Available at
www.ElsevierComputerScience.com
POWERED BY SCIENCE DIRECT®

Electronic Notes in Theoretical Computer Science

ELSEVIER

Electronic Notes in Theoretical Computer Science 91 (2004) 4–20
www.elsevier.com/locate/entcs

Checking Strong Open Congruence in χ-Calculus [1]

Taolue Chen [2]

State Key Laboratory of Novel Software Technology
Nanjing University, Nanjing, P.R.China

Jingyang Zhou [3]

State Key Laboratory of Novel Software Technology
Nanjing University, Nanjing, P.R.China

Tingting Han [4]

State Key Laboratory of Novel Software Technology
Nanjing University, Nanjing, P.R.China

Jian Lu[5]

State Key Laboratory of Novel Software Technology
Nanjing University, Nanjing, P.R.China

Abstract

The χ-calculus is an important evolution for mobile process calculi. Open congruence is widely studied in χ-calculus. However, there still lacks an algorithm for checking this bisimulation relation. In this paper, the symbolic technique is applied to the research of χ-calculus and an efficient characterization for strong open congruence which does not involve quantification over substitutions is given. Based on it, an algorithm, which instantiates bound names 'on-the-fly', is developed to check strong open congruence for finite control processes.

Keywords: Mobile processes, Chi-calculus, Open congruence, Symbolic bisimulation, Algorithm.

doi:10.1016/j.entcs.2003.12.003

1 Introduction

For more than ten years, various calculi of mobile processes, notably the π-calculus [13], have been the focus of research in concurrency theory. These calculi are distinguished from the traditional process calculi like CCS [12] in that they allow processes to exchange specific values-channel names, thus are capable of dealing with processes whose communication structures can change during their evolution. The ability of dynamic creation of communication links lies at the heart of mobility.

Recently, several publications have focused on many variants of the π-calculus, such as asynchronous π-calculus [8], the πI-calculus [17]. Apart from the π-calculus more than any of the above variants, the χ-calculus [2] introduced by Fu, is an important evolution for mobile process calculi. The χ-calculus is introduced with two motivations in mind [2]. One is to remove the ad hoc nature of prefix operation in π-calculus by having a uniform treatment of names, thus arriving at a conceptually simpler language. The other is to materialize a communication-as-cut-elimination viewpoint, therefore taking up a proof theoretical approach to concurrency theory, an approach that has been proved very fruitful in the functional world. The difference between π-calculus and χ-calculus lies mainly in the way communications happen. The former adopts the familiar value-passing mechanism whereas the latter takes an information exchange or information update viewpoint.

In the research of algebra theory for mobile process, many studies focus on the bisimulation equivalence between processes, which is the most influential one in process algebra. To carry out a systematic study of bisimilarities for χ-calculus, Fu introduces the notion of L-bisimilarity [3], and shows that the set of L-bisimilarities forms a four-element lattice. It can also be shown that the well-known bisimilarity, that is, the open bisimilarity and barbed bisimilarity are respectively the bottom and the top element of the bisimulaiton lattice. We refer reader elsewhere [3] for more details. In [3], all of these congruences are investigated in weak case. However, in strong case, all of the four congruence coincide, i.e. the strong version bisimulation lattice collapses to only one element, in this sense, the open congruence is the only sensible congruence in the research of χ-calculus when we only deal with strong case.

For process algebra, the algorithm which is used to check one process is

[1] Supported by NNSFC (60273034, 60233010), 863 Program (2001AA113110, 2002AA116010), 973 Program of China (2002CB312002), JSFC (BK2002203, BK2002409)
[2] Email: ctl@ics.nju.edu.cn
[3] Email: jingyang@nju.edu.cn
[4] Email: hantt@ics.nju.edu.cn
[5] Email: lj@nju.edu.cn

bisimular to another is important, which is key to make one prototypical process description language be used in practice. In CCS and π-calculus, people have designed all sorts of such algorithms. Generally speaking, all available algorithms can be classified into two types. One is partition refinement algorithm [1][16], the main advantage of this kind of algorithm is that it can be used to obtain a minimal realization of a process P, i.e. a process which has the minimum number of states and transitions among all those bisimilar with P, however, it involves the computation of finite transition graph, which is a rather complex task and is inefficient in space; the other is 'on the fly' algorithm [19][10], in which the state spaces of processes being compared are created at the same time as the candidate bisimulation relation, so in general, it has a low space complexity, although it can not be used to give the minimal realization of process.

Since χ-calculus is an important mobile process calculi, it is necessary to study bisimulation verification problem for it. However, to the author's knowledge, this aspect of work has not been reported. The main work of this paper is to present an algorithm for the verification of open congruence in χ-calculus. Since it is well known that the full π-calculus is not decidable and it is easy to show this result also valid in χ-calculus, in this paper, we focus on the so called finite control [1] χ-calculus, that is, the process expression disallows parallel composition $|$ to appear in the bodies of recursive definitions. By confining the process expression to *finite control* processes, which is the syntactic counterpart of CCS finite state processes, we get a decidable equivalence problem. To get our destination, we first introduce an alternative characterization of the the open congruence in symbolic framework [7][18], thus we can avoid the universal quantification over substitution in the original definition which may represent a rather heavy requirement in practice. Based on it, an efficient algorithm is presented since it is enough to instantiate the bound name of the action with a single fresh name. Note that in this paper, the mismatch operator is not included and as usual, we only focus on strong open congruence.

The rest of the paper is organized as follows: The χ-calculus is reviewed in the following section. In Section 3, the symbolic characterization for open congruence is introduced with a proof to show that it captures (concrete) open congruence. The 'on the fly' algorithm is presented in Section 4. The paper is concluded with Section 5 where related work is also discussed.

2 χ-calculus and Open congruence

In this section, we will review some background material for χ-calculus, we refer the reader elsewhere [4] for more details. We will write $\mathcal{C}$ for the set of χ-processes defined by the following grammar:

$$P := 0 \mid \alpha x.P \mid P|P \mid (x)P \mid [x = y]P \mid P + P \mid A(y_1, \ldots, y_n)$$

where $\alpha \in \mathcal{N} \cup \bar{\mathcal{N}}$. Here $\mathcal{N}$ is the set of names ranged over by small case letters. The set $\{\bar{x}|x \in \mathcal{N}\}$ of co-names is denoted by $\bar{\mathcal{N}}$. The name x in $(x)P$ is bound. A name is free in P if it is not bound in P. The free names, the bound names and names of P, as well as the notations *fn(P)*,*bn(P)* and *n(P)*, are used in their standard meanings. In sequel we will use the functions *fn*(-),*bn*(-) and *n*(-) without explanation. We write $\bar{\alpha}$ for $\bar{a}$ if $\alpha = a$ and for a if $\alpha = \bar{a}$. Bound names induce the notion of α-equivalence as usual. In the sequel we will identify processes or actions which differ only on the bound names, therefore the symbol $\equiv$ means syntactical equality and α-equivalence. Furthermore whenever we introduce a binary relation on terms we will always assume it is closed w.r.t. $\equiv$.

The operational semantics is defined by the following labelled transition system:

Sequentialization:

$$\frac{}{\alpha x.P \xrightarrow{\alpha x} P} Sqn$$

Composition:

$$\frac{P \xrightarrow{\mu} P' \; bn(\mu) \cap fn(Q) = \emptyset}{P|Q \xrightarrow{\mu} P'|Q} Cmp_0, \quad \frac{P \xrightarrow{y/x} P'}{P|Q \xrightarrow{y/x} P'|Q\{y/x\}} Cmp_1$$

Communication:

$$\frac{P \xrightarrow{\alpha(x)} P' \quad Q \xrightarrow{\bar{\alpha}y} Q'}{P|Q \xrightarrow{\tau} P'\{y/x\}|Q'} Cmm_0, \quad \frac{P \xrightarrow{\alpha(x)} P' \quad Q \xrightarrow{\bar{\alpha}(x)} Q'}{P|Q \xrightarrow{\tau} (x)(P'|Q')} Cmm_1$$

$$\frac{P \xrightarrow{\alpha x} P' \quad Q \xrightarrow{\bar{\alpha}y} Q' \quad x \neq y}{P|Q \xrightarrow{y/x} P'\{y/x\}|Q'\{y/x\}} Cmm_2, \quad \frac{P \xrightarrow{\alpha x} P' \quad Q \xrightarrow{\bar{\alpha}x} Q'}{P|Q \xrightarrow{\tau} (P'|Q')} Cmm_3$$

Restriction:

$$\frac{P \xrightarrow{\lambda} P' \quad x \notin n(\lambda)}{(x)P \xrightarrow{\lambda} (x)P'} Loc_0, \quad \frac{P \xrightarrow{\alpha x} P' \quad x \notin \{\alpha, \bar{\alpha}\}}{(x)P \xrightarrow{\alpha(x)} P'} Loc_1, \quad \frac{P \xrightarrow{y/x} P'}{(x)P \xrightarrow{\tau} P'} Loc_2,$$

Condition:

$$\frac{P \xrightarrow{\lambda} P'}{[x = x]P \xrightarrow{\lambda} P'} Match$$

Summation:

$$\frac{P \xrightarrow{\lambda} P'}{P + Q \xrightarrow{\lambda} P'} Sum$$

Identity:

$$\frac{P\{y_1, \cdots, y_n/x_1, \cdots, x_n\} \xrightarrow{\lambda} P'}{A(y_1, \cdots, y_n) \xrightarrow{\lambda} P'} A(x_1 \cdots x_n) \stackrel{def}{=} P$$

We have omitted all the symmetric rules. In the above rules the letter μ ranges over the set $\{\alpha(x), \alpha x | \alpha \in \mathcal{N} \cup \bar{\mathcal{N}},\ x \in \mathcal{N}\} \cup \{\tau\}$ of non-update actions and the letter λ over the set $\{\alpha(x), \alpha x, y/x | \alpha \in \mathcal{N} \cup \bar{\mathcal{N}},\ x \in \mathcal{N}\} \cup \{\tau\}$ of all actions. In αx and y/x all names are free. In $\alpha(x)$, the name x is bounded whereas the other name is free.

The process $P\{y/x\}$ in the above labelled transitional system is obtained by substituting y for x throughout P. A substitution $\{y_1/x_1, \cdots y_n/x_n\}$ is a function from $\mathcal{N}$ to $\mathcal{N}$ that maps x_i onto y_i for $i \in \{1, \cdots, n\}$ and x onto itself for $x \notin \{x_1, \cdots, x_n\}$. Substitutions are usually denoted by σ, ρ, etc. The empty substitution, that is the identity function on $\mathcal{N}$, is written as $\{\}$. The result of applying σ to P is denoted by $P\sigma$. In the below, by α-conversion it is assumed that a substitution σ acts as an identity on the bound names of the process and keeps the separation between bound and free names. We follow this convention in the below and will use it implicitly in the proof.

The following technical lemma [4] states the properties of (concrete) labelled transitions.

Lemma 2.1 *The following properties hold:*

(i) If $P \xrightarrow{\mu} P'$ *then* $P\sigma \xrightarrow{\mu} P'\sigma$.

(ii) If $P \xrightarrow{y/x} P'$ *and* $x\sigma \neq y\sigma$ *then* $P\sigma \xrightarrow{y\sigma/x\sigma} P'\sigma\{y\sigma/x\sigma\}$.

(iii) If $P \xrightarrow{y/x} P'$ *and* $x\sigma = y\sigma$ *then* $P\sigma \xrightarrow{\tau} P'\sigma$.

As done previously [18], we shall use M, N, L to stand for finite (and possibly empty) *match sequences*. A match sequence defines an equivalence relation on names. Given a match sequence M and the equivalence relation R_M associated to it, we denote by σ_M a special substitution which selects a representative out of each equivalence class of R_M and maps all names in the same class to their representative. We write $M \Rightarrow N$ if M implies N, i.e. whenever the tests in M are true, then also the tests in N are true. Similarly, we write $M \Leftrightarrow N$ if both $M \Rightarrow N$ and $N \Rightarrow M$ hold. We say that a substitution σ satisfies a match sequence M if for each x, y, it holds that $M \Rightarrow [x = y]$ implies $\sigma(x) = \sigma(y)$, that is, the condition $M\sigma$ is true.

We extend the notation $M \Rightarrow N$ to $\sigma \Rightarrow \rho$ if σ equates more than ρ, i.e. $\rho(x) = \rho(y)$ implies $\sigma(x) = \sigma(y)$.

The relations between match sequence and substitution are revealed by the following lemma [18].

Lemma 2.2 *The following properties hold:*

(i) If $M \Rightarrow N$, then $M\sigma \Rightarrow N\sigma$.

(ii) If $M \Rightarrow N$, then $\sigma_M \Rightarrow \sigma_N$.

(iii) If $\sigma \Rightarrow \sigma'$, then ρ exists s.t. $\sigma = \sigma'\rho$.

In what follows, some notations need be fixed. Let fa denote the set $\{\alpha x | \alpha, x \in \mathcal{N} \cup \bar{\mathcal{N}}\}$ of free actions, ba the set $\{\alpha(x) | \alpha, x \in \mathcal{N} \cup \bar{\mathcal{N}}\}$ of bound actions and u the set $\{y/x | x, y \in \mathcal{N} \cup \bar{\mathcal{N}}\}$ of updates and we let δ, λ range over the set $\{\alpha(x), \alpha x, y/x | \alpha \in \mathcal{N} \cup \bar{\mathcal{N}}, x, y \in \mathcal{N}\} \cup \{\tau\}$.

The strong open congruence, given in this section, is the strong version of open congruence studied earlier [3][4], which adapts Sangiorgi's definition for π-calculus [18] to χ-calculus.

Definition 2.3 Let $\mathcal{R}$ be a symmetric relation on $\mathcal{C}$, The relation $\mathcal{R}$ is called an open congruence relation if whenever $P\mathcal{R}Q$, then for any substitution σ, it holds that if $P\sigma \xrightarrow{\delta} P'$, then some Q' exists, such that $Q\sigma \xrightarrow{\delta} Q'$ and $P'\mathcal{R}Q'$. The open bisimilarity $\sim$ is the largest open congruence.

It is easy to show that $\sim$ is indeed a congruence relation. Note that this definition can be rehearsed as follows:

Definition 2.4 Let $\mathcal{R}$ be a symmetric relation on $\mathcal{C}$, The relation $\mathcal{R}$ is called an open congruence if it is closed under substitution and whenever $P\mathcal{R}Q$ and $P \xrightarrow{\delta} P'$, then Q' exists s.t. $Q \xrightarrow{\delta} Q'$ and $P'\mathcal{R}Q'$.

The form of this definition has been considered elsewhere [18] and will be used in the proof of Section 3.

3 Symbolic Characterization

As we state in Section 1, one immediate difficulty to check open congruence is that the definition of Section 2 involves quantification over substitutions. In this section, we show a more efficient method to check process bisimilaries, which is based on the well known *symbolic transition system*. A symbolic transition is of the form $P \overset{M,\delta}{\mapsto} P'$, where, M is a match sequence and δ is an action. Intuitively, M represents the environments under which the action δ can actually be fired from P. Comparing with it, the transitions defined in

Section 2 are *concrete* in the sense that they can always be fired regardless of the context in which terms are placed. Base on it, we define a symbolic open congruence following Sangiorgi [18].

The symbolic transitional semantics of the χ-calculus is given as follows. For notational convenience we write MN for the union of M and N. Also the symmetric rules for Sum and Par have been omitted.

Sequentialization:

$$\frac{}{\alpha x.P \overset{\emptyset,\alpha x}{\mapsto} P} Sqn$$

Composition:

$$\frac{P \overset{\mu}{\mapsto} P' \; bn(\mu) \cap fn(Q) = \emptyset}{P|Q \overset{\mu}{\mapsto} P'|Q} Cmp_0, \quad \frac{P \overset{y/x}{\mapsto} P'}{P|Q \overset{y/x}{\mapsto} P'|Q\{y/x\}} Cmp_1$$

Communication:

$$\frac{P \overset{M,a(x)}{\mapsto} P' \quad Q \overset{N,\bar{b}y}{\mapsto} Q'}{P|Q \overset{L,\tau}{\mapsto} P'\{y/x\}|Q'} L = \begin{cases} MN[a=b] & \text{if } a \neq b \\ MN & \text{if } a = b \end{cases} Cmm_0$$

$$\frac{P \overset{M,a(x)}{\mapsto} P' \quad Q \overset{N,\bar{b}(x)}{\mapsto} Q'}{P|Q \overset{L,\tau}{\mapsto} (x)(P'|Q')} L = \begin{cases} MN[a=b] & \text{if } a \neq b \\ MN & \text{if } a = b \end{cases} Cmm_1$$

$$\frac{P \overset{M,ax}{\mapsto} P' \quad Q \overset{N,\bar{b}y}{\mapsto} Q' \quad x \neq y}{P|Q \overset{L,y/x}{\mapsto} P'\{y/x\}|Q'\{y/x\}} L = \begin{cases} MN[a=b] & \text{if } a \neq b \\ MN & \text{if } a = b \end{cases} Cmm_2$$

$$\frac{P \overset{M,ax}{\mapsto} P' \quad Q \overset{N,\bar{b}x}{\mapsto} Q'}{P|Q \overset{L,\tau}{\mapsto} P'|Q'} L = \begin{cases} MN[a=b] & \text{if } a \neq b \\ MN & \text{if } a = b \end{cases} Cmm_3$$

Restriction:

$$\frac{P \overset{M,\lambda}{\mapsto} P' \quad x \notin n(M,\lambda)}{(x)P \overset{M,\lambda}{\mapsto} (x)P'} Loc_0, \quad \frac{P \overset{M,\alpha x}{\mapsto} P' \quad x \notin n(M) \cup \{\alpha, \bar{\alpha}\}}{(x)P \overset{M,\alpha(x)}{\mapsto} P'} Loc_1$$

$$\frac{P \overset{M,y/x}{\mapsto} P' \quad x \notin n(M)}{(x)P \overset{M,\tau}{\mapsto} P'} Loc_2,$$

Condition:

$$\frac{P \overset{M,\lambda}{\mapsto} P'}{[x=y]P \overset{L,\lambda}{\mapsto} P'} L = \begin{cases} M[x=y] & \text{if } x \neq y \\ M & \text{if } x = y \end{cases} Match$$

Summation:

$$\frac{P \overset{M,\lambda}{\mapsto} P'}{P + Q \overset{M,\lambda}{\mapsto} P'} Sum$$

Identity:

$$\frac{P\{y_1, \cdots, y_n/x_1, \cdots, x_n\} \overset{M,\lambda}{\mapsto} P'}{A(y_1, \cdots, y_n) \overset{M,\lambda}{\mapsto} P'} A(x_1 \cdots x_n) \overset{def}{=} P$$

The following technical lemma shows some properties of symbolic transition.

Lemma 3.1 *The following properties hold:*

(i) *If* $P \overset{M,\delta}{\mapsto} Q$, *then* $n(M) \cup fn(\delta) \subseteq fn(P)$.

(ii) *If* $P \overset{M,\delta}{\mapsto} Q$, *then the following properties holds:*
- *If* $\delta \notin u$, *then* $P\sigma \overset{M\sigma,\delta\sigma}{\mapsto} Q\sigma$.
- *If* $\delta = y/x$ *and* $y\sigma = x\sigma$, *then* $P\sigma \overset{M\sigma,\tau}{\mapsto} Q\sigma$.
- *If* $\delta = y/x$ *and* $y\sigma \neq x\sigma$, *then* $P\sigma \overset{M\sigma,y\sigma/x\sigma}{\mapsto} Q\sigma\{y\sigma/x\sigma\}$.

(iii) *If* $P\sigma \overset{M',\delta'}{\mapsto} Q'$, *then the following properties holds:*
- *If* $\delta' \in ba \cup fa$, *then* $P \overset{M,\delta}{\mapsto} Q$ *with* $M' \Leftrightarrow M\sigma, \delta' \equiv \delta\sigma$ *and* $Q' \equiv Q\sigma$.
- *If* $\delta' = \tau$, *then one of the following properties holds:*
  $P \overset{M,\tau}{\mapsto} Q$ *s.t.* $M' \Leftrightarrow M\sigma$ *and* $Q' \equiv Q\sigma$.
  $P \overset{M,y/x}{\mapsto} Q$ *s.t.* $M' \Leftrightarrow M\sigma, x\sigma = y\sigma$ *and* $Q' \equiv Q\sigma$.
- *if* $\delta' = y'/x'$, *then* $P \overset{M,y/x}{\mapsto} Q$ *with* $M' \Leftrightarrow M\sigma, y' = y\sigma$, $x' = x\sigma$ *and* $Q' \equiv Q\sigma\{y'/x'\}$.

Proof. By transition induction. □

The following two technical lemmas relate concrete and symbolic transitions.

Lemma 3.2 *If* $P\sigma_M \xrightarrow{\delta} P''$, *then the following properties holds:*

- *If* $\delta = \tau$, *then one of the following properties holds:*
  - $P \overset{N,y/x}{\mapsto} P'$ *with* $M \Rightarrow N[x = y]$ *and* $P'' = P'\sigma_M$.
  - $P \overset{N,\tau}{\mapsto} P'$ *with* $M \Rightarrow N$ *and* $P'' = P'\sigma_M$.
- *if* $\delta \neq \tau$, *then* $P \overset{N,\lambda}{\mapsto} P'$ *with* $M \Rightarrow N$, $\delta = \lambda\sigma$ *and* $P'' = P'\sigma_M$.

Proof. By transition induction. We only check some typical cases.

- $(P_1|P_2)\sigma_M \xrightarrow{\tau} (P_1''|P_2'')$ is derived from $P_1\sigma_M \xrightarrow{az} P_1''$ and $P_2\sigma_M \xrightarrow{\bar{a}z} P_2''$. By

the inductive assumption, we get

$$P_1 \overset{N_1,bx}{\mapsto} P_1' \quad P_2 \overset{N_2,\bar{c}y}{\mapsto} P_2'$$

with

$$M \Rightarrow N_1, az \equiv (bx)\sigma_M, P_1'' \equiv P_1'\sigma_M$$
$$M \Rightarrow N_2, az \equiv (cy)\sigma_M, P_2'' \equiv P_2'\sigma_M$$

Now, we can infer(suppose $b \neq c$, the converse case is similar)

$$P_1|P_2 \overset{N_1N_2[b=c],y/x}{\mapsto} P_1'|P_2'$$

with $(P_1'|P_2')\sigma_M \equiv P_1''|P_2''$ and $M \Rightarrow N_1N_2[b=c][x=y]$ since $x\sigma_M = y\sigma_M$.

- $((x)P)\sigma_M \overset{\tau}{\rightarrow} P''$ is derived from $P\sigma_M \overset{y/x}{\rightarrow} P''$(Note that by the convention of Section 2, $((x)P)\sigma_M = (x)P\sigma_M$ and $x \notin n(\sigma)$)). By inductive assumption, we get $P \overset{N,z/x}{\mapsto} P'$, with $M \Rightarrow N$, $z\sigma_M = y$ and $P'' \equiv P'\sigma_M$. Now, we infer $(x)P \overset{N,\tau}{\mapsto} P'$, since by Lemma 3.1(i), $x \notin n(N)$.
- $(P_1|P_2)\sigma_M \overset{y/x}{\rightarrow} (P_1''\{y/x\}|P_2''\{y/x\})$ is derived from $P_1\sigma_M \overset{ax}{\rightarrow} P_1''$ and $P_2\sigma_M \overset{\bar{a}y}{\rightarrow} P_2''$. By the inductive assumption, we get

$$P_1 \overset{N_1,bx'}{\mapsto} P_1' \quad P_2 \overset{N_2,\bar{c}y'}{\mapsto} P_2'$$

with

$$M \Rightarrow N_1, ax \equiv (bx')\sigma_M, P_1'' \equiv P_1'\sigma_M$$
$$M \Rightarrow N_2, ay \equiv (cy')\sigma_M, P_2'' \equiv P_2'\sigma_M$$

Now, we can infer(suppose $b \neq c$, the converse case is similar)

$$P_1|P_2 \overset{N_1N_2[b=c],y'/x'}{\mapsto} P_1'\{y'/x'\}|P_2'\{y'/x'\}$$

with

$$\begin{aligned}(P_1'\{y'/x'\}|P_2'\{y'/x'\})\sigma_M &\equiv (P_1'\sigma_M\{\sigma_M(y')/\sigma_M(x')\}|P_2'\{\sigma_M(y')/\sigma_M(x')\}) \\ &\equiv P_1''\{y/x\}|P_2''\{y/x\}\end{aligned}$$

and $M \Rightarrow N_1N_2[b=c]$.

The other cases are similar, due to space restriction, we omit the details. □

Lemma 3.3 *If $P \overset{M,\delta}{\mapsto} P'$, then the following properties hold:*

- *If $\delta = y/x$ and $M \Rightarrow x = y$, then $P\sigma_M \overset{\tau}{\rightarrow} P'\sigma_M$.*
- *If $\delta \notin u$ or $\delta = y/x$ and $x\sigma_M \neq y\sigma_M$, then $P\sigma_M \overset{\delta\sigma_M}{\rightarrow} P'\sigma_M$.*

Proof. By transition induction. We only check some typical cases.

- Suppose $a \neq b$ and $(P_1|P_2) \overset{M_1M2[a=b],y/x}{\mapsto} (P_1'\{y/x\}|P_2'\{y/x\})$ is derived from $P_1 \overset{M_1,ax}{\mapsto} P_1'$ and $P_2 \overset{M_2,\bar{a}y}{\mapsto} P_2'$. We can decompose $\sigma_{M_1M2[a=b]}$ as $\sigma_{M_1\rho_1}$ and

$\sigma_{M_2\rho_2}$ for some ρ_1,ρ_2 by Lemma 2.2. From the inductive assumption, we get

$$P_1\sigma_{M_1} \overset{(ax)\sigma_{M_1}}{\rightarrow} P_1'\sigma_{M_1} \quad P_2\sigma_{M_1} \overset{(\bar{b}y)\sigma_{M_2}}{\rightarrow} P_2'\sigma_{M_2}$$

and moreover it can be inferred that

$$\frac{P_1\sigma_{M_1}\rho_1 \overset{(ax)\sigma_{M_1}\rho_1}{\rightarrow} P_1'\sigma_{M_1}\rho_1 \quad P_2\sigma_{M_2}\rho_2 \overset{(\bar{b}y)\sigma_{M_2}\rho_2}{\rightarrow} P_2'\sigma_{M_2}\rho_2}{(P_1|P_2)\sigma_{M_1M_2[a=b]} \overset{\tau}{\rightarrow} (P_1'|P_2')\sigma_{M_1M_2[a=b]}}$$

since by $M_1M_2[a=b] \Rightarrow x = y$ and Lemma 2.2, $x\sigma_{M_1\rho_1}\sigma_{M_2\rho_2} = y\sigma_{M_1\rho_1}\sigma_{M_2\rho_2}$ and substitution $\{y/x\}$ is subsumed by $\sigma_{M_1\rho_1}\sigma_{M_2\rho_2}$

- $[x=y]P \overset{M,\delta}{\mapsto} P'$ is derived from $P \overset{N,\delta}{\mapsto} P'$, we suppose $x \neq y$ and $\delta \notin u$, then $M = N[x=y]$. By Lemma 2.2, we can decompose σ_M as $\sigma_N\rho$ for some ρ. By inductive assumption, we get $P\sigma_N \overset{\delta\sigma_N}{\rightarrow} P'\sigma_N$, hence $([x=y]P)\sigma_M \overset{\delta\sigma_M}{\rightarrow} P'\sigma_M$.

The other cases are similar and can be proved without difficulty.

□

Now, we give the symbolic characterization for open congruence, which is called symbolic open congruence.

Definition 3.4 Let $\mathcal{R}$ be a symmetric relation on $\mathcal{C}$, The relation $\mathcal{R}$ is called a symbolic open congruence if $P\mathcal{R}Q$ implies:

(i) If $\delta \in fa \cup ba$ and $P \overset{M,\delta}{\mapsto} P'$, then N, λ, Q' exists,s.t. $Q \overset{N,\lambda}{\mapsto} Q'$, with $M \Rightarrow N$, $\delta\sigma_M = \lambda\sigma_M$ and $P'\sigma_M\mathcal{R}Q'\sigma_M$.

(ii) If $P \overset{M,\tau}{\mapsto} P'$, then one of the following properties holds:
- N, Q' exists s.t. $Q \overset{N,\tau}{\mapsto} Q'$ with $M \Rightarrow N$ and $P'\sigma_M\mathcal{R}Q'\sigma_M$.
- $N, y/x, Q'$ exists s.t. $Q \overset{N,y/x}{\mapsto} Q'$ with $M \Rightarrow N[x=y]$ and $P'\sigma_M\mathcal{R}Q'\sigma_M$.

(iii) If $P \overset{M,y/x}{\mapsto} P'$, then one of the following properties holds:
- N, Q' exists s.t. $Q \overset{N,\tau}{\mapsto} Q'$ with $M \Rightarrow N[x=y]$ and $P'\sigma_M\mathcal{R}Q'\sigma_M$.
- $N, y'/x', Q'$ exists s.t. $Q \overset{N,y'/x'}{\mapsto} Q'$ with $M \Rightarrow N$, $(y/x)\sigma_M = (y'/x')\sigma_M$ and $P'\sigma_M\mathcal{R}Q'\sigma_M$.
- $N, y'/x', Q'$ exists s.t. $Q \overset{N,y'/x'}{\mapsto} Q'$ with $M \Rightarrow N[x=y][x'=y']$ and $P'\sigma_M\mathcal{R}Q'\sigma_M$.

The symbolic open bisimilarity $\asymp$ is the largest symbolic open congruence.

Now, we embark into the proof that $\sim$ and $\asymp$ coincide.

Lemma 3.5 $\asymp$ *implies* $\sim$.

Proof. Let

$$R = \{(P,Q) | P \asymp Q\}$$

first, we prove that R is closed under substitution. Let $S = \{(P\sigma, Q\sigma) | P \asymp Q, \sigma \text{ arbitrary}\}$, we show S is a symbolic open congruence. Consider the action δ that $P\sigma$ can take, we have following cases:

- $\delta \in fa \cup ba$, then by Lemma 3.1, the transition can be written as $P\sigma \overset{M\sigma,\delta\sigma}{\mapsto} P'\sigma$ for some M, δ and P' s.t. $P \overset{M,\delta}{\mapsto} P'$. Since $P \asymp Q$, we have $Q \overset{N,\lambda}{\mapsto} Q'$ with $M \Rightarrow N$, $\delta\sigma_M \equiv \lambda\sigma_M$ and $P'\sigma_M \asymp Q'\sigma_M$. By Lemma 2.2, we have $M\sigma \Rightarrow N\sigma$, and following Sangiorgi's proof [18], it is not difficult to verify that $\sigma\sigma_{M\sigma} \Rightarrow \sigma_M$, so we have $\delta\sigma\sigma_{M\sigma} \equiv \lambda\sigma\sigma_{M\sigma}$ and similarly, $P'\sigma\sigma_{M\sigma}\mathcal{S}Q'\sigma\sigma_{M\sigma}$, hence $Q\sigma \overset{N\sigma,\lambda\sigma}{\mapsto} Q'\sigma$ gives the counterpart to the transition of $P\sigma$ by Lemma 3.1 and Definition 3.4. In this case, the proposition is correct.
- $\delta = \tau$, then the transition can be written as one of two subcases as follows:
  - $P\sigma \overset{M\sigma,\tau}{\mapsto} P'\sigma$ for some M and P' s.t. $P \overset{M,\tau}{\mapsto} P'$. By the definition of $\asymp$, there are two cases to verify. However, using the same argument as the first case, the two cases are easily to be checked.
  - $P\sigma \overset{M\sigma,\tau}{\mapsto} P'\sigma$ for some M and P' s.t. $P \overset{M,y/x}{\mapsto} P'$ and $x\sigma = y\sigma$. By the definition of $\asymp$, there are three cases need to be verified. Here we only check one as example, the other two are similar. Take $Q \overset{N,\tau}{\mapsto} Q'$ with $M \Rightarrow N[x = y]$ and $P'\sigma_M \asymp Q'\sigma_M$, then $Q\sigma \overset{N\sigma,\tau}{\mapsto} Q'\sigma$. Since $x\sigma = y\sigma$, we have $M\sigma \Rightarrow N\sigma$ and use the same argument in the first case, $P'\sigma\sigma_{M\sigma}\mathcal{S}Q'\sigma\sigma_{M\sigma}$, hence, the transition of $P\sigma$ can be matched.
- $\delta \in u$, then the transition can be written as $P\sigma \overset{M\sigma,y'/x'}{\mapsto} P'\sigma\{y'/x'\}$ for some M and P' s.t. $P \overset{M,y/x}{\mapsto} P'$ and $x\sigma = x'$, $y\sigma = y'$. By the definition of $\asymp$, there are also three cases need to be verified, here we chose the most difficult one. Take $Q \overset{N,y''/x''}{\mapsto} Q'$ with $M \Rightarrow N[x = y][x'' = y'']$ and $P'\sigma_M \asymp Q'\sigma_M$. Now, we have two subcases.
  - $x''\sigma = y''\sigma$, then $Q\sigma \overset{N\sigma,\tau}{\mapsto} Q'\sigma$. Note that $M\sigma \Rightarrow N[x' = y']$ and $P'\sigma\{y'/x'\}\sigma_{M\sigma} \equiv P'\sigma\sigma_{M\sigma}$, and by the same argument in the first case $P'\sigma\sigma_{M\sigma}\mathcal{S}Q'\sigma\sigma_{M\sigma}$;
  - $x''\sigma \neq y''\sigma$, then $Q\sigma \overset{N\sigma,y''\sigma/x''\sigma}{\mapsto} Q'\sigma\{y''\sigma/x''\sigma\}$, and we have $M\sigma \Rightarrow N\sigma[x' = y'][x''\sigma = y''\sigma]$ and $P'\sigma\{y'/x'\}\sigma_{M\sigma}\mathcal{S}Q'\sigma\{y'/x'\}\sigma_{M\sigma}$.

Now we check naked transition match in Definition 2.4, which is rather easy. Suppose $P \overset{\delta}{\to} P'$, then by Lemma 3.2, $P \overset{\emptyset,\delta}{\mapsto} P'$. Since $P \asymp Q$, by Definition 3.4, the only possible case for Q to match such transition is that $Q \overset{\emptyset,\delta}{\mapsto} Q'$, and $P' \asymp Q'$. By Lemma 3.3, $Q \overset{\delta}{\to} Q'$. Due to Definition 2.4, the proof is completed. □

Lemma 3.6 $\sim$ *implies* $\asymp$.

Proof. Let

$$R = \{(P, Q) | P \sim Q\}$$

We show that R is a symbolic congruence. Suppose $P \overset{M,\delta}{\mapsto} P'$. We check the following cases:

- $\delta = y/x$ and $M \Rightarrow x = y$, then by Lemma 3.3, $P\sigma_M \xrightarrow{\tau} P'\sigma_M$. By $P \sim Q$, we have $Q\sigma_M \xrightarrow{\tau} Q''$ with $P'\sigma_M \sim Q''$. By Lemma 3.2, there are two subcases, we only chose the more difficult one to verify. $Q \overset{N,y'/x'}{\mapsto} Q'$ with $M \Rightarrow N[x' = y']$ and $Q'' = Q'\sigma_M$, then we have $M \Rightarrow N[x = y][x' = y']$ and $P'\sigma_M \mathcal{R} Q'\sigma_M$.
- $\delta = y/x$ and $x\sigma_M \neq y\sigma_M$, then $P\sigma_M \xrightarrow{y'/x'} Q'\sigma_M$, where $y' = y\sigma_M$ and $x' = x\sigma_M$. By $P \sim Q$, we have $Q\sigma_M \xrightarrow{y'/x'} Q''$ with $P'\sigma_M \sim Q''$. By Lemma 3.2, N, Q' exists s.t. $Q \overset{N,y''/x''}{\mapsto} Q'$, with $M \Rightarrow N$, $y'/x' = (y''/x'')\sigma_M$ and $Q'' \equiv Q'\sigma_M$, thus $(y/x)\sigma_M = (y''/x'')\sigma_M$ and $P'\sigma_M \mathcal{R} Q'\sigma_M$.
- $\delta = \tau$, then $P\sigma_M \xrightarrow{\tau} P'\sigma$, by $P \sim Q$, we have $Q\sigma_M \xrightarrow{\tau} Q''$ with $P'\sigma_M \sim Q''$. By Lemma 3.2, there are two subcases, here we also check one of them. $Q \overset{N,y/x}{\mapsto} Q'$ with $M \Rightarrow N[x = y]$ and $P'' = P\sigma_M$, then $P'\sigma_M \mathcal{R} Q'\sigma_M$.

The other cases are easier. □

Combining the above two lemmas gives the main result of this section:

Theorem 3.7 *$P \sim Q$ iff $P \asymp Q$.*

Proof. By Lemma 3.5 and Lemma 3.6. □

4 On-the-fly Algorithm

The definition of symbolic open congruence does not involve the requirement for considering a universal quantification over substitution, thus paving the way for an efficient bisimulation checking algorithm.

The algorithm present in Fig.1 and Fig.2 is adapted from the 'on-the-fly' algorithm for value-passing processes [10] and π-calculus [11]. It combines the well-known on-the-fly bisimulation algorithm for standard labelled transition systems with the strategy of instantiating bound name with a single fresh name. As done previous [11], it assumes a countably infinite subset $\mathcal{SN} \subset \mathcal{N}$ which is totally ordered. The function $nextSN(P, Q)$ returns the smallest name in $\mathcal{SN}$ that does not appear in the set of free names at states P and Q.

The function $bisim(P, Q)$ starts with the initial pair (P, Q), and calls the core function, *match*, which tries to find the smallest bisimulation containing

$bisim(P,Q)$={
  $NotBisim$:=$\emptyset$;
  return $match(P,Q,\emptyset)\neq\emptyset$;
}

$match(P,Q,visited)$={
  if $(P,Q)\in visited$
   then return $visited$;
  else if $(P,Q)\in NotBisim$
        then return $\emptyset$;
       else {
        $PS_{\{\lambda|\lambda\in\{fa,u,\tau\}\}}=\{(P',M,\delta)|P\overset{M,\delta}{\mapsto}P'\}$;
        $QS_{\{\lambda|\lambda\in\{fa,u,\tau\}\}}=\{(Q',M,\delta)|Q\overset{M,\delta}{\mapsto}Q'\}$;
        $PS_{ba}=\{(P'\{z/x\},M,\alpha(z)|P\overset{M,\alpha(x)}{\mapsto}P',z=nextSN(P,Q),\alpha\in\mathcal{N}\cup\bar{\mathcal{N}}\}$;
        $QS_{ba}=\{(Q'\{z/x\},M,\alpha(z))|Q\overset{M,\alpha(x)}{\mapsto}Q',z=nextSN(P,Q),\alpha\in\mathcal{N}\cup\bar{\mathcal{N}}\}$;
        $PS=\bigcup_{\lambda\in\{ba,fa,u,\tau\}}PS_\lambda$;
        $QS=\bigcup_{\lambda\in\{ba,fa,u,\tau\}}QS_\lambda$;
        $temp$:=$close(PS,QS,visited\cup\{P,Q\})$;
        if $temp=\emptyset$
         then {
          $NotBisim:=NotBisim\cup\{(P,Q)\}$; **return** $\emptyset$;
         }
        else if $(temp:=close(QS,PS,temp))\neq\emptyset$
             then return $temp$;
            else {
             $NotBisim:=NotBisim\cup\{(P,Q)\}$; **return** $\emptyset$;
            }
       }
}

Fig. 1. The on-the-fly Algorithm (PartI)

```
close(PSet,QSet,visited)={
  if PSet = ∅
   then return visited;
  else if QSet = ∅
        then return ∅;
       else {
        hasvisit:=visited;
        for each (P, M, δ) ∈ PSet {
         if exists (Q, N, λ) ∈ QSet s.t.
          (i) δ ∈ fa ∪ ba ⇒ (M ⇒ N ∧ δσ_M = λσ_M)
          (ii) δ = τ ⇒ ((λ = τ ∧ M ⇒ N) ∨ (λ = y/x ∧ M ⇒
N[x = y]))
          (iii) δ = y/x ⇒ ((λ = τ ∧ M ⇒ N[x = y])
              ∨ (λ = y'/x' ∧ M ⇒ N ∧ (y/x)σ_M = (y'/x')σ_M)
              ∨ (λ = y'/x' ∧ M ⇒ N[x = y][x' = y']))
          (iv)(temp:=match(Pσ_M, Qσ_M, hasvisit)) ≠ ∅
          then hasvisit:=temp;
         else return ∅;
        }
        return hasvisit;
       }
}
```

Fig. 2. The on-the-fly Algorithm (PartII)

the pair if two processes in question(P and Q) are open congruent otherwise returns empty set by matching transitions from them.

The core function, *match*, performs a depth-first travel on the product of the two transition graphs which are never fully created, instead, they are generated together during the construction of the candidate bisimulation relation which equates them. The parameter *visited* is used to store pairs that are encountered before. If two states fail to match each other's transitions then they are not bisimilar and the pair is inserted into $NotBisim$, which is introduced to improve the efficiency of algorithm. Thus given two processes P and Q and a relation *visited*, function *match* first checks if (P, Q) has been already in the relation. If so, return the relation, otherwise if (P, Q) is in $NotBisim$, then return $\emptyset$, in converse, we produce the outgoing transitions and the next states set. Note that for bound action, a single fresh name is used to instantiate the bound name. Then the function *close* is called to match each other's deriva-

tives by the definition of symbolic open congruence which involves recursive calling of function *match* when applying the substitution σ_M to the derivatives and using the extended *visited* relations.

The pseudo code of the algorithm is described in more detail in Fig.1 and Fig.2. The correctness of the algorithm is not difficult to justify. First, we give the following lemma to ensure the termination of the algorithm.

Lemma 4.1 *For finite control process P, Q, bisim(P,Q) always terminates.*

Proof. Since we only consider the finite control process, that is, the process expression disallows parallel composition — to appear in the bodies of recursive definitions, and indeed it is built up by parallel composition of processes that do not contain parallel composition. So it is easy to see that the process space searched is finite and by parameter *visited*, the algorithm stores and checks if the current processes in question have been checked for each recursion. Thus the algorithm always terminates. □

The partial correctness can be get by the following lemmas.

Lemma 4.2 *If $\forall(P,Q) \in visited, P \asymp Q$ and $R = match(P, Q, visited) \neq \emptyset$, then for $\forall(P,Q) \in R, P \asymp Q$ holds.*

Proof. By induction on the number of recursions of function *match*, n.

- Initial case: for $n = 0$, trivial.
- Induction step. if $R = match(P, Q, visited)$ does not return immediately, then all derivatives of P have been matched by Q for open congruence, and each of the derivatives is by induction open congruent. So R is built up by the 'for' loop when matching the derivatives. By the definition of symbolic open congruence, provided *visited* is a set satisfied the condition, then for $\forall(P,Q) \in R, P \asymp Q$ also holds.

□

By the above lemma, Let $visited = \emptyset$, then we can get: if $R = match(P, Q, \emptyset) \neq \emptyset$, for $\forall(P,Q) \in R, P \asymp Q$.

The following lemma is easy to show when examining the code in Fig.1 and Fig.2.

Lemma 4.3 *If $R = match(P, Q, \emptyset) = \emptyset$, then $P \not\asymp Q$.*

Now, we have the following theorem:

Theorem 4.4 *For finite control process P and Q,$R = bisim(P, Q, \emptyset)$ always terminate and $R \neq \emptyset$ if and only if $P \asymp Q$.*

To see the complexity of the algorithm, since it is well known that the problem of deciding bisimulaition equivalence of data-independent processes in value-passing CCS is NP-hard in the size of processes [9], the decision problem for the mobile process algebra, such as π-calculus and χ-calculus, is at least NP-hard.

5 Conclusion

In this section, we summarize our work. Our contribution lies in that: (1) We give a symbolic characterization for strong open congruence in χ-calculus. Symbolic technique, introduced by Hennessy and Lin [7] is widely used in the research of process algebra, especially for axiomatizing observation equivalence and bisimulation checking related problem. In this paper, the symbolic transition system and the definition for symbolic open congruence are given and the relation to the concrete ones is discussed. The main result is that the concrete and symbolic open congruence coincide. To our knowledge, this is the first time to apply the symbolic technique to the research of χ-calculus; (2) We present an algorithm to check strong open congruence for finite control processes. Our algorithm falls into the on-the-fly category. Since in strong case, the open barbed congruence coincide with the open congruence, as a byproduct, the algorithm can also be used to check open barbed congruence. It is worth pointing out that although we only deal with χ-calculus in this paper, our results can be easily adapted to update calculus [14] and fusion calculus [15] without mismatch, which are the asymmetric and polyadic version of χ-calculus.

The present work relies heavily on the symbolic technique, which is widely used in π-calculus. The characterization for open congruence in this paper follows the Sangiorgi's approach [18], in which open style bisimulation is first introduced a symbolic characterization is also given. However, due to the nature of χ-calculus, the symbolic definition in this paper is more involved. There is a sea of publication on checking bisimulation relations in process algebra. For π-calculus, checking open congruence has also been tackled. The *Mobility Workbench* [19] implements an 'on-the-fly' algorithm for open congruence and Pistore and Sangiorgi [16] present a partition refinement algorithm. As we know, similar problems for χ-calculus is not considered in the literature.

There are several directions for further research. The weak version congruence has not been investigated. Although we don't find any difficulty to adapt the definition and method in this paper to weak open congruence, the details need to be verified; and more, in weak case, barbed congruence is different from open congruence, checking this bisimulation relation is also very

interesting. The χ-calculus studied in this paper does not include mismatch operator, and it is interesting to consider bisimulation checking problem in $\chi^{\neq}$-calculus which has been provided and studied elsewhere [5][6].

References

[1] Dam, M., On the decidability of process equivalence for the π-calculus. Theoretical Computer Science, 1997, 163:214-228.

[2] Fu, Y., A Proof Theoretical Approach to Communications. ICALP'97, Lecture Notes in Computer Science, Vol 1256, Springer,Berlin, 1997, pp.325-335.

[3] Fu, Y., Bisimulation lattice of Chi calculus, ASIAN'98, Lecture Notes in Computer Science 1256, Springer 1998.

[4] Fu, Y., Bisimulation Congruence of Chi Calculus. To appear in Information and Computation.

[5] Fu, Y. and Z.Yang. Chi calculus with mismatch, CONCUR 2000, Lecture Notes in Computer Science, Vol 1877, Springer,Berlin, 2000, pp.385-396.

[6] Fu, Y. and Z.Yang. Understanding the Mismatch Combinator in Chi Calculus. Theoretical Computer Science, 290, 779-830, 2003.

[7] Hennessy, M. and H. Lin, Symbolic bisimulations. Theoretical Computer Science, 138(2): 353-389, 1995.

[8] Honda, K. and M. Tokoro, On asychronous communication semantics, object-based concurrent computing, Lecture Notes in Computer Science, Vol.612, Springer, Berlin, 1991, pp.21-51.

[9] Jonsson, B. and J.Parrow, Deciding bisimulation equivalences for a class of non-finite-state programs. Journal of Information and Computation, 107(2):272-302, 11,1993.

[10] Lin, H., 'On-the-fly instantiation' of value-passing processes. In FORTE/PSTV'98, pp 215-230. Kluwer Academic Publishers, 1998.

[11] Lin, H., Computing bisimulation for finite-control π-calculus, Journal of Computer Science and Technology, Vol.15, No. 1, 2000.

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

[13] Milner, R., J.Parrow and D.Walker, A Calculus of Mobile Process, part I/II. Journal of Information and Computation, 100:1-77, Sept.1992.

[14] Parrow, J. and B. Victor, The update calculus, AMAST'97, Lecture Notes in Computer Science, Vol 1119, Springer,Berlin, 1997, pp.389-405.

[15] Parrow, J. and B. Victor, The fusion calculus: expressiveness and symmetry in mobile processes, Logics in Computer Science'98, IEEE Computer Society Press, Silver Spring, MD, 1998, pp.176-185.

[16] Pistore, M. and D. Sangiorgi, A partition refinement algorithm for the π-calculus, In Proc. CAV'96, Lecture Notes in Computer Science 1102, Springer Verlag.

[17] Sangiorgi, D., π-calculus, internal mobility and agent-passing calculi, Theoret.Comput.Sci. 167(1996), 235-274.

[18] Sangiorgi, D., A theory of bisimulation for π-calculus, Acta Informatica 3 (1996) 69-97.

[19] Victor, B. and F.Moller, The mobility workbench-a tool for the π-calculus. In Proc. CAV'94, Lecture Notes in Computer Science, pp 428-440. Springer-Verlag, 1994.