu\overline{a_1'}\,\langle \Delta_1',\, c!\langle\mathcal{V}'\rangle.\mathcal{P}_3' \mid \mathcal{P}_4'\rangle\\
&\xrightarrow{c!\kappa} \nu\overline{a_1'}\,\langle \Delta_1'[\kappa\mapsto\mathcal{V}'],\, \mathcal{P}_3' \mid \mathcal{P}_4'\rangle\\
&\overset{\tau}{\Longrightarrow} \nu\overline{c_1'}\,\langle \Delta_1'[\kappa\mapsto\mathcal{V}'],\, Q_1'\rangle\\
\nu\overline{b'}\,\langle \Delta_2',\, \mathcal{P}_2'\rangle &\overset{\tau}{\Longrightarrow} \nu\overline{b_1'}\,\langle \Delta_2',\, c?(x).\mathcal{P}_5' \mid \mathcal{P}_6'\rangle\\
&\xrightarrow{c?\alpha} \nu\overline{b_1'}\,\langle \Delta_2'[\alpha],\, \mathcal{P}_5'\{\alpha/x\} \mid \mathcal{P}_6'\rangle\\
&\overset{\tau}{\Longrightarrow} \nu\overline{c_2'}\,\langle \Delta_2'[\alpha],\, Q_2'\rangle
\end{aligned}$$

By Proposition 3.5 (i) and because $\nu\overline{a'}, \overline{b'}\,\langle \Delta_1' \cup \Delta_2',\, \mathcal{P}_1' \mid \mathcal{P}_2'\rangle$ is well-formed, using the rules Par-Fst-Trans and its symmetric,

$$\nu\overline{a'}, \overline{b'}\,\langle \Delta_1' \cup \Delta_2',\, \mathcal{P}_1' \mid \mathcal{P}_2'\rangle \overset{\tau}{\Longrightarrow} \nu\overline{a_1'}, \overline{b_1'}\,\langle \Delta_1' \cup \Delta_2',\, c!\langle\mathcal{V}'\rangle.\mathcal{P}_3' \mid \mathcal{P}_4' \mid c?(x).\mathcal{P}_5' \mid \mathcal{P}_6'\rangle$$

by rule Comm-Proc-Trans

$$\xrightarrow{\tau} \nu\overline{a_1'}, \overline{b_1'}\,\langle \Delta_1' \cup \Delta_2',\, \mathcal{P}_3' \mid \mathcal{P}_4' \mid \mathcal{P}_5'\{\alpha/x\}\{\mathcal{V}'/\alpha\} \mid \mathcal{P}_6'\rangle$$

and by applying Proposition 3.5 (i), rule Par-Fst-Trans and its symmetric, Proposition 3.8 (i), and Proposition 3.5 (ii)

$$\overset{\tau}{\Longrightarrow} \nu\overline{c_1'}, \overline{c_2'}\,\langle \Delta_1' \cup \Delta_2',\, Q_1' \mid Q_2'\{\mathcal{V}'/\alpha\}\rangle$$

Moreover, by rules Cxt-R and Cxt-Par,

$$\begin{array}{ll} \vdash & \nu\overline{a}, \overline{b}\,\langle(\Delta_1 \cup \Delta_2) \uplus \{\alpha, \kappa{\mapsto}\mathcal{V}\},\, \mathcal{Q}_1 \mid \mathcal{Q}_2\rangle \\ (\approx)^{\mathsf{cxt}} & \nu\overline{c_1'}, \overline{c_2'}\,\langle(\Delta_1' \cup \Delta_2') \uplus \{\alpha, \kappa{\mapsto}\mathcal{V}'\},\, \mathcal{Q}_1' \mid \mathcal{Q}_2'\rangle \end{array}$$

and because $\alpha \notin \Delta_1 \cup \Delta_2$ (thus $\alpha \notin \mathit{avars}(\mathcal{V}, \mathcal{V}')$), by rule Cxt-Subst,

$$\nu\overline{a}, \overline{b}\,\langle \Delta_1 \cup \Delta_2,\, \mathcal{Q}_1 \mid \mathcal{Q}_2\{\mathcal{V}/\alpha\}\rangle\ (\approx)^{\mathsf{cxt}}\ \nu\overline{c_1'}, \overline{c_2'}\,\langle \Delta_1' \cup \Delta_2',\, \mathcal{Q}_1' \mid \mathcal{Q}_2'\{\mathcal{V}'/\alpha\}\rangle$$

Case Cxt-Hide: By the induction hypothesis

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

and by Lemma 4.12

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

Case Cxt-Subst: By the induction hypothesis

$$\nu\overline{a}\,\langle \Delta \uplus \{\alpha, k{\mapsto}\mathcal{V}\},\, \mathcal{P}\rangle \approx \nu\overline{a'}\,\langle \Delta' \uplus \{\alpha, k{\mapsto}\mathcal{V}'\},\, \mathcal{P}'\rangle \tag{3}$$

It remains to show that if

$$\nu\overline{a}\,\langle \Delta\{\mathcal{V}/\alpha\},\, \mathcal{P}\{\mathcal{V}/\alpha\}\rangle \xrightarrow{\eta} \nu\overline{b}\,\langle \Delta_1\{\mathcal{V}/\alpha\},\, \mathcal{Q}\{\mathcal{V}/\alpha\}\rangle$$

then there exist $\overline{b'}$, Δ_1', $\mathcal{Q}'$ such that

$$\begin{array}{l} \nu\overline{a'}\,\langle \Delta'\{\mathcal{V}'/\alpha\},\, \mathcal{P}'\{\mathcal{V}'/\alpha\}\rangle \overset{\eta}{\Longrightarrow} \nu\overline{b'}\,\langle \Delta_1'\{\mathcal{V}'/\alpha\},\, \mathcal{Q}'\{\mathcal{V}'/\alpha\}\rangle \\ \nu\overline{b}\,\langle \Delta_1\{\mathcal{V}'/\alpha\},\, \mathcal{Q}\{\mathcal{V}'/\alpha\}\rangle\ (\approx)^{\mathsf{cxt}}\ \nu\overline{b'}\,\langle \Delta_1'\{\mathcal{V}'/\alpha\},\, \mathcal{Q}'\{\mathcal{V}'/\alpha\}\rangle \end{array}$$

Proving the symmetric is analogous.

Let

$$\nu\overline{a}\,\langle \Delta\{\mathcal{V}/\alpha\},\, \mathcal{P}\{\mathcal{V}/\alpha\}\rangle \xrightarrow{\eta} \nu\overline{b}\,\langle \Delta_1\{\mathcal{V}/\alpha\},\, \mathcal{Q}\{\mathcal{V}/\alpha\}\rangle$$

We distinguish three cases:

♦ η is a τ-action involving the rule App-Trans: It is easy to see that such an action is possible by applying rule Par-Fst-Trans (and its symmetric) an arbitrary number of times and then rule App-Trans. Hence,

$$\mathcal{P}\{\mathcal{V}/\alpha\} \mathrel{\hat{=}} \mathsf{app}\,\lambda\mathcal{P}_1 \mid \mathcal{P}_2\{\mathcal{V}/\alpha\}$$

Moreover, by Proposition 3.4 (vii), $\Delta_1 = \Delta$, $\overline{a} = \overline{b}$. By the properties of substitution, either $\mathcal{P} \mathrel{\hat{=}} \mathsf{app}\,\lambda\mathcal{P}_0 \mid \mathcal{P}_2$ and $\mathcal{P}_1 = \mathcal{P}_0\{\mathcal{V}/\alpha\}$, or $\mathcal{P} \mathrel{\hat{=}} \mathsf{app}\,\alpha \mid \mathcal{P}_1$ and $\mathcal{V} = \lambda\mathcal{P}_1$.

• $\mathcal{P} \mathrel{\hat{=}} \mathsf{app}\,\lambda\mathcal{P}_0 \mid \mathcal{P}_2$ and $\mathcal{P}_1 = \mathcal{P}_0\{\mathcal{V}/\alpha\}$: We have $\mathcal{Q} = \mathcal{P}_0 \mid \mathcal{P}_2$ and

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

By (3), Definition 4.4, and Proposition 3.4 (vii), there exist $\overline{b'}$, $\mathcal{Q}'$ such that

$$\begin{array}{l} \nu\overline{a'}\,\langle \Delta' \uplus \{\alpha, \kappa{\mapsto}\mathcal{V}'\},\, \mathcal{P}'\rangle \overset{\tau}{\Longrightarrow} \nu\overline{b'}\,\langle \Delta' \uplus \{\alpha, \kappa{\mapsto}\mathcal{V}'\},\, \mathcal{Q}'\rangle \\ \nu\overline{a}\,\langle \Delta \uplus \{\alpha, k{\mapsto}\mathcal{V}\},\, \mathcal{Q}\rangle \approx \nu\overline{b'}\,\langle \Delta' \uplus \{\alpha, k{\mapsto}\mathcal{V}'\},\, \mathcal{Q}'\rangle \end{array}$$

By applying Propositions 3.8 (i) and 3.5 (ii)

$$\nu\overline{a'}\,\langle\Delta'\{\mathcal{V}'/\alpha\},\,\mathcal{P}'\{\mathcal{V}'/\alpha\}\rangle \overset{\tau}{\Longrightarrow} \nu\overline{b'}\,\langle\Delta'\{\mathcal{V}'/\alpha\},\,\mathcal{Q}'\{\mathcal{V}'/\alpha\}\rangle$$

and by rule Cxt-Subst

$$\nu\overline{a}\,\langle\Delta\{\mathcal{V}'/\alpha\},\,\mathcal{Q}\{\mathcal{V}'/\alpha\}\rangle\ (\approx)^{\mathsf{cxt}}\ \nu\overline{b'}\,\langle\Delta'\{\mathcal{V}'/\alpha\},\,\mathcal{Q}'\{\mathcal{V}'/\alpha\}\rangle$$

• $\mathcal{P} \triangleq \mathsf{app}\,\alpha \mid \mathcal{P}_2$ and $\mathcal{V} = \lambda\mathcal{P}_1$: We have $\mathcal{Q} \triangleq \mathcal{P}_1 \mid \mathcal{P}_2$, and for some $\mathcal{Q}_1 \triangleq \mathcal{P}_2$ and $\mathcal{Q}_2 \triangleq \mathsf{app}\,\mathcal{V} \mid \mathcal{Q}$

$$\begin{aligned}\nu\overline{a}\,\langle\Delta \uplus \{\alpha, \kappa{\mapsto}\mathcal{V}\},\,\mathcal{P}\rangle &\overset{\mathsf{app}\,\alpha}{\longrightarrow} \nu\overline{a}\,\langle\Delta \uplus \{\alpha, \kappa{\mapsto}\mathcal{V}\},\,\mathcal{Q}_1\rangle\\ &\overset{\mathsf{app}\,\kappa}{\longrightarrow} \nu\overline{a}\,\langle\Delta \uplus \{\alpha, \kappa{\mapsto}\mathcal{V}\},\,\mathcal{Q}_2\rangle\\ &\overset{\tau}{\longrightarrow} \nu\overline{a}\,\langle\Delta \uplus \{\alpha, \kappa{\mapsto}\mathcal{V}\},\,\mathcal{Q}\rangle\end{aligned}$$

By (3), Definition 4.4, and Proposition 3.4, there exist $\overline{a'_1}$, $\overline{a'_2}$, $\overline{b'}$, $\mathcal{Q}'_1$, $\mathcal{Q}'_2$, $\mathcal{Q}'_3$, and $\mathcal{Q}'_4$ such that $\mathcal{Q}'_1 \triangleq \mathsf{app}\,\alpha \mid \mathcal{Q}'_2$, $\mathcal{Q}'_4 \triangleq \mathsf{app}\,\mathcal{V} \mid \mathcal{Q}'_3$, and

$$\begin{aligned}\nu\overline{a'}\,\langle\Delta' \uplus \{\alpha, \kappa{\mapsto}\mathcal{V}'\},\,\mathcal{P}'\rangle &\overset{\tau}{\Longrightarrow} \nu\overline{a'_1}\,\langle\Delta' \uplus \{\alpha, \kappa{\mapsto}\mathcal{V}'\},\,\mathcal{Q}'_1\rangle\\ &\overset{\mathsf{app}\,\alpha}{\longrightarrow} \nu\overline{a'_1}\,\langle\Delta' \uplus \{\alpha, \kappa{\mapsto}\mathcal{V}'\},\,\mathcal{Q}'_2\rangle\\ &\overset{\tau}{\Longrightarrow} \nu\overline{a'_2}\,\langle\Delta' \uplus \{\alpha, \kappa{\mapsto}\mathcal{V}'\},\,\mathcal{Q}'_3\rangle\\ &\overset{\mathsf{app}\,\kappa}{\longrightarrow} \nu\overline{a'_2}\,\langle\Delta' \uplus \{\alpha, \kappa{\mapsto}\mathcal{V}'\},\,\mathcal{Q}'_4\rangle\\ &\overset{\tau}{\Longrightarrow} \nu\overline{b'}\,\langle\Delta' \uplus \{\alpha, \kappa{\mapsto}\mathcal{V}'\},\,\mathcal{Q}'\rangle\end{aligned}$$

$$\nu\overline{a}\,\langle\Delta \uplus \{\alpha, k{\mapsto}\mathcal{V}\},\,\mathcal{Q}\rangle \approx \nu\overline{b'}\,\langle\Delta' \uplus \{\alpha, k{\mapsto}\mathcal{V}'\},\,\mathcal{Q}'\rangle$$

It is easy to show that there also exist $\overline{a'_3}$, $\mathcal{Q}'_5$, $\mathcal{Q}'_6$, and $\mathcal{Q}'_7$ such that $\mathcal{Q}'_5 \triangleq \mathsf{app}\,\alpha \mid \mathcal{Q}'_6$, $\mathcal{Q}'_7 \triangleq \mathsf{app}\,\mathcal{V} \mid \mathcal{Q}'_6$, and

$$\begin{aligned}\nu\overline{a'}\,\langle\Delta' \uplus \{\alpha, \kappa{\mapsto}\mathcal{V}'\},\,\mathcal{P}'\rangle &\overset{\tau}{\Longrightarrow} \nu\overline{a'_3}\,\langle\Delta' \uplus \{\alpha, \kappa{\mapsto}\mathcal{V}'\},\,\mathcal{Q}'_5\rangle\\ &\overset{\mathsf{app}\,\alpha}{\longrightarrow} \nu\overline{a'_3}\,\langle\Delta' \uplus \{\alpha, \kappa{\mapsto}\mathcal{V}'\},\,\mathcal{Q}'_6\rangle\\ &\overset{\mathsf{app}\,\kappa}{\longrightarrow} \nu\overline{a'_3}\,\langle\Delta' \uplus \{\alpha, \kappa{\mapsto}\mathcal{V}'\},\,\mathcal{Q}'_7\rangle\\ &\overset{\tau}{\Longrightarrow} \nu\overline{b'}\,\langle\Delta' \uplus \{\alpha, \kappa{\mapsto}\mathcal{V}'\},\,\mathcal{Q}'\rangle\end{aligned}$$

By applying Propositions 3.8 (i) and 3.5 (ii), and rule App-Trans

$$\begin{aligned}\nu\overline{a'}\,\langle\Delta'\{\mathcal{V}'/\alpha\},\,\mathcal{P}'\{\mathcal{V}'/\alpha\}\rangle &\overset{\tau}{\Longrightarrow} \nu\overline{a'_3}\,\langle\Delta'\{\mathcal{V}'/\alpha\},\,\mathcal{Q}'_5\{\mathcal{V}'/\alpha\}\rangle\\ &\overset{\tau}{\Longrightarrow} \nu\overline{b'}\,\langle\Delta'\{\mathcal{V}'/\alpha\},\,\mathcal{Q}'\{\mathcal{V}'/\alpha\}\rangle\end{aligned}$$

and by rule Cxt-Subst

$$\nu\overline{a}\,\langle\Delta\{\mathcal{V}/\alpha\},\,\mathcal{Q}'\{\mathcal{V}'/\alpha\}\rangle\ (\approx)^{\mathsf{cxt}}\ \nu\overline{b'}\,\langle\Delta'\{\mathcal{V}'/\alpha\},\,\mathcal{Q}'\{\mathcal{V}'/\alpha\}\rangle$$

♦ $\eta = \mathsf{app}\,\alpha_1$: Because α is substituted by $\mathcal{V}$ and $\mathcal{V}'$ and $\alpha \notin \mathit{fav}(\mathcal{V}, \mathcal{V}')$, it must be $\alpha_1 \neq \alpha$; by Proposition 3.4 (v)

$$\mathcal{P}\{\mathcal{V}/\alpha\} \triangleq \mathsf{app}\,\alpha_1 \mid \mathcal{Q}\{\mathcal{V}/\alpha\}$$

Then either $\mathcal{P} \triangleq \mathsf{app}\,\alpha_1 \mid \mathcal{Q}$ or $\mathcal{P} \triangleq \mathsf{app}\,\alpha \mid \mathcal{Q}$ and $\mathcal{V} = \alpha_1$.

• $\mathcal{P} \triangleq \mathsf{app}\,\alpha_1 \mid \mathcal{Q}$: We have

$$\nu\overline{a}\,\langle\Delta \uplus \{\alpha, \kappa{\mapsto}\mathcal{V}\},\,\mathcal{P}\rangle \overset{\mathsf{app}\,\alpha_1}{\longrightarrow} \nu\overline{a}\,\langle\Delta \uplus \{\alpha, \kappa{\mapsto}\mathcal{V}\},\,\mathcal{Q}\rangle$$

By (3), Definition 4.4, and Proposition 3.4, there exist $\overline{b'}$ and Q' such that

$$\nu\overline{a'}\,\langle \Delta' \uplus \{\alpha, \kappa \mapsto \mathcal{V}'\},\, \mathcal{P}' \rangle \overset{\mathsf{app}\,\alpha_1}{\Longrightarrow} \nu\overline{b'}\,\langle \Delta' \uplus \{\alpha, \kappa \mapsto \mathcal{V}'\},\, Q' \rangle$$
$$\nu\overline{a}\,\langle \Delta \uplus \{\alpha, \kappa \mapsto \mathcal{V}\},\, Q \rangle \approx \nu\overline{b'}\,\langle \Delta' \uplus \{\alpha, \kappa \mapsto \mathcal{V}'\},\, Q' \rangle$$

By Propositions 3.8 (i) and 3.5 (ii), and by rule Cxt-Subst

$$\nu\overline{a'}\,\langle \Delta'\{\mathcal{V}'/\alpha\},\, \mathcal{P}'\{\mathcal{V}'/\alpha\} \rangle \overset{\mathsf{app}\,\alpha_1}{\Longrightarrow} \nu\overline{b'}\,\langle \Delta'\{\mathcal{V}'/\alpha\},\, Q'\{\mathcal{V}'/\alpha\} \rangle$$
$$\nu\overline{a}\,\langle \Delta\{\mathcal{V}/\alpha\},\, Q\{\mathcal{V}/\alpha\} \rangle\, (\approx)^{\mathsf{cxt}}\, \nu\overline{b'}\,\langle \Delta'\{\mathcal{V}'/\alpha\},\, Q'\{\mathcal{V}'/\alpha\} \rangle$$

- $\mathcal{P} \triangleq \mathsf{app}\,\alpha \mid Q$ and $\mathcal{V} = \alpha_1$: We have

$$\nu\overline{a}\,\langle \Delta \uplus \{\alpha, \kappa \mapsto \mathcal{V}\},\, \mathcal{P} \rangle \overset{\mathsf{app}\,\alpha}{\longrightarrow} \nu\overline{a}\,\langle \Delta \uplus \{\alpha, \kappa \mapsto \mathcal{V}\},\, Q \rangle$$

By (3), Definition 4.4, and Proposition 3.4, there exist $\overline{a'_1}$, $\overline{b'}$, $\mathcal{P}'_1$, $\mathcal{P}'_2$, and Q' such that $\mathcal{P}'_1 \triangleq \mathsf{app}\,\alpha \mid \mathcal{P}'_2$ and

$$\begin{aligned}\nu\overline{a'}\,\langle \Delta' \uplus \{\alpha, \kappa \mapsto \mathcal{V}'\},\, \mathcal{P}' \rangle &\overset{\tau}{\Longrightarrow} \nu\overline{a'_1}\,\langle \Delta' \uplus \{\alpha, \kappa \mapsto \mathcal{V}'\},\, \mathcal{P}'_1 \rangle \\ &\overset{\mathsf{app}\,\alpha}{\longrightarrow} \nu\overline{a'_1}\,\langle \Delta' \uplus \{\alpha, \kappa \mapsto \mathcal{V}'\},\, \mathcal{P}'_2 \rangle \\ &\overset{\tau}{\Longrightarrow} \nu\overline{b'}\,\langle \Delta' \uplus \{\alpha, \kappa \mapsto \mathcal{V}'\},\, Q' \rangle \\ \nu\overline{a}\,\langle \Delta \uplus \{\alpha, \kappa \mapsto \mathcal{V}\},\, Q \rangle &\approx \nu\overline{b'}\,\langle \Delta' \uplus \{\alpha, \kappa \mapsto \mathcal{V}'\},\, Q' \rangle\end{aligned}$$

By Propositions 3.8 (i) and 3.5 (ii), and by rule Abs-App-Trans

$$\begin{aligned}\nu\overline{a'}\,\langle \Delta'\{\mathcal{V}'/\alpha\},\, \mathcal{P}'\{\mathcal{V}'/\alpha\} \rangle &\overset{\tau}{\Longrightarrow} \nu\overline{a'_1}\,\langle \Delta'\{\mathcal{V}'/\alpha\},\, \mathcal{P}'_1\{\mathcal{V}'/\alpha\} \rangle \\ &\overset{\mathsf{app}\,\alpha_1}{\longrightarrow} \nu\overline{a'_1}\,\langle \Delta'\{\mathcal{V}'/\alpha\},\, \mathcal{P}'_2\{\mathcal{V}'/\alpha\} \rangle \\ &\overset{\tau}{\Longrightarrow} \nu\overline{b'}\,\langle \Delta'\{\mathcal{V}'/\alpha\},\, Q'\{\mathcal{V}'/\alpha\} \rangle\end{aligned}$$

and by rule Cxt-Subst

$$\nu\overline{a}\,\langle \Delta\{\mathcal{V}/\alpha\},\, Q\{\mathcal{V}/\alpha\} \rangle\, (\approx)^{\mathsf{cxt}}\, \nu\overline{b'}\,\langle \Delta'\{\mathcal{V}'/\alpha\},\, Q'\{\mathcal{V}'/\alpha\} \rangle$$

♦ Otherwise: By Propositions 3.8 (ii) and 3.5 (i), and because by definition of $(\,)^{\mathsf{cxt}}$ for the case where $\eta = c?\alpha_1$ or $\eta = c!\kappa_1$ it must be $\alpha_1 \neq \alpha$ or $\kappa_1 \neq \kappa$, respectively,

$$\nu\overline{a}\,\langle \Delta_1 \uplus \{\alpha, \kappa \mapsto \mathcal{V}\},\, \mathcal{P} \rangle \overset{\eta}{\longrightarrow} \nu\overline{b}\,\langle \Delta_2 \uplus \{\alpha, \kappa \mapsto \mathcal{V}\},\, Q \rangle$$

By (3) and Definition 4.4, there exist $\overline{b'}$, Δ'_2 Q' such that

$$\nu\overline{a}\,\langle \Delta'_1 \uplus \{\alpha, \kappa \mapsto \mathcal{V}'\},\, \mathcal{P}' \rangle \overset{\eta}{\Longrightarrow} \nu\overline{b'}\,\langle \Delta'_2 \uplus \{\alpha, \kappa \mapsto \mathcal{V}'\},\, Q' \rangle$$
$$\nu\overline{b}\,\langle \Delta_2 \uplus \{\alpha, \kappa \mapsto \mathcal{V}\},\, Q \rangle \approx \nu\overline{b'}\,\langle \Delta'_2 \uplus \{\alpha, \kappa \mapsto \mathcal{V}'\},\, Q' \rangle$$

By Propositions 3.8 (i) and 3.5 (ii), and rule Cxt-Subst

$$\nu\overline{a}\,\langle \Delta'_1\{\mathcal{V}'/\alpha\},\, \mathcal{P}'\{\mathcal{V}'/\alpha\} \rangle \overset{\eta}{\Longrightarrow} \nu\overline{b'}\,\langle \Delta'_2\{\mathcal{V}'/\alpha\},\, Q'\{\mathcal{V}'/\alpha\} \rangle$$
$$\nu\overline{b}\,\langle \Delta_2\{\mathcal{V}/\alpha\},\, Q\{\mathcal{V}/\alpha\} \rangle\, (\approx)^{\mathsf{cxt}}\, \nu\overline{b'}\,\langle \Delta'_2\{\mathcal{V}'/\alpha\},\, Q'\{\mathcal{V}'/\alpha\} \rangle \qquad \square$$

Corollary 5.8 (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 4.7, there exist $\overline{b}$ such that $\langle\{\overline{b}\}, P\rangle \approx \langle\{\overline{b}\}, P'\rangle$.

Let Q be a process with $fn(Q) \subseteq \{\overline{n}\}$. By Lemma 4.11, $\langle\{\overline{b}\} \cup \{\overline{n}\}, P\rangle \approx \langle\{\overline{b}\} \cup \{\overline{n}\}, P'\rangle$ and $\langle\{\overline{b}\}\cup\{\overline{n}\}, Q\rangle \approx \langle\{\overline{b}\}\cup\{\overline{n}\}, Q'\rangle$. By rules Cxt-R and Cxt-Par of Figure 7 $\langle\{\overline{b}\}\cup\{\overline{n}\}, Q \mid P\rangle \ (\approx)^{\mathsf{cxt}}$ $\langle\{\overline{b}\} \cup \{\overline{n}\}, Q \mid P'\rangle$, and by Theorem 5.7 $\langle\{\overline{b}\} \cup \{\overline{n}\}, Q \mid P\rangle \approx \langle\{\overline{b}\} \cup \{\overline{n}\}, Q \mid P'\rangle$, Hence, by Definition 4.7, $Q \mid P \simeq Q \mid P'$. □

Theorem 5.9 (Soundness)**.** $(\simeq) \subseteq (\cong_{\mathrm{pcxt}})$.

Proof. In Propositions 5.4 and 5.3 and Corollary 5.8 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 $(\cong_{\mathrm{pcxt}})$. □

6 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.

6.1 Concretion of Configurations

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

Definition 6.1 (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 variables 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.

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

Lemma 6.2. *If $\mathcal{P}/f \equiv Q$ then there exists Q_0 such that $Q = Q_0/f$.*

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

Lemma 6.3. *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 6.4. *If $\nu\overline{a}.\mathcal{P}/f \to Q$ 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 Q_0 such that

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

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

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

Proof. By induction on the height of the derivation tree $\mathcal{P}/f \to Q$, using Proposition 4.9 and Lemma 6.3. □

Proposition 6.5. *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

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

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

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

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 6.4 there exist $\overline{b}$ and Q_0 such that

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

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

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

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

$$Q \equiv \nu\overline{b}.\,(\Delta/f \mid Q_1) \equiv \nu\overline{b}\,\langle \Delta,\, Q_0\rangle/f \qquad \nu\overline{a}\,\langle \Delta,\, \mathcal{P}\rangle \stackrel{\tau}{\Longrightarrow} \nu\overline{b}\,\langle \Delta,\, Q_0\rangle$$

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

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

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

Corollary 6.6. *If $\nu\overline{a}\,\langle \Delta,\, \mathcal{P}\rangle/f \to^* Q$ then there exist $\overline{b}$, $\overline{c,\alpha}$, Q_0 such that*

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

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

Proposition 6.7. *If* $\nu\overline{a}\,\langle\Delta,\,\mathcal{P}\rangle \xrightarrow{\tau} \nu\overline{b}\,\langle\Delta,\,\mathcal{Q}\rangle$ *then one of the following holds.*

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

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

Proof. By rule induction. □

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

Lemma 6.8. $P \cong_{\mathrm{pcxt}} \mathtt{app}\,\lambda P$

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. □

6.2 Completeness

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

Definition 6.9 ($\mathbb{X}$)**.**

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

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

Proposition 6.10. *If* $\nu\overline{a}\,\langle\Delta,\,\mathcal{P}\rangle\ \mathbb{X}\ \nu\overline{a'}\,\langle\Delta',\,\mathcal{P}'\rangle$ *and* $\nu\overline{a}\,\langle\Delta,\,\mathcal{P}\rangle \xrightarrow{\tau} \nu\overline{b}\,\langle\Delta,\,\mathcal{Q}\rangle$ *then there exist* $\overline{b'}$ *and* $\mathcal{Q}'$ *such that*

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

Proof. By the first premise and Definition 6.9 we get that there exists f such that

$$\nu\overline{a}\,\langle\Delta,\,\mathcal{P}\rangle/f \cong_{\mathrm{pcxt}} \nu\overline{a'}\,\langle\Delta',\,\mathcal{P}'\rangle/f \tag{1}$$

By the second premise and Proposition 6.7 we get that either $\nu\overline{a}\,\langle\Delta,\,\mathcal{P}\rangle/f \equiv \nu\overline{b}\,\langle\Delta,\,\mathcal{Q}\rangle/f$, or $\nu\overline{a}\,\langle\Delta,\,\mathcal{P}\rangle/f \to \nu\overline{b}\,\langle\Delta,\,\mathcal{Q}\rangle/f$. In the former case the proof is completed because, by Proposition 4.9 and Theorem 5.9, $(\equiv) \subseteq (\simeq) \subset (\cong_{\mathrm{pcxt}})$, hence $\nu\overline{b}\,\langle\Delta,\,\mathcal{Q}\rangle/f \cong_{\mathrm{pcxt}} \nu\overline{a'}\,\langle\Delta',\,\mathcal{P}'\rangle/f$ and $\nu\overline{b}\,\langle\Delta,\,\mathcal{Q}\rangle\ \mathbb{X}\ \nu\overline{a'}\,\langle\Delta',\,\mathcal{P}'\rangle$. In the latter case, by (1) and Definition 2.7, we have that there exists $\mathcal{Q}'$ such that

$$\nu\overline{a'}\,\langle\Delta,\,\mathcal{P}'\rangle/f \to^{*} \mathcal{Q}' \tag{2}$$

$$\nu\overline{b}\,\langle\Delta,\,\mathcal{Q}\rangle/f \cong_{\mathrm{pcxt}} \mathcal{Q}' \tag{3}$$

By (2) and Corollary 6.6, there exist $\overline{b'}$, $\overline{c,\alpha}$, and $\mathcal{Q}'_0$ such that $\overline{f(\alpha) = c}$ and

$$\mathcal{Q}' \equiv \nu\overline{b'}\,\langle\Delta',\,\mathcal{Q}'_0\rangle/f \mid \prod_{c\in\{\overline{c}\}} c?.\mathtt{0} \tag{4}$$

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

By (3), (4), the fact that $(\equiv) \subseteq (\simeq) \subset (\cong_{\text{pcxt}})$, and Lemma 6.8 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 6.11. *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),

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

Thus by the first premise and Definition 6.9 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

$$\begin{aligned} \nu\overline{a'}\,\langle \Delta'_1,\, \mathcal{P}'\rangle/f \mid c!.\mathtt{0} \to^* Q' \qquad Q' \not\downarrow_c \\ \nu\overline{a}\,\langle \Delta_1,\, Q\rangle/f \cong_{\text{pcxt}} Q' \end{aligned} \tag{1}$$

Therefore, there exists Q'_1 such that

$$\begin{aligned} \nu\overline{a'}\,\langle \Delta'_1,\, \mathcal{P}'\rangle/f \to^* Q'_1 \mid c?.\mathtt{0} \\ \nu\overline{a'}\,\langle \Delta'_1,\, \mathcal{P}'\rangle/f \mid c!.\mathtt{0} \to^* Q'_1 \mid c?.\mathtt{0} \mid c!.\mathtt{0} \to^* Q'_1 \to^* Q' \end{aligned} \tag{2}$$

and by Corollary 6.6 there exist $\overline{b'}$, $\overline{c,\alpha}$, Q'_0 such that $f(\alpha_i) = c_i$ and

$$Q'_1 \mid c?.\mathtt{0} \equiv \nu\overline{b'}\,\langle \Delta'_1,\, Q'_0\rangle/f \mid c?.\mathtt{0} \mid \prod_{c_i\in\{\overline{c}\}} c_i?.\mathtt{0} \tag{3}$$

$$\nu\overline{a'}\,\langle \Delta'_1,\, \mathcal{P}'\rangle \overset{\tau}{\Longrightarrow} \nu\overline{b'}\,\langle \Delta'_1,\, Q'_0 \mid \mathsf{app}\,\alpha \mid \prod_{\alpha_i\in\{\overline{\alpha}\}} \mathsf{app}\,\alpha_i\rangle \tag{4}$$

By (2) and (3),

$$\nu\overline{b'}\,\langle \Delta_1', \mathcal{Q}_0' \rangle / f \mid \prod_{c_i \in \{\overline{c}\}} c_i?.\mathtt{0} \to^* \mathcal{Q}'$$

and thus

$$\nu\overline{b'}\,\langle \Delta_1', \mathcal{Q}_0' \mid \prod_{\alpha_i \in \{\overline{\alpha}\}} \mathtt{app}\,\alpha_i \rangle / f \to^* \nu\overline{b'}\,\langle \Delta_1', \mathcal{Q}_0' \rangle / f \mid \prod_{c_i \in \{\overline{c}\}} c_i?.\mathtt{0} \to^* \mathcal{Q}'$$

By Corollary 6.6 there exist $\overline{d'}$, $\overline{c', \alpha'}$, $\mathcal{Q}_2'$ such that $f(\alpha_i') = c_i'$ and

$$\mathcal{Q}' \equiv \nu\overline{d'}\,\langle \Delta_1', \mathcal{Q}_2' \rangle / f \mid \prod_{c_i' \in \{\overline{c'}\}} c_i'?.\mathtt{0} \tag{5}$$

$$\nu\overline{b'}\,\langle \Delta_1', \mathcal{Q}_0' \mid \prod_{\alpha_i \in \{\overline{\alpha}\}} \mathtt{app}\,\alpha_i \rangle \stackrel{\tau}{\Longrightarrow} \nu\overline{d'}\,\langle \Delta_1', \mathcal{Q}_2' \mid \prod_{\alpha_i' \in \{\overline{\alpha'}\}} \mathtt{app}\,\alpha_i' \rangle \tag{6}$$

Hence by (4) and (6)

$$\begin{aligned} \nu\overline{a'}\,\langle \Delta_1', \mathcal{P}' \rangle &\stackrel{\tau}{\Longrightarrow} \nu\overline{b'}\,\langle \Delta_1', \mathcal{Q}_0' \mid \mathtt{app}\,\alpha \mid \textstyle\prod_{\alpha_i \in \{\overline{\alpha}\}} \mathtt{app}\,\alpha_i \rangle \\ &\stackrel{\mathtt{app}\,\alpha}{\longrightarrow} \nu\overline{b'}\,\langle \Delta_1', \mathcal{Q}_0' \mid \mathtt{0} \mid \textstyle\prod_{\alpha_i \in \{\overline{\alpha}\}} \mathtt{app}\,\alpha_i \rangle \\ &\stackrel{\tau}{\Longrightarrow} \nu\overline{d'}\,\langle \Delta_1', \mathcal{Q}_2' \mid \mathtt{0} \mid \textstyle\prod_{\alpha_i' \in \{\overline{\alpha'}\}} \mathtt{app}\,\alpha_i' \rangle \end{aligned}$$

Furhtermore, by (1) and (5), the fact that $(\equiv) \subseteq (\simeq) \subset (\cong_{\mathrm{pcxt}})$, and Lemma 6.8 we have

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

Hence,

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

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

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

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

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

  otherwise. Here $\mathtt{if}\, x \in \mathit{names}(\Delta_1)\, \mathtt{then}\, P\, \mathtt{else}\, Q$ is expressible in terms of $\mathtt{if}\, x = n_i\, \mathtt{then}\, P\, \mathtt{else}\, Q$ 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

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

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

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

  where c_{fr} is a fresh name.

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

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

  where c_{fr} is a fresh name.

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

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

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

Proposition 6.12. *$\mathbb{X}$ is a weak bisimulation.*

Proof. By Propositions 6.10 and 6.11 and by symmetry, $\mathbb{X}$ satisfies the conditions of Definition 4.4 for a weak bisimulation. □

Theorem 6.13 (Completeness)**.** $(\cong_{\text{pcxt}}) \subseteq (\simeq)$.

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

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

By Definition 6.9 $\langle\{\overline{n}\}, P\rangle\ \mathbb{X}\ \langle\{\overline{n}\}, P'\rangle$ and by Proposition 6.12, $\langle\{\overline{n}\}, P\rangle \approx \langle\{\overline{n}\}, P'\rangle$. Thus, by Definition 4.7, $P \simeq P'$. □

7 Full Contextual Equivalence

In this section we study reduction-closed barbed congruence ($\cong_{\text{cxt}}$) with arbitrary contexts. We show that weak bisimilarity implies reduction-closed barbed congruence and therefore, as with the first-order *picalculus*, parallel contextual equivalence coincides with reduction-closed barbed congruence for pp-π.

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

First we give the definition for contexts.

Definition 7.1 (Contexts)**.** *A context C is derived from the following grammar:*

$$\begin{aligned} C &::= [\cdot] \mid \mathtt{0} \mid V_c!\langle V_C{:}t\rangle.C \mid V_c?(x{:}t).C \mid C \mid C \\ &\quad \mid\ \nu n.\, C \mid \mathtt{app}\, V_c \mid *(C) \mid \mathtt{if}\, V_c = V_c\, \mathtt{then}\, C\, \mathtt{else}\, C \\ V_C &::= x \mid \lambda C \mid n \mid \mathtt{bv} \end{aligned}$$

We write $C[P]$ (resp. $V_C[P]$) to mean the replacement of all holes in C (resp. V_C) with P.

Cxt-C

$$\frac{\langle \{\overline{n}\}, P\rangle \;\mathbb{R}\; \langle \{\overline{n}\}, P'\rangle}{\nu\overline{a}\,\langle \{\overline{n}, \overline{\kappa \mapsto \mathcal{V}[P]}\}, C[P]\rangle \;\mathbb{R}^{\mathsf{cxt}}\; \nu\overline{a}\,\langle \{\overline{n}, \overline{\kappa \mapsto \mathcal{V}[P']}\}, C[P']\rangle}$$

Figure 7: Context Closure.

Definition 7.2 (Reduction-Closed Barbed Congruence ($\cong_{\mathrm{cxt}}$))**.** ($\cong_{\mathrm{cxt}}$) *is the largest congruence on closed processes that preserves barbs and is reduction closed; i.e.* $P \cong_{\mathrm{cxt}} P'$ *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* $P_1 \cong_{\mathrm{cxt}} P'_1$, *and vice-versa, and*

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

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

Proposition 7.3. ($\cong_{\mathrm{cxt}}$) $\subseteq$ ($\cong_{\mathrm{pcxt}}$).

We will show that ($\simeq$) $\subseteq$ ($\cong_{\mathrm{cxt}}$). As we proved in Section 5.2, ($\simeq$) is reduction-closed and barb-preserving. Therefore it suffices to show that ($\simeq$) preserves contexts. For this proof we extend the definition of Parallel Context Closure (Definition 5.5) by adding the rule of Figure 7. Bisimilarity is closed under this new $(\ )^{\mathsf{cxt}}$.

Theorem 7.4 (Context Closure of ($\approx$))**.** $(\approx)^{\mathsf{cxt}} \subseteq (\approx)$.

Proof. We show that $(\approx)^{\mathsf{cxt}}$ is a weak bisimulation up to limited structural equivalence ($\hat{=}$). The proof proceeds as the one of Theorem 5.7, by induction on the rules of $(\approx)^{\mathsf{cxt}}$. The only new proof obligation is the case Cxt-C shown in Figure 7.

We proceed by induction on C. The cases $\mathtt{0}$, $\nu n.C$, $c?(x{:}t).C$, $c!\langle n{:}t\rangle.C$, $*(C)$, $\mathsf{app}\,\alpha$, $\mathsf{app}\,\lambda C$, and $\mathtt{if}\,b = c\,\mathtt{then}\,C_1\,\mathtt{else}\,C_2$, are easy.

Let $\Delta = \{\overline{\kappa \mapsto \mathcal{V}[P]}\}$, $\Delta' = \{\overline{\kappa \mapsto \mathcal{V}[P']}\}$, and $\langle \{\overline{n}\}, P\rangle \approx \langle \{\overline{n}\}, P'\rangle$.

Case $[\cdot]$: we need to show that if $\nu\overline{a}\,\langle \Delta \uplus \{\overline{n}\},\, P\rangle \xrightarrow{\eta} \nu\overline{b}\,\langle \Delta_2,\, Q\rangle$ then there exist $\overline{b'}$, Δ'_2, and Q' such that

$$\nu\overline{a}\,\langle \Delta' \uplus \{\overline{n}\},\, P'\rangle \stackrel{\eta}{\Longrightarrow} \nu\overline{b'}\,\langle \Delta'_2,\, Q'\rangle \qquad \nu\overline{b}\,\langle \Delta_2,\, Q\rangle\; (\approx)^{\mathsf{cxt}}\; \nu\overline{b'}\,\langle \Delta'_2,\, Q'\rangle$$

By Proposition 3.3, there exist Δ_1 and Δ'_1 such that $\Delta_2 = \Delta \uplus \Delta_1$ and $\Delta'_2 = \Delta' \uplus \Delta'_1$.

We distinguish three cases for η.

♦ $\eta = \mathsf{app}\,\kappa_i$ and $\kappa_i \in \Delta$: we have

$$\nu\overline{a}\,\langle \Delta \uplus \{\overline{n}\},\, P\rangle \xrightarrow{\eta} \nu\overline{a}\,\langle \Delta \uplus \{\overline{n}\},\, P \mid \mathsf{app}\,V_i[P]\rangle$$
$$\nu\overline{a}\,\langle \Delta' \uplus \{\overline{n}\},\, P\rangle \xrightarrow{\eta} \nu\overline{a}\,\langle \Delta' \uplus \{\overline{n}\},\, P' \mid \mathsf{app}\,V_i[P']\rangle$$
$$\nu\overline{a}\,\langle \Delta \uplus \{\overline{n}\},\, P \mid \mathsf{app}\,V_i[P]\rangle\; (\approx)^{\mathsf{cxt}}\; \nu\overline{a}\,\langle \Delta' \uplus \{\overline{n}\},\, P' \mid \mathsf{app}\,V_i[P']\rangle$$

♦ $\eta = n_i?n_j$ and $n_j \in \mathit{names}(\Delta)$: vacuously true since Δ does not contain any names.

♦ Otherwise: by Proposition 3.5 (ii) there exist $\overline{c}$ such that $\{\overline{c}\} = \{\overline{b}\}\backslash\{\overline{a}\}$ and

$$\langle\{\overline{n}\}, P\rangle \xrightarrow{\eta} \nu\overline{c}\,\langle\Delta_1, Q\rangle$$

and because $\langle\{\overline{n}\}, P\rangle \approx \langle\{\overline{n}\}, P'\rangle$ there exist $\overline{c'}$, Δ'_1, and Q' such that

$$\langle\{\overline{n}\}, P'\rangle \stackrel{\eta}{\Longrightarrow} \nu\overline{c'}\,\langle\Delta'_1, Q'\rangle$$
$$\nu\overline{c}\,\langle\Delta_1, Q\rangle \approx \nu\overline{c'}\,\langle\Delta'_1, Q'\rangle$$

By Proposition 3.5 (i)

$$\nu\overline{a}\,\langle\Delta' \uplus \{\overline{n}\},\ P'\rangle \stackrel{\eta}{\Longrightarrow} \nu\overline{a}, \overline{c'}\,\langle\Delta' \uplus \Delta'_1,\ Q'\rangle$$

and because $\nu\overline{a}\,\langle\Delta,\ \mathtt{0}\rangle\ (\approx)^{\mathsf{cxt}}\ \nu\overline{a}\,\langle\Delta',\ \mathtt{0}\rangle$ we get

$$\nu\overline{a}, \overline{c}\,\langle\Delta \uplus \Delta_1,\ Q\rangle \mathrel{\hat{=}} (\approx)^{\mathsf{cxt}} \mathrel{\hat{=}} \nu\overline{a}, \overline{c'}\,\langle\Delta' \uplus \Delta'_1,\ Q'\rangle$$

Case $C_1 \mid C_2$: by Lemma 4.11

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

By the induction hypothesis

$$\langle\Delta \uplus \{\overline{n}, \overline{a}\}, C_1[P]\rangle \approx \langle\Delta' \uplus \{\overline{n}, \overline{a}\}, C_1[P']\rangle$$
$$\langle\Delta \uplus \{\overline{n}, \overline{a}\}, C_2[P]\rangle \approx \langle\Delta' \uplus \{\overline{n}, \overline{a}\}, C_2[P']\rangle$$

and by Theorem 5.7

$$\langle\Delta \uplus \{\overline{n}, \overline{a}\}, C_1[P] \mid C_2[P]\rangle \approx \langle\Delta' \uplus \{\overline{n}, \overline{a}\}, C_1[P'] \mid C_2[P']\rangle$$

and again by the same lemma

$$\nu\overline{a}\,\langle\Delta \uplus \{\overline{n}\},\ C_1[P] \mid C_2[P]\rangle \approx \nu\overline{a}\,\langle\Delta' \uplus \{\overline{n}\},\ C_1[P'] \mid C_2[P']\rangle \qquad \square$$

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

Theorem 7.5 (Compositionality)**.** *If $P \simeq P'$ then for any context C, $C[P] \simeq C[P']$.*

Theorem 7.6 (Soundness w.r.t. $(\cong_{\mathrm{cxt}})$)**.** $(\simeq) \subseteq (\cong_{\mathrm{cxt}})$

Proof. In Propositions 5.4 and 5.3 and Corollary 7.5 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 $(\cong_{\mathrm{cxt}})$. $\square$

Theorem 7.7. $(\cong_{\mathrm{pcxt}}) = (\cong_{\mathrm{cxt}})$.

Proof. By the definitions of $(\cong_{\mathrm{cxt}})$ and $(\cong_{\mathrm{pcxt}})$ we have $(\cong_{\mathrm{cxt}}) \subseteq (\cong_{\mathrm{pcxt}})$, by Theorem 6.13 we have $(\cong_{\mathrm{pcxt}}) \subseteq (\simeq)$, and by Theorem 7.6 we have $(\simeq) \subseteq (\cong_{\mathrm{cxt}})$. Hence,

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

and therefore $(\cong_{\mathrm{pcxt}}) = (\cong_{\mathrm{cxt}})$. $\square$

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 $(\cong_{\mathrm{cxt}})$.

8 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 $(\hat{=})$ technique similar to that in [1], Chapter 6.

β-moves are τ-transitions that are confluent with all other transitions.

Definition 8.1 (β-move ($\overset{\tau_\beta}{\longrightarrow}$))**.** *A τ-transition $C_1 \overset{\tau}{\longrightarrow} C_2$ is a β-move and we write $C_1 \overset{\tau_\beta}{\longrightarrow} C_2$ if and only if for all transitions $C_1 \overset{\eta}{\longrightarrow} C_3$ one of the following is true:*

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

(ii) there exists C_4 such that $C_2 \overset{\eta}{\longrightarrow} C_4$ and $C_3 \overset{\tau}{\longrightarrow} C_4$.

Definition 8.2 (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 $C \overset{\eta}{\longrightarrow} C_1$ then, for some C_1',

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

(ii) the converse of (i)

Proposition 8.3. *The relation $(\overset{\tau_\beta}{\longrightarrow}^* \hat{=})$ 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 8.4. *If $\mathbb{R}$ is a weak bisimulation up-to β and $(\hat{=})$, then $\mathbb{R} \subseteq (\overset{\tau_\beta}{\longrightarrow}^* \hat{=}\ \mathbb{R}\ \hat{=} \overset{\tau_\beta}{\longleftarrow}^*) \subseteq (\approx)$.*

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

Let

$$C_1 \overset{\tau_\beta}{\longrightarrow}^* C_2 \hat{=} C_3\ \mathbb{R}\ C_3' \hat{=} C_2' \overset{\tau_\beta}{\longleftarrow}^* C_1'$$

then

	$C_1 \overset{\eta}{\longrightarrow} C_4$	
implies	$C_2 \overset{\eta}{\longrightarrow} C_5 \ \wedge\ C_4 \overset{\tau_\beta}{\longrightarrow}^* C_5$	(for some C_5, by definition of $(\overset{\tau_\beta}{\longrightarrow})$)
implies	$C_3 \overset{\eta}{\longrightarrow} C_6 \ \wedge\ C_5 \hat{=} C_6$	(for some C_6, because $(\hat{=})$ is a strong bisimulation)
implies	$C_3' \overset{\eta}{\Longrightarrow} C_6' \ \wedge\ C_6 \overset{\tau_\beta}{\longrightarrow}^* \hat{=}\ \mathbb{R}\ \hat{=} C_6'$	(for some C_6', because $\mathbb{R}$ is a weak bisimulation up-to β and $(\hat{=})$)
implies	$C_2' \overset{\eta}{\Longrightarrow} C_5' \ \wedge\ C_6' \hat{=} C_5'$	(for some C_5', because $(\hat{=})$ is a strong bisimulation)
implies	$C_1' \overset{\eta}{\Longrightarrow} C_5'$	(because τ_β-moves are τ-steps)

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

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

and by Proposition 8.3 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.

8.1 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.

$$Rec \overset{\text{def}}{=} p?(X).\nu a.\,(R \mid a!\langle \lambda\, \mathtt{app}\, X \mid R\rangle.\mathtt{0})$$

$$R \overset{\text{def}}{=} a?(X).(\mathtt{app}\, X \mid a!\langle X\rangle.\mathtt{0})$$

We show that this is weakly bisimilar to

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

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.

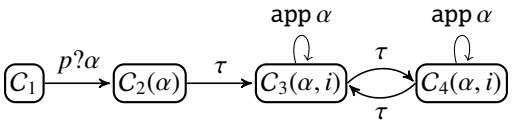
$$\begin{aligned}
C_1 &\mathrel{\hat{=}} \langle\{p\}, Rec\rangle \\
C_2(\alpha) &\mathrel{\hat{=}} \langle\{p,\alpha\}, \nu a.\ R \mid a!\langle\lambda(\mathtt{app}\,\alpha \mid R)\rangle.\mathtt{0}\rangle \\
C_3(\alpha,i) &\mathrel{\hat{=}} \nu a\,\langle\{p,\alpha\},\ R \mid a!\langle\lambda(\mathtt{app}\,\alpha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

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

$$\begin{aligned}
C_1' &\mathrel{\hat{=}} \langle\{p\}, Rec'\rangle \\
C_2'(\alpha,i) &\mathrel{\hat{=}} \langle\{p,\alpha\}, *(\mathtt{app}\,\alpha) \mid \textstyle\prod_i \mathtt{app}\,\alpha\rangle
\end{aligned}$$

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.

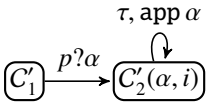
Here the possible-worlds structure that corresponds to Rec is



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 variable 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 $\lambda(\mathsf{app}\,\alpha \mid R)$ over the channel a. The remaining τ-arrow is a result of the application of $\lambda(\mathsf{app}\,\alpha \mid R)$: $C_4(\alpha, i) \xrightarrow{\tau} C_3(\alpha, i+1)$ that produces one more process $\mathsf{app}\,\alpha$. The self-loops on the configurations C_3 and C_4, labelled $\mathsf{app}\,\alpha$, are because of the transitions that apply a process $\mathsf{app}\,\alpha$:

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

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

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

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 $C_3(\alpha, i+1) \xrightarrow{\mathsf{app}\,\alpha} C_3(\alpha, i)$ and $C_4(\alpha, i+1) \xrightarrow{\mathsf{app}\,\alpha} C_4(\alpha, i)$ are matched by $C'_2(\alpha, i+1) \xrightarrow{\mathsf{app}\,\alpha} C'_2(\alpha, i)$. All resulting configurations of matching transitions are related in $\mathbb{R}$.

8.2 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 7.5), it suffices to show that for the processes under the replication

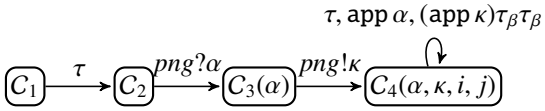
$$\begin{aligned} M_1 &\stackrel{\text{def}}{=} \nu tr.\, P_1(tr) \\ M_2 &\stackrel{\text{def}}{=} png?(X{:}\mathtt{Pr}).png!\langle X\rangle.\mathtt{0} \end{aligned}$$

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

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

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 variable α and transition $C_3(\alpha) \xrightarrow{png!\kappa} C_4(\alpha, \kappa, 0, 0)$ is the subsequent output of the concrete variable 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.

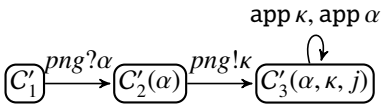
(ii) $C_4(\alpha,\kappa,i+1,j) \xrightarrow{\mathtt{app}\,\kappa}\xrightarrow{\tau_\beta}\xrightarrow{\tau_\beta} C_4(\alpha,\kappa,i,j+1)$ is an $\mathtt{app}\,\kappa$-transition, which puts $\mathtt{app}\,\lambda tr!.\mathtt{0}$ in parallel with the current process, followed by two β-moves. The first 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 variable α.

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

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

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 ($\xrightarrow{\tau_\beta}{}^* \hat{=} \mathbb{R} \hat{=}$).

8.3 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, it 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

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

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

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

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

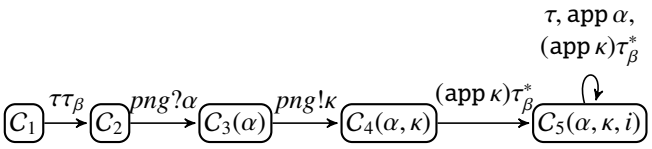
$$M_3 \stackrel{\text{def}}{=} \nu inst.\, \nu tr.\, png?(X{:}\mathtt{Pr}).png!\langle\mathcal{U}(X, inst, tr)\rangle.inst!.\mathbb{0}$$

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 $\langle\{png\}, M_3\rangle$. Here we omit subscripts to parallel products that do not affect the equivalence between configurations.

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

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 variable 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 $(\triangleq)$.

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

8.4 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 8.3, and

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

[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.

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 ($\hat{=}$).

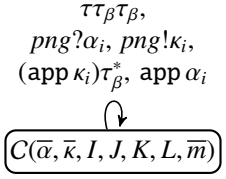
We devise the following family of configurations that describes the configurations that are reachable from $\langle\{png\}, Ping_3\rangle$ and relevant to the bisimulation proof.

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

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 8.3. 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 variables, and concrete variables that correspond to the different instances of C_2 to C_5, respectively. Moreover, the transitions that in Section 8.3 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.

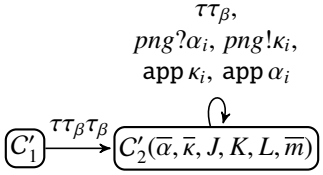


The configurations reachable from $\langle\{png\}, Ping_4\rangle$ and relevant to this bisimulation proof belong in the following families of configurations.

$$
\begin{aligned}
&C'_1 \hat{=} \langle\{png\}, rec(M_4)\rangle \\
&C'_2(\overline{\alpha}, \overline{\kappa}, J, K, L, \overline{m}) \\
&\quad \hat{=} \nu a\, \langle\, \{png, \overline{\alpha}, \overline{\kappa \mapsto \alpha}^{K \cup L}\},\ R \mid a!\langle \lambda\, M_4 \mid R\rangle.\mathbb{0} \\
&\qquad \mid \textstyle\prod M_4 \mid \prod_{j \in J} png!\langle \alpha_j\rangle.\mathbb{0} \mid \prod_{l \in L} (\prod_{m_l} \mathtt{app}\,\alpha_l)\rangle
\end{aligned}
$$

Notice that C'_2 is similar to configuration C_3 of Section 8.1. C_4 is not necessary here because of the use of the up-to β technique.

The abstract LTS is the following.



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

$$\begin{aligned}\mathbb{R} = \{\ & (C(\cdot,\cdot,\emptyset,\emptyset,\emptyset,\emptyset,\cdot), C'_1),\\ & (C(\overline{\alpha},\overline{\kappa},I,J,K,L,\overline{m}), C'_2(\overline{\alpha},\overline{\kappa},J,K,L,\overline{m})),\\ & \mid \overline{\alpha},\overline{\kappa},\overline{m}, \forall k \in K.\ i_k + j_k + l_k = m_k,\\ & I,J,K,L \text{ pairwise disjoint } \}\end{aligned}$$

8.5 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:

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

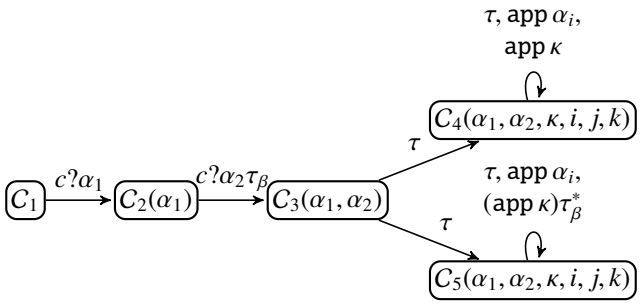
The equivalence. First we consider the equivalence (†) in Figure 1. We will show that $P \simeq P'$; i.e. we will show $\langle\{c\}, P\rangle \approx \langle\{c\}, P'\rangle$, where

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

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

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

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 inequivalence ($\ddagger$) in Figure 1, written in pp-π:

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

Let us assume that we match the output

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

on the left-hand side with the output

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

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 $\langle\{c\}, Q\rangle$ but not by $\langle\{c\}, Q'\rangle$. This formula is

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

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

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

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

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

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

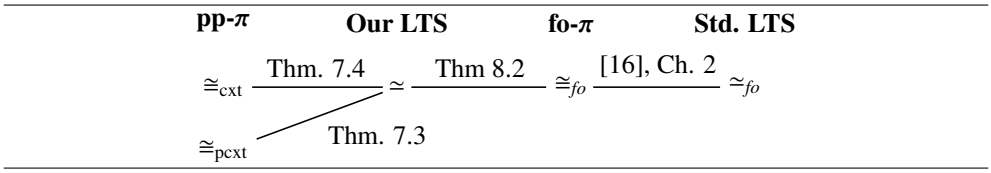


Figure 8: The big picture

9 First-Order Processes

The first-order *picalculus*, from Chapter 1 of [16], 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 7.2: 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 4.7: 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 [3, 16].

For the purpose of analysis let us now introduce a fourth [3].

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 ($\cong_{fo}$) ([16], Chapter 2); we show that ($\simeq$) also coincides with ($\cong_{fo}$):

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

Proof. (Outline) A very easy adaptation of Theorems 5.9 and 6.13 □

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 8. 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

The main achievement of this paper has been a simple and effective proof technique for program equivalence in a higher-order setting, which reduces reasoning about higher-order processes to checking Kripke-like structures. We have concentrated on the relatively straightforward higher-order language pp-π in order to emphasise the challenges of pure higher-order concurrency but we believe it can also be adapted to HOπ and other higher-order concurrent languages.

The proof technique, first-order in nature, has emerged not from a completely new behavioural theory, but from a combination and improvement of existing theories, particularly those in [4] and [15]. Compared to [15], our method does not require an up-to context or other techniques to effectively reason about higher-order processes. In particular, a proof of the example in Section 8.4 using environmental bisimulations up-to context would require a manual induction on contexts; a stronger up-to technique (e.g. up-to context and reduction) would be required to remove the need for such an induction. It is unclear, though, if up-to techniques can always mitigate the strong proof obligations due to the quantification over contexts that test related inputs in [15]. Compared to [4], our use of knowledge environments simplifies the labelled transition system by removing extrusion from its labels, thus allowing the characterisation of (weak) bisimulation by a propositional Hennessy-Milner Logic. Using this logic we are able to easily prove inequivalences of higher-order processes by providing a distinguishing formula.

References

[1] M. Hennessy. *A Distributed Picalculus*. Cambridge University Press, 2007.

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

[3] K. Honda and N. Yoshida. On reduction-based process semantics. *Theoretical Computer Science*, 151(2):437 – 486, 1995.

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

[5] A. Jeffrey and J. Rathke. Full abstraction for polymorphic pi-calculus. In *Proc. Foundations of Software Science and Computation Structures (FoSSaCS)*, volume 3441 of *Lecture Notes in Computer Science*, pages 266–281. Springer-Verlag, 2005.

[6] V. Koutavas and M. Wand. Bisimulations for untyped imperative objects. In P. Sestoft, editor, *Proc. 15th European Symposium on Programming (ESOP), Programming Languages and Systems*, volume 3924 of *Lecture Notes in Computer Science*, pages 146–161. Springer-Verlag, 2006.

[7] V. Koutavas and M. Wand. Small bisimulations for reasoning about higher-order imperative programs. In *Proc. 33rd ACM SIGPLAN-SIGACT symposium on Principles of Programming Languages (POPL)*, pages 141–152. ACM Press, 2006.

[8] V. Koutavas and M. Wand. Reasoning about class behavior. In International Workshop on Foundations and Developments of Object-Oriented Languages (FOOL/WOOD), January 2007.

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

[10] J. H. Morris, Jr. *Lambda Calculus Models of Programming Languages*. PhD thesis, MIT, Cambridge, MA, USA, 1968.

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

[12] D. Sangiorgi. From pi-calculus to higher-order pi-calculus – and back. In M.-C. Gaudel and J.-P. Jouannaud, editors, *Proc. International Joint Conference CAAP/FASE on Theory and Practice of Software Development (TAPSOFT)*, Lecture Notes in Computer Science, pages 151–166. Springer-Verlag, 1993.

[13] D. Sangiorgi. Bisimulation for higher-order process calculi. *Information and Computation*, 131(2):141–178, 1996.

[14] D. Sangiorgi. Bisimulation: From the origins to today. In *Proc. 19th IEEE Symposium on Logic in Computer Science (LICS)*, pages 298–302. IEEE Computer Society, 2004.

[15] D. Sangiorgi, N. Kobayashi, and E. Sumii. Environmental bisimulations for higher-order languages. In *Proc. 22th Annual IEEE Symposium on Logic in Computer Science (LICS)*, pages 293–302. IEEE Computer Society, 2007.

[16] D. Sangiorgi and D. Walker. *The π-calculus: a Theory of Mobile Processes*. Cambridge University Press, 2001.

[17] Colin Stirling. Bisimulation, modal logic and model checking games. *Logic Journal of the IGPL*, 7(1):103–124, 1999.

[18] E. Sumii and B. C. Pierce. A bisimulation for dynamic sealing. *Theoretical Computer Science*, 375(1–3):169–192, May 2007.

[19] E. Sumii and B. C. Pierce. A bisimulation for type abstraction and recursion. *Journal of the ACM*, 54(5):1–43, 2007.