Decidability of Behavioral Equivalences in Process Calculi with Name Scoping*

Chaodong He, Yuxi Fu, Hongfei Fu

BASICS, Department of Computer Science
Shanghai Jiao Tong University, Shanghai 200240, China
MOE-MS Key Laboratory for Intelligent Computing and Intelligent Systems

Abstract. Local channels and their name scoping rules play a significant role in the study of the expressiveness of process calculi. The paper contributes to the understanding of the expressiveness in the context of CCS by studying the decidability issues of the bisimilarity/similarity checking problems. The strong bisimilarity for a pair of processes in the calculi with only static local channels is shown Π^0_1-complete. The strong bisimilarity between those processes and the finite state processes is proved decidable. The strong similarity between the finite state processes and the processes without name-passing capability is also shown decidable.

1 Introduction

Process calculi are usually Turing complete. The known proofs of Turing completeness share the same guideline that counting is represented as the nesting of suitable components [4, 6, 20]. In the name-passing calculi [24, 26], the encodings of counter [4, 6] depend on the existence of *local channels* and some degrees of *name-passing capabilities*. In the setting of CCS-like calculi, there are several Turing complete variants in which local channels are provided by the localization operation while name-passing capabilities are partly obtained by an explicit operation such as *parametric definition* [23, 11] or *relabeling* [22], or by an implicit *dynamic-scoping* recursion [28, 4].

A fundamental problem in the area of system verification is that of *equivalence (or preorder) checking* [3]. In concurrency theory these are the problems of deciding whether two given processes are behaviorally equal, or whether one process is behavioral close to the other. Among these equivalences (or preorders), bisimilarity (or similarity) plays a prominent role.

This paper explores the decidability issues of bisimilarity/similarity checking problems for various subcalculi of CCS classified by different name scoping rules, in which the capability of producing and manipulating local channels becomes weaker and weaker. These decidability results contribute to the understanding of the way productions and mobilities of local channels affect the expressiveness.

* The work is supported by NSFC (60873034, 61033002).

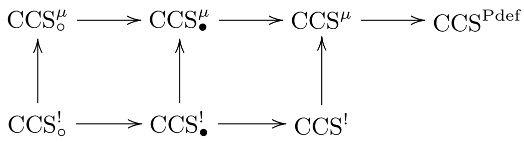

Fig. 1. CCS Hierarchy

The seven subcalculi of CCS studied in this paper are given in Fig. 1. In the diagram an arrow '$\longrightarrow$' indicates the sub-language relationship. These seven subcalculi are further divided into four classes in which the scoping rules of local channel names are weakened gradually.

The first class contains $\mathrm{CCS}^{\mathrm{Pdef}}$, the full CCS with parametric definition (but without relabeling), which is known to be Turing complete [11]. In $\mathrm{CCS}^{\mathrm{Pdef}}$ process copies can be nested at arbitrary depth by the name-passing capability offered by parametric definition. Turing completeness implies that all behavioral equivalences and preorders for $\mathrm{CCS}^{\mathrm{Pdef}}$ are undecidable.

The second class contains CCS^{μ} and $\mathrm{CCS}^{!}$. These two subcalculi have the power of producing new local channels but do not have the power of passing names around. In both models the infinite behaviors are specified by (static scoping) recursion and replication respectively. They are not Turing complete because they are not expressive enough to define the process *Counter* in the sense of Section 2.5 of [10]. For the readers unfamiliar with the static scoping recursion, we give the following illustration. Static scoping and dynamic scoping are different ways of manipulating local names when unfolding recursions [11, 10]. When a process is defined as $P \stackrel{\mathrm{def}}{=} \mu X.(a \,|\, (a)(\overline{a} \,|\, X))$, the static scoping requires that the local a and the global a must be distinguished before unfolding. That is, $\mu X.(a \,|\, (a)(\overline{a} \,|\, X))$ is understood the same as $\mu X.(a \,|\, (a')(\overline{a'} \,|\, X))$. The recursion used in [4, 6] admits dynamic scoping, meaning that P should be understood as $a \,|\, (a)(\overline{a} \,|\, a \,|\, (a)(\overline{a} \,|\, P))$, which induces the infinite computation $P \xrightarrow{\tau} a \,|\, (a)(\mathbf{0} \,|\, \mathbf{0} \,|\, (a)(\overline{a} \,|\, P)) \xrightarrow{\tau} \ldots$. It is pointed out in [11] that the dynamic scoping recursion can be encoded via parametric definition. For this reason we shall only consider the parametric definition in this paper.

The third class contains $\mathrm{CCS}^{\mu}_{\bullet}$ and $\mathrm{CCS}^{!}_{\bullet}$. They are the subcalculi of CCS^{μ} and $\mathrm{CCS}^{!}$ which have only static local names. Here 'static' means that no local channels can be produced during the evolution of a process. In these situations, localizations can only act as the outermost constructors, and processes in $\mathrm{CCS}^{\mu}_{\bullet}$ and $\mathrm{CCS}^{!}_{\bullet}$ can be assumed in the form $(\widetilde{a})P$ where the inner process P is localization-free. In this paper the word 'static' is only used in the context of 'static local names' in order to avoid confusion with the 'static scoping recursion'.

The fourth class contains $\mathrm{CCS}^{\mu}_{\circ}$ and $\mathrm{CCS}^{!}_{\circ}$, where the localization operator are removed completely. For those subcalculi, strong bisimilarity is decidable [7].

We will use notation $\mathcal{L}_1 \sim \mathcal{L}_2$ (or $\mathcal{L}_1 \precsim \mathcal{L}_2$) to indicate the problem of checking strong bisimilarity (or strong similarity) between an $\mathcal{L}_1$ process and an $\mathcal{L}_2$

$\mathcal{L}$	$\mathcal{L} \sim \mathcal{L}$	$\mathcal{L} \sim \mathbf{FS}$	$\mathbf{FS} \precsim \mathcal{L}$	$\mathcal{L} \precsim \mathbf{FS}$
$\text{CCS}^{!}_{\circ}$	✓ [7]	✓ [7]	?	?
$\text{CCS}^{\mu}_{\circ}$	✓ [7]	✓ [7]	?	?
$\text{CCS}^{!}_{\bullet}$	?	?	?	?
$\text{CCS}^{\mu}_{\bullet}$	?	?	?	?
$\text{CCS}^{!}$	?	?	?	?
CCS^{μ}	?	?	?	?
CCS^{Pdef}	× [4, 11]	× [4, 11]	× [4, 11]	× [4, 11]

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

"✓": known decidable
"×": known undecidable
"?": unknown

Fig. 2. Problems to Explore

process. These problems are indicated by the question marks in the table of Fig. 2. The notation **FS** stands for the class of the finite state processes. The contributions of this paper are summarized as follows.

- We show the undecidability (Π^0_1-hardness) of $\text{CCS}^{\mu}_{\bullet} \sim \text{CCS}^{\mu}_{\bullet}$ by a reduction from the halting problem of Minsky Machine. The relevant technique is called 'Defender's Forcing' [14, 18], which is widely used in undecidability proofs for bisimilarity checking. Typical examples of this technique can also be found in [17, 18]. The reduction is then modified to show the undecidability (Π^0_1-hardness) of $\text{CCS}^{!}_{\bullet} \sim \text{CCS}^{!}_{\bullet}$. This resolves the four problems in the first column of the table.
- Busi, Gabbrielli and Zavattaro establish in [5] the undecidability (Σ^0_1-hardness) of the weak bisimilarity of $\text{CCS}^{!}$. By modifying the proof of Busi *et al.*, $\text{CCS}^{!} \sim \mathbf{FS}$ is shown undecidable (Π^0_1-hard), which immediately implies the undecidability (Π^0_1-hardness) of $\text{CCS}^{\mu} \sim \mathbf{FS}$.
- By constructing a translation from $\text{CCS}^{!}_{\bullet}$ to the Labeled Petri Net, we demonstrate the decidability of $\text{CCS}^{!}_{\bullet} \sim \mathbf{FS}$, $\text{CCS}^{!}_{\bullet} \precsim \mathbf{FS}$ and $\mathbf{FS} \precsim \text{CCS}^{!}_{\bullet}$, making use of Jančar and Moller's decidability result [16] on the Labeled Petri Nets. The same approach applies to $\text{CCS}^{\mu}_{\bullet}$.
- We show that $\mathbf{FS} \precsim \text{CCS}^{!}$ is decidable. The technique used in the proof is *simulation base*, originated from the technique of *bisimulation base* pioneered by Caucal and widely used in decidability proofs of bisimilarity. Our proof also makes use of expansion tree presented in [17] and the *well-structured transition system* [8] for $\text{CCS}^{!}$ [4, 10]. In literature there are examples of formalisms [19] in which bisimilarity is decidable while similarity is not. We are not aware of any examples showing that the opposite situation happens. This result is more or less surprising.

The finite branching property guarantees that the bisimilarity can be approximated in the sense that $P \not\sim Q$ if and only if $P \not\sim_n Q$ for some n. The approximation can also be applied to the similarity relation. It necessarily implies that all the problems in Fig. 2 are actually in Π^0_1. So we only need to show Π^0_1-hardness to get Π^0_1-completeness. We remark that a relation $R(x)$ is in Σ^0_1 (resp. Π^0_1) in arithmetic hierarchy if it can be expressed by $\exists y.S(x, y)$ (resp.

$$\text{Choice}\ \frac{}{\sum_{i=1}^{n}\lambda_i.E_i \xrightarrow{\lambda_i} E_i} \qquad \text{Composition}\ \frac{E \xrightarrow{\lambda} E'}{E \mid F \xrightarrow{\lambda} E' \mid F} \quad \frac{E \xrightarrow{l} E' \quad F \xrightarrow{\overline{l}} F'}{E \mid F \xrightarrow{\tau} E' \mid F'}$$

$$\text{Localization}\ \frac{E \xrightarrow{\lambda} E' \quad a \text{ not appear in } \lambda}{(a)E \xrightarrow{\lambda} (a)E'} \qquad \text{Fixpoint}\ \frac{E\{\mu X.E/X\} \xrightarrow{\lambda} E'}{\mu X.E \xrightarrow{\lambda} E'}$$

Fig. 3. Semantics of CCS^{μ}

$\forall y.S(x,y))$ for some decidable relation $S(x,y)$. Clearly $R(x)$ is in Σ_1^0 if and only if its complement is in Π_1^0.

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 $\text{CCS}^{!}$ processes. Section 4 considers the problem of deciding the strong bisimilarity/similarity between a $\text{CCS}^{!}/\text{CCS}^{\mu}$ process and a finite state process. Section 5 gives concluding remarks.

Most proofs and technical details are omitted. See [13] for complete coverage.

2 Basic Definition and Notation

To describe the interactions between systems, we need channel 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 $l, \ldots$. The set of the action labels $\mathcal{A} = \mathcal{N} \cup \overline{\mathcal{N}} \cup \{\tau\}$ is ranged over by λ. To define the fixpoint operator and we need a set of *process variables* $\mathcal{V}$ ranged over by X, Y, Z.

The set $\mathcal{E}_{\text{CCS}^{\mu}}$ of CCS^{μ} terms is generated by the following grammar.

$$E ::= \mathbf{0} \quad \mid \quad X \quad \mid \quad \sum_{i=1}^{n} \lambda_i.E_i \quad \mid \quad E \mid E' \quad \mid \quad (a)E \quad \mid \quad \mu X.E.$$

A name a appeared in a localization term $(a)E$ is *local*. A name is *global* if it is not local. The variable X in the fixpoint term $\mu X.E$ is *bound*. A variable is *free* if it is not bound. A CCS^{μ} term containing no free variables is a CCS^{μ} *process*.

In $\mu X.E$ it is not required that X be guarded in E because unguarded recursion can be encoded by guarded recursion in CCS^{μ} [10]. With guarded recursion and guarded choice $\sum_{i=1}^{n} \lambda_i.E_i$, finite branching property is guaranteed. Once unguarded recursion is admitted, replication $!P$ can be defined by the recursion $\mu X.(X \mid P)$.

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

Standard notations and conventions in process calculi will be used throughout the paper. The inactive process $\mathbf{0}$ is omitted in most occasions. For instance $a.b.\mathbf{0}$ is abbreviated to $a.b$. A finite sequence (or set) of names $a_1, \ldots, a_n$ is often abbreviated to $\widetilde{a}$. The guarded choice term $\sum_{i=1}^{n} \lambda_i.E_i$ is usually written as $\lambda_1.E_1 + \cdots + \lambda_n.E_n$. Processes are not distinguished syntactically up to the commutative monoid generated by '$+$' and '$\,|\,$'. We shall write $\prod_{i=1}^{n} P_i$ for $P_1 \,|\, \ldots \,|\, P_n$. The notation '$\equiv$' is used to indicate syntactic congruence. We shall write $\mathcal{P}_{\mathcal{L}}$ for the set of the processes definable in $\mathcal{L}$. The set of the *derivatives* of a process P, denoted by $\texttt{Drv}(P)$, is the set of the processes P' such that $P \xrightarrow{\lambda_1} \cdots \xrightarrow{\lambda_n} P'$ for some $n \geq 0$ and $\lambda_1, \ldots, \lambda_n \in \mathcal{A}$.

CCS$^!$ is obtained from CCS$^\mu$ by using the *replication* instead of the fixpoint operation. The grammar is defined as follows:

$$P ::= \mathbf{0} \quad | \quad \sum_{i=1}^{n} \lambda_i.P_i \quad | \quad P \,|\, P' \quad | \quad (a)P \quad | \quad !P.$$

The operational semantics of the replication stated below is from [4, 5], which enjoys the finite branching property.

$$\textsf{Replication} \; \frac{P \xrightarrow{\lambda} P'}{!P \xrightarrow{\lambda} P' \,|\, !P} \quad \frac{P \xrightarrow{l} P' \quad P \xrightarrow{\bar{l}} P''}{!P \xrightarrow{\tau} P' \,|\, P'' \,|\, !P}$$

The advantage of the replication is that one could give a first order presentation of CCS. There is no need for process variables. This is why the above grammar and rules are defined on the set of the processes, not on the set of the terms.

A binary relation $\mathcal{R}$ on $\mathcal{P}_{\mathcal{L}}$ is a *strong simulation* if, for each pair $(P, Q) \in \mathcal{R}$, 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.

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 $(\mathcal{P}_{\mathcal{L}}, \mathcal{A}, \longrightarrow)$ is perceived as a game-board. During the play the current position is described by a pair of states $(P_1, P_{-1}) \in \mathcal{P}_{\mathcal{L}} \times \mathcal{P}_{\mathcal{L}}$. 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 $P_i' \in \mathcal{P}_{\mathcal{L}}$ such that $P_i \xrightarrow{\lambda} P_i'$.
2. The defender responds by choosing some $P_{-i}' \in \mathcal{P}_{\mathcal{L}}$ 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 position (P, Q) if and only if $P \sim Q$.

3 Undecidability of Strong Bisimilarity

This section aims at the undecidability of $\mathrm{CCS}^{\mu} \sim \mathrm{CCS}^{\mu}$ and $\mathrm{CCS}^{!} \sim \mathrm{CCS}^{!}$. In fact, by many-one reductions from the halting problem of Minsky Machines, it can be shown that both $\mathrm{CCS}^{\mu}_{\bullet} \sim \mathrm{CCS}^{\mu}_{\bullet}$ and $\mathrm{CCS}^{!}_{\bullet} \sim \mathrm{CCS}^{!}_{\bullet}$ are Π^0_1-complete.

Two-register *Minsky Machine* is a well-known Turing complete computational model [25]. 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 $\{(1 : I_1), (2 : I_2), \ldots, (n-1 : I_{n-1}), (n : \mathtt{halt})\}$. For each $i \in \{1, \ldots, n-1\}$, the i-th instruction may be in one of two forms:

- $(i : \mathit{Succ}(r_j))$: The instruction adds 1 to the content of the register r_j and $i + 1$ becomes the value of the program counter.
- $(i : \mathit{Decjump}(r_j, s))$: If the content 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. 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 halting problem of Two-register Minsky Machines, whose undecidability is well-known, is formally stated as follows:

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

Lemma 1. HaltingMinskyMachine *is undecidable. It is Σ^0_1-complete in the arithmetic hierarchy.*

If a process calculus $\mathcal{L}$ is able to encode the computation of a Minsky Machine faithfully, undecidability of $\mathcal{L} \sim \mathcal{L}$ can be obtained by a straightforward reduction from HaltingMinskyMachine, which confirms that the i-th Minsky Machine $\mathbb{R}_i$ does not halt if and only if the interpretation $P_{\mathbb{R}_i}$ of $\mathbb{R}_i$ is strongly bisimilar to $!\tau$. Recall that there is no such reduction for any calculi in Fig. 1 except for $\mathrm{CCS}^{\mathrm{Pdef}}$.

In the rest of this section, we outline the reductions that demonstrate the undecidability of $\mathrm{CCS}^{\mu}_{\bullet} \sim \mathrm{CCS}^{\mu}_{\bullet}$ and $\mathrm{CCS}^{!}_{\bullet} \sim \mathrm{CCS}^{!}_{\bullet}$.

3.1 Undecidability of $\mathrm{CCS}^{\mu}_{\bullet} \sim \mathrm{CCS}^{\mu}_{\bullet}$

The idea is to construct a $\mathrm{CCS}^{\mu}_{\bullet}$ process which models a given Minsky Machine $\mathbb{R}$ in a nondeterministic fashion. The encoding is nondeterministic because it introduces unfaithful computations which do not follow the expected behavior of

$\mathbb{R}$. Two slightly modified copies of the constructed process are taken for bisimilarity checking. The modifications guarantee that in the bisimulation game, whenever the attacker takes the 'unfaithful' move at some round, the defender have the ability to punish the attacker by moving to a pair of trivially bisimilar states. Thus the attacker are 'forced' to take the 'faithful' move at each round and the defender will lose the game if $\mathbb{R}$ ever halts. This technique is known as 'Defender's Forcing' [14, 18].

The construction is motivated by a construction in [17]. For convenience constant definitions are used instead of μ-operations. Since localization operator must not appear underneath any μ-operations, no confusion will arise. Two slightly modified copies are given directly instead of describing the encoding in advance.

Let $\mathbb{R}$ be an instance of HaltingMinskyMachine whose instruction set is $\{(1 : I_1), (2 : I_2), \ldots, (n-1 : I_{n-1}), (n : \mathtt{halt})\}$. Without using the localization operator the processes $\{P_i\}_{i=1}^n$ and $\{Q_i\}_{i=1}^n$ are defined as follows:

- $P_i \stackrel{\text{def}}{=} \overline{\mathsf{inc}_j}.P_{i+1}$ and $Q_i \stackrel{\text{def}}{=} \overline{\mathsf{inc}_j}.Q_{i+1}$ if the i-th instruction is $(i : Succ(r_j))$.
- If the i-th instruction is $(i : Decjump(r_j, s))$, then let

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

- $P_n \stackrel{\text{def}}{=} \overline{\mathsf{halt}}.\mathbf{0}$ and $Q_n \stackrel{\text{def}}{=} \mathbf{0}$ for the n-th instruction $(n : \mathtt{halt})$.

The processes $\{P_i\}_{i=1}^n$ and $\{Q_i\}_{i=1}^n$ are two families of slightly different processes that interpret the instructions of $\mathbb{R}$. Special attention should be paid to the gadget $\overline{\mathsf{ff}}.z.Q_s$ (or $\overline{\mathsf{ff}}.z.P_s$) in the defining equation of P_i (or Q_i) for instruction $(i : Decjump(r_j, s))$. This gadget is designed to 'force' the attacker to stick to the faithful moves. Also notice that the only asymmetry between P_i's and Q_i's is that P_n can perform a special action $\overline{\mathsf{halt}}$ whereas Q_n cannot.

The processes $PseudoCounter_j(k)$, for $j \in \{1, 2\}$, introduced below are used to partially model the registers of $\mathbb{R}$.

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

where O_j and C_j are defined as follows without using the localization operation:

$$O_j \stackrel{\text{def}}{=} \mathsf{inc}_j.(C_j \mid O_j) + \mathsf{zero}_j.\mathsf{tt}.O_j,$$
$$C_j \stackrel{\text{def}}{=} \mathsf{dec}_j.0 + \mathsf{zero}_j.\mathsf{ff}.C_j.$$

The process $PseudoCounter_j$'s are the weak forms of the counter, for they lack the ability to zero-test — they can make a 'zero' move while the actual value of the counters are positive. However $PseudoCounter_j$'s are good enough for the purpose of deriving the undecidability results we want.

Finally every configuration of $\mathbb{R}$ is modeled by the following two slightly different processes.

$$\begin{aligned} Config_P(i; c_1, c_2) &\stackrel{\text{def}}{=} (\widetilde{\mathsf{inc}})(\widetilde{\mathsf{dec}})(\widetilde{\mathsf{zero}})(\mathsf{tt})(\mathsf{ff}) \\ &\quad (P_i \mid PseudoCounter_1(c_1) \mid PseudoCounter_2(c_2)), \\ Config_Q(i; c_1, c_2) &\stackrel{\text{def}}{=} (\widetilde{\mathsf{inc}})(\widetilde{\mathsf{dec}})(\widetilde{\mathsf{zero}})(\mathsf{tt})(\mathsf{ff}) \\ &\quad (Q_i \mid PseudoCounter_1(c_1) \mid PseudoCounter_2(c_2)). \end{aligned}$$

The correctness of the above encoding is guaranteed by Lemma 2, Lemma 3, and Lemma 4, which eventually lead to Theorem 1.

Lemma 2. *Let $(i; c1, c2)$ be a configuration of $\mathbb{R}$ and $(i : Succ(r_j))$ be the i-th instruction. Then there is a unique continuation of the bisimulation game from the pair of processes $Config_P(i; c_1, c_2)$ and $Config_Q(i; c_1, c_2)$ such that, after one round, the players reach the pair $Config_P(i; c_1', c_2')$ and $Config_Q(i; c_1', c_2')$ 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 $(i : Decjump(r_j, s))$ be the i-th instruction. Assume that a bisimulation game is played from the pair $Config_P(i; c_1, c_2)$ and $Config_Q(i; c_1, c_2)$. 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 $Config_P(s; c_1, c_2)$ and $Config_Q(s; c_1, c_2)$.
(b) If $c_j > 0$ and the attacker chooses the τ action induced by the synchronization via channel dec_j, then the defender has a way to continue the game such that, after two rounds, $Config_P(i; c_1', c_2')$ and $Config_Q(i; c_1', c_2')$ 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 zero_j, then there is a way for the defender to win the game.

Lemma 4. *The execution of $\mathbb{R}$ from the configuration $(1; 0, 0)$ terminates if and only if $Config_P(1; 0, 0) \not\sim Config_Q(1; 0, 0)$.*

Theorem 1. *Both $\mathrm{CCS}^{\mu}_{\bullet} \sim \mathrm{CCS}^{\mu}_{\bullet}$ and $\mathrm{CCS}^{\mu} \sim \mathrm{CCS}^{\mu}$ are Π^0_1-complete.*

3.2 Undecidability of $\mathrm{CCS}^{!}_{\bullet} \sim \mathrm{CCS}^{!}_{\bullet}$

The result established in Section 3.1 does not immediately imply the same result for $\mathrm{CCS}^{!}/\mathrm{CCS}^{!}_{\bullet}$. A well known fact is that recursion can be turned into replication [26, 11] by the encoding $[\![\_]\!]$ whose nontrivial part is given by $[\![X_i]\!] = \overline{a_i}.\mathbf{0}$ and $[\![\mu X_i.E]\!] = (a_i)(\overline{a_i} \mid !a_i.[\![E]\!])$, where names a_i's are fresh. However this encoding does not give rise to a strong bisimulation. Another problem is that an encoding from CCS^{μ} to $\mathrm{CCS}^{!}$ would not always produce an encoding from $\mathrm{CCS}^{\mu}_{\bullet}$ to $\mathrm{CCS}^{!}_{\bullet}$ automatically since they introduce additional local names.

Undecidability of $\mathrm{CCS}^{!}_{\bullet} \sim \mathrm{CCS}^{!}_{\bullet}$ does not rely on the existence of such an encoding. The basic idea and the construction in Section 3.1 can be repeated with subtle modifications. The intuition of the next encoding is to interpret every instruction of a Minsky Machine $\mathbb{R}$ by a process of the form $!addr.opr$, where $addr$ should be understood as the address of the instruction and opr the operation of the instruction. The difficulty is to guarantee that only a finite number of local channels are necessary. In the following definition $2n$ extra static local channels $\{\mathsf{inst}^i_P, \mathsf{inst}^i_Q\}_{i=1}^n$ are used.

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

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

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

$$P_i \stackrel{\text{def}}{=} !\mathsf{inst}^i_P.(\overline{\mathsf{dec}_j}.d.\overline{\mathsf{inst}^{i+1}_P} + \overline{\mathsf{zero}_j}.(\overline{\mathsf{tt}}.\tau.\tau.z.\overline{\mathsf{inst}^s_P} + \overline{\mathsf{ff}}.\mathsf{ack}.z.\overline{\mathsf{inst}^s_Q})),$$
$$Q_i \stackrel{\text{def}}{=} !\mathsf{inst}^i_Q.(\overline{\mathsf{dec}_j}.d.\overline{\mathsf{inst}^{i+1}_Q} + \overline{\mathsf{zero}_j}.(\overline{\mathsf{tt}}.\tau.\tau.z.\overline{\mathsf{inst}^s_Q} + \overline{\mathsf{ff}}.\mathsf{ack}.z.\overline{\mathsf{inst}^s_P})).$$

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

$$P_n \stackrel{\text{def}}{=} !\mathsf{inst}^n_P.\overline{\mathsf{halt}}.\mathbf{0}, \qquad Q_n \stackrel{\text{def}}{=} !\mathsf{inst}^n_Q.\mathbf{0}.$$

In the following modification of $PseudoCounter_j(k)$, $\{\mathsf{m}_j\}_{j=1}^2$ and ack are the only extra local channels introduced.

$$PseudoCounter_j(k) \stackrel{\text{def}}{=} \underbrace{C_j \mid C_j \mid \ldots \mid C_j}_{k} \mid O_j \mid !\mathsf{m}_j.\overline{\mathsf{ack}}.C_j,$$

where $O_j \stackrel{\text{def}}{=} !(\mathsf{inc}_j.C_j + \mathsf{zero}_j.\mathsf{tt})$, and $C_j \stackrel{\text{def}}{=} \mathsf{dec}_j + \mathsf{zero}_j.\mathsf{ff}.\overline{\mathsf{m}_j}$. When zero_j is triggered on some C_j, channel m_j is used to require a new copy of C_j from the resource $!\mathsf{m}_j.\overline{\mathsf{ack}}.C_j$, and after that, the channel ack ais used to inform the process that triggers the action zero_j. Such treatment will make the whole system sequential. As a side-effect it will take two more computation steps when the zero-testing is unfaithfully chosen by the attacker, and for the defender, two extra τ's are introduced into the definition of P_i and Q_i. The configuration $(i; c_1, c_2)$ of $\mathbb{R}$ is interpreted by the following two processes:

$$Config^!_P(i; c_1, c_2) \stackrel{\text{def}}{=} (\widetilde{\mathsf{inst}})(\widetilde{\mathsf{inc}})(\widetilde{\mathsf{dec}})(\widetilde{\mathsf{zero}})(\widetilde{\mathsf{m}})(\mathsf{tt})(\mathsf{ff})(\mathsf{ack}) \left(\overline{\mathsf{inst}^i_P} \mid \prod_{i=1}^n P_i \mid \prod_{i=1}^n Q_i \mid \prod_{j=1}^2 PseudoCounter_j(c_j)\right),$$

$$Config^!_Q(i; c_1, c_2) \stackrel{\text{def}}{=} (\widetilde{\mathsf{inst}})(\widetilde{\mathsf{inc}})(\widetilde{\mathsf{dec}})(\widetilde{\mathsf{zero}})(\widetilde{\mathsf{m}})(\mathsf{tt})(\mathsf{ff})(\mathsf{ack}) \left(\overline{\mathsf{inst}^i_Q} \mid \prod_{i=1}^n P_i \mid \prod_{i=1}^n Q_i \mid \prod_{j=1}^2 PseudoCounter_j(c_j)\right).$$

Using the same argument as in Section 3.1 we can prove the following.

Theorem 2. *Both* $\mathrm{CCS}^!_\bullet \sim \mathrm{CCS}^!_\bullet$ *and* $\mathrm{CCS}^! \sim \mathrm{CCS}^!$ *are* Π^0_1*-complete.*

4 Strong (Bi)similarity on Finite State Processes

We investigate in this section the decidability of strong bisimilarity/similarity between a $\mathrm{CCS}^!/\mathrm{CCS}^\mu$ process and a finite state process.

4.1 Undecidability of $\mathbf{CCS^! \sim FS}$

The general problem $\mathrm{CCS}^! \sim \mathbf{FS}$ is undecidable. This result depends on the construction of Busi *et al* in Section 3 of [5], where Minsky Machines are encoded by $\mathrm{CCS}^!$ processes in a nondeterministic fashion. Using this encoding, one can show that if a Minsky Machine $\mathbb{R}$ does not halt, the encoding of $\mathbb{R}$ is a $\mathrm{CCS}^!$ process strongly bisimilar to $!\tau$, which cannot perform any visible actions and is divergent in every computation branch. If $\mathbb{R}$ does halt, the encoding of $\mathbb{R}$ has at least one divergent computation branch. This fact leads to Theorem 3.

Theorem 3. *The strong bisimilarity between a process $P \in \mathcal{P}_{\mathrm{CCS}^!}$ (or $P \in \mathcal{P}_{\mathrm{CCS}^\mu}$) and a fixed finite state process $F \in \mathcal{P}_{\mathbf{FS}}$ is Π^0_1-complete.*

It is worth noting that Theorem 1 of [5] confirms that the Minsky Machine $\mathbb{R}$ halts if and only if $\mathbb{R}$ is interpreted as a $\mathrm{CCS}^!$ process P satisfying $P \approx \tau.P + \overline{\mathsf{halt}}$, which establishes the Σ^0_1-hardness of the weak bisimilarity checking problem of $\mathrm{CCS}^!$. An interesting question is how to establish the Π^0_1-hardness of $\mathrm{CCS}^! \approx \mathbf{FS}$. It is widely believed that checking weak bisimilarity is harder than checking the strong bisimilarity. However the above construction does not immediately offer an answer to the latter problem.

4.2 Decidability of $\mathbf{CCS^!_\bullet \sim FS}$

Although both $\mathrm{CCS}^! \sim \mathbf{FS}$ and $\mathrm{CCS}^\mu \sim \mathbf{FS}$ are undecidable in the general case, their restricted versions, $\mathrm{CCS}^!_\bullet \sim \mathbf{FS}$ and $\mathrm{CCS}^\mu_\bullet \sim \mathbf{FS}$, turn out to be decidable. These results are motivated by the following observations. Suppose $P \in \mathcal{P}_{\mathrm{CCS}^!_\bullet}$ or $P \in \mathcal{P}_{\mathrm{CCS}^\mu_\bullet}$. We may assume that P is of the form $(\tilde{a}) \prod_{i\in I} P_i$ in which $\widetilde{a}$ are all the local names of P and every P_i is localization free and is not a composition. We call $(\tilde{a}) \prod_{i\in I} P_i$ a *concurrent normal form* of P, and every P_i a *concurrent component* of P. The key opoint is that no local names can be produced during the evolution of P, and the number of the possible concurrent components of all derivatives of P must be finite.

Based on the above observations, a strongly bisimilar encoding from $\mathrm{CCS}^!_\bullet$ (or $\mathrm{CCS}^\mu_\bullet$) to the Labeled Petri Net is constructed. With the help of the results of Jančar *et al.* [16], we know that the same problem for the Labeled Petri Net is decidable. Hence the decidability of $\mathrm{CCS}^!_\bullet \sim \mathbf{FS}$ and $\mathrm{CCS}^\mu_\bullet \sim \mathbf{FS}$.

Definition 1. *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 $M \stackrel{t}{\longrightarrow}$, 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 $M \stackrel{t}{\longrightarrow} M'$, where $M'(p) = M(p) - F(p,t) + F(t,p)$ for all $p \in Q$. For each $\lambda \in \mathcal{A}$, we write $M \stackrel{\lambda}{\longrightarrow}$, respectively $M \stackrel{\lambda}{\longrightarrow} M'$ to mean that $M \stackrel{t}{\longrightarrow}$, respectively $M \stackrel{t}{\longrightarrow} M'$ for some t with $L(t) = \lambda$.*

In the above definition $\mathcal{A}$ is the set of the action labels. A Labeled Petri Net N can be viewed as a labeled transition system $(\mathbb{M}, \mathcal{A}, \longrightarrow)$ with $\mathbb{M}$ being the markings of N. Strong bisimilarity is defined accordingly. Suppose $Q = \{S_1, S_2, \ldots, S_n\}$ is the finite set of places. Labeled transition rules of the form $S_1^{m_1} S_2^{m_2} \ldots S_n^{m_n} \stackrel{\lambda}{\longrightarrow} S_1^{m'_1} S_2^{m'_2} \ldots S_n^{m'_n}$ are used to indicate that there is a transition t whose label is λ and the flow function for 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 and Moller [16].

Lemma 5. *The strong bisimilarity between a marking M_0 of a Labeled Petri Net N and a finite state process $F \in \mathcal{P}_{\mathbf{FS}}$ is decidable.*

To describe the encoding from $\mathrm{CCS}^!_\bullet$ to the Labeled Petri Net, we need the following definitions and lemma, borrowed from [10].

Definition 2. *Suppose the $\mathcal{P}_{\mathrm{CCS}^!}$ process P does not contain any local names. The* concurrent subprocesses *of P, notation $\mathrm{Csub}(P)$, is defined inductively by*

$$\begin{aligned}
\mathrm{Csub}(\mathbf{0}) &\stackrel{\text{def}}{=} \emptyset, \\
\mathrm{Csub}(P' \mid P'') &\stackrel{\text{def}}{=} \mathrm{Csub}(P') \cup \mathrm{Csub}(P''), \\
\mathrm{Csub}(\sum_{i=1}^{n} \lambda_i.P_i) &\stackrel{\text{def}}{=} \{\sum_{i=1}^{n} \lambda_i.P_i\} \cup \bigcup_{i \in I} \mathrm{CSub}(P_i), \\
\mathrm{Csub}(!P') &\stackrel{\text{def}}{=} \{!P'\} \cup \mathrm{Csub}(P').
\end{aligned}$$

Clearly if $P \equiv (a)P'$ is in concurrent normal form, then $\mathrm{Csub}(P) \stackrel{\text{def}}{=} \mathrm{Csub}(P')$.

Lemma 6. *For every process P of $\mathrm{CCS}^!_\bullet$ in concurrent normal form, $\mathrm{Csub}(P)$ is finite, and for every $P' \in \mathtt{Drv}(P)$, $\mathrm{Csub}(P') \subseteq \mathrm{Csub}(P)$.*

By letting $\mathrm{Csub}(\mu X.E) \stackrel{\text{def}}{=} \{\mu X.E\} \cup \mathrm{Csub}(E\{\mu X.E/X\})$, the counterpart of Lemma 6 for $\mathrm{CCS}^\mu_\bullet$ can be established. Now an encoding from the concurrent normal forms of $\mathrm{CCS}^!_\bullet$ or $\mathrm{CCS}^\mu_\bullet$ to the Labeled Petri Net is given in the proof of Lemma 7.

Lemma 7. *There is an algorithm such that, given process $P \in \mathcal{P}_{\mathrm{CCS}^!_\bullet}$ (or $P \in \mathcal{P}_{\mathrm{CCS}^\mu_\bullet}$) in concurrent normal form, it outputs a Labeled Petri Net N_P with the same set of the action labels and $P \sim N_P$.*

Proof. Let $\mathrm{Csub}(P) = \{C_i \mid i \in I\}$ and $P = (\tilde{a})(\prod_{i\in I} C_i^{n_i})$. The Labeled Petri Net $N_P = (Q, \mathcal{A}, \longrightarrow, M_0)$ is defined as follows. The set of the places is $Q \stackrel{\mathrm{def}}{=} \{[C_i] \mid i \in I\}$ and the initial marking is $M_0 \stackrel{\mathrm{def}}{=} \prod_{i\in I} [C_i]^{n_i}$. The transition rules are defined inductively:

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

The remaining work is to confirm that

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

is a bisimulation. □

The combination of Lemma 7 and Lemma 5 produces the following.

Theorem 4. *The strong bisimilarity between a process $P \in \mathcal{P}_{\mathrm{CCS}^!_\bullet}$ (or $P \in \mathcal{P}_{\mathrm{CCS}^\mu_\bullet}$) and a finite state process $F \in \mathcal{P}_{\mathbf{FS}}$ is decidable.*

4.3 Decidability Results of Simulation Preorder

This part focuses on the problems $\mathcal{L} \precsim \mathbf{FS}$ and $\mathbf{FS} \precsim \mathcal{L}$. In the case that $\mathcal{L}$ is $\mathrm{CCS}^!_\bullet$ or $\mathrm{CCS}^\mu_\bullet$, the decidability result can be obtained via the same encoding provided in Section 4.2 with the help of the results already known for the Labeled Petri Net stated in Theorem 3.2 and Theorem 3.5 of [16].

Theorem 5. $\mathbf{FS} \precsim \mathrm{CCS}^!_\bullet$*,* $\mathbf{FS} \precsim \mathrm{CCS}^\mu_\bullet$*,* $\mathrm{CCS}^!_\bullet \precsim \mathbf{FS}$*,* $\mathrm{CCS}^\mu_\bullet \precsim \mathbf{FS}$ *are decidable.*

Now let's turn to $\mathrm{CCS}^!$ or CCS^μ. It has been suggested that the similarity checking is computational harder than the bisimilarity checking. This point is supported by two general proof methods applied to many process classes in a paper by Kučera and Mayr [19]. These two proof methods however cannot be used to show similar results for $\mathrm{CCS}^!$ or CCS^μ. As a matter of fact we will prove that $\mathbf{FS} \precsim \mathrm{CCS}^!$ is decidable, despite of the fact that $\mathbf{FS} \sim \mathrm{CCS}^!$ is undecidable by Theorem 3.

Our proof makes use of *simulation bases*. A simulation base is a finite subset of $\precsim$ consisting only of 'crucial' similar pairs from which a possibly infinite simulation relation can be produced algorithmically. Similarity will be decidable if simulation bases can be effectively constructed. For more on this technique, the reader is referred to [3, 17, 18].

In order to get a simulation base, we shall make good use of the *well-structured transition system* [8] of $\mathcal{P}_{\mathrm{CCS}^!}$, which was first pointed out by Busi *et al* in [4]. Here we follow the definition from [10] with slight amendment.

Definition 3. *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 indexes $i < j$ such that $x_i \leq x_j$.*

Definition 4. *The* structural expansion $\preccurlyeq$ *on the* $\text{CCS}^!$ *processes is defined inductively as follows:*

– $P \preccurlyeq Q$ *whenever* $Q \equiv P \,|\, R$ *for some* R*;*
– $(a)P \preccurlyeq (a)Q$ *whenever* $P \preccurlyeq Q$*;*
– $P \preccurlyeq Q$ *whenever* $P \equiv P_1 \,|\, P_2$*,* $Q \equiv Q_1 \,|\, Q_2$*,* $P_1 \preccurlyeq Q_1$ *and* $P_2 \preccurlyeq Q_2$*.*

Notice that Definition 4 works up to structural congruence. Intuitively $P \preccurlyeq Q$ means that Q contains at least as many possible individual processes running concurrently as P. The relation $\preccurlyeq$ is transitive. Due to the syntactical nature of the definition, $\preccurlyeq$ is decidable. The next two technical lemmas, due to Busi *et al*, are crucial to the effective production of the simulation bases. The proof of Lemma 8 is straightforward. For a detailed proof of Lemma 9, one may consult [10].

Lemma 8 (Compatibility Lemma). *Suppose that* P*,*Q *are* $\text{CCS}^!$ *processes. If* $P \preccurlyeq Q$ *and* $P \xrightarrow{\lambda} P'$*, then* Q' *exists such that* $Q \xrightarrow{\lambda} Q'$ *and* $P' \preccurlyeq Q'$*.*

Lemma 9 (Expansion Lemma). *Let* $P \in \mathcal{P}_{\text{CCS}^!}$*, then* $(\texttt{Drv}(P), \preccurlyeq)$ *is a well quasi order.*

Using the techniques and lemmas discussed above, one can infer the following main result of the section.

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

5 Concluding Remark

Summary. We have studied several decidability and undecidability issues on the bisimilarity and similarity checking problems of some subcalculi of CCS. We have concentrated on the question of how the solutions are affected when the capability of producing and manipulating local channels becomes weaker. An instance is identified that similarity checking is decidable while bisimilarity checking is not. Fig. 4 summarizes the status quo of our understanding of the decidability property. These results offer a different angle to look at the relative expressiveness of the subcalculi of CCS.

Related Work. The relative expressiveness of CCS is studied in [4, 5, 11, 6, 10, 2]. It is proved in [5, 11] that $\text{CCS}^!$ and CCS^μ are less expressive than CCS^{Pdef}. Two problems are left open in [11, 2]. Both are answered in [10]. One answer is given by an encoding from CCS^μ to $\text{CCS}^!$ that is codivergent and branching bisimilar. The other is by an encoding from CCS^μ to itself with only guarded recursion. A more formal approach to the expressiveness study is proposed in [9]. In [15] the bisimilarity checking problem between the infinite-state processes and the finite-state ones is reduced to the model checking problem of *reachability of Hennessy-Milner property.* A recent survey on the decidability and complexity results of bisimilarity checking for the processes defined in Process Rewrite Systems [21] is given in [27]. A surprising result is pointed out in [20] that strong

$\mathcal{L}$	$\mathcal{L} \sim \mathcal{L}$	$\mathcal{L} \sim \mathbf{FS}$	$\mathbf{FS} \precsim \mathcal{L}$	$\mathcal{L} \precsim \mathbf{FS}$
$CCS^!_\circ$	✓ [7]	✓ [7]	✓	✓
$CCS^\mu_\circ$	✓ [7]	✓ [7]	✓	✓
$CCS^!_\bullet$	× (Th.2)	✓ (Th.4)	✓ (Th.5)	✓ (Th.5)
$CCS^\mu_\bullet$	× (Th.1)	✓ (Th.4)	✓ (Th.5)	✓ (Th.5)
$CCS^!$	×	× (Th.3)	✓ (Th.6)	?
CCS^μ	×	× (Th.3)	?	?
CCS^{Pdef}	× [4, 11]	× [4, 11]	× [4, 11]	× [4, 11]

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

"✓": known decidable
"×": known undecidable
"?": unknown

Fig. 4. Summary of the Results

bisimilarity is decidable for a higher-order calculus. The Petri Net semantics is proposed in [12] for $CCS^\mu_\circ$ with guarded recursion. In [2] a similar encoding of $CCS^!_\circ$ into the Petri Nets is presented. Our results assert the nonexistence of reasonable encodings from $CCS^!/CCS^\mu$ to the Labeled Petri Net. The interplay between $CCS^!$ and the Chomsky Hierarchy are studied in [1].

Future Work. Recently we have attempted to set up an expansion order for CCS^μ, which we hope would help us prove the decidability of $\mathbf{FS} \precsim CCS^\mu$. The problem $CCS^! \precsim \mathbf{FS}$ is interesting. It appears undecidable, but nothing seems to indicate that a positive answer is unlikely. Finally notice that the number of the static local channels used to show Theorem 1 is bounded, whereas we have not got such a bound for Theorem 2. This may suggest that $CCS^\mu_\bullet$ cannot be encoded into $CCS^!_\bullet$.

Acknowledgements The authors are indebted to all the anonymous referees for their detailed reviews on the previous version of the paper. Their criticisms, questions and suggestions have led to a significant improvement of the paper.

References

1. Jesús Aranda, Cinzia Di Giusto, Mogens Nielsen, and Frank D. Valencia. Ccs with replication in the chomsky hierarchy: The expressive power of divergence. In *APLAS*, pages 383–398, 2007.
2. Jesús Aranda, Frank D. Valencia, and Cristian Versari. On the expressive power of restriction and priorities in ccs with replication. In *FOSSACS*, pages 242–256, 2009.
3. Olaf Burkart, Didier Caucal, Faron Moller, and Bernhard Steffen. Verification on infinite structures. In *Handbook of Process Algebra*, pages 545–623, 2001.
4. Nadia Busi, Maurizio Gabbrielli, and Gianluigi Zavattaro. Replication vs. recursive definitions in channel based calculi. In *ICALP*, pages 133–144, 2003.
5. Nadia Busi, Maurizio Gabbrielli, and Gianluigi Zavattaro. Comparing recursion, replication, and iteration in process calculi. In *ICALP*, pages 307–319, 2004.
6. Nadia Busi, Maurizio Gabbrielli, and Gianluigi Zavattaro. On the expressive power of recursion, replication and iteration in process calculi. *Mathematical Structures in Computer Science*, 19(6):1191–1222, 2009.

7. Søren Christensen, Yoram Hirshfeld, and Faron Moller. Decidable subsets of CCS. *Comput. J.*, 37(4):233–242, 1994.
8. Alain Finkel and Ph. Schnoebelen. Well-structured transition systems everywhere! *Theor. Comput. Sci.*, 256(1-2):63–92, 2001.
9. Yuxi Fu. Theory of interaction. 2010. `http://basics.sjtu.edu.cn/~yuxi/`.
10. Yuxi Fu and Hao Lu. On the expressiveness of interaction. *Theor. Comput. Sci.*, 411(11-13):1387–1451, 2010.
11. Pablo Giambiagi, Gerardo Schneider, and Frank D. Valencia. On the expressiveness of infinite behavior and name scoping in process calculi. In *FoSSaCS*, pages 226–240, 2004.
12. Ursula Goltz. CCS and petri nets. In *Semantics of Systems of Concurrent Processes*, pages 334–357, 1990.
13. Chaodong He, Yuxi Fu, and Hongfei Fu. Decidability of behavioural equivalences in process calculi with name scoping. 2010. `http://basics.sjtu.edu.cn/~chaodong/`.
14. Petr Jancar and Jirí Srba. Undecidability of bisimilarity by defender's forcing. *J. ACM*, 55(1), 2008.
15. Petr Jančar, Antonín Kučera, and Richard Mayr. Deciding bisimulation-like equivalences with finite-state processes. *Theor. Comput. Sci.*, 258(1-2):409–433, 2001.
16. Petr Jančar and Faron Moller. Checking regular properties of petri nets. In *CONCUR*, pages 348–362, 1995.
17. Petr Jančar and Faron Moller. Techniques for decidability and undecidability of bisimilarity. In *CONCUR*, pages 30–45, 1999.
18. Antonín Kučera and Petr Jancar. Equivalence-checking on infinite-state systems: Techniques and results. *TPLP*, 6(3):227–264, 2006.
19. Antonín Kučera and Richard Mayr. Why is simulation harder than bisimulation? In *CONCUR*, pages 594–610, 2002.
20. Ivan Lanese, Jorge A. Pérez, Davide Sangiorgi, and Alan Schmitt. On the expressiveness and decidability of higher-order process calculi. In *LICS*, pages 145–155, 2008.
21. Richard Mayr. Process rewrite systems. *Inf. Comput.*, 156(1-2):264–286, 2000.
22. Robin Milner. *Communication and concurrency*. Prentice-Hall, 1989.
23. Robin Milner. *Communicating and Mobile Systems: the π-calculus*. Cambridge University Press, 1999.
24. Robin Milner, Joachim Parrow, and David Walker. A calculus of mobile processes. *Inf. Comput.*, 100(1):1–77, 1992.
25. Marvin L. Minsky. *Computation: finite and infinite machines*. Prentice-Hall, 1967.
26. Davide Sangiorgi and David Walker. *PI-Calculus: A Theory of Mobile Processes*. Cambridge University Press, 2001.
27. Jiří Srba. *Roadmap of Infinite results*, volume Vol 2: Formal Models and Semantics. World Scientific Publishing Co., 2004.
28. Dirk Taubner. *Finite Representations of CCS and TCSP Programs by Automata and Petri Nets*, volume 369 of *Lecture Notes in Computer Science*. Springer, 1989.