

Contextual equivalence for higher-order π -calculus revisited

Alan Jeffrey
CTI, DePaul University
ajeffrey@cs.depaul.edu

Julian Rathke*
COGS, University of Sussex
julianr@cogs.susx.ac.uk

October 2002

Abstract

The higher-order π -calculus is an extension of the π -calculus to allow communication of abstractions of processes rather than names alone. It has been studied intensively by Sangiorgi in his thesis where a characterisation of a contextual equivalence for higher-order π -calculus is provided using labelled transition systems and *not a* bisimulations. Unfortunately the proof technique used there requires a restriction of the language to only allow finite types.

We revisit this calculus and offer an alternative presentation of the labelled transition system and a novel proof technique which allows us to provide a fully abstract characterisation of contextual equivalence using labelled transitions and bisimulations for higher-order π -calculus with recursive types also.

1 Introduction

It is evident that there is growing interest in the study of mobile code in process languages [2, 1, 7, 12]. It is also clear that there is some relationship between the use of higher-order features and mobility. Indeed, code mobility can be expressed as communication of process abstractions. For this reason then it is important for us to develop a clear understanding of the use of higher-order features in process languages.

Work towards this began several years ago with various proposals for higher-order versions of known calculi [11, 3], including the higher-order π -calculus or HO π [8]. This calculus was studied intensively by Sangiorgi and one of his achievements was to provide a translation of the higher-order language which supports code mobility, to a first-order π

-ca94(p3-

In this paper we present an alternative description of labelled transition systems and normal bisimulations for $\text{HO}\pi$

$\triangleright, ::=$ \cdot $\triangleright \text{ch}[]$ $\triangleright \rightarrow \diamond$ Z $\text{rec} \triangleright Z.$	Value Types Unit type Channel type Abstraction type Type variable Recursive type
$, ::=$ $v \cdot w$ $v(x \triangleright)$ $v \langle w \rangle$ $\text{if } v = w \text{ then } \quad \text{else}$ $v(d \triangleright). ()$ \parallel $*$ $\mathbf{0}$	Terms Application Input Output Matching Name creation Concurrency Repetition Termination
$v, w ::=$ \cdot a x $(x \triangleright)$	Values Unit value Channel name Variable Abstractions

Figure 1: The Syntax

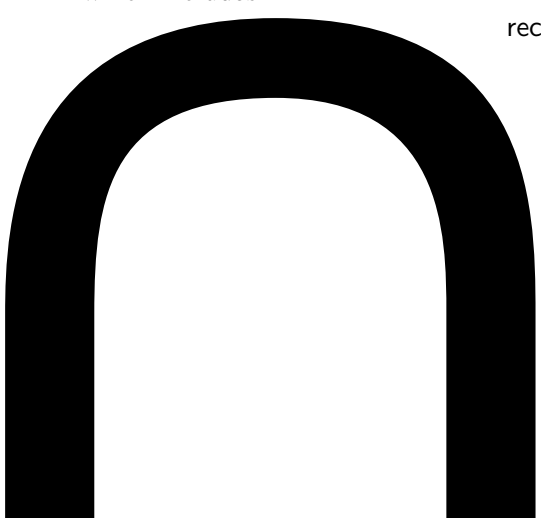
following axioms hold

(comm)	$a \langle v \rangle \parallel a(x)$	\rightarrow	$\parallel (x) \cdot v$
(β -redn)	$(x) \cdot v$	\rightarrow	$[v/x]$
(cond—tt)	$\text{if } a = a \text{ then } \quad \text{else}$	\rightarrow	
(cond—ff)	$\text{if } a = b \text{ then } \quad \text{else}$	\rightarrow	$(a \neq b)$

In a standard notation we write \Longrightarrow to denote the reflexive, transitive closure of \rightarrow .

We introduce a simple type system for the language which comprises types for channels and abstractions. We also allow recursive types of the form $\text{rec} \triangleright Z.$ where $\text{rec} \cdot$ forms a binder and Z is drawn from a countably infinite supply of type variables. We must insist that for any $\text{rec} \triangleright Z.$ that Z does not appear unguarded in \triangleright , that is to say that any free occurrence of Z lies within a subexpression of \triangleright of the form $\text{ch}[]$ or $\rightarrow \diamond$. To allow us to infer recursive types for terms we make use of type isomorphism. We define this by letting \sim_{so} be the least congruence on types which includes

$\text{rec} \triangleright Z.$



$$\begin{array}{c}
\frac{}{\Gamma \vdash \cdot : \cdot} \quad \frac{\Gamma(v) \stackrel{\blacktriangle}{=} \bullet}{\Gamma \vdash v : \bullet} \quad \frac{\Gamma, x : \bullet \vdash \bullet}{\Gamma \vdash (x : \bullet) \rightarrow \diamond} \quad \frac{\Gamma \vdash v : \bullet \quad \blacktriangle \sim_{so} \blacktriangle}{\Gamma \vdash v : \bullet} \\
\\
\frac{\Gamma \vdash v : \text{ch}[\bullet], w : \text{ch}[\bullet]}{\Gamma \vdash \text{if } v = w \text{ then } \bullet \text{ else } \bullet} \quad \frac{\Gamma, d : \bullet \vdash \bullet}{\Gamma \vdash v(d : \bullet) \cdot (\bullet)} \quad \frac{\Gamma \vdash \bullet}{\Gamma \vdash \parallel \bullet, * \bullet, \mathbf{0}} \\
\\
\frac{\Gamma \vdash v : \bullet \rightarrow \diamond \quad \Gamma \vdash w : \bullet}{\Gamma \vdash v \cdot w} \quad \frac{\Gamma, x : \bullet \vdash \bullet \quad \Gamma \vdash v : \text{ch}[\bullet]}{\Gamma \vdash v(x : \bullet)} \quad \frac{\Gamma \vdash \bullet \quad \Gamma \vdash w : \bullet \quad \Gamma \vdash v : \text{ch}[\bullet]}{\Gamma \vdash v \langle w \rangle}
\end{array}$$

Figure 2: The Typing Rules

well-typed process, \bullet , closed if it can be typed as $\Delta \vdash \bullet$ for some closed Δ . It is easily shown that subject reduction holds for closed terms for the reduction relation and type inference system given.

2.1 Contextual equivalence

We will now define an appropriate notion of behavioural equivalence based on contexts and barbs.

Contexts are defined by extending the syntax of processes by allowing typed holes $[\Gamma]$ in terms. The type inference system is extended to contexts by using the rule

$$\frac{}{\Gamma, \Gamma' \vdash [\Gamma]}$$

We write $C[\]$ to denote contexts with at most one hole and $C[\bullet]$ for the term which results from substituting \bullet into the hole.

For any given channel name a such that $\Delta \vdash a : \text{ch}[\bullet]$ we write $\Delta \models \Downarrow a$ if there exists some \bullet, \bullet' such that

Reduction closure: A type-indexed relation \approx is reduction-closed whenever $\Delta \Vdash \tau$ and $\tau \rightarrow \tau'$ implies there exists some τ'' such that $\tau \Longrightarrow \tau''$ and $\Delta \Vdash \tau'' \approx \tau'$.

Contextuality: A type-indexed relation \approx is contextual whenever $\Gamma' \Vdash \tau$ and $\Gamma \vdash C[\cdot]_{\Gamma'}$ implies $\Gamma \Vdash C[\tau] \approx C[\tau']$.

Barb preservation: A type-indexed relation \approx is barb-preserving if $\Delta \Vdash \tau$ and $\Delta \Vdash \Downarrow a$ implies $\Delta \Vdash \Downarrow a$.

Definition 2.1 (Contextual equivalence) Let \cong be the open extension of the largest type-indexed relation which is symmetric, reduction-closed, contextual and barb-preserving. \square

For technical convenience it will be useful to work with a lighter definition of contextuality. We say that a relation \approx is \parallel -contextual if it is preserved by all contexts of the form $[\cdot]_{\Gamma} \parallel$ and we let \cong_p denote the open extension of the largest typed relation over processes which is symmetric, \parallel -contextual, reduction-closed and barb-preserving. The following lemma demonstrates that this lighter definition is sufficient.

Lemma 2.2 (Context lemma) $\Gamma \Vdash \tau \cong \tau'$ and on $y \nabla \tau \Vdash \tau' \cong_p$

Proof: The ‘only if’ direction is immediate. For the converse it is sufficient to show that \cong_p is preserved by each process operator of $\text{HO}\pi$. The majority of these are straightforward so we only show the case for input prefixing. Suppose then (without loss of generality) that $\Delta, a : \text{ch}[\tau], x : \tau \Vdash \tau' \cong_p \tau''$. We need to show that $\Delta \Vdash \tau \cong \tau''$.

3 Labelled transitions

We will use a labelled transition system to characterize \cong over higher-order π -calculus terms. The style of the labelled transition system differs a little from previous transition systems offered for HO π . Most notably, the nodes of the transition system are described using an augmented syntax rather than process terms alone. Specifically, for each a drawn from a countable set of names disjoint from \mathcal{N} and ν , we introduce two new operators:

$$\tau \quad \text{and} \quad \langle \Leftarrow \nu \rangle$$

with the intuitive reading that τ is an indirect reference to an abstraction and $\langle \Leftarrow \nu \rangle$ stores the abstraction to which a refers so that access to ν is provided through interaction with a . The augmented syntax for nodes is given the grammar of configurations C obtained by extending Figure 1 with:

$$\begin{aligned} \nu &::= \dots(\text{as Figure 1})\dots \mid \tau \\ C &::= \dots \mid \langle \Leftarrow \nu \rangle \mid \nu d^? .(C) \mid C \parallel C \end{aligned}$$

We impose a syntactic restriction on the augmented syntax so that in any configuration C for any given a then $\langle \Leftarrow \nu \rangle$ appears at most once in C

C

$$\begin{array}{c}
\frac{\Delta \vdash v \text{ a base type}}{(\Delta; \Theta \vdash a(x) \cdot v) \xrightarrow{a(v)?} (\Delta; \Theta \vdash (x) \cdot v)} \\
\\
\frac{\Theta(\langle \rangle) \stackrel{\Delta \vdash w \text{ a base type}}{\cong}}{(\Delta; \Theta \vdash \langle \leftarrow v \rangle) \xrightarrow{\langle w \rangle?} (\Delta; \Theta \vdash v \cdot w \parallel \langle \leftarrow v \rangle)} \\
\\
\frac{\Delta \vdash v \text{ a base type}}{(\Delta; \Theta \vdash a(v)) \xrightarrow{a(v)!} (\Delta; \Theta \vdash \bullet)} \\
\\
\frac{\Theta(\tau \cdot v) \stackrel{\Delta \vdash \tau \text{ a base type}}{\cong}}{(\Delta; \Theta \vdash \tau \cdot v) \xrightarrow{\langle v \rangle!} (\Delta; \Theta \vdash \mathbf{0})}
\end{array}$$

Figure 5: Basic first-order labelled transition rules

the reference graph of $\langle \leftarrow v \rangle \parallel C$ has the node v removed and any edges such that

$$v' \mapsto v \mapsto$$

for $v', v \neq$, are replaced with an edge

$$v' \mapsto$$

all other edges involving v are removed. So if node v is involved in a cycle

So either, $\tau_1 = \tau_2$ in which case $C_1 \equiv C_2$ or $\tau_1 \neq \tau_2$ and

$$C'_1 \equiv C'_3 \parallel \langle \tau_2 \leftarrow v_2 \rangle \quad \text{and} \quad C'_2 \equiv C'_3 \parallel \langle \tau_1 \leftarrow v_1 \rangle$$

We notice that

$$\begin{aligned}
C_1 &\equiv C'_1[v_1/\tau_1] \\
&\equiv (C'_3 \parallel \langle \tau_2 \leftarrow v_2 \rangle)[v_1/\tau_1] \\
&\equiv C'_3[v_1/\tau_1] \parallel \langle \tau_2 \leftarrow v_2[v_1/\tau_1] \rangle \\
(\text{acyclicity implies } \tau_2 \notin v_2[v_1/\tau_1]) &\rightarrow C'_3[v_1/\tau_1][v_2[v_1/\tau_1]/\tau_2] \\
&\equiv C'_3[v_1[v_2[v_1/\tau_1]/\tau_2]/\tau_1, v_2[v_1/\tau_1]/\tau_2] \\
(\text{acyclicity}) &\equiv C'_3[v_1[v_2/\tau_2]/\tau_1, v_2[v_1/\tau_1]/\tau_2] \\
(\text{def}) &\equiv C_3
\end{aligned}$$

By a symmetric argument we see that $C_2 \rightarrow C'_3[v_2[v_1/\tau_1]/\tau_2, v_1[v_2/\tau_2]/\tau_1]$ and, by definition, this is just C_3 so we have $C_2 \rightarrow C_3$. Thus \rightarrow is strongly confluent for acyclic terms and hence $\langle\langle \cdot \rangle\rangle$ is well-defined. \square

Lemma 4.2 (Composition/Decomposition) For $\Delta; \Theta \vdash C, D$

$\nexists \langle\langle C \parallel D \rangle\rangle \equiv E$ and

$$(\Delta; \Theta \vdash C) \xrightarrow{\alpha} (\Delta, \Delta'; \Theta, \Theta' \vdash C') \quad \text{and} \quad (\Delta; \Theta \vdash D) \xrightarrow{\tilde{\alpha}} (\Delta, \Delta'; \Theta, \Theta' \vdash D')$$

then there exists a E' such that $E \equiv E'$

It is easy to see that $\langle\langle C \parallel D \rangle\rangle \rightarrow \langle\langle \nu\Delta', \Delta'' . ((x^{\blacktriangleleft}) \cdot \nu \parallel C'' \parallel \parallel D'') \rangle\rangle$ let us call the target of this reduction E' . We simply need to check

$$\begin{aligned} E' &\equiv \langle\langle \nu\Delta', \Delta'' . ((x^{\blacktriangleleft}) \cdot \nu \parallel C'' \parallel \parallel D'') \rangle\rangle \\ (\tau \notin \nu) &\equiv \langle\langle \nu\Delta' . ((x^{\blacktriangleleft}) \cdot \tau \parallel C'') \parallel \nu\Delta'' . (\langle \leftarrow \nu \rangle \parallel \parallel D'') \rangle\rangle \\ &\equiv \langle\langle C' \parallel D' \rangle\rangle \end{aligned}$$

Case: $\Delta; \Theta \vdash C \xrightarrow{\nu \cdot \langle \tau \rangle?} \Delta; \Theta, \blacktriangleleft \vdash C'$ and $\Delta; \Theta \vdash D \xrightarrow{\nu \cdot \langle \tau \rangle!} \Delta; \Theta, \blacktriangleleft \vdash D'$. Again, by inspection we see that

- $C \equiv \nu\Delta' . (\langle \leftarrow \nu \rangle \parallel C'')$
- $C' \equiv \nu\Delta' . (\nu \cdot \tau \parallel \langle \leftarrow \nu \rangle \parallel C'')$
- $D \equiv \nu\Delta'' . (\tau \cdot w \parallel D'')$
- $D' \equiv \nu\Delta'' . (\langle \leftarrow w \rangle \parallel D'')$

Note that the previous proposition tells us that $(\mathcal{G} \parallel D)$ must be acyclic — in particular, $\tau \notin \nu$. Here we see that

$$\begin{aligned} \langle\langle C \parallel D \rangle\rangle &\equiv \langle\langle \nu\Delta', \Delta'' . (\langle \leftarrow \nu \rangle \parallel C'' \parallel \tau \cdot w \parallel D'') \rangle\rangle \\ (\tau \notin \nu) &\equiv \langle\langle \nu\Delta', \Delta'' . (\langle \leftarrow \nu \rangle \parallel C'' \parallel \nu \cdot w \parallel D'') \rangle\rangle \\ (\tau \notin \nu, w, C'', D'') &\equiv \langle\langle \nu\Delta', \Delta'' . (\langle \leftarrow \nu \rangle \parallel C'' \parallel \nu \cdot \tau \parallel \langle \leftarrow w \rangle \parallel D'') \rangle\rangle \\ &\equiv \langle\langle C' \parallel D' \rangle\rangle \end{aligned}$$

Case:

Corollary 4.5 (Soundness) For a term t , $\vartheta \in \text{HO}\pi$

$$\Gamma \models t \approx^o p \text{ es } \Gamma \models \vartheta \cong$$

Proof: Follows from the previous theorem and Lemma 2.2. □

5 Completeness of bisimilarity for contextual equivalence

The interactions described by the labelled transition system are not obviously derived by genuine contextual observations in $\text{HO}\pi$ because of the use of the extra syntax for indirect references. In order to show completeness of our bisimilarity for contextual equivalence we must demonstrate that the indirect references are in fact definable as terms of the language proper. Following Sangiorgi [10], we implement the implicit protocol outlined by the indirect references by using the following translation of the augmented terms into $\text{HO}\pi$:

$$\begin{aligned} \llbracket [1 : \text{ch}[c_1], \dots, n : \text{ch}[c_n]] \rrbracket &= [1 : \text{ch}[c_1], \dots, n : \text{ch}[c_n]] \\ \llbracket [\Gamma ; \Theta \vdash C] \rrbracket &= \Gamma, \llbracket [\Theta] \rrbracket \vdash \llbracket [C] \rrbracket_{\Theta} \\ \llbracket [\tau] \rrbracket_{\Theta} &= (x : \text{ch}[c]) \langle x \rangle \mathbf{0} \quad \text{if } \Theta(c) \stackrel{\text{def}}{=} \\ \llbracket [\langle \leftarrow v \rangle] \rrbracket_{\Theta} &= * \llbracket [v] \rrbracket_{\Theta} \end{aligned}$$

The translation acts homomorphically on all other terms. We abuse notation here by using identifiers as channel names in the translation. It is evident that this translation is well-defined in the sense that the translation of well-typed augmented terms are indeed well-typed terms of $\text{HO}\pi$.

We would now like to prove a correspondence between reductions from the terms of the augmented syntax and reductions between their translations. However, we note that in translating a term containing both $\langle \leftarrow v \rangle$ and τ we provide matching input and output prefixes, which, in $\text{HO}\pi$ may create a communication which was not possible in the source term. This turns out not to be of particular concern to us though as we see that if we starting with terms of $\text{HO}\pi$, then terms reachable by transitions are *ba nced* in the following sense: we call a term C of the augmented language *ba nced* if for each τ then C contains at most one of τ

By inspecting the translation $\llbracket \cdot \rrbracket$ and using the fact that C is balanced we see that

$$\llbracket C \rrbracket \xrightarrow{h} \longrightarrow \quad \text{implies} \quad \llbracket C \rrbracket \longrightarrow \xrightarrow{h}$$

thus we may assume that the first reduction in (\dagger) above is not of the form \xrightarrow{h} . This means that

$\llbracket C \rrbracket \longrightarrow \llbracket C' \rrbracket \implies$ for some C' such that $C \longrightarrow C'$. It is clear that C' is also balanced so we may apply the induction

to obtain a D such that $C' \implies D'$ and $\llbracket C' \rrbracket \implies \llbracket D' \rrbracket$

$$\longrightarrow C' \implies D \quad \text{and} \quad \llbracket C \rrbracket \longrightarrow \llbracket C' \rrbracket \implies \llbracket D \rrbracket \xrightarrow{h^*}$$

as required. \square

Proposition 5.2 For each α, Δ and fresh channels δ, δ' of appropriate type given by α and Δ there exists a process Δ_α defined in Figure 5.1 such that at \mathfrak{f}

$$\Delta; \Theta \vdash C \xrightarrow{\alpha} \Delta, \Delta'; \Theta, \Theta' \vdash C'$$

then

$$\Delta, \llbracket \Theta, \Theta' \rrbracket, \delta : \text{ch}[0], \delta' : \text{ch}[\cdot] \vdash \Delta_\alpha^{\llbracket \Theta \rrbracket}$$

and moreover for balanced D

$$(\Delta; \Theta \vdash D) \xrightarrow{\alpha} (\Delta, \Delta'; \Theta, \Theta' \vdash D')$$

if and only if $\Delta; \Theta \vdash D$ and

$$\Delta_\alpha^{\llbracket \Theta \rrbracket} \parallel \llbracket D \rrbracket_\Theta \implies \nu \Delta'. (\delta \langle \Delta' \rangle \parallel \cdot) \quad \text{with} \quad \llbracket D' \rrbracket_{\Theta, \Theta'} \xrightarrow{h^*} \cdot$$

Proof: It is straightforward to check that $\Delta, \llbracket \Theta, \Theta' \rrbracket, \delta : \text{ch}[0], \delta' : \text{ch}[\cdot] \vdash \Delta_\alpha^{\llbracket \Theta \rrbracket}$ whenever

$$\Delta; \Theta \vdash C \xrightarrow{\alpha} \Delta, \Delta'; \Theta, \Theta' \vdash C'.$$

For the remainder, to show the ‘only if’ direction we use Lemma 5.1 Part 1 to reduce our obligation to the case of a single transition $\xrightarrow{\alpha}$, and we must consider each label α . By way of example we show the case for

$$\xrightarrow{\alpha} (\Delta; \Theta, \cdot \vdash D').$$

then we know that

$$D \equiv \nu \Delta'' . (\tau \cdot \nu \parallel D'')$$

and

$$D' \equiv \nu \Delta'' . (\langle \leftarrow \nu \rangle \parallel D'').$$

We see that for $\sim_{so} \rightarrow \diamond$

$$\begin{aligned} \Delta_\alpha^{\llbracket \Theta \rrbracket} \parallel \llbracket D \rrbracket_\Theta &\equiv (x : \cdot) (* (y : \cdot) x \cdot y \parallel (\delta \langle \cdot \rangle \oplus \delta' \langle \cdot \rangle)) \parallel \nu \Delta'' . (((z : \cdot) \langle z \rangle \mathbf{0}) \cdot \llbracket \nu \rrbracket_\Theta \parallel \llbracket D'' \rrbracket_\Theta) \\ &\implies (\delta \langle \cdot \rangle \oplus \delta' \langle \cdot \rangle) \parallel \nu \Delta'' . (* (y : \cdot) \llbracket \nu \rrbracket_\Theta \cdot y \parallel \llbracket D'' \rrbracket_\Theta) \\ &\implies \delta \langle \cdot \rangle \parallel \llbracket D' \rrbracket_{\Theta, \cdot} \end{aligned}$$

as required.

For the converse direction we suppose that

$$\alpha^{\Delta, [\Theta]} \parallel [[D]]_{\Theta} \Longrightarrow v\Delta' . (\delta\langle\Delta'\rangle \parallel \)$$

Again, we must perform a case analysis on α . We show the case in which α is $v . \langle\tau\rangle$? (the other cases can be treated similarly). We know Δ' is empty so $\alpha^{\Delta, [\Theta]} \parallel [[D]]_{\Theta} \Longrightarrow \delta\langle\rangle \parallel \ .$ Note that $\alpha^{\Delta, [\Theta]}$ has no reductions of its own and can only interact with $[[D]]_{\Theta}$ so we can detail the assumed reductions as

$$\alpha^{\Delta, [\Theta]} \parallel [[D]]_{\Theta} \Longrightarrow \alpha^{\Delta, [\Theta]} \parallel \ 0 \rightarrow (\delta\langle\rangle \oplus \delta'\langle\rangle) \parallel \ 1 \Longrightarrow \delta$$

$$\begin{aligned}
\Delta \text{d}\langle v \rangle? &= d\langle v \rangle(\delta\langle \rangle \oplus \delta'\langle \rangle) \\
\Delta \text{d}\langle v \rangle! &= d\langle x \rangle \text{ if } x = v \text{ then } (\delta\langle \rangle \oplus \delta'\langle \rangle) \text{ else } \mathbf{0} && \text{where } \Delta(d) = \text{ch}[\] \\
vb.\Delta \text{d}\langle b \rangle? &= vb \text{d}\langle b \rangle(\delta\langle b \rangle \oplus \delta'\langle \rangle) && \text{where } \Delta(d) = \text{ch}[\] \\
vb.\Delta \text{d}\langle b \rangle! &= d\langle x \rangle \text{ if } x \notin \Delta \text{ then } (\delta\langle x \rangle \oplus \delta'\langle \rangle) \text{ else } \mathbf{0} && \text{where } \Delta(d) = \text{ch}[\] \\
v.\Delta \text{d}\langle \tau \rangle? &= d\langle (x : \) \ \langle x \rangle \mathbf{0} \rangle(\delta\langle \rangle \oplus \delta'\langle \rangle) && \text{where } \Delta(d) = \text{ch}[\] \text{ and } \sim_{so} \rightarrow \diamond \\
v.\Delta \text{d}\langle \tau \rangle! &= d\langle x \rangle (* (y : \)x \cdot y \parallel (\delta\langle \rangle \oplus \delta'\langle \rangle)) && \text{where } \Delta(d) = \text{ch}[\] \text{ and } \sim_{so} \rightarrow \diamond
\end{aligned}$$

(\oplus represents an encoding of internal choice in $\text{HO}\pi$)

Figure 6: Testing processes for labelled transitions

Theorem 5.4 (Completeness) For a closed term s , $\not\in \text{HO}\pi$

$$\Delta \models \cong_p \quad p \text{ es} \quad \Delta \models \approx$$

Proof: We define over terms of the augmented language to be

$$\Delta; \Theta \models C \ D \quad \text{iff} \quad \Delta, [\Theta] \models [[C]]_{\Theta} \cong_p [[D]]_{\Theta}$$

and show that is a bisimulation. Take $\Delta; \Theta \models C \ D$ and suppose that

$$(\Delta; \Theta \vdash C) \xrightarrow{\alpha} (\Delta, \Delta'; \Theta, \Theta' \vdash C').$$

We know from Proposition 5.2 that

$$\Delta, [\Theta, \Theta'], \delta : \text{ch}[\]_0, \delta' : \text{ch}[\] \vdash \alpha^{\Delta, [\Theta]}$$

and that

$$\alpha^{\Delta, [\Theta]} \parallel [[C]]_{\Theta} \Longrightarrow v\Delta'. (\delta\langle \Delta' \rangle \parallel \)$$

with $[[C']]_{\Theta, \Theta'} \xrightarrow{h}^*$. We know that

$$\Delta, [\Theta] \models [[C]]_{\Theta} \cong_p [[D]]_{\Theta}$$

by the definition of

and we now must show that $\Delta, \Delta' ; \Theta, \Theta$

- [7] J. Riely and M. Hennessy. A typed language for distributed mobile processes. In *Proc O L*. ACM Press, 1998.
- [8] D. Sangiorgi. *Expressing Mobility in Process Algebras: First-Order and Higher-Order Paradigms*. PhD thesis, University of Edinburgh, 1993.
- [9] D. Sangiorgi. Bisimulation for higher-order process calculi. *Information and Computation*, 131(2):141–178, 1996.
- [10] Davide Sangiorgi and David Walker. *Process Calculus: Theory of Mobile Processes*. Cambridge University Press, 2001.
- [11] B. Thomsen. *Calculus for Higher-Order Communication Systems*. PhD thesis, University of London, 1990.
- [12] Jan Vitek and Giuseppe Castagna. Seal: A framework for secure mobile computations. In *Internet Programming Languages*, volume 1686 of *LNC*. Springer-Verlag, 1999.