On Decidability and Small Model Property of Process Equations*

Xinxin Liu

School of Cognitive and Computing Sciences

University of Sussex

Brighton BN1 9QH

England

October 1993

Abstract

This paper studies the *decidability* and *small model property* of process equations of the form

$$(P|\prod_{i=1}^{n} C_i(X_i))\backslash L \equiv (Q|\prod_{j=1}^{m} D_j(Y_j))\backslash K$$

where P, Q are finite state processes, X_i, Y_j are process variables, and $C_i(X_i)$, $D_j(Y_j)$ are process expressions *linear* in X_i and Y_j respectively. It shows that, when $n + m > 1$, the equation problem is not decidable and does not have small model property for any equivalence relation $\equiv$ which is at least as strong as complete trace equivalence but not stronger than strong bisimulation equivalence.

1 Introduction

This paper examines *small model property* and *decidability* of equations in process algebras [Mil80, Mil89, Hoa85, BK85, Bou85, Hen88]. In general, process equations have the following form

$$C(X_1, \dots, X_n) \equiv D(Y_1, \dots, Y_m) \tag{1}$$

where C, D are arbitrary process contexts, $X_1, \dots, X_n$ and $Y_1, \dots, Y_m$ are process variables, $\equiv$ is some equivalence relation on processes. Some well studied equivalence relations on processes are strong and weak bisimulation equivalences $\sim$ and $\approx$

*This work has been supported by the SERC grant GR/H16537

[Mil80, Mil89], branching bisimulation equivalence $\approx_b$ [vGW89], testing equivalence [dNH83], failure equivalence [BHR84], GSOS trace congruence or $\frac{2}{3}$–bisimulation equivalence [BIM88, LS89], and 2–nested simulation equivalence [GV89]. Equation (1) is said to be solved by processes $P_1, \ldots, P_n$ and $Q_1, \ldots, Q_m$ if the following equivalence holds

$$C(P_1, \ldots, P_n) \equiv D(Q_1, \ldots, Q_m)$$

In this case we say that (1) is *solvable*. A type of equation is said to be *decidable* if the solvability of that type of equation is decidable. A type of equation is said to have *small model property* if whenever an equation of that type is solvable then it can be solved by a finite state process. We are interested in these two properties because decidability indicates the possibility of solving the problem by automatic tools and small model property often suggests some simple method of finding solutions.

Throughout the paper it is assumed that the reader is familiar with Milner's CCS [Mil80, Mil89]. We study the decidability and small model property of a class of very natural process equations which take the following form

$$(P|\prod_{i=1}^{n} C_i(X_i))\backslash L \equiv (Q|\prod_{j=1}^{m} D_j(Y_j))\backslash K \qquad (2)$$

where P, Q are finite state processes, $|$ and $\prod$ are the parallel operator and the parallel product of CCS, $\backslash L$ and $\backslash K$ are restriction operators of CCS, $\equiv$ is some equivalence relation, $C_i(X_i)$ and $D_j(Y_j)$ are CCS expressions *linear* in X_i and Y_j respectively. An expression $C(X)$ is said to be linear in X if no two occurrences of X are subexpressions of two parallel process. More precisely, for a given variable X, the set $LE(X)$ of the expressions linear in X is the smallest set such that $X \in LE(X)$, $P \in LE(X)$, if $C(X), D(X) \in LE(X)$ then $C(X) + D(X) \in LE(X)$, and if $C(X)$ is in $LE(X)$ then so are $a.C(X), C(X)|P, P|C(X), C(X)\backslash L, C(X)[f]$, where a is any action, P is any finite state process expression not containing X, f is any rename function, and L is any set of labels. Because some variables may duplicate, the total number of variables in equation (2) may be less than $m + n$. For convenience we will call equations of form (2) which satisfy these restrictions k–ary $n \equiv m$ equations, where k is the number of variables in the equation.

Many $1 \equiv 0$ equations (unary of cause) have already been studied in the literature. In [Shi89, Par89, QL90], some subclasses of $1 \approx 0$ equations of form $(P|X)\backslash L \approx Q$ are studied with various restrictions on P, L, and Q. The results show that these subclasses are all decidable with the small model property. Some general results in [LL90b] show that the whole $1 \sim 0$ class is decidable with the small model property. Later in [Liu92], the whole $1 \approx 0$ class and $1 \approx_b 0$ class are shown to be decidable with the small model property, thus this latter work subsumes the previous results.

These references mainly concentrate on how to actually find a solution whenever solutions exist, rather than simply to decide whether the equation is solvable. In fact, from the decidability and small model property of the modal μ–calculus [Koz82, KP83, SE89] one can also conclude the decidability and small model property of many $1 \equiv 0$ equation problems. To see this, consider the following equation where Q is a finite state process and $C(X)$ is linear in X:

$$C(X) \equiv Q \tag{3}$$

Results in [ZIG87, SI91, Liu92] show that, for many equivalence relations $\equiv$ (including $\sim$, $\approx$, and $\approx_b$), Q has a *characteristic formula* $F_Q^{\equiv}$ of the modal μ–calculus such that, for any process P, $C(P) \equiv Q$ if and only if $C(P)$ satisfies $F_Q^{\equiv}$. According to [LL90a], a formula $\mathcal{W}(C, F_Q^{\equiv})$ can be effectively constructed such that $C(P)$ *satisfies* $F_Q^{\equiv}$ just in case P *satisfies* $\mathcal{W}(C, F_Q^{\equiv})$. Thus the solutions to equation (3) are exactly those processes satisfying $\mathcal{W}(C, F_Q^{\equiv})$. It is obvious that any $1 \equiv 0$ equation has form (3), so the decidability and small model property of these $1 \equiv 0$ equations are guaranteed by the fact that the satisfiability of a modal μ–calculus formula is decidable and that if such a formula is satisfiable then it is satisfiable by a finite state process.

Now it is tempting to try to obtain similar results for equation problems with more variables. In [JJLL93], the problem of constructing processes $P_1, \ldots, P_n$ such that $C(P_1, \ldots, P_n)$ satisfies F is considered, where C is an arbitrary context described as an action transducer and F is a formula of model μ–calculus with pure maximal fixed point operators. In that paper, a procedure for constructing models is presented. If this procedure terminates successfully then a finite state model can be constructed while if it terminates unsuccessfully it is guaranteed that no model exists. It is conjectured in the paper that the method guarantees termination in all circumstances. This conjecture implies that n–ary $n \sim 0$ equation problem is decidable and has small model property. This is because for a n–ary $n \sim 0$ equation

$$(P|\prod_{i=1}^{n} C_i(X_i))\backslash L \sim Q$$

the left hand side can be expressed as $C(X_1, \ldots, X_n)$ where C is a context described as an action transducer, and the characteristic formula $F_Q^{\sim}$ is a formula with pure maximal fixed point operators.

In summary, the picture as we see it at the moment is that, $1 \equiv 0$ equation problems are decidable with the small model property for many useful equivalence relations and the cases for k–ary $n \equiv m$ equations such that $n + m > 1$ are not clear. In this paper, we show that for most of the interesting equivalence relations $\equiv$ including those mentioned earlier, any k–ary $n \equiv m$ equation problem is not decidable and does not have the small model property when $n + m > 1$. In particular, we show that the unary and binary $1 \equiv 1$ equation problems, the unary and binary $2 \equiv 0$ equation problems are already undecidable and do not have small model property for those equivalence relations. This gives a negative answer to the conjecture put forward in [JJLL93].

2 Some Useful CCS Processes and Contexts

A standard way to show undecidability is to demonstrate an effective reduction from some known undecidable problem. The first undecidable problem to come to

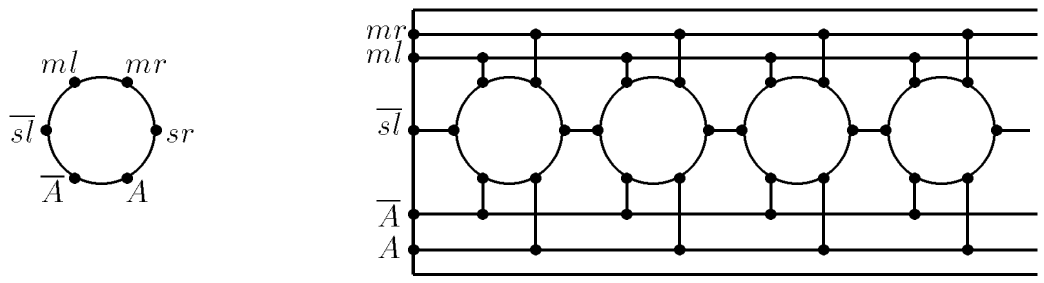

Figure 1: A cell and the tape

one's mind would probably be the halting problem of Turing machines. We will show this kind of reduction to prove the undecidability of equation problems in the next section. For this purpose, in this section we construct some useful CCS processes and contexts and show some of their properties.

First we construct a process T which imitates a blank Turing machine tape. A Turing machine tape can be seen as a row of cells, each holding a letter of a finite alphabet A or holding the blank symbol b. Each cell must be in one of the following states according to its relative position to the read and write head and the letter it holds:

1. It holds x with the head pointing to it. We write $W(x)$ for this state and say that the cell is awake.

2. It holds x with the head on its left. We write $S_l(x)$ for this state and say that the cell is asleep and waiting to be awakened from the left.

3. It holds x with the head on its right. We write $S_r(x)$ for this state and say that the cell is asleep and waiting to be awakened from the right.

When it is in $W(x)$, that is awake and holding x, the following things are possible. The environment can read out its contents by synchronizing on the port $\bar{x}$. The environment can also change its contents by synchronizing on the port y and there by changing its state to $W(y)$. On receiving a signal on the port ml (move to the left), the cell will wake its left side neighbor by signaling on the port $\overline{sl}$ (signal left) and then enter the state $S_l(x)$. Likewise, on receiving a signal on the port mr (move to the right), the cell will wake its right side neighbor by signaling on the port sr (signal right) and then enter the state $S_r(x)$. When it is asleep in $S_l(x)$ ($S_r(x)$), the only possible action for a cell is to receive a signal on the port $\overline{sl}$ (sr) and enter the state $W(x)$. The precise behavior is expressed by the following CCS expressions:

$$\begin{aligned} W(x) &\stackrel{d}{=} \bar{x}.W(x) + \textstyle\sum_{y\in A} y.W(y) + ml.\overline{sl}.S_l(x) + mr.sr.S_r(x) \\ S_l(x) &\stackrel{d}{=} \overline{sl}.W(x) \\ S_r(x) &\stackrel{d}{=} sr.W(x) \end{aligned}$$

Now, the blank tape in its initial state is just a row of cells in state $S_l(b)$. The following recursive definition gives the blank tape T

$$T \overset{d}{=} (S_l(b)[link/sr]|T[link/sl])\backslash link$$

The construction is pictured in Figure 1, where A and $\overline{A}$ are in fact two sets of ports named by the letters and barred letters in the alphabet of the Turing machine.

A Turing machine tape can also be described by the following infinite set of equations about $B(s_1, s_2)$, where $s_1 \in (A \cup \{b\})^*$ is the contents of the tape between the end of the tape and the current position of the head, $s_2 \in (A \cup \{b\})^*\{b\}^\omega$ is the contents of the rest of the tape, $\mathtt{hd}, \mathtt{tl}$, and $\widehat{\ }$ are the usual head, tail and concatenation functions on strings

$$\begin{array}{lll} B(s_1, s_2) & \overset{d}{=} & \overline{\mathtt{hd}(s_1)}.B(s_1, s_2) + \sum_{y \in A} y.B(y\widehat{\ }\mathtt{tl}(s_1), s_2) \\ & & +ml.\tau.B(\mathtt{tl}(s_1), \mathtt{hd}(s_1)\widehat{\ }s_2) + mr.\tau.B(\mathtt{hd}(s_2)\widehat{\ }s_1, \mathtt{tl}(s_2)) \quad \text{when } s_1 \neq \epsilon \\ B(\epsilon, s_2) & \overset{d}{=} & \overline{sl}.I(s_2) \\ I(s_2) & \overset{d}{=} & \overline{sl}.B(\mathtt{hd}(s_2), \mathtt{tl}(s_2)) \end{array}$$

We can take $I(b^\omega)$ to be a blank tape in its initial state.

It is not difficult to check that both $I(b^\omega)$ and T solve the following equation for $\equiv$ being $\sim$

$$X \equiv \overline{sl}.(W(b)[link/sr]|X[link/sl])\backslash link \tag{4}$$

(notice that the τ actions in the definition of $I(b^\omega)$ are not necessary apart from making $I(b^\omega)$ into a solution to that equation). Because this equation is weakly guarded, it has unique solution modulo $\sim$ [Mil89]. Thus $T \sim I(b^\omega)$, which is a formal justification that T indeed simulates a blank Turing machine tape. We will talk more about this equation later, and in the rest of the paper, we will write $D(X)$ for $\overline{sl}.(W(b)[link/sr]|X[link/sl])\backslash link$.

At this point we clarify some necessary notations. The reader is referred to [Mil89] for a full account of the operational semantics of CCS processes. We follow the notations [Mil89] and write $P \xrightarrow{a} Q$ to mean that the process P (in this state) can perform action a and change its state to Q, $P \not\xrightarrow{a}$ to mean P is not capable of any a action. Let $a_1, \ldots, a_n$ be a sequence of visible actions (not τ), then $P \xRightarrow{a_1 \ldots a_n} Q$ means that P can reach Q by performing action sequence $a_1, \ldots, a_n$ with finitely many τ actions interleaved.

Definition 2.1 *A sequence of visible actions s is a* trace *of P if $P \xRightarrow{s} P'$ for some P'. A sequence of visible actions s is a* complete trace *of P if $P \xRightarrow{s} P'$ for some P' with $P' \not\xrightarrow{a}$ for any a. Let $Tr(P)$ and $CTr(P)$ denote the trace set and complete trace set of P respectively, two processes P and Q are said to be* trace equivalent, *write $P \approx_t Q$, if $Tr(P) = Tr(Q)$, and two processes P and Q are said to be* complete trace equivalent, *write $P \approx_{ct} Q$, if $Tr(P) = Tr(Q)$ and $CTr(P) = CTr(Q)$.*

Lemma 2.2 *When $\equiv$ is $\approx_t$, equation (4) has unique solution modulo $\approx_t$.*

Proof For any process P, lets write $D^n(P)$ for $D(D^{n-1}(P))$ when $n > 0$ and $D^0(P)$ for P. Notice that if $D^n(P) \overset{s}{\Longrightarrow} R$ and the number of mr's in s is less than n, then the transitions must be independent of P. More precisely, in this case R has the form $H\{P/X\}$ for some expression H such that for any process Q, $D^n(Q) \overset{s}{\Longrightarrow} H\{Q/X\}$. Now suppose P and Q are two solutions and $P \overset{s}{\Longrightarrow} P'$ for some P'. We can choose a sufficiently large n such that the number of mr's in s is less than n. By the congruence property of $\approx_t$ [Mil89], $P \approx_t D(P)$ implies $P \approx_t D^n(P)$. Thus $D^n(P) \overset{s}{\Longrightarrow} R$ for some R. By the above argument, there exists an expression H such that $D^n(Q) \overset{s}{\Longrightarrow} H\{Q/X\}$. So $Q \overset{s}{\Longrightarrow} Q'$ for some Q' because $Q \approx_t D(Q)$ implies $Q \approx_t D^n(Q)$. Thus $Tr(P) \subseteq Tr(Q)$. We can show $Tr(Q) \subseteq Tr(P)$ in the same way, so $P \approx_t Q$. It is interesting to observe that equation (4) is not sequential, thus does not satisfy the sufficient condition for having unique solution in [Mil89]. $\square$

Lemma 2.3 *If $P \approx_t I(b^\omega)$ then P is an infinite state process.*

Proof Assume that P has finite states, then $Tr(P)$ is a regular set because by treating each state of P as acceptance state and each τ move as empty move we obtain a finite automaton which accepts $Tr(P)$. Since we start from $P \approx_t I(b^\omega)$, this implies that $Tr(I(b^\omega))$ is a regular set which further implies that $Tr(I(b^\omega)) \cap \overline{sl}\,(ml+mr)^*\overline{sl}$ is also regular. However it is easy to see that the later is not regular, because it consists of stings of the form $\overline{sl}\,\widehat{}\,s\,\widehat{}\,ml\,\widehat{}\,\overline{sl}$ where s is a sequence of mr and ml balenced like left and right parenthesis. Thus P must have infinite states. $\square$

Next, we construct a dyadic context C as a coordinator, which will deadlock if one of the subprocesses cannot follow the actions of the other. Let

$$L = A \cup \overline{A} \cup \{ml, mr, \overline{sl}\}$$

where A is the finite alphabet of the Turing machine. It is obvious that all possible visible actions of T are contained in L. Let L_1, L_2 be two disjoint sets of actions such that they are both isomophic to L with f_1, f_2 being the corresponding isomophic maps (that is f_i is one one and onto and $f_i(\bar{a}) = \overline{f_i(a)}$ for $a \in L$). Let syn, err (for synchronizing and error) be two labels not in $L_1 \cup L_2$. Then f_1, f_2 can be extended to rename functions by defining $f_1(a) = f_2(a) = err$ for any action $a \notin L$. Now define

$$\begin{array}{lcll}
R & \overset{d}{=} & \sum_{a\in L-\{\overline{ml},\overline{mr}\}} f_1(a).Q_a + f_1(\overline{ml}).f_2(\overline{ml}).syn.R + f_1(\overline{mr}).f_2(\overline{mr}).syn.R & \\
Q_a & \overset{d}{=} & f_2(a).Q_1 + \tau.f_2(a).Q_2 & a \in L - \{\overline{ml}, \overline{mr}\} \\
Q_1 & \overset{d}{=} & syn.\tau.\tau.R + \tau.Q_2 & \\
Q_2 & \overset{d}{=} & syn.\tau.R + \tau.syn.R &
\end{array}$$

This definition of R is slightly complicated by the insertion of τ actions in various places. Basicly, R could be defined as $R \overset{d}{=} \sum_{a\in L} f_1(a).f_2(a).syn.R$, which may still give a better idea how R actually works. However, these τ's are necessary in order to satisfy the equation in the first part of Lemma 2.4 which enables us to derive more general conclusions. Otherwise, using the simpler definition, we can only show a weaker version of that equation with $\approx$ in place of $\sim$. From now on we will write $C(X,Y)$ for $(X[f_1]|R|Y[f_2])\backslash L_1 \cup L_2$. It is not difficult to see that a necessary condition for $C(P,Q)$ to be always capable of doing *syn* and nothing else (no *err*) is that whatever visible action P may perform, Q must be able to follow. This and some other useful properties of C are stated in the following Lemma. In the Lemma, as well as the rest of the paper, T_0 is a finite state process defined by $T_0 \overset{d}{=} sl.T_0 + \overline{mr}.\tau.T_0$.

Lemma 2.4 *For C, T, and T_0 constructed above, the following hold*

1. $C(T,T) \sim C(T_0,T_0)$, *and*
2. $CTr(C(P,Q)) \subseteq CTr(C(T_0,T_0))$ *implies* $Tr(P) \subseteq Tr(Q)$.

Proof Let $Dir(I(b^\omega)) = \{P \mid \exists s \in L^*.I(b^\omega) \overset{s}{\Longrightarrow} P\}$ be the set of derivatives of $I(b^\omega)$. It is not difficult to see that the following is a strong bisimulation relation containing $(C(I(b^\omega), I(b^\omega)), C(T_0,T_0))$. Hence $C(I(b^\omega), I(b^\omega)) \sim C(T_0,T_0)$ and $C(T,T) \sim C(T_0,T_0)$.

$$
\begin{aligned}
&\{((P[f_1]|Q|P[f_2])\backslash L_1 \cup L_2,\ (T_0[f_1]|Q|T_0[f_2])\backslash L_1 \cup L_2) \mid \\
&\qquad P \in Dir(I(b^\omega)),\ P \overset{\tau}{\not\longrightarrow},\ Q \in \{R,\ \tau.R,\ \tau.\tau.R,\ syn.R,\ Q_1,\ Q_2\}\} \cup \\
&\{((P'[f_1]|Q_a|P[f_2])\backslash L_1 \cup L_2,\ (T_0[f_1]|Q_{sl}|T_0[f_2])\backslash L_1 \cup L_2) \mid \\
&\qquad P \in Dir(I(b^\omega)),\ a \in L - \{\overline{ml}, \overline{mr}\},\ P \overset{a}{\longrightarrow} P'\} \cup \\
&\{((P'[f_1]|f_2(a).Q_2|P[f_2])\backslash L_1 \cup L_2,\ (T_0[f_1]|f_2(sl).Q_2|T_0[f_2])\backslash L_1 \cup L_2) \mid \\
&\qquad P \in Dir(I(b^\omega)),\ a \in L - \{\overline{ml}, \overline{mr}\},\ P \overset{a}{\longrightarrow} P'\} \cup \\
&\{((P'[f_1]|f_2(a).syn.R|P[f_2])\backslash L_1 \cup L_2,\ (\tau.T_0[f_1]|f_2(\overline{mr}).syn.R|T_0[f_2])\backslash L_1 \cup L_2) \mid \\
&\qquad P \in Dir(I(b^\omega)),\ a \in \{\overline{ml}, \overline{mr}\},\ P \overset{a}{\longrightarrow} P'\} \cup \\
&\{((P'[f_1]|f_2(a).syn.R|P[f_2])\backslash L_1 \cup L_2,\ (T_0[f_1]|f_2(\overline{mr}).syn.R|T_0[f_2])\backslash L_1 \cup L_2) \mid \\
&\qquad P \in Dir(I(b^\omega)),\ a \in \{\overline{ml}, \overline{mr}\},\ \exists P''.P \overset{a}{\longrightarrow} P'', P'' \overset{\tau}{\longrightarrow} P'\} \cup \\
&\{((P[f_1]|Q|\tau.P[f_2])\backslash L_1 \cup L_2,\ (T_0[f_1]|Q|\tau.T_0[f_2])\backslash L_1 \cup L_2) \mid \\
&\qquad P \in Dir(I(b^\omega)),\ P \overset{\tau}{\not\longrightarrow},\ Q \in \{R,\ syn.R\}\}
\end{aligned}
$$

Suppose $CTr(C(P,Q)) \subseteq CTr(C(T_0,T_0))$. Note $C(T_0,T_0) \approx_{ct} syn^\omega$, where syn^ω is the process which performs *syn* for ever. It is easy to see that if $P \overset{a}{\Longrightarrow} P'$ for any action a, then $a \in L$ (otherwise $f_1(a) = err$ and $C(P,Q) \overset{err}{\Longrightarrow} C(P',Q)$) and $Q \overset{a}{\Longrightarrow} Q'$ for some Q' such that $CTr(C(P',Q')) \subseteq CTr(syn^\omega)$. From this it is easy to see that $CTr(C(P,Q)) \subseteq CTr(C(T_0,T_0))$ implies $Tr(P) \subseteq Tr(Q)$. In fact $CTr(C(P,Q)) \subseteq CTr(C(T_0,T_0))$ implies that (P,Q) is contained in a *simulation* relation [Mil89]. $\Box$

The processes T, T_0, R, contexts C, D, and label sets L, L_1, L_2 with rename functions f_1, f_2 defined here are referred in the proof of the main theorems in the next section.

3 Main Results

With the preparation in the last section, we are now ready to show the main results of the paper, namely that many equation problems are undecidable and do not have small model property.

Theorem 3.1 *For any $\equiv$ such that $\sim \subseteq \equiv \subseteq \approx_t$, both the unary $1 \equiv 1$ equation problem and the binary $1 \equiv 1$ equation problem are not decidable and do not have the small model property.*

Proof It is sufficient to construct some effective reductions from the divergence problem of Turing machines, which is well known to be not semi–decidable. There is a systematic way of constructing a finite state process M_i which simulates the finite–state control mechanism of the i–th Turing machine for each i. Thus $(M_i|T)\backslash L$ will simulate the i–th Turing machine such that $(M_i|T)\backslash L \sim \tau^\omega$ if and only if the i–th Turing machine does not halt on a blank tape, and also that $(M_i|T)\backslash L$ outputs something if and only if the i–th Turing machine halts, where τ^ω is the process which only performs internal actions for ever. Now we can show that the i–th Turing machine diverges if and only if the following unary $1 \equiv 1$ equation is solvable when $\sim \subseteq \equiv \subseteq \approx_t$ and $a \neq b$

$$a.(M_i|X)\backslash L + b.X \equiv a.\tau^\omega + b.D(X) \tag{5}$$

For one direction, suppose the i–th Turing machine diverges, that is to say $(M_i|T)\backslash L \sim \tau^\omega$. Since $T \sim D(T)$, and $\sim \subseteq \equiv$, T solves equation (6).

For the converse direction, suppose T' solves equation (6). Because $\equiv \subseteq \approx_t$ and $a \neq b$, in this case $(M_i|T')\backslash L \approx_t \tau^\omega$ and $T' \approx_t D(T')$. By Lemma 2.2 $T' \approx_t T$, thus $(M_i|T)\backslash L \approx_t (M_i|T')\backslash L \approx_t \tau^\omega$, and the i–th Turing machine diverges on a blank tape (otherwise $(M_i|T)\backslash L$ should be able to output something).

Similarly, it is easy to work out that the i–th Turing machine diverges if and only if the following binary $1 \equiv 1$ equation is solvable when $\sim \subseteq \equiv \subseteq \approx_t$ and a, b, c are three different actions

$$a.(M_i|X)\backslash L + b.X + c.D(X) \equiv a.\tau^\omega + b.Y + c.Y \tag{6}$$

Thus we showed effective reductions from the divergence problem of Turing machines to the unary and binary $1 \equiv 1$ equation problems. So the unary $1 \equiv 1$ equation problem and the binary $1 \equiv 1$ equation problem are not semi-decidable and thus not decidable.

In order to prove that a type of equation does not have the small model property, we only need to find a solvable equation of that type and show that any solution to the equation has infinite states. It is easy to see from Lemma 2.2 and Lemma 2.3 that, when $\equiv \subseteq \approx_t$, equation (4) is a solvable unary $1 \equiv 1$ equation which only has infinite state solutions. Also for the same reason, when $\equiv \subseteq \approx_t$ and $a \neq b$, the following is a solvable binary $1 \equiv 1$ equation which only has infinite state solutions.

$$a.X + b.D(X) \equiv a.Y + b.Y$$

Thus both unary and binary $1 \equiv 1$ equation problems do not have small model property. $\square$

Theorem 3.2 *For any $\equiv$ such that $\sim \subseteq \equiv \subseteq \approx_{ct}$, both unary $2 \equiv 0$ equation problem and binary $2 \equiv 0$ equation problem are not decidable and do not have small model property.*

Proof Again we will construct some effective reductions from the divergence problem of Turing machines. Lets say that M_i is a finite state process which simulates the finite–state control mechanism of the i–th Turing. Thus $(M_i|T)\backslash L$ will simulate the i–th Turing machine such that $(M_i|T)\backslash L \sim \tau^\omega$ if and only if the i–th Turing machine does not halt on blank tape, and if and only if $(M_i|T)\backslash L$ has no complete trace. We will show that, when $\sim \subseteq \equiv \subseteq \approx_{ct}$, the i–th Turing machine diverges if and only if the following equation is solvable

$$\tau.(M_i[f_2]|X[f_2])\backslash L_1 \cup L_2 + \tau.C(D(X), X) + \tau.C(X, D(X)) \equiv \tau^\omega + \tau.C(T_0, T_0) \quad (7)$$

which, by the expansion theorem of [Mil89], has the exact solution set as the following unary $2 \equiv 0$ equation, where $a, b \in L_1 \cup L_2$ and $a \neq b$

$$\begin{aligned} &((a.M_i[f_2] + a.(D(X)[f_1])|R) + b.X[f_1]) \\ &\quad |(\bar{a}.X[f_2] + \bar{b}.(R|D(X)[f_2])))\backslash L_1 \cup L_2 \equiv \tau^\omega + \tau.C(T_0, T_0) \end{aligned}$$

For one direction, suppose the i–th Turing machine diverges, that is to say $(M_i|T)\backslash L \sim \tau^\omega$ and therefor $(M_i[f_2]|T[f_2])\backslash L_1 \cup L_2 \sim ((M_i|T)\backslash L)[f_2]\backslash L_1 \sim \tau^\omega$. Since $T \sim D(T)$ and by Lemma 2.4 $C(T, T) \sim C(T_0, T_0)$, thus T solves equation (7) since $\sim \subseteq \equiv$.

For the converse direction, suppose equation (7) is solvable by T'. Because $\equiv \subseteq \approx_{ct}$, this implies that $(M_i[f_2]|T'[f_2])\backslash L_1 \cup L_2$ has no complete trace, and that $CTr(C(T', D(T')))$ and $CTr(C(D(T'), T'))$ are subsets of $CTr(C(T_0, T_0))$. By Lemma 2.4, $T' \approx_t D(T')$, and by Lemma 2.2 $T' \approx_t T$. Thus $(M_i[f_2]|T[f_2])\backslash L_1 \cup L_2$ has no complete trace, nor has $((M_i|T)\backslash L)[f_2]\backslash L_1$ nor $(M_i|T)\backslash L$. So the i–th Turing machine diverges on a blank tape.

In a similarly way, we can work out that the i–th Turing machine diverges if and only if the following binary $2 \equiv 0$ equation is solvable when $\sim \subseteq \equiv \subseteq \approx_{ct}$ and $a, b \in L_1 \cup L_2, a \neq b$

$$\begin{aligned} &((a.M_i[f_2] + a.(X[f_1]|R) + a.(D(X)[f_1]|R) + b.X[f_2] \\ &\quad + \bar{b}.D(X)[f_2])|(\bar{a}.Y[f_2] + \bar{b}.(Y[f_1]|R))\backslash L_1 \cup L_2 \equiv \tau^\omega + \tau.C(T_0, T_0) \end{aligned}$$

Thus we showed effective reductions from the divergence problem of Turing machine to the unary and binary $2 \equiv 0$ equation problems. So the unary $2 \equiv 0$ and binary $2 \equiv 0$ equation problems are not semi-decidable and thus not decidable.

Using Lemma 2.2,2.3,2.4, it is not difficult to work out that, when $\equiv \subseteq \approx_{ct}$ and $a, b \in L_1 \cup L_2, a \neq b$, the following unary and binary $2 \equiv 0$ equations are all solvable but only have infinite state solutions

$$\begin{aligned}
((a.(D(X)[f_1])|R) + b.(X[f_1]|R))|(\bar{a}.X[f_2] + \bar{b}.D(X)[f_2]))\backslash L_1 \cup L_2 &\equiv \tau.C(T_0, T_0) \\
((a.X[f_1] + a.D(X)[f_1] + b.X[f_2] + \bar{b}.D(X)[f_2]) & \\
|(\bar{a}.(R|Y[f_2]) + \bar{b}.(Y[f_1]|R))\backslash L_1 \cup L_2 &\equiv \tau.C(T_0, T_0)
\end{aligned}$$

Thus both unary and binary $2 \equiv 0$ equation problems do not have small model property. □

4 Conclusion and Related Works

In the last section we showed that four types of $n \equiv m$ equation problems are not decidable and do not have small model property for any equivalence relation which is as least as strong as complete trace equivalence (this can be relaxed to trace equivalence in the case of $1 \equiv 1$) but not stronger than strong bisimulation equivalence. These four types of equation problems are the unary and binary $1 \equiv 1$ equation problems and the unary and binary $2 \equiv 0$ equation problems. Undecidability of $1 \equiv 1$ equation problems is somewhat expected because recursion can be coded into such an equation problem, but undecidability of $2 \equiv 0$ equation problems is rather unexpected. This shows the computation power of communication.

The negative results about these four basic types of equation problems have very general implications. Because any k–ary $n \equiv m$ equation problem with $m + n > 1$ would have one of these basic problems as special case, such k–ary $n \equiv m$ equation problem is surely undecidable and does not have the small model property if $\equiv$ is at least as strong as complete trace equivalence but not stronger than bisimulation equivalence. Also the range of equivalence relations from complete trace equivalence to strong bisimulation equivalence covers the most interesting ones in the study of concurrency, including all the equivalence relations mentioned in the beginning of the Introduction. Indeed, one would expect a reasonable equivalence relation of concurrent systems to be in this range. Although the constructions are specific for CCS, similar reductions could be constructed to show similar results within other process algebras.

Undecidability results show the limitation of automatic tools for solving process equations. Lack of small model property means the need for more insight in order to construct solutions for these equations. Thus machine assisted semi-automatic tools such as the one proposed in [JJLL93] and techniques of constructing infinite state solutions seem to be the direction for future research.

Recently many decidability results about bisimilarity of infinite state processes have been established [HS91, CHS92, CHM93b, CHM93a]. This may give some hope to push the decidability of $1 \sim 0$ equations to equations of the form $C(X) \sim P$ where P is some type of infinite state process. Although in this case small model property fails immediately, results in [LL90b] already implies that the non–solvability of these equations is semi–decidable. Thus we only need to show that the solvability is also semi–decidable to show the decidability of such equations.

Another interesting line of research is to identify some decidable subclass of k–ary $n \equiv m$ equation problems. Here the results of [MM90, Mol89, CHM93b] about unique decomposition of processes may provide some clue. To be somewhat more precise, results in [MM90, Mol89, CHM93b] show that for certain processes P, there exists a unique decomposition $P_1 \| \ldots \| P_m$ where $\|$ is the merge operator which is like $|$ but without communication. Thus for such process P, we can decompose this kind of k–ary $n \sim 0$ equation

$$C_1(X_1) \| \ldots \| C_n(X_n) \sim P$$

into a set of $1 \sim 0$ equations. It is easy to see that the possibility of such decompositions are finite, thus methods of solving $1 \sim 0$ equations can come into play.

Instead of solving a particular type of equations in a general process algebra, a different approach is to consider arbitrary equations in some simpler process algebra. Using the idea of unification, two algorithms are devised in [Dro92] for solving arbitrary equations in a very simple process algebra with choice operator as the only operator. It is interesting to see how far this approach can go by adding more operators into the algebra while still being able to find an algorithm to solve arbitrary equations of the algebra. The undecidability results here show that there is a limit for that.

Acknowledgments

The author had some very inspiring discussions with Kim Larsen on this topic. Matthew Hennessy read through a previous manuscript and suggested many improvements. Several discussions with Alan Jeffrey about uniqueness of solution to process equations also improved the presentation.

References

[BHR84] S.D. Brookes, C.A.R. Hoare, and A.W. Roscoe. A theory of communicating sequential processes. *Journal of the ACM*, 31:560–599, 1984.

[BIM88] B. Bloom, S. Istrail, and A.R. Meyer. Bisimulation can't be traced. *Proceedings of Principles of Programming Languages*, 1988.

[BK85] J.A. Bergstra and J.W. Klop. Algebra of communicating processes with abstraction. *Theoretical Computer Science*, 37:77–121, 1985.

[Bou85] G. Boudol. Calcul de processus et verification. Technical Report 424, INRIA, 1985.

[CHM93a] S. Christensen, Y. Hirshfeld, and F. Moller. Bisimulation equivalence is decidable for all basic parallel processes. *Lecture Notes In Computer Science, Springer Verlag*, (630), 1993.

[CHM93b] S. Christensen, Y. Hirshfeld, and F. Moller. Decomposbility, decidability, and axiomatisability for bisimulation equivalence. In *Proceedings on Logic in Computer Science*, 1993.

[CHS92] S. Christensen, H. Hüttel, and C. Stirling. Bimimulation equivalence is decidable for all context–free processes. *Lecture Notes In Computer Science, Springer Verlag*, 630, 1992.

[dNH83] R. de Nicola and M. Hennessy. Testing equivalence for processes. *Theoretical Computer Science*, 34:205–228, 1983.

[Dro92] N. J. Drost. Unification in the algebra of sets with union and empty set. Technical Report Report P9213, University of Amsterdam, July 1992.

[GV89] J.F. Groote and F.W. Vaandrager. Structured operational semantics and bisimulation as a congruence. *Lecture Notes In Computer Science, Springer Verlag*, 372, 1989. Proceedings of ICALP89.

[Hen88] M. Hennessy. *An Algebraic Theory of Processes.* MIT Press, 1988.

[Hoa85] C.A.R. Hoare. *Communicating Sequential Processes.* Prentice–Hall, 1985.

[HS91] H. Hüttel and C. Stirling. Action speaks louder than words: Proving bisimilarity for context-free processes. In *Proceedings on Logic in Computer Science*, 1991.

[JJLL93] O. Jensen, C. Jeppesen, J. Lang, and K. Larsen. Model construction for implicit specifications in modal logic. *Lecture Notes In Computer Science, Springer Verlag*, 715, 1993. Proceedings of CONCUR'93.

[Koz82] D. Kozen. Results on the propositional mu–calculus. *Lecture Notes In Computer Science, Springer Verlag*, 140, 1982. in Proc. of International Colloquium on Algorithms, Languages and Programming 1982.

[KP83] D. Kozen and Rohit Parikh. A decision procedure for the propositional mu–calculus. *Lecture Notes In Computer Science, Springer Verlag*, 164, 1983. in Proc. of Logics of Programming.

[Liu92] X. Liu. *Specification and Decomposition in Concurrency.* PhD thesis, University of Aalborg, Fredrik Bajers Vej 7, DK 9220 Aalborg ø, Denmark, 1992.

[LL90a] K.G. Larsen and Xinxin Liu. Compositionality through an operational semantics of contexts. *Lecture Notes In Computer Science, Springer Verlag*, 443, 1990. In proceedings of International Colloquium on Algorithms, Languages and Programming 1990.

[LL90b] K.G. Larsen and Xinxin Liu. Equation solving using modal transition systems. In *Proceedings on Logic in Computer Science*, 1990.

[LS89] K.G. Larsen and A. Skou. Bisimulation through probabilistic testing. *Proceedings of Principles of Programming Languages*, 1989.

[Mil80] R. Milner. *Calculus of Communicating Systems*, volume 92 of *Lecture Notes In Computer Science, Springer Verlag*. Springer Verlag, 1980.

[Mil89] R. Milner. *Communication and Concurrency*. Prentice–Hall, 1989.

[MM90] R. Milner and F. Moller. Unique decomposition of processes. *Bulletin of the European Association for Theoretical Computer Science*, 41:226–232, 1990.

[Mol89] F. Moller. *Axioms for Concurrency*. PhD thesis, University of Edinburgh, Mayfield Road, Edinburgh, Scotland, 1989.

[Par89] J. Parrow. Submodule construction as equation solving in CCS. *Theoretical Computer Science*, 1989.

[QL90] H. Qin and P. Lewis. Factorization of finite state machines under observational equivalence. *Lecture Notes In Computer Science, Springer Verlag*, 458, 1990.

[SE89] Robert S. Streett and E. Allen Emerson. An automata theoretic decision procedure for the propositional mu-calculus. *Information and Computation*, 81:249–264, 1989.

[Shi89] M.W. Shields. A note on the simple interface equation. *The Computer Journal*, 32(5), 1989.

[SI91] B. Steffen and A. Ingolfsdottir. Characteristic formulae for processes with divergence. Technical Report Technical Report 1/91, School of Congnitive and Computing Sciences, University of Sussex, 1991.

[vGW89] R. J. van Glabbeek and W.P. Weijland. Branching time and abstraction in bisimulation semantics (extended abstract). *Information Processing 89*, pages 613–618, 1989.

[ZIG87] Michael Zeeberg, Anna Ingolfsdottir, and Jens Christian Godskesen. Fra hennessy–milner logig til ccs–processer. Master's thesis, Aalborg University, 1987.