On the Complexity of Relational Problems for Finite State Processes *

(Extended Abstract)

Sandeep K. Shukla *Harry B. Hunt III* *Daniel J. Rosenkrantz* *R. E. Stearns*
Department of Computer Science
University at Albany – State University of New York
Albany, NY 12222
Email: {sandeep,hunt,djr,res}@cs.albany.edu

Abstract

We study the complexity of the following two relational problems: Let $\sim$ be a binary relation on finite state processes; and let p_0 be a *fixed* finite state process.

P_1: Determine for processes p and q, if $p \sim q$.
P_2: Determine for process p, if $p \sim p_0$.

We study the complexities of these problems, when processes are represented by *sequential* transition systems and by *parallel composition* of transition systems.

First, for parallel compositions of transition systems, we show the following:

1. For all relations $\sim$ between *strong bisimulation* and *trace preorder*, the problem P_1 is PSPACE-hard, even *without* hiding. (This extends the results in [Rab92, Rab95]).
2. For all *fixed* processes p_0 and for all relations $\sim$ between *weak bisimulation* and *trace preorder*, the problem P_2 is PSPACE-hard *with hiding*.
3. Given any nontrivial predicate π on processes such that $\pi(x) = \pi(y)$ whenever x, y are weakly bisimilar, determining, for a process p *with* hiding, if $\pi(p) = true$ is PSPACE-hard.

Second, we outline a uniform approach for developing polynomial time algorithms for various simulation and equivalence relations for finite state processes represented by sequential transition systems. This approach involves efficient reductions to the satisfiability problem for Horn formulas. It applies directly to a number of simulation relations and equivalences considered in the literature. Here we use our approach to develop polynomial time algorithms for deciding the *forward* and *backward simulation* relations of [LV91, LV95]. We also use our approach to develop an NC algorithm for the bisimulation equivalence problem for deterministic sequential processes (mentioned as an open problem in [GHR95]).

A number of additional results are obtained both for sequential and parallel finite state systems.

*This research was supported by NSF Grants CCR-90-06396 and CCR-94-06611.

1 Introduction

1.1 Motivation

Recently, a number of formalisms have been proposed for the specification and verification of concurrent systems. These formalisms include such process algebraic formalisms as the Communicating Sequential Processes(CSP) [Hoa84], the Calculus of Communicating Systems(CCS) [Mil89], the Algebra of Communicating Processes(ACP) [BK84], etc. These formalisms also include such automata theoretic formalisms as the I/O automata of Lynch et al. [LV91, LV95]. One can view these formalisms as abstract programming languages for the specification of the data and control flow in concurrent systems. To each such abstract specification one associates an appropriate semantic object. Often these semantic objects are transition systems [Mil89, Hoa84], which may be related via appropriate equivalence relations (e.g., *strong and weak bisimulation, trace equivalence*, etc. [Mil89, Hoa84]). Transition systems are associated with expressions in the algebraic languages by a Plotkin style operational semantics [Plo81]. For I/O automata the association of transition systems to I/O automata is more direct.

We call a transition system represented by an explicit enumeration of its states and transition relation a **sequential** transition system. Transition systems represented as parallel compositions of sequential transition systems are called **parallel** transition systems. We consider parallel transition systems both *with* and *without* hiding.

Let $\sim$ be a binary relation on finite processes, and let p_0 be a fixed finite state process. Consider the following two relational problems.

P_1: Determine for processes p and q, if $p \sim q$.
P_2: Determine for a process p, if $p \sim p_0$.

We study these problems for various relations $\sim$ defined in the literature [Mil89, BHR84, HT94, Rab92, LV95] for the following reasons:
1. Many verification problems are modelled as problem P_1 for some appropriate relation $\sim$, where the specification of a system is represented as a process q and implementation as a process p [LV95, Kur94].
2. In fact, in many industrial automated verification tools, libraries of transition systems corresponding to desired properties or operations of systems have been implemented [Kur94]. Hence, to prove that some implementation P conforms to a given specification, one establishes relations with a fixed transition system from the library [Kur94, Wol95].

For modelling concurrent/parallel systems, *parallel composition* and *hiding abstraction* have been widely used [Hoa84, Mil89, BK84]. For parallel transition systems with or without hiding, the state space of the system can be exponentially larger than size of the system description. As a result, problems P_1, P_2 may be harder for processes with such succinct descriptions. Thus, we study the complexity of problems P_1, P_2 for such systems. We also, consider the complexity of nontrivial predicates on finite processes when processes are represented as parallel transition systems and present some results.

We also consider problem P_1 for sequential transition systems for some relations defined in the context of I/O automata in [LV91, LV95]. The complexity of these relations has not been studied in the literature previously. However, these are worth investigating because automated verification tools based on the I/O automata model needs to implement these decision procedures.

1.2 Results and Contributions

This paper presents two sets of results. Our first set of results are for processes represented by parallel transition system *with* or *without* hiding.

The results obtained are as follows:

1. [Rab92, Rab95] showed the problem P_1 is PSPACE-hard for all equivalences between bisimulation and trace equivalence when processes are represented by parallel composition **with** hiding. We show that for all these relations problem P_1 is PSPACE-hard even **without** hiding. Our proof is based on the following observation which also leads to easier proofs of most of the other results in [Rab92, Rab95]. We show that all relations between bisimulation and trace preorder (which includes all equivalences between

bisimulation and trace equivalence) are as hard as the **nonreachability problem** for finite state processes for the corresponding representations. The reachability problem for finite state processes is the problem of deciding if a given state is reachable in any execution of a given process specification. The complement of reachability is called the nonreachability problem.

Moreover, we show that these lower bound results hold even when all the individual transition systems in the parallel composition (without hiding) have the same two-symbol *external* action alphabet.

2. We also show that the *trace, failure* and *readiness* equivalences for systems represented in this way are PSPACE-Complete.

   Stockmeyer [Sto92] showed that with hiding abstraction and parallel composition, bisimulation is DEXPTIME-Complete. Since we show that trace equivalence without hiding is PSPACE-Complete, it is clear that the **uniform** lower bound for all relations between bisimulation and trace equivalence (for systems expressed with parallel composition without hiding) can not be strengthened to DEXPTIME unless PSPACE = DEXPTIME. Thus our uniform lower bound is the strongest one can obtain for this class of systems.

3. For problem P_2 we show that, for all relations $\sim$, between *weak* bisimulation and trace preorder, and for *any* fixed process p_0, deciding if $P \sim p_0$, for a process P represented by parallel composition *with* hiding, is PSPACE-hard. This result follows from an observation similar to the nonreachability observation. We also show that, deciding $P \sim p_0$, when $\sim$ is the trace preorder relation, is in PSPACE which shows that this lower bound is the best one can get uniformly. We also observe that there are fixed processes for which bisimulation is easy to decide, even for parallel composition with hiding. Hence, our lower bound as stated above can not be extended to bisimulation. In that sense, we obtained the best possible characterization which holds for *all* fixed processes.

4. For a process P represented by a parallel composition of acyclic sequential processes with hiding, we show that, for any fixed acyclic process p_0 and any relation $\sim$ between *weak* bisimulation and trace preorder, deciding if $P \sim p_0$, is Co-NP-hard. This proof yields a simplified Co-NP-hard lower bound proof of a stronger version of a result in [Rab92]. The result in [Rab92] proved that for all equivalences between bisimulation and trace equivalence, for parallel composition of acyclic processes without hiding, problem P_1 is Co-NP-hard. However, we significantly strengthen that result by showing that for trace equivalence, problem P_1 is indeed Co-NP-Complete, which proves that this lower bound is the best possible lower bound that holds for all these relations. We also show that the lower bound holds even for cases when the individual acyclic transition systems are deterministic with as few as six states.

Tables 1 and 2 in Section 5 in the Appendix summarizes the new complexity results obtained here for problem P_1 and P_2 respectively.

We also consider problem P_3 defined as follows:

> P_3**:** Let π be a nontrivial predicate on finite state processes such that $\pi(x) = \pi(y)$ whenever x and y are weakly bisimilar. Determine, for a process p, if $\pi(p)$.

We show that, for any nontrivial predicate on processes, problem P_3 for processes represented by parallel composition *with* hiding is PSPACE-hard.

In the second set of results we consider *forward* and *backward* simulation relations, *refinement* mapping, *history* and *prophecy* relations as defined in [LV91, LV95]. We consider problem P_1 for these relations for sequential systems. The complexity of these problems has not been considered in the literature previously. We present a uniform technique to obtain polynomial time algorithms for various simulation and equivalence relations for these systems. This technique is based on efficient reducibility of these relations to HORNSAT. This is a quite general technique which applies to a number of relations. Here we apply this technique to develop polynomial time algorithms for *forward simulation* and *backward simulation* relations. We also apply this technique to develop an NC algorithm for deciding the bisimulation equivalence between two deterministic sequential transition systems.

[1] Finally, we show that finding existence of *refinement mapping* is NP-Complete, and finding existence of *history relation* and finding existence of *prophecy relation* are graph-isomorphism-hard.

Table 3 in Section 5 of the Appendix shows a summary of our complexity results for sequential transition systems.

1.3 Organization of the Paper

Section 2 consists of some relevant definitions and background results. Section 3 presents our hardness results related to problems P_1, P_2, and P_3. Section 4 contains our results related to sequential transition systems. Selected proof sketches appear in the Appendix.

2 Transition Systems, Simulations and Equivalences

This section presents relevant definitions and results required for the exposition of our results.

Definition 2.1 *1. Act is a set of actions containing a special action τ called the* **internal action** *or* **unobservable action**.
2. A **transition system** $\mathcal{T}$ *over Act is a triple $\langle S, D, s_1 \rangle$ where S is the set of states, $D \subseteq S \times Act \times S$ is the set of transitions and $s_1 \in S$ is the starting state.*
3. $\mathcal{T}$ is **finite** *if both S and Act are finite.*
4. $ext(\mathcal{T}) = Act - \{\tau\}$ is the set of **external** *or* **visible** *actions.*
5. If σ is a sequence over Act, then $\hat{\sigma}$ is the sequence over $ext(\mathcal{T})$ obtained by deleting all the τ actions from σ.
6. If (p_1, a, p_2) is in D then we write $p_1 \xrightarrow{a} p_2$. Also if σ is a sequence of actions such that there is a transition from state p_1 to a state p_2 through some intermediate steps such that the sequence of actions is σ, then we write $p_1 \overset{\sigma}{\Longrightarrow} p_2$ and call this an **extended step**.
7. Given $\langle \mathcal{T} = S, D, s_1 \rangle$, let $\overline{D} = \{(p, a, p') \mid p \in S \wedge a \in Act \wedge p' \in S \wedge \exists \sigma \in \tau^* a \tau^*, p \overset{\sigma}{\Longrightarrow} p'\}$. We call $\overline{D}$ the **extended transition** *relation of $\mathcal{T}$.*

Definition 2.2 *Let $\mathcal{T} = \langle S, D, s_1 \rangle$ be a transition system and let $p \in S$. The* **initial set** *of p is defined as $init(p) = \{a \in Act \mid \exists t \in S((p, a, t) \in D)\}$.*

Let $\mathcal{T}_1 = \langle S, D_1, s_1 \rangle$ and $\mathcal{T}_2 = \langle T, D_2, t_1 \rangle$ be two transition systems.

Definition 2.3 *Let $R \subseteq S \times T$ be a binary relation between S and T. R is a* **simulation** *if*

$$\begin{aligned} &\forall (s,t) \in R \\ &\quad (\forall a \in Act, \forall s' \in S \\ &\qquad ((s, a, s') \in D_1 \Rightarrow (\exists t' \in T\ ((t, a, t') \in D_2 \wedge (s', t') \in R)))). \end{aligned}$$

R is a **bisimulation** *if R and R^{-1} are both simulations.*
R is a **ready simulation** *if R is a simulation and for each $(s,t) \in R$, $init(s) = init(t)$.*
R is a **complete simulation** *if R is a simulation and for all $(s,t) \in R$, $init(s) = \phi \Leftrightarrow init(t) = \phi$.*

Definition 2.4 *Let $\mathcal{T}_1 = \langle S, D_1, s_1 \rangle$ and $\mathcal{T}_2 = \langle T, D_2, t_1 \rangle$ be two transition systems. We define $\mathcal{T}_1$ to be* **bisimulation equivalent** *to $\mathcal{T}_2$, denoted by $\mathcal{T}_1 \sim_{bsim} \mathcal{T}_2$, iff there is a bisimulation R such that $(s_1, t_1) \in R$. $\mathcal{T}_1$ is said to be* **simulated** *by $\mathcal{T}_2$, denoted by $\mathcal{T}_1 \preceq_{sim} \mathcal{T}_2$, iff there is a simulation R such that $(s_1, t_1) \in R$.*

Definition 2.5 *Let $\mathcal{T}_1 = \langle S, D_1, s_1 \rangle$ and $\mathcal{T}_2 = \langle T, D_2, t_1 \rangle$ be two transition systems. $B \subseteq S \times T$ is an* **weak bisimulation** *relation from $\mathcal{T}_1$ to $\mathcal{T}_2$ if the following conditions are satisfied.*

1. $(s_1, t_1) \in B$
2. $\forall (r, s) \in B, a \in Act$:
if $\exists \gamma \in \tau^* a \tau^* : r \overset{\gamma}{\Rightarrow} r'$ then $\exists s' \exists \beta \in \tau^* a \tau^* : s \overset{\beta}{\Rightarrow} s' \wedge (r', s') \in B$.
and if $\exists \beta \in \tau^* a \tau^* : s \overset{\gamma}{\Rightarrow} s'$ then $\exists r' \exists \gamma \in \tau^* a \tau^* : r \overset{\beta}{\Rightarrow} r' \wedge (r', s') \in B$.

[1] This is stated as an open problem in [GHR95]. However, it must be mentioned that in [HT94], the problem was shown to be in NL and thus implying that it is in NC. However, our technique yields a *direct* NC algorithm for the problem.

If there exists a weak bisimulation from $\mathcal{T}_1$ to $\mathcal{T}_2$, then we say that they are weak bisimulation equivalent, denoted by $\mathcal{T}_1 \sim_{wbsim} \mathcal{T}_2$.

Definition 2.6 *Let $\mathcal{T}_1 = \langle S, D_1, s_1 \rangle$ and $\mathcal{T}_2 = \langle T, D_2, t_1 \rangle$ be two transition systems. We say γ is a finite* **trace** *of a transition system $\mathcal{T} = \langle S, D, s \rangle$ if there is a finite sequence $\sigma \in Act^*$ for which there is a state $q \in S$ such that $s \stackrel{\sigma}{\Longrightarrow} q$ and $\gamma = \hat{\sigma}$. Let* **traces**$(\mathcal{T})$ *denote the set of all finite traces of a transition system $\mathcal{T}$. We define* **trace preorder** [2] *and* **trace equivalence** *as follows. If traces($\mathcal{T}_1$) $\subseteq$ traces($\mathcal{T}_2$) then we say that $(\mathcal{T}_1, \mathcal{T}_2)$ is in the* **trace preorder** *and denote this by $(\mathcal{T}_1 \preceq_{trace} \mathcal{T}_2)$. If traces($\mathcal{T}_1$) = traces($\mathcal{T}_2$) then we say that $(\mathcal{T}_1, \mathcal{T}_2)$ are* **trace equivalent** *and denote this by $(\mathcal{T}_1 \sim_{trace} \mathcal{T}_2)$.*

Next, we define **forward simulation** and **backward simulation** following [LV91]. We have adapted these definitions for transition systems.

Definition 2.7 *Let $\mathcal{T}_1 = \langle S, D_1, s_1 \rangle$ and $\mathcal{T}_2 = \langle T, D_2, t_1 \rangle$ be two finite transition systems.*

Let F be a binary relation between S and T. F is a **forward simulation** *(or* **possibilities mapping***) from $\mathcal{T}_1$ to $\mathcal{T}_2$ if the following conditions are satisfied.*

1. $(s_1, t_1) \in F$.
2. $\forall (s, t) \in F$
   $(\forall a \in Act, \forall s^{'} \in S,$
   $((s, a, s^{'}) \in D_1 \Rightarrow (\exists t^{'} \in T(\exists \sigma \in \tau^* a \tau^* (t \stackrel{\sigma}{\Rightarrow} t^{'} \wedge (s^{'}, t^{'}) \in F))))).$

We write $\mathcal{T}_1 \preceq_F \mathcal{T}_2$ if there is a forward simulation from $\mathcal{T}_1$ and $\mathcal{T}_2$. $\mathcal{T}_1$ and $\mathcal{T}_2$ are **forward-simulation equivalent** *denoted by $\mathcal{T}_1 \sim_F \mathcal{T}_2$ iff both $\mathcal{T}_1 \preceq_F \mathcal{T}_2$ and $\mathcal{T}_2 \preceq_F \mathcal{T}_1$*

Definition 2.8 *Let $\mathcal{T}_1 = \langle S, D_1, s_1 \rangle$ and $\mathcal{T}_2 = \langle T, D_2, t_1 \rangle$ be two finite transition systems.*

Let B be a **total**[3] *binary relation between S and T. B is a* **backward simulation** *from $\mathcal{T}_1$ to $\mathcal{T}_2$ if the following conditions are satisfied.*

1. $(s_1, t_1) \in B$ *and there is no other $t \in T$ such that $(s_1, t) \in B$.*
2. $\forall (s^{'}, t^{'}) \in B$
   $(\forall a \in Act, \forall s \in S,$
   $((s, a, s^{'}) \in D_1 \Rightarrow (\exists t \in T(\exists \sigma \in \tau^* a \tau^* (t \stackrel{\sigma}{\Rightarrow} t^{'} \wedge (s, t) \in B))))).$

We write $\mathcal{T}_1 \preceq_B \mathcal{T}_2$ if there is a backward simulation from $\mathcal{T}_1$ and $\mathcal{T}_2$. $\mathcal{T}_1$ and $\mathcal{T}_2$ are **backward simulation equivalent** *denoted by $\mathcal{T}_1 \sim_B \mathcal{T}_2$ iff both $\mathcal{T}_1 \preceq_B \mathcal{T}_2$ and $\mathcal{T}_2 \preceq_B \mathcal{T}_1$*

We now define **refinement** mapping, **history relation** and **prophecy relation**, adapting the definitions given in [LV95] for transition systems.

Definition 2.9 *Let $\mathcal{T}_1 = \langle S, D_1, s_1 \rangle$ and $\mathcal{T}_2 = \langle T, D_2, t_1 \rangle$ be two finite transition systems. A* **refinement** *from $\mathcal{T}_1$ to $\mathcal{T}_2$ is a function $r : S \rightarrow T$ such that the following conditions hold.*

1. $r(s_1) = t_1$.
2. *If $s \stackrel{a}{\rightarrow} s'$ in $\mathcal{T}_1$ then $\exists \sigma \in \tau^* a \tau^*$ such that $r(s) \stackrel{\sigma}{\Rightarrow} r(s')$ in $\mathcal{T}_2$.*

We write $\mathcal{T}_1 \preceq_R \mathcal{T}_2$ if there exists a refinement from $\mathcal{T}_1$ to $\mathcal{T}_2$.

A relation h over S and T is a **history** *relation from $\mathcal{T}_1$ to $\mathcal{T}_2$ if h is a forward simulation from $\mathcal{T}_1$ to $\mathcal{T}_2$ and h^{-1} is a refinement from $\mathcal{T}_2$ to $\mathcal{T}_1$. We write $\mathcal{T}_1 \preceq_H \mathcal{T}_2$ if there exists a history relation from $\mathcal{T}_1$ to $\mathcal{T}_2$.*

A relation p over S and T is a **prophecy** *relation from $\mathcal{T}_1$ to $\mathcal{T}_2$ if p is a backward simulation from $\mathcal{T}_1$ to $\mathcal{T}_2$ and p^{-1} is a refinement from $\mathcal{T}_2$ to $\mathcal{T}_1$. We write $\mathcal{T}_1 \preceq_P \mathcal{T}_2$ if there exists a prophecy relation from $\mathcal{T}_1$ to $\mathcal{T}_2$.*

[2] A binary relation is called a **preorder** if it is reflexive and transitive.

[3] A binary relation $R \subseteq S \times T$ is **total** if for all $s \in S$ there exists $t \in T$ such that $(s, t) \in R$.

Now we define $NHORNSAT$, to which we reduce some of our problems.

Definition 2.10 *A Boolean clause is a* **Horn Clause** *if it has at most one positive literal. The satisfiability problem for conjunctions of Horn clauses is called* **HORNSAT**.

If instead of at most one positive literal, each clause has at most one negative literal then the satisfiability of the conjunction of such weakly negative [Sch78] clauses will be called **NHORNSAT**.

It is well known that HORNSAT is in P. A linear time algorithm for $HORNSAT$ appears in [DG84]. It easily follows that,

Lemma 2.11 *NHORNSAT is solvable in linear time.*

Before we define parallel composition, we note that in the context of parallel composition a transition system is represented as a 4-tuple, rather than 3-tuple as in Definition 2.1. Here, a transition system $\langle S, D, s\rangle$ over an action alphabet Act is represented as $\langle S, s, A, \rightarrow\rangle$, where $A = Act - \{\tau\}$ and $\rightarrow = D$. [4] Although the composition we define here is in the style of CSP [Hoa84], the complexity bounds obtained in this paper also hold for variants of this style of parallel composition. For example, composition of I/O automata, composition in CCS [Mil89] etc.

The **parallel composition** of two transition systems T_1 and T_2 denoted by $T_1||T_2$ is defined as follows. Here we are defining synchronous parallel composition following Hoare [Hoa84].

Definition 2.12 *Let* $T_1 = \langle Q^1, q_0^1, A^1, \rightarrow_1 \rangle$ *and* $T_2 = \langle Q^2, q_0^2, A^2, \rightarrow_2 \rangle$. *Let* $T = T_1||T_2 = \langle Q, q_0, A, \rightarrow \rangle$. *Then* $Q = Q_1 \times Q_2$, $q_0 = (q_0^1, q_0^2)$, $A = A^1 \cup A^2$. *The transition relation* $\rightarrow$ *for* T *is given by the following inference rules.*

1. *If* $a \notin A^1 \cap A^2$ *then*
   $\frac{q_1 \xrightarrow{a}_1 q_2}{(q_1, q) \xrightarrow{a} (q_2, q)}$ *and* $\frac{q_1 \xrightarrow{a}_2 q_2}{(q, q_1) \xrightarrow{a} (q, q_2)}$.
2. τ*-transitions.*
   $\frac{q_1 \xrightarrow{\tau}_1 q_2}{(q_1, q) \xrightarrow{\tau} (q_2, q)}$ *and* $\frac{q_1 \xrightarrow{\tau}_2 q_2}{(q, q_1) \xrightarrow{\tau} (q, q_2)}$.
3. *if* a *is in* $A^1 \cap A^2$ *then*
   $\frac{q_1 \xrightarrow{a}_1 q_1' \quad q_2 \xrightarrow{a}_1 q_2'}{(q_1, q_2) \xrightarrow{a} (q_1', q_2')}$

Now we define the **Hiding** operation on transition systems.

Definition 2.13 *Let* $T_1 = \langle Q^1, q_0^1, A^1, \rightarrow_1 \rangle$ *be a transition system. Then* $T = hide\, a\, in T_1$ *is the transition system* $\langle Q, q_0, A, \rightarrow \rangle$ *where* $Q = Q^1$, $A = A^1 - \{a\}$, $q_0 = q_0^1$, *and transition relation* $\rightarrow$ *of* T *is defined by the following inference rules.*

1. *If* $a' \neq a$ *then*
   $\frac{q_1 \xrightarrow{a'}_1 q_2}{(q_1) \xrightarrow{a'} (q_2)}$.
2. τ*-transitions*
   $\frac{q_1 \xrightarrow{\tau}_1 q_2}{(q_1) \xrightarrow{\tau} (q_2)}$ *and* $\frac{q_1 \xrightarrow{a}_1 q_2}{(q_1) \xrightarrow{\tau} (q_2)}$.

Let $A \subseteq Act$ *be a set of actions. Then* $hide\, A\, in T$ *means* $hide\, a_1\, in(hide\, a_2\, in(....in(hide\, a_n\, in T)....))$..

[4] This notational change is needed because in parallel composition, the action alphabet differs from one transition system to another. However, it is assumed that all transition systems may have τ transitions, unless they are deterministic.

3 Hardness Results For Problems P_1, P_2 and P_3

3.1 Reachability Problem and Uniform Lower Bounds:

We define the **reachability problem** as follows.

Definition 3.1 *Let $\mathcal{T} = \langle Q, q_0, A, \rightarrow \rangle$ be a transition system. Let $s \in Q$ be a state of $\mathcal{T}$. The* **reachability** *problem is to decide if there is a finite sequence $\sigma \in (A \cup \{\tau\})^*$, such that $q_0 \stackrel{\sigma}{\Longrightarrow} s$. We call the complement of this problem, the* **nonreachability** *problem.*

Theorem 3.3 shows that the **nonreachability** problem can be used to obtain a uniform lower bound for a number equivalences and preorders.

Definition 3.2 *Let T_1 and T_2 be two transition systems. Let ρ and θ be two binary relations defined on transition systems. We say $\rho \Rightarrow \theta$ (ρ implies θ) if and only if whenever $(T_1, T_2) \in \rho$, also $(T_1, T_2) \in \theta$. For any three binary relations ρ, θ, ϑ, we say that the relation ϑ is between ρ and θ, if $\rho \Rightarrow \vartheta \Rightarrow \theta$.*

Theorem 3.3 *Consider a particular representation (e.g., sequential or parallel transition systems)for finite transition systems. The* **nonreachability problem** *for transition systems with that representation is $n\,polylog\,n$ time reducible to the problem of deciding any relation $\sim$ such that $\sim_{bsim} \Rightarrow \sim \Rightarrow \sim_{trace}$ and the problem of deciding any relation $\preceq$ such that $\preceq_{sim} \Rightarrow \preceq \Rightarrow \preceq_{trace}$ for transition systems with the same representation.*

Proof: See Section 6 in the Appendix.

Now we show using Theorem 3.3 that for transition systems represented succinctly using parallel composition(without hiding), any relation $\sim$ between bisimulation equivalence and trace equivalence as well as any relation $\preceq$ between simulation preorder and trace preorder is PSPACE-hard. Moreover, we also show that trace equivalence, failure equivalence and readiness equivalence are PSPACE-Complete for parallel transition systems. This shows that the **uniform lower bound** obtained here is the strongest that holds *uniformly* for all relations in these ranges.

3.1.1 Reachability in Parallel Transition Systems

First we present a few relevant definitions and results to prove the other theorems in this section.

Definition 3.4 *The* $\tau-$**closure** *of a state s of a transition system $\langle Q, q_0, A, \rightarrow \rangle$, is the set of states that the system could reach from s, either by executing no transition, or by a sequence of τ transitions . In other words, $\tau-\mathbf{closure}(s) = \{s' \mid \exists \sigma \in \tau^* : s \stackrel{\sigma}{\Longrightarrow} s' \}$. Given a set of states S, $\tau-\mathbf{closure}(S) = \bigcup_{s \in S} \tau-\mathbf{closure}(s)$.*

Lemma 3.5 *Given a parallel transition system $T_1 || T_2 || ... T_n$ and a state of the parallel system $s = \langle s_1, s_2, .., s_n \rangle$, $\tau-\mathbf{closure}(s) = S_1 \times S_2 \times ... \times S_n$, where $S_i = \tau-\mathbf{closure}(s_i)$.*

Proof : Follows from definition of $\tau-$**closure** □

Lemma 3.6 *Given a parallel transition system $T_1 || T_2 || ... T_n$ and a set of states S of the parallel system described as the cartesian product of sets of states of the components (i.e., $S = S_1 \times S_2 \times ... \times S_n$), $\tau-\mathbf{closure}(S) = S_1' \times S_2' \times ... \times S_n'$, where $S_i' = \tau-\mathbf{closure}(S_i)$.*

Proof: Follows from Lemma 3.5 and Definition 3.4. □

Now we prove a theorem about parallel transition systems that is the basis for some of the PSPACE upper bounds at the end of this section. Note that this theorem applies only when **no** hiding abstraction is allowed. Intuitively, this theorem states the following. Let the set of states that a parallel transition system could be in via a given trace σ be $S = \{s \mid \exists \gamma : \langle q_0^1, q_0^2, ..., q_0^n \rangle \stackrel{\gamma}{\Rightarrow} s \wedge \hat{\gamma} = \sigma\}$. Then S is the cartesian product of sets S_i for $i = 1..n$. S_i is the set obtainable by keeping track of the set of states that T_i could be in via the projection of the trace σ on the action alphabet of T_i.

Definition 3.7 **Projection** *of a trace γ over an action alphabet A, denoted as $\gamma \uparrow A$, is the result of deleting all the symbols in γ which are not in A. (Note that $\gamma \uparrow A$ might be ϵ where ϵ is the empty string.*

Theorem 3.8 *Given a parallel transition system $T_1||T_2||...T_n$, and $\sigma \in (\cup_i A_i)^*$, $\{\langle s_1, s_2, ..., s_n\rangle \mid \exists\gamma : \langle q_0^1, q_0^2, ..., q_0^n\rangle \stackrel{\gamma}{\Rightarrow} \langle s_1, s_2, ..., s_n\rangle \wedge \hat{\gamma} = \sigma\}$ equals $S_1 \times S_2 \times ... \times S_n$ where $S_i = \{s \mid \exists\beta : q_0^i \stackrel{\beta}{\Rightarrow}_i s \wedge \hat{\beta} = \sigma \uparrow A_i\}$.*

Proof sketch: By induction on the length of σ and application of Lemmas 3.5 and 3.6. □

Theorem 3.8 allows us to obtain a PSPACE algorithm for the reachability problem as described below.

Let $T_1||T_2||....||T_n$ be a parallel transition system. Let $(s_1, s_2, ..., s_n)$ be the state to be reached as specified in the input of the reachability problem. We can guess a string $\sigma \in (A^1 \cup A^2 \cup .... \cup A^n)^*$, one symbol at a time, and guess corresponding transitions to confirm that $(q_0^1, q_0^2, ..., q_0^n) \stackrel{\sigma}{\Rightarrow} (s_1, s_2, ..., s_n)$. We apply the proof of Theorem 3.8 to show that this can be done in PSPACE. To show hardness, we reduce the LBA acceptance problem to the Reachability problem in parallel Transition systems.

Recall the following theorem from [GJ79]:

Proposition 3.9 *There exists a fixed deterministic linear bounded automata(LBA) for which the word acceptance problem is PSPACE hard.*

By the above Proposition 3.9 and a construction detailed in Section 6 of the Appendix we obtain the following theorem.

Theorem 3.10 *For a transition system specified as a parallel composition of deterministic sequential transition systems, the reachability problem is PSPACE-hard. Moreover, this is true even when all the individual transition systems in the composition have the same two-symbol external action alphabet.*

Now refering to Theorem 3.3, we can see that for deterministic parallel transition systems all relations between bisimulation and trace equivalences are PSPACE-complete. The fact that they are PSPACE-hard follows from Theorem 3.10 and Theorem 3.3. For deterministic systems all these equivalences coincide [vG90] and can be easily decided using a nondeterministic PSPACE-Algorithm because trace inequivalence is decidable in PSPACE by guessing a trace one symbol at a time. So, by Savitch's theorem, that implies that trace inequivalence is in PSPACE. On the other hand, PSPACE is closed under complementation. Hence, for deterministic transition systems, any relation between bisimulation equivalence and trace equivalence is PSPACE-complete.

For parallel composition of nondeterministic sequential transition systems without hiding, we obtain the following results.

Theorem 3.11 *Given two finite state systems represented as parallel transition systems without hiding, deciding any relation between bisimulation and trace equivalence and any relation between simulation preorder and trace preorder is PSPACE-hard. Moreover, this uniform lower bound holds even when all the individual transition systems in the composition have the same two-symbol external alphabet.*

Proof: Follows from Theorems 3.3 and 3.10. □
The following theorem states that the trace equivalence, Failure equivalence and Readiness equivalence are PSPACE-Complete for this class of systems. A proof sketch of this theorem for the trace equivalence case appears in Section 6 in the Appendix.

Theorem 3.12 *For parallel transition systems without hiding, the trace equivalence, failure equivalence and readiness equivalence [5] problems are PSPACE-Complete.*

[5] failure and readiness equivalences are defined in Section 6 of the Appendix

3.2 Complexity of Deciding Relations to a Fixed Process

Let p_0 be any fixed process described as a sequential transition system or parallel transition system with or without hiding. We are interested in finding out the complexity of the problem P_2 as defined below. Given any binary relation $\sim$, and given a process P represented as a parallel composition of finite processes, *with* hiding, decide if $P \sim p_0$. We obtained the following result.

Theorem 3.13 *For any fixed process p_0, and for any binary relation on processes between* **weak** *bisimulation and trace preorder, problem P_2 is PSPACE-hard.*

Proof sketch: See Section 6 in the Appendix.
Now we show that the lower bound obtained above is tight in the following sense. There are relations between weak bisimulation and trace preorder for which this lower bound matches the upper bound. This shows that , we have obtained the best possible lower bound one could get *uniformly* for problem P_2 for any fixed process and for all the relations between weak bisimulation and trace preorder. On the other hand, Theorem 3.13 does not apply to bisimulation equivalence because there are fixed processes, (for example, a two state process that has a single action from the start state to end state,) for which problem P_2 for bisimulation equivalence can be easily solved in polynomial time. Hence, Theorem 3.13 cannot be strengthened in that direction.

Theorem 3.14 *For any fixed process p_0, for trace preorder, problem P_2 is PSPACE-complete.*

Proof sketch: See Section 6 in the Appendix.

3.2.1 Complexity of P_2 for composition of acyclic processes with hiding

First we present an much easier proof of a stronger version of the Co-NP-hard lower bound result in [Rab92] for the problem P_1, for all relations $\sim$ between bisimulation and trace equivalences, for parallel composition of acyclic sequential transition systems without hiding. We then use our proof to obtain the uniform lower bound result for problem P_2 for composition of this class of systems *with* hiding. We also significantly strengthen the result in [Rab92, Rab95] by showing that for trace equivalence, the lower bound obtained is tight. To obtain the lower bound result we first prove Theorem 3.16 via a polynomial time reduction from the following NP-Complete problem [Sch78].

Definition 3.15 Ex-1-ex-3 Monotone 3SAT:
Instance: $C = \bigwedge_{i=1}^{m} C_i$ *is a conjunction of m clauses each containing exactly* 3 **positive** *literals.*
Question: *Is there is a satisfying assignment to this instance such that each clause is satisfied by setting* **exactly one** *literal to* true.

Theorem 3.16 *The Reachability problem for parallel composition of acyclic transition systems is NP-Complete. The NP-hard lower bound applies even when each transition system in the composition is deterministic with only six states.*

Proof sketch: See Section 6 of the Appendix.

Corollary 3.16.1 *The non-reachability problem for parallel composition of acyclic transition systems is Co-NP-Complete.*

Hence by Theorem 3.3 we obtain the following corollary which is a stronger version of a theorem in [Rab92, Rab95].

Corollary 3.16.2 *All equivalences between bisimulation and trace equivalence and all preorders between the simulation and trace preorder for parallel composition of acyclic and deterministic transition systems are Co-NP-hard.*

Now notice that the length of the longest trace for these systems is polynomially bounded by the size of the system description. As a result, trace nonequivalence is in NP because one can guess the trace which witnesses nonequivalence and verify in polynomial time. Also we obtain the following theorem:

Theorem 3.17 *Trace equivalence for parallel composition for acyclic transition systems is Co-NP-Complete. For parallel composition of deterministic acyclic transition systems all equivalences between bisimulation and trace equivalence and all preorders between simulation and trace preorder are Co-NP-Complete.*

Now consider problem P_2 for parallel transition system with hiding, where each component is acyclic. Consider a fixed acyclic process p_0. We can use the reduction from the Ex-1-ex-3 Monotone 3SAT as in the proof of Theorem 3.16, in the same way we used the reduction used in theorem 3.10 in proving theorem 3.13. In this case, we modify the reduction as follows. If the transition system constructed from the instance of the Ex-1-ex-3-Monotone 3SAT reaches the state $\langle s_1, s_2, ..., s_m \rangle$, then it takes a & labelled transition to a state from which the system will mimick p_0. On the other hand if a process P_i, reaches s_{i5}, then it takes a $\#_i$ transition, and the transitions in all P_i is suitably modified so that when ever some P_i reaches s_{i5}, the whole system takes a $\#_i$ transition to a state from which the whole system mimicks the process p_0. Now hide $\{x_1, x_2, ..., x_n, \#_1, \#_2, ..., \#_m\}$ from this newly constructed system. If the instance was Exactly one in three satisfiable, then the & transition makes sure that this new system is not trace equivalent to p_0. On the other hand, if it is not 1-3 satisfiable, then they are weak bisimilar.

Theorem 3.18 *For any fixed acyclic process p_0, and for any binary relation on processes between* **weak** *bisimulation and trace preorder, problem P_2 is Co-NP-hard for processes which are represented as parallel composition of acyclic processes with hiding abstraction.*

We can also show that this uniform lower bound is also tight for this set of relations by showing that the trace preorder is in fact Co-NP-Complete.

Theorem 3.19 *For any fixed process p_0, for trace preorder, problem P_2 is co-NP-complete for processes which are parallel composition of acyclic processes with hiding abstraction.*

We also observe, that this uniform lower bound can not extend to bisimulation equivalence because of the same counter example discussed above.

3.3 Complexity of Predicates that Respects Weak Bisimilarity

Suppose π is a *nontrivial* predicate on finite state processes which respects weak bisimulation relation. In other words, for any two finite processes x and y, if they are *weak bisimulation* equivalent then $\pi(x) = \pi(y)$. Here we prove the PSPACE-hardness of determining, for a process P, represented as a parallel transition system with hiding, if $\pi(P) = true$.

Theorem 3.20 *Given a nontrivial predicate π on finite processes that respects weak bisimulation, the problem of determining, for a process P represented by a parallel transition system* **with hiding**, *if $\pi(P) = true$ is PSPACE-hard.*

Proof sketch: See Section 6 in the Appendix.

4 Results on Sequential Transition Systems

4.1 An overview of our Uniform Reduction Technique

First we present a uniform technique for developing polynomial time algorithms for various equivalences and preorders for finite state processes. Our technique is based upon efficient reduction to satisfiability problem for Horn formulas. After a brief overview of our technique we outline the development of polynomial time algorithms for Forward simulation and Backward simulation using our technique. We also show how our reduction technique yields an NC algorithm for deciding bisimulation equivalence between two deterministic transition systems.

Given two transition systems T_1 and T_2 and a simulation relation R, our method entails a **top-down** construction of a CNF formula f as follows:
1. The variables in the formula f are $X_{p,q}$ where p and q are the states in the two transition systems.
2.The clauses in the formula f are of the following three types.

(a) A single positive literal $X_{p,q}$. If we want (p,q) in the simulation relation we construct this type of clause.
(b) A single negated literal $\overline{X_{p,q}}$. If such a clause is constructed then (p,q) cannot be in any simulation relation of the given type.
(c) Implication clauses of the form $X_{p,q} \Rightarrow \bigvee_{i,j} X_{i,j}$. If a clause of this form is constructed then it means that for (p,q) to be in the simulation relation one of the (i,j)'s must also be in the simulation relation.

We create the CNF formulas depending on the properties of the relation being considered. For some relations, we consider all pairs of states (e.g. backward simulation) and in other cases we consider only simultaneously reachable[6] pairs of states. The effectiveness of the reduction relies on the property that if we generate a clause of the form $\overline{X_{s,t}}$, then it is guaranteed that no relation satisfying the properties of that particular relation can contain the pair (s,t). So, the basic idea involves encoding the properties of a relation into a type of CNF formula whose satisfiability is polynomial time decidable, and proving that the pair of processes is in the relation if and only if the resulting CNF formula is satisfiable.
Hence the decision problem on the process domain is encoded as a Horn formula satisfiability problem which can be solved in polynomial time.
The resulting CNF formulas in our method are called weakly negative Horn formulas[7] [Sch78]. The satisfiability of such formulas is known to be polynomial time decidable [DG84].

4.1.1 An NC Algorithm for Bisimulation between Deterministic Transition Systems

We show that the reduction outlined in the previous subsection can be carried out in **NC**; and for *deterministic* transition systems it results in 2-CNF formula (a 2-SAT instance). It is well known that **2-SAT** has an NC-algorithm [GHR95]. Combining these two facts, we obtain that bisimulation equivalence for deterministic transition systems is in **NC**. Note that for deterministic transition systems for each state there is only one transition labelled by a particular action. Using this fact, the reduction to NHORNSAT can be carried out as follows. For each pair of states (p,q), where p is from T_1 and q is from T_2, and for each action symbol $a \in Act$, we use a processor denoted by P^a_{pq}. Thus the total number of processor used is $n \times m \times k$ where n and m is the number of states in the two transition systems and k is the size of the action alphabet. Processor P^a_{pq} generates the clause $X_{p,q} \rightarrow X_{s,t}$ if there is a transition (p,a,s) in T_1 and a transition (q,a,t) in T_2. If there is a transition in one but not in the other, then the clause generated is $\overline{X_{p,q}}$. If there is no transition on both on the action a then this processor does not generate any clause. Thus each process takes constant time to generate the appropriate clause. As mentioned above, these clauses consist of either a single literal or 2-literals. Hence, the reduction produces an instance of 2-SAT. Thus, we obtain the following:

Theorem 4.1 Bisimulation equivalence *of* deterministic *transition systems is in* **NC**.

4.2 Complexity of Forward Simulation

Given two finite transition systems $\mathcal{T}_1$ and $\mathcal{T}_2$, we have to decide if $\mathcal{T}_1 \preceq_F \mathcal{T}_2$. First we outline how our generic reduction can be used to do this in polynomial time. Due to lack of space and time we do not provide any of the correctness proofs but they are similar to the correctness proofs for backward simulation given in the Appendix.

First compute, using the definition of extended transition relation given in Definition 2.1, $\overline{D_2} = \{(p,a,p') \mid \exists\sigma \in \tau^* a \tau^* \wedge p \stackrel{\sigma}{\Longrightarrow} p' \wedge p, p' \in T \wedge a \in Act(\mathcal{T}_2)\}$. As proved in [KS90] this will take polynomial time because this computation is a *transitive closure* computation. Now we describe the macros used in the description of the algorithm in Figure 1. Let $STEP_1(a,p) = \{p' \mid (p,a,p') \in D_1\}$, $STEP_2(a,q) = \{q' \mid (q,a,q') \in \overline{D_2}\}$.
Let $CLAUSE(p,a,p',q)$ equals $\bigvee_{q' \in STEP_2(a,q)} X_{p',q'}$ if $STEP_2(a,q) \neq \phi$, and equals FALSE otherwise.
This function is used to create the conditional clauses. Whenever we are considering the pair (p,q), we want to represent the conditions for their inclusion in a Forward Simulation relation. Given a transition $(p,a,p') \in D_1$

[6] We call a pair of states (p,q) simultaneously reachable if there is a $\sigma \in Act^*$ such that both p and q are reachable from the respective start states via the action sequence σ.
[7] A weakly negative clause is a clause which contains at most one negative literal.

there must be a transition $(q,a,q') \in \overline{D_2}$ and (p',q') must be in the Forward Simulation relation. Function $CLAUSE$ computes clauses expressing this fact.
$VARS(p,a,p^{'},q) = \{X_{p^{'},q^{'}} \mid q^{'} \in STEP_2(a,q)\}$ if $STEP_2(a,q) \neq \phi$ else ϕ.
Function $VARS$ keeps track of the variable occurrences in a newly created conditional clause created by a call to $CLAUSE$.

COMMENT: *The sets V and W keep track of all the variables generated and processed respectively. The set C will contain the set of clauses of the NHORNSAT instance.*

1. $C := \{X_{s_1,t_1}\}$; $V := \{X_{s_1,t_1}\}$; $W := \phi$; COMMENT: *The pair (s_1,t_1) must be in the relation.*
2. **do** until V is empty.
   (a) Let $X_{p,q}$ be any element in V.
   (b) $V := V - \{X_{p,q}\}$; $W := W \cup \{X_{p,q}\}$;

   COMMENT: *If for the pair of states (p,q), there is no transition from p, then trivially, the pair may be in the relation.*

   (c) If for no $a \in Act$ and $p' \in S : (p,a,p^{'}) \in D_1$
   $C := C \cup \{X_{p,q}\}$;

   COMMENT: *Step (d) creates implication clauses to encode the conditions of Forward simulation relation.*

   (d) **for each** $a \in Act, p' \in S$ such that $(p,a,p^{'}) \in D_1$ **do**
   $C := C \cup \{\overline{X_{p,q}} \vee CLAUSE(p,a,p^{'},q)\}$;
   $V := V \cup (VARS(p,a,p^{'},q) - W)$;
   endfor

   end do
3. Output C.

Figure 1: Algorithm for reducing Forward Simulation instance to $NHORNSAT$ instance.

The following theorem proves the correctness of the reduction and polynomial time decidability of the forward simulation relation between finite transition systems as defined in Section 2.

Theorem 4.2 *There is an algorithm that takes two finite transition systems $\mathcal{T}_1$ and $\mathcal{T}_2$ as input and outputs an instance of $NHORNSAT$ such that*

1. *The instance of $NHORNSAT$ has a satisfying assignment if and only if $\mathcal{T}_1 \preceq_F \mathcal{T}_2$.*
2. *The algorithm runs in $O(\mid \mathcal{T}_1 \mid\mid \mathcal{T}_2 \mid)$ time and produces $NHORNSAT$ instance with at most $\mid S \mid\mid T \mid$ variables and at most $O(\mid \mathcal{T}_1 \mid * \mid \mathcal{T}_2 \mid)$ clauses.*

Since there is a linear time algorithm for solving HORNSAT [DG84], we obtain the following corollary :

Corollary 4.2.1 *Given two finite transition systems $\mathcal{T}_1$ and $\mathcal{T}_2$, we can decide if $\mathcal{T}_1 \preceq_F \mathcal{T}_2$ in polynomial time.*

4.3 Complexity of Backward Simulation

The reduction from the *backward simulation* instance to $NHORNSAT$ is different from the reduction in the previous subsection in a subtle way. These differences arise because of the following three facts.

1. *Backward simulation* is a **total** relation
2. The start state of $\mathcal{T}_1$ can be related only to the start state of $\mathcal{T}_2$.
3. Backward simulation between two states depends on whether the same actions can lead *to* them from (backward) similar states rather than on what actions are possible *from* them.

As a result, unlike the other problems , here we have to consider all possible pairs of states, and the computation of the conditional clauses is slightly different from the other problems. A detailed description of the reduction algorithm and relevant definitions appear in Section 7 in the Appendix.
Our algorithm constructs an instance h of $NHORNSAT$ such that h is satisfiable iff $\mathcal{T}_1 \preceq_B \mathcal{T}_2$. We start with the single literal clause X_{s_1,t_1}. We include a single negative literal clauses $\overline{X_{s_1,t_j}}$ for all $j \neq 1$ (to satisfy the first condition of backward simulation).This is done because we do not want any other state in T to be related to s_1 other than t_1. Since we want the relation to be total we add clauses of the form $\bigvee_{t \in T} X_{s,t}$ for each $s \in S$ such that $s \neq s_1$. If $(p^{'}, q^{'}) \in B$ then for each step of the form $(p, a, p^{'}) \in D_1$ we express the fact that there is an extended step in $\overline{D_2}$ of the form $(q, a, q^{'})$ and (p, q) is in the backward simulation relation which witnesses the backward simulation from $\mathcal{T}_1$ to $\mathcal{T}_2$. Thus if there is a satisfying assignment of h then in that satisfying assignment $X_{p,q} = 1$ iff $(p, q) \in B$ where B is a particular witness backward simulation from S to T.

The correctness of following theorem is proved in the Section 7 of the Appendix.

Theorem 4.3 *Given two transition systems $\mathcal{T}_1 = \langle S, D_1, s_1 \rangle$ and $\mathcal{T}_2 = \langle T, D_2, t_1 \rangle$, the instance of $NHORNSAT$ output by Algorithm* **BACKNHorn** *is satisfiable iff $\mathcal{T}_1 \preceq_B \mathcal{T}_2$. Further, algorithm* **BACKNHorn** *runs in $O(\mid \mathcal{T}_1 \mid * \mid \mathcal{T}_2 \mid)$ time and produces an instance of $NHORNSAT$ with at most $(\mid S \mid * \mid T \mid)$ variables and $O(\mid \mathcal{T}_1 \mid * \mid \mathcal{T}_2 \mid)$ clauses.*

4.4 Complexity of Refinement Mapping, History Relation and Prophecy Relation

In this section we consider the complexity of deciding the existence of refinement mapping, history relation and prophecy relation between two transition systems $\mathcal{T}_1$ and $\mathcal{T}_2$.

For each of these decision problems, membership in NP can be established easily. Before we give the lower bound proofs, we need to recall the definitions of the the **K-Clique** problem and the **Graph Isomorphism** problem. We have included these definitions in the Section 8 of the Appendix. In all the graph theoretical problems, we assume, with out loss of generality, that the graphs are connected and simple.

Given a graph $G = (V, E)$, in all the following proofs we construct a corresponding transition system $T_G = \langle S_G, \rightarrow_G, s_G \rangle$ over action alphabet $\{a, b\}$ as follows. $S_G = V \cup \{s_G\}$ where $s_G \notin V$. $\rightarrow_G = \{(s_G, a, v) \mid v \in V\} \bigcup \{(u, b, v), (v, b, u) \mid (u, v) \in E\}$.

Due to lack of space proofs of the following theorems are given in Section 8 of the Appendix. In essence, Theorem 4.4 is proved by reducing the **K-clique** problem to the refinement mapping problem as follows. We show that a graph G has a K size clique, if and only if $T_{\mathcal{K}_K} \preceq_R T_G$. Similarly, we prove Theorem 4.5 by showing that two graphs G, G' of equal size are isomorphic if and only if $T_G \preceq_H T_{G'}$. A similar reduction exists for for Theorem 4.6.

Theorem 4.4 *Given two transition systems $\mathcal{T}_1$ and $\mathcal{T}_2$. The problem of determining whether there is a refinement mapping from $\mathcal{T}_1$ to $\mathcal{T}_2$ is NP-Complete.*

Theorem 4.5 *Given two transition systems $\mathcal{T}_1$ and $\mathcal{T}_2$, the problem of determining whether $\mathcal{T}_1 \preceq_H \mathcal{T}_2$ is graph-isomorphism-hard.*

Theorem 4.6 *Given two transition systems $\mathcal{T}_1$ and $\mathcal{T}_2$, the problem of deciding whether $\mathcal{T}_1 \preceq_P \mathcal{T}_2$ is graph-isomorphism-hard.*

Acknowledgements: We thank Prof. S. S. Ravi for reading various versions of the draft, giving insightful comments and helping in the improvement of the presentation. We also thank Dr. Alexander Rabinovich and Dr. Larry Stockmeyer for making their drafts available. Special thanks to Prof. Frits Vaandrager and Prof. Pierre Wolper for helpful discussions.

References

[BHR84] S. D. Brookes, C. A. R. Hoare, and W. Roscoe. A theory of communicating sequential processes. *J. Association of Computing Machinery*, 31:560–599, 1984.

[BK84] J. A. Bergstra and J. W. Klop. Process algebra for synchronous communications. *Information and Control*, 60:109–137, 1984.

[DG84] W.F. Dowling and J.H. Gallier. Linear time algorithm for testing the satisfiability of propositional horn formulae. *Journal of Logic Programming*, 3:267–284, 1984.

[GHR95] R. Greenlaw, H. J. Hoover, and W. L. Ruzzo. *Limits to Parallel Computation: P-completeness Theory*. Oxford University Press, 1995.

[GJ79] M. Garey and D. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. Freeman, SanFrancisco, 1979.

[Hoa84] C. A. R. Hoare. *Communicating Sequential Processes*. Prentice Hall International, 1984.

[HT94] Dung T. Huynh and Lu Tian. On deciding some equivalences for concurrent processes. *Theoretical Informatics and Applications*, 28(1):51–71, 1994.

[KS90] Paris C Kanellakis and Scott A Smolka. CCS expressions, finite state processes and three problems of equivalence. *Information and Computation*, 86:43–68, 1990.

[Kur94] R. Kurshan. *Computer Aided Verification of Coordinating processes : An Automata Theoretic Approach*. Princeton University Press, 1994.

[LV91] Nancy Lynch and Frits Vaandrager. Forward and backward simulation:untimed systems. In *REX Workshop on Real Time systems*, 1991.

[LV95] Nancy Lynch and Frits Vaandrager. Forward and backward simulations-part i: Untimes systems. *Information and Computation*, 1995.

[Mil89] R. Milner. *Communication and Concurrency*. International Series in Computer Science. Prentice Hall, 1989. SU Fisher Research 511/24.

[Plo81] Gordon D. Plotkin. A structural approach to operational semantics. Technical Report DAIMI FN-19, Computer Science Department, Aarhus University, Aarhus University, Denmark, 1981.

[Rab92] Alexander Rabinovich. Checking equivalences between concurrent systems of finite state agents. In *ICALP, LNCS 623*, pages 696–707, 1992.

[Rab95] A. Rabinovich. Complexity of Equivalence Problems for Concurrent Systems of Finite Agents. (Draft), May 1995.

[Sch78] Thomas J. Schaefer. The complexity of satisfiability problems. In *Tenth Annual Symposium on Theory of Computing*, 1978.

[Sto92] L. J. Stockmeyer. Dexp-time hardness of bisimulation equivalence of concurrent system of finite state processes with hiding. (Unpublished Notes), 1992.

[vG90] R.J. van Glabbeek. The linear time - branching time spectrum. Technical Report CS-R9029, Computer Science Department, CWI, Centre for Mathematics and Computer Science, Netherlands, 1990.

[Wol95] P. Wolper. Private communications. 1995.

Appendix

5 Summary of Our Results in Tabular Form

<table>
<tr><th>Type of Systems</th><th></th><th>Decision Problem</th><th>lower bound</th><th>upper bound</th></tr>
<tr><td rowspan="5">Unrestricted</td><td rowspan="4">Nondeterministic</td><td>Reachability</td><td rowspan="5">PSPACE-hard</td><td>PSPACE</td></tr>
<tr><td>Equivalences between bisimulation and trace</td><td rowspan="2"></td></tr>
<tr><td>Preorders between simulation and trace</td></tr>
<tr><td>Trace, Failure, Readiness Equivalence</td><td>PSPACE</td></tr>
<tr><td>Deterministic</td><td>Equivalences between bisimulation and trace</td><td>PSPACE</td></tr>
<tr><td rowspan="5">Acyclic</td><td rowspan="4">Nondeterministic</td><td>Reachability</td><td>NP-hard</td><td>NP</td></tr>
<tr><td>Equivalences between bisimulation and trace</td><td rowspan="3">co-NP-hard</td><td rowspan="2"></td></tr>
<tr><td>Preorders between simulation and trace</td></tr>
<tr><td>Trace, Failure, Readiness Equivalence</td><td>co-NP</td></tr>
<tr><td>Deterministic</td><td>Equivalence between bisimulation and Trace</td><td>co-NP-hard</td><td>co-NP</td></tr>
</table>

Table 1: Table showing our complexity results for the problem P_1 for Parallel Composition without Hiding

<table>
<tr><th>Types of Systems</th><th>Decision problem</th><th>lower bound</th><th>upper bound</th></tr>
<tr><td rowspan="2">Unrestricted</td><td>Relations between weak bisimulation and trace preorder</td><td rowspan="2">NSPACE(n)-hard</td><td></td></tr>
<tr><td>Trace preorder</td><td>NSPACE(n)</td></tr>
<tr><td rowspan="2">Acyclic</td><td>Relations between weak bisimulation and trace preorder</td><td rowspan="2">Co-NP-hard</td><td></td></tr>
<tr><td>Trace preorder</td><td>Co-NP</td></tr>
</table>

Table 2: Table showing our results for problem P_2 for processes represented by Parallel Composition **with** hiding.

Relation	Lower bound	Upper bound
Forward Simulation		P
Backward Simulation		P
Refinement Mapping	NP-hard	NP
History Relation	Graph-isomorphism-hard	NP
Prophecy Relation	Graph-Isomorphism-hard	NP

Table 3: Table showing the complexity of relations for sequential systems

6 Proofs of Theorems in Section 3

Proof sketch of Theorem 3.3

Let $\sim$ be any equivalence between bisimulation equivalence and trace equivalence and let $\preceq$ be any preorder between $\preceq_{sim}$ and $\preceq_{trace}$. In other words $\sim_{bsim}\Rightarrow\sim\Rightarrow\sim_{trace}$. and $\preceq_{sim}\Rightarrow\preceq\Rightarrow\preceq_{trace}$. Now suppose we are given an instance of the Reachability problem. So we are given (T, s) where $T = \langle Q, q_0, A, \rightarrow \rangle$is a transition system and $s \in Q$. We create an instance of the $\sim$ decision problem as follows. Construct two new transition systems T_1 and T_2 such that $T_1 = \langle Q^1, q_0^1, A^1, \rightarrow_1\rangle$, $T_2 = \langle Q^2, q_0^2, A^2, \rightarrow_2\rangle$where $Q^1 = Q^2 = Q \cup \{d\}$, (d is not in Q) $q_0^1 = q_0^2 = q_0$, $A^1 = A \cup \{\$\}$ and $A^2 = A \cup \{\#\}$ such that \$ and # are not in $A \cup \{\tau\}$.
$\rightarrow_1=\rightarrow \cup\{(s, \$, d)\}$ and $\rightarrow_2=\rightarrow \cup\{(s, \#, d)\}$.

Now it is not difficult to verify that $T_1 \sim_{bsim} T_2$ if and only if s is not reachable in T. On the other hand if s is reachable in T, then $T_1 \not\sim_{trace} T_2$.

Now if we want to decide the reachability, then test $T_1 \sim T_2$ for any $\sim$ between $\sim_{bsim}$ and $\sim_{trace}$. Or any preorder $\preceq$ between $\preceq_{sim}$ and $\preceq_{bsim}$. Let $reachable(T, s)$ be $true$ if and only if the answer to this simulation problem instance is "no". Since $reachable(T, s) \Rightarrow T_1 \not\sim_{trace} T_2 \Rightarrow T_1 \not\sim T_2 \Rightarrow T_1 \not\sim_{bsim} T_2$ and $not(reachable(T, s) \Rightarrow T_1 \sim_{bsim} T_2 \Rightarrow T_1 \sim T_2$, it is clear that testing for $\sim$ will give a decision algorithm for the reachability problem. Hence, any lower bound on the nonreachability problem applies to the problem of deciding any equivalence $\sim$ such that $\sim_{bsim}\Rightarrow\sim\Rightarrow\sim_{trace}$. Similarly for any preorders between $\preceq_{sim}$ and $\preceq_{trace}$. □

Proof sketch for Theorem 3.10

So we will fix the deterministic LBA by Proposition 3.9 and for any given input x, we will construct a parallel transition system T and a particular state s of the constructed transition system such that the LBA accepts x if and only if s is reachable in T.

Let the LBA be $M = \langle Q, T, \sigma, q_0, q_y, q_n, \delta \rangle$where Q is the finite state set, T is the tape alphabet, q_0 is the initial state, q_y final accepting state, q_n is the rejecting state, $\delta : Q \times T \rightarrow Q \times T \times \{L, R\}$ is the transition function.

Let $x = x_1x_2.....x_n$ be an input to the LBA. Now we create the parallel transition system $\mathcal{T} = T_1||T_2||......||T_n$ as follows.

For each T_i the state set is given by $Q \times T \cup T \cup \{\$\}$. The initial state for T_1 is (q_0, x_1) and for all T_i $(n+1 > i > 1)$ the initial state is x_i.

The transition relations are given as follows: for all transitions of M of the form $\delta(q, x) = (q', x', L)$ for all $1 < i \leq n$, T_i has a transition $\langle (q, x), < i, q, x >^L, x') \rangle$and the correspondingly T_{i-1} has transitions (for all $b \in T$) $\langle (b, < i, q, x >^L, (q', b))\rangle$.

Similarly for all transitions of M of the form $\delta(q, x) = (q', x', R)$ for all $1 \leq i < n$, T_i has a transition $\langle (q, x), < i, q, x >^R, x') \rangle$ and the correspondingly T_{i+1} has transitions (for all $b \in T$) $\langle (b, < i, q, x >^R, (q', b))\rangle$. Also for each T_i, for all $b \in T$, $\{ \langle (q_y, b), \#, \$\rangle, \langle b, \#, \$ \rangle\}$ are in the transition relation.

So whenever, a T_i reaches the accepting state it can do the # action and synchronizes with all $T_j, j \neq i$, so that all of them pass into a state called \$. So if the machine M reaches an accepting state, the system $\mathcal{T}$ reaches $(\$, \$, ..., \$)$ state and vice versa.

It is easy to check that each state of the parallel transition system is actually a machine instantaneous description(ID) of M. So $\mathcal{T}$ starts with the initial ID and in each step it simulates the machine. Whenever T_i is ready to do an action, it can do that only if T_{i-1} or T_{i+1} are ready to do that action depending on with which of T_{i-1} or T_{i+1}, T_i is sharing that action. As a result whenever T_i is in a state of the form (q, x) only then it can do an action because depending on $\delta(q, x)$, it will be synchronizing with an action with T_{i-1} or with T_{i+1}. So eventually, the parallel transition system reaches the state $(\$, \$, ..., \$)$ iff the machine M accepts x. The proof that this theorem holds even when the individual processes have the same action alphabet of size as low as two depends on a complicated encoding of the above reduction using two symbols. □

Definitions of Failure and Readiness Equivalences

Definition 6.1 *The* failure set *of a state s in a transition system T, denoted by $Failures(s)$, is defined by* $Failures(s) = \{(x, Z) \in (Act - \{\tau\})^* \times 2^{(Act-\{\tau\})} \mid \exists q \in S : s \stackrel{x}{\Longrightarrow} q \text{ and } init(q) \cap Z = \phi\}$

Two transition systems T_1 and T_2 are Failure Equivalent if $Failures(s_1) = Failures(s_2)$ where s_1 and s_2 are the start states of T_1 and T_2.

Definition 6.2 *The* ready set *of a state s in a transition system T, denoted by $Readies(s)$, is defined by* $Readies(s) = \{(x, Z) \in (Act - \{\tau\})^* \times 2^{(Act-\{\tau\})} \mid \exists q \in S : s \stackrel{x}{\Longrightarrow} q \text{ and } init(q) = Z\}$

Two transition systems T_1 and T_2 are Readiness Equivalent if $Readies(s_1) = Readies(s_2)$ where s_1 and s_2 are the starting states of T_1 and T_2.

Proof sketch for Theorem 3.12

The PSPACE-hardness is shown already. To show Completeness, we show that the complementary problem i.e., **not** trace equivalent is in PSPACE. Then by the fact that PSPACE is closed under complementation and by Savitch's theorem, it follows that the trace equivalence problem for this class of systems is in PSPACE. The nondeterministic algorithm is as follows. In order that the two systems are not trace equivalent, one of them must have a finite trace σ, which the other does not have. Guess the transition system which has a trace σ. Let us call that transition system T_1 and the other T_2. Then guess one symbol of the trace at a time and then compute the set of states that T_2 can be in and take $\tau - closure$ of these states. The crucial point is that the set of states that the system could be in can be represented as a cartesian product of the set of states that each individual component could be in. This follows from Theorem 3.8. □

Proof sketch for Theorem 3.16

Proof: *NP-hardness:* We reduce the **Ex-1-ex-3 Monotone 3SAT** problem to the reachability problem for parallel composition of acyclic and deterministic transition systems to prove NP-hardness. Given an instance of the Ex-1-ex-3 Monotone 3SAT, let the clause C_i contain the positive occurrences of the variables x_{i1}, x_{i2}, x_{i3}. We construct the transition system $P_i = \langle S_i, \rightarrow_i, A_i, s_{i1}\rangle$ as shown in Figure 2 corresponding to the clause C_i where $S_i = \{s_{i1}, s_{i2}, s_{i3}, s_{i4}, s_{i5}, s_i\}$, $\rightarrow_i$ is the transition relation shown in the Figure 2, $A_i = \{x_{i1}, x_{i2}, x_{i3}, \$\}$ and s_{i1} is the starting state. Now consider the parallel composition $P_1||P_2||...P_m$. Envision taking a transition labelled with a variable as corresponding to setting the truth value of that variable to 1. Being monotone, that sets the truth value of each clause containing that variable to 1 also. Also note that if a variable is common to a number of clauses, the parallel transition systems cannot take a transition labelled by that variable unless all the transition systems corresponding to those clauses which have that variable in them take that transition.

It is not difficult to show that in $P_1||P_2||...P_m$, $\langle s_1, s_2, ..., s_m\rangle$ is reachable if and only if the given instance of Ex-1-ex-3 Monotone 3SAT has a satisfying assignment with exactly one literal in each clause being set to 1.

Membership in NP: Note that given a $P_1||P_2||...P_m$ where the number of states in P_i is n_i, and each individual P_i is acyclic, the maximum length of a sequence of transitions is bounded by $\Sigma_{i=1}^{m} n_i$ which is a polynomial bound. Hence, one can guess the sequence of transitions which may lead to a required state from the start state and verify that in polynomial time by keeping track of the set of states that the parallel system could be in at every

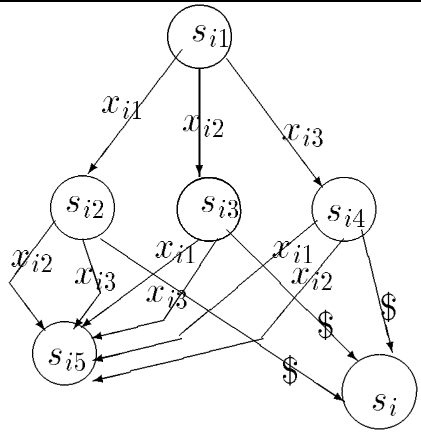

Figure 2: Transition system P_i corresponding to clause C_i in the proof of Theorem 3.16

transition. (As shown in the previous section these set of states can be compactly represented). □

Proof sketch for Theorem 3.13
Let M be a linear space bounded nondeterministic Turing machine. Without loss of generality, we can assume that M always halts and it halts either in q_r which is the rejecting state, or in q_a, which is the accepting state. Let p_0, be any fixed process. Given an input x to M, we construct a parallel transition system T_M, similar to the construction in Theorem 3.10. T_M mimicks the computation of M on x and its states correspond to the instantaneous description(ID) of M during its computation on x. We construct T_M in such a way such that its action alphabet A_M does not intersect the action alphabet of the given fixed process p_0. Add two new transitions, one labelled \$, and another labelled #, such that when T_M reaches an accepting state it makes the transition labelled \$ and starts mimicking p_0. If T_M reaches q_r, then it takes the # transition and starts mimicking p_0. (Technicalities for doing this is easy and consists of a parallel composition of a small modification of T_M and p_0.) Now hide $A_M \cup \{\#\}$ from this new system. It is easy to check that this new system is not trace equivalent to p_0 if M accepts x. On the other hand , if M rejects x, then this system is weakly bisimilar to p_0. Thus if $\sim$ is any relation between weak bisimulation and trace, then to decide if M does not accepts x we have to decide if this newly constructed system is $\sim$ related. This shows $\sim$ is NSPACE(n)-hard. Notice that our reduction is a linear space bounded reduction also. □

Proof Sketch of Theorem 3.14
Given a parallel transition system P with hiding and any fixed transition system p_0, we want to decide if $P \preceq_{trace} p_0$. Since p_0 is fixed a priori, determinize it w.r.t trace equivalence (i.e., create a deterministic transition system which is trace equivalent to p_0.) Then guess σ which is not in $traces(p_0)$ but in $traces(P)$. One has to guess the occurrences of the hidden symbols also, but since one has to verify that σ is a trace of P, tracing a path on a string γ such that the projection of γ on the unhidden symbols is σ, is enough. On the other hand showing σ is cannot be traced in p_0 can be done in linear space because p_0 is determinized a priori. □

Proof sketch for Theorem 3.20
The proof is very similar to our proofs based on nonreachability arguments. Consider a deterministic LBA M which always halts either in q_r or q_a (rejecting state or accepting state) [8]. Suppose x is a finite process such that $\pi(x)$ is true and y is another process such that $\pi(y)$ is false. Now use a similar reduction as in the proof of Theorem 3.13. In this case make sure whenever the parallel transition system corresponding to computation of M on input z, (call it T_M) reaches the accepting state, a transition labelled \$ is taken to a state from where

[8] It is always possible to obtain such an LBA equivalent to an other deterministic LBA in polynomial time

process x is mimicked. Also whenever, the transition system for M reaches a rejecting state then the system takes a \$ labelled transition to a state from where it mimicks the computation of y. Now in this new system hide all action symbols of T_M and \$. It is easy to verify that this new system after hiding, is weakly bisimilar to x if M accepts and weakly bisimilar to y if it rejects. Hence, M accepts if and only if π on this new transition system is true. This deciding π on processes represented by parallel transition systems is DSPACE(n)-hard. $\square$

7 Algorithm and Correctness Proofs For the Backward Simulation

Given $\mathcal{T}_1 = (S, D_1, s_1)$, $\mathcal{T}_2 = (T, D_2, t_1)$ we want to find if there is a total relation $B \subseteq S \times T$ such that B is a backward simulation.
The first step is to compute $\overline{D_2} = \{(p, a, p') \mid \exists \sigma \in \tau^* a \tau^*, p \stackrel{\sigma}{\Longrightarrow} p', a \in Act(\mathcal{T}_2)\}$. This can be done in polynomial time by the same method as in [KS90].
Once we have this **extended transition relation** $\overline{D_2}$, we can apply algorithm **BACKNHorn** as shown in Figure 3 in Section 7 of the Appendix. **BACKNHorn** is a polynomial time algorithm that takes $\mathcal{T}_1$ and $\mathcal{T}_2$ as input and outputs an instance of $NHORNSAT$ where the number of variables is $\leq nm$ (where $\mid S \mid= n$ and $\mid T \mid= m$). The number of clauses in this $NHORNSAT$ instance will be $O(\mid D_1 \mid\mid D_2 \mid)$. We now define the functions used in the description of the reduction algorithm.
The functions $BACKCLAUSES$ and $BACKVARS$ are defined as follows:
Let $BACKEXTSTEP(a, q') = \{q \mid (q, a, q') \in \overline{D_2}\}$, $BACKSTEP(a, p') = \{p \mid (p, a, p') \in D_1\}$.
Let $BACKCLAUSES(p', a, p, q') = \bigvee_{q \in BACKEXTSTEP(a,q')} X_{p,q}$
if $BACKEXTSTEP(a, q') \neq \phi$ else false
$BACKVARS(p', a, p, q') = \{X_{p,q} \mid q \in BACKEXTSTEP(a, q')\}$
if $BACKEXTSTEP(a, q') \neq \phi$ else ϕ.

Correctness Proof:

Proof of Theorem 4.3 :
Proof of the Theorem 4.3 follows directly from the following lemmas.

Lemma 7.1 *If $\overline{X_{p,q}}$ is a clause in h then there is no backward simulation $B \subseteq S \times T$ such that $(p, q) \in B$.*

Proof: There are two ways in which the algorithm BACKNHorn produces clauses of the form $\overline{X_{p,q}}$. Step 1 produces clauses of the form $\overline{X_{s_1,t_j}}$ for all $t_j \neq t_1$. A pair (s_1, t_j) cannot be in any backward simulation for any $t_j \neq t_1$ by the definition of backward simulation. The other clauses of this form are produced if $BACKCLAUSES(p', a, p, q')$ is false for some $a \in Act$ and some (p', q') pair. Thus, by the definition of $BACKCLAUSES$, the condition for (p', q') to be in B is not satisfied. $\square$.

Lemma 7.2 *If $X_{p,q}$ is a clause of h and $p \neq s_1$, then there is no transition leading to p in D_1.*

Proof: A single literal clause of the form $X_{p,q}$ with $p \neq s_1$ can be produced only in step 3c of Algorithm BACKNHorn. The conclusion of the lemma is *exactly* the condition for executing step 3c. $\square$.

Lemma 7.3 *If there is a single literal clause of the form $X_{p,q}$ and if $\mathcal{T}_1 \preceq_B \mathcal{T}_2$, then there exists a backward simulation relation $R \subseteq S \times T$ containing (p, q).*

Proof: Since $\mathcal{T}_1 \preceq_B \mathcal{T}_2$ there is a witness backward simulation relation $R \subseteq S \times T$. If R does not contain (p, q) then by lemma 7.2, there are no transitions into p. (Refer to step 3(c) of the algorithm.) Therefore the conditions of backward simulation will be trivially satisfied for (p, q). Hence, $R \cup \{(p, q)\}$ will also be a backward simulation. $\square$

COMMENT: *The sets V and W keep track of all the variables generated and processed respectively. The set C will contain the set of clauses of the NHORNSAT instance.*

1. $C := \{X_{s_1,t_1}\}$; $V := \{X_{s_1,t_1}\}$; $W := \phi$; COMMENT: *The pair (s_1, t_1) must be in the relation.*

   COMMENT: *Also, (s_1, t_j) should not be in the relation for $t_j \neq t_1$.*

   for all $t_j \neq t_1$ **do**
   $C \cup \{\overline{X_{s_1,t_j}}\}$; $V := V \cup \{X_{s_1,t_j}\}$
   endfor

   COMMENT: *The relation must be* total.

2. **for each** $s \in S - \{s_1\}$ **do**
   $C := C \cup \{\vee_{t \in T} X_{s,t}\}$; $V := V \cup \{\cup_{t \in T} X_{s,t}\} - W$;
   endfor

3. **do until** V is empty.
   (a) Let $X_{p',q'}$ be any element of V.
   (b) $V := V - \{X_{p',q'}\}$; $W := W \cup \{X_{p',q'}\}$;

   COMMENT: *If for the pair of states (p', q') if no actions leads to p', the pair may be in the relation.*

   (c) If there is no $t \in D_1$ such that $t = (p, a, p')$
   $C := C \cup \{X_{p',q'}\}$;

   COMMENT: *Step (d) creates implication clauses to encode the conditions of backward simulation relation.*

   (d) For each $a \in Act, p' \in S$
   such that $(p, a, p') \in D_1$
   $C := C \cup \{\overline{X_{p',q'}} \vee BACKCLAUSES(p', a, p, q')\}$
   $V := (V \cup BACKVARS(p', a, p, q')) - W$;

   end do

4. Output C.

Figure 3: Algorithm for reducing backward simulation instance to $NHORNSAT$ instance: Algorithm BACKN-Horn.

Lemma 7.4 *If $\mathcal{T}_1 \preceq_B \mathcal{T}_2$ then h is satisfiable.*

Proof: Suppose $\mathcal{T}_1 \preceq_B \mathcal{T}_2$. Then there is a witness backward simulation $R \subseteq S \times T$. By applying Lemma 7.3, we choose a witness R which contains (p, q) for every single positive literal clauses. We construct a satisfying assignment for h as follows: Let $X_{p,q} = 1$ iff $(p, q) \in R$. We claim that this is a satisfying assignment for h. First note that $(s_1, t_1) \in R$ by the definition of $\mathcal{T}_1 \preceq_B \mathcal{T}_2$. Thus the positive literal clause X_{s_1,t_1} will be satisfied. All clauses of the form $\overline{X_{p,q}}$ will be satisfied by Lemma 7.1. Any other single positive literal clause of the form $X_{p,q}$ will be satisfied because by Lemma 7.3 we could choose a witness R which contains (p, q) for every such single positive literal clauses. Now consider the implication clauses. If (s_i, t_j) is not in R then the implications where X_{s_i,t_j} appear on the right hand side are trivially satisfied. If $(s_i, t_j) \in R$, then $X_{s_i,t_j} = 1$ and hence one of the disjuncts on the right hand side should be satisfied in order that the whole implication is satisfied. However, by the definition of Backward simulation, it is easy to see that one of the disjuncts $X_{s_i',t_j'}$ must be satisfied because in order that $(s_i, t_j) \in R$, there must exist $(s_i', t_j') \in R$ satisfying the condition in the definition of backward simulation. Also notice that since any backward simulation is a total relation all clauses containing all positive literals generated at step 2 of the algorithm will be also satisfied. □.

Lemma 7.5 *If h is satisfiable then $\mathcal{T}_1 \preceq_B \mathcal{T}_2$.*

Proof: We construct a relation $R \subseteq S \times T$ from the satisfying assignment to h as follows: Include (p, q) in R iff in the satisfying assignment of h, $X_{p,q} = 1$. We claim that R is a backward simulation relation. Since X_{s_1,t_1} is a single positive literal clause in h, $X_{s_1,t_1} = 1$ in any satisfying assignment of h. Further, since $\overline{X_{s_1,t_j}}$ are single negated literal clauses in h for all $t_j \neq t_1$, (s_1, t_j) is not in R for any $t_j \neq t_1$. Also because of the clauses of the form $\bigvee_{t \in T} X_{s,t}$ generated at step 2, being satisfied, makes sure that R is a total relation. Thus relation R satisfies the first condition in the definition of backward simulation.

Now consider the implication clauses in h. They are all satisfied. Now if an implication clause is trivially satisfied because its left hand side literal $X_{p,q}$ is false (in any $NHORNSAT$ instance, left hand sides of all implications are positive literals), then the corresponding (p, q) is not in R. However, if $X_{p,q} = 1$, i.e., $(p, q) \in R$, then we must show that the second condition in the definition of backward simulation is satisfied. However, since $X_{p,q}$ is 1, one of the disjuncts in the right hand side must be satisfied and by the construction of these disjuncts, it follows that the second condition of the definition is satisfied. □

8 Definitions and Proofs of Section 4.4

The following definitions are from [GJ79]. The first problem is known to be NP-Complete [GJ79]. We reduce this problem to the deciding the existence of refinement mapping to establish the corresponding NP-hardness. The graph-isomorphism problem is not known to be in P nor is it known to be NP-hard [GJ79]. Here, we reduce the graph-isomorphism problem to the existence of history relation and prophecy relation, thus proving the graph-isomorphism-hardness of these decision problems.

Definition 8.1 K-Clique Problem*: Given a graph $G = (V, E)$, and an integer $4 \leq K \leq \mid V \mid$, does there exist a subset $V' \subseteq V$ such that $\mid V' \mid = K$ such that every pair of vertices in V' are joined by an edge in E?*

Definition 8.2 Graph Isomorphism Problem: *Given two graphs $G = (V, E)$ and $G' = (V', E')$, does there exist a bijection $f : V \rightarrow V'$ such that $(u, v) \in E$ if and only if $(f(u), f(v)) \in V'$?*

In the above definitions, it is assumed that the graphs specified are simple (i.e., no multi edges or self loops) and connected.

Proof sketch for Theorem 4.4

The problem is obviously in NP. We now show how to reduce the K-Clique problem to the refinement mapping problem. Let $\mathcal{K}_K$ denote the complete graph on K vertices (a clique on K nodes). and let G be the input graph to the K-Clique problem instance. We construct the transition system $T_{\mathcal{K}_K}$ and T_G as described above. We claim that there is a refinement mapping from $T_{\mathcal{K}_K}$ to T_G if and only if there is a K-clique in G.

Suppose G has a $K-clique$ consisting of nodes $v_{i1}, v_{i2}, ..., vik$. $v_{i1}, v_{i2}, ..., vik$. Then it can be verified that any function r that maps the start state of $T_{\mathcal{K}_K}$ to the start state of T_G and which sets a one-to-one correspondence between the other states of $T_{\mathcal{K}_K}$ to the states corresponding to vertices $v_{i1}, v_{i2}, ..., vik$ in T_G, is a refinement mapping from $T_{\mathcal{K}_K}$ to T_G.

Now recall that refinement mapping is a function from the states of $T_{\mathcal{K}_K}$ to the states of T_G. Suppose there is a refinement mapping r from $T_{\mathcal{K}_K}$ to T_G. First note that r has to be one-to-one because of the following reasons. First note that the start state of $T_{\mathcal{K}_K}$ must be mapped to start state of T_G. Secondly, if for two distinct states s and s', $r(s) = r(s')$, then there must be a self loop at $r(s)$ because by construction $s \stackrel{b}{\rightarrow}_{T_{\mathcal{K}_K}} s'$ and $s' \stackrel{b}{\rightarrow}_{T_{\mathcal{K}_K}} s$, But that is not possible because we assumed G does not have a self-loop. So each of the states of $T_{\mathcal{K}_K}$ are mapped to distinct states of T_G via r. Moreover, since r is a refinement mapping, for any pair of states (x, y) of

these K states in T_G, $x \xrightarrow{b} y$. Thus, the corresponding K vertices in G form a K-clique. □

The Proof sketch for Theorem 4.5

It is easy to show that the problem is in NP. We now show how to reduce the K-Clique problem to the refinement mapping problem. Let $\mathcal{K}_K$ denote the complete graph on K vertices (a clique on K nodes). and let G be the input graph to the K-Clique problem instance. We construct the transition system $T_{\mathcal{K}_K}$ and T_G as described above. We claim that there is a refinement mapping from $T_{\mathcal{K}_K}$ to T_G if and only if there is a K-clique in G.

Suppose G has a $K-clique$ consisting of nodes $v_{i1}, v_{i2}, ..., vik$. $v_{i1}, v_{i2}, ..., vik$. Then it can be verified that any function r that maps the start state of $T_{\mathcal{K}_K}$ to the start state of T_G and which sets a one-to-one correspondence between the other states of $T_{\mathcal{K}_K}$ to the states corresponding to vertices $v_{i1}, v_{i2}, ..., vik$ in T_G, is a refinement mapping from $T_{\mathcal{K}_K}$ to T_G.

Now recall that refinement mapping is a function from the states of $T_{\mathcal{K}_K}$ to the states of T_G. Suppose there is a refinement mapping r from $T_{\mathcal{K}_K}$ to T_G. First note that r has to be one-to-one because of the following reasons. First note that the start state of $T_{\mathcal{K}_K}$ must be mapped to start state of T_G. Secondly, if for two distinct states s and s', $r(s) = r(s')$, then there must be a self loop at $r(s)$ because by construction $s \rightarrow^{b}_{T_{\mathcal{K}_K}} s'$ and $s' \rightarrow^{b}_{T_{\mathcal{K}_K}} s$, But that is not possible because we assumed G does not have a self-loop. So each of the states of $T_{\mathcal{K}_K}$ are mapped to distinct states of T_G via r. Moreover, since r is a refinement mapping, for any pair of states (x, y) of these K states in T_G, $x \xrightarrow{b} y$. Thus, the corresponding K vertices in G form a K-clique. □