Decidability of Behavioral Equivalences in Process Calculi with Name Scoping

Chaodong He (Joint work with Yuxi Fu and Hongfei Fu)

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

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. In the name-passing calculi, the encodings of counter 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* or *relabeling*, or by an implicit *dynamic-scoping* recursion.

A fundamental problem in the area of system verification is that of *equivalence (or preorder) checking*. 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 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. 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*. 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. When a process is defined as $P \overset{\mathrm{def}}{=} \mu X.(a\,|\,(a)(\overline{a}\,|\,X))$, the static scoping requires that the local a and the

$$\begin{tikzcd}
\mathrm{CCS}^{\mu}_{\circ} \arrow[r] & \mathrm{CCS}^{\mu}_{\bullet} \arrow[r] & \mathrm{CCS}^{\mu} \arrow[r] & \mathrm{CCS}^{\mathrm{Pdef}} \\
\mathrm{CCS}^{!}_{\circ} \arrow[r] \arrow[u] & \mathrm{CCS}^{!}_{\bullet} \arrow[r] \arrow[u] & \mathrm{CCS}^{!} \arrow[u] &
\end{tikzcd}$$

Fig. 1. CCS Hierarchy

global a must be distinguished before unfolding. That is, $\mu X.(a\,|\,(a)(\overline{a}\,|\,X))$ is understood the same as $\mu X.(a\,|\,(a')(\overline{a'}\,|\,X))$. Using dynamic scoping recursion, 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 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.

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 $\mathcal{L}_2$ 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 are summarized as follows.

– We show the undecidability (Π^0_1-hardness) of $\mathrm{CCS}^{\mu}_{\bullet} \sim \mathrm{CCS}^{\mu}_{\bullet}$ by a reduction from the halting problem of Minsky Machine. The relevant technique is called 'Defender's Forcing', which is widely used in undecidability proofs for bisimilarity checking. Typical examples of this technique can also be found

$\mathcal{L}$	$\mathcal{L} \sim \mathcal{L}$	$\mathcal{L} \sim \mathbf{FS}$	$\mathbf{FS} \precsim \mathcal{L}$	$\mathcal{L} \precsim \mathbf{FS}$
$\mathrm{CCS}^{!}_{\circ}$	✓	✓	?	?
$\mathrm{CCS}^{\mu}_{\circ}$	✓	✓	?	?
$\mathrm{CCS}^{!}_{\bullet}$	?	?	?	?
$\mathrm{CCS}^{\mu}_{\bullet}$	?	?	?	?
$\mathrm{CCS}^{!}$	?	?	?	?
CCS^{μ}	?	?	?	?
$\mathrm{CCS}^{\mathrm{Pdef}}$	×	×	×	×

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

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

Fig. 2. Problems to Explore

in. The reduction is then modified to show the undecidability (Π^0_1-hardness) of $\mathrm{CCS}^!_\bullet \sim \mathrm{CCS}^!_\bullet$. This resolves the four problems in the first column of the table.

- Busi, Gabbrielli and Zavattaro establish the undecidability (Σ^0_1-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^0_1-hard), which immediately implies the undecidability (Π^0_1-hardness) of $\mathrm{CCS}^\mu \sim \mathbf{FS}$.
- By constructing a translation from $\mathrm{CCS}^!_\bullet$ to the Labeled Petri Net, we demonstrate the decidability of $\mathrm{CCS}^!_\bullet \sim \mathbf{FS}$, $\mathrm{CCS}^!_\bullet \precsim \mathbf{FS}$ and $\mathbf{FS} \precsim \mathrm{CCS}^!_\bullet$, making use of Jančar and Moller's decidability result on the Labeled Petri Nets. The same approach applies to $\mathrm{CCS}^\mu_\bullet$.
- We show that $\mathbf{FS} \precsim \mathrm{CCS}^!$ 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 and the *well-structured transition system* for $\mathrm{CCS}^!$. In literature there are examples of formalisms 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.