UNIVERSITY OF SUSSEX

COMPUTER SCIENCE

Typed behavioural equivalences for processes in the presence of subtyping

Typed behavioural equivalences for processes in the presence of subtyping

Matthew Hennessy and Julian Rathke

Abstract. We study typed behavioural equivalences for the π-calculus, in which the type system allows a form of subtyping. This enables processes to selectively distribute different capabilities on communication channels.

The equivalences considered include typed versions of testing equivalences and barbed bisimulation equivalences.

We show that these can be characterised via standard techniques applied to a novel labelled transition system of *configurations*. These consist of a process term together with two related type environments; one constraining the process and the other its computing environment.[1]

1 Introduction

Type systems are playing an increasingly important role in the theory of distributed systems. They are essentially a form of static analysis which help in the elimination of run-time errors from programs. Within the theory of distributed systems this intuitive notion of run-time error has been extended to include a diverse range of properties. For example in [12, 3] type systems have been designed to detect potential deadlocks while [18] introduced a system of types for the π-calculus which are used to control the interpretation of the λ-calculus. This system of types was extended further in [17] and now forms the basis for the powerful type system implemented in the programming language **Pict**, [19]; related type systems for higher-order concurrent languages may be found in [10, 11]. In papers such as [21, 20] types have been used to manage access control to resources, while in [22] notions of trust have been incorporated in order to protect good host sites from bad computing agents.

Sub-typing is an essential part of most of these systems. For example in **Pict** (according to [19], page 9) it is relatively rare for communication channels to be used for both input and output in the same "region" of a program. Typically servers have one form of access while clients require a different form. These access requirements can be implemented and managed using a subtype relation on the set of types. For example a particular channel may be declared with a type which allows both read and write

[1] esearch partially funded by EPS C grant G /M71169

access; this channel could be passed to one process, say a server, at a subtype which only allows read, or input access, and passed to a client at a different subtype, allowing write, or output access only. Indeed in papers such as [21, 25] types are viewed as sets of capabilities, such as read access and write access, and sending a name to a process at a subtype amounts to sending it with a subset of the declared capabilities.

The subject of this paper is the investigation of behavioural equivalences for typed process languages, particularly in the

$T,\ U$ ::=		**Terms**
	$u?(X : \mathrm{A})\, T$	Input
	$u!\langle v\rangle\, T$	Output
	if $u = v$ th n T ls U	Matching
	(n w $n : \mathrm{A}$) T	Name Creation
	$T \mid T$	Concurrency
	$*T$	Repetition
	0	Termination
X, Y ::=		**Patterns**
	x	variable
	$(X_1, \ldots, X_n)$	tuple
u, v, w ::=		**Values**
	bv	base value
	n	name
	x	variable
	$(u_1, \ldots, u_n)$	tuple

FIGURE 1. The Syntax

review our version of the π-calculus, which uses a set of types derived from those in [21], although they are only a minor variation of those from [18]; the section contains a standard operational semantics, in terms of an lts, that is a *labelled transition system*, a type inference system and a statement of Subject Reduction. In Section 3 we define the typed behavioural equivalences which are the main concern of the paper. This is followed by the principal section of the paper, Section 4, where we define the set *typed actions* which gives rise to the lts **Conf**; this section also contains an analysis of **Conf** and proofs of the various properties we require of it. This is followed by two technical sections, Section 5 which contains a characterisation of the typed testing equivalences, and Section 6

(L-OUT)

$$\frac{}{a!\langle v\rangle\, P \xrightarrow{a!v} P}$$

(L-IN)

$$\frac{}{a?(X : \mathrm{A})\, P \xrightarrow{a?v} P\{\!|v/X|\!\}}$$

(L-OPEN)

$$\frac{P \xrightarrow{(\tilde{c}:\tilde{\mathrm{C}})a!v} P'}{(\mathsf{n}\ \ \mathsf{w}}$$

Typed

$$\text{(T-ID)}\ \frac{\Gamma(u) <: \mathrm{A}}{\Gamma \vdash u : \mathrm{A}} \qquad \text{(T-BASE)}\ \frac{bv \in \mathbf{Base}}{\Gamma \vdash bv : \mathbf{Base}} \qquad \text{(T-TUP)}\ \frac{\Gamma \vdash v_i : A_i \quad (\forall i)}{\Gamma \vdash (v_1, \ldots, v_k) : (\mathrm{A}_1, \ldots, \mathrm{A}_k)}$$

$$\text{(T-IN)}\ \frac{\begin{array}{l}\Gamma, X : \mathrm{A} \vdash T \\ \Gamma \vdash u : \mathsf{r}\langle \mathrm{A} \rangle\end{array}}{\Gamma \vdash u?(X : \mathrm{A})\, T} \qquad \text{(T-OUT)}\ \frac{\begin{array}{l}\Gamma \vdash u : \mathsf{w}\langle \mathrm{A} \rangle \\ \Gamma \vdash v : \mathrm{A} \\ \Gamma \vdash T\end{array}}{\Gamma \vdash u!\langle v \rangle\, T} \qquad \text{(T-EQ)}\ \frac{\begin{array}{l}\Gamma \vdash u : \mathrm{A}, v : \mathrm{B} \\ \Gamma \vdash U \\ \Gamma \sqcap \{u : \mathrm{B}, v : \mathrm{A}\} \vdash T\end{array}}{\Gamma \vdash \mathsf{if}\ u = v\ \mathsf{th\ n}\ T\ \ \mathsf{ls}\ \ U}$$

$$\text{(T-NEW)}\ \frac{\Gamma, a : \mathrm{A} \vdash T}{\Gamma \vdash (\mathsf{n\ w}\, a : \mathrm{A})\, T} \qquad \text{(T-STR)}\ \frac{\Gamma \vdash T,\ U}{\Gamma \vdash T \mid U,\ *T,\ \mathbf{0}}$$

FIGURE 4. The Typing ules

of the π-calculus has name matching and therefore a name at type $\top$ can be compared to other names.

Thus our types are a generalisation of those introduced in [18]. The subtyping relation $<:$ can also be viewed as the obvious generalisation of their subtyping relation. In fact our types, and our subtyping relation, are a mild variation of those used in [21], to which the reader is referred for more details, particularly with respect to the following result:

Proposition 2.1 *The set of types* **Types** *is a preorder with respect to* $<:$, *with both a partial meet operation* $\sqcap$ *and a partial join* $\sqcup$. $\square$.

The essential point here is that if two types $\mathrm{A}_1, \mathrm{A}_2$ are bounded below, that is $B <: \mathrm{A}_1,\ B <: \mathrm{A}_1$ for some type B then they have a greatest lower bound, $\mathrm{A}_1 \sqcap \mathrm{A}_2$. Intuitively $\mathrm{A}_1 \sqcap \mathrm{A}_2$ is the "union of the capabilities" in A_1 and A_2. Because the write capability $\mathsf{w}\langle - \rangle$ is contravariant with respect to $<:$ the definition of $\sqcap$ requires the existence of a partial join $\sqcup$.

We now present the type inference rules for process terms in Figure 4. The judgements are of the form $\Gamma \vdash T$ where Γ is a *type environment*, that is a finite mapping from *identifiers*, variables and names, to types.

For an identifier *id* we write $\Gamma, id : \mathrm{A}$ for the type environment obtained by augmenting Γ so as to map *id* to A; this notation is only defined if *id* is not already in the domain of Γ. More generally we use $\Gamma \sqcap id : \mathrm{A}$ to mean the type environment $\Gamma, id : \mathrm{A}$ if *id* is not in the domain of Γ and Γ' otherwise, where Γ' is equal to Γ except possibly at *id*, where Γ' takes the value $\Gamma(id) \sqcap A$ (if defined). This notation is generalised in the obvious way to values. We will often write Δ for *closed* type environments whose domain consists solely of names.

The reader familiar with the input/output capability types of π-calculus, [18], should find little surprise in the inference rules except perhaps for the type rule for conditionals, taken from [21]:

(T-EQ)

$\Gamma \vdash u : \mathrm{A}, v : \mathrm{B}$

$\Gamma \vdash U$

$\Gamma \sqcap \{$

necessarily known to the current type environment Γ, although it does not allow us to extend the types of values which are already in the domain of Δ. However even on closed terms there may be a difference between a relation $\mathcal{R}$ and its open extension $\mathcal{R}^o$; in general for $\Delta \models P\ \mathcal{R}^o$ to be true we must have $\Delta, \Delta' \models P\ \mathcal{R}$ for every allowed Δ'. Note that this is a form of *weakening.*

Definition 3.3 *A typed relation $\mathcal{R}$ is said to* closed with respect to weakening*, or* w-closed*, if $\mathcal{R}^o = \mathcal{R}$.*

All the behavioural equivalences we will consider will be *w-closed.* to define these we need to consider a number of properties of typed relations.

Reduction closed: The typed relation $\mathcal{R}$ is reduction closed whenever $\Delta \models P\ \mathcal{R}$ and $P \xrightarrow{\tau} P'$ implies there exists some $'$ such that $\Longrightarrow\ '$ and $\Delta \models P'\ \mathcal{R}\ '$.

Contextual: Contexts are defined by extending the syntax in Figure 1, allowing *typed holes* $[\cdot_\Gamma]$ in terms. The typing system in Figure 4 is extended to contexts in the obvious way, by adding the rule

$$\text{(T-CXT)} \qquad \frac{}{\Gamma, \Gamma' \vdash [\cdot_\Gamma]}$$

We use $C[\,]$ to denote contexts with at most one hole and $C[T]$ the term which results from substituting the term T into the hole. We leave the reader to establish

Proposition 3.4 $\Gamma' \vdash T$ *and* $\Gamma \vdash C[\cdot_{\Gamma'}]$ *implies* $\Gamma \vdash C[T]$. $\square$

Then we say the typed relation $\mathcal{R}$ is contextual whenever $\Gamma' \models T\ \mathcal{R}^o\ U$ and $\Gamma \vdash C[\cdot_{\Gamma'}]$ implies $\Gamma \models C[T]\ \mathcal{R}^o\ C[U]$.

Unravelling this definition gives the following example consequences for contextual relations over closed terms.

- $\Delta \models P\ \mathcal{R}\ P'$ implies $\Delta, \Delta' \models P\ \mathcal{R}\ P'$
- $\Delta \models P\ \mathcal{R}\ P'$ and $\Delta \vdash$ implies $\Delta \models P \mid \ \ \mathcal{R}\ P' \mid$.
- $\Delta \models P\ \mathcal{R}\ P'$ and $\Delta \vdash a!\langle v\rangle\,\mathbf{0}$ implies $\Delta \models a!\langle v\rangle\,P\ \mathcal{R}\ a!\langle v\rangle\,P'$.
- If $\Delta \vdash a : \mathsf{r}\langle \mathrm{A}\rangle$ and for every v, Δ', such that $\Delta, \Delta' \vdash v : \mathrm{A}$ we have $\Delta, \Delta' \models T\{\!|v/X|\!\}\ \mathcal{R}\ U\{\!|v/X|\!\}$ then $\Delta, \Delta' \models a?(X : \mathrm{A})\,T\ \mathcal{R}\ a?(X : \mathrm{A})\,U$.
- $\Delta, a : \mathrm{A} \models P\ \mathcal{R}\ P'$ implies $\Delta \models (\mathsf{n\ w}\,a : \mathrm{A})\ \ P\ \mathcal{R}\ \ (\mathsf{n\ w}\,a : \mathrm{A})$

that $\Delta, \Delta' \vdash v : \mathrm{A}$; this includes values v which are not known in the current environment Δ.

Barb Preserving: For a given name a such that $\Delta \vdash a : \mathsf{rw}\langle\top\rangle$. we write $\Delta \models P \Downarrow^{\text{barb}} a$ if there exists some P' such that $P \xrightarrow{\tau}^{*} P'$ and $P' \xrightarrow{a!\langle\rangle}$. Then we say the typed relation $\mathcal{R}$ is *barb preserving* if $\Delta \models P\ \mathcal{R}$ and $\Delta \models P \Downarrow^{\text{barb}} a$ implies Δ

(TYLTS-RED)

$$\frac{P \xrightarrow{\tau} P'}{\mathcal{I}; \Delta \vdash P \xrightarrow{\tau} \mathcal{I}; \Delta \vdash P'}$$

(TYLTS-OUT)

$$\frac{\mathcal{I}^r(a)\downarrow}{\mathcal{I}; \Delta \vdash a!\langle v\rangle\, P \xrightarrow{a!v} \mathcal{I} \sqcap v : \mathcal{I}^r(a); \Delta \vdash P}$$

(TYLTS-IN)

$$\frac{\mathcal{I}^w(a)\downarrow \qquad \mathcal{I} \vdash v : \mathcal{I}^w(a)}{\mathcal{I}; \Delta \vdash a?(X : \mathrm{A})\, P \xrightarrow{a?v} \mathcal{I}; \Delta \vdash P\{\!|v/X|\!\}}$$

(TYLTS-OPEN)

$$\frac{\mathcal{I}, b : \top; \Delta, b : \mathrm{B} \vdash P \xrightarrow{(\tilde{c})a!v} \mathcal{I}'; \Delta' \vdash P'}{\mathcal{I}; \Delta \vdash (\mathsf{n}\ \ \mathsf{w}\, b : \mathrm{B})\ P \xrightarrow{(b\tilde{c})a!v} \mathcal{I}'; \Delta' \vdash P'} \quad \begin{array}{l} b \neq a \\ b \in \mathsf{fn}(v) \end{array}$$

(TYLTS-CTXT)

$$\frac{\mathcal{I}; \Delta \vdash P \xrightarrow{\mu} \mathcal{I}'; \Delta' \vdash P'}{\mathcal{I}; \Delta \vdash *P \xrightarrow{\mu} \mathcal{I}'; \Delta' \vdash *P \mid P'}$$

(TYLTS-EQUIV)

$$\frac{\mathcal{I}; \Delta \vdash P \xrightarrow{\mu} \mathcal{I}'; \Delta' \vdash P'}{\mathcal{I}; \Delta \vdash \quad \xrightarrow{\mu} \mathcal{I}; \Delta' \vdash P'} \quad P \equiv_{\alpha} Q$$

$$\frac{\mathcal{I}; \Delta \vdash P \xrightarrow{\mu} \mathcal{I}'; \Delta' \vdash P'}{\begin{array}{l} \mathcal{I}; \Delta \vdash P \mid \quad \xrightarrow{\mu} \mathcal{I}'; \Delta' \vdash P' \mid \\ \mathcal{I}; \Delta \vdash \quad \mid P \xrightarrow{\mu} \mathcal{I}'; \Delta' \vdash \quad \mid P' \vdash \end{array}}$$

ό!

- *$\mathcal{I}$ is compatible with Δ*
- $\Delta \vdash T$

The environment $\mathcal{I}$ represents the environment's view of the types allocated to the names in the process. For this reason this view must accord with the actual types allocated to these names. This is guaranteed by requiring $\mathcal{I} :> \Delta$ where Δ is the actual type context for the term under investigation. Essentially this says that the environment cannot know capabilities for a channel which simply do not exist. The requirement that the domains of the environments be the same is a technical means of ensuring uniqueness of fresh names. We use **Conf**, ranged over by $\mathcal{C}, \mathcal{D}$, to denote the set of all configurations.

The generating rules for the transition system of *typed actions* are defined in Figure 5 and are to be understood as acting on configurations. The rules are obtained from those in Figure 2 by taking the type environment of the computing context, $\mathcal{I}$, into account; essentially actions are only possible if they are allowed by $\mathcal{I}$. Note also that the type annotations on the bound names of output actions are dropped; they are only required in Figure 2 for the definition of the untyped reduction relation $\xrightarrow{\tau}$.

Note also that a priori the rule (TYLTS-OUT) is partial in the sense that the conclusion can only be formed if the extended environment $\mathcal{I} \sqcap v : \mathcal{I}^r(a)$ is well defined. However the first part of the next Proposition establishes that this meet always exists. It also proves that the set of configurations is preserved by the transitions.

Proposition 4.2

- *If $\mathcal{I}^r(v)$ exists and $\Delta <: \mathcal{I}$ then $\mathcal{I} \sqcap v : \mathcal{I}^r(a)$, is always defined.*
- *If $\mathcal{I}; \Delta \vdash P \xrightarrow{\mu} \mathcal{I}'$*a:

This means that $\Delta(v) <: \mathrm{A}$ (by (ii)), and that $\mathrm{A} <: \Delta^w(a)$ (by (i)). We know that $\mathcal{I}^r(a)\downarrow$ and, as $\mathcal{I} :> \Delta$, thus $\Delta^r(a)\downarrow$ also. By virtue of being a well-formed type it must be the case that $\Delta^w(a) <: \Delta^r(a)$

We leave the reader to check that

Lemma 4.4 *If* $\mathcal{I};\Delta$

For the converse, part (ii), we use the case $\alpha = a?b$ as an illustrative example. Suppose then that $\mathcal{I}, \delta : \mathsf{rw}\langle(\mathcal{I})\rangle \vdash \mathcal{C}^{\mathcal{I}}_{\alpha}$ and

$$P \mid \mathcal{C}^{\mathcal{I}}_{\alpha} \Longrightarrow P' \mid \delta!\langle v_0 \rangle$$

for some v_0. It must be the case, as δ is fresh to P, that v_0 is $v_{\mathcal{I}}$ and, by analysis of the reduction rules, that $P \Longrightarrow\xrightarrow{a?b}\Longrightarrow P'$. Now, we know that $\mathcal{I}, \delta : \mathsf{rw}\langle(\mathcal{I})\rangle \vdash \mathcal{C}^{\mathcal{I}}_{\alpha}$ so from this we can deduce that this must have been inferred from $\mathcal{I} \vdash a : \mathsf{w}\langle \mathrm{B} \rangle$ and $\mathcal{I} \vdash b : \mathrm{B}$ for some type B such that $\mathrm{B} <: \mathcal{I}^w(a)$. This ensures $\mathcal{I}^w(a)\downarrow$ and $\mathcal{I} \vdash b : \mathcal{I}^w(a)$, allowing us to apply Lemma 4.3 to obtain the required

$$\mathcal{I}; \Delta \vdash P \Longrightarrow\xrightarrow{a?b}\Longrightarrow \mathcal{I}; \Delta \vdash P'$$

$\square$**a**

Proposition 5.2 *The relations $\sqsubseteq_{may}$ and $\eqsim_{must}$ over configurations are* w-closed.

Proof: A simple corollary of Lemma 2.3. □

We now give alternative characterisations to these behavioural preorders.

5.1 May Testing

Typed actions are extended to *typed traces* a straightforward manner:

- $\mathcal{I}; \Delta \vdash P \overset{\varepsilon}{\Longrightarrow} \mathcal{I}; \Delta \vdash P$
- $\mathcal{I}; \Delta \vdash P \xrightarrow{\tau} \mathcal{I}; \Delta' \vdash P'$ and $\mathcal{I}; \Delta' \vdash P' \overset{s}{\Longrightarrow} \mathcal{I}; \Delta'' \vdash P''$ implies $\mathcal{I}; \Delta \vdash P \overset{s}{\Longrightarrow} \mathcal{I}; \Delta'' \vdash P''$
- $\mathcal{I}; \Delta \vdash P \xrightarrow{\alpha} \mathcal{I}; \Delta' \vdash P'$ and $\mathcal{I}; \Delta' \vdash P' \overset{s}{\Longrightarrow} \mathcal{I}; \Delta' \vdash P''$ implies $\mathcal{I}; \Delta \vdash P \overset{\alpha \cdot s}{\Longrightarrow} \mathcal{I}; \Delta'' \vdash P''$

The characterisation depends on a Decomposition and Composition result for these sequences. This requires an asymmetric definition of *complementary action.* For a visible action α we let $\overline{\alpha}$ denote $a!v$ if α is $a?v$ and $a?v$ if α has the form $(\tilde{c} : \tilde{\mathrm{C}})a?v$. Thus $\overline{\alpha}$ transforms an action from the untyped semantics in Figure 2 to one from the typed semantics in Figure 5. It is extended to sequences in the natural way.

Theorem 5.3 (Trace Decomposition) *Suppose* $T \mid P \xrightarrow{\tau}{}^{*}$

OUTPUT FROM P TO T: Here we have

$$T \mid P \xrightarrow{\tau} (\mathsf{n}\ \ \mathsf{w}\,\tilde{d} : \tilde{D})\ (T' \mid P') \xrightarrow{\tau}{}^{*} (\mathsf{n}\ \ \mathsf{w}\,\tilde{d} : \tilde{D})\ R'$$

where

$$T \xrightarrow{a?v} T'$$
$$P \xrightarrow{(\tilde{d}:\tilde{D})a!v} P'.$$

Again we can apply Subject Reduction to the first action to obtain $\mathcal{I}^r(a)\downarrow$ and we can apply the third part of Lemma 4.3, this time to obtain the typed action $\mathcal{I}; \Delta \vdash P \xrightarrow{(\tilde{d}:\tilde{D})a!v} \mathcal{I} \sqcap v : \mathcal{I}^r(a); \Delta, \tilde{d} : \tilde{\mathrm{D}} \vdash$

Also, since $\mathcal{I} \sqcap v : \mathcal{I}^r(a)$ is defined, Subject Reduction gives $\mathcal{I} \sqcap v : \mathcal{I}^r(a) \vdash T'$ and so we can apply induction to the sequence $T'|P' \xrightarrow{\tau}{}^{*} R'$ to obtain the remainder of the typed trace.

$\square$

Note that this result is not true if T contains any occurrences of $(\mathsf{n}\ \ \mathsf{w}\,n)$ ( ).

Example 5.4 *Suppose T, P represent the terms $(\mathsf{n}\ \ \mathsf{w}\,c : \mathrm{C})\ a!\langle c\rangle\ c?()\,T'$ and $a?(x)\ x!\langle\rangle\,P'$ respectively, where C is the type $\mathsf{rw}\langle\rangle$, and suppose that $\mathcal{I}$ and Δ are compatible environments such that $\mathcal{I} \vdash T$, $\Delta \vdash P$ and $\mathcal{I}^r(a) = \Delta^r(a) = \mathsf{w}\langle\rangle$; these are easy to construct. Then the derivation $T|P \xrightarrow{\tau}\xrightarrow{\tau} (\mathsf{n}\ \ \mathsf{w}\,c : \mathrm{C})\ T' \mid P'$ can not be decomposed.*

This is a consequence of the assumption built into our configurations *$\mathcal{I}; \Delta \vdash P$, that $\mathcal{I} <: \Delta$.*

Theorem 5.5 (Trace Composition) *Suppose $\mathcal{I}; \Delta$*

Typed

- $\mathcal{C}^{\mathcal{I}}_{\alpha,n}$ will be used to denote the context associated with the action α in Proposition 4.6, however with occurrences of $\delta!\langle v\rangle$ replaced by $n?()\,\delta!\langle v\rangle$.

If s is the empty sequence then $C(s)$ is simply $\omega!\langle\rangle \oplus \omega!\langle\rangle$. Otherwise suppose it has the form $\alpha \cdot s'$. Then $C(s)$ is defined to be

$$\begin{aligned}&(\mathsf{n}\ \ \mathsf{w}\,n)\ \ n!\langle\rangle \mid n?()\,\omega!\langle\rangle \ \ \mid \\ &\qquad\qquad (\mathsf{n}\ \ \mathsf{w}\,\delta : \mathsf{rw}\langle\mathcal{I}'\rangle)\ \ \mathcal{C}^{\mathcal{I}}_{\alpha,n} \mid \delta?(X : (\mathcal{I}'))\ \ (C(s\end{aligned}$$

- $\mathcal{I} \vdash A(s, D)$
- **m̸ust** $A(s, D)$ because of the derivation from which gives rise to the acceptance set D
- but by construction P **must** $A(s, D)$; note this holds even in the case when $Acc(\mathcal{I}; \Delta \vdash P, s)$ is empty.

□

6 Bisimulation

We now describe our characterisation of the co-inductively defined behavioural equivalence, $\approxeq^{\text{cxt}}_{\text{obs}}$, outlined in Section 3.2.

First we recall the definition of weak bisimulation from [13].

Definition 6.1 *Given a labelled transition system $\mathcal{T}$, we say that a binary relation $\mathcal{R}$ on $\mathcal{T}$ is a* bisimulation *if whenever $n \mathrel{\mathcal{R}} m$ then*

- *if $n \xrightarrow{\mu} n'$ then there exists a $m \overset{\hat{\mu}}{\Longrightarrow} m'$ such that $n' \mathrel{\mathcal{R}} m'$*
- *if $m \xrightarrow{\mu} m'$ then there exists a $n \overset{\hat{\mu}}{\Longrightarrow} n'$ such that $n' \mathrel{\mathcal{R}} m'$*

where $\hat{\mu}$ is ε, the empty string, if μ is τ and μ otherwise.

Our intention is to show that $\approxeq^{\text{cxt}}_{\text{obs}}$ can be characterised in terms of a bisimulation over **Conf**.

However as in Section 5 we have a mismatch between the formalisation of this relation, $\approxeq^{\text{cxt}}_{\text{obs}}$, in Section 3.2, which only uses one type environment, of the process being observed, and that of bisimulation equivalence, which uses two type environments. As with testing, we reconcile this difference by extending the definition of $\approxeq^{\text{cxt}}_{\text{obs}}$ so that it takes into account both environments.

First we generalise Definition 3.2 by now saying that an (extended) typed relation is a family $\mathcal{R}$ of relations over *typed processes*, parametrised, as before, by closed type environments, which satisfies: $(\Delta \vdash P)\ R_{\mathcal{I}}\ (\Delta' \vdash\ \ )$ implies $\mathcal{I}; \Delta \vdash P$ and $\mathcal{I}; \Delta \vdash\ \ $ are configurations. To conform to our previous notation we write this as

$$\mathcal{I} \models (\Delta \vdash P)\ \ R\ \ (\Delta' \vdash\ \ ).$$

although effectively these are restricted forms of relations over configurations.

Definition 6.2 *Let* (typed) bisimulation equivalence *be the largest typed relation $\approx$ which is*

- *a weak bisimulation*
- w-closed*, that is satisfying* $\mathcal{I} \models (\Delta \vdash P)\ \mathcal{R}\ (\Delta' \vdash \quad)$ *implies* $\mathcal{I}, \Delta'' \models (\Delta, \Delta'' \vdash P)\ \mathcal{R}\ (\Delta', \Delta'' \vdash \quad)$

Bisimulation equivalence will be written as

$$\mathcal{I} \models (\Delta \vdash P) \approx (\Delta' \vdash \quad).$$

Note that the second requirement is required because we have already seen that $\eqsim^{\mathrm{cxt}}_{\mathrm{obs}}$ is *w-closed.* Intuitively its inclusion allows environments to pass *new* values to processes under investigation.

Two natural properties of (typed) bisimulation equivalence is given in the following proposition:

Proposition 6.3 *Suppose* $\mathcal{I} \models (\Delta \vdash P) \approx (\Delta' \vdash \quad)$*. Then*

- *for any appropriate* Δ'', $\mathcal{I}, \Delta'' \models (\Delta, \Delta'' \vdash P) \approx (\Delta', \Delta'' \vdash \quad)$*.*
- *If* $\mathcal{I} <: \mathcal{I}'$ *then* $\mathcal{I}' \models (\Delta \vdash P) \approx (\Delta' \vdash \quad)$

Proof: The first result is simply a re-iteration of the fact that $\approx$ is *w-closed.* Intuitively the second property is true because $\mathcal{I}$ constrains the behaviour under which P and are compared. If they are equivalent under the constraint $\mathcal{I}$ then they should remain equivalent when they are constrained further, by $\mathcal{I}'$. To prove it formally let the family $\mathcal{R}$ be defined by

$$\mathcal{I}' \models (\Delta \vdash P)\ \mathcal{R}\ (\Delta' \vdash \quad)$$

if $\mathcal{I} \models (\Delta \vdash P) \approx (\Delta' \vdash \quad)$ for some $\mathcal{I} <: \mathcal{I}'$. This family is *w-clo* definition, and it is straightforward to show that it is a bisimulatio follows that $\mathcal{R} \ \subseteq \approx$, pointwise, from which the result follows.

Let us now turn our attention to giving a similar formulation to $\eqsim_{\mathrm{b}}$

aobstatts2s(a)1222012121(bos1(1(1(a)(t)12()barb0(er)2

(CXT-SPEC)

$$\frac{\mathcal{I} \models (\Delta \vdash P)\; \mathcal{R}\; (\Delta' \vdash \quad),\;\; \mathcal{I} <: \mathcal{I}'}{\mathcal{I}' \models (\Delta, \vdash P)\; \mathcal{R}\; (\Delta', \vdash \quad)}$$

(CXT-WEAK)

$$\frac{\mathcal{I} \models (\Delta \vdash P)\; \mathcal{R}\; (\Delta' \vdash \quad)}{\mathcal{I}, \Delta'' \models (\Delta, \Delta'' \vdash P)\; \mathcal{R}\; (\Delta', \Delta'' \vdash \quad)}$$

(CXT-IN)

$$\frac{\begin{array}{l}\mathcal{I} \vdash a : \mathsf{r}\langle \mathrm{A} \rangle \\ \mathcal{I}, \Delta'' \models (\Delta, \Delta'' \vdash T[v/X])\; \mathcal{R}\; (\Delta', \Delta'' \vdash U[v/X]),\;\; \text{whenever } \mathcal{I}, \Delta'' \vdash v : \mathrm{A}\end{array}}{\mathcal{I} \models (\Delta \vdash a?(X : \mathrm{A})\, T)\; \mathcal{R}\; (\Delta' \vdash a?(X : \mathrm{A})\, .U)}$$

(CXT-OUT)

$$\frac{\begin{array}{l}\mathcal{I} \vdash u : \mathsf{w}\langle \mathrm{A} \rangle \\ \mathcal{I} \vdash v : \mathrm{A} \\ \mathcal{I} \models (\Delta \vdash P)\; \mathcal{R}\; (\Delta' \vdash \quad)\end{array}}{\mathcal{I} \models (\Delta \vdash u!\langle v \rangle\, P)\; \mathcal{R}\; (\Delta' \vdash u!\langle v \rangle \quad)}$$

(CXT-MATCH)

$$\frac{\begin{array}{l}\Delta \vdash u : \mathrm{A}, v : \mathrm{A}' \qquad \Delta' \vdash u : \mathrm{B}, v : \mathrm{B}' \\ \mathcal{I} \models (\Delta \vdash P')\; \mathcal{R}\; (\Delta' \vdash \quad ') \\ \mathcal{I} \models (\Delta \sqcap \{u : \mathrm{A}', v : \mathrm{A}\} \vdash P)\; \mathcal{R}\; (\Delta' \sqcap \{u : \mathrm{B}', v : \mathrm{B}\} \vdash \quad)\end{array}}{\mathcal{I} \models (\Delta \vdash \mathsf{if}\ u = v\ \mathsf{th\ n}\ P\ \mathsf{\ ls}\ \ P')\; \mathcal{R}\; (\Delta' \vdash \mathsf{if}\ u = v\ \mathsf{th\ n}\ \quad \mathsf{\ ls}\ \quad ')}$$

(CXT-NEW)

$$\frac{\mathcal{I}, a : \top \models (\Delta, a : \mathrm{A} \vdash P)\; \mathcal{R}\; (\Delta', a : \mathrm{A} \vdash \quad)}{\mathcal{I} \models (\Delta \vdash (\mathsf{n\ w}\, a : \mathrm{A})\; P)\; \mathcal{R}\; (\Delta' \vdash (\mathsf{n\ w}\, a : \mathrm{A}) \quad)}$$

(CXT-PAR)

$$\frac{\begin{array}{l}\mathcal{I} \models (\Delta \vdash P)\; \mathcal{R}\; (\Delta' \vdash \quad) \\ \mathcal{I} \vdash R\end{array}}{\begin{array}{l}\mathcal{I} \models (\Delta \vdash P \mid R)\; \mathcal{R}\; (\Delta' \vdash \quad \mid R) \\ \mathcal{I} \models (\Delta \vdash R \mid P)\; \mathcal{R}\; (\Delta' \vdash R \mid \quad)\end{array}}$$

(CXT-ITER)

$$\frac{\mathcal{I} \models (\Delta \vdash P)\; \mathcal{R}\; (\Delta' \vdash \quad)}{\mathcal{I} \models (\Delta \vdash *P)\; \mathcal{R}\; (\Delta' \vdash * \quad)}$$

FIGURE 6. Contextuality for indexed relations over configurations

however that the first two rules, (CXT-SPEC) and (CXT-WEAK), automatically build in *specialisation* and *weakening* properties, respectively. This may seem artificial but is justified by the following result, which shows that we do indeed have a generalisation of the definition of $\approxeq^{\text{cxt}}_{\text{obs}}$ from Section 3.2:

Proposition 6.4 *$\Delta \models (\Delta \vdash P) \approxeq^{cxt}_{obs} (\Delta \vdash \quad)$ if and only if $\Delta \models P \approxeq^{cxt}_{obs}$*
.

Proof: We first show the *if* direction. Define a typed relation $\mathcal{R}$ by letting

$$\mathcal{I} \models (\Delta \vdash P)\, \mathcal{R}\, (\Delta \vdash \quad)$$

if $\Delta \models P \approxeq^{\text{cxt}}_{\text{obs}} \quad$ and $\Delta <: \mathcal{I}$. $\mathcal{R}$ is symmetric, reduction closed and barb preserving. Using the fact that $\approxeq^{\text{cxt}}_{\text{obs}}$, as a family of relations over processes, is contextual, we can show that it satisfies all of the rules in Figure 6.

Therefore $\mathcal{R}$ is contained pointwise in $\approxeq^{\text{cxt}}_{\text{obs}}$, from which the result follows, since $\Delta \models P \approxeq^{\text{cxt}}_{\text{obs}} \quad$ implies $\Delta \models (\Delta \vdash P)\, \mathcal{R}\, (\Delta \vdash \quad)$.

The converse is similar. Let the family of relations $\mathcal{R}$, over processes, be defined by

$$\Delta \models P\;\; \mathcal{R} \qquad \text{if } \Delta \models (\Delta \vdash P) \approxeq^{\text{cxt}}_{\text{obs}} (\Delta \vdash \quad).$$

Here the result will follow if we can show that $\mathcal{R}$ is contained pointwise in $\approxeq^{\text{cxt}}_{\text{obs}}$, which in turn will follow if we can show that $\mathcal{R}$ satisfies all the defining properties of $\approxeq^{\text{cxt}}_{\text{obs}}$. The proof that it is symmetric, reduction closed and barb preserving is straightforward.

It remains to show contextuality, that $\Gamma' \models T\; \mathcal{R}^o\; U$ and $\Gamma \vdash C[\cdot_{\Gamma'}]$ implies $\Gamma \models C[T]\; \mathcal{R}^o\; C[U]$. This is proved by induction on the derivation of $\Gamma \vdash C[\cdot_\Gamma]$, using the rules in Figure 6. Note that the rule (CXT-SPEC) is essential in the proof of the case in which the context is deduced using (T-NEW).

□

The remainder of this section is devoted to showing that this generalised contextual equivalence coincides with weak bisimulation on **Conf**; that is $\mathcal{I} \models (\Delta \vdash P) \approxeq^{\text{cxt}}_{\text{obs}} (\Delta' \vdash \quad)$ if and only if $\mathcal{I} \models (\Delta \vdash P) \approx (\Delta' \vdash \quad)$.

6.1 oundness

First let us show that typed bisimulation equivalence is preserved, in some appropriate manner, by the principal operators of the language.

Proposition 6.5 *If* $\mathcal{I}, a : \top \models (\Delta, a : \mathrm{A} \vdash P) \approx (\Delta', a : \mathrm{A} \vdash \quad)$ *then* $\mathcal{I} \models (\Delta \vdash (\mathsf{n\ w}\, a : \mathrm{A})\ P) \approx (\Delta' \vdash (\mathsf{n\ w}\, a : \mathrm{A}) \quad)$*.*

Proof: Let the relation $\mathcal{R}$ over typed processes be defined by

$$\mathcal{I} \models (\Delta \vdash R)\ \mathcal{R}\ (\Delta \vdash S)$$

if

- $\mathcal{I} \models (\Delta \vdash R) \approx (\Delta \vdash S)$
- or R S, have the form $(\mathsf{n\ w}\, a : \mathrm{A})\ P$, $(\mathsf{n\ w}\, a : \mathrm{A}) \quad$, respectively, and $\mathcal{I}, a : \top \models (\Delta, a : \mathrm{A} \vdash P) \approx (\Delta', a : \mathrm{A} \vdash \quad)$.

  Then $\mathcal{R}$ is *w-closed* by definition. We show it is a bisimulation, from which the result will follow since we will have established that, pointwise, $\mathcal{R}$ is contained in $\approx$.

  We show how every possible move from $\mathcal{I}; \Delta \vdash R$ can be matched by one from $\mathcal{I}; \Delta \vdash S$. The only non-trivial cases are when R, S have the second form above. From the definition of typed actions in Figure 5 there are two possibilities.

  1. The move is inferred using the rule (TYLTS-OPEN):

$$\mathcal{I}; \Delta \vdash (\mathsf{n\ w}\, a : \mathrm{A})\ P \xrightarrow{(a)\alpha} \mathcal{I}'; \Delta_\alpha \vdash P'$$

where $a \notin \mathsf{n}(\mu)$.
Here the proof is similar. We can find a matching move from $\mathcal{I}, a : \top; \Delta', a : \mathrm{A} \vdash$ and then use (TYLTS-CTXT) to obtain the required matching move from $\mathcal{I}; \Delta \vdash (\mathsf{n\ w}\, a : \mathrm{A})$.

□

Proposition 6.6 *Suppose* $\mathcal{I} \vdash R$*. Then* $\mathcal{I} \models (\Delta \vdash P) \approx (\Delta' \vdash \quad)$ *implies* $\mathcal{I} \models (\Delta \vdash P \mid R) \approx (\Delta' \vdash \quad \mid R)$*.*

Proof: Here, because of the possible internal communications between R and P, , the required definition of the relation over typed processes is somewhat complicated.

Define the relation $\mathcal{R}$ such that

$$\mathcal{I} \models (\Delta \vdash (\mathsf{n\ w}\, \Delta_0)\ P \mid R)\ \mathcal{R}\ (\Delta' \vdash (\mathsf{n\ w}\, \Delta_0')\quad \mid R)$$

if and only if there exists an $\mathcal{I}_0$ compatible with Δ_0 and Δ_0' such that

$$\mathcal{I}, \mathcal{I}_0 \models (\Delta, \Delta_0 \vdash P) \approx (\Delta', \Delta_0' \vdash \quad) \text{ and } \mathcal{I}, \mathcal{I}_0 \vdash R$$

and show that $\mathcal{R}$ forms a bisimulation.

Suppose then that

$$\mathcal{I} \models (\Delta \vdash (\mathsf{n\ w}\, \Delta_0)\ P \mid R)\ \mathcal{R}\ (\Delta' \vdash (\mathsf{n\ w}\, \Delta_0')\quad \mid R)$$

and that

$$\mathcal{I}; \Delta \vdash P \mid R \xrightarrow{\mu} \mathcal{I}'; \Delta'' \vdash P'.$$

This presupposes the existence of an environment $\mathcal{I}_0$ compatible with both Δ_0 and Δ_0' with the properties outlined in the definition of $\mathcal{R}$. If μ is a not a τ-action then we know that the transition derives either from P or from R. In either case, we can use the hypothesis to obtain a corresponding transition from or from R again. So, the interesting case is when μ is a τ action. Consider how this can occur:

(i) P or R performs a τ action independently.

(ii) $P \xrightarrow{(\tilde{c}:\tilde{C})a!v} P_0$ and $R \xrightarrow{a?v} R'$ so that P' is $(\mathsf{n\ w}\, \Delta_0, \tilde{c} : \tilde{\mathrm{C}})\ P_0 \mid R'$ for some $\tilde{\mathrm{C}}$.

(iii) $P \xrightarrow{a?v} P_0$ and $R \xrightarrow{(\tilde{c}:\tilde{C})a!v} R'$ so that P' is $(\mathsf{n\ w}\, \Delta_0, \tilde{c} : \tilde{\mathrm{C}})\ P_0 \mid R'$

Obviously the first case (i) is treated as the case above when μ is not a τ action.

and, again by Subject Reduction, Theorem 2.2, it is easy to see that $\mathcal{I}^{+} \vdash R'$, whence

$$\mathcal{I} \models (\Delta \vdash (\mathsf{n\ \ w}\,\Delta_0, \tilde{c} : \tilde{\mathrm{C}})\ P_0 \mid R')\ \mathcal{R}\ (\Delta' \vdash (\mathsf{n\ \ w}\,\Delta_0', \tilde{c} : \tilde{\mathrm{C}})\ \quad {}_0 \mid R')$$

as required.

□

We now have most of the ingredients to prove:

Theorem 6.7 *(Soundness)*

If $\mathcal{I} \models (\Delta \vdash P) \approx (\Delta' \vdash \quad)$ *then* $\mathcal{I} \models (\Delta \vdash P) \mathrel{\overset{cxt}{\approx}}_{obs} (\Delta' \vdash \quad)$.

Proof: It is easy to see that $\approx$ is a reduction closed, symmetric and barb preserving relation over typed processes. If we can demonstrate that it is also contextual then, because of the the fact that $\approx^{\mathrm{cxt}}_{\mathrm{obs}}$ is the largest such relation we have our result. Therefore we only have to prove that $\approx$ satisfies all the rules in Figure 6.

The rules (CXT-SPEC) and (CXT-WEAK) are covered by Proposition 6.3, while (CXT-NEW) and (CXT-PAR) have just been established in the previous two Propositions. The remaining rules can be handled in a similar manner, by setting up an appropriate *w-closed* relation over typed processes and showing it is a bisimulation.

□

6.2 Completeness

Here we show the converse of Theorem 6.7, *completeness*, namely that contextual equivalence implies bisimularity. To do so we only need a restricted version of contextual equivalence. Let $\approx^{\mathrm{p\text{-}cxt}}_{\mathrm{obs}}$ denote the largest relation over configurations which is reduction closed, barb preserving and contextual with respect to parallel and new name contexts, that is satisfies the rules (CXT-SPEC), (CXT-WEAK), (CXT-PAR) and (CXT-NEW) from Figure 6. It is clear that $\approx^{\mathrm{cxt}}_{\mathrm{obs}}$ implies $\approx^{\mathrm{p\text{-}cxt}}_{\mathrm{obs}}$ so, in fact, it suffices to prove completeness for the latter and we shall use this relation from now on.

Before we prove this theorem it will be useful to present a technical lemma. It is here that we utilize the exported names in the terms which witness the contextuality of labels. Essentially, the lemma states that the environment really can collate the information gained via the lts.

Lemma 6.8 *Suppose* $\mathcal{I}'$ *is compatible with* $\Delta, \tilde{c} : \tilde{\mathrm{C}}$ *and* $\Delta', \tilde{c} : \tilde{\mathrm{C}}$ *and* δ *is fresh to* P, $\quad$. *Then*

$$\mathcal{I}, \delta : \mathsf{rw}\langle(\mathcal{I}')\rangle \models (\Delta, \delta : \mathsf{rw}\langle(\mathcal{I}')\rangle \vdash (\mathsf{n\ \ w}\,\tilde{c} : \tilde{\mathrm{C}})\ P \mid \delta!v_{\mathcal{I}'})$$
$$\mathrel{\overset{p\text{-}cxt}{\underset{obs}{\approxeq}}} (\Delta', \delta : \mathsf{rw}\langle(\mathcal{I}')\rangle \vdash (\mathsf{n\ \ w}\,\tilde{c} : \tilde{\mathrm{C}}')\quad \mid \delta!v_{\mathcal{I}'})$$

implies

$$\mathcal{I}' \models (\Delta, \tilde{c} : \tilde{\mathrm{C}} \vdash P) \mathrel{\overset{p\text{-}cxt}{\underset{obs}{\approxeq}}} (\Delta', \tilde{c} : \tilde{\mathrm{C}}' \vdash \quad).$$

Proof: We prove this by co-induction. Let the relation $\mathcal{R}_{\mathcal{I}'}$ be defined for $\mathcal{I}'$ compatible with Δ, Δ_0 and Δ', Δ'_0, so that

$$\mathcal{I}' \models (\Delta, \Delta_0 \vdash (\mathsf{n\ \ w}\,\Delta_1)\ P)\ \mathcal{R}\ (\Delta', \Delta_0 \vdash (\mathsf{n\ \ w}\,\Delta'_1)\quad)$$

if and only if there is some $\delta : \mathsf{rw}\langle(\mathcal{I}')\rangle$ such that

$$\mathcal{I}, \delta : \mathsf{rw}\langle(\mathcal{I}')\rangle \models (\Delta, \delta : \mathsf{rw}\langle(\mathcal{I}')\rangle \vdash (\mathsf{n\ \ w}\,\Delta_0, \Delta_1)\ P \mid \delta!\langle v_{\mathcal{I}'}\rangle)$$
$$\mathrel{\overset{\text{p-cxt}}{\underset{\text{obs}}{\approxeq}}} (\Delta', \delta : \mathsf{rw}\langle(\mathcal{I}')\rangle \vdash (\mathsf{n\ \ w}\,\Delta'_0, \Delta'_1)\quad \mid \delta!\langle v_{\mathcal{I}'}\rangle).$$

We simply need to show that $\mathcal{R}$ is reduction closed, barb preserving, and closed with respect to rules (CXT-SPEC), (CXT-WEAK), (CXT-PAR) and (CXT-NEW). Reduction closure is immediate by the definition of $\mathcal{R}$, as is closure with respect to (CXT-SPEC) and (CXT-WEAK). For the other requirements we proceed by supposing that

$$\mathcal{I}' \models (\Delta, \Delta_0 \vdash P)\ \mathcal{R}\ (\Delta', \Delta'_0 \vdash \quad)$$

such that $\delta : \mathsf{rw}\langle(\mathcal{I}')\rangle$ with

$$\mathcal{I}, \delta \models (\Delta, \delta \vdash (\mathsf{n\ \ w}\,\Delta_0)\ P \mid \delta!\langle v_{\mathcal{I}'}\rangle) \mathrel{\overset{\text{p-cxt}}{\underset{\text{obs}}{\approxeq}}} (\Delta', \delta \vdash (\mathsf{n\ \ w}\,\Delta'_0)\quad \mid \delta!\langle v_{\mathcal{I}'}\rangle).$$

In the above equation, for the sake of presentation, we have omitted, and shall continue to do so for the remainder of this proof, to give the type information associated with the barb δ.

We first show closure with respect to (CXT-PAR). Suppose $\mathcal{I}' \vdash R$. We need to show that $\mathcal{I}' \models (\Delta, \Delta_0 \vdash P \mid R)\ \mathcal{R}\ (\Delta', \Delta'_0 \vdash \quad \mid R)$. To do this we choose some fresh δ' and construct $R' = \delta?(X : \mathcal{I}')\ (R[X/\mathsf{n}(\mathcal{I}')] \mid \delta'!\langle\rangle)$ (recall that $\mathsf{n}(\Gamma)$ refers to the names in the domain of Γ). It should be evident that $\delta, \delta' \vdash R'$ and, by closure of $\overset{\text{p-cxt}}{\underset{\text{obs}}{\approxeq}}$ with respect to (CXT-SPEC), (CXT-WEAK), (CXT-NEW) and (CXT-PAR) we ha)and(C̃xtcxtshol

[illegible]

and similarly for . Hence,

$$\mathcal{I}, \delta' \models (\Delta, \delta' \vdash (\mathsf{n}\ \ \mathsf{w}\, \Delta_0)\ P|R|\delta'!\langle v_{\mathcal{I}'}\rangle) \approxeq^{\text{p-cxt}}_{\text{obs}} (\Delta', \delta' \vdash (\mathsf{n}\ \ \mathsf{w}\, \Delta'_0)\quad |R|\delta'!\langle v_{\mathcal{I}'}\rangle)$$

This serves to witness

$$\mathcal{I}' \models (\Delta, \Delta_0 \vdash P \mid R)\ \mathcal{R}\ (\Delta', \Delta'_0 \vdash \quad \mid R)$$

as required.

The closure of $\mathcal{R}$ with respect to (CXT-NEW) follows easily from the closure of $\approxeq^{\text{p-cxt}}_{\text{obs}}$ with respectto

We choose a fresh $\delta : \mathrm{A}_\delta$ where A_δ denotes $\mathsf{rw}\langle(\mathcal{I}')\rangle$, and use Proposition 4.6 to find a term such that $\mathcal{I}, \delta : \mathrm{A}_\delta \vdash \mathcal{C}^{\mathcal{I}}_\mu$ with the appropriate properties. In fact, the first property tells us that

$$\mathcal{I}, \delta : \mathrm{A}_\delta; \Delta, \delta : \mathrm{A}_\delta \vdash P \,|\, \mathcal{C}^{\mathcal{I}}_\mu \Longrightarrow \mathcal{I}, \delta : \mathrm{A}_\delta; \Delta, \delta : \mathrm{A}_\delta \vdash (\mathsf{n\ \ w}\, \tilde{c} : \tilde{\mathrm{C}})\ (P' \,|\, \delta!\langle v_{\mathcal{I}'}\rangle)$$

Using $\mathcal{C}^{\mathcal{I}}_\mu$ we can build a test term by choosing further fresh names $\delta' : \mathrm{A}_\delta,\ a : \mathsf{rw}\langle\top\rangle$ and letting

$$\mathcal{C}_{\delta'} = a!\langle\rangle \mid \delta?(x)\ \ a?(y)\,.\delta'!\langle x\rangle$$

we note immediately that $\mathcal{C}_{\delta'} \Downarrow^{\text{barb}} a$.

From contextual closure (omitting some type information) we know that

$$\mathcal{I}, \delta' \models (\Delta, \delta' \vdash (\mathsf{n\ \ w}\, \delta)\ (P \,|\, \mathcal{C}^{\mathcal{I}}_\mu \,|\, \mathcal{C}_{\delta'})) \approx^{\text{p-cxt}}_{\text{obs}} (\Delta', \delta' \vdash (\mathsf{n\ \ w}\, \delta)\ (\quad \,|\, \mathcal{C}^{\mathcal{I}}_\mu \,|\, \mathcal{C}_{\delta'}))$$

We also know that the left hand side of this equation may reduce (up to a minor structural equivalence) to

$$\mathcal{I}, \delta'; \Delta, \delta' \vdash (\mathsf{n\ \ w}\, \tilde{c} : \tilde{\mathrm{C}})\ \ P' \mid \delta'!\langle v_{\mathcal{I}'}\rangle\,.$$

We use C_P to refer to this configuration and observe that $C_P \not\Downarrow^{\text{barb}} a$ but $C_P \Downarrow^{\text{barb}} \delta'$.

Reduction closure now tells us that there must exist some matching reductions

$$\mathcal{I}, \delta'; \Delta', \delta' \vdash (\mathsf{n\ \ w}\, \delta)\ (\quad \mid \mathcal{C}^{\mathcal{I}}_\mu \mid \mathcal{C}_{\delta'}) \Longrightarrow C_Q$$

for some C_Q such that C_P

$\mathcal{C}$

$\Downarrow^{+}$

tereTdbutTTTdeTTdCT

Soundness, Completeness and Proposition 6.4 allows us to now conclude with the main result of the paper:

Corollary 6.10 *If* $\Gamma \models P \approxeq^{cxt}_{obs}$ *if and only if* $\Gamma \models (\Gamma \vdash P) \approx^{o} (\Gamma \vdash \quad)$.

6.3 Example

The characterisation of the previous sections provide a convenient coinductive method for establishing contextual observational equivalence between terms. We provide a short example which demonstrates the utility of the bisimulation proof method. The processes that we consider provide two different implementations of a producer/consumer unit server.

Clients send requests for service along a global channel *req*, which must be accompanied by a reply channel which has type at least $\mathrm{R} = \mathsf{w}\langle(\mathsf{w}\langle\top\rangle, \mathsf{r}\langle\top\rangle)\rangle$. The server creates dedicated produce and consume channels, exclusively for the client, at type $\mathrm{A} = \mathsf{rw}\langle\top\rangle$, and returns these along the reply channel. Note that because of the type of the return channel the client only receives the write capability on the produce channel and the read capability on the consume channel. The server then manages the simple protocol that for every call on the produce channel, a corresponding request can be made of the consume channel:

$$CU_1 = *req?(x:\mathrm{R})\,(\mathsf{n\ \ w}\,p, c : \mathrm{A})\ \ x!\langle p, c\rangle \quad *\,p?()\,.c!\langle\rangle$$

Here the server and uses the process $*p?()\,.c!\langle\rangle$ to manage the produce and consume requests.

Another implementation is given by:

$$CU_2 = *req?(x:\mathrm{R})\,(\mathsf{n\ \ w}\,p, c : \mathrm{A})\ \ x!(p, c).\ \ (*p?()\,.c!\langle\rangle \mid *c?()\,.p!\langle\rangle).$$

The behaviour of this server, when managing the produce/consume requests is a little different. Here the server itself, in addition to the client, may consume a request; if it does so it then reproduces a message in recompense.

There is a bug in the second implementation because, having set up a protocol for a client, when a message is sent by the client it may be consumed by the server itself, consequently unleashing an infinite sequence of produce/consume messages internal to the server. This can be formally demonstrated using the *must* testing equivalence.

Let $\Delta^{<}_{req}$ be any typing environment such that $\Delta^{<}_{req}(req) = \mathsf{rw}\langle\mathsf{rw}\langle(\mathrm{B}, \mathrm{C})\rangle\rangle$, where $\mathrm{B} <: \mathsf{w}\langle\top\rangle$. Then

$$\Delta^{<}_{req} \models CU_1 \not\eqsim_{must} CU_2.$$

To prove this result we exhibit a test T such that $\Delta^{<}_{req} \vdash T$, CU_1 **must** T

but $CU_1 \not\!\!\textbf{must}\ T$. The required T is given by

$$(\mathsf{n}\ \ \mathsf{w}\, r : \mathrm{A}_r)\ \ req!\langle r\rangle\ \ r?((x,y) : (\mathsf{w}\langle\top\rangle, \mathsf{r}\langle\top\rangle))\ \ x!\langle\rangle\, \omega!\langle\rangle$$

where A_r is the type $\mathsf{rw}\langle(\mathrm{B},\mathrm{C})\rangle$. It is straightforward to show that this can be typed by Δ_{req}, that it is guaranteed by CU_1 but when applied to CU_2, may lead to a non-terminating computation.

It is well-known that contextual observational equivalence is insensitive to such and for any $\mathcal{I} :> p : \mathrm{B}, c : \mathrm{C}$, we have

$$\mathcal{I} \models (\Delta_A \vdash P_1$$

are two possibilities for these: an interaction between P_2 and $c!^m$ and an interaction between P_2 and $p!^k$. Note that in each case the resulting state is

$$P_2 \mid c!^{m-1} \mid p!^{k+1}$$

for the former and

$$P_2 \mid c!^{m+1} \mid p!^{k-1}$$

for the latter. In either case the total $m + k$ is invariant. This means that the extra internal transitions exhibited by $P_2 \mid c!^m \mid p!^k$ may be matched in $\mathcal{R}$ by an empty transition from $P_1 \mid c!^n$.

It is worth mentioning here that it is not possible to observe output transitions of the form $p!\langle\rangle$ from $\mathcal{I}; \Delta_A \vdash P_2 \mid c!^m \mid p!^k$ as we have supposed that $\mathcal{I}(p) :> \mathrm{B} :> \mathsf{w}\langle\top\rangle$ and thus cannot the read capability required to make this observation. Similarly, it is not possible to observe input transitions of the form $c?\langle\rangle$ from $\mathcal{I}; \Delta_A \vdash P_2$.

$\square$

This short example demonstrates the use of a co-inductive proof for establishing contextual observational equivalence. The use of the bisimulation method allows us to establish equivalence without quantifying over all possible clients for these servers. In effect, the environment plays the role of an arbitrary client.

7 Conclusion

In this paper we have studied typed behavioural equivalences for the π-calculus. In particular we have shown that natural typed versions testing and barbed congruences can be captured by applying standard techniques to a new lts of *typed actions*, **Conf**. Thus, at least in principle, it should be possible to use, or adapt, existing proof methodologies and verification systems, [4, 5] to prove type dependent equivalences between processes. Admittedly the *states*, $\mathcal{I}; \Delta \vdash P$, in the lts are a priori complicated, consisting of a process term P, a type environment for its computing context $\mathcal{I}$ and a separate type environment for the process itself Δ. But the observant reader will have noticed that in the rules for generated **Conf**, in Figure 5, the last type environment Δ plays no role. Technically its presence has been convenient for deriving our results, which depend on the fact that processes are well-typed with respect to some environment coherent with $\mathcal{I}$, but in an implementation of **Conf** they could be safely omitted.

Typed process equivalences, as opposed to untyped ones, have nu-

equations holding in this setting vary considerably from ours. For instance the well-known Replication Theorem of π-calculus used to illustrate their technique fails to hold in the presence of equality testing.

Our system allows for a gradual increase in knowledge about types of names and provides a fresh approach to understanding the effects of subtyping on process equivalence.

References

[1] oberto M. Amadio, Ilaria Castellani, and Davide Sangiorgi. On bisimulations for the asynchronous π-calculus. *Theoretical Computer Science*, 195(2):291–324, 30 March 1998.

[2] M. Boreale and D. Sangiorgi. Bisimulation in name-passing calculi without matching. In *Proc. 13th LICS Conf.* IEEE Computer Society Press, 1998.

[3] G. Boudol. Typing the use of resources in a concurrent calculus. In *Proceedings of the ASIAN'97*, number 1345 in Lecture Notes in Computer Science, pages 239–253, 1997.

[4] . Cleaveland, J. Parrow, and B. Steffen. The concurrency workbench: A semantics based verification tool for finite state systems. *ACM Transactions on Programming S stems*, 15:36–72, 1989.

[5] ance Cleaveland. The concurrency factory: A development environment for concurrent systems. In . Alur and T. Henzinger, editors, *Proceedings of CAV'96*, volume 1102 of *Lecture Notes in Computer Science*, pages 398–401. Springer-Verlag, 1988.

[6] C. Fournet and G.Gonthier. A hierarchy of equivalences for asynchronous calculi (extended abstract). In *Proceedings of ICALP'98*, volume 1443 of *Lecture Notes in Computer Science*, pages 844–855. Springer-Verlag, 1988.

[7] C. Fournet, G. Gonthier, J.J. Levy, L. Marganget, and D. emy. A calculus of mobile agents. In U. Montanari and V. Sassone, editors, *Proceedings of CONCUR'96*, volume 1119 of *Lecture Notes in Computer Science*, pages 406–421, Pisa, August 1996. Springer Verlag.

[8] M. Hennessy. *An Algebraic Theor of Processes.* MIT Press, 1988.

[9] Kohei Honda and Mario Tokoro. On asynchronous communication semantics. In P. Wegner M. Tokoro, O. Nierstrasz, editor, *Proceedings of the ECOOP '91 Workshop on Object-Based Concurrent Computing*, volume 612 of *LNCS 612.* Springer-Verlag, 1992.

[10] A. Jeffrey. A distributed object calculus. In *Proc. ACM Foundations of Object Oriented Languages.* IEEE Computer Society Press, 2000.

[11] A. Jeffrey and J. athke. A theory of bisimulation for a fragment of concurrent ml with local names. In *Proc. LICS2000, 15^{th} Annual S mposium on Logic in Computer Science, Santa Barbara*, pages 311–321. IEEE Computer Society Press, 2000.

[12] Naoki Kobayashi. A partially deadlock-free typed process calculus. In *Proceedings, Twelth Annual IEEE S mposium on Logic in Computer Science*, pages 128–139, Warsaw, Poland, 29 June–2 July 1997. IEEE Computer Society Press.

[13] . Milner. *Communication and Concurrenc .* Prentice-Hall, 1989.

[14] . Milner. *Comunicating and mobile s stems: the π-calculus.* Cambridge Univer-