Expressing First-order Pi-calculus in Higher-order CCS

Xian Xu

BASICS Lab
Department of Computer Science and Technology
Shanghai Jiao Tong University
800 Dong Chuan Road (200240), Shanghai, China
Email: xuxian@sjtu.edu.cn

Abstract

In the research field of process calculi, encoding between different calculi is an effective way to compare the expressive power of them and can shed light on the essence of where the difference lies. Thomsen and Sangiorgi have worked on the higher-order calculi (higher-order CCS and higher-order pi-calculus, respectively) and the encoding to and from first-order pi-calculus. But the work is not complete, in that the encoding of first-order pi-calculus with higher-order CCS is missing. In this paper, we try to settle this part. We first review the results of related work. Then we introduce the index technique that is used to abstract from extra silent actions, which are brought about during the encoding, in correspondence of operational semantics of first-order pi-calculus processes and higher-order CCS processes encoding them. The encoding strategy (an indexed version of the encoding by Thomsen but without renaming) is then presented, where indexed wires play a crucial role. To arrive at the full abstraction property, we introduce the indexed wired processes, and the indexed wired factorization theorem is presented. Then the coincidence between the indexed wired bisimilarity and indexed context bisimilarity on indexed wired processes is obtained. After that, we prove the major result of the encoding, the full abstraction theorem, with respect to indexed ground bisimilarity (first-order pi-calculus) and indexed context bisimilarity (higher-order CCS). Finally, we make some discussions on our work and suggest some future work.

1 Introduction

In this paper, we present some recent work on the relation of expressive power between HOCCS (High-Order CCS) and FOPi (First-Order Pi-calculus), by working on the encoding from FOPi to Plain CHOCS, which was first studied by Thomsen [15]. But Thomsen simply put forward the idea and a rough framework of the relationship between HOCCS and FOPi, and the full abstraction was not obtained. We intend to further this study and make more detailed the ‘bi-simulation’ underlying the encoding. The encoding in the other direction is discussed by Sangiorgi in his PhD thesis [8].

We work from the basic encoding strategy of Thomsen [15] and make necessary refinement, such as removing the renaming in the language. To get through the technical stuff, we make use of a helpful technique, the index technique [1] that abstracts from extra silent actions in the correspondence of operational semantics of FOPi in the encoding with Plain CHOCS, to get over the rough detail, and formulate the encoding strategy under index technique. We need a theorem we call indexed wired factorization, which implies that two indexed Plain CHOCS processes (processes granted with indices on prefixes to indicate the source of an action) can be compared by concentrating on the corresponding indexed wired processes, which 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 of the two processes. We have a mapping that maps an indexed Plain CHOCS process to an indexed wired process. On indexed wired processes we define indexed wired bisimulation, and moreover we show that indexed wired bisimilarity coincides with indexed context bisimilarity on indexed wired processes. With the help of all the necessary intermediate results, we finally prove the important

*The author is supported by The National Distinguished Young Scientist Fund of NNSFC (60225012), BDCC (03DZ14025), The National Nature Science Foundation of China (60473006), MSRA, and The BoShiDian Research Fund (20010248033).
full abstraction theorem of the indexed version of the encoding, with respect to ground bisimulation (FOPi) and indexed context bisimulation (Plain CHOCs).

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

Related work

As well as we are concerned, Thomsen is the first researcher to obtain some preliminary results on the bisimulation theories of Plain CHOCs \cite{14,15,16}. His contribution mainly consists of the following parts.

- **Bisimulation.** The bisimulation put forward by Thomsen is considered basic and structural yet not intuitively reasonable. It is called applicative higher-order bisimulation (its former is the higher-order bisimulation in which restriction is dynamic \cite{13}), where restriction operator is statically scoped. The applicative higher-order bisimulation treats process being sent and the residual process separately. For example, two processes $P$ and $Q$ below

  \[
  P \triangleq \pi A.P', \quad Q \triangleq \pi B.Q'
  \]

  are considered equivalent under applicative higher-order bisimulation if

  \[
  A, B \quad \text{and} \quad P', Q'
  \]

  are equivalent respectively (we eliminate local names to make the example simple). This is a natural manipulation, but can cause some counter-intuitive problems \cite{11}. Despite the disadvantage of it, the applicative higher-order bisimilarity possesses some good algebraic properties, such as congruence. Thomsen’s work does contribute to the higher-order field in providing a relatively complete framework of bisimulation theory.

- **Encoding Plain CHOCS with FOPi, and vice versa.** Thomsen makes a try to encode Plain CHOCS with FOPi and also the converse. But the full abstraction, with respect to the ground bisimilarity (FOPi) and applicative higher-order bisimilarity (Plain CHOCS), is not available, probably because the bisimulations he uses may not be suitable for a natural encoding with full abstraction. He merely makes two conjectures on the full abstraction property of the two encodings, with no proof. Later on, some flaw is found by Sangiorgi in the encoding, with an counterexample worked out \cite{8}, and some mend is tried. Yet the idea of Thomsen has profound effect on following work, including ours.

Sangiorgi, as far as we know, is one of the first researchers to study the bisimulation theories of higher-order pi-calculus \cite{8,9,11}. His contribution mainly covers the following respects.

- **Bisimulations: context bisimulation, triggered bisimulation, normal bisimulation and barbed bisimulation.** Based on the previous study on bisimulation, Sangiorgi makes an intuitive analysis and puts forth the more reasonable context bisimulation, which observes the communicated process and residual process at the same time, rather than comparing them separately. For instance, two processes $P$ and $Q$ below

  \[
  P \triangleq \pi A.P', \quad Q \triangleq \pi B.Q'
  \]

  are considered equivalent under context bisimulation if

  \[
  E[A] \mid P' \quad \text{and} \quad E[B] \mid Q'
  \]

  are equivalent for any higher-order process $E$ with the process variable $X$ appearing in it (we simplify by eliminating local names), and $E[A]$ means the process obtained by replacing all appearance of $X$ in $E$ with $A$ (with no name capturing). Moreover, to overcome the universal quantifier in the context bisimulation, the normal bisimulation is designed, where the universal quantifier disappears and the comparison of the output action is focused on triggers, a special
kind of processes to activate an instance of a process sent when necessary. To get over the difficulty of proving the coincidence between context bisimilarity and normal similarity, triggered bisimulation is used as an intermediate. 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 shows that context bisimilarity coincides with barbed congruence.

- Encoding higher-order pi-calculus with first-order pi-calculus, and the full abstraction with respect to the barbed congruence on either side. Sangiorgi gives an encoding of higher-order pi-calculus with first-order pi-calculus, on the basis of the bisimulations mentioned above, and proves that the encoding satisfies the full abstraction property, in the sense that the barbed congruence [12][18], which is a reduction based bisimulation coinciding with context bisimulation and normal bisimulation (in the largest sense), preserves in the encoding. The strategy is relatively simple after the theories on triggers. It simply substitutes the trigger with its subject name to demote its order and makes necessary technical handling to get through the encoding. And the full abstraction is shown by taking advantage of the coincidence of the three bisimilarities, that is the context, normal and barbed.

Cao [1] proves the coincidence between context bisimilarity and normal bisimilarity in the strong versions that is left as unsolved in Sangiorgi’s earlier paper, using the index technique he put forward, which assigns locations (in the form of indices) to a process (on its action capabilities) and later can help selectively filter out some extra silent actions in terms of requirement of some bisimulation. The technique is not essential from certain point of view; it only makes some refinement on action sequences, and the proof is not new compared with Sangiorgi’s work. However, the index technique can offer some handy approach to cope with the twisted actions in some technical handling of bisimulation related job, such as encoding between different calculi, since the encoding may introduce some extra internal actions.

There is also other work on bisimulation theory of higher-order process, such as [4][5] and that by Fu[3][2] on linear higher-order pi-calculus to make axiomatization possible, to name a few. Our work does not depend on them very much.

To summarize, in the research field of process calculi, encoding between different calculi is an effective way to compare the expressive power of them and can shed light on the essence of where the difference lies. Though Thomsen and Sangiorgi have worked on the higher-order calculi (Plain CHOCS and higher-order pi-calculus, respectively) and the encoding to and from first-order pi-calculus, the work is not complete yet, in that the encoding of first-order pi-calculus with Plain CHOCS is missing. Inspired by the ideas in Thomsen’s and Sangiorgi’s research, we try to settle this part in our paper.

Compared with the work by Thomsen and Sangiorgi, our work primarily contributes to the field in the following points.

- We refine Thomsen’s encoding strategy [15] by getting rid of the renaming operator. Based on the strategy, we put forth the wired processes and present related results, including the wired mapping (from Plain CHOCS processes to wired processes), the wired factorization theorem, wired bisimilarity and its coincidence with context bisimilarity.

- Applying index technique to the encoding strategy above which is now free of renaming operator. The results in the first point above (under non-indexed encoding) are regenerated in the indexed version of encoding. We show the action correspondence lemmas, where the correspondence of actions in the two calculi is made more clear, thanks to the index technique.

- The major result, that is, the full abstraction theorem, which justifies the reasonability of the encoding. The full abstraction is with respect to the ground bisimilarity on the FOPi side and indexed context bisimilarity on the indexed Plain CHOCS side.

In the rest of this paper, we first introduce the three calculi involved in the encoding, including their syntax, operational semantics and bisimulations. Next we present the indexed version of the
encoding from first-order pi-calculus to Plain CHOCS. We then go through a discussion on the properties of indexed wires, including the coincidence of indexed wired bisimilarity with indexed context bisimilarity. Combining these properties and the action correspondence, we finally prove the full abstraction theorem. In the end, we also consider some future work worth working on, before the conclusion.

2 Basic definitions of the calculi

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

2.1 FOPi calculus

The definition of a FOPi process is as follows.

**Definition 1 (FOPi Syntax)**

\[ P ::= 0 \mid x(y).P \mid \tau.P \mid (x)P \mid P\{y/x\} \]

We use capital letters \( P, Q, \ldots \) to represent FOPi processes. A first-order pi process is built from the inactive process 0, three prefixes called input, output and silent, restriction, parallel composition, and replication. The precedence of operators is (from high to low): restriction, prefix, replication, parallel composition. When the object communicated in a prefix is not important, it is usually omitted. A name \( a \) is said to be bound in \( b(a)P \) and \( (a)P \), otherwise it is free. We use \( fn(\cdot), bn(\cdot), n(\cdot) \) for free name, bound name and name, respectively. We allow alpha-conversion, so generally we can always assume no name capturing shall occur. We use \( \{y/x\} \) to indicate substitution on first-order names. \( P\{y/x\} \) can be structurally defined.

We eliminate the (nondeterministic) choice to down-scale the expressive power of the FOPi calculus, because it is both a convenient and a troublesome operator both in the first-order and higher-order paradigm, as indicated in [11],[15].

We give the operational semantics (the labeled transition system) of FOPi in the Appendix [A]. The input \( (x(y)) \), output \( (xy) \) and silent \( (\tau) \) actions are not hard to understand, and bound output \( (\tau(y)) \) is an abbreviation for outputting a local name. The others all have their usual meaning in pi-calculus paradigm. So we skip most of them, but note that we use static binding for restriction operator, that is, the scope of a restricted name will be extruded when the name is sent to another process. We use \( \alpha, \beta, \lambda, \ldots \) to denote actions.

We give the basic bisimulation of FOPi, called ground bisimulation, which was given in [7], and will be used to show the reasonability of the encoding. Here the action sequence \( \alpha \Rightarrow \) denotes \( \Rightarrow \alpha \Rightarrow \), and \( \Rightarrow \Rightarrow \) denotes a sequence of \( \tau \) actions (possibly zero), where \( \alpha \) is an action.

**Definition 2 (Ground bisimulation)**. (Weak) ground bisimilarity, \( \approx \), is the largest symmetric binary relation \( R \) on FOPi processes, if \( P \approx Q \), then:

1. \( P \stackrel{x(y)}{\Rightarrow} P' \), then for every name \( z \), \( Q \stackrel{x(y)}{\Rightarrow} Q' \) for some \( Q' \) and \( P'\{z/y\}RQ'\{z/y\} \);
2. \( P \stackrel{\tau(y)}{\Rightarrow} P' \), then \( Q \stackrel{\tau(y)}{\Rightarrow} Q' \) for some \( Q' \) and \( P'RQ' \);
3. \( P \stackrel{\tau(y)}{\Rightarrow} P' \), then \( Q \stackrel{\tau(y)}{\Rightarrow} Q' \) for some \( Q' \) and \( P'RQ' \);
4. \( P \Rightarrow P' \), then \( Q \Rightarrow Q' \) for some \( Q' \) and \( P'RQ' \).

2.2 Plain CHOCS

The higher-order CCS we study is Plain CHOCS [15].

**Definition 3 (Plain CHOCS Syntax)**

\[ p ::= nil | x | a ? x . p | a ! p \cdot p | \tau . p | p | p' | p \backslash a | p+p' | ! p \]
Plain CHOCS defined above is a higher-order calculus, with no first-order fragment. We use lowercase letters \( p, q, r, \ldots \) to represent Plain CHOCS processes. A Plain CHOCS process is built from the inactive process \( \text{nil} \), process variable \( x \), three prefixes called input, output and silent, parallel composition, restriction, (nondeterministic) choice, and replication (in fact it is not needed and can be simulated, but we include it for convenience). The precedence of operators is (from high to low): restriction, prefix, replication, parallel composition, choice. When the object in a prefix is not important, it is omitted, for example \( a?\text{nil} \). A name \( a \) is said to be bound in \( p|_a \) otherwise it is free. A process variable \( x \) is bound in \( a(x).p \), otherwise it is free. We use \( fn(\cdot), bm(\cdot), n(\cdot) \) for free name, bound name and name, respectively, and \( f v(\cdot), bv(\cdot), v(\cdot) \) for free variable, bound variable and variable, respectively. A Plain CHOCS process is closed if it has no free process variables, otherwise it is open. We generally concentrate on closed processes. So if not mentioned, all processes are closed. And we allow alpha-conversion. We use \([r/x] \) to indicate substitution on higher-order names. 

\[ p[r/x] \text{ and } p[y/x] \] can be structurally defined.

We give the operational semantics of Plain CHOCS in Appendix A. The input, output and silent actions are defined as usual. In output action, if \( B = \emptyset \) it is sometimes omitted. We use \( \alpha, \beta, \lambda, \ldots \) to denote actions. By \( p \setminus B = \{b_1, b_2, \ldots, b_n\} \) is a set of names, we mean that \( p \setminus b_1 \setminus b_2 \cdots \setminus b_n \). The rules for parallel composition and choice are regular. The rules worthy of some attention are the \textit{com} rule and \textit{open} rule, where local names are involved. In the former a set of local names are extruded (static binding), and in the latter a new name is localized. The \textit{non-struct} rule allows more names than relevant to the sent process.

The applicative higher-order bisimulation put forward by Thomsen is not intuitive in that it separates the process sent and remaining, so later the context bisimulation improved this in combining the comparison of them in the same environment. The definition is given below.

**Definition 4 (Context bisimulation).** (Weak) context bisimilarity, \( \approx_{Ct} \), is the largest symmetric binary relation \( R \) on closed Plain CHOCS processes, if \( pRq \) then:

- \( p \xrightarrow{a} p' \), then \( q \xrightarrow{a} q' \) for some \( q' \), and \( p'[r/x]Rq'[r/x] \) for all closed process \( r \);
- If \( p \xrightarrow{a|b} p'' \), there exists s.t. \( q \xrightarrow{a|b} q'' \), and for all \( E(x) \) with \( fn(E(x)) \cap B = \emptyset \),
  \[ (p''|E[p']) \setminus B \approx (q''|E[q']) \setminus B. \]
- If \( p \xrightarrow{r} p' \), then \( q \xrightarrow{r} q' \) for some \( q' \), and \( p' \approx R q' \).

We use \( \sim_{Ct} \) to indicate strong context bisimilarity, whose definition is similar to the weak version above (by replacing double-arrow with single-arrow and some slight modification). We will use this notation henceforth. Context bisimulation up-to \( \approx_{Ct} \) can be defined in a traditional way. And one can easily prove that a relation \( R \) that is a context bisimulation up-to \( \approx_{Ct} \) is contained in \( \approx_{Ct} \).

**Some properties**

Next we give a few properties and definitions on context bisimilarity. The first is that \( \approx_{Ct} \) is a well-defined relation.

**Lemma 5.** \( \approx_{Ct} \) is an equivalence relation on (closed) Plain CHOCS processes.

We give some simple algebraic laws for \( \approx_{Ct} \), and their proofs are not hard.

**Lemma 6.** Suppose \( p, p_1, p_2, p_3 \) are (closed) Plain CHOCS processes. Then the following holds:

\[ p|\text{nil} \approx_{Ct} p; \quad p_1|p_2 \approx_{Ct} p_2|p_1; \quad p_1|(p_2|p_3) \approx_{Ct} (p_1|p_2)|p_3; \]
\[ p+\text{nil} \approx_{Ct} p; \quad p_1|p_2 \approx_{Ct} p_2|p_1; \quad p_1|(p_2|p_3) \approx_{Ct} (p_1|p_2)|p_3; \]
\[ p_1+(p_2+p_3) \approx_{Ct} (p_1+p_2)+p_3; \quad p\cdot a\cdot b \approx_{Ct} p\cdot b\cdot a; \quad (p_1|p_2)|a \approx_{Ct} (p_1|a)|p_2, \text{ if } a \notin fn(p_2); \]
\[ 1p \approx_{Ct} 1p. \]

Congruence property of two bisimilarities can be concluded in the following lemma.

**Lemma 7.** \( \sim_{Ct} \) is a congruence on (closed) Plain CHOCS processes.

\( \approx_{Ct} \) is a congruence on (closed) Plain CHOCS processes without choice operator.

Note the up-to technique can also be applied to context bisimulations. And we will mention them when necessary.
2.3 Indexed Plain CHOCS

The index technique was first put forth in \[1\]. Below we will make use of this technique in the discussion of action correspondence in the encoding strategy. Intuitively, the indexed bisimulation family lies across a series of bisimulations, with the weak and strong bisimulations on two ends. We think some instance between the two ends can be used in characterizing of the full abstraction of the encoding from FOPi to Plain CHOCS.

The index technique can be justified in the following points.

- The idea of the technique is to assign indices to actions to indicate the identities of the communication components, and interaction is indexed by the union of the indices of the participants.
- Intuitively, an index of (the prefixes/actions of) a process can be seen as a certain kind of name or location of the process in the interaction scene, just like in a distributed environment.
- The general idea of our making use of index technique is to use it to filter out some technically introduced silent actions in the encoding strategy.

2.3.1 Indexed processes

We give the syntax of indexed Plain CHOCS processes, which differs from the non-indexed processes in that each prefix has an identity called index, which can be used to identify the source of it.

Definition 8 (Indexed Plain CHOCS syntax).

\[ h ::= \text{nil} \mid x \mid \{a?x\}_i.h \mid \{a!p\}_i.h \mid \{\tau\}_{i,j}.h \mid h.h' \mid h \backslash a \mid h + h' \mid !h, \]

where \(i, j \in I\) and \(I\) is a countable set of indices, typically the set of natural numbers \(\mathbb{N}\). Indexed Plain CHOCS processes (or simply indexed processes) are usually denoted by \(h, k, l, m, m, p, q, \ldots\), and most time they will not be confused with non-indexed processes if the context is considered.

We use \(i, j, k, \ldots\) to denote indices. \(\{\tau\}_{i,j}, \{a?k\}_i, \{a!k\}_i, \{a!Bk\}_i\) are actions in indexed Plain CHOCS, and they are usually denoted by \(\alpha, \beta, \lambda, \ldots\). We do not differentiate them from non-indexed actions, for their meaning can be identified from context. In restriction, we revoke the indices. Its meaning is as usual, except that they point to the names in indexed prefixes (or actions). Free (bound) names and free (bound) variables can be similarly defined like those in Plain CHOCS. Weak action sequence is defined as follows (suppose \(S \subseteq I\), \(\alpha\) is an indexed action, and in each pair in \((i_1, j_1), \ldots, (i_n, j_n)\) at least one component belongs to \(S\)):

\[ \alpha \xrightarrow{\varepsilon,S} S \triangleq \{\tau\}_{i_1,j_1}, \ldots, \{\tau\}_{i_n,j_n}; \]

Note the length of \(\varepsilon,S\) can be zero. It indicates the idea that now silent actions are not all neglected, but selectively, with the help of the index set \(S \subseteq I\), as will be clear in the bisimulation definition. Also note the weak transition here is slightly from that in \(\Pi\), and this results from the need for manipulating indexed wires, which will be clear below.

We give the operational semantics of indexed Plain CHOCS in appendix \(A\). The LTS is similar to the LTS for Plain CHOCS. The only difference is that the actions are all indexed with indices. Note the \(\text{com}\) rule, where the silent action is indexed by the union of the positions of its participants, that is \(\{\tau\}_{i,j}\). The other rules are not difficult to understand.

2.3.2 Indexed bisimulation

We give the indexed versions of the bisimulations in Plain CHOCS. They correspond to the non-indexed version, only different in that actions are indexed and an index set \(S \subseteq I\) is used to selectively filter out some silent actions of interest.

Below comes the indexed context bisimulation. It is slightly different from that in \(\Pi\). And we make this adjustment to get over the technical details concerning extra silent actions involving indexed wires defined below.
Definition 9 (Indexed context bisimulation).  (Weak) indexed context bisimilarity with respect to index set $S \subseteq I$, $\approx_{Ct}^S$, is the largest symmetric binary relation $R^S$ on (closed) indexed Plain CHOCS processes, if $pR^S q$ then:

1. $p \stackrel{a \cdot x_i}{\Rightarrow} p'$, then $q \stackrel{a \cdot x_i}{\Rightarrow} S q'$ for some $q'$, and $p'[r/x]R^S q'[r/x]$ for all closed indexed Plain CHOCS process $r$;
2. If $p \stackrel{a \cdot b \cdot y}{\Rightarrow} p''$, $q'$ exists s.t. $q \stackrel{a \cdot b \cdot y}{\Rightarrow} S q''$, and for all indexed $E(x)$ with $f_n(C(x)) \cap B = \emptyset$, $(p'' | E[p']) \backslash B R^S (q'' | E[q']) \backslash B$
3. $p \stackrel{(\tau)_i,j}{\Rightarrow} p'$, $i, S) \Rightarrow q'$ for some $q'$, and $p'^R^S q'$.
4. $p \stackrel{(\tau)_i,j}{\Rightarrow} p'$, $i, j \notin S$, then $q \stackrel{(\tau)_i,j}{\Rightarrow} S q'$ for some $q'$, and $p'^R^S q'$.

We note that the silent actions are treated differently from the non-indexed bisimulations. The main difference is that only those silent actions with at least one of the indices in $S$ is neglected, not all of them. For those silent actions whose indices do not belong to $S$, they are treated in a strong simulation way. So this definition can be somewhat regarded as between strong and weak bisimulations.

Below we give some properties and definitions on indexed context bisimilarity.

Lemma 10. $\approx_{Ct}^S$ is an equivalence relation on (closed) indexed Plain CHOCS processes.

Below we give a technical lemma. The proof of it is like those for an ordinary weak bisimulation.

Lemma 11. A symmetric binary relation $R^S (S \subseteq I)$ on (closed) indexed Plain CHOCS processes, if $pR^S q$ implies:

1. $p \stackrel{a \cdot x_i}{\Rightarrow} p'$, then $q \stackrel{a \cdot x_i}{\Rightarrow} S q'$ for some $q'$, and $p'[r/x]R^S q'[r/x]$ for all indexed closed process $r$;
2. If $p \stackrel{a \cdot b \cdot y}{\Rightarrow} p''$, $q'$ exists s.t. $q \stackrel{a \cdot b \cdot y}{\Rightarrow} S q''$, and for all indexed $E(x)$ with $f_n(C(x)) \cap B = \emptyset$, $(p'' | E[p']) \backslash B R^S (q'' | E[q']) \backslash B$
3. $p \stackrel{a \cdot b \cdot y}{\Rightarrow} p'$, then $q \stackrel{a \cdot b \cdot y}{\Rightarrow} S q'$ for some $q'$, and $p'^R^S q'$.
4. $p \stackrel{(\tau)_i,j}{\Rightarrow} p'$, $i, j \notin S$, then $q \stackrel{(\tau)_i,j}{\Rightarrow} S q'$ for some $q'$, and $p'^R^S q'$.

then $R^S \subseteq \approx_{Ct}^S$.

Remark

We include generally choice operator in (indexed) Plain CHOCS process definition, however, we restrain their appearance in the discussion of the encoding. That is, we merely permit conditional usage of choice operator in some cases, no abusing allowed. Specifically, for instance, in discussing indexed wired processes, including indexed wired mapping, indexed wired factorization theorem and a series properties, we do not allow choice in processes (in premises) to avoid its notorious damage in congruence property of bisimulation relations. But we still need to use the choice operator in some places, typically in the wires, because wires serve to simulate names in FOPi and choice seems indispensable in the encoding, if no other alternative approach is available. Therefore, we shall constrain the usage of choice operator, only in wires, its corresponding mapping, and related positions in processes, for example in $\{ \phi_n \}$ as will be defined later, no other appearance is allowed, otherwise most of the important lemmas and theorems will fail. Similar discussion on the choice operator can be found in [8], where only guarded summation is allowed. In this paper, we will point out those condition on choice whenever necessary.

This convention will not do any harm, from the viewpoint of the formulation of the theories in wires. In fact, it reflects in a sense that the choice operator is powerful but sometimes so strong that some trouble may come along. Hence some limit must be imposed to gain the properties in need.
3 Encoding strategy under indexed technique

Now we consider the encoding using index technique. The original non-indexed encoding strategy was first put forth by Thomsen [15]. The idea is similar, except that we apply index technique to the encoding strategy and we do not use renaming.

An indexed wire, written \( a \rightarrow ichan \), with respect to a name \( a \) and an index \( n \) in the encoding strategy, is defined as:

\[
a \rightarrow ichan \triangleq \{i?\} n \cdot \{a?x\} n \cdot \{c!x\} n \cdot nil + \{o?\} n \cdot \{c?x\} n \cdot \{a!x\} i \cdot nil,
\]

where \( i, n \in I \). We use \( fn(a \rightarrow ichan) \) to indicate \( \{a, i, c, o\} \), that is the indexed names in it. The name \( i \) and the index \( i \) shall not be confused.

**Remark 12.** Note an indexed wire, which is designed to simulate a name in FOPi, is (strictly) a higher-order indexed Plain CHOCS process. The static restriction operator assures the scope extrusion when communications occur with output of bound names. Also note the silent actions caused by components indexed \( n \) is to be ignored in the action sequences related in certain bisimulations. \( i \in I \) (different from \( n \)) is a fresh index to be assigned to each prefix appearing in the encoded process. We only need one index \( i \neq n \) to get over all the technical detail here (to index actions other than those indexed by \( n \)).

Now we give the indexed version of the encoding.

3.1 The indexed version of the encoding

**Definition 13** (The indexed encoding from FOPi to Plain CHOCS). The definition also has two parts, taking index technique into account.

- \( [] \downarrow_1 \).
  \[
  \begin{align*}
  [0]_1^n & \triangleq \text{nil} \\
  [x(y).P]_1^n & \triangleq (x \mid \{l!\} n \cdot \{c?y\} n \cdot [P]_1^n) \backslash c \backslash l \circ o \\
  [\tau y.P]_1^n & \triangleq (x \mid \{o!\} n \cdot \{c!y\} n \cdot [P]_1^n) \backslash c \backslash l \circ o \\
  [\tau.P]_1^n & \triangleq \{\tau\} i, i, [P]_1^n \\
  [P[P']_1^n & \triangleq [P]_1^n \mid [P']_1^n \\
  [P]_1^n & \triangleq \lfloor [P]_1^n \rfloor \\
  [(x)P]_1^n & \triangleq ([P]_1^n((a \rightarrow ichan) \backslash x)) \backslash a, a \notin fn(P)
  \end{align*}
  \]

- \( [] \downarrow_2 \).
  \[
  [P]_2^n \triangleq (\cdots ([P]_1^n[(a_1 \rightarrow ichan) \backslash x_1]) \cdots )[(a_n \rightarrow ichan) \backslash x_n]
  \]

A free name in FOPi is first mapped to a process variable (an identity mapping), then it is substituted by an indexed wire. There is a one–to–one mapping between FOPi names and Plain CHOCS indexed wires. That is, it is from \( fn(P) = fn([P]_1^n) = \{x_1, x_2, \cdots, x_n\} \) to \( \{a_1, a_2, \cdots, a_n\} \). Usually for convenience the mapping can be an identity, since names can be distinguished in the two calculi.

**Remark.** We add indices to the original encoding strategy, to help filter out extra silent actions in the simulating input and output on a name. The renaming (or relabeling) used in the original encoding is discarded here, because it would not do any harm to the full abstraction. Since renaming is usually considered that renaming is quite a complicated and even dubious operator, for its behavior, especially with recursion, can result in rather involved computations, with the additional choice of static or dynamic binding of the restriction operator.

3.2 Characterizing indexed wires

In this section, we formulate the results concerning indexed encoding strategy. And this will serve as the basis for proving the full abstraction of the indexed version of the encoding. We have considered the counterparts in the original non-indexed encoding strategy, and put them in Appendix [B] for reference.
The lemmas and theorems have essentially the same proofs as those corresponding counterparts in the non-indexed versions. The critical point is that under index technique, all the processes are treated in the uniform way that silent actions are selectively neglected according the index set $S$ in a bisimulation. So the effect is that all the properties still hold in the indexed paradigm, with the difference in that silent actions are abstracted out in another uniform way. So we generally do not give the details of the proofs and focus on the framework, however you can find the proofs for the non-indexed encoding strategy in Appendix [13].

### 3.2.1 Indexed wired processes

Indexed wired processes are indexed Plain CHOCS processes that only send or receive wires during communication, and have no choice operator in all positions other than wires and $p\{c_i\}$, as will be defined below. We use $\mathcal{IWPr}$ to denote indexed wired processes.

The indexed wired mapping $\mathcal{W}_n[\cdot]$ transfers an indexed Plain CHOCS process into an indexed wired process.

**Definition 14** (Indexed wired mapping). We define a mapping from indexed Plain CHOCS processes without choice operator to indexed wired processes as follows:

\[
\begin{align*}
\mathcal{W}_n[\text{nil}] & \triangleq \text{nil} \\
\mathcal{W}_n[x] & \triangleq x \\
\mathcal{W}_n[(a?x)_i,p] & \triangleq \{a?x\}_i,\mathcal{W}_n[p] \\
\mathcal{W}_n[(a!p)_i,p] & \triangleq \{(a!\text{ichan})\}_i,\mathcal{W}_n[p]
\end{align*}
\]

Note that the mapping will limitedly introduce choice (in indexed wires and related positions). Below is an obvious result on the indexed version of the encoding (Definition [13]).

**Lemma 15.** Suppose $P$ is a FOPi process. then $\mathcal{W}_n[P]$ is a Plain CHOCS indexed wired process.

### 3.2.2 Indexed wired bisimulation

Here we give the definition of indexed wired bisimulation.

**Definition 16** (Indexed wired bisimulation). $\approx_{\mathcal{W}_n}$ is the largest symmetric binary relation $\mathcal{R}_n$ with respect to index set $S$ on (closed) indexed wired processes, if $p\mathcal{R}_n q$ implies:

1. $p \xrightarrow{\{a?x\}_i} p'$, then $q \xrightarrow{\{a?x\}_i,\mathcal{R}_n} q'$ for some $q'$, and $p'[\text{ichan}/x]\mathcal{R}_n q'[\text{ichan}/x]$ for some indexed wire $b\text{ichan}$, where $b$ is a fresh name;

2. $p \xrightarrow{a\{b\text{ichan}\}_i} p''$, where $b$ is a fresh name and $B = \{i, o, c\}$, then $q \xrightarrow{a\{b\text{ichan}\}_i,\mathcal{R}_n} q''$ for some $q''$, and $p''\mathcal{R}_n q''$;

3. $p \xrightarrow{\tau}_i p'$, $i \in S$ or $j \in S$, then $q \xrightarrow{\mathcal{R}_n} q'$ for some $q'$, and $p'\mathcal{R}_n q'$.

4. $p \xrightarrow{\tau}_i p'$, $i,j \notin S$, then $q \xrightarrow{\tau}_i,\mathcal{R}_n q'$ for some $q'$, and $p'\mathcal{R}_n q'$.

Note the third and fourth requirements. They say that if one of the participants have index $n$, then the silent may be neglected. Otherwise, the simulation should be done in a strong style. In fact, indexed wired bisimilarity ($\approx_{\mathcal{W}_n}$) coincidences with the indexed context bisimilarity ($\approx_{\mathcal{C}_1}$) ($S = \{n\}$). We will see this in the following arguments. Up-to technique also exist in the index paradigm, and we will explain whenever necessary.
3.2.3 Indexed wired factorization theorem

To arrive at the coincidence of indexed wired bisimilarity $(\approx_{\mathcal{W}}^S)$ with indexed context bisimilarity $(\approx_{\mathcal{Ct}}^S(S = \{n\}))$, we need a number of concepts and related properties, among which the indexed wired factorization theorem is most important.

We use $p\{\phi^i_a\ r\}$ to abbreviate $(p!!\{\{i\}\}_n,\{c?x\}_n,\{d!\}_n,\{a?\}_n,\{a?\}_n,\{r\}_n)\ a\ (i, n \in I)$, where the superscript $i$ indicates “index”. We give several properties on the abbreviation. The first is about the substitution of $\{\phi^i_a\}$.

Lemma 17 (Substitution lemma on $\{\phi^i_a\}$). Let $p, q, r$ be (closed) indexed Plain CHOCS processes without choice operator. Then the following holds:

\[ p[q/x]\{\phi^i_a\ r\} \approx_{\mathcal{Ct}}^S p[\phi^i_a\ r][q\{\phi^i_a\ r\}/x], \]

where $S = \{n\}$

Next is another lemma on the distributivity of $\{\phi^i_a\}$.

Lemma 18 (Distributivity on $\{\phi^i_a\}$). Let $p, q, q', r, r'$ be (closed) indexed Plain CHOCS processes without choice operator. Then it holds that $(S = \{n\})$:

1. $\text{nil}\{\phi^i_a\ r\} \approx_{\mathcal{Ct}}^S \text{nil}$;
2. $\{(a!p')\}_n.p\{\phi^i_a\ r\} \approx_{\mathcal{Ct}}^S (\{i!\}_n,\{\tau\}_n,\{c?x\}_n,\{a?\}_n,\{p[r']p'/x\}) + (\{d!\}_n,\{c!x\}_n,\{\tau\}_n,\{p[r']p'/x\})\{\phi^i_a\ r\}$,

\[ a \text{ appears in } \{\phi^i_a\ r\} \text{ and } r \text{ is something of the form } a(x)r'; \]
3. $(\{\tau\}_n, p\{\phi^i_a\ r\} \approx_{\mathcal{Ct}}^S \{\tau\}_n, p\{\phi^i_a\ r\}$;
4. $(p|p')\{\phi^i_a\ r\} \approx_{\mathcal{Ct}}^S p\{\phi^i_a\ r\}|p'\{\phi^i_a\ r\}$;
5. $(p|b)\{\phi^i_a\ r\} \approx_{\mathcal{Ct}}^S (p\{\phi^i_a\ r\})\ b \text{ if } b \text{ does not appear in } f_n(r) \text{ and } \{\phi^i_a\ r\} \text{ related names};$
6. $(l_p)\{\phi^i_a\ r\} \approx_{\mathcal{Ct}}^S l_p\{\phi^i_a\ r\}$.

Below are several lemmas necessary for the indexed wired factorization theorem. The following lemma is immediately provable.

Lemma 19. It holds that $p \approx_{\mathcal{Ct}}^S \{\tau\}_n, p$, for all indexed Plain CHOCS processes without choice operator and $S = \{n\}$.

The following lemma shows that instead of passing an indexed process, it can be replaced with an indexed wire, the effect of which is adding an extra $\{\tau\}_n, p$ action before the replaced process.

Lemma 20.

\[ p[\{\tau\}_n, p/x] \approx_{\mathcal{Ct}}^S (p[a\text{-chan}/x])\{\phi^i_a\ r\}, \]

for all indexed processes $p$ without choice operator and indexed process $r$.

Now we can give the indexed wired factorization theorem, that is,

Theorem 21 (Indexed wired factorization theorem).

\[ p[r/x] \approx_{\mathcal{Ct}}^S (p[a\text{-ichan}/x])\{\phi^i_a\ r\}, \]

for all indexed processes $p$ without choice operator and indexed process $r$, and $S = \{n\}$.
3.2.4 The coincidence theorem

Now we reach the crucial theorem that is, the coincidence between indexed higher-order context bisimilarity $\approx_{\text{Ct}}$ and indexed wired bisimilarity $\approx_{\text{Wr}}$ (S = \{n\}). We first give the congruence property of indexed wired bisimulation ($\approx_{\text{Wr}}$).

**Lemma 22.** Suppose $p, q, r$ are indexed wired processes, and $a$ is a name. If $p \approx_{\text{Wr}} q$ with $S = \{n\}$, then $p \\rightarrow_{\Delta} q$ \approx_{\text{Wr}} q \\rightarrow_{\Delta} r$.

Now we prove two lemmas comprising the needed theorem.

**Lemma 23.** Suppose $p, q$ are (closed) indexed wired processes. If $p \approx_{\text{Wr}} q$, then $p \approx_{\text{Ct}} q$, where $S = \{n\}$.

**Lemma 24.** Suppose $p, q$ are closed indexed wired processes. If $p \approx_{\text{Wr}} q$, then $p \approx_{\text{Ct}} q$, where $S = \{n\}$.

The two lemmas above immediately lead to the following theorem.

**Theorem 25.** $\approx_{\text{Ct}}$ coincides with $\approx_{\text{Wr}}$ on all indexed wired processes free of choice operator, where $S = \{n\}$.

3.3 Full abstraction theorem

In this section we try to gain the full abstraction of the indexed version of the encoding. The traditional definition of full abstraction is as follows.

**Definition 26 (Full abstraction).** If $P \approx_1 Q$ iff $[P]_2 \approx_2 [Q]_2$, where $[\cdot]_2$ is some encoding from one calculus to the other. And $\approx_1$ and $\approx_2$ are certain bisimilarities in the two calculi, respectively.

From the indexed version of the encoding strategy, we can deduce the following lemma on the correspondence in actions between the FOPi process and the indexed Plain CHOCS process encoding it. And its proof differs slightly from the non-indexed version.

**Lemma 27.** The following two equations hold for $S = \{n\}$ and $i \in I$:

$$[a(x).P]_2 \approx_{\text{Ct}} \{a?x\}_{1,i},[P]_{2,i},$$
$$[a\hat{b}.P]_2 \approx_{\text{Ct}} \{a!(b\text{-ichan})\}_{1,i},[P]_{2,i}.$$  

The next two lemmas form the basis of the proof of full abstraction below. They clarify the correspondence of actions before and after the indexed version of the encoding. Lemma 28 corresponds to the action correspondence lemma in the non-indexed encoding, and the proof is straightforward from the encoding strategy.

**Lemma 28 (Indexed action correspondence).** Suppose $P, Q$ are FOPi processes. The correspondence in actions between a process and its indexed encoding is as follows ($S = \{n\}$, $i \in I$, $B = \{b\}$):

- If $P \xrightarrow{a(x)} P'$, then $[P]_2 \xrightarrow{\{a?x\}_{1,i}} [P']_2$;
- If $P \xrightarrow{\text{mb}} P'$, then $[P]_2 \xrightarrow{\{a!(b\text{-ichan})\}_{1,i}} [P']_2$;
- If $P \xrightarrow{\text{mb}} P'$, then $[P]_2 \xrightarrow{\{a!(b\text{-ichan})\}_{1,i}} [P']_2$;
- If $P \xrightarrow{\tau} P'$, then $[P]_2 \xrightarrow{(\tau)_{1,i}} [P']_2$ or $[P]_2 \xrightarrow{(\tau)_{1,i}} [P']_2$.

We also have the converse of the above lemma. Note that in 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 a desired detail. To analyze the action sequence to the extent that silent actions are treated properly, we need the index technique.

**Lemma 29.** Suppose $P, Q$ are FOPi processes. The converse of Lemma 28 ($S = \{n\}$, $i \in I$, $B = \{b\}$) is stated as follows:
• If \([P]_2 \overset{(a^x)_{i,S}}{\longrightarrow} p', \) then \(P \overset{a^x}{\longrightarrow} P'\) for some \(P'\) and \(p' \equiv [P']_2\);

• If \([P]_2 \overset{(a_0(b\text{-ichan}))_{i,S}}{\longrightarrow} p', \) then \(P \overset{a_0(b\text{-ichan})}{\longrightarrow} P'\) for some \(P'\) and \(p' \equiv [P']_2\);

• If \([P]_2 \overset{(a_2(b\text{-ichan}))_{i,S}}{\longrightarrow} p', \) then \(P \overset{a_2(b\text{-ichan})}{\longrightarrow} P'\) for some \(P'\) and \(p' \equiv [P']_2\);

• If \([P]_2 \overset{(r)_{i,S}}{\longrightarrow} p'\) or \([P]_2 \overset{(r)_{i,S}}{\longrightarrow} p', \) then \(P \overset{r}{\longrightarrow} P'\) for some \(P'\) and \(p' \equiv [P']_2\).

**Proof.** The proof is generally similar to Lemma 28. With the help of indices, the first three correspondences are not difficult to show. We focus on the last one concerning silent actions. Formally, the proof is by induction on the structure of \(P\).

Here we give an intuitive analysis. The silent action(s) can come from two sources. One is from an indexed \(\tau\) 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 \(\tau\) actions indexed by \(n\) before and/or after the observable \(\{r\}_{i,S}\) action that simulates the communication in the original FOPi process. So we know again \(P \overset{\tau}{\longrightarrow} P'\).

Next we show two lemmas comprising the theorem of full abstraction. The first lemma is as follows.

**Lemma 30.** If \(P \approx Q\), then \([P]_2 \approx [Q]_2\) for \(S = \{n\}\).

**Proof.** Define \(\mathcal{R}^S\) as follows:

\[\mathcal{R}^S \triangleq \{(P, Q) | P \approx Q\} \cup \approx_{W, r},\]

where \(S = \{n\}\). We prove that \(\mathcal{R}^S\) is an indexed wired bisimulation with respect to \(S\).

First of all, it is clear that \([P]_2\) and \([Q]_2\) are all indexed wired processes, by Lemma 15

There are several cases to consider (\(i \in I\)):

• If \([P]_2 \overset{(a^x)_S}{\longrightarrow} p', \) then by Lemma 29, \(P \overset{a^x}{\longrightarrow} P'\) for some \(P'\) and \(p' \equiv [P']_2\). Since \(P \approx Q\), for every name \(z\), \(Q \overset{a^y}{\longrightarrow} Q'\) for some \(Q'\) and \(P'[z/x] \approx Q'[z/y]\). By definition, \([P']_2 \overset{a^z}{\longrightarrow} [Q']_2\). And by Lemma 28, \([Q]_2 \overset{(a^x)_S}{\longrightarrow} [Q']_2\). So we choose a fresh name \(h\) to correspond to name \(z\), and it holds that \([P]_2 \overset{a^z}{\longrightarrow} [Q']_2\) according to the definition of the encoding, (it is indeed \([P']_2 \overset{a^z}{\longrightarrow} [Q']_2\)).

• If \([P]_2 \overset{(a_0(b\text{-ichan}))_S}{\longrightarrow} p''\), where \(B \cap (fn([P]_2) \cup fn([Q]_2) = \emptyset\) and \(B = \{b\}\) for some \(b\) of \(b\text{-ichan}\), then by Lemma 29, \(P \overset{a_0(b\text{-ichan})}{\longrightarrow} P'\) for some \(P'\) and \(p'' \equiv [P']_2\). Since \(P \approx Q\), \(Q \overset{a_0(b\text{-ichan})}{\longrightarrow} Q'\) for some \(Q'\) and \(P' \approx Q'\), so we have \([Q]_2 \overset{(a_0(b\text{-ichan}))_S}{\longrightarrow} [Q']_2\). Thus it holds that \([P]_2 \overset{a_0(b\text{-ichan})}{\longrightarrow} [Q']_2\).

• If \([P]_2 \overset{(a_2(b\text{-ichan}))_S}{\longrightarrow} p'\). Similar to the last case.

• If \([P]_2 \overset{(r)_{i,S}}{\longrightarrow} p', \) then it results in an empty simulation. In fact, this case shall not appear.

• If \([P]_2 \overset{(r)_{i,S}}{\longrightarrow} p', \) then by Lemma 29, \(P \overset{r}{\longrightarrow} P'\) for some \(P'\) and \(p' \equiv [P']_2\). Since \(P \approx Q\), \(Q \overset{r}{\longrightarrow} Q'\) for some \(Q'\) and \(P' \approx Q'\). Then by Lemma 28, \([Q]_2 \overset{(r)_{i,S}}{\longrightarrow} [Q']_2\). And by definition, \([Q]_2 \overset{(r)_{i,S}}{\longrightarrow} [Q']_2\).

Therefore, \(\mathcal{R}^S\) is an indexed wired bisimulation with respect to \(S\), so \([P]_2 \approx_{W, r} [Q]_2\). Then by Theorem 25, it holds that \([P]_2 \approx_{Ct} [Q]_2\) for \(S = \{n\}\).

The other direction of the full abstraction is stated in the following lemma.

**Lemma 31.** If \([P]_2 \approx_{Ct} [Q]_2\) for \(S = \{n\}\), then \(P \approx Q\).

**Proof.** The other direction is not difficult to prove. We give it below. Define:

\[\mathcal{R} \triangleq \{(P, Q) | [P]_2 \approx_{W, r} [Q]_2\} \cup \approx,\]

it suffices to show that \(\mathcal{R}\) is a ground bisimulation, since \(\approx_{W, r}\) coincides with \(\approx_{Ct}\) by Theorem 25.

There are several cases to consider: (\(i \in I\))

12
technique). Here we point out some work that can be carried out in the future. Some possible topics silent actions) brought about by the encoding strategy, with the help of some technique (the index

Our result on full abstraction of the encoding shows that the encoding strategy is reasonable and

4 Future work

Proof. A concluding result from the previous two lemmas, Lemma 30 and Lemma 31.

Therefore, \( \mathcal{R} \) is a ground bisimulation, that is \( \mathcal{R} \subseteq \warsz\), so \( P \approx Q \).

Now we finally reach the full abstraction theorem.

Theorem 32 (Full abstraction 3). Suppose \( P, Q \) are (closed) FOPi processes. Then \( P \approx Q \) iff

\[ [P]_2 \approx^S \{ a \} \Rightarrow [Q]_2 \approx^S \{ b \} \]

for \( S = \{ n \} \).

Our result on full abstraction of the encoding shows that the encoding strategy is reasonable and correct, though to go through the encoding some care should be taken to skip something (the extra silent actions) brought about by the encoding strategy, with the help of some technique (the index technique). Here we point out some work that can be carried out in the future. Some possible topics are:

- More full abstraction consideration, with respect to different bisimilarities on either side. For example, the barbed bisimilarities on either side, or the late version of the bisimilarities. We believe these cases can be studied in a much similar fashion. Moreover, some refinement on the bisimilarities related to full abstraction of the encoding may exist. That is, there may exist some not-found relations contained by the bisimilarities under discussion, and this can make the encoding more precise.
- Using the full calculi in the encoding, that is keeping the choice operator. Is it possible? It seems that it is not a trivial task. A completely new approach would be needed to get over the technical details, if the overall strategy were correct. As a matter of fact, (nondeterministic) choice is always a powerful but troublesome operator to coexist with, but this research can probably offer some insight into the essence of this operator in influencing the expressive capability of a certain calculus.
- Applications. There can be some applications to modeling some procedures, such as biological processes in signaling transduction, to show the effect of the encoding. The point is that sometimes it is much difficult to use FOPi to model some biological procedure, because the procedure may contain some “higher-order” elements and not so easy to capture using name-passing mechanism. In such cases, a higher-order calculus (for example Plain CHOCS) can provide a much more convenient method to describe the biological procedure more closely. And one can be assured that the modeling capability is the same.
• More study on the computational power of the two calculi. The encoding sheds some light on the comparison of expressive power between them. We can continue to study more such pairs on different process calculi, such as asynchronous process calculi on one side of comparison since it is closer to real-life behavior of processes in a sense.

5 Conclusion

In conclusion, we have done in this paper an essential task which, as far as we know, complements the encoding work in the field. That is, the encoding of first-order pi-calculus with higher-order CCS (Plain CHOCS), with the desired result of full abstraction with respect to ground bisimilarity (first-order pi-calculus) and indexed context bisimilarity (Plain CHOCS) with the help of the index technique. To achieve this, we propose the indexed wired processes, prove the indexed wired factorization theorem, and show the coincidence between the indexed wired bisimilarity and indexed context bisimilarity on indexed wired processes, which eases the proof of the full abstraction. We also use the index technique to make smooth the correspondence of operational semantics between first-order pi-calculus processes and the indexed Plain CHOCS processes encoding them. Thus we eventually make it to prove the full abstraction theorem, which says that the expressive capabilities of the two calculi (FOPi and Plain CHOCS) coincide in the sense of the corresponding of the two designated bisimilarities (ground bisimilarity and indexed context bisimilarity) on them respectively.

Acknowledgements. The author would like to express his gratitude to Yuxi Fu for his encouragement and many helpful discussions. The author is also much indebted to Sangiorgi for his reviewing of this paper and some overall suggestions on improvements.

References


Appendix

Below is the appendix of the paper.

A The operational semantics of the calculi

We give the operational semantics of the three calculi in our paper, that is, FOPi, Plain CHOCS and indexed Plain CHOCS.

We give the operational semantics of FOPi in Figure 1

\[
\begin{align*}
\text{(alp)} & \quad \frac{P \xrightarrow{\lambda} Q}{P' \xrightarrow{\lambda} Q} \quad P \equiv_\alpha P' \\
\text{(prefix)} & \quad x(y).P \xrightarrow{z} P\{z/y\} \quad \overline{\pi y}.P \xrightarrow{y} P' \quad \tau.P \xrightarrow{\tau} P \\
\text{(com)} & \quad \frac{P \xrightarrow{\pi y}, P', Q \xrightarrow{z} Q'}{P|Q \xrightarrow{z} P'|Q'\{y/z\}} \\
\text{(close)} & \quad \frac{P \xrightarrow{\pi (y)}, P', Q \xrightarrow{z} Q'}{P|Q \xrightarrow{z} (y)(P'|Q'\{y/z\})} \\
\text{(par)} & \quad \frac{P \xrightarrow{\lambda}, P', Q \rightarrow P'|Q}{P|Q \xrightarrow{\lambda} P'|Q}, \quad bn(\lambda) \cap fn(Q) = \emptyset \\
\text{(res)} & \quad \frac{P \xrightarrow{\lambda}, P'}{(x)P \xrightarrow{\nu(x)} P'}, \quad x \notin n(\lambda) \\
\text{(open)} & \quad \frac{P \xrightarrow{\pi y}, P'}{(y)P \xrightarrow{\pi(y)} P'}, \quad y \neq x \\
\text{(rep)} & \quad \frac{P \xrightarrow{\lambda}, P'}{!P \xrightarrow{\lambda}, P'|!P} 
\end{align*}
\]

Figure 1: LTS of FOPi.

We give the operational semantics of Plain CHOCS in Figure 2
We give the operational semantics of indexed Plain CHOCS in Figure 3
Figure 2: LTS of Plain CHOCS.

Figure 3: LTS of indexed Plain CHOCS.
The original encoding strategy

The original encoding strategy is due to Thomsen \[15\], but he did not further its study, only some action correspondence is considered. We inherit the basic encoding strategy in \[15\] and make some discussions on the strategy. What is worth noting is that we revoke the renaming operator, which is a questionable operator in the cooperation with restriction operator. And it will be shown that the cleansed strategy is enough to have the properties desired. The result here will serve the main body of this paper in reaching the full abstraction.

The core idea is to use the apparatus called \textit{wire} \[15\]. It is a special process serving the function of a name, which can be a \textit{channel}, a \textit{variable} and a \textit{local} one of the above.

A wire, written \( a \rightarrow \text{chan} \), with respect to a name \( a \) in the encoding strategy is defined as:

\[
a \rightarrow \text{chan} \triangleq i\cdot a\cdot x.c!x.nil + o\cdot c\cdot x.a!x.nil
\]

Note the wire is (strictly) a \textit{higher-order} (Plain CHOCS) process\[1\] \( i\cdot a\cdot x.c!x.nil \) and \( o\cdot c\cdot x.a!x.nil \) 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 communicated. We use \( fn(a \rightarrow \text{chan}) \) to indicate \( \{a, i, c, o\} \), that is the names in it.

Also note \( \sim \) is the strong applicative higher-order bisimilarity, and \( \approx \) is the weak applicative higher-order bisimilarity. The definition of the latter can be found in \[15\]. Note an immediate property is that \( \approx \) implies \( \approx_{\text{Cl}} \).

Now we give the encoding strategy. \[15\].

\textbf{Definition 33} (The Encoding from FOPi to Plain CHOCS). The definition of the encoding strategy has two parts. We give them below:

- \( \llbracket \cdot \rrbracket_1 \):

  \[
  \begin{align*}
  [0]_1 & \triangleq \text{nil} \\
  [x(y).P]_1 & \triangleq (x \mid !l.c?y.[P]_1) \setminus i \setminus o \\
  [\tau y.P]_1 & \triangleq (x \mid o.l.c?y.[P]_1) \setminus c \setminus i \setminus o \\
  [\tau .P]_1 & \triangleq \tau .[P]_1 \\
  [P\mid P']_1 & \triangleq [P]_1 \mid [P']_1 \\
  ![P]_1 & \triangleq ![P]_1 \\
  [(x)P]_1 & \triangleq ([P]_1[(a \rightarrow \text{chan})/x]) \setminus a, a \notin fn(P)
  \end{align*}
  \]

- \( \llbracket \cdot \rrbracket_2 \):

  \[
  [P]_2 \triangleq (\cdots ([P]_1[(a_1 \rightarrow \text{chan})/x_1]) \cdots )[(a_n \rightarrow \text{chan})/x_n]
  \]

A free name in FOPi is first mapped to a process variable (an identity mapping), then it is substituted by a wire. It is set that there is a \textit{one-to-one} mapping between FOPi names and Plain CHOCS wires. That is, it is from \( fn(P) = fv([P]_1) = \{x_1, x_2, \ldots, x_n\} \) to \( \{a_1, a_2, \ldots, a_n\} \). Sometimes for convenience the mapping can be identity, since names can be distinguished in the two related calculi, so we can use this with enough care.

\textbf{Remark}. We modifies the original encoding strategy that uses the renaming (or relabeling) and relinquish the dubious operator, for the reason that it suffices to obtain the important results, as will be clear in the discussion below. It is generally deemed that renaming is a much complicated operator, for its behavior, especially with recursion, can lead to rather involved computations.

Note there is an implicit operation that every the local name variable of FOPi is transferred to a process variable in Plain CHOCS. Below we give two lemmas related to the encoding above, and they are similar to those in \[15\].

\[1\] It can have a prefix without any parameters, like \( i.P \) or \( \overrightarrow{i}.P \), and that means the communicated process is not important, similar to those in FOPi.
Lemma 34 (Wire). The following two equations hold:

\[ a(x).P \subseteq [P]_2 \]

\[ \bar{\text{ab}}.P \subseteq [P]_2 \]

Proof. The proof is quite direct from the encoding strategy. It is not difficult to make a routine analysis.

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

Lemma 35 (Action correspondence). There are several corresponding actions between a process and its encoding:

- If \( P \xrightarrow{a(x)} P' \), then \( [P]_2 \xrightarrow{\tau.\bar{a}x.\tau} [P']_2 \);
- If \( P \xrightarrow{?a} P' \), then \( [P]_2 \xrightarrow{\tau.\bar{a}x.\tau} [P']_2 \);
- If \( P \xrightarrow{?b} P' \), then \( [P]_2 \xrightarrow{\tau.\bar{a}x.\tau} [P']_2 \);
- If \( P \xrightarrow{a} P' \), then \( [P]_2 \xrightarrow{\tau.\bar{a}x.\tau} [P']_2 \).

B.1 Characterizing wires

Here we study the wires. In light of the triggers and triggered bisimulation \([11]\), we work out a series of properties on behavior of wires, and pave the way for proving the full abstraction theorem.

B.1.1 Wired processes

We first define a category of processes called wired processes.

Wired processes are higher-order processes that only send or receive wires during communication, and have no choice operator in positions other than wires and \( p\{\diamond a\} \), as will be defined later. We use \( WP_r \) to denote wired processes.

The next mapping, called wired mapping, denoted by \( W[\cdot] \), translate a Plain CHOCS process without choice operator into a wired process.

Definition 36 (wired mapping). We define a mapping from Plain CHOCS processes without choice operator to wired processes as follows:

\[

to W[\cdot]
\]

Note that the mapping will limitedly introduce choice (in wires and related positions). An obvious result from the encoding strategy (Definition 33) is as the following lemma states.

Lemma 37. Suppose \( P \) is a (closed) FOPi process, then \( [P]_2 \) is a Plain CHOCS wired process.
B.1.2 Wired bisimulation

We give the definition of a bisimulation called wired bisimulation.

Definition 38 (Wired bisimulation). \( \approx_{W} \) is the largest symmetric binary relation \( R \) on (closed) Plain CHOCS wired processes, if \( pRq \) implies:

1. \( p \xrightarrow{a} p' \), then \( q \xrightarrow{a} q' \) for some \( q' \), and \( p'[b\text{-}chan/x]Rq'[b\text{-}chan/x] \) for some wire \( b\text{-}chan \), where \( b \) is a fresh name;
2. \( p \xrightarrow{a \{ w \} (b\text{-}chan)} p'' \), where \( b \) is a fresh name and \( B = \{ i, o, c \} \), then \( q \xrightarrow{a \{ w \} (b\text{-}chan)} q'' \) for some \( q'' \), and \( p''Rq'' \);
3. \( p \xrightarrow{\tau} p' \), then \( q \xrightarrow{\tau} q' \) for some \( q' \), and \( p'Rq' \).

The definition is confined to wired processes, and the bisimulation criteria is designed with regard to the behavior on communicating wires. Though it is a simplification on bisimulation checking compared to context bisimulation, as we will see, wired bisimilarity coincides with context bisimilarity on wired processes. Note up-to technique of the \( \approx_{W} \) also holds, we will mention this when necessary.

B.1.3 Wired factorization theorem

Here we derive the important theorem called factorization.

We use \( p\{ a \ r \} \) to abbreviate \( (p\{ (i!a)x.c!x.r+o!c!x.a!x.r) \})\{ a \} \) note the \( \{ a \} \) involves four names in it. And the precedence of \( \{ a \ r \} \) is the same as that of substitution. To some extent, \( \{ a \ r \} \) shares a similar idea with the implicit substitution in [II], and likewise wires behave like triggers to some extent.

Several properties on the abbreviation are given below. The first is on the substitution of \( \{ a \} \).

Lemma 39 (Substitution lemma on \( \{ a \} \)). Let \( p, q, r \) be (closed) Plain CHOCS processes without choice operator. Then the following holds:

\[
p[q/x]\{ a \ r \} \approx_{C_1} p\{ a \ r \}[q\{ a \ r \}/x]
\]

Proof. This is an important lemma serving as the basis for Lemma 40. The proof of it, however, is not so trivial and need some long step-by-step derivation. The framework is quite similar to the Lemma (4.1) in [II], whose proof is given in the appendix B of [II].

Below is another lemma on the distribution of \( \{ a \} \).

Lemma 40 (Distributivity on \( \{ a \} \)). Let \( p, q, q', r, r' \) be (closed) Plain CHOCS processes without choice operator, and suppose the local names related to \( \{ a \} \) occur only in output subjects. Then it holds that:

1. \( \text{nil}\{ a \ r \} \approx_{C_1} \text{nil} \);
2. \( (a!p')\{ a \ r \} \approx_{C_1} (i!a!c!x.x.(p|p'[p'/x])+o!c!x.x.(p|p'[p'/x]))\{ a \ r \} \), \( a \) appears in \( \{ a \ r \} \) and \( r \) is something of the form \( a(x).r' \);
3. \( (\tau.p)\{ a \ r \} \approx_{C_1} \tau.p\{ a \ r \} \);
4. \( (p|p')\{ a \ r \} \approx_{C_1} p\{ a \ r \}|p'\{ a \ r \} \);
5. \( (p\backslash b)\{ a \ r \} \approx_{C_1} (p\{ a \ r \})\{ b \} \) if \( b \) does not appear in \( fn(r) \) and \( \{ a \ r \} \) related names;
6. \( (lp)\{ a \ r \} \approx_{C_1} (p\{ a \ r \}) \).

Proof. Most of the properties are straightforward, with the help of Lemma 39.

Next we give the lemmas necessary for the wired factorization theorem. The following lemma is immediately provable.

Lemma 41. It holds that \( p \approx_{C_1} \tau.p \), for all Plain CHOCS processes without choice operator.
The following lemma says that instead of passing a process, we can replace it with a wire, the effect of which is adding an extra $\tau$ action before the replaced process.

**Lemma 42.**

$$p[\tau.r/x] \approx_{Ct} (p[a-chan/x])\{\diamond a \ r\}$$

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

**Proof.** Proof by induction on the structure of $p$. The basic cases are $\mathtt{nil}$ and $p = y$, which are immediately provable, with the possible help of Lemma 40. The rest of the proof is a relatively routine induction analysis. We take the parallel composition as a typical one and omit the rest.

$$(p[p'])[\tau.r/x] = (p[\tau.r/x])((p'[\tau.r/x])) \quad \text{(by substitution definition)}$$

$$\approx_{Ct} (p[a-chan/x])((p'[a-chan/x])\{\diamond a \ r\}) \quad \text{(by induction hypothesis)}$$

$$\approx_{Ct} ((p[a-chan/x])((p'[a-chan/x])\{\diamond a \ r\}) \quad \text{(by Lemma 40)}$$

$$= (p[p'])[a-chan/x]\{\diamond a \ r\} \quad \text{(by substitution definition)} \quad \square$$

Now we arrive at the wired factorization theorem, that is,

**Theorem 43** (Wired factorization theorem).

$$p[r/x] \approx_{Ct} (p[a-chan/x])\{\diamond a \ r\}$$

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

**Proof.** A simple deduction can prove this important theorem.

$$(p[a-chan/x])\{\diamond a \ r\} \approx_{Ct} p[r/x] \quad \text{(by Lemma 41 and congruence of $\approx_{Ct}$)}$$

Now we come to the critical theorem of the wire technique, that is, the coincidence between context bisimilarity and wired bisimilarity on wired processes. We first give the congruence property of wired bisimilarity.

**Lemma 44.** Suppose $p, q, r$ are Plain CHOCS wired (closed) processes, and $a$ is a name. If $p \approx_{Wr} q$, then:

1. $p\backslash a \approx_{Wr} q\backslash a$;
2. $p|r \approx_{Wr} q|r$.

**Proof.** We need to prove each of them by the definition of wired bisimulation. Since they are similar to some extent and routine, we only examine the second one. To prove it, we define the following relation on processes:

$$R \triangleq \{(p_1|r)\backslash B, (p_2|r)\backslash B) \mid p_1, p_2, r \text{ are wired (closed) processes, }$$

$$B \text{ are set of names, and } p_1 \approx_{Wr} p_2 \}.$$}

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

Suppose $((p_1|r)\backslash B_1, (p_2|r)\backslash B_2) \in R$, and $(p_1|r)\backslash B_1 \overset{\alpha}{\rightarrow} q_1$. The work can be expanded on the analysis of the action $\alpha$ and which process caused it. Most of them are routine and the difficult one
is when \( \alpha = \tau \), where \( p_1 \) makes an output action. And we simply analyze this case. As mentioned, we know that

\[
p_1 \xrightarrow{a!_B(m-chan)} p'_1 \setminus B_2, \quad r \xrightarrow{a?_x} r',
\]

for some fresh \( m \), and

\[
q_1 \equiv (p'_1 | r'[m-chan/x]) \setminus B_1 \cup B_2.
\]

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

\[
p_2 \xrightarrow{a!_B(m-chan)} p'_2 \setminus B_2, \quad \text{but } r \xrightarrow{a?_x} r',
\]

so we have

\[
(p_2|r) \setminus B_1 \xrightarrow{\alpha} (p'_2 | r'[m-chan/x]) \setminus B_1 \cup B_2 \equiv q_2.
\]

Hence we obtain that \( q_1 R q_2 \).

We prove two lemmas comprising the wanted theorem. One direction is stated in the following lemma.

**Lemma 45.** Suppose \( p, q \) are closed wired processes. If \( p \approx_{Ct} q \), then \( p \approx_{Wr} q \).

**Proof.** Define the following relation:

\[
R \triangleq \{(p_1,p_2) \mid p_1, p_2 \in WP \text{ and are closed}; p_1 \approx_{Ct} p_2 \}.
\]

We show \( R \) is a wired bisimulation.

Suppose \( (p_1,p_2) \in R \), and \( p_1 \xrightarrow{\alpha} p'_1 \). There are several cases to analyze in terms of wired bisimulation, but the most non-trivial case is when \( \alpha \equiv a!(m-chan) \), so we simply consider this case. Now we know that

\[
p_1 \xrightarrow{a!_B(m-chan)} p'_1,
\]

since \( p_1 \approx_{Ct} p_2 \), we have

\[
p_2 \xrightarrow{a!_B(m-chan)} p'_2,
\]

and for every \( E(x) \) with \( fn(E(x)) \cap fn(m-chan) = \emptyset \),

\[
(p'_1 | E[m-chan]) \setminus B \approx_{Ct} (q'_2 | E[m-chan]) \setminus B,
\]

where \( B = \{m, i, o, c\} \).

To fulfill the simulating step, we have to prove that \( p'_1 \approx_{Ct} p'_2 \). Now that we know the basic fact that \( E \) is any arbitrary process with process variable \( x \) in it, we can flexibly choose the form of \( E \) to make the simulation closed. For example, we can choose \( nil \), then we get immediately that \( (nil | p'_1) \setminus B \approx_{Ct} (nil | p'_2) \setminus B \), that is \( p'_1 \approx_{Ct} p'_2 \). Noting that the names in \( B \) should be chosen not to cause any conflict (they are fresh), we can also choose \( E \) to be just \( x \). Now we have

\[
(m-chan | p'_1) \setminus B \approx_{Ct} (m-chan | p'_2) \setminus B.
\]

By a simple bisimulation proof we can obtain the desired result that \( p'_1 \approx_{Ct} p'_2 \), because the action simulation must be performed between \( p_1 \) and \( p_2 \). We leave out the detailed construction.

**Remark.** An observation is that we can choose \( E \) to be empty, so the proof is straightforward. This is somewhat intuitive in that the context bisimulation is stronger than the wired bisimulation. However, we can also opt not to do this because usually the receiving environment of the sent process is not empty. But in this way the proof can encounter some technical difficulty. In detail, the factorization theorem cannot be applied, so the simulating step will break into a complicated situation.

The other direction is as the following lemma stated.

**Lemma 46.** Suppose \( p, q \) are closed wired processes. If \( p \approx_{Wr} q \), then \( p \approx_{Ct} q \).
Proof. Let
\[ \mathcal{R} \triangleq \{ (p, q) \mid p, q \in WP_r \text{ and are closed; } p \approx_{Wr} q \} \].

We show that \( \mathcal{R} \) is a context bisimulation up-to \( \approx_{Cl} \), which is sufficient since that will give us \( \mathcal{R} \subseteq \approx_{Cl} \).

Suppose \((p, q) \in \mathcal{R}\), and \( p \xrightarrow{\alpha} p' \). There are several cases to analyze according to the definition of context bisimulation up-to \( \approx_{Cl} \), and the most non-trivial case is when \( \alpha \equiv a?x \), so we only consider this case. We know that
\[ p \xrightarrow{a?x} p', \]
since \( p \approx_{Wr} q \), there exists \( q' \) such that
\[ q \xrightarrow{a?x} q', \]
and for \( B = \{ m, i, o, c \} \) with fresh names in it,
\[ p'[m-channel/x] \approx_{Wr} q'[m-channel/x]. \]

To match the action of \( p \), we have to prove that for every closed Plain CHOCS process (without choice operator) \( r \), we have
\[ p'[r/x] \approx_{Cl} \mathcal{R} \approx_{Cl} q'[r/x]. \]

We make the following reasoning:
\[ \approx_{Cl} \xrightarrow{p'[r/x]} p'[m-channel/x]\{\diamond_m \ r\} \quad \text{(by Theorem 43)} \]
\[ \Delta \]
\[ p''. \]

Similarly, we have:
\[ \approx_{Cl} \xrightarrow{q'[r/x]} q'[m-channel/x]\{\diamond_m \ r\} \quad \text{(by Theorem 43)} \]
\[ \Delta \]
\[ q''. \]

Now by Lemma 44, we know that \( \approx_{Wr} \) is closed under restriction and parallel composition, so we can get from \( p'[m-channel/x] \approx_{Wr} q'[m-channel/x] \) that \( p'' \approx_{Wr} q'' \), which completes the simulating step. \( \square \)

**Theorem 47.** \( \approx_{Cl} \) coincides with \( \approx_{Wr} \) on all wired processes free of choice operator.

**Proof.** By the two lemmas above: Lemma 45 and Lemma 46, we can attain this coincidence. \( \square \)