Contextual equivalence for higher-order π-calculus revisited

Alan Jeffrey [1]

School of Computer Science, Telecommunications and Information Systems
DePaul University
Chicago, US

Julian Rathke [2]

School of Cognitive and Computing Sciences
University of Sussex
Brighton, UK

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 *normal* 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 [1,2,7,12]. 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.

[1] Email: ajeffrey@cs.depaul.edu
[2] Email: julianr@cogs.susx.ac.uk

This is a preliminary version. The final version will be published in Electronic Notes in Theoretical Computer Science
URL: www.elsevier.nl/locate/entcs

Work towards this began several years ago with various proposals for higher-order versions of known calculi [3,11], including the higher-order π-calculus or HOπ [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-calculus which supports only name mobility. This translation is proved to be fully abstract with respect to barbed congruence, but with the restriction to a language of finite types.

While the translation is of interest in its own right, it also turned out to be very useful for providing a powerful fully abstract characterisation of barbed congruence in terms of labelled transition systems and *normal* bisimulations. Providing direct proof techniques for contextual equivalences in higher-order process languages is often considered to be hard [10]. The difficulty arises in establishing soundness of the proof technique, which is tantamount to establishing some sort of contextuality property. It has been seen that the use of a translation of higher- to first-order communication can alleviate this problem and such translations have been employed to this effect [5,9].

However, due to the restriction to finite types for the correctness of these translations, the soundness of the proof technique is only guaranteed for finite types. Given that recursive types are used extensively in π-calculus, for encodings of datatypes and functions, this poses a significant restriction. Sangiorgi has shown that by studying various subcalculi, such as the asynchronous π-calculus, he is able to remove the restriction to finite types [10]. To date, there has been no proof of full abstraction for full HOπ in the presence of recursive types.

In this paper we present an alternative description of labelled transition systems and normal bisimulations for HOπ, which is informed by Sangiorgi's translation of higher-order to first-order communication. Our alternative presentation allows a *direct* proof of soundness for contextual equivalence which makes no use of the translation to first-order π-calculus and, more importantly, makes no restriction on types.

The innovation here lies in the introduction of operators τ_k and $\langle k \Leftarrow v \rangle$ which simulate the triggers Tr_k and meta-notation $\{k := v\}$ of Sangiorgi [9] where k is a unique identifier for the trigger and v is a process abstraction. The crucial difference is that where Sangiorgi gives definitions as HOπ terms for these devices:

$$Tr_k = (x)k\langle x\rangle \qquad \text{and} \qquad \{k := v\} = *k(x)v \cdot x$$

where $k\langle x\rangle$ represents an output on name k and $*k(x)P$ represents a replicated input on name k, we leave the operators uninterpreted. There are no interactions between the operators τ_k and $\langle k \Leftarrow v \rangle$. Rather, we just mimic the behaviour of triggers in the labelled transition systems. The benefit of doing this is that it allows us to obtain a direct soundness proof that (normal) bisimilarity implies contextual equivalence without recourse to any translation

in its correctness proof.

A challenge of approaching the problem in this way is that it is not immediately clear that bisimilarity will be complete for contextual equivalence in HOπ. That is to say, it is not obvious whether each transition has a genuine HOπ context which validates it. At this point however we can interpret the operators τ_k and $\langle k \Leftarrow v \rangle$ as HOπ terms exactly as Sangiorgi does. It is then a simple matter to demonstrate completeness following familiar techniques [2,4,5]. The real payoff is that not only do we obtain a direct soundness proof but the postponement of interpreting the triggers allows us to finesse any restrictions to finite types.

The remainder of the paper is organised as follows: in Section 2 we recall the syntax and semantics of HOπ along with the definition of contextual equivalence which we will be using. This is followed in Section 3 by a presentation of the novel labelled transition system using the operators τ_k and $\langle k \Leftarrow v \rangle$. We prove that bisimilarity over this labelled transition system is sound for contextual equivalence in Section 4 and conversely, that it is complete for contextual equivalence in Section 5. We conclude in Section 6 with some closing remarks.

In this extended abstract, we elide some relatively routine proofs. Since much of the novelty of this paper is in our technique for establishing soundness, we provide all of the proofs in Section 4.

2 Higher-order π calculus

We present the syntax of HOπ in Figure 1. Except for small changes in notation the language is as can be found in [10] with two main differences: firstly, we assume two distinct countably infinite sets of identifiers, $\mathcal{V}$ and $\mathcal{N}$, for variables and channel names respectively. In general we will use x, y, z to range over variables and a, b, c to range over channel names. This variable/name distinction makes the algebraic properties of the language a little cleaner and we are confident that the techniques proposed here would also be applicable if we identified these sets. Secondly we allow communication of channel names as well as process abstractions so that there is a core π-calculus as a direct subcalculus of HOπ.

The reduction semantics for the language is defined in a standard manner: we first introduce the evaluation contexts

$$\mathcal{E} ::= [\,\cdot\,] \quad | \quad \mathcal{E} \parallel P \quad | \quad \nu a \,.\, \mathcal{E}$$

Structural equivalence, $\equiv$ is defined to be the least congruence with respect to $\mathcal{E}$ contexts such that it makes $(\parallel, \mathbf{0})$ into a commutative monoid and moreover satisfies

$$\nu a \,.\, (P \parallel Q) \equiv \nu a \,.\, P \parallel Q \qquad \text{if } a \notin \mathsf{fn}(P)$$
$$*P \equiv *P \parallel P$$

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

Fig. 1. The Syntax

We will now consider processes up to structural equivalence throughout the remainder. We define the reduction relation $\longrightarrow$ as the least precongruence with respect to $\mathcal{E}$ contexts such that the following axioms hold

(comm)	$a\langle v\rangle P \parallel a(x)Q$	$\longrightarrow\ P \parallel (x)Q \cdot v$	
$(\beta-\text{redn})$	$(x)P \cdot v$	$\longrightarrow\ P[v/x]$	
(cond—tt)	$\mathsf{if}\ a = a\ \mathsf{then}\ P\ \mathsf{else}\ Q$	$\longrightarrow\ P$	
(cond—ff)	$\mathsf{if}\ a = b\ \mathsf{then}\ P\ \mathsf{else}\ Q$	$\longrightarrow\ Q$	$(a \neq b)$

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

We introduce a simple type system for the language which comprises types for channels and abstractions. We also allow recursive types of the form $\mathsf{rec}\, Z.T$ where $\mathsf{rec}\,.$ forms a binder and Z is drawn from a countably infinite supply of type variables. We must insist that for any $\mathsf{rec}\, Z.T$ that Z does not appear unguarded in T, that is to say that any free occurrence of Z lies

$$\frac{}{\Gamma \vdash \cdot : \cdot} \qquad \frac{\Gamma(v) = T}{\Gamma \vdash v : T} \qquad \frac{\Gamma \vdash v : T \quad T \sim_{iso} U}{\Gamma \vdash v : U}$$

$$\frac{\Gamma, x : T \vdash P}{\Gamma \vdash (x : T)P : T \to \diamond} \qquad \frac{\Gamma \vdash v : T \to \diamond \quad \Gamma \vdash w : T}{\Gamma \vdash v \cdot w}$$

$$\frac{\begin{array}{c}\Gamma \vdash v : \mathsf{ch}[T], w : \mathsf{ch}[T] \\ \Gamma \vdash P \qquad \Gamma \vdash Q\end{array}}{\Gamma \vdash \mathsf{if}\ v = w\ \mathsf{then}\ P\ \mathsf{else}\ Q} \qquad \frac{\Gamma, a : T \vdash P}{\Gamma \vdash \nu(a : T)\,.\,(P)} \qquad \frac{\Gamma \vdash P,\ Q}{\Gamma \vdash P \parallel Q,\ *P,\ \mathbf{0}}$$

$$\frac{\Gamma, x : T \vdash P \quad \Gamma \vdash v : \mathsf{ch}[T]}{\Gamma \vdash v(x : T)P} \qquad \frac{\Gamma \vdash P \quad \Gamma \vdash w : T \quad \Gamma \vdash v : \mathsf{ch}[T]}{\Gamma \vdash v\langle w \rangle P}$$

Fig. 2. The Typing Rules

within a subexpression of T of the form $\mathsf{ch}[U]$ or $U \to \diamond$. To allow us to infer recursive types for terms we make use of type isomorphism. We define this by letting $\sim_{iso}$ be the least congruence on types which includes

$$\mathsf{rec}\, Z.T \sim_{iso} T[\mathsf{rec}\, Z.T/Z]$$

A type environment Γ is a finite set of mappings from identifiers (channel names or variables) to types with the restriction that channel names a must be mapped to channel types of the form $\mathsf{ch}[T]$. We write $\Gamma, n : T$ to represent the environment made up of the disjoint union of Γ and the mapping n to T. We will call an environment *closed* if it contains mappings of channel names only and will write Δ to indicate this. Type inference rules for the calculus are given in Figure 2. We will call a well-typed process, P, closed if it can be typed as $\Delta \vdash P$ 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 $[\ \cdot_\Gamma\ ]$ in terms. The type inference system is extended to contexts by using the rule

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

We write $C[]$ to denote contexts with at most one hole and $C[P]$ for the term

which results from substituting P into the hole.

For any given channel name a such that $\Delta \vdash a : \mathsf{ch}[\cdot]$ we write $\Delta \models P \Downarrow a$ if there exists some P', P'' such that $P \Longrightarrow \nu\Delta' . (a\langle\cdot\rangle P'' \parallel P')$ with $a \notin \Delta'$. We use type-indexed families of relations $\{\mathcal{R}_\Delta\}$ between closed process terms to describe equivalence. We will write $\mathcal{R}$ to refer to the whole family of relations and

$$\Delta \models P \; \mathcal{R} \; Q$$

to indicate that P and Q are well-typed with respect to Δ and related by $\mathcal{R}_\Delta$. For general process terms we define the *open extension* $\mathcal{R}^o$ of a typed relation $\mathcal{R}$ as

$$\Delta, x_1 : T_1, \ldots, x_n : T_n \models P \; \mathcal{R}^o \; Q$$

holds if for every Δ' disjoint from Δ and every v_i such that $\Delta, \Delta' \vdash v_i : T_i$ we have (for $1 \leq i \leq n$)

$$\Delta, \Delta' \models P[v_1, \ldots, v_n / x_1, \ldots, x_n] \; \mathcal{R} \; Q[v_1, \ldots, v_n / x_1, \ldots, x_n]$$

Note that, in general, for closed terms $\Delta \models P \; \mathcal{R} \; Q$ is not equivalent to $\Delta \models P \; \mathcal{R}^o \; Q$ as $\mathcal{R}^o$ enjoys the weakening property that $\Delta, \Delta' \models P \; \mathcal{R}^o \; Q$ whenever $\Delta \models P \; \mathcal{R}^o \; Q$, even when $\mathcal{R}$ does not. However, the contextual equivalence which we study in this paper is defined as an open extension and therefore will satisfy this weakening.

There are a number of properties of type-indexed relations that we must define:

Symmetry: A type-indexed relation $\mathcal{R}$ is symmetric whenever $\Delta \models P \; \mathcal{R} \; Q$ implies $\Delta \models Q \; \mathcal{R} \; P$.

Reduction closure: A type-indexed relation $\mathcal{R}$ is reduction-closed whenever $\Delta \models P \; \mathcal{R} \; Q$ and $P \to P'$ implies there exists some Q' such that $Q \Longrightarrow Q'$ and $\Delta \models P' \; \mathcal{R} \; Q'$.

Contextuality: A type-indexed relation $\mathcal{R}$ is contextual whenever $\Gamma' \models P \; \mathcal{R}^o \; Q$ and $\Gamma \vdash C[\cdot_{\Gamma'}]$ implies $\Gamma \models C[P] \; \mathcal{R}^o \; C[Q]$.

Barb preservation: A type-indexed relation $\mathcal{R}$ is barb-preserving if $\Delta \models P \; \mathcal{R} \; Q$ and $\Delta \models P \Downarrow a$ implies $\Delta \models Q \Downarrow a$.

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.

For technical convenience it will be useful to work with a lighter definition of contextuality. We say that a relation $\mathcal{R}$ is $\parallel$-contextual if it is preserved by all contexts of the form $[\; \cdot_\Gamma \;] \parallel R$ 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) $\Gamma \models P \cong Q$ *if and only if* $\Gamma \models P \cong_p Q$

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 HOπ. Most notably, the nodes of the transition system are described using an augmented syntax rather than process terms alone. Specifically, for each k drawn from a countable set of names disjoint from $\mathcal{N}$ and $\mathcal{V}$, we introduce two new operators:

$$\tau_k \qquad \text{and} \qquad \langle k \Leftarrow v \rangle$$

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

$$\begin{array}{l} v ::= \ldots (\text{as Figure 1}) \ldots \mid \tau_k \\ C ::= P \mid \langle k \Leftarrow v \rangle \mid \nu a : T \,.\, (C) \mid C \parallel C \end{array}$$

We impose a syntactic restriction on the augmented syntax so that in any configuration C for any given k then $\langle k \Leftarrow v \rangle$ appears at most once in C. Structural equivalence and reduction lift to C in the obvious manner — note that there are no reduction rules given for τ_k and $\langle k \Leftarrow v \rangle$ though. We augment the type rules by considering judgements of the form

$$\Gamma \,;\, \Theta \vdash v : T \qquad \text{and} \qquad \Gamma \,;\, \Theta \vdash C$$

where Θ represents a set of mappings from reference names to types T. The rules in Figure 2 are easily decorated with the extra Θ environment. The further rules required are given by

$$\frac{\Theta(k) = T}{\Gamma \,;\, \Theta \vdash \tau_k : T \to \diamond} \qquad \frac{\Theta(k) = T \quad \Gamma \,;\, \Theta \vdash v : T \to \diamond}{\Gamma \,;\, \Theta \vdash \langle k \Leftarrow v \rangle}$$

Nodes of our labelled transition system then are well-typed closed terms of the augmented language of the form

$$(\Delta \,;\, \Theta \vdash C)$$

The transitions are of the form

$$(\Delta \,;\, \Theta \vdash C) \xrightarrow{\alpha} (\Delta \,;\, \Theta \vdash C) \text{ or } (\Delta \,;\, \Theta \vdash C) \xrightarrow{\tau} (\Delta \,;\, \Theta \vdash C)$$

where visible labels α are given by the grammar:

$$\alpha ::= \nu a\,.\,\alpha \quad | \quad \nu k\,.\,a\langle \tau_k \rangle! \quad | \quad \nu k\,.\,a\langle \tau_k \rangle? \quad | \quad a\langle v \rangle? \quad | \quad a\langle v \rangle!$$

and are presented in Figures 3,4,5 where we write d to mean either a channel name a or an indirect reference name k. The rules in Figures 3, 4 deserve some comment: a convenient way to think of τ_k is as a call to a function *named* k and similarly $\langle k \Leftarrow v \rangle$ as the definition of function named k with body v. With this idea in mind we can look at Figure 4. The first rule is imported directly from HOπ. The second input rule represents the case in which the system under test has previously tried to pass out a function which the tester named k. The tester now can interact with this function by applying it to a base value. Of course, the function definition itself remains intact. Of the two rules for output we see the first one as in HOπ whereas the latter captures the case in which a named function, which the tester has previously sent in to the system, has been applied to a base value. Given that in a context the tester may provide any definition it likes for this function then it is reasonable to expect that the tester may identify the base value which its function has been applied to. This covers the rules for transmission of values of base type. For values of higher-types we look to Figure 3. These follow the same pattern as the rules in Figure 4 but, for input actions, rather than being required to send in a process abstraction, the tester has the lighter task of simply sending in a dummy for an abstraction in the form of a (freshly) named function. For output actions, as the tester cannot identify abstractions through equality checks, each time an abstraction is passed out of the system the tester simply names it and leaves it stored as a function definition. This allows the tester to uniquely identify each abstraction it has been passed and can interrogate them repeatedly at a later stage.

We write $\bar{\alpha}$ to denote the complement of an action α, which is defined to be the action α with the input/output annotation inversed. We will often write $\Longrightarrow$ to mean the reflexive transitive closure of $\stackrel{\tau}{\longrightarrow}$ and $\stackrel{\alpha}{\Longrightarrow}$ to mean $\Longrightarrow \stackrel{\alpha}{\longrightarrow} \Longrightarrow$. The following proposition states that the labelled transition system is well-defined in the sense that the transition relation only relates well-typed terms.

Proposition 3.1 *If* $\Delta\ ;\ \Theta \vdash C$ *and* $(\Delta\ ;\ \Theta \vdash C) \stackrel{\alpha}{\longrightarrow} (\Delta, \Delta'\ ;\ \Theta, \Theta' \vdash C')$ *then* $\Delta, \Delta'\ ;\ \Theta, \Theta' \vdash C'$ *is a valid typing judgement.*

Proof. Straightforward induction. □

We use a standard definition of (weak) bisimilarity to provide our characterisation of $\cong$ for HOπ:

Definition 3.2 We call a symmetric relation, $\mathcal{R}$, between nodes of the labelled transition system a *bisimulation* if whenever $(n, m) \in \mathcal{R}$ we have

- $n \stackrel{\tau}{\longrightarrow} n'$ implies there exists some m' such that $m \Longrightarrow m'$ and $(n', m') \in \mathcal{R}$
- $n \stackrel{\alpha}{\longrightarrow} n'$ implies there exists some m' such that $m \stackrel{\alpha}{\Longrightarrow} m'$ and $(n', m') \in \mathcal{R}$

$$\frac{T \sim_{iso} U \to \diamond}{(\Delta\ ;\ \Theta \vdash a(x:T)P) \xrightarrow{\nu k.a\langle\tau_k\rangle?} (\Delta\ ;\ \Theta, k:U \vdash (x:T)P \cdot \tau_k)}$$

$$\frac{\Theta(k) \sim_{iso} T \to \diamond}{(\Delta\ ;\ \Theta \vdash \langle k \Leftarrow v\rangle) \xrightarrow{\nu l.k\langle\tau_l\rangle?} (\Delta\ ;\ \Theta, l:T \vdash v \cdot \tau_l \parallel \langle k \Leftarrow v\rangle)}$$

$$\frac{\Delta\ ;\ \Theta \vdash v : T \to \diamond}{(\Delta\ ;\ \Theta \vdash a\langle v\rangle P) \xrightarrow{\nu k.a\langle\tau_k\rangle!} (\Delta\ ;\ \Theta, k:T \vdash \langle k \Leftarrow v\rangle \parallel P)}$$

$$\frac{\Theta(k) \sim_{iso} T \to \diamond}{(\Delta\ ;\ \Theta \vdash \tau_k \cdot v) \xrightarrow{\nu l.k\langle\tau_l\rangle!} (\Delta\ ;\ \Theta, l:T \vdash \langle l \Leftarrow v\rangle)}$$

Fig. 3. Basic higher-order labelled transition rules

Let bisimulation equivalence, or bisimilarity, $\approx$ be the largest bisimulation relation.

We will write

$$\Delta\ ;\ \Theta \models C \approx D$$

to mean that $\Delta\ ;\ \Theta \vdash C$ and $\Delta\ ;\ \Theta \vdash D$ are valid typing judgements and moreover, they are related by $\approx$ as nodes of the lts. In order to provide a bisimulation characterisation of $\cong$ over HOπ we will consider a subrelation of $\approx$ by restricting our attention to nodes of the form

$$(\Delta\ ;\ \vdash P)$$

whose terms are clearly definable in HOπ. We will simply write (when Θ is empty)

$$\Delta \models P \approx Q$$

to indicate bisimilarity between such terms of HOπ considered as nodes of the labelled transition system.

4 Soundness of bisimilarity for contextual equivalence

We need to demonstrate that bisimilarity implies contextual equivalence for all HOπ processes. In particular, because of Lemma 2.2, we need only show that bisimilarity is contained in some symmetric, reduction-closed, barb preserving and $\parallel$-contextual relation. The key to achieving this is to study the $\parallel$-context closure of bisimilarity. If we can demonstrate that this is reduction-closed then we have our result. To do this we must establish a decomposition theorem for interactions. For instance, if P and Q are bisimilar and we compose each of them with a process R then suppose

$$P \parallel Q \to S$$

$$\frac{\Delta \vdash v : T \text{ a base type}}{(\Delta \ ; \ \Theta \vdash a(x:T)P) \xrightarrow{a\langle v\rangle?} (\Delta \ ; \ \Theta \vdash (x:T)P \cdot v)}$$

$$\frac{\Theta(k) = T \quad \Delta \vdash w : T \text{ a base type}}{(\Delta \ ; \ \Theta \vdash \langle k \Leftarrow v\rangle) \xrightarrow{k\langle w\rangle?} (\Delta \ ; \ \Theta \vdash v \cdot w \parallel \langle k \Leftarrow v\rangle)}$$

$$\frac{\Delta \vdash v : T \text{ a base type}}{(\Delta \ ; \ \Theta \vdash a\langle v\rangle P) \xrightarrow{a\langle v\rangle!} (\Delta \ ; \ \Theta \vdash P)}$$

$$\frac{\Theta(k) = T \quad T \text{ a base type}}{(\Delta \ ; \ \Theta \vdash \tau_k \cdot v) \xrightarrow{k\langle v\rangle!} (\Delta \ ; \ \Theta \vdash \mathbf{0})}$$

Fig. 4. Basic first-order labelled transition rules

$$\frac{C \rightarrow C'}{(\Delta \ ; \ \Theta \vdash C) \xrightarrow{\tau} (\Delta \ ; \ \Theta \vdash C')} \qquad \frac{(\Delta \ ; \ \Theta \vdash C) \xrightarrow{\alpha} (\Delta' \ ; \ \Theta' \vdash C')}{(\Delta \ ; \ \Theta \vdash C \parallel D) \xrightarrow{\alpha} (\Delta' \ ; \ \Theta' \vdash C' \parallel D)}$$

$$\frac{(\Delta, a:T \ ; \ \Theta \vdash C) \xrightarrow{\alpha} (\Delta, a:T, \Delta' \ ; \ \Theta, \Theta' \vdash C') \quad (a \notin \mathsf{fn}(\alpha))}{(\Delta \ ; \ \Theta \vdash \nu a:T \,.\, C) \xrightarrow{\alpha} (\Delta, \Delta' \ ; \ \Theta, \Theta' \vdash \nu a:T \,.\, C')}$$

$$\frac{(\Delta, b:T \ ; \ \Theta \vdash C) \xrightarrow{d\langle b\rangle!} (\Delta, b:T \ ; \ \Theta \vdash C') \quad (d \neq b)}{(\Delta \ ; \ \Theta \vdash \nu b:T \,.\, C) \xrightarrow{\nu b.d\langle b\rangle!} (\Delta, b:T \ ; \ \Theta \vdash C')}$$

$$\frac{(\Delta, b:T \ ; \ \Theta \vdash C) \xrightarrow{d\langle b\rangle?} (\Delta, b:T \ ; \ \Theta \vdash C') \quad (d \neq b)}{(\Delta \ ; \ \Theta \vdash C) \xrightarrow{\nu b.d\langle b\rangle?} (\Delta, b:T \ ; \ \Theta \vdash C')}$$

Fig. 5. Structural labelled transition rules

represents an interaction between P and R. We decompose this into complementary actions

$$P \xrightarrow{\alpha} P' \quad \text{and} \quad R \xrightarrow{\bar{\alpha}} R'$$

respectively. Note however that S is not necessarily obtained by a parallel composition of the targets of the transitions: $P' \parallel R'$. Instead, P' and R' may contain indirect references and their corresponding resources. These need to be matched up correctly to obtain S. We achieve this by introducing the *merge* (partial) operator $\langle\!\langle \cdot \rangle\!\rangle$ which will match up these terms and replace every indirect reference to an abstraction with the abstraction itself. We write

$$C[v/\tau_k]$$

to denote the substitution of the value v for every instance of the indirect reference τ_k. We define $\langle\!\langle C \rangle\!\rangle$ then as the operator on terms of the augmented syntax (up to $\equiv$) such that

$$\begin{aligned} \langle\!\langle C \rangle\!\rangle &= C && \text{if } \langle k \Leftarrow v \rangle \notin C \text{ for any } k, v \\ \langle\!\langle \nu(\overline{a} : \overline{T}) \,.\, (\langle k \Leftarrow v \rangle \parallel C) \rangle\!\rangle &= \langle\!\langle \nu(\overline{a} : \overline{T}) \,.\, (C[v/\tau_k]) \rangle\!\rangle && \text{if } \tau_k \notin v \end{aligned}$$

Intuitively, this says that we substitute any values stored at a $\langle k \Leftarrow v \rangle$ through for the corresponding τ_k. Note that this need not substitute for all the indirect reference identifiers in C. It is clear that the above definitions are only partial. For example, if C contains an occurrence of $\langle k \Leftarrow v \rangle$ for which τ_k occurs in v, then then $\langle\!\langle C \rangle\!\rangle$ is undefined. In order to identify for which terms the merge is defined we make use of the notion of *reference graph*: For a term C we define the graph $\mathsf{rg}(C)$ to be the graph which has nodes as the indirect reference identifiers k in C and edges

$$k \mapsto l \quad \text{if} \quad \tau_l \in v \quad \text{for} \quad \langle k \Leftarrow v \rangle \quad \text{in} \quad C$$

Proposition 4.1 *$\langle\!\langle \cdot \rangle\!\rangle$ is a well-defined partial function such that $\langle\!\langle C \rangle\!\rangle$ is defined if and only if $\mathsf{rg}(C)$ is acyclic.*

Proof. We consider the rewriting relation $\twoheadrightarrow$ which we will define as the one-step rewriting used to define the merge operation:

$$\begin{aligned} C &\twoheadrightarrow \checkmark && \text{if } \langle k \Leftarrow v \rangle \notin C \text{ for any } k, v \\ \nu(\overline{a} : \overline{T}) \,.\, (\langle k \Leftarrow v \rangle \parallel C) &\twoheadrightarrow \nu(\overline{a} : \overline{T}) \,.\, (C[v/\tau_k]) && \text{if } \tau_k \notin v \end{aligned}$$

It is easy to see that $\twoheadrightarrow$ is a terminating rewriting relation. Moreover, the rewriting will terminate with a $\checkmark$ from C (so that $\langle\!\langle C \rangle\!\rangle$ is defined) exactly when $\mathsf{rg}(C)$ is acyclic. To see this we consider the effect of $\twoheadrightarrow$ on reference graphs: for

$$\langle k \Leftarrow v \rangle \parallel C \qquad \twoheadrightarrow \qquad C[v/\tau_k]$$

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

$$l' \mapsto k \mapsto l$$

for $l', l \neq k$, are replaced with an edge

$$l' \mapsto l$$

all other edges involving k are removed. So if node k is involved in a cycle before rewriting occurs, that is

$$l \mapsto^* k \mapsto^* l$$

for some l, then either it is a *tight loop*, that is $l = k$ and $k \mapsto k$, or $l \neq k$ and the cycle still exist after rewriting as $l \mapsto^* l$. The side-condition on the

rewrite rule forbids tight loops hence we see that $\twoheadrightarrow$ preserves cyclicity. That is:

if $C \twoheadrightarrow C'$ then $\mathsf{rg}(C)$ is acyclic if and only if $\mathsf{rg}(C')$ is acyclic.

Now, suppose that $\langle\!\langle C \rangle\!\rangle$ is defined. We know that there exists a finite sequence

$$C \twoheadrightarrow C_1 \twoheadrightarrow \cdots \twoheadrightarrow C_n \twoheadrightarrow \checkmark$$

with $\langle\!\langle C \rangle\!\rangle = C_n$. We know that $\mathsf{rg}(C_n)$ is acyclic as it contains no edges. Thus, $\mathsf{rg}(C)$ is acyclic also. Conversely, suppose that $\mathsf{rg}(C)$ is acyclic. Then as $\twoheadrightarrow$ is terminating there must be a finite sequence

$$C \twoheadrightarrow C_1 \twoheadrightarrow \cdots \twoheadrightarrow C_n$$

such that C_n cannot be rewritten. There are two possibilities for this: either $\mathsf{rg}(C_n)$ contains a tight loop, or C_n is $\checkmark$. We see that $\mathsf{rg}(C)$ is acyclic, so C_n is acyclic too and therefore cannot contain a tight loop. Thus C_n is $\checkmark$ and $\langle\!\langle C \rangle\!\rangle$ is defined.

To show that $\langle\!\langle \cdot \rangle\!\rangle$ is a well-defined partial function it suffices to show that it is strongly confluent for acyclic terms. Note that if $\nu a : T \,.\, (C) \twoheadrightarrow C'$ then either C' is $\checkmark$ or $C' \equiv \nu a : T \,.\, (C'')$ such that $C \twoheadrightarrow C''$. So without loss of generality suppose that

$$C \twoheadrightarrow C_1 \qquad \text{and} \qquad C \twoheadrightarrow C_2$$

for

$$C \equiv C_1' \parallel \langle k_1 \Leftarrow v_1 \rangle \qquad \text{and} \qquad C \equiv C_2' \parallel \langle k_2 \Leftarrow v_2 \rangle$$

so that

$$C_1 \equiv C_1'[v_1/\tau_{k_1}] \qquad \text{and} \qquad C_2 \equiv C_2'[v_2/\tau_{k_2}].$$

So either, $k_1 = k_2$ in which case $C_1 \equiv C_2$ or $k_1 \neq l_2$ and

$$C_1' \equiv C_3' \parallel \langle k_2 \Leftarrow v_2 \rangle \qquad \text{and} \qquad C_2' \equiv C_3' \parallel \langle k_1 \Leftarrow v_1 \rangle$$

We notice that

$$\begin{aligned}
C_1 &\equiv C_1'[v_1/\tau_{k_1}] \\
&\equiv (C_3' \parallel \langle k_2 \Leftarrow v_2 \rangle)[v_1/\tau_{k_1}] \\
&\equiv C_3'[v_1/\tau_{k_1}] \parallel \langle k_2 \Leftarrow v_2[v_1/\tau_{k_1}] \rangle \\
&\twoheadrightarrow C_3'[v_1/\tau_{k_1}][v_2[v_1/\tau_{k_1}]/\tau_{k_2}] && \text{(acyclicity implies } \tau_{k_2} \notin v_2[v_1/\tau_{k_1}]) \\
&\equiv C_3'[v_1[v_2[v_1/\tau_{k_1}]/\tau_{k_2}]/\tau_{k_1}, v_2[v_1/\tau_{k_1}]/\tau_{k_2}] \\
&\equiv C_3'[v_1[v_2/\tau_{k_2}]/\tau_{k_1}, v_2[v_1/\tau_{k_1}]/\tau_{k_2}] && \text{(acyclicity)} \\
&\equiv C_3 && \text{(def)}
\end{aligned}$$

By a symmetric argument we see that $C_2 \twoheadrightarrow C_3'[v_2[v_1/\tau_{k_1}]/\tau_{k_2}, v_1[v_2/\tau_{k_2}]/\tau_{k_1}]$ and, by definition, this is just C_3 so we have $C_2 \twoheadrightarrow C_3$. Thus $\twoheadrightarrow$ is strongly confluent for acyclic terms and hence $\langle\langle\cdot\rangle\rangle$ is well-defined. □

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

(i) *If* $\langle\langle C \parallel D\rangle\rangle \equiv E$
and $(\Delta\ ;\ \Theta \vdash C) \xrightarrow{\alpha} (\Delta, \Delta'\ ;\ \Theta, \Theta' \vdash C')$
and $(\Delta\ ;\ \Theta \vdash D) \xrightarrow{\bar{\alpha}} (\Delta, \Delta'\ ;\ \Theta, \Theta' \vdash D')$
then there exists a E' *such that* $E \Longrightarrow E'$ *and* $\langle\langle \nu\Delta' \,.\, (C' \parallel D')\rangle\rangle = E'$

(ii) *If* $\langle\langle C\rangle\rangle \equiv E$ *and* $C \longrightarrow C'$
then there exists a E' *such that* $E \longrightarrow E'$ *and* $\langle\langle C'\rangle\rangle \equiv E'$

(iii) *If* $\langle\langle C \parallel D\rangle\rangle \equiv E$ *and* $E \longrightarrow E'$ *then one of the following holds*
$C \longrightarrow C'$ *with* $\langle\langle C' \parallel D\rangle\rangle \equiv E'$
or $D \longrightarrow D'$ *with* $\langle\langle C \parallel D'\rangle\rangle \equiv E'$
or $(\Delta\ ;\ \Theta \vdash C) \overset{\alpha}{\Longrightarrow} (\Delta, \Delta'\ ;\ \Theta, \Theta' \vdash C')$
and $(\Delta\ ;\ \Theta \vdash D) \overset{\bar{\alpha}}{\Longrightarrow} (\Delta, \Delta'\ ;\ \Theta, \Theta' \vdash D')$
with $\langle\langle \nu\Delta' \,.\, (C' \parallel D')\rangle\rangle \equiv E'$.

Proof. Part (ii) is straightforward as the merge operator $\langle\langle\ \rangle\rangle$ simply removes subterm of the form $\langle k \Leftarrow v\rangle$, which cannot be involved in reductions, and substitutes higher-order values through for variables of higher-order type. Reductions are based on structure alone except for the conditionals which can be affected by first-order substitutions of channel names only.

To show (i) we must consider all the possible cases for α. By symmetry there are four distinct pairs of complementary actions. We only consider the cases where α is $\nu k \,.\, a\langle\tau_k\rangle?$ and $\nu l \,.\, k\langle\tau_l\rangle?$ as the first-order actions can be treated similarly.

Case: $\Delta\ ;\ \Theta \vdash C \xrightarrow{\nu k.a\langle\tau_k\rangle?} \Delta\ ;\ \Theta, k : U \vdash C'$ and $\Delta\ ;\ \Theta \vdash D \xrightarrow{\nu k.a\langle\tau_k\rangle!} \Delta\ ;\ \Theta, k : U \vdash D'$. By inspection we see that

- $C \equiv \nu\Delta' \,.\, (a(x : T)P \parallel C'')$ with $T \sim_{iso} U \to \diamond$
- $C' \equiv \nu\Delta' \,.\, ((x : T)P \cdot \tau_k \parallel C'')$
- $D \equiv \nu\Delta'' \,.\, (a\langle v\rangle Q \parallel D'')$
- $D' \equiv \nu\Delta'' \,.\, (\langle k \Leftarrow v\rangle \parallel Q \parallel D'')$

It is easy to see that $\langle\langle C \parallel D\rangle\rangle \longrightarrow \langle\langle \nu\Delta', \Delta'' \,.\, ((x : T)P \cdot v \parallel C'' \parallel Q \parallel D'')\rangle\rangle$ let us call the target of this reduction E'. We simply need to check

$$\begin{aligned} E' &\equiv \langle\langle \nu\Delta', \Delta'' \,.\, ((x : T)P \cdot v \parallel C'' \parallel Q \parallel D'')\rangle\rangle \\ (\tau_k \notin v) \quad &\equiv \langle\langle \nu\Delta' \,.\, ((x : T)P \cdot \tau_k \parallel C'') \parallel \nu\Delta'' \,.\, (\langle k \Leftarrow v\rangle \parallel Q \parallel D'')\rangle\rangle \\ &\equiv \langle\langle C' \parallel D'\rangle\rangle \end{aligned}$$

Case: $\Delta\ ;\ \Theta \vdash C \xrightarrow{\nu l.k\langle\tau_l\rangle?} \Delta\ ;\ \Theta, l : T \vdash C'$ and $\Delta\ ;\ \Theta \vdash D \xrightarrow{\nu l.k\langle\tau_l\rangle!} \Delta\ ;\ \Theta, l : T \vdash D'$. Again, by inspection we see that

- $C \equiv \nu\Delta' \,.\, (\langle k \Leftarrow v\rangle \parallel C'')$
- $C' \equiv \nu\Delta' \,.\, (v \cdot \tau_l \parallel \langle k \Leftarrow v\rangle \parallel C'')$
- $D \equiv \nu\Delta'' \,.\, (\tau_k \cdot w \parallel D'')$
- $D' \equiv \nu\Delta'' \,.\, (\langle l \Leftarrow w\rangle \parallel D'')$

Note that the previous proposition tells us that $\mathsf{rg}(C \parallel D)$ must be acyclic — in particular, $\tau_k \notin v$. Here we see that

$$\begin{aligned}
\langle\!\langle C \parallel D\rangle\!\rangle &\equiv \langle\!\langle \nu\Delta', \Delta'' \,.\, (\langle k \Leftarrow v\rangle \parallel C'' \parallel \tau_k \cdot w \parallel D'')\rangle\!\rangle \\
(\tau_k \notin v) \qquad &\equiv \langle\!\langle \nu\Delta', \Delta'' \,.\, (\langle k \Leftarrow v\rangle \parallel C'' \parallel v \cdot w \parallel D'')\rangle\!\rangle \\
(\tau_l \notin v, w, C'', D'') \quad &\equiv \langle\!\langle \nu\Delta', \Delta'' \,.\, (\langle k \Leftarrow v\rangle \parallel C'' \parallel v \cdot \tau_l \parallel \langle l \Leftarrow w\rangle \parallel D'')\rangle\!\rangle \\
&\equiv \langle\!\langle C' \parallel D'\rangle\!\rangle
\end{aligned}$$

So by letting E' be $\langle\!\langle C' \parallel D'\rangle\!\rangle$ we note that $\langle\!\langle C \parallel D\rangle\!\rangle \Longrightarrow E'$ as required.

To show (iii) we suppose $\langle\!\langle C \parallel D\rangle\!\rangle \equiv E$ and that $E \longrightarrow E'$. We must consider all possible ways in which this reduction can occur. If the reduction arises from a conditional then it is clear that we must have $C \longrightarrow C'$ or $D \longrightarrow D'$ for some C' or D'. Moreover it is easy to check that $\langle\!\langle C' \parallel D\rangle\!\rangle$ (resp $\langle\!\langle C \parallel D'\rangle\!\rangle$) $\equiv E'$. There are two more possibilities to consider:

Case: the reduction arises from a β-reduction. In this case either $C \longrightarrow C'$ or $D \longrightarrow D'$ as above and the result follows easily, or

- $C \equiv \nu\Delta' \,.\, (\tau_k \cdot w \parallel C'')$ with all names in Δ' appearing in w
- $D \equiv \nu\Delta'' \,.\, (\langle k \Leftarrow v\rangle \parallel D'')$ with $\tau_k \notin v$
- $E' \equiv \langle\!\langle \nu\Delta', \Delta'' \,.\, (v \cdot w \parallel C'' \parallel \langle k \Leftarrow v\rangle \parallel D'')\rangle\!\rangle$

or a symmetric version of these with the roles of C and D reversed. So we notice that if $\Theta(k) \sim_{iso} T \to \diamond$, we have

$$\Delta\ ;\ \Theta \vdash C \xrightarrow{\nu l.k\langle\tau_l\rangle!} \Delta\ ;\ \Theta, l : T \vdash C' \quad \text{and} \quad \Delta\ ;\ \Theta \vdash D \xrightarrow{\nu l.k\langle\tau_l\rangle?} \Delta\ ;\ \Theta, l : T \vdash D'$$

where $C' \equiv \nu\Delta' \,.\, (\langle l \Leftarrow w\rangle \parallel C'')$ and $D' \equiv \nu\Delta'' \,.\, (v \cdot \tau_l \parallel \langle k \Leftarrow v\rangle \parallel D'')$. We check:

$$\begin{aligned}
\langle\!\langle C' \parallel D'\rangle\!\rangle &\equiv \langle\!\langle \nu\Delta' \,.\, (\langle l \Leftarrow w\rangle \parallel C'') \parallel \nu\Delta'' \,.\, (v \cdot \tau_l \parallel \langle k \Leftarrow v\rangle \parallel D'')\rangle\!\rangle \\
(\tau_l \notin v, w, C'', D'') \qquad &\equiv \langle\!\langle \nu\Delta', \Delta'' \,.\, (C'' \parallel v \cdot w \parallel \langle k \Leftarrow v\rangle \parallel D'')\rangle\!\rangle \\
&\equiv E'
\end{aligned}$$

as required. Alternatively, it could be that $\Theta(k)$ is a base type, in which case

$$\Delta\ ;\ \Theta \vdash C \xrightarrow{\nu\Delta'.k\langle w\rangle!} \Delta, \Delta'\ ;\ \Theta \vdash C' \quad \text{and} \quad \Delta\ ;\ \Theta \vdash D \xrightarrow{\nu\Delta'.k\langle w\rangle?} \Delta, \Delta'\ ;\ \Theta \vdash D'$$

where $C' \equiv C''$ and $D' \equiv \nu\Delta''\,.\,(v \cdot w \parallel \langle k \Leftarrow v\rangle \parallel D'')$. It is easy to check that $\langle\!\langle C' \parallel D'\rangle\!\rangle \equiv E'$ as required.

Case: the reduction arises from communication. Again we see that either $C \longrightarrow C'$ or $D \longrightarrow D'$, in which case we easily obtain the result, or

- $C \equiv \nu\Delta'\,.\,(a\langle v\rangle P \parallel C'')$
- $D \equiv \nu\Delta''\,.\,(a(x:T)Q \parallel D'')$
- $E' \equiv \langle\!\langle \nu\Delta'\,.\,(P \parallel C'') \parallel \nu\Delta''\,.\,((x:T)Q \cdot v \parallel D'')\rangle\!\rangle$

or a symmetric version of this with the roles of C and D reversed. Again we must consider whether the type T is a base type or higher-order. We omit the details of the former case. Suppose then that $\Delta\ ;\ \Theta \vdash v : T \sim_{iso} U \to \diamond$ we know

$$\Delta\ ;\ \Theta \vdash C \xrightarrow{\nu k.a\langle\tau_k\rangle!} \Delta\ ;\ \Theta, k:U \vdash C' \quad \text{and} \quad \Delta\ ;\ \Theta \vdash D \xrightarrow{\nu k.a\langle\tau_k\rangle?} \Delta\ ;\ \Theta, k:U \vdash D'$$

where $C' \equiv \nu\Delta'\,.\,(\langle k \Leftarrow v\rangle \parallel P \parallel C'')$ and $D' \equiv \nu\Delta''\,.\,((x:T)Q \cdot \tau_k \parallel D'')$. We check:

$$\begin{aligned} \langle\!\langle C' \parallel D'\rangle\!\rangle &\equiv \langle\!\langle \nu\Delta'\,.\,(\langle k \Leftarrow v\rangle \parallel P \parallel C'') \parallel \nu\Delta''\,.\,((x:T)Q \cdot \tau_k \parallel D'')\rangle\!\rangle \\ (\tau_k \notin v, P, C'', D'') \qquad &\equiv \langle\!\langle \nu\Delta', \Delta''\,.\,(P \parallel C'' \parallel (x:T)Q \cdot v \parallel D'')\rangle\!\rangle \\ &\equiv E' \end{aligned}$$

as required. □

Let $\approx_m$ be defined to be

$$\Delta\ ;\ \Theta \models \langle\!\langle C_1 \parallel D\rangle\!\rangle \approx_m \langle\!\langle C_2 \parallel D\rangle\!\rangle$$

if and only if

$$\Delta\ ;\ \Theta \models C_1 \approx C_2 \quad \text{and} \quad \Delta\ ;\ \Theta \vdash D$$

whenever $\langle\!\langle C_1 \parallel D\rangle\!\rangle$ and $\langle\!\langle C_2 \parallel D\rangle\!\rangle$ are defined.

Lemma 4.3 *$\approx_m$ is reduction-closed.*

Proof. Follows easily from the previous lemma. Take $\Delta\ ;\ \Theta \models \langle\!\langle C_1 \parallel D\rangle\!\rangle \approx_m \langle\!\langle C_2 \parallel D\rangle\!\rangle$ and suppose $\langle\!\langle C_1 \parallel D\rangle\!\rangle \longrightarrow E$. We must show that $\langle\!\langle C_2 \parallel D\rangle\!\rangle \longrightarrow E'$ for some E' such that $\Delta\ ;\ \Theta \models E \approx_m E'$. We know from Part (iii) of the previous lemma that one of three cases must hold. Either, $C_1 \longrightarrow C_1'$, $D \longrightarrow D'$ or there are complementary actions from both C_1 and D. We only deal with the last case as the others follow easily from the hypothesis that $\Delta\ ;\ \Theta \models C_1 \approx C_2$ and Part (ii) of the previous lemma.

We have then that

$$\Delta\ ;\ \Theta \vdash C_1 \stackrel{\alpha}{\Longrightarrow} \Delta, \Delta'\ ;\ \Theta, \Theta' \vdash C_1'$$

and

$$\Delta\ ;\ \Theta \vdash D \stackrel{\bar{\alpha}}{\Longrightarrow} \Delta, \Delta'\ ;\ \Theta, \Theta' \vdash D'$$

such that $E \equiv \langle\langle C_1' \parallel D' \rangle\rangle$. We know by hypothesis that there must exist some

$$\Delta \; ; \; \Theta \vdash C_2 \stackrel{\alpha}{\Longrightarrow} \Delta, \Delta' \; ; \; \Theta, \Theta' \vdash C_2'$$

such that

$$\Delta, \Delta' \; ; \; \Theta, \Theta' \models C_1' \approx C_2'. \qquad (\dagger)$$

We can now use Parts (i) and (ii) of the previous lemma to see that $\langle\langle C_2 \parallel D \rangle\rangle \Longrightarrow E'$ such that $E' \equiv \langle\langle C_2' \parallel D' \rangle\rangle$. Note that ($\dagger$) guarantees $\Delta \; ; \; \Theta \models E \approx_m E'$ to finish. □

Theorem 4.4 *For all closed terms P, Q of HOπ:*

$$\Delta \models P \approx Q \quad \textit{implies} \quad \Delta \models P \cong_p Q$$

Proof. We let $\approx_p$ denote the relation

$$\Delta, \Delta' \models (P \parallel R) \approx_p (Q \parallel R) \text{ iff } \Delta \models P \approx Q \text{ and } \Delta, \Delta' \vdash R$$

It is easy to see that $\approx_p$ is a $\parallel$-contextual relation over terms of HOπ. It is also easy to see that $\approx_p$ is symmetric and barb preserving and coincides with $\approx_m$ for closed terms of HOπ, thus Lemma 4.3 can be instantiated to demonstrate that $\approx_p$ is reduction-closed and, given that $\cong_p$ is defined to be the largest symmetric, $\parallel$-contextual, reduction-closed, and barb-preserving relation over terms of HOπ, then we have our result. □

Corollary 4.5 (Soundness) *For all terms P, Q of HOπ:*

$$\Gamma \models P \approx^o Q \quad \textit{implies} \quad \Gamma \models P \cong Q$$

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 HOπ 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 HOπ:

$$\begin{aligned}
[\![k_1 : T_1, \ldots, k_n : T_n]\!] &= k_1 : \mathsf{ch}[T_1], \ldots, k_n : \mathsf{ch}[T_n] \\
[\![\Gamma \; ; \; \Theta \vdash C]\!] &= \Gamma, [\![\Theta]\!] \vdash [\![C]\!]_\Theta \\
[\![\tau_k]\!]_\Theta &= (x : T)k\langle x \rangle \mathbf{0} \qquad \text{if } \Theta(k) = T \\
[\![\langle k \Leftarrow v \rangle]\!]_\Theta &= *k[\![v]\!]_\Theta
\end{aligned}$$

The translation acts homomorphically on all other terms. We abuse notation here by using identifiers k 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 HOπ.

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 $\langle k \Leftarrow v \rangle$ and τ_k we provide matching input and output prefixes, which, in HOπ 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 HOπ, then terms reachable by transitions are *balanced* in the following sense: we call a term C of the augmented language *balanced* if for each k then C contains at most one of τ_k (possible multiple times) or $\langle k \Leftarrow v \rangle$. Unfortunately the translation may introduce extra reductions which aren't present in the source term. These arise through the translation of terms of the form $\tau_k \cdot v$. Note that

$$[\![\tau_k \cdot v]\!] = (x : T)k\langle x\rangle \mathbf{0} \cdot [\![v]\!] \xrightarrow{\tau} k\langle [\![v]\!]\rangle \mathbf{0}$$

but $\tau_k \cdot v$ has no corresponding reduction. We will identify these rogue reductions as housekeeping reductions and indicate them with $\xrightarrow{\mathsf{h}}$ defined as any reduction which can be derived using the axiom

$$(\mathsf{h} - \text{redn}) \qquad (x : T)k\langle x\rangle \mathbf{0} \cdot v \longrightarrow k\langle v\rangle \mathbf{0}$$

Lemma 5.1 *If* $\Delta \,;\, \Theta \vdash C$ *is balanced then*

(i) *If* $C \Longrightarrow C'$ *then* $[\![C]\!]_\Theta \Longrightarrow [\![C']\!]_\Theta$

(ii) *If* $[\![C]\!]_\Theta \Longrightarrow P$ *then* $[\![C]\!]_\Theta \Longrightarrow [\![D]\!]_\Theta \xrightarrow{\mathsf{h}}{}^* P$ *for some* $\Delta \,;\, \Theta \vdash D$ *such that* $C \Longrightarrow D$.

Proposition 5.2 *For each* α, Δ *and fresh channels* δ, δ' *of appropriate type given by* α *and* Δ, *there exists a process* $\mathcal{T}_\alpha^\Delta$ *(defined in Figure 6) in HO*π *such that if*

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

then

$$\Delta, [\![\Theta, \Theta']\!], \delta : \mathsf{ch}[T_0], \delta' : \mathsf{ch}[\cdot] \vdash \mathcal{T}_\alpha^{\Delta, [\![\Theta]\!]}$$

and moreover, for balanced D

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

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

$$\mathcal{T}_\alpha^{\Delta, [\![\Theta]\!]} \parallel [\![D]\!]_\Theta \Longrightarrow \nu\Delta' \,.\, (\delta\langle\Delta'\rangle \parallel P) \qquad \textit{with} \qquad [\![D']\!]_{\Theta,\Theta'} \xrightarrow{\mathsf{h}}{}^* P.$$

$$
\begin{array}{lll}
\mathcal{T}^{\Delta}_{d\langle v\rangle?} & = d\langle v\rangle(\delta\langle\rangle \oplus \delta'\langle\rangle) & \\
\mathcal{T}^{\Delta}_{d\langle v\rangle!} & = d(x:T)\mathsf{if}\ x = v\ \mathsf{then}\ (\delta\langle\rangle \oplus \delta'\langle\rangle)\ \mathsf{else}\ \mathbf{0} & \text{where } \Delta(d) = \mathsf{ch}[T] \\
\mathcal{T}^{\Delta}_{\nu b.d\langle b\rangle?} & = \nu b:T\,.\,(d\langle b\rangle(\delta\langle b\rangle \oplus \delta'\langle\rangle)) & \text{where } \Delta(d) = \mathsf{ch}[T] \\
\mathcal{T}^{\Delta}_{\nu b.d\langle b\rangle!} & = d(x:T)\mathsf{if}\ x \notin \Delta\ \mathsf{then}\ (\delta\langle x\rangle \oplus \delta'\langle\rangle)\ \mathsf{else}\ \mathbf{0} & \text{where } \Delta(d) = \mathsf{ch}[T] \\
\mathcal{T}^{\Delta}_{\nu k.d\langle\tau_k\rangle?} & = d\langle(x:U)k\langle x\rangle\mathbf{0}\rangle(\delta\langle\rangle \oplus \delta'\langle\rangle) & \text{where } \Delta(d) \sim_{iso} \mathsf{ch}[U \to \diamond] \\
\mathcal{T}^{\Delta}_{\nu k.d\langle\tau_k\rangle!} & = d(x:T)(*l(y:U)x\cdot y \parallel (\delta\langle\rangle \oplus \delta'\langle\rangle)) & \text{where } \Delta(d) \sim_{iso} \mathsf{ch}[U \to \diamond]
\end{array}
$$

($\oplus$ represents an encoding of internal choice in HOπ)

Fig. 6. Testing processes for labelled transitions

Lemma 5.3 (Extrusion) *If* $\Delta \models \nu\Delta'.(\delta\langle\Delta'\rangle \parallel P) \cong_p \nu\Delta'.(\delta\langle\Delta'\rangle \parallel Q)$ *then* $\Delta, \Delta' \models P \cong_p Q$.

Theorem 5.4 (Completeness) *For all closed terms* P, Q *of HO*π*:*

$$\Delta \models P \cong_p Q \qquad \textit{implies} \qquad \Delta \models P \approx Q$$

Proof. We define $\mathcal{R}$ over terms of the augmented language to be

$$\Delta\ ;\ \Theta \models C\ \mathcal{R}\ D \qquad \text{iff} \qquad \Delta, [\![\Theta]\!] \models [\![C]\!]_\Theta \cong_p [\![D]\!]_\Theta$$

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

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

We know from Proposition 5.2 that

$$\Delta, [\![\Theta, \Theta']\!], \delta : \mathsf{ch}[T_0], \delta' : \mathsf{ch}[\cdot] \vdash \mathcal{T}_\alpha^{\Delta, [\![\Theta]\!]}$$

and that

$$\mathcal{T}_\alpha^{\Delta, [\![\Theta]\!]} \parallel [\![C]\!]_\Theta \Longrightarrow \nu\Delta'\,.\,(\delta\langle\Delta'\rangle \parallel P)$$

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

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

by the definition of $\mathcal{R}$, and hence, by contextuality we also have

$$\Delta, [\![\Theta, \Theta']\!], \delta : \mathsf{ch}[T_0], \delta' : \mathsf{ch}[\cdot] \models \mathcal{T}_\alpha^{\Delta, [\![\Theta]\!]} \parallel [\![C]\!]_\Theta \cong_p \mathcal{T}_\alpha^{\Delta, [\![\Theta]\!]} \parallel [\![D]\!]_\Theta$$

This tells us that

$$\mathcal{T}_\alpha^{\Delta, [\![\Theta]\!]} \parallel [\![D]\!]_\Theta \Longrightarrow Q'$$

such that

$$\Delta, [\![\Theta, \Theta']\!] \models \nu\Delta' \,.\, (\delta\langle\Delta'\rangle \parallel P) \cong_p Q'. \qquad (\dagger)$$

But by the construction of $\mathcal{T}_\alpha^{\Delta,[\![\Theta]\!]}$ we notice that $\nu\Delta' \,.\, (\delta\langle\Delta'\rangle \parallel P)$ barbs on δ but not on δ'. Therefore, by the preservation of barbs property of $\cong_p$, we know that Q' must also barb on δ but not on δ'. This constrains Q' so that $Q' \equiv \nu\Delta' \,.\, (\delta\langle\Delta'\rangle \parallel Q)$. We apply Lemma 5.1 Part ii to $\mathcal{T}_\alpha^{\Delta,[\![\Theta]\!]} \parallel [\![D]\!]_\Theta \Longrightarrow Q'$ to see that there is some D'' such that $\mathcal{T}_\alpha^{\Delta,[\![\Theta]\!]} \parallel [\![D]\!]_\Theta \Longrightarrow [\![D'']\!]_{\Theta,\Theta'} \xrightarrow{\mathsf{h}}{}^* \nu\Delta' \,.\, (\delta\langle\Delta'\rangle \parallel Q)$ from which it clearly follows that $D'' \equiv \nu\Delta' \,.\, (\delta\langle\Delta'\rangle \parallel D')$ and $[\![D']\!]_{\Theta,\Theta'} \xrightarrow{\mathsf{h}}{}^* Q$. We use Proposition 5.2 again to see that

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

and we now must show that $\Delta, \Delta' \,;\, \Theta, \Theta' \models C' \;\mathcal{R}\; D'$. To do this we use Lemma 5.3 on $(\dagger)$ (note that $Q' \equiv \nu\Delta'.(\delta\langle\Delta'\rangle \parallel Q)$) to see that $\Delta, \Delta', [\![\Theta, \Theta']\!] \models P \cong_p Q$. It is also easy to check that h-reductions are confluent with respect to all other reductions and hence preserve contextual equivalence, that is $\xrightarrow{\mathsf{h}}{}^* \subseteq \cong_p$, so we also have $\Delta, \Delta', [\![\Theta, \Theta']\!] \models [\![C']\!]_{\Theta,\Theta'} \cong_p [\![D']\!]_{\Theta,\Theta'}$ because $[\![C']\!]_{\Theta,\Theta'} \xrightarrow{\mathsf{h}}{}^* P$ and $[\![D']\!]_{\Theta,\Theta'} \xrightarrow{\mathsf{h}}{}^* Q$. This allows us to conclude $\Delta, \Delta' \,;\, \Theta, \Theta' \models C' \;\mathcal{R}\; D'$ as required. We must also consider transitions of the form $(\Delta \,;\, \Theta \vdash C) \xrightarrow{\tau} (\Delta, \Delta' \,;\, \Theta, \Theta' \vdash C')$. These can be dealt with as above but in this case no $\mathcal{T}_\alpha^\Delta$ is needed. □

Corollary 5.5 (Full abstraction) *For all terms P, Q of HOπ:*

$$\Gamma \models P \approx^o Q \qquad \textit{if and only if} \qquad \Gamma \models P \cong Q$$

Proof. Follows from Corollary 4.5, Lemma 2.2, and the previous theorem. □

6 Concluding remarks

We have re-examined the use of labelled transitions to characterise contextual equivalence in the higher-order π calculus. The technique of augmenting the core syntax with extra operators to assist in the definition of the labelled transitions allows use to give a direct proof of soundness of bisimilarity for contextual equivalence. This advances Sangiorgi's analagous result by allowing recursive types also.

We believe that the technique of using extra operators to describe the *points of interaction* with the environment in the lts is fairly robust and should be applicable to many higher-order languages. Indeed, this was the approach that the authors developed for their work on concurrent objects [6].

We have only concerned ourselves with the characterisation of contextual equivalence in HOπ and so far have not studied Sangiorgi's translation of higher-order to first-order mobility. Thus, the restriction to finite types for

his translation is still necessary. It would be interesting to investigate whether the current work could be of use in removing this type restriction for his translation also.

References

[1] Cardelli, L. and A. Gordon, *Mobile ambients*, in: *Proc. FoSSaCS '98*, LNCS (1998).

[2] Fournet, C., G. Gonthier, J.-J. Levy, L. Maranget and D. Remy, *A calculus of mobile agents*, in: *Proc. CONCUR*, Lecture notes in computer science **1119** (1996).

[3] Giacalone, A., P. Mishra and S. Prasad, *Facile: A symmetric integration of concurrent and functional programming*, in: *Proceedings TAPSOFT89 conference*, Lecture Notes in Computer Science **352** (1989), pp. 184–209.

[4] Hennessy, M. and J. Rathke, *Typed behavioural equivalences for processes in the presence of subtyping*, in: *Proceedings Computing: the Australasian Theory Symposium CATS 2002*, Electronic notes in theoretical computer science (2002).

[5] Jeffrey, A. and J. Rathke, *A theory of bisimulation for a fragment of concurrent ml with local names*, in: *Proc. LICS2000,* 15^{th} *Annual Symposium on Logic in Computer Science, Santa Barbara* (2000), pp. 311–321.

[6] Jeffrey, A. and J. Rathke, *A fully abstract may testing semantics for concurrent objects*, in: *Proc. Lics2002,* 17^{th} *Annual Symposium on Logic in Computer Science, Copenhagen* (2002), pp. 101–112.

[7] Riely, J. and M. Hennessy, *A typed language for distributed mobile processes*, in: *Proc. POPL* (1998).

[8] Sangiorgi, D., "Expressing Mobility in Process Algebras: First-Order and Higher-Order Paradigms," Ph.D. thesis, University of Edinburgh (1993).

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

[10] Sangiorgi, D. and D. Walker, "The *pi*-calculus: A Theory of mobile processes," Cambridge University Press, 2001.

[11] Thomsen, B., "Calculi for Higher-Order Communicating Systems," Ph.D. thesis, University of London (1990).

[12] Vitek, J. and G. Castagna, *Seal: A framework for secure mobile computations*, in: *Internet Programming Languages*, LNCS **1686** (1999).