On the Decidability of Checking Problems for CCS with Static μ-Operator*

Chaodong He, Yuxi Fu, and Hongfei Fu

BASICS**, Department of Computer Science
Shanghai Jiaotong University, Shanghai 200240, China
galois@sjtu.edu.cn, fu-yx@cs.sjtu.edu.cn, jt002845@sjtu.edu.cn

Abstract. The paper considers the strong bisimilarity/similarity checking problems for CCS^{μ} and $\mathrm{CCS}^{!}$, two typical variants of CCS. The strong bisimilarity of the $\mathrm{CCS}^{\mu}/\mathrm{CCS}^{!}$ processes is shown to be Π_1^0-complete, even when the processes are restricted to the restriction prenex normal forms. The strong bisimilarity between a $\mathrm{CCS}^{!}/\mathrm{CCS}^{\mu}$ process and a fixed regular process is also shown to be Π_1^0-complete. The paper also establishes the decidability of the similarity problem of a regular process by a $\mathrm{CCS}^{!}$ process.

1 Introduction

One of the most important problems in the area of system verification is that of *equivalence checking* [1]. In concurrency theory, these are the problems of deciding whether two given processes are behavioral equivalent or not. Among these equivalences, the bisimilarity plays a prominent role. Many decidability and complexity results have been established during the past two decades for a number of models, especially those in the PRS-hierarchy [13], for which a recent survey of results is given in [17] by Srba. Some classical results and proof techniques are summarized in [12, 6, 1]. One interesting observation from these investigations is that the bisimilarity is easier to check than the other equivalences in the linear-branching time spectrum.

CCS is a classical model of interaction introduced by Milner [14]. In [2], Busi, Gabbrielli and Zavattaro confirm that $\mathrm{CCS}^{\mathrm{d}\mu}$, the full CCS with communication, restriction and the *dynamic* fixpoint (i.e. μ-operator), is Turing Complete. It follows immediately that neither the strong bisimilarity nor the weak bisimilarity on the $\mathrm{CCS}^{\mathrm{d}\mu}$ processes is decidable. On the other hand, Hirshfeld, Christensen and Moller demonstrate in [9] that, if the restriction operator is removed from CCS, or the composition operator is replaced by parallel composition, the strong subbisimilarity becomes decidable. If we admit both restriction and communication in the calculi, there are still several variations that are not Turing complete [4],

* The work is supported by The National 973 Project (2003CB317005), The National Nature Science Foundation of China (60573002, 60703033).

** Laboratory for Basic Studies in Computing Science (`http://basics.sjtu.edu.cn`).

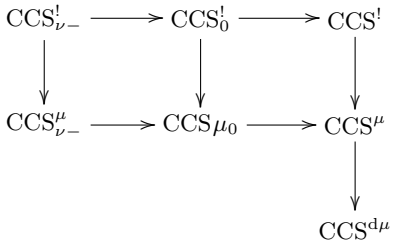

Fig. 1. CCS Hierarchy

some of which are given in Fig. 1. In the diagram an arrow "$\longrightarrow$" indicates the *subbisimilarity* relationship, which can be interpreted as saying that the target calculus is at least as expressive as the source calculus. Apart from $\mathrm{CCS}^{\mathrm{d}\mu}$, the six calculi in the top of Fig. 1 are not expressive enough to define counters in the sense of [4]. So the decidability of the equivalence checking problems for these systems arises naturally. Among the six variants, CCS^μ and $\mathrm{CCS}^!$ are most important. In both CCS^μ and $\mathrm{CCS}^!$, the *dynamic* μ-operator of $\mathrm{CCS}^{\mathrm{d}\mu}$ is replaced by the *static* μ-operator and replication respectively. The difference between these two kinds of μ-operator will be explained in Section 2. CCS^μ_0 and $\mathrm{CCS}^!_0$ are subcalculi of CCS^μ and $\mathrm{CCS}^!$, in which no restriction may occur underneath any μ-operator, respectively replication operator. Finally $\mathrm{CCS}^\mu_{\nu-}$ and $\mathrm{CCS}^!_{\nu-}$ are obtained from CCS^μ and $\mathrm{CCS}^!$ by removing the restriction operator. To fit into the PRS-hierarchy, we write **FS** for the class of the *finite state* (i.e. *regular*) processes.

In this paper we study the decidability issue of several strong bisimilarity/similarity checking problems for the models in the CCS-Hierarchy. These problems are indicated by the question marks in Fig. 2. The contributions of this paper is summarized as follows.

– Using the technique of 'Defender's Choice', a reduction from the halting problem of *Minsky Machine* establishes the undecidability (Π^0_1-hardness) of $\mathrm{CCS}^\mu_0 \sim \mathrm{CCS}^\mu_0$. The reduction is then modified to show the undecidabil-

X	$X \sim X$	$X \sim \mathbf{FS}$	$\mathbf{FS} \precsim X$	$X \precsim \mathbf{FS}$
$\mathrm{CCS}^!_{\nu-}$	✓	✓	?	?
$\mathrm{CCS}^\mu_{\nu-}$	✓	✓	?	?
$\mathrm{CCS}^!_0$	?	?	?	?
CCS^μ_0	?	?	?	?
$\mathrm{CCS}^!$	?	?	?	?
CCS^μ	?	?	?	?
$\mathrm{CCS}^{\mathrm{d}\mu}$	×	×	×	×

"$\sim$" for strong bisimilarity
"$\precsim$" for strong similarity

"✓" for decidable
"×" for undecidable
"?" for unknown

Fig. 2. Problems to Explore

ity (Π_1^0-hard) of $\mathrm{CCS}_0^! \sim \mathrm{CCS}_0^!$. This resolves the four problems in the first column of the table.

- Busi, Gabbrielli and Zavattaro establish in [3] the undecidability result (Σ_1^0-hardness) of the weak bisimilarity of $\mathrm{CCS}^!$. By modifying the proof of Busi *et al.*, $\mathrm{CCS}^! \sim \mathbf{FS}$ is shown undecidable (Π_1^0-hard), which immediately implies the undecidability (Π_1^0-hardness) of $\mathrm{CCS}^\mu \sim \mathbf{FS}$. Jančar, Kučera and Mayr point out in [7] the close relationship between $X \sim \mathbf{FS}$ and the *reachability of HM property*. It follows that the reachability of HM property for $\mathrm{CCS}^!/\mathrm{CCS}^\mu$ is also undecidable. By constructing a translation from $\mathrm{CCS}_0^!$ to *Labeled Petri Net*, we demonstrate the decidability of $\mathrm{CCS}_0^! \sim \mathbf{FS}$, $\mathrm{CCS}_0^! \precsim \mathbf{FS}$ and $\mathbf{FS} \precsim \mathrm{CCS}_0^!$, making use of Jančar and Moller's decidability result [8] on the Labeled Petri Nets. The same approach applies to CCS_0^μ.
- We show that $\mathbf{FS} \precsim \mathrm{CCS}^!$ is decidable. The proof follows the idea of bisimulation base and the technique of expansion tree presented in [6]. It also makes use of the well-quasi-order of $\mathrm{CCS}^!$ introduced in [2].

Only finite branching processes are considered in this paper. This guarantees that the bisimilarity/similarity can be approximated in the sense that $P \not\sim Q$ if and only if $P \not\sim_n Q$ for some n. It necessarily implies that all the problems in Fig. 2 are actually in Π_1^0. So we only need to show Π_1^0-hardness to get Π_1^0-completeness. We remark that a relation $R(x)$ is in Σ_1^0 (resp. Π_1^0), if it can be expressed by $\exists y.S(x,y)$ (resp. $\forall y.S(x,y)$) for some decidable relation $S(x,y)$. Note that $R(x)$ is in Σ_1^0 if and only if its complement is in Π_1^0. See [16] for further details about the arithmetic hierarchy.

The rest of the paper is organized as follows. Section 2 lays down the preliminaries. Section 3 investigates the problems of deciding the strong bisimilarity on the CCS^μ processes and the $\mathrm{CCS}^!$ processes. Section 4 considers the problem of deciding strong bisimilarity/similarity between a $\mathrm{CCS}^!$ or CCS^μ process and a finite state process. Section 5 gives some remarks.

2 Basic Definition and Notation

To describe the interactions between systems, we need *names*. The set of the names $\mathcal{N}$ is ranged over by $a, b, c, \ldots$, and the set of the names and the conames $\mathcal{N} \cup \overline{\mathcal{N}}$ is ranged over by $\alpha, \beta, \ldots$. To define the operational semantics, we need *action labels*. The set of the action labels $\mathcal{A} = \mathcal{N} \cup \overline{\mathcal{N}} \cup \{\tau\}$ is ranged by λ. To introduce the fixpoint operator, we need *variables*. The set of the variables $\mathcal{V}$ is ranged over by X, Y, Z.

The set $\mathcal{E}$ of CCS^μ expressions is generated inductively by the grammar

$$E \ ::= \ \mathbf{0} \mid X \mid \lambda.E \mid E \,|\, E' \mid (a)E \mid E + E' \mid \mu X.E$$

The relabeling operator is ruled out for the reason that it is too powerful. Since our purpose is to demonstrate the hardness of bisimilarity checking, the model should be restricted as less expressive as possible. The binary choice $E + E'$ will be used in its guarded form, meaning that both E and E' are in prefix form. The

$$\textsf{Prefix}\ \frac{}{\lambda.E \xrightarrow{\lambda} E} \qquad \textsf{Composition}\ \frac{E \xrightarrow{\lambda} E'}{E \,|\, F \xrightarrow{\lambda} E' \,|\, F} \quad \frac{E \xrightarrow{\alpha} E' \quad F \xrightarrow{\overline{\alpha}} F'}{E \,|\, F \xrightarrow{\tau} E' \,|\, F'}$$

$$\textsf{Restriction}\ \frac{E \xrightarrow{\lambda} E' \quad a \notin n(\lambda)}{(a)E \xrightarrow{\lambda} (a)E'} \qquad \textsf{Choice}\ \frac{E \xrightarrow{\lambda} E'}{E + F \xrightarrow{\lambda} E'} \qquad \textsf{Fixpoint}\ \frac{E\{\mu X.E/X\} \xrightarrow{\lambda} E'}{\mu X.E \xrightarrow{\lambda} E'}$$

Fig. 3. The Semantics for CCS$^{\mu}$

guardedness guarantees the finite branching property. The variable X in $\mu X.E$ is *bound*. A variable is *free* if it is not bound. A CCS$^{\mu}$ expression containing no free variables is a *process*. The set of the processes is denoted by $\mathcal{P}$.

The standard semantics of CCS$^{\mu}$ is given by the *labeled transition system* $(\mathcal{E}, \mathcal{A}, \rightarrow)$, where the elements of $\mathcal{E}$ are often referred to as *states*. The relation $\rightarrow \subseteq \mathcal{E} \times \mathcal{A} \times \mathcal{E}$ is the *transition* relation. The membership $(E, \lambda, E') \in \rightarrow$ is always indicated by $E \xrightarrow{\lambda} E'$. The relation $\rightarrow$ is generated inductively by the rules defined in Fig. 3.

Special attention should be paid to the Fixpoint rule. The higher order substitutions used in this paper, like $\{\mu X.E/X\}$, is *static*, meaning that when applying the substitution to a process expression E, the bound names need be renamed to avoid name capture. Under the static interpretation the process $\mu X.(a|(a)(\overline{a}|X))$ can not perform any action. If on the other hand the *dynamic* μ-operator is adopted, one would have the infinite computation

$$\mu X.(a \,|\, (a)(\overline{a} \,|\, X)) \xrightarrow{\tau} a \,|\, (a)(\mathbf{0} \,|\, \mathbf{0} \,|\, (a)(\overline{a} \,|\, \mu X.(a \,|\, (a)(\overline{a} \,|\, X)))) \xrightarrow{\tau} \ldots$$

The notation CCS$^{\mathrm{d}\mu}$ appeared in Fig. 1 refers to the CCS with the dynamic fixpoint operator. For more discussions on the relative expressive power of these two kinds of μ-operator, see [2] and [4].

A binary relation $\mathcal{R}$ on the set of processes is a *strong simulation* if, for each $(P, Q) \in \mathcal{R}$, P can be simulated by Q in the following sense:

If $P \xrightarrow{\lambda} P'$, then $Q \xrightarrow{\lambda} Q'$ for some Q' such that $(P', Q') \in \mathcal{R}$.

A binary relation $\mathcal{R}$ is a *strong bisimulation* if both $\mathcal{R}$ and its inverse $\mathcal{R}^{-1}$ are strong simulations. The *strong similarity* $\precsim$ is the largest strong simulation, and the *strong bisimilarity* $\sim$ is the largest strong bisimulation. The former is a preorder and the latter is an equivalence.

The strong bisimilarity has a game theoretic characterization known as the *bisimulation game*. It is a complete-information dynamic game played by two players named 'attacker' and 'defender'. The labeled transition system $(\mathcal{P}, \mathcal{A}, \rightarrow)$ is perceived as a chessboard. During the play the current position is described by a pair of states $(P_1, P_{-1}) \in \mathcal{P} \times \mathcal{P}$. The game is played in rounds. In each round the players change the position according to the following rules:

1. The attacker chooses a state $i \in \{1, -1\}$, an action $\lambda \in \mathcal{A}$, and some $P_i' \in \mathcal{P}$ such that $P_i \xrightarrow{\lambda} P_i'$.

2. The defender responds by choosing some $P'_{-i} \in \mathcal{P}$ such that $P_{-i} \xrightarrow{\lambda} P'_{-i}$; and then (P'_1, P'_{-1}) becomes the current position of the next round.

If the defender never gets stuck, it wins. Otherwise the attacker wins. It is easy to see that the defender has a winning strategy in the bisimulation game starting from the initial position (P, Q) if and only if $P \sim Q$.

The variant CCS$^!$ is obtained from CCS$^\mu$ by replacing the fixpoint operator by the *replication* operator. The grammar of the process is defined as follows:

$$P ::= \mathbf{0} \mid \lambda.P \mid P \mid P' \mid (a)P \mid P + P' \mid !P$$

The operational semantics of the replication is given by the following two rules:

$$\textsf{Replication} \quad \frac{P \xrightarrow{\lambda} P'}{!P \xrightarrow{\lambda} P' \mid !P} \qquad \frac{P \xrightarrow{\alpha} P' \quad P \xrightarrow{\overline{\alpha}} P''}{!P \xrightarrow{\tau} P' \mid P'' \mid !P}$$

Throughout this paper we do not distinguish processes or process expressions up to the congruence induced by the commutativity and the associativity of the choice operator "+" and the composition operator "|".

3 Undecidability of Strong Bisimilarity

In this section, we show that the strong bisimilarity between two CCS$^\mu$ processes is Π_1^0-hard by a many-one reduction from HaltingMinskyMachine. The latter is known to be Σ_1^0-complete. These two problems are formally stated as follows:

Problem:	CCS$^\mu \sim$ CCS$^\mu$
Instance:	Two CCS$^\mu$ processes P_1 and P_2.
Question:	$P_1 \sim P_2$?

Problem:	HaltingMinskyMachine
Instance:	A Minsky Machine $\mathbb{R}$.
Question:	Does the computation of $\mathbb{R}$ terminate when $\mathbb{R}$ starts from the configuration $(1; 0, 0)$?

Although the construction of Busi *et al* [3] can work for the general case, the construction presented here allows us to prove much stronger results. The proof technique is known as 'Defender's Choice' [12]. Using this technique and additional manipulations, the same problem for CCS$^!$ can also be shown to be Π_1^0-hard.

Two-register *Minsky Machine* is a Turing complete computational model [15]. A Minsky Machine $\mathbb{R}$ has two registers r_1 and r_2 that can hold arbitrary large natural numbers. The behavior of $\mathbb{R}$ is specified by a sequence of instructions $\{(1 : I_1), (2 : I_2), \ldots, (n-1 : I_{n-1}), (n : \texttt{halt})\}$. For each $i \in \{1, \ldots, n-1\}$, the i-th instruction may be in one of two forms:

- $(i : Succ(r_j))$: The instruction adds 1 to the content of the register r_j and $i + 1$ becomes the value of the program counter.

- $(i : Decjump(r_j, s))$: If the contents of the register r_j is not zero, the instruction decreases it by 1 and $i+1$ becomes the value of the program counter; otherwise s becomes the value of the program counter.

The configuration of $\mathbb{R}$ is given by the tuple $(i; c1, c2)$ where i is the program counter indicating the instruction to be executed, and $c1$,$c2$ are the current contents of the registers r_1,r_2. The computation of $\mathbb{R}$ is defined in a natural way via a (finite or infinite) sequence of configurations starting from a certain initial configuration. Whenever the n-th instruction (known as the halting state) is reached, the computation terminates. The following lemma is well known.

Lemma 1. HaltingMinskyMachine *is undecidable. It is* Σ_1^0*-complete in the arithmetic hierarchy.*

3.1 Basic Construction

Our goal is to show that there is an effective construction such that given a Minsky Machine $\mathbb{R}$, it produces a pair of CCS$^\mu$ processes A and B with the property that $\mathbb{R}$ halts if and only if A and B are not strongly bisimilar. For convenience the construction given below makes use of the constant definition instead of the μ-operator. Notice however that since there is no restriction operator underneath any μ-operations, the μ-operator used here is trivially static.

Let $\mathbb{R}$ be an instance of HaltingMinskyMachine whose instructions are $\{(1 : I_1), (2 : I_2), \ldots, (n-1 : I_{n-1}), (n : \texttt{halt})\}$. For every i from 1 to n, we define a pair of processes P_i and Q_i as follows:

– If the i-th instruction is $(i : Succ(r_j))$, let

$$P_i \stackrel{\text{def}}{=} \overline{\mathsf{inc_j}}.P_{i+1}$$
$$Q_i \stackrel{\text{def}}{=} \overline{\mathsf{inc_j}}.Q_{i+1}$$

– If the i-th instruction is $(i : Decjump(r_j, s))$, let

$$P_i \stackrel{\text{def}}{=} \overline{\mathsf{dec_j}}.d.P_{i+1} + \overline{\mathsf{zero_j}}.(\overline{\mathsf{tt}}.z.P_s + \overline{\mathsf{ff}}.z.Q_s)$$
$$Q_i \stackrel{\text{def}}{=} \overline{\mathsf{dec_j}}.d.Q_{i+1} + \overline{\mathsf{zero_j}}.(\overline{\mathsf{tt}}.z.Q_s + \overline{\mathsf{ff}}.z.P_s)$$

– For the n-th instruction $(n : \texttt{halt})$, let

$$P_n \stackrel{\text{def}}{=} \mathsf{halt}.\mathsf{0}$$
$$Q_n \stackrel{\text{def}}{=} \mathsf{0}$$

The processes P_i's and Q_i's will be used to simulate the execution of the i-th instruction of $\mathbb{R}$. The idea behind this definition will be explained later. Note that the only difference between the P_i's and the Q_i's is that P_n can perform a

special action halt whereas Q_n can not. The processes $Counter_j(k)$ for $j \in \{1,2\}$ defined below are used to partially simulate the registers of $\mathbb{R}$.

$$Counter_j(k) \stackrel{\text{def}}{=} \underbrace{C_j \mid C_j \mid \ldots \mid C_j}_{k} \mid O_j$$

where O_j and C_j are defined as follows:

$$O_j \stackrel{\text{def}}{=} \mathsf{inc_j}.(C_j \mid O_j) + \mathsf{zero_j}.\mathsf{tt}.O_j$$
$$C_j \stackrel{\text{def}}{=} \mathsf{dec_j}.0 + \mathsf{zero_j}.\mathsf{ff}.C_j$$

The process $Counter_1(0)$ for example is one of the weak forms of counter used in this paper. They are not the counter proper. However they are good enough for the purpose of deriving some undecidability results.

The next two processes are used to describe the configurations of $\mathbb{R}$.

$$Config_A(i;c_1,c_2) \stackrel{\text{def}}{=} (\widetilde{\mathsf{inc}})(\widetilde{\mathsf{dec}})(\widetilde{\mathsf{zero}})(\mathsf{tt})(\mathsf{ff})(P_i \mid Counter_1(c1) \mid Counter_2(c2))$$
$$Config_B(i;c_1,c_2) \stackrel{\text{def}}{=} (\widetilde{\mathsf{inc}})(\widetilde{\mathsf{dec}})(\widetilde{\mathsf{zero}})(\mathsf{tt})(\mathsf{ff})(Q_i \mid Counter_1(c1) \mid Counter_2(c2))$$

Some of the properties of these definitions are described in the next two lemmas. The first one is immediate by definition.

Lemma 2. *Let $(i;c1,c2)$ be a configuration of $\mathbb{R}$ and the i-th instruction is $(i : Succ(r_j))$, then there is a unique continuation of the bisimulation game from the pair of processes $Config_A(i;c_1,c_2)$ and $Config_B(i;c_1,c_2)$ such that after one round, the players reach the pair $Config_A(i;c'_1,c'_2)$ and $Config_B(i;c'_1,c'_2)$ where $c'_j = c_j + 1$ and $c'_{3-j} = c_{3-j}$.*

Lemma 3. *Let $(i;c1,c2)$ be a configuration of $\mathbb{R}$ and the i-th instruction is $(i : Decjump(r_j,s))$. Assume that a bisimulation game is played from the pair $Config_A(i;c_1,c_2)$ and $Config_B(i;c_1,c_2)$. The followings hold:*

(a) If $c_j = 0$, then there is a unique continuation of the game such that after three rounds, the players reach the pair $Config_A(s;c_1,c_2)$ and $Config_B(s;c_1,c_2)$.
(b) If $c_j > 0$ and the attacker chooses the τ action induced by the synchronization via channel $\mathsf{dec_j}$, then the defender has a way to continue the game such that, after two rounds, $Config_A(i;c'_1,c'_2)$ and $Config_B(i;c'_1,c'_2)$ are reached, where $c'_j = c_j - 1$ and $c'_{3-j} = c_{3-j}$. If the defender does not play in this way, there is a way for the attacker to win the game.
(c) If $c_j > 0$ and the attacker chooses the τ action induced by the synchronization via channel $\mathsf{zero_j}$, then there is a way for the defender to win the game.

Proof. Without loss of generality, assume that $j = 1$ and the attacker always chooses the process containing P_i at the first step. For part (a), the players' choices cannot affect the continuation of the game. See the following diagrammatic illustration:

$$
\begin{array}{ccc}
(\ldots)(P_i|O_1|\ldots) & & (\ldots)(Q_i|O_1|\ldots) \\
\downarrow{\scriptstyle \tau} & & \downarrow{\scriptstyle \tau} \\
(\ldots)((\overline{\mathsf{tt}}.z.P_s + \overline{\mathsf{ff}}.z.Q_s)|\mathsf{tt}.O_1|\ldots) & & (\ldots)((\overline{\mathsf{tt}}.z.Q_s + \overline{\mathsf{ff}}.z.P_s)|\mathsf{tt}.O_1|\ldots) \\
\downarrow{\scriptstyle \tau} & & \downarrow{\scriptstyle \tau} \\
(\ldots)(z.P_s|O_1|\ldots) & & (\ldots)(z.Q_s|O_1|\ldots) \\
\downarrow{\scriptstyle z} & & \downarrow{\scriptstyle z} \\
(\ldots)(P_s|O_1|\ldots) & & (\ldots)(Q_s|O_1|\ldots)
\end{array}
$$

For part (b), if the attacker chooses the τ action induced by $P_i \xrightarrow{\overline{\mathsf{dec_j}}} d.P_{i+1}$ and $C_j \xrightarrow{\mathsf{dec_j}} \mathsf{0}$, the defender's match is induced by $Q_i \xrightarrow{\overline{\mathsf{dec_j}}} d.Q_{i+1}$ and $C_j \xrightarrow{\mathsf{dec_j}} \mathsf{0}$. See

$$
\begin{array}{ccc}
(\ldots)(P_i|\ldots|C_1|O_1|\ldots) & & (\ldots)(Q_i|\ldots|C_1|O_1|\ldots) \\
\downarrow{\scriptstyle \tau} & & \downarrow{\scriptstyle \tau} \\
(\ldots)(d.P_{i+1}|\ldots|O_1|\ldots) & & (\ldots)(d.Q_{i+1}|\ldots|O_1|\ldots) \\
\downarrow{\scriptstyle d} & & \downarrow{\scriptstyle d} \\
(\ldots)(P_{i+1}|\ldots|O_1|\ldots) & & (\ldots)(Q_{i+1}|\ldots|O_1|\ldots)
\end{array}
$$

If the defender chooses the other τ action, say the one induced by $Q_i \xrightarrow{\overline{\mathsf{zero_j}}} \overline{\mathsf{tt}}.z.Q_s + \overline{\mathsf{ff}}.z.P_s$ and $C_j \xrightarrow{\mathsf{zero_j}} \mathsf{ff}.C_j$ (or $O_j \xrightarrow{\mathsf{zero_j}} \mathsf{tt}.O_j$), the attacker can win the game by performing d in the next round.

$$
\begin{array}{ccc}
(\ldots)(P_i|\ldots|C_1|O_1|\ldots) & & (\ldots)(Q_i|\ldots|C_1|O_1|\ldots) \\
\downarrow{\scriptstyle \tau} & & \downarrow{\scriptstyle \tau} \\
(\ldots)((\overline{\mathsf{tt}}.z.P_s + \overline{\mathsf{ff}}.z.Q_s)|\ldots|\mathsf{ff}.C_j|O_1|\ldots) & & (\ldots)((\overline{\mathsf{tt}}.z.Q_s + \overline{\mathsf{ff}}.z.P_s)|\ldots|C_j|\mathsf{tt}.O_1|\ldots) \\
\downarrow{\scriptstyle \tau} & & \downarrow{\scriptstyle \tau} \\
(\ldots)(z.Q_s|\ldots|C_j|O_1|\ldots) & & (\ldots)(z.Q_s|\ldots|C_j|O_1|\ldots)
\end{array}
$$

For part (c), there are two choices for the attacker. In the first case, the attacker chooses the τ action induced by $P_i \xrightarrow{\overline{\mathsf{zero_j}}} \overline{\mathsf{tt}}.z.P_s + \overline{\mathsf{ff}}.z.Q_s$ and $C_j \xrightarrow{\mathsf{zero_j}} \mathsf{ff}.C_j$. The defender can simulate this action by performing the τ-action induced by $Q_i \xrightarrow{\overline{\mathsf{zero_j}}} \overline{\mathsf{tt}}.z.Q_s + \overline{\mathsf{ff}}.z.P_s$ and $O_j \xrightarrow{\mathsf{zero_j}} \mathsf{tt}.O_j$. See the above diagram for illustration. When the play arrives at this position, the two processes would become syntactically the same! The other case is similar. □

The next two lemmas relate the termination of the machine $\mathbb{R}$ to the inequality of the processes $Config_A(1;0,0), Config_B(1;0,0)$.

Lemma 4. *If the execution of* $\mathbb{R}$ *from the configuration* $(1;0,0)$ *terminates, then* $Config_A(1;0,0) \not\sim Config_B(1;0,0)$.

Proof. We show that the attacker has a winning strategy to win the bisimulation game if the run of $\mathbb{R}$ from the configuration $(1;0,0)$ terminates. The attacker starts the game by choosing the process $Config_A(1;0,0)$, and he always chooses this process during the play. When the process reaches $Config_A(i;c_1,c_2)$ and the i-th instruction of $\mathbb{R}$ is '$i : Decjump(r_j, s)$' and $c_j > 0$, the attacker always chooses the τ action induced by the synchronization via channel $\mathsf{dec_j}$, i.e. the attacker let the process simulate the execution of $\mathbb{R}$ honestly. According to part (b) of Lemma 3, the defender must play the corresponding moves in order not to lose immediately. However, since $\mathbb{R}$ will terminate eventually, the attacker can reach to $Config_A(n;c_1,c_2)$ and forces the defender to reach to $Config_B(n;c_1,c_2)$. Now the attacker performs $Config_A(n;c_1,c_2) \xrightarrow{\mathsf{halt}}$ and the defender gets stuck. □

Lemma 5. *If the execution of* $\mathbb{R}$ *from the configuration* $(1;0,0)$ *does not terminate, then* $Config_A(1;0,0) \sim Config_B(1;0,0)$.

Proof. We show that the defender has a winning strategy to win the bisimulation game if the execution of $\mathbb{R}$ from the configuration $(1;0,0)$ does not terminate. During the play, if the attacker chooses a process and lets it simulate the execution of $\mathbb{R}$ honestly, the defender can do the same thing on the other process. If the play goes this way, no one gets stuck and the defender wins. If the attacker chooses $Config_A(i;c_1,c_2)$ or $Config_B(i;c_1,c_2)$ and the i-th instruction is '$i : Decjump(r_j, s)$ and $c_j > 0$, it may perform the τ action induced by the synchronization via channel $\mathsf{zero_j}$. But then according to part (c) of Lemma 3, the defender has a way to win the game. □

To state the main result of this section, we need to define the restriction prenex normal form.

Definition 1. *A process is in* restriction prenex normal form *if* P *is of the form* $(a_1)(a_2)\ldots(a_k)P'$ *for some restriction free* P'.

Theorem 1. *The strong bisimilarity of* CCS^{μ} *is* Π_1^0*-complete even for the restriction prenex normal forms.*

Proof. Two CCS^{μ} processes, $Config_A(i;0,0)$ and $Config_B(i;0,0)$, have been effectively constructed from the Minsky Machine $\mathbb{R}$. By Lemma 4 and Lemma 5, the execution of $\mathbb{R}$ from the initial configuration $(1;0,0)$ terminates if and only if $Config_A(i;0,0) \not\sim Config_B(i;0,0)$. Therefore a many-one reduction from HaltingMinskyMachine to the complement of $\mathrm{CCS}^{\mu} \sim \mathrm{CCS}^{\mu}$ has been created. By Lemma 1, the complement of $\mathrm{CCS}^{\mu} \sim \mathrm{CCS}^{\mu}$ is Σ_1^0-hard, which means that $\mathrm{CCS}^{\mu} \sim \mathrm{CCS}^{\mu}$ is Π_1^0-hard. On the other hand, since $\sim = \bigcup_{i\in\omega} \sim_i$ for the finite-branching processes, there is a trivial procedure that can find a witness if

the two input processes are not bisimilar, which means that $\mathrm{CCS}^{\mu} \sim \mathrm{CCS}^{\mu}$ is in Π_1^0. Hence the Π_1^0-completeness is established. Notice that both $Config_A(i;0,0)$ and $Config_B(i;0,0)$ are in restriction prenex normal form. So Π_1^0-completeness still holds when restricted to the restriction prenex normal forms. □

3.2 Generalization to CCS$^!$

In the CCS-hierarchy established in [4], CCS^{μ} is strictly more expressive than $\mathrm{CCS}^!$ in the sense of weak bisimilarity. In the strong case, CCS^{μ} is strictly more expressive than $\mathrm{CCS}^!$ even if all the choices are guarded. One can show, for example, that $\mu X.a.b.c.X$ is not strongly bisimilar to any process of $\mathrm{CCS}^!$. A consequence of this observation is that the previous result does not immediately imply the same result for $\mathrm{CCS}^!$. The proof can be repeated but need some careful modifications. The intuition of the next encoding is to interpret every instruction by a process of the form $!addr.opr$, where *addr* should be perceived as the address of the instruction and *opr* the operation of the instruction.

Theorem 2. *The strong bisimilarity of* **CCS**$^!$ *is Π_1^0-complete for the restriction prenex normal forms.*

Proof. The simulation of the Minsky Machine defined in the previous section is modified in a way that the role of the fixpoint operator is replaced by that of the replication operator.

– If the i-th instruction is $(i : Succ(r_j))$, let

$$P_i \stackrel{\text{def}}{=} !\mathsf{inst}_P^i.\overline{\mathsf{inc}_j}.\overline{\mathsf{inst}_P^{i+1}}$$
$$Q_i \stackrel{\text{def}}{=} !\mathsf{inst}_Q^i.\overline{\mathsf{inc}_j}.\overline{\mathsf{inst}_Q^{i+1}}$$

– If the i-th instruction is $(i : Decjump(r_j, s))$, let

$$P_i \stackrel{\text{def}}{=} !\mathsf{inst}_P^i.(\overline{\mathsf{dec}_j}.d.\overline{\mathsf{inst}_P^{i+1}} + \overline{\mathsf{zero}_j}.(\overline{\mathsf{tt}}.\tau.\tau.z.\overline{\mathsf{inst}_P^s} + \overline{\mathsf{ff}}.\mathsf{rdy}.z.\overline{\mathsf{inst}_Q^s}))$$
$$Q_i \stackrel{\text{def}}{=} !\mathsf{inst}_Q^i.(\overline{\mathsf{dec}_j}.d.\overline{\mathsf{inst}_Q^{i+1}} + \overline{\mathsf{zero}_j}.(\overline{\mathsf{tt}}.\tau.\tau.z.\overline{\mathsf{inst}_Q^s} + \overline{\mathsf{ff}}.\mathsf{rdy}.z.\overline{\mathsf{inst}_P^s}))$$

– For the n-th instruction $(n : \mathtt{halt})$, let

$$P_n \stackrel{\text{def}}{=} !\mathsf{inst}_P^n.\mathsf{halt}.0$$
$$Q_n \stackrel{\text{def}}{=} !\mathsf{inst}_Q^n.0$$

The counter must be redefined in $\mathrm{CCS}^!$ without using any restrictions.

$$O_j \stackrel{\text{def}}{=} !(\mathsf{inc_j}.C_j + \mathsf{zero_j}.\mathsf{tt})$$
$$C_j \stackrel{\text{def}}{=} !(\mathsf{dec_j} + \mathsf{zero_j}.\mathsf{ff}.\overline{\mathsf{m_j}}) \mid !\mathsf{m_j}.\overline{\mathsf{rdy}}(\mathsf{dec_j} + \mathsf{zero_j}.\mathsf{ff}.\overline{\mathsf{m_j}})$$

Notice that there are extra τ's and the new name rdy in the definition of P_i and Q_i. This is because the counter defined here may perform some extra computation steps during the zero-testing wrongfully chosen. The $\mathsf{m_j}$'s are bound names in the configuration. The configuration $(i; c_1, c_2)$ of $\mathbb{R}$ is interpreted by $Config_A(1; c_1, c_2)$ and $Config_B(1; c_1, c_2)$ defined as follows:

$$\begin{aligned} Config_A(i; c_1, c_2) \stackrel{\text{def}}{=}\ & (\widetilde{\mathsf{inst}})(\widetilde{\mathsf{inc}})(\widetilde{\mathsf{dec}})(\widetilde{\mathsf{zero}})(\widetilde{\mathsf{m}})(\mathsf{tt})(\mathsf{ff})(\mathsf{rdy}) \\ & (\overline{\mathsf{inst}_P^i} \mid \prod_{i=1}^{n} P_i \mid \prod_{i=1}^{n} Q_i \mid Counter_1(c_1) \mid Counter_2(c_2)) \\ Config_B(i; c_1, c_2) \stackrel{\text{def}}{=}\ & (\widetilde{\mathsf{inst}})(\widetilde{\mathsf{inc}})(\widetilde{\mathsf{dec}})(\widetilde{\mathsf{zero}})(\widetilde{\mathsf{m}})(\mathsf{tt})(\mathsf{ff})(\mathsf{rdy}) \\ & (\overline{\mathsf{inst}_Q^i} \mid \prod_{i=1}^{n} P_i \mid \prod_{i=1}^{n} Q_i \mid Counter_1(c_1) \mid Counter_2(c_2)) \end{aligned}$$

Using the same argument given in the previous subsection, it is routine to show that $\mathbb{R}$ terminates if and only if $Config_A(1; c_1, c_2) \not\sim Config_B(1; c_1, c_2)$. □

Notice that the processes in restriction prenex normal form are automatically in $\mathrm{CCS}_0^\mu/\mathrm{CCS}_0^!$. Thus, both $\mathrm{CCS}_0^\mu \sim \mathrm{CCS}_0^\mu$ and $\mathrm{CCS}_0^! \sim \mathrm{CCS}_0^!$ are Π_1^0-complete.

4 Strong Bisimilarity on Finite State Process

The choice operator is very much a specification combinator whereas the composition operator and the localization operator are purely implementation combinators. A specification defined in CCS is a finite state process since it does not contain any occurrences of the composition and the localization operators. An abundance of specification examples in CCS are given in [14]. If an implementation Imp of a specification $Spec$ are both defined in CCS, the correctness of the implementation boils down to the question "Is $Imp \approx Spec$?". The problem is undecidable. A less ambitious question asks if $Imp \sim Spec$? We investigate in this section the (un)decidability of this problem.

The following definition introduces a classification of processes that helps to give a precise answer to the problem.

Definition 2. *The* restriction depth *of a process P of* $\mathrm{CCS}^!$*, denoted by* $\mathsf{RD}(P)$*, is defined inductively as follows:* $\mathsf{RD}(P) = 0$ *if there is no restriction operator under any replication. For $k > 0$,* $\mathsf{RD}(P) = k$ *if the maximum* $\mathsf{RD}(P')$ *is $k - 1$ for those sub-processes P' of P that are under one replication operator.*

The class of all the processes in $\mathrm{CCS}^!$ *with restriction depth no more than k is denoted by* $\mathrm{CCS}_k^!$.

A similar notion can be defined for CCS^μ. Notice that the processes in restriction prenex normal form are all in $\mathrm{CCS}_0^!$. Conversely, every process in $\mathrm{CCS}_0^!$ can be converted to a restriction prenex normal form by applying α-conversion if necessary.

4.1 The General Case

In this section we consider the following general problem:

Problem:	$\mathrm{CCS}^{!} \sim \mathbf{FS}$
Instance:	A $\mathrm{CCS}^{!}$ process P and a finite state process F.
Question:	$P \sim F$?

This problem is undecidable even for $\mathrm{CCS}_1^{!}$.

Theorem 3. *The strong bisimilarity between a process $P \in \mathrm{CCS}_1^{!}$ and a finite state process $F \in \mathbf{FS}$ is Π_1^0-complete, even if F is fixed.*

The proof of theorem 3 is strongly affected by the construction of Busi *et al* [3], which reveals that $\mathrm{CCS}_1^{!}$ processes can simulate Minsky Machines in a nondeterministic fashion. For completeness the construction and the proof details are sketched. Busi *et al*'s translation of a Minsky Machine $\mathbb{R}$ to a $\mathrm{CCS}^{!}$ process is presented at first:

- If the i-th instruction is $(i : Succ(r_j))$, let

$$P_i \stackrel{\text{def}}{=} !\mathsf{inst}^i.(\overline{\mathsf{inc}_j} | inc.\overline{\mathsf{inst}^{i+1}})$$

- If the i-th instruction is $(i : Decjump(r_j, s))$, let

$$P_i \stackrel{\text{def}}{=} !\mathsf{inst}^i.(\overline{\mathsf{dec}_j} | (dec.\overline{\mathsf{inst}^{i+1}} + zero.\overline{\mathsf{inst}^s}))$$

- If the n-th instruction is $(n : \mathtt{halt})$, let

$$P_n \stackrel{\text{def}}{=} !\mathsf{inst}^n.0$$

The counters and the configurations are defined in Fig. 4. In the definition of $Config(i; c_1, c_2)$ the components G_1 and G_2 are the deadlock garbages produced during computation.

Lemma 6. *Let $(i; c1, c2)$ be a configuration of a Minsky Machine $\mathbb{R}$. If the computation of $\mathbb{R}$ from $(i; c1, c2)$ terminates, then $Config(i; c_1, c_2)$ converges, i.e. there exists a computation that terminates. Otherwise $Config(i; c_1, c_2) \sim !\tau$.*

The proof of Lemma 6 is nothing more than a careful examination. Theorem 3 can be derived directly from the lemma.

Since CCS^{μ} is stronger than $\mathrm{CCS}_1^{!}$, the next corollary is immediate.

Corollary 1. *The strong bisimilarity between a process $P \in \mathrm{CCS}^{\mu}$ and a finite state process $F \in \mathbf{FS}$ is Π_1^0-complete, even if F is fixed.*

Another class of problems, which is closely related to the problem of the strong bisimilarity on the finite state process, is concerned with the *reachability HM property*. Given a process P, and a logic formula φ which specifies a property, the reachability property problem asks whether P can reach some states that satisfy the property φ; in other words, it asks whether $P \models \mathsf{EF}\varphi$, where EF is a connective in the logic CTL. In the case of ReachabilityHMProperty defined below, the logic formula φ is confined in Hennessy-Milner Logic [14].

$$
\begin{array}{rcl}
Counter_j(k) & \stackrel{\text{def}}{=} & \overline{\mathsf{nr}_j} \mid !\mathsf{nr}_j.(\mathrm{m})(\mathrm{i})(\mathrm{d})(\mathrm{u})(\ \overline{\mathrm{m}} \mid !\mathrm{m}.(\mathsf{inc}_j.\bar{\mathrm{i}} + \mathsf{dec}_j.\bar{\mathrm{d}}) \mid \\
 & & \quad !\mathrm{i}.(\overline{\mathrm{m}}|\overline{inc}|\overline{\mathrm{u}}|\mathrm{d.u}.(\overline{\mathrm{m}}|\overline{dec})) \mid \mathrm{d}.(\overline{zero}|\mathrm{u}.DIV|\overline{\mathsf{nr}_j}) \mid \\
 & & \quad \prod_k(\overline{\mathrm{u}}|\mathrm{d.u}.(\overline{\mathrm{m}}|\overline{dec}))\ ) \\
Config(i;c_1,c_2) & \stackrel{\text{def}}{=} & (\widetilde{\mathsf{inst}})(\widetilde{\mathsf{inc}})(\widetilde{\mathsf{dec}})(\widetilde{\mathsf{nr}})(inc)(dec)(zero)(\overline{\mathsf{inst}^1} \mid \prod_{i=1}^{n} P_i \mid \\
 & & \quad Counter_1(c_1) \mid Counter_2(c_2) \mid \prod_{k_1} G_1 \mid \prod_{k_2} G_2) \\
G_j & \stackrel{\text{def}}{=} & (\mathrm{m})(\mathrm{i})(\mathrm{d})(\mathrm{u})(\\
 & & \quad !\mathrm{m}.(\mathsf{inc}_j.\bar{\mathrm{i}} + \mathsf{dec}_j.\bar{\mathrm{d}}) \mid !\mathrm{i}.(\overline{\mathrm{m}}|\overline{inc}|\overline{\mathrm{u}}|\mathrm{d.u}.\overline{dec}) \mid \mathrm{u}.DIV\ )
\end{array}
$$

Fig. 4. Counters and Configurations Defined in $\mathrm{CCS}_1^!$

Problem:	REACHABILITYHMPROPERTY(X)
Instance:	A process $P \in X$, and a formula φ in Hennesy-Milner Logic.
Question:	$P \models \mathsf{EF}\varphi$?

In the Theorem 22 of [7], Jančar, Kučera and Mayr establishes the following.

Lemma 7. *Let X be a process class. If* REACHABILITYHMPROPERTY(X) *is decidable, then $X \sim \mathbf{FS}$ is also decidable.*

Notice that, in the proof of Theorem 22 of [7], a many-one reduction is set up from $X \sim \mathbf{FS}$ to REACHABILITYHMPROPERTY(X). Since $\mathrm{CCS}_1^! \sim \mathbf{FS}$ and $\mathrm{CCS}_1^\mu \sim \mathbf{FS}$ are both shown Π_1^0-complete, REACHABILITYHMPROPERTY($\mathrm{CCS}_1^!$) and REACHABILITYHMPROPERTY(CCS_1^μ) must be Π_1^0-complete. Hence we have the next corollary.

Corollary 2. *The reachability HM property problem for* $\mathrm{CCS}^!{}_1$ *and* CCS^μ *is Π_1^0-complete.*

4.2 The Case of Restriction Prenex Normal Form

Although both $\mathrm{CCS}^! \sim \mathbf{FS}$ and $\mathrm{CCS}^\mu \sim \mathbf{FS}$ are undecidable in the general case, their restricted versions, $\mathrm{CCS}_0^! \sim \mathbf{FS}$ and $\mathrm{CCS}_0^\mu \sim \mathbf{FS}$, turn out to be decidable. These results are summarized in the following theorem.

Theorem 4. *The strong bisimilarity between a process $P \in \mathrm{CCS}_0^!$ (or $P \in \mathrm{CCS}_0^\mu$) and a finite state process $F \in \mathbf{FS}$ is decidable. The reachability HM property problem for* $\mathrm{CCS}_0^!$ *(or* CCS_0^μ*) is also decidable.*

The proof of Theorem 4 is indirect. A bisimilarity preserving translation from $\mathrm{CCS}_0^!$ (or CCS_0^μ) to Labeled Petri Net is created. With the help of Jančar *et al.* [8, 7], we know that the same problems for Labeled Petri Net are decidable. Hence our decidability proof.

We begin with the definition of the Petri Nets and the Labeled Petri Nets. We write $\mathbb{N}$ for the set of natural numbers.

Definition 3. *A* Petri Net *is a tuple* $N = (Q, T, F, M_0)$ *and a* Labeled Petri Net *is a tuple* $N = (Q, T, F, L, M_0)$*, where* Q *and* T *are finite disjoint sets of* places *and* transitions *respectively,* $F : (Q \times T) \cup (T \times Q) \to \mathbb{N}$ *is a* flow function *and* $L : T \to \mathcal{A}$ *is a* labeling*.* M_0 *is the* initial marking*, where a* marking M *is a function* $Q \to \mathbb{N}$ *assigning the number of* tokens *to each place.*

A transition $t \in T$ *is* enabled *at a marking* M*, denoted by* $M \xrightarrow{t}$*, if* $M(p) \geq F(p,t)$ *for every* $p \in Q$*. A transition* t *enabled at* M *may* fire *yielding the marking* M'*, denoted by* $M \xrightarrow{t} M'$*, where* $M'(p) = M(p) - F(p,t) + F(t,p)$ *for all* $p \in Q$*. For each* $\lambda \in \mathcal{A}$*, we write* $M \xrightarrow{\lambda}$*, respectively* $M \xrightarrow{\lambda} M'$ *to mean that* $M \xrightarrow{t}$*, respectively* $M \xrightarrow{t} M'$ *for some* t *with* $L(t) = \lambda$*.*

Some remarks are called for. In the above definition $\mathcal{A}$ is the set of action labels. A process may contain only a finite number of action labels. A Labeled Petri Net N can be viewed as a labeled transition system $(\mathbb{M}, \mathcal{A}, \to)$ with $\mathbb{M}$ being the markings of N. The strong bisimilarity is defined accordingly. Suppose $Q = \{S_1, S_2, \ldots, S_n\}$ is the set of places. We use labeled transition rules of the form $S_1^{m_1} S_2^{m_2} \ldots S_n^{m_n} \xrightarrow{\lambda} S_1^{m'_1} S_2^{m'_2} \ldots S_n^{m'_n}$ to indicate that there is a transition t whose label is λ and the flow function is such that t is defined by $F(S_i, t) = m_i$ and $F(t, S_i) = m'_i$ for every $i = 1, \ldots, n$. A marking M is denoted by $S_1^{M(S_1)} S_2^{M(S_2)} \ldots S_n^{M(S_n)}$, which can be viewed as a multiset over Q. Thus N is specified by $(Q, \mathcal{A}, \mathrm{Tr}, M_0)$, where Tr is the set of the labeled transition rules.

The next lemma is due to Jančar, Moller [8] and Jančar, Kučera, Mayr [7].

Lemma 8. *The strong bisimilarity between a marking* M_0 *of a Labeled Petri Net* N *and a finite state process* $F \in \mathbf{FS}$ *is decidable. The corresponding reachability HM property problem is also decidable.*

For the description of our translation, the following three definitions and the lemma, borrowed from [4], are needed.

Definition 4. *A* $\mathrm{CCS}_0^!$ *process* P *is in* concurrent normal form *if* P *is of the form* $(\tilde{m}) \prod_{i \in I} P_i$ *for some names* $\tilde{m}$ *and, for each* $i \in I$*,* P_i *is restriction free and not a composition. We say that* P_i*, for each* $i \in I$*, is a concurrent component of* P*.*

Definition 5. *Suppose that* P *is a restriction free* $\mathrm{CCS}^!$ *process. The* concurrent subprocesses *of* P*, denoted by* $\mathrm{Csub}(P)$*, is defined inductively as follows:* $\mathrm{Csub}(0) \stackrel{\mathrm{def}}{=} \emptyset$*;* $\mathrm{Csub}(\lambda.P') \stackrel{\mathrm{def}}{=} \{\lambda.P'\} \cup \mathrm{Csub}(P')$*;* $\mathrm{Csub}(P' \mid P'') \stackrel{\mathrm{def}}{=} \mathrm{Csub}(P') \cup \mathrm{Csub}(P'')$*;* $\mathrm{Csub}(P' + P'') \stackrel{\mathrm{def}}{=} \{P' + P''\} \cup \mathrm{Csub}(P') \cup \mathrm{Csub}(P'')$*;* $\mathrm{Csub}(!P') \stackrel{\mathrm{def}}{=} \{!P'\} \cup \mathrm{Csub}(P')$*.*

Suppose that P *is a* $\mathrm{CCS}_0^!$ *process in concurrent normal form* $(\tilde{m})P'$ *where* P' *is restriction free, then* $\mathrm{Csub}(P)$ *is defined by* $\mathrm{Csub}(P')$*.*

Definition 6. *The set of the* descendants *of a process* P*, denoted by* $\mathtt{Dscd}(P)$*, is the set of the processes* P' *such that* $P \xrightarrow{\lambda_1} \ldots \xrightarrow{\lambda_n} P'$ *for some* $n \geq 0$ *and* $\lambda_1, \ldots, \lambda_n \in \mathcal{A}$*.*

Lemma 9. *For every process P of $\mathrm{CCS}_0^!$ in concurrent normal form,* $\mathrm{Csub}(P)$ *is finite, and for every $P' \in \mathtt{Dscd}(P)$,* $\mathrm{Csub}(P') \subseteq \mathrm{Csub}(P)$.

Since every $\mathrm{CCS}_0^!$ process can be transformed into a concurrent normal form easily, our translation will only be given on the concurrent normal forms. The key observation is that, in concurrent normal form, no new bound name is produced during the evolution.

Lemma 10. *There is an algorithm such that, given process P of $\mathrm{CCS}_0^!$ in concurrent normal form, it outputs a Labeled Petri Net N_P with the same action set and $P \sim N_P$.*

Proof. Let $\mathrm{Csub}(P) = \{C_i \mid i \in I\}$ and $P = (\tilde{m})(\prod_{i\in I} C_i^{n_i})$. The Labeled Petri Net $N_P = (Q, \mathcal{A}, \rightarrow, M_0)$ is defined as follows. The places and the initial marking are defined as follows:

$$Q \stackrel{\mathrm{def}}{=} \{[C_i] \mid i \in I\}$$
$$M_0 \stackrel{\mathrm{def}}{=} \prod_{i\in I} [C_i^{n_i}]$$

The transition rules are defined inductively:

- If $C_i \stackrel{\lambda}{\rightarrow} \prod_{j\in I} C_j^{n_j}$ then $[C_i] \stackrel{\lambda}{\rightarrow} \prod_{j\in I}[C_j]^{n_j}$ is a rule provided that $\lambda \notin \tilde{m}$.
- If $C_{i_1} \stackrel{a}{\rightarrow} \prod_{j\in I} C_j^{m_j}$ and $C_{i_2} \stackrel{\overline{a}}{\rightarrow} \prod_{j\in I} C_j^{n_j}$ then $[C_{i_1}][C_{i_2}] \stackrel{\tau}{\rightarrow} \prod_{j\in I}[C_j]^{m_j+n_j}$ is a rule.

The remaining work is to check if

$$\{((\tilde{m})(\prod_{i\in I} C_i^{n_i}), \prod_{i\in I}[C_i]^{n_i}) \mid n_i \geq 0 \text{ for } i \in I)\}$$

is a bisimulation. □

Using the same argument, one can show that Lemma 10 also holds for CCS_0^{μ}. Theorem 4 follows directly from Lemma 10 and Lemma 8.

4.3 The Simulation Preorder

One interesting relevant problem is the decidability of the *similarity equivalence (preorder)*. Formally these problems can be stated as follows:

Problem:	$\mathrm{CCS}^! \precsim \mathbf{FS}$
Instance:	A $\mathrm{CCS}^!$ process P and a finite state process F.
Question:	$P \precsim F$?

Problem:	$\mathbf{FS} \precsim \mathrm{CCS}^!$
Instance:	A $\mathrm{CCS}^!$ process P and a finite state process F.
Question:	$F \precsim P$?

It is shown by Kučera and Mayr [11] that for many kinds of process classes, similarity is harder than bisimilarity. But unfortunately, their constructions can not be used to show the similar results for CCS$^!$. So it is worth investigating the decidability of CCS$^!$ $\precsim$ **FS** and **FS** $\precsim$ CCS$^!$. We remark that by the translation provided in Section 4.2 and Theorem 3.2 and Theorem 3.5 of [8], one has the similarity relationships stated in the next theorem.

Theorem 5. **FS** $\precsim \mathrm{CCS}_0^!$, **FS** $\precsim \mathrm{CCS}_0^\mu$, $\mathrm{CCS}_0^!$ $\precsim$ **FS**, *and* $\mathrm{CCS}_0^\mu \precsim$ **FS** *are all decidable.*

The main result of this section is that **FS** $\precsim$ CCS$^!$ is decidable. For the proof, we need a notion called well quasi order [10].

Definition 7. *A* well quasi order $(X, \leq)$ *is a preorder such that, for every infinite sequence* $x_0, x_1, x_2, \ldots$ *in* X*, there exist some indexes* $i < j$ *such that* $x_i \leq x_j$*.*

Next we point out that a syntactic well quasi order can be defined on CCS$^!$. This construction was first pointed out by Busi *et al* in [2]. The following particular definition is from [4].

Definition 8. *The* structural expansion $\preccurlyeq$ *on the* CCS$^!$ *processes is defined inductively as follows:*

- $P \preccurlyeq P$*;*
- $P \preccurlyeq Q$ *whenever* $Q = P \mid R$ *for some* R*;*
- $(m)P \preccurlyeq (m)Q$ *whenever* $P \preccurlyeq Q$*;*
- $P \preccurlyeq Q$ *whenever* $P = P_1 \mid P_2$, $Q = Q_1 \mid Q_2$, $P_1 \preccurlyeq Q_1$ *and* $P_2 \preccurlyeq Q_2$*.*

Intuitively $P \preccurlyeq Q$ means that Q contains at least as many possible individual processes running concurrently as P. It is not hard to see that $\preccurlyeq$ is transitive. Due to the syntactical nature of the definition, the following fact should be clear.

Lemma 11. $\preccurlyeq$ *is decidable.*

The next two technical lemmas, due to Busi *et al.*, are crucial to our proof. The proof of Lemma 12 is straightforward. For a detailed proof of Lemma 13, one may consult [4].

Lemma 12 (Compatibility Lemma). *Suppose that* P,Q *are* CCS$^!$ *processes. If* $P \preccurlyeq Q$ *and* $P \xrightarrow{\lambda} P'$*, then* Q' *exists such that* $Q \xrightarrow{\lambda} Q'$ *and* $P' \preccurlyeq Q'$*.*

Lemma 13 (Expansion Lemma). *For every process* P *of* CCS$^!$, $(\mathtt{Dscd}(P), \preccurlyeq)$ *is a well quasi order.*

The technique of *bisimulation bases*, pioneered by Caucal, is widely used in the proof of the decidability of bisimilarity for many classes of processes. We make use of the *simulation bases*, modified from the bisimulation bases, together with Compatibility Lemma and Expansion Lemma to establish the decidability of **FS** $\precsim$ CCS$^!$. For more on the technique of the bisimulation bases, the reader is referred to [1, 6].

In the following, we use some terminologies borrowed from [6] with slightly modification. We begin with the definition of simulation base.

Definition 9. *Let $\mathcal{R}_1, \mathcal{R}_2 \subseteq \mathbf{FS} \times \mathrm{CCS}^!$. The relation $\mathcal{R}_2$ is called a* (simulation) expansion *of the relation $\mathcal{R}_1$, if for each $(F, P) \in \mathcal{R}_1$ and $\lambda \in \mathcal{A}$, we have:*

- if $F \xrightarrow{\lambda} F'$, then $P \xrightarrow{\lambda} P'$ for some P' such that $(F', P') \in \mathcal{R}_2$

A binary relation $\mathcal{R} \subseteq \mathbf{FS} \times \mathrm{CCS}^!$ is a simulation base *if $\preceq^{\mathcal{R}}$ is an expansion of $\mathcal{R}$, where $\preceq^{\mathcal{R}}$ is the least superset of $\mathcal{R}$ such that $F \preceq^{\mathcal{R}} P'$ whenever it contains $F \preceq^{\mathcal{R}} P$ and $P \preccurlyeq P'$.*

The next lemma tells us that, to prove similarity, it is enough to produce a simulation base instead of producing a complete (and infinite) simulation relation.

Lemma 14. *If $\mathcal{R}$ is a simulation base, then $\preceq^{\mathcal{R}}$ is a simulation, and consequently $\preceq^{\mathcal{R}} \subseteq \precsim$.*

Proof. Let $F \in \mathbf{FS}$, $P \in \mathrm{CCS}^!$, and $F \preceq^{\mathcal{R}} P$. By the definition of $\preceq^{\mathcal{R}}$, it is not hard to see that there exists $Q \in \mathrm{CCS}^!$ such that $F \mathcal{R} Q$ and $Q \preccurlyeq P$. Assume that $F \xrightarrow{\lambda} F'$. Since $\mathcal{R}$ is a simulation base, we have $Q \xrightarrow{\lambda} Q'$ for some Q' such that $F' \preceq^{\mathcal{R}} Q'$. Now that $Q \preccurlyeq P$, by the Compatibility Lemma (Lemma 12), we also have $P \xrightarrow{\lambda} P'$ for some P' such that $Q' \preccurlyeq P'$. Thus $F' \preceq^{\mathcal{R}} Q' \preccurlyeq P'$, which implies $F' \preceq^{\mathcal{R}} P'$.

In the proof of Theorem 6 we need to use expansion trees, which is first adopted in [5].

Definition 10. *An* expansion tree *is a (generally infinite) rooted tree whose nodes are labeled by sets of pairs of $\mathbf{FS} \times \mathrm{CCS}^!$ such that the children of a node are precisely the (finite many) expansions of that node. A* leaf *is a node without any children.*

A branch in an expansion tree is successful *if it is infinite or it finishes with a node labeled by the empty set; otherwise it is* unsuccessful.

The following proposition provides a characterization of the similarity relationship in terms of the expansion tree. It is the 'similarity' version of Fact 8 in [6].

Proposition 1. *$F_0 \precsim P_0$ if and only if the expansion tree rooted at $\{(F_0, P_0)\}$ has a successful branch.*

Proof. The union of all pairs in one successful branch is a simulation.

As the expansion tree is infinite in general, $F_0 \precsim P_0$ can not be decided by constructing the whole tree. However, there is no need to produce a complete simulation relation for our purpose. We can modify the expansion tree by omitting some pairs to produce a finite simulation base for one successful branch.

Definition 11. *A* reduced expansion tree *is a rooted tree whose nodes are labeled by sets of pairs of $\mathbf{FS} \times \mathrm{CCS}^!$ such that the children of a node are precisely the expansions of that node in which a pair (F, P) is omitted whenever there is $P' \preccurlyeq P$ such that the pair (F, P') already appears aa an ancestor node.*

The Successful *and* unsuccessful *branches are defined in the same way.*

Proposition 2. *$F_0 \precsim P_0$ if and only if the reduced expansion tree rooted at $\{(F_0, P_0)\}$ has a successful branch.*

Proof. The union of all the pairs in a successful branch is a simulation base. □

The reduced expansion tree has the following crucial property.

Lemma 15. *The reduced expansion tree is finite.*

Proof. It is enough to show that there is no infinite branch in a reduced expansion tree. It would then follow from König Lemma that the tree is finite. Notice that the union of all the pairs in an infinite branch is infinite. Let's denote it by $\mathcal{R}$. Since the first element of each pair in $\mathcal{R}$ is generated by the same finite state process, there must be some F_c that relates to an infinite umber of P's in $\mathcal{R}$, say, $(F_c, P_1), (F_c, P_2), (F_c, P_3), \ldots$. Without loss of generality we may assume that (F_c, P_i) occurs before (F_c, P_j) in the branch whenever $i < j$. By Lemma 13, there exist i and j with $i < j$ such that $P_i \preccurlyeq P_j$, which contradicts to the omitting rule in the definition of the reduced expansion trees. □

Theorem 6. **FS** $\precsim$ CCS$^!$ *is decidable.*

Proof. By Lemma 11 and Lemma 15, The reduced expansion tree can be constructed effectively. By Proposition 2, the remaining job is to search for a successful branch in the tree. □

5 Remark

We have established several decidability and undecidability results on bisimilarity/similarity checking problems related to CCS$^\mu$ and CCS$^!$. Fig. 5 summarizes the status quo of our understanding of the decidability property for several process calculi.

X	$X \sim X$	$X \sim$ **FS**	**FS** $\precsim X$	$X \precsim$ **FS**
CCS$^!_{\nu-}$	✓	✓	✓	✓
CCS$^\mu_{\nu-}$	✓	✓	✓	✓
CCS$^!_0$	×	✓	✓	✓
CCS$^\mu_0$	×	✓	✓	✓
CCS$^!$	×	×	✓	?
CCS$^\mu$	×	×	?	?
CCS$^{\mathrm{d}\mu}$	×	×	×	×

"$\sim$" for strong bisimilarity
"$\precsim$" for strong similarity

"✓" for decidable
"×" for undecidable
"?" for unknown

Fig. 5. Summary of the Results

The three open problems indicated in Fig. 5 deserve further comment. If the problem **FS** $\precsim$ CCS$^\mu$ would turn out to be undecidable, it would go along with a result of [4] that CCS$^\mu$ is strictly more powerful than CCS$^!$. The problem

$\mathrm{CCS}^{!} \precsim \mathbf{FS}$ is even more interesting. For about two decades, there has been a general belief that simulation is harder than bisimulation [11]. It is a corollary of this belief that the problem should be undecidable. But nothing seems to indicate that a positive answer is unlikely. The importance of the third question really depends on the answers to these two questions. So we will not comment on that.

Acknowledgement We thank Dr. Yuxin Deng for many helpful discussions on the topics of this paper.

References

1. Olaf Burkart, Didier Caucal, Faron Moller, and Bernhard Steffen. Verification on infinite structures. In *Handbook of Process Algebra*, pages 545–623, 2001.
2. N. Busi, M. Gabbrielli, and G. Zavattaro. Replication vs recursive definitions in channel based calculi. In *ICALP'03: Lecture Notes in Computer Science 2719*, pages 133–144, 2003.
3. N. Busi, M. Gabbrielli, and G. Zavattaro. Comparing recursion, replication and iteration in process calculi. In *ICALP'04: Lecture Notes in Computer Science 3142*, pages 307–319, 2004.
4. Y. Fu and H. Lv. On the expressiveness of interaction. *submitted, 2008.*
5. Y. Hirshfeld. Bisimulation trees and the decidability of weak bisimulations. In *ENTCS*, volume 5, pages 2–13, 1997.
6. Petr Jancar and Faron Moller. Techniques for decidability and undecidability of bisimilarity. In *CONCUR '99: Proceedings of the 10th International Conference on Concurrency Theory*, pages 30–45, London, UK, 1999. Springer-Verlag.
7. Petr Jančar, Antonin Kučera, and Richard Mayr. Deciding bisimulation-like equivalences with finite-state processes. *Theor. Comput. Sci.*, 258(1-2):409–433, 2001.
8. Petr Jančar and Faron Moller. Checking regular properties of petri nets. In *CONCUR '95: Proceedings of the 6th International Conference on Concurrency Theory*, pages 348–362, London, UK, 1995. Springer-Verlag.
9. Y. Hirshfeld JS. Christensen and F. Moller. The decidable subsets of ccs. *The Computer Journal*, 37(4):233–242, 1994.
10. Joseph B. Kruskal. The theory of well-quasi-ordering: A frequently discovered concept. *J. Comb. Theory, Ser. A*, 13(3):297–305, 1972.
11. Antonín Kucera and Richard Mayr. Why is simulation harder than bisimulation? In *CONCUR '02: Proceedings of the 13th International Conference on Concurrency Theory*, pages 594–610, London, UK, 2002. Springer-Verlag.
12. Antonín Kučera and Petr Jančar. Equivalence-checking on infinite-state systems: Techniques and results. *Theory Pract. Log. Program.*, 6(3):227–264, 2006.
13. Richard Mayr. Process rewrite systems. *Theor. Comput. Sci.*, 230(1-2):263, 2000.
14. R. Milner. *Communication and concurrency.* Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1989.
15. Marvin L. Minsky. *Computation: finite and infinite machines.* Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1967.
16. H. Rogers. *Theory of Recursive Functions and Effective Computability.* McGraw-Hill, 1967.
17. J. Srba. *Roadmap of Infinite results*, volume Vol 2: Formal Models and Semantics. World Scientific Publishing Co., 2004.