Xu X. Expressing first-order π-calculus in higher-order calculus of communicating systems. JOURNAL OF COMPUTER SCIENCE AND TECHNOLOGY 24(1): 122–137 Jan. 2009

Expressing First-Order π-Calculus in Higher-Order Calculus of Communicating Systems

Xian Xu (徐 贤)

Department of Computer Science and Engineering, Shanghai Jiao Tong University, Shanghai 200240, China

Department of Computer Science and Engineering, East China University of Science and Technology Shanghai 200237, China

E-mail: xuxian@sjtu.edu.cn; xuxian2004@gmail.com

Received July 7, 2007; revised July 9, 2008.

Abstract In the study of process calculi, encoding between different calculi is an effective way to compare the expressive power of calculi and can shed light on the essence of where the difference lies. Thomsen and Sangiorgi have worked on the higher-order calculi (higher-order Calculus of Communicating Systems (CCS) and higher-order π-calculus, respectively) and the encoding from and to first-order π-calculus. However a fully abstract encoding of first-order π-calculus with higher-order CCS is not available up-today. This is what we intend to settle in this paper. We follow the encoding strategy, first proposed by Thomsen, of translating first-order π-calculus into Plain CHOCS. We show that the encoding strategy is fully abstract with respect to early bisimilarity (first-order π-calculus) and wired bisimilarity (Plain CHOCS) (which is a bisimulation defined on wired processes only sending and receiving wires), that is the core of the encoding strategy. Moreover from the fact that the wired bisimilarity is contained by the well-established context bisimilarity, we secure the soundness of the encoding, with respect to early bisimilarity and context bisimilarity. We use index technique to get around all the technical details to reach these main results of this paper. Finally, we make some discussion on our work and suggest some future work.

Keywords process calculus, higher order, bisimulation, encoding, full abstraction

1 Introduction

In this paper, we present some recent work on the relationship of the expressive power between HOCCS (Higher-Order Calculus of Communicating Systems) and FOPi (First-Order π-calculus), by working on the encoding from FOPi to Plain CHOCS, which was first studied by Thomsen[1]. However Thomsen simply put forward the idea and a rough framework of the relationship, and he did not obtain the full abstraction. Here we made more detailed the "bi-simulation" underlying the encoding.

We worked from the basic encoding strategy of Thomsen[1] and gained some important results on full abstraction. To get through the technical stuff, we made use of the index technique[2] that was abstracted from extra silent actions corresponding with operational semantics, from FOPi to Plain CHOCS, and formulated the encoding strategy under index technique. We first characterized the original encoding strategy, then transplanted the results to the indexed version. We need a theorem we call (indexed) wired factorization theorem, which implies that two (indexed) Plain CHOCS processes can be compared by concentrating on the corresponding (indexed) wired processes, which are processes that only send and receive (indexed) wires (a special kind of processes central in the encoding strategy), and isolating the part that may bring about difference between the two processes. We used the parenthesis to indicate the result held in both non-indexed and indexed situations. On (indexed) wired processes we defined (indexed) wired bisimulation and showed that (indexed) wired bisimilarity implied (indexed) context bisimilarity on (indexed) wired processes. With the help of the necessary intermediate results, we finally proved the full abstraction theorem of the indexed version of the encoding, with respect to early bisimilarity and indexed wired bisimilarity. Hence, the soundness of the indexed encoding was attained, with respect to early bisimilarity and indexed context bisimilarity.

Regular Paper

This work is supported by the National Basic Research 973 Program of China under Grant No. 2003CB317005, the National Natural Science Foundation of China under Grant Nos. 60473006 and 60573002, and the National Research Foundation for the Doctoral Program of Higher Education of China Research Fund under Grant No. 20010248033.

Taking an advantage of the bridge between the indexed encoding and the original encoding, we arrived at the result that the original encoding enjoyed full abstraction, with respect to early bisimilarity and wired bisimilarity, and soundness, with respect to early bisimilarity and context bisimilarity.

To draw a clear background, we give an overview of the related work.

Related Work

Thomsen is among the first to study higher-order CCS[1,3−5]. His contribution mainly consists of the following parts.

• (Applicative) higher-order bisimulation. The bisimulation treats the process being sent and the residual process separately, and thus is considered structural yet not intuitively reasonable. For example, two processes P and Q: $P \triangleq \overline{a}A.P'$, $Q \triangleq \overline{a}B.Q'$ are equivalent if A, B and P', Q' are equivalent respectively. Despite the disadvantage of it, the applicative higher-order bisimilarity is appealing in that it is of structure and congruence, whose proof is non-trivial though.

• Encoding Plain CHOCS with FOPi, and vice versa. However the full abstraction, with respect to the early bisimilarity (FOPi) and applicative higher-order bisimilarity (Plain CHOCS), is not proven. Later, some flaw was found by Sangiorgi[6]. This is probably because the bisimulations he chose are not suitable for a sound encoding. Nevertheless, Thomsen's idea is profound.

Sangiorgi has made remarkable contribution to higher-order π-calculus[6−8].

• Context bisimulation. Context bisimulation observes the communicated process and residual process at the same time, rather than comparing them separately. For instance, two processes P and Q: $P \triangleq \overline{a}A.P'$, $Q \triangleq \overline{a}B.Q'$ are equivalent if $E[A] \mid P'$ and $E[B] \mid Q'$ are equivalent for any higher-order process function $E[X]$. To get rid of the universal quantifier in the context bisimulation, normal bisimulation is designed, where the comparison of the output action is focused on triggers. The trigger method replaces a process being sent with a trigger and a repertory of the same process, and that trigger later can trigger a number of instances of the process in the repertory. Sangiorgi also showed that context bisimilarity coincided with barbed bisimilarity.

• Encoding between process-passing calculi and name-passing calculi. Sangiorgi gave an encoding of higher-order π-calculus with first-order π-calculus, and proved that the encoding is fully abstract, in the sense that barbed congruence[6,9] preserves through the encoding. The strategy substitutes a trigger with its subject name to denote its order, and necessary technical handling is made to get through the encoding. In Sangiorgi-Walker's book[10], an encoding from typed asynchronous localized π-calculus to typed asynchronous higher-order π-calculus is shown. The translation applies the abstraction mechanism in higher-order calculi. It is proven that the full abstraction property with respect to barbed congruence holds on the encoding.

Cao proved the coincidence between strong context bisimilarity and strong normal bisimilarity[2], using the index technique. Such technique assigns locations to a process and helps selectively filter out some extra silent actions in terms of the requirement of some bisimulation. It can offer some handy approach to coping with the twisted actions in analyzing bisimulations. Some other contributions on higher-order process calculi also have their importance, such as [11, 12], to name a few.

In this paper, we dwell on the encoding from first-order π-calculus and to Plain CHOCS. Our contribution can be summarized as follows.

• We further investigate Thomsen's encoding strategy[1]. Based on the strategy, we put forth the wired processes and present related concepts and results, including the wired factorization theorem, wired bisimilarity and its relationship with context bisimilarity.

• We apply index technique to the encoding strategy. The results in the first point are regenerated in the indexed version. We show the action correspondence lemmas, where the correspondence of operation semantics of the two calculi is made clear. The bridge between the indexed strategy and original (non-indexed) strategy is established.

• We prove the full abstraction theorem. The full abstraction in the indexed encoding is with respect to the early bisimilarity and indexed wired bisimilarity. A corollary is that the original encoding is fully abstract with respect to the early bisimilarity and wired bisimilarity. Since wired bisimilarity is contained by context bisimilarity, we guarantee the soundness of the original encoding, with respect to the early bisimilarity and the context bisimilarity.

The rest of this paper is organized as follows. In Section 2, we introduce the three calculi involved in the encoding, including their syntax, operational semantics and bisimulations. In Section 3, we present the original (non-indexed) version of the encoding from first-order π-calculus to Plain CHOCS, and go through the exploitation of the strategy. We define wired processes and wired bisimulation. We show the wired factorization theorem and the relationship between

wired bisimilarity and context bisimilarity. In Section 4, we reformulate the results for the original encoding under the indexed version. Combining these properties and the operational correspondence of the processes before and after the encoding, we eventually arrive at the full abstraction theorem and the important corollaries. We make some discussions on the work in this paper and consider some future work (Section 5) before the conclusion (Section 6).

2 Basic Definitions of the Calculi

We introduce three calculi: first-order π-calculus (FOPi), Plain CHOCS and indexed Plain CHOCS.

2.1 FOPi

FOPi[13] processes are defined below.

Definition 1 (FOPi Syntax).

$$P ::= 0 \mid x(y).P \mid \overline{x}y.P \mid \tau.P \mid (x)P \mid P \mid P' \mid !P.$$

We use capital letters $P, Q, \ldots$ to represent FOPi processes, and use $a, b, m, n \ldots$ for names, whose total set is $\mathcal{N}$. A FOPi process is built from the inactive process 0, three prefixes called input $(x(y).P)$, output $(\overline{x}y.P)$ and silent $(\tau.P)$, restriction $((x)P)$, parallel composition $(P \mid P')$, and replication $(!P)$. The precedence of operators is (from high to low): restriction, prefix, replication, parallel composition. When the object communicated in a prefix, such as y in $x(y).P$ and $\overline{x}y.P$, is not important, it is usually omitted. A name a is said to be bound (local or restricted) in $b(a).P$ and $(a)P$, otherwise it is free. We use $fn(\cdot), bn(\cdot), n(\cdot)$ for free names, bound names and names in a process, respectively. A name is fresh if it does not appear in any of the processes under consideration. We use $\{y/x\}$ to indicate substitution on first-order names. $P\{y/x\}$ can be structurally defined. Substitution on names are ranged over by σ, σ'. Notice we eliminate (non-deterministic) choice to down-scale the expressive power.

The operational semantics of FOPi is as below.

$$\text{(alp)} \quad \frac{P \xrightarrow{\lambda} Q}{P' \xrightarrow{\lambda} Q} \quad P \equiv_\alpha P'$$

$$\text{(prefix)} \quad \frac{}{x(y).P \xrightarrow{x(z)} P\{z/y\}} \qquad \frac{}{\overline{x}y.P \xrightarrow{\overline{x}y} P}$$

$$\frac{}{\tau.P \xrightarrow{\tau} P}$$

$$\text{(com)} \quad \frac{P \xrightarrow{\overline{x}y} P', \quad Q \xrightarrow{x(z)} Q'}{P \mid Q \xrightarrow{\tau} P' \mid Q'\{y/z\}}$$

$$\text{(close)} \quad \frac{P \xrightarrow{\overline{x}(y)} P', \quad Q \xrightarrow{x(z)} Q'}{P \mid Q \xrightarrow{\tau} (y)(P' \mid Q'\{y/z\})}$$

$$\text{(par)} \quad \frac{P \xrightarrow{\lambda} P'}{P \mid Q \xrightarrow{\lambda} P' \mid Q}, \; bn(\lambda) \cap fn(Q) = \varnothing$$

$$\text{(res)} \quad \frac{P \xrightarrow{\lambda} P'}{(x)P \xrightarrow{\lambda} \nu(x)P'} \; x \notin n(\lambda)$$

$$\text{(open)} \quad \frac{P \xrightarrow{\overline{x}y} P'}{(y)P \xrightarrow{\overline{x}(y)} P'} \; y \neq x$$

$$\text{(rep)} \quad \frac{P \xrightarrow{\lambda} P'}{!P \xrightarrow{\lambda} P' \mid !P}$$

The input $(x(y))$, output $(\overline{x}y)$, silent (τ) and bound output $(\overline{x}(y))$ actions are as usual. $\overline{x}(y).P$ abbreviates $(y)\overline{x}y.P$. The other rules have their usual meanings. We use static binding for restriction operator. Actions are denoted by α, β, λ. We assume parallel composition and restriction are individually commutative under any context, like the structural equivalence in [14].

The early bisimulation[13] will be used to show the reasonability of the encoding, while the late version is used in [1]. Weak transition $\overset{\alpha}{\Longrightarrow}$ denotes $\Longrightarrow \overset{\alpha}{\rightarrow} \Longrightarrow$, where $\Longrightarrow$ denotes a sequence of τ actions (possibly zero).

Definition 2 (Early Bisimulation). *A symmetric binary relation $\mathcal{R}$ on FOPi processes is an early bisimulation, if $P \mathcal{R} Q$ the following properties hold:*

1. *if $P \xrightarrow{x(y)} P'$, then for every name z, $Q \overset{x(y)}{\Longrightarrow} Q'$ for some Q' and $P'\{z/y\} \mathcal{R} Q'\{z/y\}$;*
2. *if $P \xrightarrow{\overline{x}y} P'$, then $Q \overset{\overline{x}y}{\Longrightarrow} Q'$ for some Q' and $P' \mathcal{R} Q'$;*
3. *if $P \xrightarrow{\overline{x}(y)} P'$, then $Q \overset{\overline{x}(y)}{\Longrightarrow} Q'$ for some Q' and $P' \mathcal{R} Q'$;*
4. *if $P \xrightarrow{\tau} P'$, then $Q \Longrightarrow Q'$ for some Q' and $P' \mathcal{R} Q'$.*

The early bisimilarity $\dot{\approx}$ is the largest early bisimulation.

Notice we can also define the late bisimulation, yet since the operation semantics is of late style in input, it is irrelevant.

2.2 Plain CHOCS

The higher-order CCS we study is Plain CHOCS[1].

Definition 3 (Plain CHOCS Syntax).

$$p ::= nil \mid x \mid a?x.p \mid a!p'.p \mid \tau.p \mid p \mid p' \mid (a)p \mid p + p' \mid !p \mid p[a \mapsto b].$$

Plain CHOCS here has no first-order fragment. $p, q, r, \ldots$ represent Plain CHOCS processes. A Plain

CHOCS process is built from the inactive process *nil*, (process) variable x, three prefixes called input ($a?x.p$), output ($a!p'.p$) and silent ($\tau.p$), parallel composition ($p \mid p'$), restriction ($(a)p$), renaming (or relabelling) ($p[a \mapsto b]$), (nondeterministic) choice ($p + p'$), and replication ($!p$). We usually use S to denote renaming $[a \mapsto b]$ in a function form, so $\mathrm{dom}(S)=\{a\}$ and $\mathrm{ran}(S)=\{b\}$. A choice is guarded[15] if its components connected by + all begins with some (observable) prefix, otherwise it is unguarded (or general). The precedence of operators is (from high to low): restriction, prefix, renaming, replication, parallel composition, choice. When the object in a prefix is not important, it is omitted, for example $a?.nil$. A name a is said to be bound (restricted or local) in $(a)p$, otherwise it is free. A process variable x is bound in $a(x).p$, otherwise it is free. We use $fn(\cdot), bn(\cdot), n(\cdot)$ for free names, bound names and names, respectively, and $fv(\cdot), bv(\cdot), v(\cdot)$ for free variables, bound variables and variables, respectively. A Plain CHOCS process is closed if it has no free process variables, otherwise it is open. We generally concentrate on closed processes. A name is fresh if it does not appear in any of the processes under consideration. We allow alpha-conversion. $\{r/x\}$ indicates substitution on high-order process variables, and $\{y/x\}$ stands for substitution on names. $p\{r/x\}$ and $p\{y/x\}$ can be structurally defined. Substitution on names and on process variables are ranged over by σ and Σ respectively. We use $E[x]$ to denote a process with at most free variable x in it, and $E[p]$ for $E[x]\{p/x\}$. Strong and weak transitions can be defined similarly to that for FOPi.

The operational semantics of Plain CHOCS is given below.

(pre) $\dfrac{}{a?x.p \xrightarrow{a?x} p} \quad \dfrac{}{a!p'.p \xrightarrow{a!_{\varnothing} p'} p} \quad \dfrac{}{\tau.p \xrightarrow{\tau} p}$

(com) $\dfrac{p \xrightarrow{a!_B p''} p', \quad q \xrightarrow{a?x} q'}{p \mid q \xrightarrow{\tau} (B)(p' \mid q'\{p''/x\})}$, $fn(q) \cap B = \varnothing$

(open) $\dfrac{p \xrightarrow{a!_B p''} p'}{(c)p \xrightarrow{a!_{B\cup\{c\}} p''} p'}$ $a \neq c, c \in fn(p'')$ and $c \notin B$

(non-struct) $\dfrac{p \xrightarrow{a!_B p''} p'}{p \xrightarrow{a!_{B'} p''} p'}$

$B \cap (fn(p') \cup fn(p'')) = B' \cap (fn(p') \cup fn(p''))$

(choice) $\dfrac{p \xrightarrow{\lambda} p'}{p + q \xrightarrow{\lambda} p'}$

(par) $\dfrac{p \xrightarrow{\lambda} p'}{p \mid q \xrightarrow{\lambda} p' \mid q}$, $bn(\lambda) \cap fn(q) = \varnothing$

(res) $\dfrac{p \xrightarrow{a?x} p'}{(c)p \xrightarrow{a?x} (c)p'}$, $a \neq c$

$\dfrac{p \xrightarrow{a!_B p''} p'}{(c)p \xrightarrow{a!_B p''} (c)p'}$ $a \neq c$ and $c \notin fn(p'')$

$\dfrac{p \xrightarrow{\tau} p'}{(c)p \xrightarrow{\tau} (c)p'}$

(ren) $\dfrac{p \xrightarrow{a?x} p'}{p[S] \xrightarrow{S(a)?x} p'[S]}$

$\dfrac{p \xrightarrow{a!_B p''} p'}{p[S] \xrightarrow{S(a)!_B p''} p'[S]}$ $B \cap (\mathrm{dom}(S) \cup \mathrm{ran}(S)) = \varnothing$

$\dfrac{p \xrightarrow{\tau} p'}{p[S] \xrightarrow{\tau} p'[S]}$

The input, output and silent actions are defined as usual. In output action, if $B = \varnothing$ it is sometimes omitted. $\alpha, \beta, \lambda, \ldots$ denote actions. By $(B)p$ ($B = \{b_1, b_2, \ldots, b_n\}$), we mean $(b_1)(b_2)\cdots(b_n)p$ or $(\tilde{b})p$. The rest rules are regular. We assume parallel composition, restriction and choice are individually commutatable.

Context bisimulation combines the comparison of the processes sent and remaining[8].

Definition 4 (Context Bisimulation). *Context bisimilarity $\approx_{Ct}$ is the largest symmetric binary relation $\mathcal{R}$ on Plain CHOCS processes, if $p\ \mathcal{R}\ q$ then:*

1. *if $p \xrightarrow{a?x} p'$, then $q \overset{a?x}{\Longrightarrow} q'$ for some q', and $p'\{r/x\}\ \mathcal{R}\ q'\{r/x\}$ for all closed process r;*
2. *if $p \xrightarrow{a!_B p'} p''$, q', q'' exists s.t. $q \overset{a!_{B'} q'}{\Longrightarrow} q''$, and for all $E[x]$ with $fn(E[x]) \cap (B \cup B') = \varnothing$, $(B)(p'' \mid E[p'])\ \mathcal{R}\ (B')(q'' \mid E[q'])$;*
3. *If $p \xrightarrow{\tau} p'$, then $q \Longrightarrow q'$ for some q', and $p'\ \mathcal{R}\ q'$.*

$\sim_{Ct}$ indicates strong context bisimilarity. Context bisimulation up-to $\approx_{Ct}$ can be defined in a traditional way, but note the transitions should be weak[16]. $\approx_{Ct}$ can be extended to open processes in the usual way.

We give a few properties of context bisimilarity. Related discussions can be found in [8, 12].

Lemma 5. *$\approx_{Ct}$ is an equivalence.*

We give some algebraic laws for $\approx_{Ct}$ without proofs.

Lemma 6. *The following hold:*

$p \mid nil \approx_{Ct} p$; $p_1 \mid p_2 \approx_{Ct} p_2 \mid p_1$;
$p_1 \mid (p_2 \mid p_3) \approx_{Ct} (p_1 \mid p_2) \mid p_3$;
$p + nil \approx_{Ct} p$; $p + nil \approx_{Ct} p$; $p_1 + p_2 \approx_{Ct} p_2 + p_1$;
$p_1 + (p_2 + p_3) \approx_{Ct} (p_1 + p_2) + p_3$;
$(a)(b)p \approx_{Ct} (b)(a)p$;
$(a)(p_1 \mid p_2) \approx_{Ct} ((a)p_1) \mid p_2$, *if* $a \notin fn(p_2)$;
$!p \approx_{Ct} p \mid !p$.

Lemma 7. *$\sim_{Ct}$ is a congruence. $\approx_{Ct}$ is a congruence except on unguarded choice operator.* $\approx^{c}_{Ct}$ is the

congruence induced from $\approx_{Ct}$.

2.3 Indexed Plain CHOCS

When analyzing the encoding, we make use of the index technique[2]. Indexed Plain CHOCS is Plain CHOCS enriched with indices. The idea of the technique is to assign indices to actions to indicate the identities of the communication components. Intuitively, an index can be seen as a kind of location of the process in the interaction scene, like in a distributed environment. Our general idea is to use it to filter out some technically introduced silent actions in the encoding strategy.

2.3.1 Indexed Processes

The syntax of indexed Plain CHOCS is as follows.

Definition 8 (Indexed Plain CHOCS Syntax).

$$h ::= nil \mid x \mid \{a?x\}_i.h \mid \{a!p'\}_i.h \mid \{\tau\}_{i,j}.h \mid h \mid h' \mid (a)h \mid h + h' \mid !h \mid h[a \mapsto b].$$

$i, j, k \in I$ denote indices and I is a countable set of indices. Indexed (Plain CHOCS) processes are denoted by h, k, l, m, n, p, q. $\{\tau\}_{i,j}, \{a?k\}_i, \{a!k\}_i, \{a!_B k\}_i$ are actions and denoted by α, β, λ. We do not differentiate them from non-indexed ones, for their meanings can be identified from context. In restriction, substitution and renaming, we revoke the indices. $Index(p)$ computes the indices in process p. Operation $\{p\}_i$ (respectively $\{p\}_{-i}$) assigns index i to (respectively remove index of) each prefix in process p. Free (bound) names and free (bound) variables, and other conventions can be defined like those in Plain CHOCS.

Suppose $S \subseteq I$, and in each pair $i_k \in S$ ($k = 1, \ldots, n$). Weak transitions are defined as follows.

$$\stackrel{\varepsilon,S}{\Longrightarrow} \triangleq \xrightarrow{\{\tau\}_{i_1,i_1}} \cdots \xrightarrow{\{\tau\}_{i_n,i_n}}; \quad \stackrel{\alpha,S}{\Longrightarrow} \triangleq \stackrel{\varepsilon,S}{\Longrightarrow} \xrightarrow{\alpha} \stackrel{\varepsilon,S}{\Longrightarrow}.$$

The intuition is that now not all silent actions are neglected, but selectively, with the help of S.

Below is the operational semantics of Indexed Plain CHOCS.

(pre) $\dfrac{}{\{a?x\}_i.p \xrightarrow{\{a?x\}_i} p} \quad \dfrac{}{\{a!p'\}_i.p \xrightarrow{\{a!\varnothing p'\}_i} p}$

$\dfrac{}{\{\tau\}_{i,j}.p \xrightarrow{\{\tau\}_{i,j}} p}$

(com) $\dfrac{p \xrightarrow{\{a!_B p''\}_i} p', \quad q \xrightarrow{\{a?x\}_j} q'}{p \mid q \xrightarrow{\{\tau\}_{i,j}} (B)(p' \mid q'\{p''/x\})}, \; fn(q) \cap B = \varnothing$

(open) $\dfrac{p \xrightarrow{\{a!_B p''\}_i} p'}{(c)p \xrightarrow{\{a!_{B\cup\{c\}} p''\}_i} p'}, \; a \neq c, c \in fn(p'')$ and $c \notin B$

(non-struct) $\dfrac{p \xrightarrow{\{a!_B p''\}_i} p'}{p \xrightarrow{\{a!_{B'} p''\}_i} p'}$

$B \cap (fn(p') \cup fn(p'')) = B' \cap (fn(p') \cup fn(p''))$

(choice) $\dfrac{p \xrightarrow{\lambda} p'}{p + q \xrightarrow{\lambda} p'}$

(par) $\dfrac{p \xrightarrow{\lambda} p'}{p \mid q \xrightarrow{\lambda} p' \mid q}, \quad bn(\lambda) \cap fn(q) = \varnothing$

(res) $\dfrac{p \xrightarrow{\{a?x\}_i} p'}{(c)p \xrightarrow{\{a?x\}_i} (c)p'}, \quad a \neq c$

$\dfrac{p \xrightarrow{\{a!_B p''\}_i} p'}{(c)p \xrightarrow{\{a!_B p''\}_i} (c)p'} \quad a \neq c$ and $c \notin fn(p'')$

$\dfrac{p \xrightarrow{\{\tau\}_{i,j}} p'}{(c)p \xrightarrow{\{\tau\}_{i,j}} (c)p'}$

(ren) $\dfrac{p \xrightarrow{\{a?x\}_i} p'}{p[S] \xrightarrow{\{S(a)?x\}_i} p'[S]}$

$\dfrac{p \xrightarrow{\{a!_B p''\}_i} p'}{p[S] \xrightarrow{\{S(a)!_B p''\}_i} p'[S]} \; B \cap (\mathrm{dom}(S) \cup \mathrm{ran}(S)) = \varnothing$

$\dfrac{p \xrightarrow{\{\tau\}_{i,j}} p'}{p[S] \xrightarrow{\{\tau\}_{i,j}} p'[S]}$

The semantics is similar to that of Plain CHOCS. The difference is that the actions are all indexed. Notice the *com* rule, where the silent action is indexed by the union of the indices of its participants, that is $\{\tau\}_{i,j}$. The other rules are not difficult to understand.

2.3.2 Indexed Bisimulation

Intuitively, an indexed bisimulation (bisimulations under index technique) family lies across a series of bisimulations, with the weak and strong bisimulations on two ends. An index set $S \subseteq I$ is used for selectively filtering out silent actions of interest. For the two ends, when S is I or $\varnothing$, we have the indexed context bisimulation below.

Definition 9 (Indexed Context Bisimulation). *Indexed context bisimulation with respect to $S \subseteq I$ is a symmetric binary relation $\mathcal{R}^S$ on indexed Plain CHOCS processes s.t. whenever $p\ \mathcal{R}^S\ q$ the following properties hold:*

1. *if $p \xrightarrow{\{a?x\}_i} p'$, then $q \stackrel{\{a?x\}_i,S}{\Longrightarrow} q'$ for some q', and $p'\{r/x\}\ \mathcal{R}^S\ q'\{r/x\}$ for every r;*
2. *if $p \xrightarrow{\{a!_B p'\}_i} p''$, then q', q'' exist s.t. $q \stackrel{\{a!_{B'} q'\}_i,S}{\Longrightarrow} q''$, and for every $E[x]$ with $fn(E[x]) \cap (B \cup B') = \varnothing$, $(B)(p'' \mid E[p'])\ \mathcal{R}^S\ (B')(q'' \mid E[q'])$;*
3. *if $p \xrightarrow{\{\tau\}_{i,i}} p'$, $i \in S$, then $q \stackrel{\varepsilon,S}{\Longrightarrow} q'$ for some q', and $p'\ \mathcal{R}^S\ q'$;*

4. *if* $p \xrightarrow{\{\tau\}_{i,i}} p'$, $i \notin S$, *then* $q \overset{\{\tau\}_{i,i},S}{\Longrightarrow} q'$ *for some* q', *and* $p' \mathcal{R}^S q'$.

$\approx^S_{Ct}$ *is the largest indexed context bisimulation.*

The main difference from the non-indexed version is that only those silent actions whose indices are in S can be neglected. For those whose indices are not in S, they are treated in a strong simulation way.

Lemma 10. $\approx^S_{Ct}$ *is an equivalence.* $\approx^S_{Ct}$ *is a congruence except on unguarded choice.*

The congruence induced from $\approx^S_{Ct}$ *is denoted by* $\simeq^S_{Ct}$. *Below we give several technical lemmas.*

Lemma 11. *Let* $\mathcal{R}^S (S \subseteq I)$ *be a symmetric binary relation on indexed Plain CHOCS processes. If* $p \mathcal{R}^S q$ *it implies*:

1. *if* $p \overset{\{a?x\}_i,S}{\Longrightarrow} p'$, *then* $q \overset{\{a?x\}_i,S}{\Longrightarrow} q'$ *for some* q', *and* $p'\{r/x\} \mathcal{R}^S q'\{r/x\}$ *for every* r;
2. *if* $p \overset{\{a!_B q'\}_i,S}{\Longrightarrow} p''$, q', q'' *exists s.t.* $q \overset{\{a!_{B'} q'\}_i,S}{\Longrightarrow} q''$, *and for every* $E[x]$ *with* $fn(E[x]) \cap (B \cup B') = \varnothing$, $(B)(p'' \mid E[p']) \mathcal{R}^S (B')(q'' \mid E[q'])$;
3. *if* $p \overset{\varepsilon,S}{\Longrightarrow} p'$, *then* $q \overset{\varepsilon,S}{\Longrightarrow} q'$ *for some* q', *and* $p' \mathcal{R}^S q'$;
4. *if* $p \overset{\{\tau\}_{i,i},S}{\Longrightarrow} p'$, $i \notin S$, *then* $q \overset{\{\tau\}_{i,i},S}{\Longrightarrow} q'$ *for some* q', *and* $p' \mathcal{R}^S q'$.

Then $\mathcal{R}^S$ *is an indexed context bisimulation.*

The following lemma relates indexed bisimulation to non-indexed bisimulation.

Lemma 12. *Suppose* p, q *are indexed Plain CHOCS processes, and* $S = I$. *Then* $p \approx^S_{Ct} q$ *iff* $\{p\}_{-i} \approx_{Ct} \{q\}_{-i}$.

The following lemma is a simple corollary.

Lemma 13. *Suppose* p, q *are indexed Plain CHOCS processes, and* $S = Index(p) \cup Index(q)$. *Then* $p \approx^S_{Ct} q$ *iff* $\{p\}_{-i} \approx_{Ct} \{q\}_{-i}$.

Remark. We include choice operator in Plain CHOCS because we need it in simulating names of FOPi, as will be seen in the encoding to be given. However, their usage is limited and merely guarded choice is involved (in wires), avoiding its notorious damage in congruence property. This reflects in a sense that the choice operator is powerful but sometimes too strong, so some limit should be imposed. Similar discussion on the choice operator can be found in [6]. Notice we generally focus on closed processes in the calculi mentioned above. The related definitions and results, however, can be extended to open processes as well.

3 Encoding Strategy

The original encoding from FOPi to Plain CHOCS is due to Thomsen[1]. However, he did not study further, and only the action correspondence is considered. We inherit this basic encoding strategy.

The core idea is to use the apparatus called *wire*[1]. It is a special process serving the function of a FOPi name, which can be a *channel*, a *variable* and a *local* one of the above. A *wire*, written as *a-chan*, with respect to a FOPi name a is defined as:

$$a\text{-}chan \triangleq i?.a?x.c!x.nil + o?.c?x.a!x.nil.$$

"*chan*" stands for channel, and *a-chan* is meant to simulate name a. Notice a *wire* is (strictly) a *higher-order* process. $i?.a?x.c!x.nil$ and $o?.c?x.a!x.nil$ simulate an input action and output action on name a respectively, where names i and o are used for signaling the *wire* whether it is used for inputting or outputting, and c is an auxiliary name for passing the information to be communicated. Generally we can assume that names c, i, o shall not appear free in positions other than in *wires*.

Definition 14 (Encoding from FOPi to Plain CHOCS). *The encoding strategy has two parts.*

$[\![\ ]\!]_1$:

$$[\![0]\!]_1 \triangleq nil,$$
$$[\![x(y).P]\!]_1 \triangleq (o')(i')(c')(x[c \mapsto c'][i \mapsto i'][o \mapsto o'] \mid i'!.c'?y.[\![P]\!]_1),$$
$$[\![\overline{x}(y).P]\!]_1 \triangleq (o')(i')(c')(x[c \mapsto c'][i \mapsto i'][o \mapsto o'] \mid o'!.c'!y.[\![P]\!]_1),$$
$$[\![\tau.P]\!]_1 \triangleq \tau.[\![P]\!]_1,$$
$$[\![P \mid P']\!]_1 \triangleq [\![P]\!]_1 \mid [\![P']\!]_1,$$
$$[\![!P]\!]_1 \triangleq ![\![P]\!]_1,$$
$$[\![(x)P]\!]_1 \triangleq (a)([\![P]\!]_1\{(a\text{-}chan)/x\}),\ a \notin fn(P);$$

$[\![\ ]\!]_2$:

$$[\![P]\!]_2 \triangleq (\cdots([\![P]\!]_1\{(a_1\text{-}chan)/x_1\})\cdots)\{(a_n\text{-}chan)/x_n\}.$$

A free name in FOPi is first mapped to a process variable, and then it is substituted by a *wire*. It does no harm to set that there is a *one-to-one* mapping between FOPi names and Plain CHOCS *wires*. That is, it is from $fn(P) = \{a_1, a_2, \ldots, a_n\}$ to $fv([\![P]\!]_1) = \{x_1, x_2, \ldots, x_n\}$ and then to $\{a_1\text{-}chan, a_2\text{-}chan\}, \ldots, a_n\text{-}chan\}$. The mapping can be identity. Thus in the restriction case, the side condition can also be written as $a \notin fn([\![P]\!]_1)$. For a local name of input in FOPi, it is simply mapped to a process variable, which is because we do not know to which *wire* it will be instantiated. It is generally deemed that renaming is a much complicated operator, for its behavior, especially with recursion, can lead to rather involved computations. Examining the encoding strategy above, one can find

that the renaming is harnessed so that it localizes the input (or output) preparation on *wires*. Such intricacy would not appear in the strategy.

Below we give an instance of the encoding. Suppose we have the following two FOPi processes.

$$P \triangleq x(y).y(z).P'$$
$$Q \triangleq (y)\overline{x}y.\overline{y}z.0.$$

The system $P \mid Q$ can be considered as a basic secure transmission, in which Q first transmits a secret channel y to P, and then sends some information z on that channel, and P can do something with it. We translate the system into Plain CHOCS processes. σ denotes the substitution that maps free names in $P \mid Q$ to *wires*, where x, y, z are mapped to a_1, a_2, a_3 (and the corresponding *wires*), respectively. $\approx$ denotes some weak bisimilarity such as context bisimilarity.

$$\begin{aligned}
&[\![P \mid Q]\!]_2 = (o')(i')(c')(a_1\text{-}chan)[c \mapsto c'][i \mapsto i'][o \mapsto o'] \mid \\
&i'!.c'?y.[\![y(z).P']\!]_1)\sigma \mid \\
&(a_2)((o')(i')(c')((a_1\text{-}chan)[c \mapsto c'][i \mapsto i'][o \mapsto o'] \mid \\
&o'!.c'!(a_2\text{-}chan).[\![\overline{y}z.0]\!]_1)\{(a_2\text{-}chan)/y\})\sigma \\
&\xrightarrow{\tau}\xrightarrow{\tau}\xrightarrow{\tau}\xrightarrow{\tau}\xrightarrow{\tau}\ (a_2)(([\![y(z).P']\!]_1\{(a_2\text{-}chan)/y\} \mid \\
&[\![\overline{y}z.0]\!]_1)\{(a_2\text{-}chan)/y\})\sigma = \\
&(a_2)(((o')(i')(c')((a_2\text{-}chan)[c \mapsto c'][i \mapsto i'][o \mapsto o'] \mid \\
&i'!.c'?z.[\![P']\!]_1) \mid \\
&(o')(i')(c')((a_2\text{-}chan)[c \mapsto c'][i \mapsto i'][o \mapsto o'] \mid \\
&o'!.c'!(a_3\text{-}chan).[\![0]\!]_1))\{(a_2\text{-}chan)/y\})\sigma \\
&\xrightarrow{\tau}\xrightarrow{\tau}\xrightarrow{\tau}\xrightarrow{\tau}\xrightarrow{\tau}\ (a_2)(([\![P']\!]_1\{(a_3\text{-}chan)/z\} \mid \\
&[\![0]\!]_1)\{(a_2\text{-}chan)/y\})\sigma = \\
&(a_2)(([\![P']\!]_1\{(a_3\text{-}chan)/z\} \mid nil)\{(a_2\text{-}chan)/y\})\sigma \\
&\approx (a_2)([\![P']\!]_1\{(a_3\text{-}chan)/z\}\{(a_2\text{-}chan)/y\})\sigma.
\end{aligned}$$

It is not hard to identify the operational correspondence of either side of the encoding. By comparing the encoding and its transition with the transitions in FOPi: $P \mid Q \xrightarrow{\tau}(y)(y(z).P' \mid \overline{y}z.0) \xrightarrow{\tau}(y)P'$, one can find that the first five straight internal moves correspond to the first τ action above of $P \mid Q$, and the second five in a row correspond to the second τ action.

Below is a lemma on the property of the encoding. The proof is quite direct from the encoding strategy.

Lemma 15. *The following two equations hold*:

$$[\![a(x).P]\!]_2 \approx_{Ct} \tau.a?x.\tau.[\![P]\!]_2;$$
$$[\![\overline{a}b.P]\!]_2 \approx_{Ct} \tau.\tau.a!(b\text{-}chan).[\![P]\!]_2.$$

The next lemma states the correspondence in actions between a FOPi process and the Plain CHOCS process encoding it. The proof is straightforward.

Lemma 16. *The following action correspondence hold*:

1 *if* $P \xrightarrow{a(x)} P'$, *then* $[\![P]\!]_2 \xrightarrow{\tau}\xrightarrow{a?x}\xrightarrow{\tau}[\![P']\!]_2$;

2 *if* $P \xrightarrow{\overline{a}b} P'$, *then* $[\![P]\!]_2 \xrightarrow{\tau}\xrightarrow{\tau}\xrightarrow{a!_{\varnothing}(b\text{-}chan)}[\![P']\!]_2$;

3 *if* $P \xrightarrow{\overline{a}(b)} P'$, *then* $[\![P]\!]_2 \xrightarrow{\tau}\xrightarrow{\tau}\xrightarrow{a!_{\{b\}}(b\text{-}chan)}[\![P']\!]_2$;

4 *if* $P \xrightarrow{\tau} P'$, *then* $[\![P]\!]_2 \xrightarrow{\tau}[\![P']\!]_2$ *or* $[\![P]\!]_2 \xrightarrow{\tau}\xrightarrow{\tau}\xrightarrow{\tau}\xrightarrow{\tau}\xrightarrow{\tau}[\![P']\!]_2$.

3.1 Characterizing *Wires*

In light of triggers[8], we work out a series of properties on behavior of *wires* in this subsection, abstraction theorem.

3.1.1 Wired Processes

Wired processes are processes that only send or receive *wires* during communication, and have no unguarded choice operator. $\mathcal{WP}r$ denotes the set of wired processes.

Definition 17 (Wired Processes). $p\{\diamond_a r\}$ *abbreviates* $(c)(o)(i)(a)(p \mid !(i!.a!x.c?x.r + o!.c!x.a?x.r))$, *where the part* $(i!.a!x.c?x.r+o!.c!x.a?x.r)$, *is referred to as the choice part, where the guarded choice connects* $i!.a!x.c?x.r$ *and* $o!.c!x.a?x.r$. *Wired processes are Plain CHOCS processes that only communicate wires, with only guarded choice operators appearing in wires or the choice part in* $p\{\diamond_a r\}$.

Below is an obvious lemma from the encoding strategy (Definition 14).

Lemma 18. *Suppose* P *is a FOPi process, then* $[\![P]\!]_2$ *is a wired process.*

3.1.2 Wired Bisimulation

We define a bisimulation called wired bisimulation on wired processes.

Definition 19 (Wired Bisimulation). *A symmetric binary relation* $\mathcal{R}$ *on wired processes is a wired bisimulation, if* $p\mathcal{R}q$ *implies*:

1. *if* $p \xrightarrow{a?x} p'$, *then* $q \overset{a?x}{\Longrightarrow} q'$ *for some* q', *and* $p'\{b\text{-}chan/x\}\ \mathcal{R}\ q'\{b\text{-}chan/x\}$ *for a wire* $b\text{-}chan$, *where* b *is a fresh name*;
2. $p \xrightarrow{a!_B(b\text{-}chan)} p''$, *where* b *is a name, then* $q \overset{a!_B(b\text{-}chan)}{\Longrightarrow} q''$ *for some* q'', *and* $p''\ \mathcal{R}\ q''$;
3. $p \xrightarrow{\tau} p'$, *then* $q \Longrightarrow q'$ *for some* q', *and* $p'\ \mathcal{R}\ q'$.

Two processes p *and* q *are wired bisimilar, written as* $p \approx_{Wr} q$, *if there is a wired bisimulation* $\mathcal{R}$, *s.t.,* $p\mathcal{R}q$.

The definition is confined to *wired* processes, and the bisimulation criteria are designed with regard to the behavior on communicating wires. As we will see, wired bisimilarity implies context bisimilarity on wired processes.

3.1.3 Wired Factorization Theorem

In this subsection we derive the theorem called wired factorization theorem.

Actually $\{\diamond_a r\}$ shares a similar idea with the implicit substitution, and *wires* resemble triggers[8]. The idea of triggers is to factorize out a sub-process of a given process and fill in a trigger $(\overline{m}.nil)$. To keep the new process equivalent with the old one, some special gadget, i.e., implicit substitution ($\{z := B\}$), shall be designed and put in parallel with the new process ($A\{z := B\}$ abbreviates $(z)(A \mid !z.B)$). The obtained process has an embedded local environment that is formed by the trigger and implicit substitution, where the implicit substitution serves as a resource warehouse of the replaced sub-process and the trigger acts as a pointer to the resources. Thus the trigger can activate a copy of the resource in need. Here, the trigger's role is played by the *wire*, whereas implicit substitution by $\{\diamond_a r\}$.

Several properties are given below. The first is on the substitution of $\{\diamond_a\}$.

Lemma 20 (Substitution Lemma on $\{\diamond_a\}$). *Let p, q, r be Plain CHOCS processes without unguarded choice operator. Then the following holds*:

$$p\{q/x\}\{\diamond_a r\} \approx_{Ct} p\{\diamond_a r\}\{q\{\diamond_a r\}/x\}.$$

Proof. This is an important lemma serving as the basis for Lemma 21. The proof of it, however, is not so trivial and needs some long step-by-step derivation. The framework is quite similar to the Lemma (4.1) (on implicit substitution) in [8], whose proof is given in Appendix B of [8]. We do not give the detailed proof here, and the reader can refer to [8] for the basic idea. □

Below is a lemma on the distribution of $\{\diamond_a\}$.

Lemma 21 (Distributivity on $\{\diamond_a\}$). *Let p, q, q', r, r' be Plain CHOCS processes without unguarded choice operator, and suppose the local names related to $\{\diamond_a\}$ occur only in subjects. Then it holds that*:

1. $nil\{\diamond_a r\} \approx_{Ct} nil$;
2. $(i?.a?x.c!x.p)\{\diamond_a r\} \approx_{Ct} (\tau.\tau.\tau.(p \mid r))\{\diamond_a r\}$, *$a$ appears in $\{\diamond_a r\}$*;
3. $(o?.c?x.a!p'.p)\{\diamond_a r\} \approx_{Ct} (\tau.\tau.\tau.(p \mid r\{p'/x\}))\{\diamond_a r\}$, *$a$ appears in $\{\diamond_a r\}$ and r*;
4. $(\tau.p)\{\diamond_a r\} \approx_{Ct} \tau.p\{\diamond_a r\}$;
5. $(p \mid p')\{\diamond_a r\} \approx_{Ct} p\{\diamond_a r\} \mid p'\{\diamond_a r\}$;
6. $((b)p)\{\diamond_a r\} \approx_{Ct} (b)(p\{\diamond_a r\})$, *if b does not appear in $fn(r)$ and $\{\diamond_a r\}$ related names*;
7. $(p[b \mapsto c])\{\diamond_a r\} \approx_{Ct} (p\{\diamond_a r\})[b \mapsto c]$, *if b, c do not appear in $fn(r)$ and $\{\diamond_a r\}$ related names*;
8. $(!p)\{\diamond_a r\} \approx_{Ct} !(p\{\diamond_a r\})$.

Proof. Most of the properties are straightforward, with the help of Lemma 20. □

The following lemma is immediately provable.

Lemma 22. *It holds that $p \approx_{Ct} \tau.p$, for all Plain CHOCS processes without unguarded choice operator.*

Theorem 23 (Wired Factorization Theorem).

$$p\{r/x\} \approx_{Ct} (p\{a\text{-}chan/x\})\{\diamond_a r\}$$

for all processes p without unguarded choice operator and process r.

Proof. To prove this theorem, we first show the claim below. It says that instead of passing a process, we can replace it with a *wire*, the effect of which is remedied by adding some extra silent action before the replaced process. □

Claim. *For all processes p without unguarded choice operator and process r, it holds that*

$$p\{\tau.r/x\} \approx_{Ct} (p\{a\text{-}chan/x\})\{\diamond_a r\}.$$

Proof. The proof of the claim is by induction on the structure of p. The basic cases are *nil* and $p = y$, which are straightforward. The rest of the proof is a routine induction argument. We take the parallel composition as a typical one and omit the rest.

$$\begin{aligned}
&(p \mid p')\{\tau.r/x\} = (p\{\tau.r/x\}) \mid (p'\{\tau.r/x\}) \\
&\qquad \text{(by substitution definition)} \\
&\approx_{Ct} (p\{a\text{-}chan/x\})\{\diamond_a r\} \mid (p'\{a\text{-}chan/x\})\{\diamond_a r\} \\
&\qquad \text{(by induction hypothesis)} \\
&\approx_{Ct} ((p\{a\text{-}chan/x\}) \mid (p'\{a\text{-}chan/x\}))\{\diamond_a r\} \\
&\qquad \text{(by Lemma 21)} \\
&= (p \mid p')\{a\text{-}chan/x\}\{\diamond_a r\} \\
&\qquad \text{(by substitution definition).}
\end{aligned}$$

Now this theorem can be proved by the following deduction.

$$\begin{aligned}
(p\{a\text{-}chan/x\})\{\diamond_a r\} &\approx_{Ct} p\{\tau.r/x\} \quad \text{(by the Claim)} \\
&\approx_{Ct} p\{r/x\} \\
\text{(by Lemma 22and congruence of } &\approx_{Ct}). \qquad \square
\end{aligned}$$

3.1.4 Relationship Between $\approx_{Wr}$ and $\approx_{Ct}$

We first give some properties of $\approx_{Wr}$.

Lemma 24. *Let p, q, r be wired processes. If $p \approx_{Wr} q$ then*:

1. $(a)p \approx_{Wr} (a)q$;
2. $p \mid r \approx_{Wr} q \mid r$;
3. *if $p \simeq_{W_r} q$, then $p[a \mapsto b] \approx_{Wr} q[a \mapsto b]$.*

Proof. We only examine the second one. To prove it, we define the following relation:

$$\mathcal{R} \triangleq \left\{ ((B)(p_1 \mid r), (B)(p_2 \mid r)) \;\middle|\; \begin{array}{l} p_1, p_2, r \text{ are} \\ \text{wired processes,} \\ B \text{ is a set of names,} \\ \text{and } p_1 \approx_{Wr} p_2. \end{array} \right\}.$$

We define $\mathcal{R}$ in this way because there are local names in wire-related local environment. Below we show that $\mathcal{R}$ is a wired bisimulation.

Suppose $((B_1)(p_1 \mid r), (B_1)(p_2 \mid r)) \in \mathcal{R}$, and $(B_1)(p_1 \mid r) \xrightarrow{\alpha} q_1$. The proof can be expanded on the analysis of the action α. Most of them are routine and the difficult one is when $\alpha = \tau$, where p_1 makes an output action. We know that

$$p_1 \xrightarrow{a!_{B_2}(m\text{-}chan)} (B_2)p_1', \quad r \xrightarrow{a?x} r',$$

for some fresh m, and

$$q_1 \equiv (B_1 \cup B_2)(p_1' \mid r'\{m\text{-}chan/x\}).$$

Now that $p_1 \approx_{Wr} p_2$, we have

$$p_2 \xrightarrow{a!_{B_2}(m\text{-}chan)} (B_2)p_2', \quad \text{but } r \xrightarrow{a?x} r',$$

so we have

$$(B_1)(p_2 \mid r) \xrightarrow{\tau} (B_1 \cup B_2)(p_2' \mid r'\{m\text{-}chan/x\}) \triangleq q_2.$$

Hence we have $q_1 \mathcal{R} q_2$. □

Lemma 25. *Let p, q be wired processes. If $p \approx_{Wr} q$, then $p \approx_{Ct} q$.*

Proof. Let $\mathcal{R} \triangleq \{(p, q) \mid p, q \in \mathcal{WP}r; p \approx_{Wr} q\}$. It is sufficient to show that $\mathcal{R}$ is a context bisimulation up-to $\approx_{Ct}$.

Suppose $(p, q) \in \mathcal{R}$ and $p \stackrel{\alpha}{\Rightarrow} p'$. There are several cases to analyze, and the most non-trivial case is when $\alpha \equiv a?x$. We know that $p \stackrel{a?x}{\Longrightarrow} p'$, since $p \approx_{Wr} q$, there exists q' such that $q \stackrel{a?x}{\Longrightarrow} q'$, and for *m-chan* with m fresh in it,

$$p'\{m\text{-}chan/x\} \approx_{Wr} q'\{m\text{-}chan/x\}.$$

To match the action of p, we have to prove that for every wired process r, we have

$$p'\{r/x\} \approx_{Ct} \mathcal{R} \approx_{Ct} q'\{r/x\}.$$

We make the following reasoning.

$$\begin{aligned} p'\{r/x\} &\approx_{Ct} p'\{m\text{-}chan/x\}\{\diamond_m\ r\} \text{ (by Theorem 23)} \\ &\triangleq p''. \end{aligned}$$

Similarly, we have:

$$\begin{aligned} q'\{r/x\} &\approx_{Ct} q'\{m\text{-}chan/x\}\{\diamond_m\ r\} \quad \text{(by Theorem 23)} \\ &\triangleq q''. \end{aligned}$$

By Lemma 24, we know that $\approx_{Wr}$ is closed under restriction and parallel composition, so we get from $p'\{m\text{-}chan/x\} \approx_{Wr} q'\{m\text{-}chan/x\}$ that $p'' \approx_{Wr} q''$, which completes the simulating step. □

Lemma 26. *Let p, q be wired processes. If $p \approx_{Ct} q$, then it does not necessarily hold that $p \approx_{Wr} q$.*

Proof. To see the difference, we pretend that we tried to prove the inclusion. Define:

$$\mathcal{R} \triangleq \{(p_1, p_2) \mid p_1, p_2 \in \mathcal{WP}r; p_1 \approx_{Ct} p_2\}.$$

Suppose $(p_1, p_2) \in \mathcal{R}$, and $p_1 \xrightarrow{\alpha} p_1'$. There are several cases to analyze. We consider the case when $\alpha \equiv a!(m\text{-}chan)$. We know that $p_1 \xrightarrow{a!_B(m\text{-}chan)} p_1'$, since $p_1 \approx_{Ct} p_2$, and we have $p_2 \xrightarrow{a!_B(m'\text{-}chan)} p_2'$, for some m', p_2', and for every $E[x]$ s.t. $fn(E[x]) \cap B = \varnothing$, it holds that

$$(B)(p_1' \mid E[m\text{-}chan]) \approx_{Ct} (B)(q_2' \mid E[m'\text{-}chan]),$$

where typically $B \subseteq \{m, m', i, o, c\}$. When $m \in B$, we can assume $m = m'$.

To fulfill the simulating step, we have to prove that $p_1' \approx_{Ct} p_2'$. Since E is arbitrary, one can flexibly choose the form of E to try to make the simulation closed. Unfortunately, this does not work. It is because without the restriction outside, p_1' and p_2' may contain rather different structures (hence behavior) concerning m. Typically, one can use the counterexample that p_1' contains an occurrence of m, but p_2' has no such occurrence, such as the one given in the remark following. Therefore the trial stops here. □

Remark. The major cause for the non-inclusion is that in a wired bisimulation there is no demand on the local names' effect on the residuals after a higher-order output action. This is too strong a requirement. We give a counterexample. It actually resembles the example in the introduction section of [8], where the

difference between applicative higher-order bisimilarity and context bisimilarity is exemplified. Here the fact that $\approx_{Wr} \subsetneq \approx_{Ct}$ has a similar argument. Suppose m is fresh. Let P, Q be defined as follows:

$$P \triangleq (m)a!(m\text{-}chan).(i? + o?.c!x),$$
$$Q \triangleq (m)a!(m\text{-}chan).(i?.m!x.c?x + o?.c!x.m?x).$$

Then P and Q are context bisimilar, but not wired bisimilar. Notice m is unknown to the receiving environment, thus internal communication on it cannot be detected. However, in wired bisimulation, the continual processes of P and Q can be distinguished.

By tracing back, we stress that the design of wired bisimulation is dominated by the early bisimulation in FOPi, where first-order bound output makes a similar requirement in simulation. However, the full abstraction may be undermined by Lemma 26 in one direction. We will make more discussion later.

By the two lemmas above, Lemma 25 and Lemma 26, we attain the following theorem.

Theorem 27. *$\approx_{Wr}$ is contained by $\approx_{Ct}$ on wired processes and the inclusion is strict.*

4 Encoding Strategy Under Indexed Technique

In this section we apply index technique to the encoding in Section 3.

An indexed wire *a-ichan* with respect to name a and index n is defined as:

$$a\text{-}ichan \triangleq \{i?\}_n.\{a?x\}_m.\{c!x\}_n.nil + \{o?\}_n.\{c?x\}_n.\{a!x\}_m.nil,$$

where $m, n \in I$.

We assign index n to actions on c, i, o to mark that silent actions indexed by (n, n) are from these auxiliary names. m is an index assigned to each prefix appearing in the encoded process. We only need demand $m \neq n$ to get over all the technical details in investigating the encoding. It helps differentiate the communications occurring in the encoded process from those on the names c, i, o.

4.1 Indexed Version of the Encoding

The indexed version of the encoding is given below.

Definition 28. *The definition still has two parts.*

$[\![\ ]\!]_1^n$:

$[\![0]\!]_1^n \triangleq nil$,

$[\![x(y).P]\!]_1^n \triangleq (o')(i')(c')(x[c \mapsto c'][i \mapsto i'][o \mapsto o'] \mid \{i'!\}_n.\{c'?y\}_n.[\![P]\!]_1^n)$,

$[\![\overline{x}y.P]\!]_1^n \triangleq (o')(i')(c')(x[c \mapsto c'][i \mapsto i'][o \mapsto o'] \mid \{o'!\}_n.\{c'!y\}_n.[\![P]\!]_1^n)$,

$[\![\tau.P]\!]_1^n \triangleq \{\tau\}_{m,m}.[\![P]\!]_1^n$,

$[\![P \mid P']\!]_1^n \triangleq [\![P]\!]_1^n \mid [\![P']\!]_1^n$,

$[\![!P]\!]_1^n \triangleq ![\![P]\!]_1^n$,

$[\![(x)P]\!]_1^n \triangleq (a)([\![P]\!]_1^n\{(a\text{-}ichan)/x\})$, $a \notin fn(P)$;

$[\![\ ]\!]_2^n$:

$[\![P]\!]_2^n \triangleq (\cdots([\![P]\!]_1^n\{(a_1\text{-}ichan)/x_1\})\cdots)\{(a_n\text{-}ichan)/x_n\}$.

Remark. The conventions are similar to those for the original encoding. The only difference is that this time *wires* are modified by indices. As said, the index m is assigned to actions in FOPi processes, to distinguish from those brought about by the encoding strategy. The following lemma states an obvious property.

Lemma 29. *Let P be a FOPi process. Then $Index([\![P]\!]_2^n) \subseteq \{m, n\}$.*

We re-examine the example in the original encoding. The process after the indexed encoding and its transitions are described as follows.

$$[\![P \mid Q]\!]_2^n =$$
$$(o')(i')(c')((a_1\text{-}ichan)[c \mapsto c'][i \mapsto i'][o \mapsto o'] \mid \{i'!\}_n.\{c'?y\}_n.[\![y(z).P']\!]_1^n)\sigma \mid (a_2)((o')(i')(c')((a_1\text{-}ichan)[c \mapsto c'][i \mapsto i'][o \mapsto o'] \mid \{o'!\}_n.\{c'!(a_2\text{-}ichan)\}_n.[\![\overline{y}z.0]\!]_1^n)\{(a_2\text{-}ichan)/y\})\sigma$$
$$\xrightarrow{\{\tau\}_{n,n}}\xrightarrow{\{\tau\}_{n,n}}\xrightarrow{\{\tau\}_{n,n}}\xrightarrow{\{\tau\}_{m,m}}\xrightarrow{\{\tau\}_{n,n}}$$
$$(a_2)(([\![y(z).P']\!]_1^n\{(a_2\text{-}ichan)/y\} \mid [\![\overline{y}z.0]\!]_1^n)\{(a_2\text{-}ichan)/y\})\sigma = (a_2)(((o')(i')(c')((a_2\text{-}ichan)[c \mapsto c'][i \mapsto i'][o \mapsto o'] \mid \{i'!\}_n.\{c'?z\}_n.[\![P']\!]_1^n) \mid (o')(i')(c')((a_2\text{-}ichan)[c \mapsto c'][i \mapsto i'][o \mapsto o'] \mid \{o'!\}_n.\{c'!(a_3\text{-}ichan)\}_n.[\![0]\!]_1^n))\{(a_2\text{-}ichan)/y\})\sigma$$
$$\xrightarrow{\{\tau\}_{n,n}}\xrightarrow{\{\tau\}_{n,n}}\xrightarrow{\{\tau\}_{n,n}}\xrightarrow{\{\tau\}_{m,m}}\xrightarrow{\{\tau\}_{n,n}}$$
$$(a_2)(([\![P']\!]_1^n\{(a_3\text{-}ichan)/z\} \mid [\![0]\!]_1^n)\{(a_2\text{-}ichan)/y\})\sigma = (a_2)(([\![P']\!]_1^n\{(a_3\text{-}ichan)/z\} \mid nil)\{(a_2\text{-}ichan)/y\})\sigma$$
$$\approx (a_2)([\![P']\!]_1^n\{(a_3\text{-}ichan)/z\}\{(a_2\text{-}ichan)/y\})\sigma.$$

We can see that the internal moves of the encoding process are classified into two groups: those indexed by (m, m) comes from the encoded process, and those indexed by (n, n) are from the communication on auxiliary ports (c, i, o) during the encoding. With this clarification, one can more detailedly observe the essence of the encoding.

The following lemma is easily derivable.

Lemma 30. *The following two equations hold for* $S = \{n\}$.

$$[\![a(x).P]\!]_2^n \approx_{Ct}^S \{a?x\}_m.[\![P]\!]_2^n,$$
$$[\![\overline{a}b.P]\!]_2^n \approx_{Ct}^S \{a!(b\text{-}ichan)\}_m.[\![P]\!]_2^n.$$

4.2 Characterizing Indexed Wires

In this subsection, we exploit indexed wires. The results in this section have similar proofs as the counterparts of the original encoding in Section 3. The point is that under index technique, all the processes are treated in a uniform way that silent actions are selectively neglected according to the index set S in an indexed bisimulation. We generally do not give the details of the proofs but focus on the framework.

4.2.1 Indexed Wired Processes

Indexed wired processes are defined by the following definition.

Definition 31 (Indexed Wired Processes). $p\{\diamond_a^i\, r\}$ *abbreviates*

$$(c)(o)(i)(a)(p \mid !(\{i!\}_n.\{a!x\}_m.\{c?x\}_n.r + \{o!\}_n.\{c!x\}_n.\{a?x\}_m.r)),$$

where the part

$$(\{i!\}_n.\{a!x\}_m.\{c?x\}_n.r + \{o!\}_n.\{c!x\}_n.\{a?x\}_m.r),$$

is referred to as the choice part, where the guarded choice connects $\{i!\}_n.\{a!x\}_m.\{c?x\}_n.r$ *and* $\{o!\}_n.\{c!x\}_n.\{a?x\}_m.r$. *The superscript* i *on the diamond indicates "index".*

Indexed wired processes are indexed Plain CHOCS processes that only communicate indexed wires, with no unguarded choice operator and only guarded choice operators appearing in indexed wires or the choice part in $p\{\diamond_a^i\, r\}$. $\mathcal{IWP}r$ denotes the set of indexed wired processes.

Lemma 32. *Let* P *be a FOPi process, then* $[\![P]\!]_2^n$ *is an indexed wired process.*

4.2.2 Indexed Wired Bisimulation

Below we define indexed wired bisimulation.

Definition 33 (Indexed Wired Bisimulation). *A symmetric binary relation* $\mathcal{R}^S$ *with respect to index set* S *on indexed wired processes is an indexed wired bisimulation, if* $p\,\mathcal{R}^S\,q$ *implies*:

1. *if* $p \xrightarrow{\{a?x\}_i} p'$, *then* $q \overset{\{a?x\}_i,S}{\Longrightarrow} q'$ *for some* q', *and* $p'\{b\text{-}ichan/x\}\ \mathcal{R}^S\ q'\{b\text{-}ichan/x\}$ *for an indexed wire* b-$ichan$, *where* b *is fresh*;
2. *if* $p \xrightarrow{\{a!_B(b\text{-}ichan)\}_i} p''$, *where* b *is a name, then* $q \overset{\{a!_B(b\text{-}ichan)\}_i,S}{\Longrightarrow} q''$ *for some* q'', *and* $p''\ \mathcal{R}^S\ q''$;
3. *if* $p \xrightarrow{\{\tau\}_{i,i}} p'$, $i \in S$, *then* $q \overset{\varepsilon,S}{\Longrightarrow} q'$ *for some* q', *and* $p'\ \mathcal{R}^S\ q'$;
4. *if* $p \xrightarrow{\{\tau\}_{i,i}} p'$, $i \notin S$, *then* $q \overset{\{\tau\}_{i,i},S}{\Longrightarrow} q'$ *for some* q', *and* $p'\ \mathcal{R}^S\ q'$.

$\approx_{Wr}^S$ *is the largest indexed wired bisimulation.*

The third and fourth requirements say that if the participants have index $i \in S$, then the corresponding silent action may be neglected. Otherwise, the simulation should be done in a strong style. As will be seen, indexed wired bisimilarity implies indexed context bisimilarity.

Below are the properties of $\approx_{Wr}^S$.

Lemma 34. $\approx_{Wr}^S$ *is an equivalence and a congruence except on unguarded choice.*

Lemma 35. *Let* $\mathcal{R}^S$ *be a symmetric binary relation with respect to* $S \subseteq I$ *on indexed wired processes, if* $p\mathcal{R}^S q$ *implies*:

1. *if* $p \overset{\{a?x\}_i,S}{\Longrightarrow} p'$, *then* $q \overset{\{a?x\}_i,S}{\Longrightarrow} q'$ *for some* q', *and* $p'\{b\text{-}ichan/x\}\ \mathcal{R}^S\ q'\{b\text{-}ichan/x\}$ *for an indexed wire* b-$ichan$, *where* b *is fresh*;
2. *if* $p \overset{\{a!_B(b\text{-}ichan)\}_i,S}{\Longrightarrow} p''$, *where* b *is a name, then* $q \overset{\{a!_B(b\text{-}ichan)\}_i,S}{\Longrightarrow} q''$ *for some* q'', *and* $p''\ \mathcal{R}^S\ q''$;
3. *if* $p \overset{\varepsilon,S}{\Longrightarrow} p'$, *then* $q \overset{\varepsilon,S}{\Longrightarrow} q'$ *for some* q', *and* $p'\ \mathcal{R}^S\ q'$;
4. *if* $p \overset{\{\tau\}_{i,i}}{\Longrightarrow} p'$, $i \notin S$, *then* $q \overset{\{\tau\}_{i,i},S}{\Longrightarrow} q'$ *for some* q', *and* $p'\ \mathcal{R}^S\ q'$.

Then $\mathcal{R}^S$ *is an indexed wired bisimulation.*

The congruence induced from $\approx_{Wr}^S$ is denoted by $\simeq_{Wr}^S$.

4.2.3 Indexed Wired Factorization Theorem

We first give several properties on $p\{\diamond_a^i\, r\}$.

Lemma 36 (Substitution Lemma on $\{\diamond_{\boldsymbol{a}}^{\boldsymbol{i}}\}$**).** *Let* p, q, r *be indexed Plain CHOCS processes without unguarded choice operator. Then the following holds*:

$$p\{q/x\}\{\diamond_a^i\, r\} \approx_{Ct}^S p\{\diamond_a^i\, r\}\{q\{\diamond_a^i\, r\}/x\},$$

where $S \supseteq \{m, n\}$.

Lemma 37 (Distributivity on $\{\diamond_{\boldsymbol{a}}^{\boldsymbol{i}}\}$**).** *Let* p, q, q', r, r' *be indexed Plain CHOC processes without*

unguarded choice operator. Then it holds that ($S \supseteq \{m,n\}$):

1. $nil\{\diamond_a^i\, r\} \approx_{Ct}^S nil$;
2. $(\{i?\}_n.\{a?x\}_m.\{c!x\}_n.p)\{\diamond_a^i\, r\} \approx_{Ct}^S (\{\tau\}_{n,n}.\{\tau\}_{m,m}.\{\tau\}_{n,n}.(p \mid r))\{\diamond_a^i\, r\}$, *a appears in* $\{\diamond_a^i r\}$;
3. $(\{o?\}_n.\{c?x\}_n.\{a!p'\}_m.p)\{\diamond_a^i\, r\} \approx_{Ct}^S (\{\tau\}_{n,n}.\{\tau\}_{n,n}.\{\tau\}_{m,m}.(p \mid r\{p'/x\}))\{\diamond_a^i\, r\}$, *a appears in* $\{\diamond_a^i\, r\}$;
4. $(\{\tau\}_{k,k}.p)\{\diamond_a^i\, r\} \approx_{Ct}^S \{\tau\}_{k,k}.p\{\diamond_a^i\, r\}$, $k = m, n$;
5. $(p \mid p')\{\diamond_a^i\, r\} \approx_{Ct}^S p\{\diamond_a^i\, r\} \mid p'\{\diamond_a^i\, r\}$;
6. $((b)p)\{\diamond_a^i\, r\} \approx_{Ct}^S (b)(p\{\diamond_a^i\, r\})$, *if* b *does not appear in* $fn(r)$ *and* $\{\diamond_a^i\, r\}$ *related names*;
7. $(p[b \mapsto c])\{\diamond_a^i\, r\} \approx_{Ct}^S (p\{\diamond_a^i\, r\})[b \mapsto c]$, *if* b, c *do not appear in* $fn(r)$ *and* $\{\diamond_a^i\, r\}$ *related names*;
8. $(!p)\{\diamond_a^i\, r\} \approx_{Ct}^S !(p\{\diamond_a^i\, r\})$.

Lemma 38. *It holds that* $p \approx_{Ct}^S \{\tau\}_{k,k}.p$ ($k = m, n$), *for any indexed Plain CHOCS process* p *without unguarded choice operator and* $S \supseteq \{m,n\}$.

Now the indexed wired factorization theorem can be given.

Theorem 39 (Indexed Wired Factorization Theorem).

$$p\{r/x\} \approx_{Ct}^S (p\{a\text{-}ichan/x\}\{\diamond_a^i\, r\}),$$

for any indexed Plain CHOCS process p *without unguarded choice operator and indexed Plain CHOCS process* r, *and* $S \supseteq \{m,n\}$.

4.2.4 Relationship Between $\approx_{Wr}^S$ and $\approx_{Ct}^S$

The lemma below concludes some properties of $\approx_{Wr}^S$.

Lemma 40. *Suppose* p, q, r *are indexed wired processes. If* $p \approx_{Wr}^S q$ *with* $S \supseteq \{m,n\}$, *then* $(a)p \approx_{Wr}^S (a)q$ *and* $p \mid r \approx_{Wr}^S q \mid r$. *If* $p \simeq_{Wr}^S q$, *then* $p[a \mapsto b] \approx_{Wr}^S q[a \mapsto b]$.

Lemma 41. *Suppose* p, q *are indexed wired processes. If* $p \approx_{Ct}^S q$, *then* $p \approx_{Wr}^S q$ *does not generally hold, where* $S \supseteq \{m,n\}$.

Lemma 42. *Suppose* p, q *are indexed wired processes. If* $p \approx_{Wr}^S q$, *then* $p \approx_{Ct}^S q$, *where* $S \supseteq \{m,n\}$.

The two lemmas above immediately lead to the following theorem.

Theorem 43. $\approx_{Wr}^S$ *is strictly contained by* $\approx_{Ct}^S$ *on all indexed wired processes, where* $S \supseteq \{m,n\}$.

4.3 Toward the Full Abstraction Theorem

In this subsection we study the full abstraction of the encoding. The traditional definition of full abstraction is as follows.

Definition 44 (Full Abstraction). *If* $P \approx_1 Q$ *and iff* $[\![P]\!] \approx_2 [\![Q]\!]$, *where* $[\![\cdot]\!]$ *is some encoding from one calculus to another. And* $\approx_1$ *and* $\approx_2$ *are certain bisimilarities in the two calculi, respectively. We usually call the "only if" direction soundness, and the "if" direction completeness.*

An important property on the behavior of the indexed Plain CHOCS process encoding a FOPi process is that silent moves consisting of only those indexed by (n,n) preserve indexed wired bisimilarity.

Lemma 45. *Let* P *be a FOPi process and* $S = \{n\}$. *If* $[\![P]\!]_2^n \overset{\varepsilon,S}{\Longrightarrow} p'$, *then* $[\![P]\!]_2^n \approx_{Wr}^S p'$.

Proof. A structural induction on P. □

Notice that $\{\tau\}_{n,n}$ definitely comes from the communication on auxiliary ports (i,o,c) that are restricted in the communication and known only to the name itself being simulated, and there is no possibility of internal choice. For example, in translating an output prefix, silent actions indexed by n can merely occur before the actual output action by the encoding process.

The next two lemmas form the basis of full abstraction. They clarify the correspondence of actions before and after the encoding, or the correspondence of operational semantics.

Lemma 46 (Indexed Action Correspondence). *Suppose* P, Q *are FOPi processes. Then the following properties hold* ($S = \{m,n\}$, $B = \{b\}$) :

1. *if* $P \xrightarrow{a(x)} P'$, *then* $[\![P]\!]_2^n \overset{\{a?x\}_m,S}{\Longrightarrow} [\![P']\!]_2^n$;
2. *if* $P \xrightarrow{\overline{a}b} P'$, *then* $[\![P]\!]_2^n \overset{\{a!_\varnothing(b\text{-}ichan)\}_m,S}{\Longrightarrow} [\![P']\!]_2^n$;
3. *if* $P \xrightarrow{\overline{a}(b)} P'$, *then* $[\![P]\!]_2^n \overset{\{a!_B(b\text{-}ichan)\}_m,S}{\Longrightarrow} [\![P']\!]_2^n$;
4. *if* $P \xrightarrow{\tau} P'$, *then* $[\![P]\!]_2^n \xrightarrow{\{\tau\}_{m,m}} [\![P']\!]_2^n$ *or* $[\![P]\!]_2^n \overset{\{\tau\}_{m,m},S}{\Longrightarrow} [\![P']\!]_2^n$; *or simply* $[\![P]\!]_2^n \overset{\{\tau\}_{m,m},S}{\Longrightarrow} [\![P']\!]_2^n$.

Proof. By induction on the structure of P. It is a routine check and we skip the details. Noticing Lemma 30, one knows that the possible preceding and trailing silent actions in the weak transitions in the latter part of the clauses consist of only those indexed by (n,n). In fact, one can even figure out how many $\{\tau\}_{n,n}$ there are. □

We also have the converse of the lemma above. Note that in the non-indexed version, we do not state a corresponding lemma, since without index technique, the silent actions brought about by the encoding is not easy to identify in some desired detail.

Lemma 47. *Suppose* P, Q *are FOPi processes. The following properties hold* ($S = \{m,n\}$, $B = \{b\}$) :

1. *if* $[\![P]\!]_2^n \overset{\{a?x\}_m,S}{\Longrightarrow} p'$, *then* $P \overset{a(x)}{\Longrightarrow} P'$ *for some* P' *and* $p' \approx_{Wr}^S [\![P']\!]_2^n$;
2. *if* $[\![P]\!]_2^n \overset{\{a!_\varnothing(b\text{-}ichan)\}_m,S}{\Longrightarrow} p'$, *then* $P \overset{\overline{a}b}{\Longrightarrow} P'$ *for some* P' *and* $p' \approx_{Wr}^S [\![P']\!]_2^n$;

3. *if* $[\![P]\!]_2^n \overset{\{a!_B(b\text{-}ichan)\}_m,S}{\Longrightarrow} p'$ *where* $B=\{b\}$, *then* $P \overset{\overline{a}(b)}{\Longrightarrow} P'$ *for some* P' *and* $p' \approx_{Wr}^S [\![P']\!]_2^n$;
4. *if* $[\![P]\!]_2^n \overset{\{\tau\}_{m,m},S}{\Longrightarrow} p'$, *then* $P \overset{\tau}{\Longrightarrow} P'$ *for some* P' *and* $p' \approx_{Wr}^S [\![P']\!]_2^n$;
5. *if* $[\![P]\!]_2^n \overset{\varepsilon,S}{\Longrightarrow} p'$ *and the weak transition contains no silent actions indexed by* m, *then it does not indicate any action in* P.

Proof. By induction on the structure of P, the proof is routine, and we informally give some points. Suppose α is $\{a?x\}_m, \{a!_\varnothing(b\text{-}ichan)\}_m, \{a!_B(b\text{-}ichan)\}_m$, or $\{\tau\}_{m,m}$. The first point is that the action α in the weak transition in the premises of the clauses is the unique mark signaling the corresponding action (input, output or communication) in the encoded process. Secondly, the silent actions in $\overset{\alpha,S}{\Longrightarrow}$ other than α may comprise $\{\tau\}_{n,n}$ or $\{\tau\}_{m,m}$, so it is not sure whether all of them are from auxiliary communications or not, which is why we use weak transitions in the latter part of the clauses. Thirdly, we can only guarantee $\approx_{Wr}^S$ in each clause because $[\![P']\!]_2^n$ can differ from p' by the distance of some silent actions indexed by m or n. For example, p' can hold some $\{\tau\}_{n,n}$ generated by indexed wires, since the encoding strategy does not assert atomic translation of a certain prefix.

The first three clauses are not difficult to show. We give an intuitive analysis of the last one concerning silent actions. The silent action(s) may come from two sources. One is from an indexed τ action in a prefixed form. In this case, it is easy to see that $P \overset{\tau}{\Longrightarrow} P'$. The other case is that the silent action comes from a communication between two components after encoding. Thus it must have experienced some extra τ actions indexed by (n,n) before and/or after the $\{\tau\}_{m,m}$ action that corresponds to the communication in the original FOPi process. Taking the points mentioned above, we know again $P \overset{\tau}{\Longrightarrow} P'$. □

Based on the action correspondence properties, we have the following lemma on the relationship between indexed wired bisimulation and wired bisimulation. The result is essentially decided by the two encodings (indexed and non-indexed).

Lemma 48. *Let* P, Q *be FOPi processes and* $S = \{m,n\}$. *Then*

1. $[\![P]\!]_2^n \approx_{Wr}^S [\![Q]\!]_2^n$ *iff* $[\![P]\!]_2 \approx_{Wr} [\![Q]\!]_2$;
2. $[\![P]\!]_2^n \approx_{Ct}^S [\![Q]\!]_2^n$ *iff* $[\![P]\!]_2 \approx_{Ct} [\![Q]\!]_2$.

Proof. It is an easy consequence of Lemma 12 and Lemma 13. Notice that $Index([\![P]\!]_2^n) \cup Index([\![Q]\!]_2^n) = \{m,n\}$. □

Next we show two lemmas on the soundness and completeness of the indexed encoding.

Lemma 49. *If* $P \dot{\approx} Q$, *then* $[\![P]\!]_2^n \approx_{Ct}^S [\![Q]\!]_2^n$ *for* $S=\{m,n\}$.

Proof. Define $\mathcal{R}^S$ as follows:

$$\mathcal{R}^S \triangleq \{([\![P]\!]_2^n, [\![Q]\!]_2^n) \mid P \dot{\approx} Q\} \cup \approx_{Wr}^S,$$

where $S=\{m,n\}$. We prove that $\mathcal{R}^S$ is an indexed wired bisimulation up-to $\approx_{Wr}^S$, with respect to S. This suffices due to Theorem 43.

First of all, it is clear that $[\![P]\!]_2^n$ and $[\![Q]\!]_2^n$ are all indexed wired processes, by Lemma 32.

There are several cases to consider.

• If $[\![P]\!]_2^n \overset{\{a?x\}_m,S}{\Longrightarrow} p'$, then by Lemma 47, $P \overset{a(x)}{\Longrightarrow} P'$ for some P' and $p' \approx_{Wr}^S [\![P']\!]_2^n$. Since $P \dot{\approx} Q$, we have $Q \overset{a(y)}{\Longrightarrow} Q'$ for some Q' and $P'\{z/x\} \dot{\approx} Q'\{z/y\}$, for every name z. By definition, $[\![P'\{z/y\}]\!]_2^n \ \mathcal{R}^S \ [\![Q'\{z/y\}]\!]_2^n$. And by Lemma 46, $[\![Q]\!]_2^n \overset{\{a?x\}_m,S}{\Longrightarrow} [\![Q']\!]_2^n$. So we choose a fresh name h to correspond to name z, and it holds that $[\![P']\!]_2^n\{h\text{-}ichan/y\} \mathcal{R}^S [\![Q;]\!]_2^n\{h\text{-}ichan/y\}$. Attention to the definition of the indexed encoding, it is indeed $[\![P']\!]_2^n \mathcal{R}^S [\![Q']\!]_2^n$.

• If $[\![P]\!]_2^n \overset{\{a!_B(b\text{-}ichan)\}_m,S}{\Longrightarrow} p''$, where $B \cap (fn([\![P]\!]_2^n) \cup fn([\![Q]\!]_2^n) = \varnothing$ and $B=\{b\}$, for some b of b-*ichan*, then by Lemma 47, $P \overset{\overline{a}(b)}{\Longrightarrow} P'$ for some P' and $p'' \approx_{Wr}^S [\![P']\!]_2^n$. Since $P \dot{\approx} Q$, $Q \overset{\overline{a}(b)}{\Longrightarrow} Q'$ for some Q' and $P' \dot{\approx} Q'$, so we have $[\![Q]\!]_2^n \overset{\{a!_B(b\text{-}ichan)\}_m,S}{\Longrightarrow} [\![Q']\!]_2^n$. Thus it holds that $[\![P']\!]_2^n \ \mathcal{R}^S \ [\![Q']\!]_2^n$.

• If $[\![P]\!]_2^n \overset{\{a!_\varnothing(b\text{-}ichan)\}_m,S}{\Longrightarrow} p''$, then it is similar to the last case.

• The case $[\![P]\!]_2^n \overset{\{\tau\}_{i,i},S}{\Longrightarrow} p'$, $i \notin S$, will never appear, by Lemma 29.

• If $[\![P]\!]_2^n \overset{\varepsilon,S}{\Longrightarrow} p'$, then there are two subcases.

- $\overset{\varepsilon,S}{\Longrightarrow}$ contains no silent actions indexed by m. Then it results in an empty simulation, since P, Q do not have any clues on auxiliary silent actions brought about by indexed wires during encoding. By Lemma 45, we know $[\![P]\!]_2^n \approx_{Wr}^S p'$, so the matching step is closed, since $[\![P]\!]_2^n \ \mathcal{R} \ [\![Q]\!]_2^n$.
- If $[\![P]\!]_2^n \overset{\{\tau\}_{m,m},S}{\Longrightarrow} p'$, then by Lemma 47, $P \overset{\tau}{\Longrightarrow} P'$ for some P' and $p' \approx_{Wr}^S [\![P']\!]_2^n$. Since $P \dot{\approx} Q$, $Q \Longrightarrow Q'$ for some Q' and $P' \dot{\approx} Q'$. If the simulation is not empty, then by Lemma 46, $[\![Q]\!]_2^n \overset{\{\tau\}_{m,m},S}{\Longrightarrow} [\![Q']\!]_2^n$. And by definition, $[\![P']\!]_2^n \ \mathcal{R}^S \ [\![Q']\!]_2^n$. If the simulation is empty, then $P' \dot{\approx} Q' \equiv Q$, and $[\![P']\!]_2^n \ \mathcal{R}^S \ [\![Q]\!]_2^n$. In either case, it holds that $[\![Q]\!]_2^n \overset{\varepsilon,S}{\Longrightarrow} [\![Q']\!]_2^n$, and $[\![P']\!]_2^n \ \mathcal{R}^S \ [\![Q']\!]_2^n$.

Therefore, $\mathcal{R}^S$ is an indexed wired bisimulation

up-to $\approx^S_{Wr}$, with respect to S, so $[\![P]\!]^n_2 \approx^S_{Wr} [\![Q]\!]^n_2$. Then by Theorem 43, it holds that $[\![P]\!]^n_2 \approx^S_{Ct} [\![Q]\!]^n_2$ for $S = \{m, n\}$. □

Lemma 50. *If $[\![P]\!]^n_2 \approx^S_{Wr} [\![Q]\!]^n_2$ for $S = \{m, n\}$, then $P \dot{\approx} Q$.*

Proof. Define:

$$\mathcal{R} \triangleq \{(P, Q) \mid [\![P]\!]^n_2 \approx^S_{Wr} [\![Q]\!]^n_2\} \cup \dot{\approx}.$$

We show that $\mathcal{R}$ is an early bisimulation.

There are several cases to consider:

• $P \xrightarrow{a(x)} P'$. Then by Lemma 46, $[\![P]\!]^n_2 \overset{\{a?x\}_m, S}{\Longrightarrow} [\![P']\!]^n_2 \triangleq p'$. Since $[\![P]\!]^n_2 \approx^S_{Wr} [\![Q]\!]^n_2$, we know that $[\![Q]\!]^n_2 \overset{\{a?x\}_m, S}{\Longrightarrow} q'$ for some q', and $p'\{h\text{-}chan/x\} \approx^S_{Wr} q'\{h\text{-}chan/x\}$ for every fresh h, since we can choose an arbitrary fresh name. Thus by Lemma 47, $Q \overset{a(x)}{\Longrightarrow} Q'$ for some Q', and $q' \approx^S_{Wr} [\![Q']\!]^n_2$, so $q'\{h\text{-}chan/x\} \approx^S_{Wr} [\![Q']\!]^n_2\{h\text{-}chan/x\}$, and $[\![Q']\!]^n_2\{h\text{-}chan/x\}$ is exactly $[\![Q']\!]^n_2$, by the indexed encoding strategy. Now we have $P'\{z/x\} \mathcal{R} Q'\{z/x\}$ for all z, since $p'\{h\text{-}chan/x\} \approx^S_{Wr} q'\{h\text{-}chan/x\}$ for every fresh h (we use h to correspond to x), with attention paid to the definition of the indexed encoding.

• $P \xrightarrow{\overline{a}(b)} P'$. Then by Lemma 46, $[\![P]\!]^n_2 \overset{\{a!_B(b\text{-}ichan)\}_m, S}{\Longrightarrow} [\![P']\!]^n_2 \triangleq p'$, where $B = \{b\}$. Since $[\![P]\!]^n_2 \approx^S_{Wr} [\![Q]\!]^n_2$, we know that $[\![Q]\!]^n_2 \overset{\{a!_B(b\text{-}ichan)\}_m, S}{\Longrightarrow} q'$ for some q', and $p' \approx^S_{Wr} q'$. Thus by Lemma 47, $Q \overset{\overline{a}(b)}{\Longrightarrow} Q'$ for some Q' and $q' \approx^S_{Wr} [\![Q']\!]^n_2$. Now we have $P' \mathcal{R} Q'$, since $p' \approx^S_{Wr} q' \approx^S_{Wr} [\![Q']\!]^n_2$.

• $P \xrightarrow{\overline{a}b} P'$. This case is similar to the last one.

• $P \xrightarrow{\tau} P'$. Then by Lemma 46, $[\![P]\!]^n_2 \overset{\{\tau\}_{m,m}, S}{\Longrightarrow} [\![P']\!]^n_2 \triangleq p'$. Since $[\![P]\!]^n_2 \approx^S_{Wr} [\![Q]\!]^n_2$, we know that $[\![Q]\!]^n_2 \overset{\{\tau\}_{m,m}, S}{\Longrightarrow} q'$ for some q', or $[\![Q]\!]^n_2 \overset{\varepsilon, S}{\Longrightarrow} q'$ for some q' and the weak transition contains no silent moves indexed by m; and $p' \approx^S_{Wr} q'$ in either case. In the first case, by Lemma 47, we have $Q \overset{\tau}{\Longrightarrow} Q'$ for some Q' and $q' \approx^S_{Wr} [\![Q']\!]^n_2$. Then we have $P' \mathcal{R} Q'$, since $p' \approx^S_{Wr} q' \approx^S_{Wr} [\![Q']\!]^n_2$. In the second case, again by Lemma 47, we know Q has fired no action correspondingly, so the simulation is null, that is $Q \Longrightarrow Q' \equiv Q$. Then by Lemma 45, we have $[\![Q']\!]^n_2 \equiv [\![Q]\!]^n_2 \approx^S_{Wr} q'$. Then it holds that $[\![Q']\!]^n_2 \approx^S_{Wr} p'$. Hence we know $P' \mathcal{R} Q'$.

Therefore $\mathcal{R}$ is an early bisimulation, so $P \dot{\approx} Q$. □

We finally reach the full abstraction theorem.

Theorem 51 (Full Abstraction). *Suppose P, Q are FOPi processes. Then $P \dot{\approx} Q$ iff $[\![P]\!]^n_2 \approx^S_{Wr} [\![Q]\!]^n_2$ for $S = \{m, n\}$.*

Proof. A conclusion results from the previous two lemmas, Lemma 49 (by its proof) and Lemma 50. □

The following corollary is not hard to derive.

Corollary 52. *Suppose P, Q are FOPi processes. Then $P \dot{\approx} Q$ iff $[\![P]\!]_2 \approx_{Wr} [\![Q]\!]_2$.*

Proof. By Theorem 51 and Lemma 48. □

This corollary tells us that the original encoding is sound and complete, which is fully abstract, with respect to early bisimilarity and wired bisimilarity.

We can also attain the following corollary.

Corollary 53 (Soundness). *Suppose P, Q are FOPi processes. Then $P \dot{\approx} Q$ implies $[\![P]\!]_2 \approx_{Ct} [\![Q]\!]_2$.*

Proof. By Corollary 52 and Theorem 27. Or by Theorem 51, Theorem 43 and Lemma 48. □

This corollary tells us that the original (non-indexed) version of the encoding is sound, with respect to early bisimilarity and context bisimilarity. The major reason for the fail of completeness is that wired bisimilarity is strictly subsumed by context bisimilarity. In the next section, we discuss how to remedy this so as to retrieve the full abstraction.

5 Discussion

Our results of full abstraction of the encoding from FOPi to Plain CHOCS show that the encoding strategy is reasonable, though to go through the encoding some care should be taken to skip the extra silent actions brought about. Moreover, these results may somewhat complement the work by Sangiorgi on encoding higher-order π-calculus with first-order π-calculus[6]. Full abstraction with respect to different bisimilarities on either side can also be considered, for example, the barbed bisimilarities[6,9,17] on either side. These cases can be studied in a more similar fashion. The encoding strategy and related results still make sense.

The failure of the converse of Corollary 53 can be attributed to the selection of the bisimulation on the FOPi side. The crux is that early bisimilarity is not quite a reasonable bisimulation, and the clause for bound output, which ignores the local name in the simulation, is probably the killer of full abstraction. If we chose a more proper bisimilarity in FOPi, i.e., quasi open bisimilarity, and adjusted the wired bisimilarity in Plain CHOCS, we would arrive at the full abstraction. Below we sketch the idea.

The definition of a quasi open bisimulation involves a family of relations. An intuitive characterization of quasi open bisimilarity is local open bisimilarity[18]. Here we use the latter.

Definition 54 (Local Open Bisimilarity). *A symmetric binary relation $\mathcal{R}$ on FOPi processes is a local open bisimulation if:*

1. *it is closed under substitutions of names, and*

2. *whenever* $P \mathcal{R} Q$, *the following properties hold*:

• *if* $P \xrightarrow{\alpha} P'$, *where* α *is not a bound output, then some* Q' *exists such that* $Q \stackrel{\hat{\alpha}}{\Rightarrow} Q'$ *and* $P' \mathcal{R} Q'$;

• *if* $P \xrightarrow{\overline{a}(x)} P'$, *then for each process* A, *some* Q' *exists such that* $Q \stackrel{\overline{a}(x)}{\Rightarrow} Q'$ *and* $(x)(P' \mid A) \mathcal{R} (x)(Q' \mid A)$.

The local open bisimilarity, denoted by $\approx_{lo}$, *is the largest local open bisimulation.*

If we go through the encoding strategy, we naturally think of the adjustment of the definition of wired bisimulation, which much resembles context bisimulation, in that it is defined on wired processes.

Definition 55 (Local Wired Bisimulation). *A symmetric binary relation* $\mathcal{R}$ *on wired processes is a local wired bisimulation, whenever* $p \mathcal{R} q$ *the following properties hold*:

1. *if* $p \xrightarrow{a?x} p'$, *then for a wire b-chan, where b is a fresh name,* $q \stackrel{a?x}{\Longrightarrow} q'$ *for some* q', *and* $p'\{b\text{-}chan/x\} \mathcal{R} q'\{b\text{-}chan/x\}$;
2. *if* $p \xrightarrow{a!_B(b\text{-}chan)} p''$, *then for every* $E[x]$ *s.t.* $fn(E[x]) \cap B = \varnothing$, $q \stackrel{a!_B(b\text{-}chan)}{\Longrightarrow} q''$ *for some* q'', *and* $(B)(E[b\text{-}chan] \mid p'') \mathcal{R} (B)(E[b\text{-}chan] \mid q'')$;
3. *if* $p \xrightarrow{\tau} p'$, *then* $q \Longrightarrow q'$ *for some* q', *and* $p' \mathcal{R} q'$.

$\approx_{lw}$ *is the largest local wired bisimulation.*

It can be shown that local wired bisimilarity coincides with context bisimilarity.

Theorem 56. $\approx_{lw}$ *coincides with* $\approx_{Ct}$ *on wired processes.*

Reconsidering the encoding, we can have the following theorem. We leave out the details for space limit.

Theorem 57. *Suppose* P, Q *are FOPi processes. Then* $P \approx_{lo} Q$ *iff* $[\![P]\!]_2 \approx_{Ct} [\![Q]\!]_2$.

6 Conclusion and Future Work

Our work complements the encoding on higher-order calculi and first-order calculi. We have shown that the encoding of FOPi with Plain CHOCS possesses the full abstraction with respect to early bisimilarity and wired bisimilarity, and the soundness with respect to early bisimilarity and context bisimilarity. To achieve this, we resort to the index technique. We propose (indexed) wired processes, prove the (indexed) wired factorization theorem, and show the relationship between the (indexed) wired bisimilarity and (indexed) context bisimilarity on (indexed) wired processes. The index technique is also used to make smooth the correspondence of operational semantics between FOPi and the (indexed) Plain CHOCS. Moreover the related definitions and results of closed processes can be extended to open processes in the usual way as well. The major results of the encoding declare that the expressive power of the two calculi (FOPi and Plain CHOCS) are comparable in the sense of the correspondence of the designated bisimilarities on them respectively (early bisimilarity and wired bisimilarity or context bisimilarity). Some discussion is also made on how to achieve full abstraction on a different bisimilarity.

The following topics are worth considering in the future.

• The encoding in this paper may provide an alternative way to encode λ-calculus. We know from [6] that λ-calculus can be encoded in higher-order π-calculus in a direct way, and the encoding can be mapped to the indirect encoding strategy by Milner in [14], from (lazy) λ-calculus to first-order π-calculus. Based on the encoding strategy discussed here, one can design an indirect encoding of (lazy) λ-calculus using higher-order π-calculus by the following two steps:

1) Milner's encoding: from (lazy) λ-calculus to first-order π-calculus;

2) encoding discussed in this paper: from first-order π-calculus to higher-order π-calculus.

We think it possible to map this strategy to Sangiorgi's direct encoding. This can contribute to the research of the relationship between functional programming and concurrent programming, and enable applying tools in one language to the other.

• What if we keep the choice operator in FOPi will the major results remain true? It appears not a trivial task. A completely new approach would possibly be needed to get over the technical details, if the overall strategy were correct. Research on this can probably offer some insight into the essence of the choice operator in influencing the expressive capability of the first-order calculus. Another question is whether we can revoke relabelling in the encoding strategy. If this is possible the encoding can enjoy a neater and more intuitive style in handling wires, apart from offering a more essential understanding of names. Again this seems hard, since the use of relabelling is quite stubborn in the strategy.

• Applications are anticipated, for example, modeling biological procedures or security protocols using process-passing instead of name-passing. The point is that sometimes it is much more difficult to use FOPi to model some biological process, because it may contain some essentially "higher-order" elements (such as a cell going through a membrane) which are not easy to capture using name-passing mechanism. In such cases, a higher-order calculus can provide a way to model the procedure more closely. In the meanwhile one can be assured that the modeling capability is adequate.

Acknowledgements The author would like to

express his gratitude to Professor Yuxi Fu for his encouragement and many helpful discussions. The author is also indebted to Professor Davide Sangiorgi for his reviewing of this paper and some overall suggestions on improvements. We also thank the anonymous reviewers for their patient comments. Thanks also go to all the members of the BASICS Lab.

References

[1] Thomsen B. Plain CHOCS, a second generation calculus for higher-order processes. *Acta Informatica*, 1993, 30(1): 1–59.

[2] Cao Z. More on bisimulations for higher order π-calculus. In *Proc. FOSSACS2006*, Vienna, Austria, Aceto L, Ingólfsdóttir A (eds.), *LNCS* 3921, 2006, pp.63–78.

[3] Thomsen B. A calculus of higher order communication systems. In *Proc. POPL'89*, Austin, TX, USA, 1989, pp.143–154.

[4] Thomsen B. Calculi for higher order communicating systems [Ph.D. Dissertation]. Department of Computing, Imperial College, 1990.

[5] Thomsen B. A theory of higher order communication systems. *Information and Computation*, 1995, 116: 38–57.

[6] Sangiorgi D. Expressing mobility in process algebras: First-order and higher-order paradigms [Ph.D. Dissertation]. University of Edinburgh, 1992.

[7] Sangiorgi D. From π-calculus to higher-order π-calculus—and back. In *Proc. TAPSOFT'93*, Orsay, France, *LNCS* 668, Springer Verlag, 1992, pp.151–166.

[8] Sangiorgi D. Bisimulation for higher-order process calculi. *Information and Computation*, 1996, 131(2): 141–178, Preliminary version: In *Proc. the IFIP Working Conference on Programming Concepts, Methods and Calculi* (*PROCOMET'94*), North Holland, 1994, pp.207–224.

[9] Sangiorgi D, Walker D. On barbed equivalences in pi-calculus. In *Proc. CONCUR'01*, Aalborg, Denmark, *LNCS* 2154, 2001, pp.292–304.

[10] Sangiorgi D, Walker D. The Pi-Calculus: A Theory of Mobile Processes. Cambridge University Press, 2001.

[11] Yongjian Li, Xinxin Liu. Towards a theory of bisimulation for the higher-order process calculi. *Journal of Computer Science and Technology*, May 2004, 19(3): 352–363.

[12] Yuxi Fu. Checking equivalence for higher order processes. Technical Report, SJTU BASICS, 2005.

[13] Milner R, Parrow J, Walker D. A calculus of mobile processes (Parts I and II). *Information and Computation*, 1992, 100(1): 1–77.

[14] Milner R. Functions as processes. *Journal of Mathematical Structures in Computer Science*, 1992, 2(2): 119–141. Research Report 1154, INRIA, Sofia Antipolis, 1990.

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

[16] Sangiorgi D, Milner R. The problem of weak bisimulation up-to. In *Proc. CONCUR'92*, *LNCS* 630, 1992, pp.32–46.

[17] Milner R, Sangiorgi D. Barbed bisimulation. In *Proc. the 19th International Colloquium on Automata, Languages and Programming* (*ICALP'92*), Vienna, Austria, *LNCS* 623, Springer Verlag, 1992, pp.685–695.

[18] Yuxi Fu. On quasi open bisimulation. *Theoretical Computer Science*, 2005, 338(1-3): 96–126.

Xian Xu is a Ph.D. candidate in BASICS Lab, Department of Computer Science and Engineering, Shanghai Jiao Tong University (SJTU). Currently he is affiliated with Department of Computer Science and Engineering, East China University of Science and Technology. He obtained the B.S. degree in computer science from SJTU in 2002, and obtained his M.S. degree in computer science in DCTC (Distributed Computing Technology Center) from SJTU in 2005. His current research interests include concurrency, especially process calculi, such as their semantics, bisimulation theory, axiomatization, expressive power and etc. He also takes interest in the application of concurrency models to systems biology and (security) protocol analysis and verification, and has got some publications.