ELSEVIER

Theoretical Computer Science 170 (1996) 445–464

Theoretical Computer Science

Deciding bisimulation and trace equivalences for systems with many identical processes

Hsu-Chun Yen*, Shi-Tsuen Jian, Ta-Pang Lao

Department of Electrical Engineering, National Taiwan University, Taipei, Taiwan 106, Republic of China

Received March 1995
Communicated by M. Nivat

Abstract

In the study of process semantics, *trace equivalence* and *bisimulation equivalence* constitute the two extremes of the so-called *linear time–branching time spectrum*. In this paper, we study the complexity and decidability issues of deciding trace and bisimulation equivalences for the model of *systems with many identical processes* with respect to various interprocess communication structures. In our model, each system consists of an arbitrary number of identical finite-state processes using Milner's *calculus of communicating systems* (CCS) as the style of interprocess communication. As it turns out, deciding trace and bisimulation equivalences are undecidable for *star-like* and *linear* systems, whereas the two problems are complete for PSPACE and PTIME, respectively, for *fully connected* systems.

1. Introduction

In concurrency theory, various notions of *equivalence* have been proposed for capturing the essence of two concurrent systems being behaviorally equivalent. According to the degree of *coarseness*, the so-called *linear time–branching time hierarchy of equivalences* listed in [22] defines a rich set of equivalences (and their coarseness relationships) of interest in the semantics of concurrency theory. Of many equivalence notions defined in the literature, *bisimulation equivalence* and *trace equivalence*, constituting the two extremes of the spectrum of the linear time–branching time hierarchy, define the *finest* and the *coarsest* equivalences, respectively. *Trace equivalence* is identical to the *language equivalence* in automata theory. That is, by regarding all states as accepting states, two concurrent systems are said to be *trace equivalent* if the two languages associated with the automata representing the two concurrent systems are identical. Intuitively speaking, the branching behaviors of a concurrent system are ignored when dealing with trace equivalence; hence, such an equivalence notion is

* Corresponding author. E-mail: yen@cc.ee.ntu.edu.tw.

PII S0304-3975(96)00176-4

considered the coarsest of all. In contrast, for two concurrent systems to be *bisimulation equivalent*, any of them must be able to 'mimic' the actions of its opponent on a step-by-step basis; as a result, *bisimulation equivalence* constitutes the finest among the above hierarchy.

Of many problems that are of interest in the study of concurrent systems, the decidability and complexity issues of the *equivalence problem* (i.e., that of determining whether two systems are "behaviorly equivalent") with respect to various equivalence notions are relatively well-studied for *finite-state systems*, see, e.g., [2, 17, 19, 20]. It is not surprising that for finite-state systems, the equivalence problem is decidable for all the equivalences proposed in the literature. Recently, considerable efforts have been directed to the study of the decidability and complexity issues of the equivalence problem for *infinite-state systems* including *basic process algebra* (BPA) [11], *basic parallel processes* (BPP) [4, 13, 15], *context-free processes* [6, 14, 16], and *Petri nets* [12, 18]. The reader is referred to [5] for a nice survey of such results.

In a recent article [9], deciding bisimulation equivalence has been shown to be undecidable for *systems with indefinite number of identical processes* with respect to *propositional linear temporal logic* even without the next-time operator. (Two systems are said to be bisimulation equivalent if for every formula f written in propositional linear temporal logic, the sets of computations of the two systems satisfying f are identical.) This work can be thought of as an extension to that of [8, 21] in which the problem of determining whether a given system satisfies a specification given in *propositional linear temporal logic* without next-time operator (i.e., the *model checking problem*) for systems with many identical processes has been investigated.

In this paper, we study the decidability and complexity issues of deciding bisimulation and trace equivalences for systems with many identical processes with respect to the following structures of interprocess communication: *fully-connected* topology, *star-like* topology, and *linear* topology. In our setting, each system consists of an arbitrary number of finite-state processes using *Milner's CCS* as the style of interprocess communication. Given two such systems, our main interest lies in deciding whether the behaviors of two finite-state processes (taken from the two given systems) are *bisimulation equivalent* (or *trace equivalent*). Our results are summarized in Table 1. Despite the similarity in the underlying model, our work differs from that of [9] in many aspects. What follows are the primary differences. First, all the processes in our model are identical. In [9], however, a so-called *synchronizer* is in presence whose structure may differ from the remaining *user* processes. Second, our underlying notions of bisimulation and trace equivalences are identical to those defined in the literature [5] (tailored to the model of systems with many identical processes), whereas the notion of 'bisimulation' defined in [9] was built upon linear temporal logic (in their setting, two systems are 'bisimulation equivalent' iff they "witness" the same set of formulas). Finally, our results reveal an additional dimension of complexity regarding systems with many identical processes, namely, the structure through which processes communicate. Our results suggest that even in the absence of temporal logic, deciding bisimulation (as well as trace) equivalence is undecidable for star-like and linear systems.

Table 1
Complexities of the equivalence problem for a variety of systems with many identical processes

	Disimulation equivalence	Trace equivalence
Fully connected	PTIME-complete	PSPACE-complete
Star-like	Undecidable	Undecidable
Linear	Undecidable	Undecidable

In contrast, it is not clear whether the hardness (more precisely, undecidability) result of [9] results from the model of many identical processes itself, or from the expressiveness of temporal logic. For more about systems with many identical processes, the interested reader is referred to [1,3,7–10,21,23].

The remainder of this paper is organized as follows. In Section 2, we give the definitions of systems with many identical processes as well as the notions of trace and bisimulation equivalences for such systems. Section 3 concerns itself with the complexities of deciding trace and bisimulation equivalences for systems with a fully connected interprocess communication structure. For systems connected in a star-like (respectively, linear) fashion, deciding trace and bisimulation equivalences will be shown to be undecidable in Section 4 (respectively, Section 5).

2. Definitions

We begin with the definition of *systems with many identical processes* (or simply *systems* if they are clear from the context). Our model is essentially the one proposed in [8]. A *process* is a 6-tuple $P = (Q, \delta, \Sigma, q_0, \Gamma, L)$, where

- Q is a finite set of *states*,
- $q_0 (\in Q)$ is the *initial state*,
- $\delta (\subseteq Q \times Q \times \{+,-\} \times \Sigma)$ is the *transition relation*,
- Σ is a finite set of *messages*,
- Γ: a finite set of *labels*, and
- L: $(\delta \to \Gamma)$ a labeling function which assigns labels to transitions.

In our model, interprocess communication is based on the "hand-shaking" notion. Given a "c" in Σ, we can think of $-c$ as the action of *sending* message c and $+c$ as the action of *receiving* message c. For convenience, we denote (q, q', c) as $q \xrightarrow{c} q'$. We write $\overline{c}$ to denote the complement of c, i.e., $\overline{+c} = -c$ and $\overline{-c} = +c$.

Given two systems S_1 and S_2 and two designated processes P_1 and P_2 (in S_1 and S_2, respectively), our main concern in this paper is to determine whether P_1 and P_2 are bisimulation (or trace) equivalent in the presence of arbitrary numbers of identical processes in their respective systems with respect to the following interprocess communication structures:

1. fully connected topology (see Fig. 1(a)),

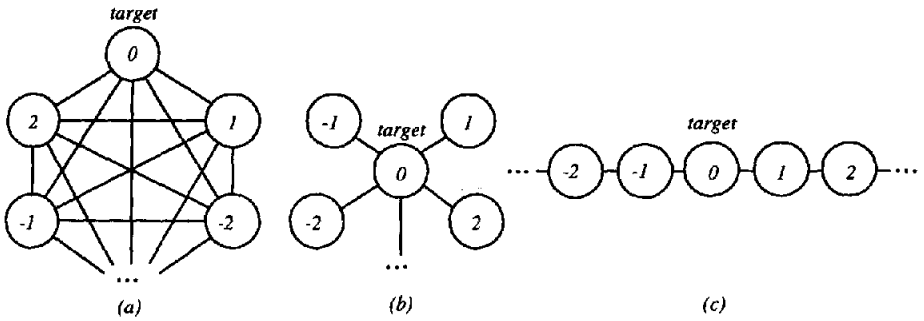


Fig. 1. System topology.

2. star-like topology (see Fig. 1(b)), and
3. linear topology (see Fig. 1(c)).

The designated process is referred to as the *target* process throughout this paper. Notice that the target and the remaining processes are identical.

Given a process P, we write P^{F} (P^{S} and P^{L}, respectively) to represent the system consisting of an arbitrary number of process P connected in a fully connected (star-like and linear, respectively) fashion, or P^{∞} if the underlying interconnection topology is not important. We also write $\{P^{\mathrm{F}}\}$, $\{P^{\mathrm{S}}\}$, and $\{P^{\mathrm{L}}\}$ to denote the classes of fully connected, star-like, and linear systems, respectively. In order to define the notion of a global state in a rigorous manner, we assign a unique integer in Z (the set of integers) to each process as indicated in Fig. 1. Notice that, without loss of generality, the target process is labeled 0. For $\{P^{\mathrm{F}}\}$ and $\{P^{\mathrm{S}}\}$, the labeling scheme for the remaining system is arbitrary; for $\{P^{\mathrm{L}}\}$, however, we assume the labeling scheme depicted in Fig. 1(c).

A *global state* s of a system P^{∞}, where $P=(Q,\delta,\Sigma,q_0,\Gamma,L)$, is a mapping from Z to Q such that $s(i)$ represents the current state of the process labeled i. Initially, the system is in its *initial state* s_0 with $s_0(i)=q_0, \forall i \in Z$. For convenience, we write $S(P^{\infty})$ to denote the set of all global states of system P^{∞}. Given two global states s,s', and an action symbol $c \in \Sigma$, we say processes i and j, for some $i,j \in Z$, can communicate through the exchange of action symbol c in state s if the following hold:

(1) Processes i and j are connected to each other,

(2) $s(i) \xrightarrow{+c} q$ (resp., $s(i) \xrightarrow{-c} q$) and $s(j) \xrightarrow{-c} q'$ (resp., $s(j) \xrightarrow{+c} q'$) are defined in states $s(i)$ and $s(j)$, respectively, for some q and q', *and*

(3) $$s'(l)=\begin{cases} q, & l=i,\\ q', & l=j,\\ s(l), & \text{otherwise.}\end{cases}$$

In this case, we write $s \mapsto^{c}_{\{i,j\}} s'$, or simply $s \xmapsto{c} s'$ if process names i and j are not important. A (global) transition $s \mapsto^{c}_{A} s'$ is said to be of *type* 1 if $0 \in A$ (i.e., the target process is involved); otherwise, it is of *type* 2.

A (*global*) *computation* is a sequence $\sigma : s_0 \mapsto^{a_1}_{A_1} s_1 \mapsto^{a_2}_{A_2} \cdots \mapsto^{a_n}_{A_n} s_n$, where $s_0, s_1, \ldots, s_n$ are global states and $\forall i, a_i \in \Sigma$. (We sometimes write $s_0 \xmapsto{\sigma} s_n$ as a

shorthand for the above.) Global state s_n is said to be *reachable* through computation σ. A local state q is said to be *reachable* if there exists a computation σ such that $s_0 \stackrel{\sigma}{\longmapsto} s$ and $s(i) = q$, for some $i \in Z$. Given a global computation $\sigma : s_0 \mapsto^{a_1}_{A_1} s_1 \mapsto^{a_2}_{A_2} \cdots \mapsto^{a_n}_{A_n} s_n$, we define $\rho(\sigma)$ to be a string $b_1 b_2 \ldots b_n \in \Gamma^*$ such that

$$b_i = \begin{cases} L(t) & \text{if } 0 \in A_i \text{ and } t \text{ is the transition performed by} \\ & \text{the target process (i.e., process 0) at } s_{i-1}, \\ \lambda \text{ (the empty string)} & \text{otherwise.} \end{cases}$$

Intuitively, $\rho(\sigma)$ is the projection of transition labels of σ on the target process. Given a system P^∞, where $P = (Q, \delta, \Sigma, q_0, \Gamma, L)$, the *trace set* of P^∞, denoted by $Trace(P^\infty)$, is $\{\rho(\sigma) \mid \sigma \text{ is a global computation in } P^\infty\}$.

Given two systems P_1^∞ and P_2^∞, where $P_1 = (Q_1, \delta_1, \Sigma_1, q_0^1, \Gamma_1, L_1)$ and $P_2 = (Q_2, \delta_2, \Sigma_2, q_0^2, \Gamma_2, L_2)$, a "bisimulation" is an equivalence relation R over $S(P_1^\infty) \times S(P_2^\infty)$ such that for every $(s_1, s_2) \in R$,

(1) for each $s_1 \stackrel{\sigma_1}{\longmapsto} s_1'$ and $\rho(\sigma_1) = a$ $(\in \Gamma_1)$, there exists an s_2', $s_2 \stackrel{\sigma_2}{\longmapsto} s_2'$, $\rho(\sigma_2) = a$, and $(s_1', s_2') \in R$, *and*

(2) for each $s_2 \stackrel{\sigma_2}{\longmapsto} s_2'$ and $\rho(\sigma_2) = a$ $(\in \Gamma_2)$, there exists an s_1', $s_1 \stackrel{\sigma_1}{\longmapsto} s_1'$, $\rho(\sigma_1) = a$, and $(s_1', s_2') \in R$.

Two systems P_1^∞ and P_2^∞ are said to be *trace equivalent*, denoted by $P_1^\infty \stackrel{T}{\sim} P_2^\infty$, iff $Trace(P_1^\infty) = Trace(P_2^\infty)$. P_1^∞ and P_2^∞ are *bisimilar* (or *bisimulation equivalent*), denoted by $P_1^\infty \stackrel{B}{\sim} P_2^\infty$, iff there exists a bisimulation R such that $(s_0^1, s_0^2) \in R$, where s_0^1 and s_0^2 are the initial global states of P_1^∞ and P_2^∞, respectively. The *bisimulation equivalence* (resp., *trace equivalence*) problem is that of, given two systems with many identical processes P_1^∞ and P_2^∞, determining whether $P_1^\infty \stackrel{B}{\sim} P_2^\infty$ (resp., $P_1^\infty \stackrel{T}{\sim} P_2^\infty$).

3. Fully connected systems

In this section, we derive the complexities of deciding trace and bisimulation equivalences for $\{P^F\}$ (i.e., fully connected systems with many identical processes). As it turns out, the trace equivalence problem is complete for PSPACE, whereas the bisimulation equivalence problem is PTIME-complete. The idea behind our derivation relies on showing that, given two systems P_1^F and P_2^F, we can construct (in polynomial time) two finite-state processes $\tilde{P}_1$ and $\tilde{P}_2$ in such a way that $P_1^F \stackrel{T}{\sim} P_2^F$ (resp., $P_1^F \stackrel{B}{\sim} P_2^F$) iff $\tilde{P}_1 \stackrel{T}{\sim} \tilde{P}_2$ (resp., $\tilde{P}_1 \stackrel{B}{\sim} \tilde{P}_2$). That is, deciding trace and bisimulation equivalences for fully connected systems with many identical processes can be equated with that for finite-state processes. As a result, our complexity results follow immediately from known results concerning the latter [20]. Before going into the details, we define trace equivalence and bisimulation equivalence for finite-state processes first (see [20] for more details).

A *finite-state process* $\tilde{P}$ is a four-tuple (Q, δ, Σ, q_0), where Q is a finite set of states, Σ is a finite set of symbols, q_0 $(\in Q)$ is the initial state, and δ defines the transition

function. A computation is a sequence $\sigma : q_0 \xrightarrow{c_1} q_1 \xrightarrow{c_2} \cdots \xrightarrow{c_n} q_n$, where $q_i \in Q$ and $c_i \in \Sigma$, $q_{i-1} \xrightarrow{c_i} q_i$ is defined in δ, for all $1 \leqslant i \leqslant n$. We write $q_0 \overset{\sigma}{\longmapsto} q_n$ as a shorthand for the above. The *label* of σ, written as $\rho(\sigma)$, is $c_1 c_2 \cdots c_n$. The *trace* of $\tilde{P}$, written as $Trace(\tilde{P}) = \{\rho(\sigma) \mid q_0 \overset{\sigma}{\longmapsto} q_n$, for some $q_n\}$. (Notice that the trace set is exactly the *language* accepted by the finite-state process, assuming that all states are accepting states.) Given two finite-state processes $\tilde{P}_1 = (Q_1, \delta_1, \Sigma_1, q_0^1)$ and $\tilde{P}_2 = (Q_2, \delta_2, \Sigma_2, q_0^2)$, $\tilde{P}_1$ and $\tilde{P}_2$ are *trace equivalent*, written as $\tilde{P}_1 \overset{T}{\sim} \tilde{P}_2$, iff $Trace(\tilde{P}_1) = Trace(\tilde{P}_2)$. A *bisimulation* is an equivalence relation $\tilde{R}$ over $Q_1 \times Q_2$ such that for every $(q_1, q_2) \in \tilde{R}$,

1. for each $q_1 \xrightarrow{c} q_1'$, there exists $q_2', q_2 \xrightarrow{c} q_2'$ and $(q_1', q_2') \in \tilde{R}$.
2. for each $q_2 \xrightarrow{c} q_2'$, there exists $q_1', q_1 \xrightarrow{c} q_1'$ and $(q_1', q_2') \in \tilde{R}$,

$\tilde{P}_1$ and $\tilde{P}_2$ are *bisimulation equivalent*, denoted by $\tilde{P}_1 \overset{B}{\sim} \tilde{P}_2$, if there is a bisimulation $\tilde{R}$ such that $(q_0^1, q_0^2) \in \tilde{R}$.

For $\{P^F\}$, any pair of processes are connected to each other; hence, a simpler form will be used in this section to describe a global state. Given a system P^F, where $P = (Q, \delta, \Sigma, q_0, \Gamma, L)$, we establish an ordering on Q by letting $Q = \{q_0, q_1, \ldots, q_k\}$, for some k. Throughout the rest of this paper, we assume the existence of such an ordering. A *global state* is a $(k+1)$-*tuple* $s = \langle q, n_1, \ldots, n_k \rangle$, where q represents a state of the target process, and n_i $(\in N)$ denotes the number of processes in state q_i in the remainder of the system. Notice that the number of processes in state q_0 need not be kept in the global state description, for it is always arbitrarily large. For ease of explanation, we use $[s]$ to denote q, and $\langle s \rangle$ to denote the k-dimensional vector $(n_1, \ldots, n_k)$. (Hence, $\langle s \rangle(i) = n_i$.) Initially, the system is in global state $s_0 = \langle q_0, 0, \ldots, 0 \rangle$.

To prove our main results, a few lemmas are required.

Lemma 1. *Given a system P^F, the set of all reachable local states of P can be constructed in polynomial time.*

Proof. We begin with a procedure that generates the set of all reachable local states in a greedy fashion. The procedure was proposed in [23]; see also [8].

Procedure Reachable_Set;
/* Given the description of a process $P = (Q, \delta, \Sigma, q_0, \Gamma, L)$, */
/* the output $\mathcal{S}$ consists of all reachable local states. */
  $\mathcal{S} := \{q_0\}$;
  For $i := 0$ **to** $|Q| - 1$ **do**
    begin
      If $\exists p, q \in \mathcal{S}$, $p \xrightarrow{+c} p'$, $q \xrightarrow{-c} q' \in \delta$, for some $c \in \Sigma$, and p'(or q') $\notin \mathcal{S}$,
      then $\mathcal{S} := \mathcal{S} \cup \{p', q'\}$;
      else return $(\mathcal{S})$;
    end;
end procedure.

It is obvious that if states p and q of processes P and Q, respectively, are reachable through computation paths belonging to two distinct groups of processes, and $p \xrightarrow{c} p'$ and $q \xrightarrow{\bar{c}} q'$ are two transitions defined in p and q, respectively, then there is a global computation reaching a global state in which the states of P and Q are p' and q', respectively. Since the number of processes is arbitrary, the validity of the procedure is easy to observe, and the complexity of the procedure is clearly in polynomial time (in the size of $(Q, \delta, \Sigma, q_0, \Gamma, L)$). □

Throughout the rest of this section, the above set of reachable local states will be denoted as $RS(P^F)$. It is important to note that, if necessary, every transition emanating from a state in $RS(P^F)$ can take part in a (global) computation an arbitrary number of times. Again, this is due to the availability of an arbitrary number of processes.

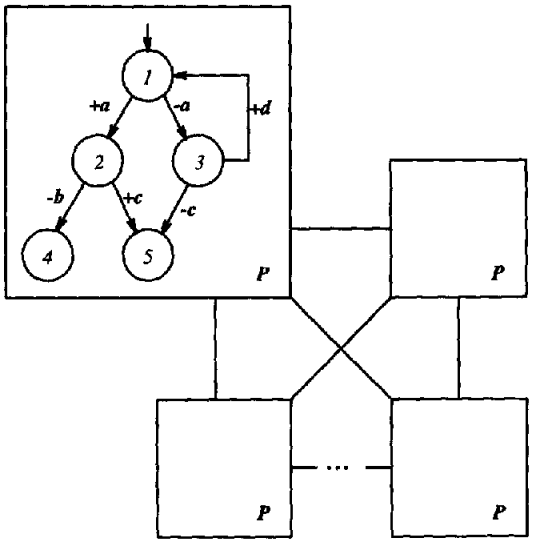
Lemma 2. *Given a system P^F and a global state s, for every $m > 0$, there exist a σ and a global state s' such that $s \overset{\sigma}{\longmapsto} s'$, $[s] = [s']$, $\langle s' \rangle(i) \geqslant m$, for every i such that $q_i \in RS(P^F)$, and all transitions in σ are of type 2.*

Proof. For any $q_i \in RS(P^F)$, q_i can be reached through a computation along which all transitions are of type 2. For any $m > 0$, we allocate m distinct groups of processes (excluding the target process) each of which participates in the above computation (i.e., reaching state q_i). By concatenating the computations of m such groups, a global state s_i ($s_0 \overset{\sigma_i}{\longmapsto} s_i$) with $\langle s_i \rangle(i) \geqslant m$ can clearly be reached for a given $q_i \in RS(P^F)$. Again by utilizing a distinct group of processes for each σ_i and letting $\sigma = \sigma_1 \sigma_2 \cdots \sigma_k$, the desired computation is constructed. Clearly, along this computation the target process remains idle; hence, $[s] = [s']$. □

As mentioned earlier, the key idea behind our derivation is to reduce the deciding equivalence between fully connected systems with many identical processes to that between finite-state processes. Consider a system P^F, where $P = (Q, \delta, \Sigma, q_0, \Gamma, L)$, the associated finite-state process $\tilde{P} = (\tilde{Q}, \tilde{\delta}, \tilde{\Sigma}, \tilde{q}_0)$ is constructed as follows:

- $\tilde{Q} = RS(P^F)$,
- $\tilde{q}_0 = q_0$
- $\tilde{\Sigma} = \Gamma$
- (q, q', l) (i.e., $q \xrightarrow{l} q'$), where $q, q' \in \tilde{Q}$ and $l \in \tilde{\Sigma}$, is defined in $\tilde{\delta}$ if there exist a $c \in \Sigma, r, r' \in RS(P^F)$, such that
  1. $(q, q', +, c), (r, r', -, c) \in \delta$ (or $(q, q', -, c), (r, r', +, c) \in \delta$), and
  2. $L((q, q', +, c)) = l$ (or $L(q, q', -, c) = l$).

The idea is to retain transition $(q, q', +, c)$ (or $(q, q', -, c)$) that can be paired with a transition emanating from a state $r \in RS(P^F)$. Furthermore, the label of such a transition is $L((q, q', +, c))$ (or $L((q, q', -, c))$). Also notice that the above construction can be carried out in polynomial time. Throughout the rest of this paper, $\tilde{P}$ is called the *reduced* finite-state process of system P^F. To give the reader a better feel for how the above construction is done, consider a simple example shown in Fig. 2. It is not hard



$L(1 \xrightarrow{+a} 2) = r$, $L(1 \xrightarrow{-a} 3) = s$, $L(2 \xrightarrow{-b} 4) = r$,
$L(2 \xrightarrow{+c} 5) = t$, $L(3 \xrightarrow{-c} 5) = t$, $L(3 \xrightarrow{+d} 1) = r$.

Fig. 2. System P^F.

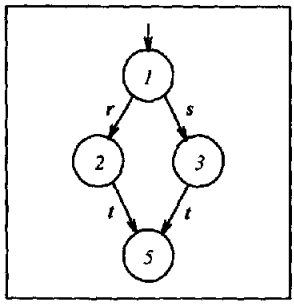


Fig. 3. The reduced finite-state process $\tilde{P}$.

to see that the set of all reachable local states for P^F shown in Fig. 2 is $RS(P^F) = \{1,2,3,5\}$. Hence, the reduced finite-state process $\tilde{P}$ is the one shown in Fig. 3.

Lemma 3. *Let $\tilde{P}$ be the reduced finite-state process of a system P^F, and s (with $[s] = q_i \in \tilde{P}$, for some i) be an arbitrary global state of P^F. Then for every transition $q_i \xrightarrow{c} q_j$, for some j, in $\tilde{P}$, there exists a sequence σ in P^F such that $s \xmapsto{\sigma} s'$, $[s'] = q_j$ and $\rho(\sigma) = c$.*

Proof. Let $q_i \xrightarrow{t} q_j$ (in P) be the transition from which $q_i \xrightarrow{c} q_j$ (in $\tilde{P}$) is obtained in the construction from P^F to $\tilde{P}$. (For simplicity, the sign of t is omitted. Also notice that $L(q_i \xrightarrow{t} q_j) = c$.) By the construction of $\tilde{P}$, there must exist a reachable local state q_r in which $q_r \xrightarrow{\tilde{t}} q_u$ is defined, for some q_u. According to Lemma 2, there exists a sequence $\sigma', s \xmapsto{\sigma'} s''$ $[s''] = q_i$, such that $\langle s'' \rangle(r) > 0$, and σ' (if not empty) utilizes only type 2 transitions. Let $\sigma = \sigma' t$. Clearly, $s \xmapsto{\sigma} s'$, for some s' with $[s'] = q_j$ and $\rho(\sigma) = c$. □

By repeatedly applying Lemma 3, we can easily show the following.

Lemma 4. *Given a system P^F, let $\tilde{P}$ be the reduced finite-state process mentioned above. Then for any $\xi \in Trace(\tilde{P})$, there exists a sequence σ in P^F such that $\rho(\sigma) = \xi$.*

Corollary 1. *$Trace(P^F) = \text{Trace}(\tilde{P})$, where $\tilde{P}$ is the reduced finite-state process of a system P^F.*

Proof. *$Trace(P^F) \subseteq Trace(\tilde{P})$* obviously holds from the definition of $\tilde{P}$; *$Trace(\tilde{P}) \subseteq Trace(P^F)$* follows immediately from Lemma 4. □

Theorem 1. *Given two systems P_1^F and P_2^F, deciding whether $P_1^F \overset{T}{\sim} P_2^F$ can be done in* PSPACE.

Proof. From Corollary 1, testing $P_1^F \overset{T}{\sim} P_2^F$ is tantamount to testing $\tilde{P}_1 \overset{T}{\sim} \tilde{P}_2$, which is known to be doable in PSPACE [20]. □

In what follows, we consider bisimulation equivalence.

Lemma 5. *Given two systems P_1^F and P_2^F, let $\tilde{P}_1$ and $\tilde{P}_2$ be their reduced finite-state processes, respectively. Then $P_1^F \overset{B}{\sim} P_2^F$ iff $\tilde{P}_1 \overset{B}{\sim} \tilde{P}_2$.*

Proof. Let $P_i = (Q_i, \delta_i, \Sigma_i, q_0^i, \Gamma_i, L_i)$, and $\tilde{P}_i = (Q_i, \tilde{\delta}_i, \tilde{\Sigma}_i, q_0^i)$, where $i = 1$ or 2.
($P_1^F \overset{B}{\sim} P_2^F \Rightarrow \tilde{P}_1 \overset{B}{\sim} \tilde{P}_2$). Let R be a bisimulation relation between P_1^F and P_2^F. Define a relation $\tilde{R} = \{(q^1, q^2) \mid (s_1, s_2) \in R$, where s_1 and s_2 are reachable global states of P_1^F and P_2^F, respectively, $[s_1] = q^1$ and $[s_2] = q^2\}$. In what follows, we show $\tilde{R}$ to be a relation witnessing the bisimilarity between $\tilde{P}_1$ and $\tilde{P}_2$.

Clearly, $(q_0^1, q_0^2) \in \tilde{R}$, where q_0^1 and q_0^2 are the initial states of $\tilde{P}_1$ and $\tilde{P}_2$, respectively. Now suppose $(q^1, q^2) \in \tilde{R}$. By the definition of $\tilde{R}$, there must exist reachable global states s_1 and s_2 of P_1^F and P_2^F, respectively, such that $(s_1, s_2) \in R$, $[s_1] = q^1$ and $[s_2] = q^2$. Consider the following two cases.

Case 1: Suppose $q^1 \xrightarrow{c} r^1$, for some $c \in \tilde{\Sigma}_1$ and $r^1 \in Q_1$. By Lemma 3, there exists a computation γ_1 in P_1^F such that $s_1 \xmapsto{\gamma_1} u_1$, for some global state u_1 with $[u_1] = r^1$, and $\rho(\gamma_1) = c$. Since $P_1^F \overset{B}{\sim} P_2^F$, there must be a computation γ_2 in P_2^F such that $s_2 \xmapsto{\gamma_2} u_2$, for some global state u_2 with $[u_2] = r^2$, and $\rho(\gamma_2) = c$. By the construction of $\tilde{P}_2, q^2 \xrightarrow{c} r^2$.

Since $s_1 \overset{\gamma_1}{\longmapsto} u_1, s_2 \overset{\gamma_2}{\longmapsto} u_2$, $\rho(\gamma_1) = \rho(\gamma_2) = c$, and u_1 and u_2 are reachable from the initial states of P_1^F and P_2^F, respectively, we immediately have $(r^1, r^2) \in \tilde{R}$ (by the definition of $\tilde{R}$).

Case 2: Suppose $q^2 \xrightarrow{c} r^2$. Using a similar argument as that in the proof of Case 1, we can show the existence of an r^1 in $\tilde{P}_1$ such that $q^1 \xrightarrow{c} r^1$ and $(r^1, r^2) \in \tilde{R}$. ($\tilde{P}_1 \overset{B}{\sim} \tilde{P}_2 \Rightarrow P_1^F \overset{B}{\sim} P_2^F$). Let $\tilde{R}$ be a bisimulation relation between $\tilde{P}_1$ and $\tilde{P}_2$. Define a relation $R = \{(s_1, s_2) \mid s_1$ and s_2 are reachable global states of P_1^F and P_2^F, respectively, and $([s_1], [s_2]) \in \tilde{R}\}$. In what follows, we show R to be a bisimulation between P_1^F and P_2^F. Suppose $(s_1, s_2) \in R$. By definition, $([s_1], [s_2]) \in \tilde{R}$. Consider two cases:

Case 1: $s_1 \overset{\gamma_1}{\longmapsto} u_1$ and $\rho(\gamma_1) = c$, for some label c. By the definition of $\tilde{P}_1$, $[s_1] \xrightarrow{c} [u_1]$ is a transition in $\tilde{P}_1$. Given the fact that $\tilde{P}_1 \overset{B}{\sim} \tilde{P}_2$, and $([s_1], [s_2]) \in \tilde{R}$, there exists a state $r^2 \in Q_2$ such that $[s_2] \xrightarrow{c} r^2$ in $\tilde{P}_2$ and $([u_1], r^2) \in \tilde{R}$. According to Lemma 3, there exists a computation γ_2 in P_2^F such that $s_2 \overset{\gamma_2}{\longmapsto} u_2$ for some global state u_2 such that $[u_2] = r^2$ and $\rho(\gamma_2) = c$. Now by the definition of R, $(u_1, u_2) \in R$ (because both u_1 and u_2 are reachable and $([u_1], [u_2]) \in \tilde{R}$).

Case 2: $s_2 \overset{\gamma_2}{\longmapsto} u_2$ and $\rho(\gamma_2) = c$, for some c. A similar argument can be used to show the existence of a computation $s_1 \overset{\gamma_1}{\longmapsto} u_1$, for some γ_1 and u_1, such that $\rho(\gamma_1) = c$ and $(u_1, u_2) \in R$. □

Lemma 5, in conjunction with the work of [20], yields the following.

Theorem 2. *Given two systems P_1^F and P_2^F, deciding whether $P_1^F \overset{B}{\sim} P_2^F$ can be done in* PTIME.

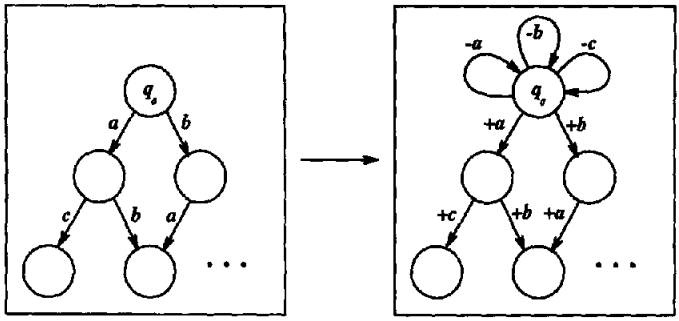
Now we show the lower bound.

Lemma 6. *Given two finite-state processes $\tilde{P}_1$ and $\tilde{P}_2$, we can construct, in polynomial-time, two systems P_1^F and P_2^F such that $\tilde{P}_1 \overset{T}{\sim} \tilde{P}_2$ (resp., $\tilde{P}_1 \overset{B}{\sim} \tilde{P}_2$) iff $P_1^F \overset{T}{\sim} P_2^F$ (resp., $P_1^F \overset{B}{\sim} P_2^F$).*

Proof. The construction is rather straightforward. P_1 retains all the states and transitions of $\tilde{P}_1$ except that each transition symbol, say "a", is replaced by "$+a$". In addition, if "a" is a transition symbol used in $\tilde{P}_1$, then add a self-loop labeled "$-a$" to the initial state of P_1; see Fig. 4. Finally, let the label of a transition be the same as the associated message type. P_2 is constructed exactly the same way. The remaining details are left to the reader. □

Using the above lemma and the fact that deciding trace and bisimulation equivalence are PSPACE-complete [20] and PTIME-complete [20], respectively, we have the following result.

Theorem 3. *Given two systems P_1^F and P_2^F, deciding $P_1^F \overset{T}{\sim} P_2^F$ and $P_1^F \overset{B}{\sim} P_2^F$ are* PSPACE-*hard and* PTIME-*hard, respectively.*



$\tilde{P}$ P

Fig. 4. An example illustrating the construction.

4. Star-like systems

To show the bisimulation equivalence problem with a "star-like" structure (see Fig. 1) to be undecidable, we reduce the halting problem for *2-counter machines* [12] to our problem. (See [15, 18] for similar proofs for the models of Petri nets and BPPs.)

Before we go further into the detailed proof, first consider the relation between a system with many identical processes system and a 2-counter machine. In a 2-counter machine, there are three operations associated with a counter: *increment*, *decrement*, and *test for zero*. More precisely, a transition of a 2-counter machine is of one of the following forms:

(1) $C := C + 1$; goto q (add one to counter C),

(2) if $C > 0$ then $C := C - 1$; goto q (subtract one from counter C),

(3) if $C = 0$ then goto q (test counter C for zero).

To simulate a 2-counter machine using a system with many identical processes, one's first attempt might be to use the structure depicted in Fig. 5 in which the target process enters the box (through the exchange of message "start") which plays the role of the finite-state control of the 2-counter machine. The remaining processes (i.e. those emanating from the center of the star) execute those transitions that are above the box (see Fig. 5). Furthermore, the values of the two counters are represented by the numbers of processes in states q_1 and q_2, respectively. The "$+i_1$" and "$+d_1$" (resp., "$+i_2$" and "$+d_2$") transitions correspond to the actions of "adding one to" and "subtracting one from" the first counter (resp., the second counter). A careful examination, however, reveals the insufficiency for the above mechanism to faithfully simulate "test for zero" transitions. More precisely, we cannot prevent the target process from cheating on a "test for zero" transition, by performing the $r \xrightarrow{-a} r''$ transition while the first counter is not zero. In view of the above, it seems that a system with many identical processes does not have enough power to simulate a 2-counter machine. Our next theorem says that if we are given two systems instead, the notion of bisimulation

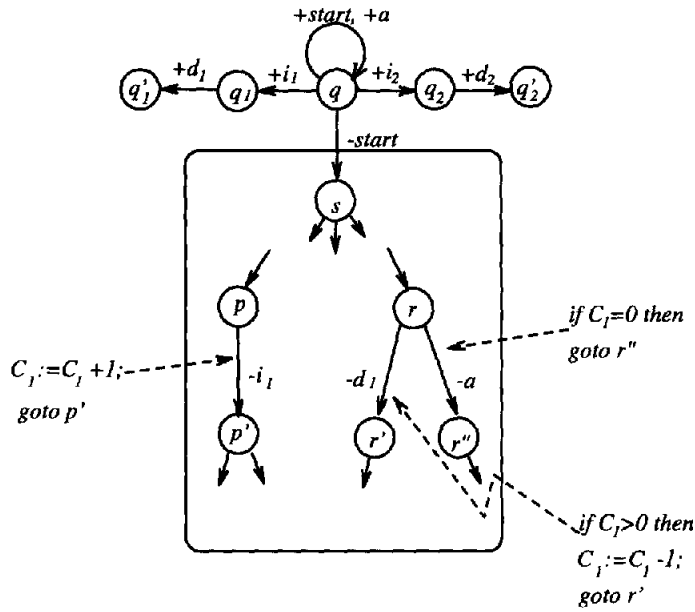


Fig. 5. Process "simulating" a 2-counter machine.

equivalence is sufficient to "force" one of the two systems to behave exactly like a 2-counter machine.

Theorem 4. *The bisimulation equivalence problem is undecidable for* $\{P^S\}$.

Proof. It suffices to show that, given a 2-counter machine M, we can construct two systems P_1^S and P_2^S in a way that *M does not halt iff P_1^S and P_2^S are bisimilar.* Let C_1 and C_2 be the two counters which are zero initially. As mentioned earlier, additions and subtractions of the two counters can be done easily; in what follows, we focus on how we can enforce the "test for zero" operation through the use of two systems on which the bisimilarity requirement is imposed.

As shown in Fig. 6, process P_1 consists of two copies of the finite-state control of M. (For convenience, they are labeled $M_{1,1}$ and $M_{1,2}$.) Let states $\overline{h_1}, \overline{r_1}, \overline{r_1}'$, and $\overline{r_1}''$ be the "images" of states h_1, r_1, r_1', and r_1'', respectively. (The "h_1" represents the halt state of M.)

Suppose "if $C_1=0$ then goto r_1''" is defined in state r_1 of M, then $M_{1,1}$ and $M_{1,2}$ have transitions $r_1 \xrightarrow{-a} r_1''$ and $\overline{r_1} \xrightarrow{-a} \overline{r_1}''$, respectively. (These transitions can be paired with the "$\xrightarrow{+a}$" transition defined in state q_0^1.) In addition, we add a transition from r_1 to $\overline{r_1}''$ labeled $-e$ (see Fig. 6). Also we let $L(r_1 \xrightarrow{-a} r_1'') = L(r_1 \xrightarrow{-e} \overline{r_1}'')$. P_2 is identical to P_1 except with an additional transition labeled $-a$ from h_2 to z_2. We let

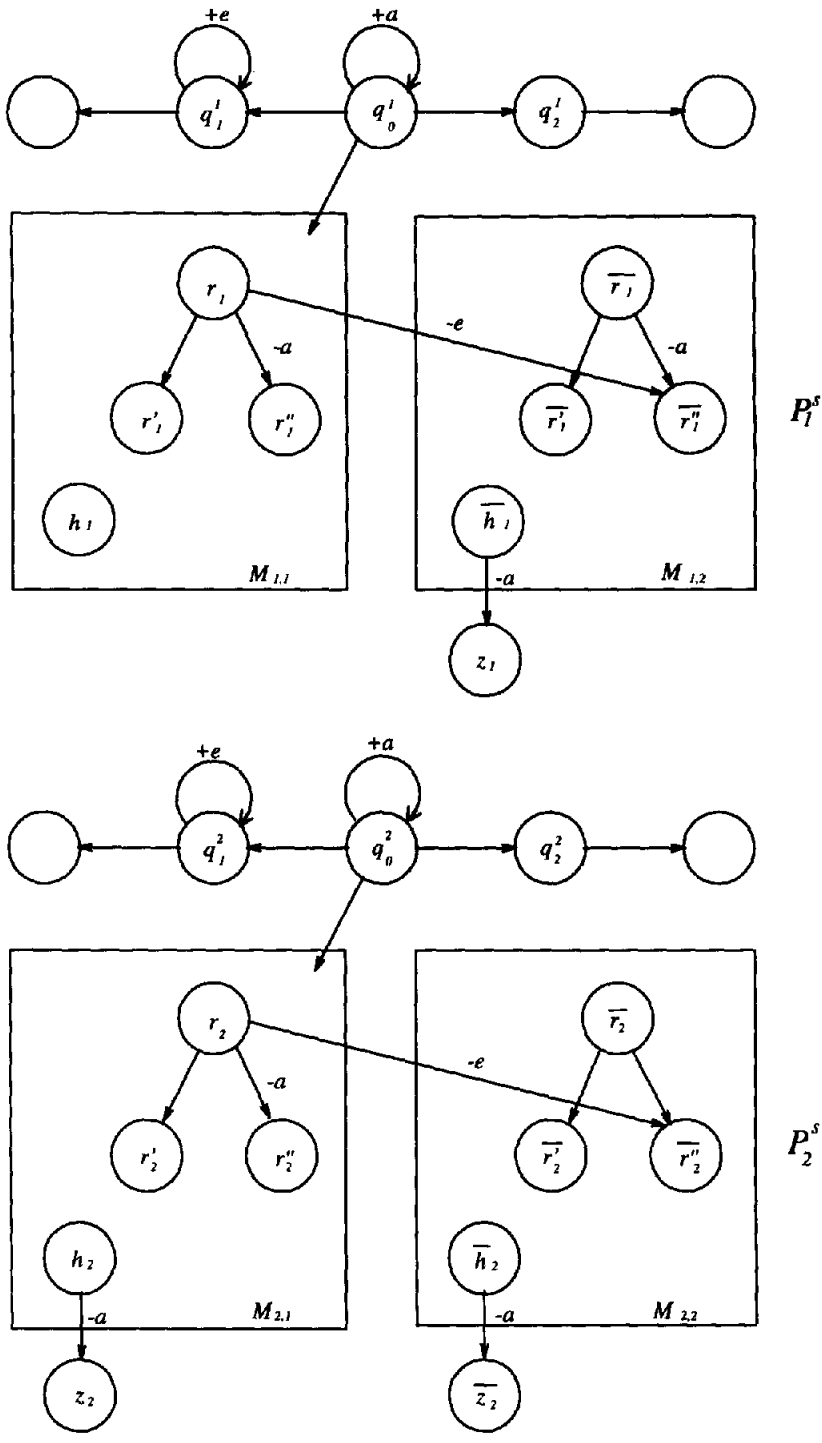


Fig. 6. The structures of P_1^s and P_2^s.

$L(\overline{h_1} \xrightarrow{-a} \overline{z_1})$ $(= L(h_2 \xrightarrow{-a} z_2) = L(\overline{h_2} \xrightarrow{-a} \overline{z_2}))$ be a unique symbol not used elsewhere in P_1^S and P_2^S.

A computation in P_1^S is said to be "valid" if during the course of the computation, the following hold:

1. the target process utilizes only those transitions belonging to $M_{1,1}$, and

2. when taking a transition like the $r_1 \xrightarrow{-a} r_1''$ (in Fig. 6) which simulates a "test for zero" for counter C_1 (resp., C_2), none of the remaining processes is in state q_1^1 (resp., q_2^1) at the moment. (What it says is that any "test for zero" transition can only be taken while the corresponding counter of M is indeed zero.)

A "valid" computation for P_2^S can be defined similarly.

We are in a position to argue that M does not halt iff $P_1^S \overset{B}{\sim} P_2^S$. Consider the following two cases.

Case 1: Suppose M halts. To disprove $P_1^S \overset{B}{\sim} P_2^S$, first notice that F_2^S can do whatever F_1^S does. Since M halts, there exists a "valid" computation in P_2^F reaching h_2 using only those transitions belonging to $M_{2,1}$. To "keep pace with" P_2^S (intuitively, this is exactly what bisimilarity is all about), P_1^S must follow exactly the same trace as that of P_2^S in a step by step fashion. (Here, it is important to point out that P_1^S cannot "cheat" by entering $M_{1,2}$, for none of the remaining processes can exchange message "e" with the target process – condition (2) of a valid computation.) In the end, P_1^S and P_2^S end up in states h_1 and h_2, respectively. Hence, P_1^S and P_2^S cannot be bisimilar.

Case 2: Suppose M does not halt. To prove $P_1^S \overset{B}{\sim} P_2^S$, consider the following subcases:

- P_2^S does not enter h_2. In this case, P_1^S can always follow suit.
- P_2^S enters h_2 through an invalid computation σ. (Notice that there exists no valid computation reaching h_2, for M does not halt.)

Suppose $r_2 \xrightarrow{-a} r_2''$ is the first test-for-zero transition (in σ) at which P_2^S cheats, i.e., executing a test-for-zero transition while the respective counter is non-zero. In our design, $L(r_1 \xrightarrow{-e} \overline{r_1''}) = L(r_1 \xrightarrow{-a} r_1'') = L(r_2 \xrightarrow{-a} r_2'')$. P_1^S, in this case, can take the $r_1 \xrightarrow{-e} \overline{r_1}''$ transition pairing with the self-loop labeled "$+e$" in state q_1^1 of some of the remaining processes (because counter C_1 is not zero at this moment). From this point and beyond, P_1^S will stay in $M_{1,2}$, and therefore can do whatever P_2^S does.

In view of the above, we have that M does not halt iff $P_1^S \overset{B}{\sim} P_2^S$. Following the undecidability of the halting problem for 2-counter machines, the bisimulation equivalence problem for $\{P^S\}$ is also undecidable. □

Theorem 5. *The trace equivalence problem is undecidable for* $\{P^S\}$.

Proof. Using the same construction as the one used in the proof of Theorem 4, $P_1^S \overset{T}{\sim} P_2^S$ iff M does not halt. The details are left to the reader. □

5. Linear systems

Theorem 6. *The bisimulation equivalence and trace equivalence problems for* $\{P^L\}$ *are undecidable.*

Proof. In what follows, we show that 2-counter machine computations can be faithfully simulated by systems in $\{P^L\}$. The labeling scheme depicted in Fig. 1 is assumed here for naming processes in $\{P^L\}$.

Given a 2-counter machine M with two counters C_1 and C_2, the process structure designed to simulate M is shown in Fig. 7, which, for ease of explanation, is partitioned into five regions, namely, A, B, G_A, G_B, and D. For the sake of simplicity, messages used in regions B and G_B are omitted. Here, it is important to point out that messages used in A and G_A differ from that in B and G_B. Before presenting the detailed simulation, we first give the intuition. As before, region D is used for simulating the finite-state control of M. (In particular, transitions involving "$p - p'$", "$r - r'$", and "$r - r''$" are examples of "adding one to C_1", "subtracting one from C_1", and "testing C_1 for zero", respectively.) Regions A and B are designed to simulate the two counters. Region G_A and G_B serve as two "gateways" through which unwanted messages can be "filtered out" for performing counter operations correctly. More will be said about this as our discussion progresses. A process is said to be "dormant" if it is in state q_0 (i.e., the initial state of the process).

To simulate M, consider the following computation:

1. By exchanging messages "g_a" and "g_b" with its two neighbors, process 0 enters region D and starts simulating M's computations. Depending on the order in which P_0 communicates with P_{-1} and P_1, we have the following two cases:

(i) P_{-1} and P_1 enter regions G_A and G_B, respectively, or

(ii) P_{-1} and P_1 enter regions G_B and G_A, respectively.

In case (i), process P_i, $i \leqslant -2$ (respectively, $i \geqslant 2$), takes part in the simulation of counter C_1 (respectively, C_2).

Case (ii) is symmetric. Without loss of generality, we assume case (i) throughout the rest of the proof.

2. At any time, the number of processes P_i, $i \leqslant -2$, whose current states reside in region A represents the value of counter C_1. Furthermore, if the state of P_i, for some $i \leqslant -2$, is in region A, so are $P_{-2}, \ldots, P_i$. (In other words, if j, $j \leqslant -2$, is the smallest index such that $P_j's$ current state is not in A, then counter C_1 equals $|j| - 2$.)

3. At any time, the number of processes P_i, $i \geqslant 2$, whose current states reside in region B represents the value of counter C_2. Furthermore, if the state of P_i, for some $i \geqslant 2$, is in region B, so are $P_2, \ldots, P_i$. (In other words, if j, $j \geqslant 2$, is the smallest index such that $P_j's$ current state is not in B, then counter C_2 equals $j - 2$.)

We are in a position to describe the detailed simulation.

Increment C_1: The simulation begins with P_0 sending message "i" (using $p \xrightarrow{-i} 2$) to its left neighbor (i.e., P_{-1}) and ends with receiving message "*ack*" (using $2 \xrightarrow{+ack} p'$) from P_{-1}. Recall that P_{-1} is in region G_A at this moment. Upon receiving i from P_0

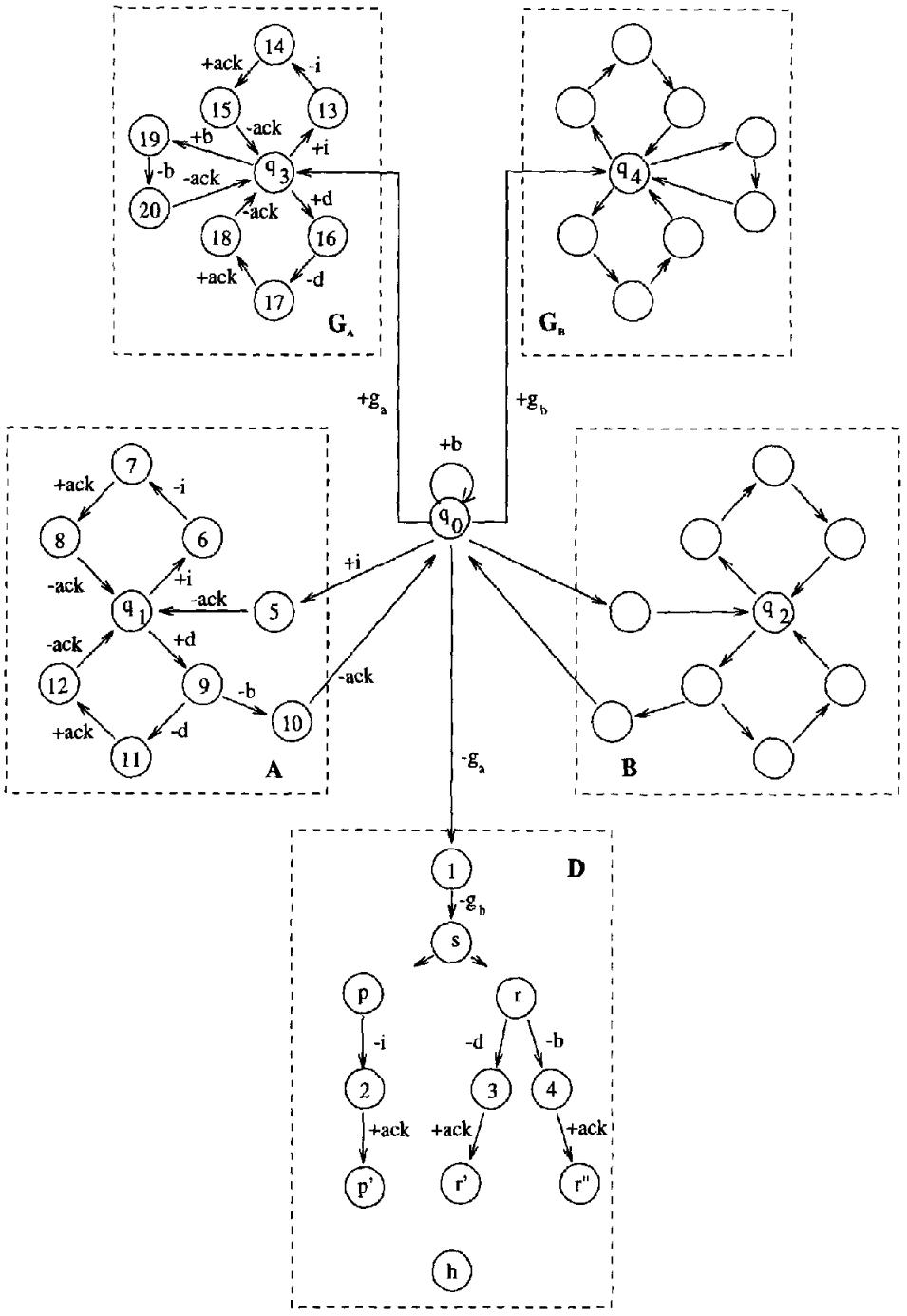


Fig. 7. The structure of the process used in proving the undecidability result for $\{P^L\}$.

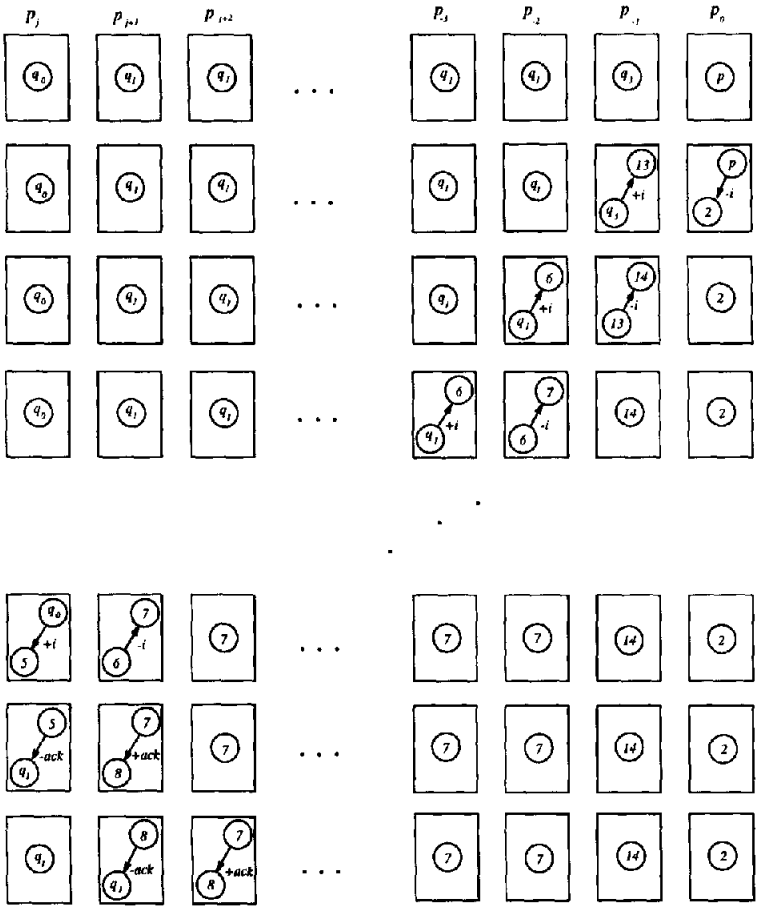


Fig. 8. Increment of C_1.

(using $q_3 \xrightarrow{+i} 13$), P_{-1} propagates message i using transition $13 \xrightarrow{-i} 14$ to P_{-2}. The propagation continues until a dormant process, say P_j, $j \leqslant -2$, is reached. P_j then executes $q_0 \xrightarrow{+i} 5 \xrightarrow{-ack} q_1$, constituting an increment in C_1. While doing so, message "*ack*" is sent to the right which eventually reaches P_0. See Fig. 8 for a pictorial description of some of the above moves (with irrelevant information omitted). It is worth pointing out that process P_{-1} serves as a "gateway" to filter out messages not belonging to those recognizable by P_{-2}, P_{-3},... (Recall that $P_{-2}, P_{-3}, \ldots$ simulate the content of counter C_1.) Without this gateway, however, it is possible for P_{-1}, P_0, P_1 to enter regions A, D, A, respectively, violating properties 2 and 3 mentioned earlier.

Decrement C_1: The simulation begins with P_0 sending message "d" (using $r \xrightarrow{-d} 3$) to P_{-1}, and ends with receiving "*ack*" (using $3 \xrightarrow{+ack} r'$) from P_{-1}. After passing through the gateway (i.e., P_{-1}), message "d" is propagated to the left (using transitions $q_1 \xrightarrow{+d}$

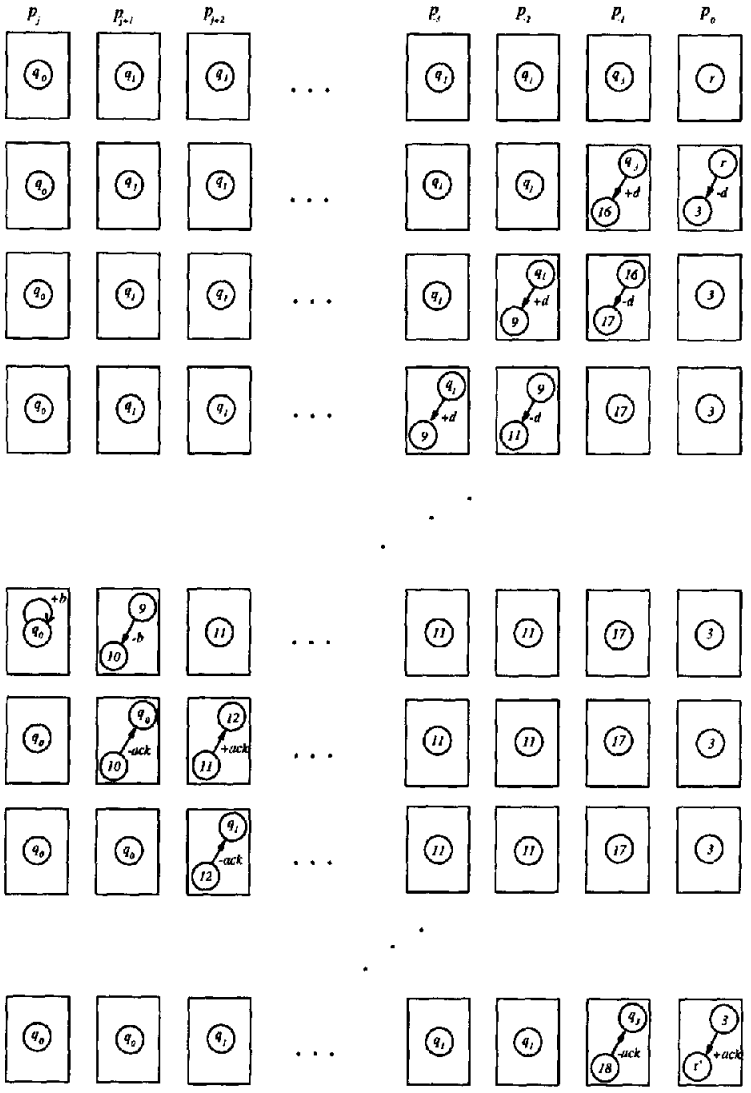


Fig. 9. Decrement of C_1.

$9 \xrightarrow{-d} 11$) until a dormant process, say P_j ($j \leqslant -2$), is reached. P_j and P_{j+1} exchange message "b"; P_{j-1} propagates message "*ack*" to the right and then becomes dormant, constituting a decrement in C_1. The remaining details are left to the reader; see also Fig. 9. Notice that if C_1 is empty, then P_{-1} and P_0 will get stuck in states 16 and 3, respectively.

Test C_1 for zero: It is reasonably easy to observe that the test for zero transition $r \xrightarrow{-b} 4 \xrightarrow{+ack} r''$ can be executed iff P_{-2} is dormant; see Fig. 7.

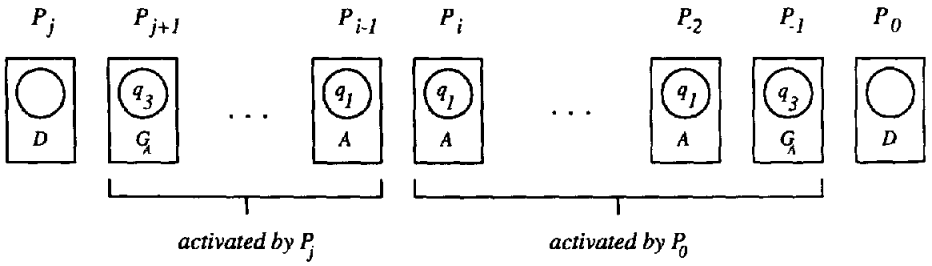


Fig. 10. Two processes entering region D.

What we have discussed so far is based on the assumption that exactly one process (i.e., P_0) enters region D. Furthermore, M halts iff there exists a computation leading P_0 to state h. To complete the proof, it remains to show that even in the presence of two or more processes entering region D, state h can only be reached through "valid" computation. In other words, if, for some reason, the operations of those processes activated by P_0 (in order to keep track of the contents of C_1 and C_2) are obstructed by other processes, then state h cannot be reached as a result of such an "invalid" computation. Take Fig. 10 for example. Processes $P_{-1},\ldots,P_i$ and $P_{i-1},\ldots,P_{j+1}$ are activated by P_0 and P_j, respectively, and both are simulating counter C_1. Now suppose P_0 issues an "add one to C_1" command. An i message will be sent from P_0 toward P_i; this message will pass through $P_{i-1},\ldots,P_{j+1}$ before reaching P_j, which is in region D. In our design, however, no "$+i$" can be paired with the arrival of this i message. As a consequence, the computation originated from P_0 gets stuck thereafter, which is perfectly okay for no further computation can mistakenly enter the halt state. The remaining cases in which normal computations are obstructed are left to the reader.

From the above discussion, it is easy to observe that M halts iff there exists a computation behaving in the above way, and eventually reaching the halt state h. Hence, systems in $\{P^L\}$ are powerful enough to simulate 2-counter machines; the two problems are undecidable. □

References

[1] P. Attie and E. Emerson, Synthesis of concurrent systems with many similar sequential processes, in: *Proc. POPL'89* (Austin, TX, 1989) 191–201.

[2] J. Balcázar, J. Gabarró and M. Sántha, Deciding bisimilarity is P-complete, *Formal Aspects Comput.* **4** (1992) 638–648.

[3] M. Browne, E. Clark and O. Grümberg, Reasoning about networks with many identical finite-state processes, *Inform. and Comput.* **81** (1989) 13–31.

[4] S. Christensen, Y. Hirshfeld and F. Moller, Bisimulation equivalence is decidable for basic parallel processes, in: *Proc. CONCUR'93*, Lecture Notes in Computer Science, Vol. 715 (Springer, Berlin, 1993) 143–157.

[5] S. Christensen and H. Hüttel, Decidability issues for infinite-state processes – a survey, *Bull. of the EATCS*, **51** (1993) 156–166.

[6] S. Christensen, H. Hüttel and C. Stirling, Bisimulation equivalence is decidable for all context-free process, *Inform. and Comput.* **121** (1995) 143–148.

[7] V. Garg, Analysis of distributed systems with many identical processes, in: *Proc. IEEE Int. Conf. on Distributed Computing Systems* (1988) 358–365.
[8] S. German and A. Sistla, Reasoning about systems with many processes, *J. ACM* **39** (1992) 675–735.
[9] M. Girkar and R. Moll, Undecidability of bisimulations in concurrent systems with indefinite number of identical processes, CMPSCI Tech. Report 93-86, Dept. of Computer Science, Univ. of Massachusetts, December 1993.
[10] M. Girkar and R. Moll, New results on the analysis of concurrent systems with an indefinite number of processes, in: *Proc. CONCUR'94*, Lecture Notes in Computer Science, Vol. 836 (Springer, Berlin, 1994).
[11] J. Groote and H. Hüttel, Undecidable equivalences for basic process algebra, *Inform. and Comput.* **115** (1994) 354–371.
[12] Y. Hirshfeld, Petri nets and the equivalence problem, in: *Proc. CSL'93, 1993 Conf. of European Association for Computer Science Logic*, Lecture Notes in Computer Science, Vol. 832 (Springer, Berlin, 1994) 165–174.
[13] Y. Hirshfeld, M. Jerrum and F. Moller, A polynomial time algorithm for deciding bisimulation equivalence of normed basic parallel processes, *Mathematical Structures in Computer Science* **6** (1996) 251–259.
[14] Hirshfeld, Y. and F. Moller, A fast algorithm for deciding bisimilarity of normed context-free processes, in *Proc. CONCUR'94*, Lecture Notes in Computer Science, Vol. 836 (Springer, Berlin, 1994) 48–63.
[15] H. Hüttel, Undecidable equivalences for basic parallel processes, in: *Proc. Int. Symp. on Theoretical Aspects of Computer Software*, Lecture Notes in Computer Science, Vol. 789 (Springer, Berlin, 1994) 454–464.
[16] D. Huynh and L. Tian, Deciding bisimilarity of normed context-free process is in Σ_2^p, *Theoret. Comput. Sci.* **123** (1994) 183–197.
[17] D. Huynh and L. Tian, On deciding some equivalences for concurrent processes, *Theoret. Inform. and Appl.* **28** (1994) 51–71.
[18] P. Jančar, Undecidability of bisimilarity of Petri nets and some related problems, *Theoret. Comput. Sci.* **148** (1995) 281–301.
[19] P. Kanellakis and S. Smolka, On the analysis of cooperation and antagonism in networks of communicating processes, *Algorithmica* **3** (1988) 421–450.
[20] P. Kanellakis and S. Smolka, CCS expression, finite-state processes, and three problems of equivalence, *Inform. and Comput.* **86** (1990) 43–68.
[21] A. Sistla and S. German, Reasoning with many processes, in: *Proc. 2nd IEEE Symp. on Logic in Computer Science* (New York, 1987) 138–152.
[22] R. van Glabbeek, The linear time – branching time spectrum, in: *Proc. CONCUR'90*, Lecture Notes in Computer Science, Vol. 458 (Springer, Berlin, 1990) 278–297.
[23] H. Yen, Priority systems with many identical processes, *Acta Inform.* **28** (1991) 681–692.