A Uniform Approach for Proving the Polynomial Time Decidability of Simulation Relations for Finite State Processes *

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

Abstract

We present a uniform approach for proving the polynomial time decidability of various simulation and equivalence relations for finite state processes. Our approach applies directly to a number of simulation relations and equivalences considered in the literature including

> *strong bisimulation* [Par81, Mil80, KS90], *simulation equivalence* [vG90, HT94], *ready-simulation equivalence* [vG90, HT94], *complete simulation equivalence* [vG90], *possibilities equivalence* (also known as *forward simulation*) and *backward simulation equivalence* [LV91].

The first four relations have been shown to be polynomial time decidable using different techniques [KS90, HT94]. The complexities of the last two equivalences have not been studied in the literature. Our approach uses efficient reductions to the satisfiability problem for Horn formulas.

As a corollary we show that for *deterministic* transition systems bisimulation equivalence is in **NC** which resolves an open problem posed in [GHR95].

Contact Author Information

Name: Sandeep K. Shukla

Address: Department of Computer Science, Room LI 67A
University at Albany - State University of New York
Albany, NY 12222

Email: sandeep@cs.albany.edu

Telephone: (518) 442-4285

Fax: (518) 442-5638

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

1 Introduction

1.1 Motivation and Results

Different simulation and equivalence relations between processes have been considered in various approaches to concurrency and automatic verification. These relations are used to establish the correctness of implementations of systems with respect to their specifications. For example, **bisimulation** equivalence is used in CCS [Mil80, Mil89] to identify processes which are equivalent under a particular semantic notion [Par81, Mil89]. Similarly, establishing that a given simulation relation holds has been used as a partial procedure for proving some safety properties [LV91]. Extensive discussion on these relations appears in [vG90]. Motivated by the importance of these relations in automated verification, several researchers have studied the complexity of decision procedures for these relations [KS90, HT94, ABGS91].

In this paper we present a uniform technique to *efficiently* reduce several such decision problems to the satisfiability problem for Horn formulas. Our ideas apply to a number of simulation and equivalence relations. Here, we use this generic methodology to prove the polynomial time decidability of the following equivalence relations on processes:

1. *Strong-Bisimulation Equivalence*[1][Par81, Mil89, KS90]
2. *Simulation Equivalence* [vG90, HT94]
3. *Ready-Simulation Equivalence* [vG90, HT94]
4. *Complete-Simulation Equivalence* [vG90]
5. *Possibilities Equivalence* or *Forward-Simulation Equivalence* [LV91]
6. *Backward-Simulation Equivalence* [LV91]

.

In our methodology we reduce these decision problems to the satisfiability problem for weakly negative Horn formulas and thereby show that all these equivalences are decidable in polynomial time. The first three of these equivalences have been shown to be polynomial time decidable in [KS90, HT94]. The fourth equivalence considered here can be easily shown to be polynomial time decidable from the results in [HT94]. The last two equivalences are based on two simulation relations in [LV91]. To the best of our knowledge, their decision complexities have not been considered in the literature. We show their polynomial time decidability for the first time.

As a corollary we show that the bisimulation equivalence and all other equivalences between bisimulation and trace equivalence for deterministic transition systems is in **NC**. Note that for determinsitic transition systems all equivalences between bisimulation and trace equivalence coincide. However in [GHR95], it has been posed as an open problem and our corollary solves that open problem.

1.2 Overview of the Uniform Reduction Technique

In [KS90], Kanellakis and Smolka proved that *bisimulation* equivalence of finite transition systems is polynomial time decidable by reducing the decision problem to the generalized partitioning problem.

[1] We use the word bisimulation for strong-bisimulation in this paper. We reduce the decidability of strong-bisimulation to the satisfiability problem for Horn formulas. But using the reduction of weak bisimulation [Mil89] to strong bisimulation given in [KS90], one can also prove the polynomial time decidability of weak bisimulation using our reduction.

In [HT94], Huynh and Tian proved that two other finite state process equivalences, namely *simulation equivalence* and *ready-simulation equivalence*, are also polynomial time decidable. Their proof involved showing that each of these simulation relations is the intersection of a chain of polynomially decidable relations. In both the cases the methods use **bottom-up** refinement techniques. For example, the partition refinement discussed in [KS90] starts with all possibilities and then computes (in a bottom up fashion) the maximal bisimulation or the bisimulation congruence.

In contrast, given a type of simulation relation, 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.

Depending on the properties of the relation under consideration, we create the CNF formulas. For some relations, we consider all pairs of states (e.g. backward simulation) and in other cases we consider only simultaneously reachable[2] 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

1. encoding the properties of a relation into a type of CNF formula whose satisfiability is polynomial time decidable, and
2. 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[3] [Sch78]. The satisfiability of such formulas is known to be polynomial time decidable [DG84].

1.3 Organization of the Paper

In the next section we give the definitions of transition systems and various simulation and equivalence relations considered in this paper. We also define the special form of **HORNSAT** used here (we call it **NHORNSAT**). In section 3 we illustrate our reduction method by describing in detail the reduction of **bisimulation equivalence** problem to **NHORNSAT**. Since there are some additional computations involved in the reduction for **backward simulation**, we present the details of the corresponding reduction in section 4. The other reductions are very similar to the bisimulation reduction and we

[2] 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.

[3] A weakly negative clause is a clause which contains at most one negative literal.

discuss them briefly in section 5. In the last section we show that as a corollary of our reduction we can show that all equivalences between bisimulation and trace equivalence for determinsitic transition system is in **NC**.

2 Transition Systems, Simulations and Equivalences

In this section we briefly discuss transition systems as semantic objects corresponding to CCS or CSP terms. The operational semantics of an abstract programming notation like CSP/CCS has two stages. The first is a labelled transition system, whose states correspond to the terms of the language and whose transitions are given in the structured operational semantics technique of Plotkin [Plo81]. Thus given a term P in the language, the state corresponding to this term is the start state of the transition system corresponding to P. The second stage is an equivalence relation on the transition systems. Two terms P and Q are defined to have the same operational semantics if their transition systems are in the same equivalence class. We now define these concepts more precisely.

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 $p_1 \stackrel{\sigma}{\Longrightarrow} p_2$ and call this an **extended step**.

7. Given $\langle \mathcal{T} = S, D, s_1 \rangle$, let $\overline{D} = \{(p, a, p') \mid p \in S \wedge a \in Act \wedge p' \in S \wedge \exists \sigma \in \tau^* a \tau^*, p \stackrel{\sigma}{\Longrightarrow} p'\}$. We call $\overline{D}$ the **extended transition** *relation of $\mathcal{T}$.*

A transition system may be represented graphically by an edge labelled directed graph with a special vertex denoting the starting state. Usually the vertex corresponding to the starting state is understood from the context or is indicated by an arrow adjacent to it. Transition systems are similar to classical NFAs with two important differences. First, that transition systems have a single starting state and second, there is no concept of a final state because these are not language recognizers.

This definition of transition systems corresponds to the untimed automata of Lynch and Vaandrager [LV91], except that the transition systems have a single starting state. Our results can also be extended to transition systems with multiple starting states. However, to simplify the presentation, we assume here that there is only one starting state.

We now define some terms that are used in later definitions. In the following definitions, we use the terminology of [HT94].

Definition 2.2 *Let $\mathcal{T} = \langle S, D, s_1 \rangle$ be a transition system and let $p \in S$.*

1. *The* **initial set** *of p is defined as $init(p) = \{a \in Act \mid \exists t \in S((p, a, t) \in D)\}$.*

2. *p is said to be a* **terminal state** *if $init(p) = \phi$.*

We now define the various relations we considered in this paper. 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*
$\forall (s,t) \in R$
$(\forall a \in Act, \forall s' \in S$
$((s,a,s') \in D_1 \Rightarrow (\exists t' \in T\ ((t,a,t') \in D_2 \wedge (s',t') \in R))))$.

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 $(s,t) \in R$, $init(s) = \phi \Leftrightarrow init(t) = \phi$.*

We now define the equivalences considered in this paper.

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 $\mathcal{T}_1 \sim_{bis} \mathcal{T}_2$, 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 $(s_1, t_1) \in R$.*

$\mathcal{T}_1$ and $\mathcal{T}_2$ are **simulation equivalent***, denoted by $\mathcal{T}_1 \sim_{sim} \mathcal{T}_2$, iff both $\mathcal{T}_1 \preceq_{sim} \mathcal{T}_2$ and $\mathcal{T}_2 \preceq_{sim} \mathcal{T}_1$.*

$\mathcal{T}_1$ is said to be **ready-simulated** *by $\mathcal{T}_2$, denoted by $\mathcal{T}_1 \preceq_{rsim} \mathcal{T}_2$, iff there is a ready-simulation R such that $(s_1, t_1) \in R$.*

$\mathcal{T}_1$ and $\mathcal{T}_2$ are **ready-simulation equivalent***, denoted by $\mathcal{T}_1 \sim_{rsim} \mathcal{T}_2$, iff both $\mathcal{T}_1 \preceq_{rsim} \mathcal{T}_2$ and $\mathcal{T}_2 \preceq_{rsim} \mathcal{T}_1$.*

$\mathcal{T}_1$ is said to be **complete-simulated** *by $\mathcal{T}_2$ [vG90], denoted by $\mathcal{T}_1 \preceq_{csim} \mathcal{T}_2$, iff there is a complete-simulation R such that $(s_1, t_1) \in R$.*

$\mathcal{T}_1$ and $\mathcal{T}_2$ are **complete-simulation equivalent***, denoted by $\mathcal{T}_1 \sim_{csim} \mathcal{T}_2$, iff both $\mathcal{T}_1 \preceq_{csim} \mathcal{T}_2$ and $\mathcal{T}_2 \preceq_{rsim} \mathcal{T}_1$.*

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

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

Let B be a **total**[4] *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 $\mathcal{T}_2 \preceq_B \mathcal{T}_1$*

Now we consider a generalization of simulation relations.

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 γ-simulation be a relation from S to T with certain constraints. As in all our previous definitions, we say $\mathcal{T}_1 \preceq_\gamma \mathcal{T}_2$ if there is a γ-simulation $R \subseteq S \times T$ containing (s_1, t_1). We also define γ-simulation equivalence as $\mathcal{T}_1 \sim_{\gamma sim} \mathcal{T}_2$ iff both $\mathcal{T}_1 \preceq_\gamma \mathcal{T}_2$ and $\mathcal{T}_2 \preceq_\gamma \mathcal{T}_1$.*

We also define a γ-simulation relation R as γ-bisimulation if R^{-1} is also a γ-simulation relation.

Suppose for some γ-simulation relation defined on the transition systems, $\mathcal{T}_1 \preceq_\gamma \mathcal{T}_2$ is shown by constructing a binary relation R from S to T such that R is a γ-simulation relation containing (s_1, t_1). Then we say that R is a **witness** *of the γ-simulation of $\mathcal{T}_1$ by $\mathcal{T}_2$.*

To determine whether two transition systems are γ-simulation equivalent, we have to solve two decision problems namely, whether $\mathcal{T}_1 \preceq_\gamma \mathcal{T}_2$ and whether $\mathcal{T}_1 \preceq_\gamma \mathcal{T}_2$. Suppose existence of γ-simulation is reducible to $NHORNSAT$. Then according to our reduction scheme this amounts to solving two $NHORNSAT$ instances independently. However, in order to decide if they are **γ-bisimulation** equivalent, we have to solve the conjunction of those two instances of $NHORNSAT$. However, the conjunction may be unsatisfiable while each of the two separate instances are satisfiable. This would point out that **γ-bisimulation** is a finer relation than **γ-simulation**.

Now we define $NHORNSAT$ to which we reduce all our problems.

Definition 2.8 *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 follows that NHORNSAT is also solvable in linear time.

Lemma 2.9 *NHORNSAT is solvable in linear time [DG84].*

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

3 Complexity of Bisimulation Equivalence

It is well known that bisimulation equivalence of finite transition systems is decidable in polynomial time [KS90]. In [KS90] there is an efficient algorithm for this problem with time complexity $O(mlogn + n)$ where n is the number of states in the two transition systems and m is the number of transitions in the transition relations. Their algorithm is a bottom-up refinement taking advantage of Paige and Tarjan's efficient algorithm for *relational coarsest partition* problem [PT87]. Here, we give an alternative proof of the polynomial decidability of the strong bisimulation equivalence by a polynomial time reduction to the satisfiability problem of Horn formulas in order to illustrate our methodology.

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.

We give a polynomial time algorithm that takes $\mathcal{T}_1$ and $\mathcal{T}_2$ as input and outputs an instance h of $NHORNSAT$. In instance h the number of variables is $\leq nm$ (where $\mid S \mid= n$ and $\mid T \mid= m$) and the number of clauses is $O(\mid D_1 \mid + \mid D_2 \mid)$.

The algorithm is given in Figure 1. Before giving an informal description of the algorithm, we explain some of the functions used in the description of the algorithm.

Let $STEP_1(a, p) = \{p^{'} \mid (p, a, p^{'}) \in D_1\}$.

Let $STEP_2(a, q) = \{q^{'} \mid (q, a, q^{'}) \in D_2\}$.

Now $CLAUSE(p, a, p^{'}, q) = \bigvee_{q^{'} \in STEP_2(a,q)} X_{p^{'},q^{'}}$ if $STEP_2(a, q) \neq \phi$ else false.

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 bisimulation relation. Given a transition $(p, a, p') \in D_1$ there must be a transition $(q, a, q') \in D_2$ and (p', q') must be in the bisimulation relation. $CLAUSE$ 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 $CLAUSE$.

$CLAUSE'$ and $VARS'$ are defined in a similar manner except that $STEP_1$ is used instead of $STEP_2$. The algorithm constructs an instance h of $NHORNSAT$ such that h is satisfiable iff $\mathcal{T}_1$ and $\mathcal{T}_2$ are bisimulation equivalent. So we start with the single literal clause X_{s_1,t_1}. Then, for each step from s_1 via some action a to some state s, we construct a clause specifying that there is some step in $\mathcal{T}_2$ from t_1 via action a to some state t such that (s, t) is in the bisimulation relation which **witnesses** the bisimulation between $\mathcal{T}_1$ and $\mathcal{T}_2$. The constructed $NHORNSAT$ instance h has the property that given any satisfying assignment, if we specify a relation R by saying that $(p, q) \in R$ iff $X_{p,q} = 1$ in the satisfying assignment and $X_{p,q}$ appears in h, then R witnesses the bisimulation equivalence between $\mathcal{T}_1$ and $\mathcal{T}_2$. Further more, if $\mathcal{T}_1 \sim_{bsim} \mathcal{T}_2$ then there exists a witness bisimulation $R \subseteq S \times T$ such that, if we specify the truth assignment by saying that $X_{p,q} = 1$ iff $(p, q) \in R$, then this truth assignment satisfies h.

Before we prove the correctness of the reduction algorithm and prove its polynomial time complexity let us consider the examples in figure 2. Consider $\mathcal{T}_1$ and $\mathcal{T}_2$ in Fig A. They are not bisimulation equivalent and we shall see that the $NHORNSAT$ instance will not be satisfiable.

In the example of Figure 2(a) the set of clauses output by algorithm Bisim-NHorn are is the following:

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), both have no actions out of them, the pair may be in the relation.*

   (c) If for no $a \in Act$ there is $p' \in S$ such that $(p, a, p^{'}) \in D_1$
   and for no $a \in Act$ there is a $q' \in T$ such that $(q, a, q^{'}) \in D_2$ then
   $C := C \cup \{X_{p,q}\}$;

   Comment: *Steps (d) and (e) create implication clauses to encode the conditions of bisimulation 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
   (e) **for** each $a \in Act, q' \in T$
   such that $(q, a, q') \in D_2$ **do**
   $C := C \cup \{\overline{X_{p,q}} \vee CLAUSE'(q, a, q^{'}, p)\}$
   $V := V \cup (VARS'(q, a, q^{'}, p) - W)$
   endfor

   end do
3. Output C.

Figure 1: Algorithm for reducing the bisimulation instance to $NHORNSAT$ instance: Algorithm Bisim-NHorn.

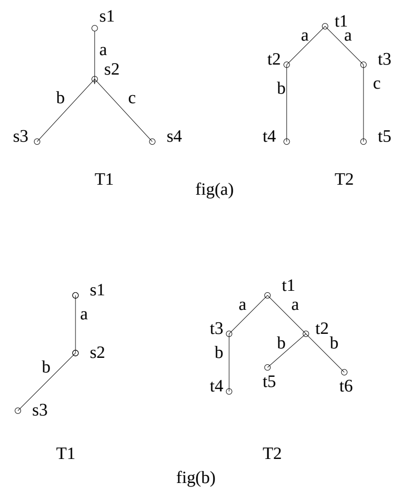

Figure 2: 2(a) Illustrating $\mathcal{T}_1 \not\sim_{bsim} \mathcal{T}_2$ 2(b) Illustrating $\mathcal{T}_1 \sim_{bsim} \mathcal{T}_2$.

$\{X_{s_1,t_1}, \overline{X_{s_1,t_1}} \vee X_{s_2,t_2} \vee X_{s_2,t_3}, \overline{X_{s_1,t_1}} \vee X_{s_2,t_2}, \overline{X_{s_1,t_1}} \vee X_{s_2,t_3}, \overline{X_{s_2,t_2}} \vee X_{s_3,t_4}, \overline{X_{s_2,t_2}}, \overline{X_{s_2,t_3}} \vee X_{s_4,t_5}, \overline{X_{s_2,t_3}}, X_{s_3,t_4}, X_{s_4,t_5}\}$

One can easily check that this set of weakly negative clauses are not satisfiable and hence the two finite transition systems are not bisimulation equivalent. Now consider the example of Figure 2(b). These transition systems are bisimulation equivalent [Mil89]. The set of clauses output by algorithm Bisim-NHorn is the following:

$\{X_{s_1,t_1}, \overline{X_{s_1,t_1}} \vee X_{s_2,t_2} \vee X_{s_2,t_3}, \overline{X_{s_1,t_1}} \vee X_{s_2,t_2}, \overline{X_{s_1,t_1}} \vee X_{s_2,t_3}, \overline{X_{s_2,t_3}} \vee X_{s_3,t_4}, \overline{X_{s_2,t_2}} \vee X_{s_3,t_5} \vee X_{s_3,t_6}, \overline{X_{s_2,t_2}} \vee X_{s_3,t_5}, \overline{X_{s_2,t_2}} \vee X_{s_3,t_6}, X_{s_3,t_4}, X_{s_3,t_5}, X_{s_3,t_6}\}$

One can easily check that this set of clauses has a satisfying assignment namely $X_{s_1,t_1} = X_{s_2,t_2} = X_{s_2,t_3} = X_{s_3,t_4} = X_{s_3,t_5} = X_{s_3,t_6} = 1$ and all other variables 0. Thus the witness bisimulation relation from S to T is $\{(s_1,t_1),(s_2,t_2),(s_2,t_3),(s_3,t_4),(s_3,t_5),(s_3,t_6)\}$.

Now that we have explored two examples, we are ready to prove the correctness and polynomial time complexity of this algorithm for reducing *bisimulation equivalence of two finite transition systems* to $NHORNSAT$. The fact that $NHORNSAT$ is linear time solvable [DG84] together with the correctness of the polynomial time reduction proves that bisimulation equivalence problem for finite transition systems is in P.

Theorem 3.1 *The $NHORNSAT$ instance produced by the algorithm* **Bisim-NHorn** *from two finite transition systems has a satisfying assignment iff the two input transition systems are bisimulation equivalent.*

The following sequence of lemmas will establish this theorem.

Lemma 3.2 *If $\mathcal{T}_1 \sim_{bsim} \mathcal{T}_2$ and $R \subseteq S \times T$ is a witness bisimulation relation then following conditions hold: for all $\sigma \in Act^*$, if $s_1 \stackrel{\sigma}{\Longrightarrow} p$ then there exists $q \in T$ such that $t_1 \stackrel{\sigma}{\Longrightarrow} q$ and $(p,q) \in R$ and if $t_1 \stackrel{\sigma}{\Longrightarrow} q$ then there exists $p \in S$ such that $s_1 \stackrel{\sigma}{\Longrightarrow} p$ and $(p,q) \in R$.*

Proof: By induction on the length of σ. $\Box$.

The following lemma asserts that if $\overline{X_{p,q}}$ is a clause in the NHORNSAT instance then there is no bisimulation relation that can contain the pair (p,q).

Lemma 3.3 *If clause $\overline{X_{p,q}}$ (a clause consisting of a single negative literal) appears in the $NHORNSAT$ instance created by the algorithm* **Bisim-NHorn**, *then there is no bisimulation relation R from S to T such that $(p,q) \in R$.*

Proof: The claim in the above lemma is that no bisimulation relation witnessing the bisimulation equivalence of $\mathcal{T}_1$ and $\mathcal{T}_2$ can have the pair (p,q) in it. Such a clause can be generated either in step 3(d) or in step 3(e) of the algorithm **Bisim-NHorn**. That is possible when $CLAUSE(p,a,p',q)$ or $CLAUSE'(q,a,q',p)$ returns **false**. This can happen if $STEP_2(a,q) = \phi$ for some $a \in Act$ or if $STEP_1(a,p) = \phi$ for some $a \in Act$ respectively. In the first case this means that $(p,a,p') \in D_1$ for some $p' \in S$ but there is no $q' \in T$ such that $(q,a,q') \in D_2$. Then obviously by the definition of bisimulation relation p and q are not in any bisimulation relation. The latter case can be argued similarly. $\Box$.

A pair of states $(p,q) \in S \times T$ is called a **simultaneously reachable pair** if there is a $\sigma \in Act^*$ such that $s_1 \stackrel{\sigma}{\Longrightarrow} p$ as well as $t_1 \stackrel{\sigma}{\Longrightarrow} q$. If there is a bisimulation relation containing (s_1,t_1) then there is a bisimulation that does not contain any pair $(p,q) \in S \times T$ which is not simultaneously reachable. The following lemma states this fact.

Lemma 3.4 *If $\mathcal{T}_1 \sim_{bsim} \mathcal{T}_2$ and $R = \{(s_1,t_1)\} \cup \{(p,q) \mid \exists_{\sigma \in Act^*}(s_1 \stackrel{\sigma}{\Longrightarrow} p \wedge t_1 \stackrel{\sigma}{\Longrightarrow} q)\}$ then R is a witness bisimulation.*

Proof: Consider $(p,q) \in R$. By definition of R there exists $\sigma \in Act^*$ such that $s_1 \stackrel{\sigma}{\Longrightarrow} p$ and $t_1 \stackrel{\sigma}{\Longrightarrow} q$. Suppose $(p,a,p') \in D_1$. Then $s_1 \stackrel{\sigma a}{\Longrightarrow} p'$. By lemma 3.2 there is $q' \in T$ such that $t_1 \stackrel{\sigma a}{\Longrightarrow} q'$. But since $t_1 \stackrel{\sigma}{\Longrightarrow} q$, $(q,a,q') \in D_2$. And since $s_1 \stackrel{\sigma a}{\Longrightarrow} p'$ as well as $t_1 \stackrel{\sigma a}{\Longrightarrow} q'$, by definition of R, $(p,q') \in R$. Similarly we can also show that if $(q,a,q') \in D_2$ then there is a $p' \in S$ such that $(p,a,p') \in D_1$ and $(p',q') \in R$. More over R contains (s_1,t_1), hence R is an witness bisimulation. $\Box$.

Lemma 3.5 *If the instance of $NHORNSAT$ output by the algorithm* **Bisim-NHorn** *is satisfiable then $\mathcal{T}_1 \sim_{bsim} \mathcal{T}_2$.*

Proof: Suppose the $NHORNSAT$ instance (say h) is satisfiable. Let γ be truth assignment to the variables $X_{p,q}$. Let $R \subseteq S \times T$ be defined by $(p,q) \in R$ iff the variable $X_{p,q}$ appears in h and $\gamma(X_{p,q}) = 1$. We claim that R is an witness bisimulation. Since X_{s_1,t_1} appears as a single literal clause in h, $\gamma(X_{s_1,t_1}) = 1$ and thus $(s_1,t_1) \in R$. For single negated literal clauses of the form $\overline{X_{p,q}}$ $\gamma(X_{p,q}) = 0$ and that means (p,q) is not in R which by lemma 3.3 should be the case. The variables of the form $X_{i,j}$ which does not appear in h correspond to simultaneously unreachable pairs. By lemma 3.4 we can always construct a bisimulation without those pairs. Now for any $(p,q) \in R$ we have to show that for all $a \in Act$ $(p,a,p') \in D_1$ there is $q' \in T$ such that $(q,a,q') \in D_2$ and $(p',q') \in R$ and the reverse. Now if $(p,q) \in R$ then $\gamma(X_{p,q}) = 1$ and thus in the conditional clauses where it appears on the left-hand side, one of the disjuncts on the right-hand side must be true. By our construction the corresponding left hand side is $CLAUSE(p,a,p',q)$ or $CLAUSE'(q,a,q',p)$ Thus one of the disjunct $X_{p',q'}$ being true means that there is a $q' \in T$ such that $(q,a,q') \in D_2$ and $X_{p',q'} = 1$ which means $(p',q') \in R$. The reverse condition can also be argued in a similar way. $\Box$.

Lemma 3.6 *If $\mathcal{T}_1 \sim_{bsim} \mathcal{T}_2$ then the instance of $NHORNSAT$ produced by algorithm* **Bisim-NHorn** *is satisfiable.*

Proof: Let R be a witness bisimulation. Let γ be the assignment of truth values to the variables $X_{p,q}$, given by

$$X_{p,q} = 1 \text{ if } (p,q) \in R$$
$$X_{p,q} = 0 \text{ if } (p,q) \not\in R.$$

Since R is a witness bisimulation $(s_1,t_1) \in R$, and so $\gamma(X_{s_1,t_1}) = 1$ and thus the single positive literal clause X_{s_1,t_1} is satisfied. The single negated literal clauses of the form $\overline{X_{p,q}}$ will be satisfied by lemma 3.3. All other clauses are of the form $X_{p,q} \Rightarrow \bigvee_{i,j} X_{i,j}$. If (p,q) is not in R then such a clause will be trivially satisfied. But if $(p,q) \in R$, then to satisfy such a clause one of the disjuncts on the right hand side must be satisfied. But the disjunct is computed in such a manner in our construction such that if the corresponding condition for (p,q) being in a bisimulation is true then one of the disjuncts must be true and (p,q) is indeed in a bisimulation. $\Box$.

An examination of Algorithm Bisim-NHorn proves the following:

Theorem 3.7 *The algorithm* **Bisim-NHorn** *runs in* $O(|\mathcal{T}_1||\mathcal{T}_2|)$ *time and produces an instance of* $NHORNSAT$ *with at most* $(|S||T|)$ *variables and* $O(|\mathcal{T}_1|+|\mathcal{T}_2|)$ *clauses.*

We note that the above algorithm decides strong-bisimulation of transition systems. However, it could easily be extended to the problem of deciding weak bisimulation or observational equivalence by reducing that problem to the strong bisimulation problem as in [KS90].

4 Complexity of Backward Simulation

The reduction from the *backward simulation* instance to $NHORNSAT$ is different from the reduction in the previous section 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 same actions can lead *to* them rather than on what actions are possible *from* them.

As a result, unlike the other cases, here we have to consider all possible pairs, and the computation of the conditional clauses is slightly different from the other cases.

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 $\overline{D_2} = \{(p, a, p') \mid \exists \sigma \in \tau^* a \tau^*, p \stackrel{\tau^* a \tau^*}{\Longrightarrow} p', a \in Act(\mathcal{T}_2)\}$. This can be done 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. **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 $|S| = n$ and $|T| = m$). The number of clauses in this $NHORNSAT$ instance will be $O(|D_1||D_2|)$. We now define the functions used in the description of the reduction algorithm.

The functions $BACKCLAUSES$ and $BACKVARS$ are defined as follows:

Let $BACKEXTSTEP_2(a, q') = \{q \mid (q, a, q') \in \overline{D_2}\}$.

Let $BACKSTEP_1(a, p') = \{p \mid (p, a, p') \in D_1\}$.

Now $BACKCLAUSES(p', a, p, q') = \bigvee_{q \in BACKEXTSTEP_2(a,q')} X_{p,q}$
if $BACKEXTSTEP_2(a, q') \neq \phi$ else false

$BACKVARS(p', a, p, q') = \{X_{p,q} \mid q \in BACKEXTSTEP_2(a, q')\}$
if $BACKEXTSTEP_2(a, q') \neq \phi$ else ϕ.

Our algorithm constructs an instance h of $NHORNSAT$ such that h is satisfiable iff $\mathcal{T}_1 \preceq_B \mathcal{T}_2$. So we start with the single literal clause X_{s_1,t_1}. We also add single negative literal clauses $\overline{X_{s_1,t_j}}$ for all

$j \neq 1$ (to satisfy the first condition of backward simulation). 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 backward simulation from S to T.

To prove that algorithm BACKNHorn produces an instance h of $NHORNSAT$ such that h is satisfiable iff $\mathcal{T}_1 \preceq_B \mathcal{T}_2$ we begin with the following lemmas.

Lemma 4.1 *If $\overline{X_{p,q}}$ is clause in h then there is no backward simulation $B \subseteq S \times T$ such that $(s_1, t_1) \in B$ as well as $(p, q) \in B$.*

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 $t_j \neq t_1$. Now certainly the pairs (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. That shows, by the definition of $BACKCLAUSES$ the condition for (p', q') to be in B is not satisfied. □.

Lemma 4.2 *If $X_{p,q}$ is a clause of h and $p \neq s_1$, then for no $p' \in S$ transition $(p', a, p) \in D_1$ and for no $q' \in T$ $(q', a, q) \in \overline{D_2}$.*

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. □.

Lemma 4.3 *If there is a single literal clause of the form $X_{p,q}$ with $p \neq s_1$ 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 4.2, there are no transitions into p and q. Therefore the conditions of backward simulation will be trivially satisfied. Hence, $R \cup \{(p, q)\}$ will also be a backward simulation. □

Lemma 4.4 *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 $R \subseteq S \times T$. 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 4.1. Any other single positive literal clause of the form $X_{p,q}$ will be satisfied because by Lemma 4.3 we can 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

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') both have no actions leading to them, the pair may be in the relation.*

   (c) If there is no $t \in D_1$ such that $t = (p, a, p')$
   and there is no $t \in \overline{D_2}$ such that $t = (q, a, q')$ then
   $C := C \cup \{X_{p',q'}\}$;

   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 BACKNHorn.

order that $(s_i, t_j) \in R$, there must exist $(s_i', t_j') \in R$ satisfying 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. $\Box$.

Lemma 4.5 *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 the backward simulation.

Now consider the implication clauses in h. They are all satisfied. Now if an implication clause is trivially satisfied because its lefthand side literal $X_{p,q}$ is false (in any $NHORNSAT$ instance, lefthand 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. $\Box$

Theorem 4.6 *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.*

Now by the definition of **backward simulation equivalence** to decide whether $\mathcal{T}_1 \sim_B \mathcal{T}_2$ we first decide whether $\mathcal{T}_1 \preceq_B \mathcal{T}_2$ and then decide whether $\mathcal{T}_2 \preceq_B \mathcal{T}_1$. This amounts to solving two separate instances of the $NHORNSAT$ problem.

However, suppose one wants to solve **Bi-Backward-Simulation** (defined as a backward simulation relation $R \subseteq S \times T$ witnessing $\mathcal{T}_1 \preceq_B \mathcal{T}_2$ such that R^{-1} is a backward simulation witnessing $\mathcal{T}_2 \preceq_B \mathcal{T}_1$), then conjunction of the two $NHORNSAT$ instances needs to be solved. Clearly that can be done in polynomial time.

5 Complexity of Simulation Equivalence, Forward Simulation, Ready Simulation and Complete Simulation

In the previous sections we have proved the polynomial time decidability of bisimulation equivalence and backward simulation using our uniform technique. Since both examples are illustrative of our methodology, in this section we briefly discuss, how the same reduction technique works for proving the polynomial time decidability of the other four relations considered in this paper. First we consider the simulation equivalence and forward simulation together because their definitions are very similar. In fact, forward simulation decidability can be reduced to simulation decidability exactly in the same way as in [KS90], where observational equivalence is reduced to strong bisimulation equivalence by precomputing the extended transition relation in polynomial time. Subsequently, we briefly discuss the modifications needed for ready simulation and complete simulation. Due to lack of space we do not provide any of the correctness proofs but they are similar to the correctness proofs in the previous sections.

5.1 Simulation and Forward Simulation

Given two finite transition systems $\mathcal{T}_1$ and $\mathcal{T}_2$, deciding $\mathcal{T}_1 \preceq_{sim} \mathcal{T}_2$ is decidable in polynomial time [HT94]. Here we outline how our generic reduction can be used to prove this. To decide $\mathcal{T}_1 \sim_{sim} \mathcal{T}_2$ we have to apply the above method twice, once to check whether $\mathcal{T}_1 \preceq_{sim} \mathcal{T}_2$ and then to check whether $\mathcal{T}_2 \preceq_{sim} \mathcal{T}_1$.

The reduction algorithm in this case is very similar to the algorithm **bisim-NHorn** except that step 3(e) is omitted. The polynomial time reduction algorithm produces an instance of $NHORNSAT$ which has a satisfying assignment iff $\mathcal{T}_1 \preceq_{sim} \mathcal{T}_2$. This gives us a polynomial time algorithm to decide if $\mathcal{T}_1 \preceq_{sim} \mathcal{T}_2$.

The same method will work for deciding $\mathcal{T}_1 \preceq_F \mathcal{T}_2$ with a small modification. In this case, we have to compute $\overline{D_2} = \{(p, a, p') \mid p \stackrel{\hat{a}}{\Longrightarrow} p', a \in Act(\mathcal{T}_2)\}$. As proved in [KS90] this will take polynomial time. With $\overline{D_2}$ as the transition relation of $\mathcal{T}_2$ the same algorithm will work for forward simulation.

5.2 Ready simulation and Complete simulation

The reduction algorithm in the ready simulation case is almost similar to the simulation case except that the definitions of $CLAUSE$ and $VARS$ are to be modified as described below.

As before, 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.

If $(p_i, a, p_i^{'}) \in D_1$ then let $STEPR(a, q_j) = \{q_j^{'} \mid \exists (q_j, a, q_j^{'}) \in D_2 \wedge init(p_i^{'}) = init(q_j^{'})\}$.

This modification is needed because the definition of ready simulation requires that for (p, q) to be in the ready simulation relation they must satisfy $init(p) = init(q)$ in addition to satisfying the conditions for being in a simulation relation.

Let $RCLAUSE(p_i, a, p_i^{'}, q_j) = \bigcup_{q_j^{'} \in STEPR(a,q_j)} X_{p_i^{'}, q_j^{'}}$ if $STEPR(a, q_j) \neq \phi$ else false

$RVARS(p_i, a, p_i^{'}, q_j) = \{X_{p_i^{'}, q_j^{'}} \mid q_j^{'} \in STEPR(a, q_j)\}$ if $STEPR(a, q_j) \neq \phi$ else ϕ.

To get the desired polynomial time reduction algorithm we just have to

1. Replace $CLAUSE$ and $VARS$ by their new definitions given above in the reduction algorithm for simulation case.
2. Add a first step that checks whether $init(s_1) \neq init(t_1)$ and returns an unsatisfiable formula of the form $X_{s_1,t_1} \wedge \overline{X_{s_1,t_1}}$ and terminates the algorithm.

This the algorithm produces an instance h of $NHORNSAT$ such that h has a satisfying assignment iff $\mathcal{T}_1 \preceq_{rsim} \mathcal{T}_2$. Thus to decide, $\mathcal{T}_1 \sim_{rsim} \mathcal{T}_2$, one can use this method twice.

In case of complete simulation we slightly modify the definitions of the function $RCLAUSE$ and $RVARS$ in the following manner. Define $STEPC(a, q_j) = \{q_j^{'} \mid \exists (q_j, a, q_j^{'}) \in D_2 \wedge (init(p_i^{'}) = \phi \Leftrightarrow init(q_j^{'}) = \phi)\}$.

This modification is needed because the definition of complete simulation requires a pair (p, q) to satisfy the conditions for being in a simulation relation and moreover $init(p) = \phi \Leftrightarrow init(q) = \phi$.

Now to get a polynomial time reduction to NHORNSAT

1. Use $STEPC$ in place of $STEPR$ in the definitions of $RCLAUSE$ and $RVARS$.
2. Add a first step which adds X_{s_1,t_1} as a clause if the extra condition for complete simulation is satisfied. If not, the algorithm terminates producing an unsatisfiable formula of the form $X_{s_1,t_1} \wedge \overline{X_{s_1,t_1}}$.

As a result of our brief discussion above we reach the following conclusion.

Theorem 5.1 *Problems of deciding* **simulation, forward simulation, ready simulation** *and* **complete simulation** *equivalences of transitions systems are all polynomial time reducible to the problem of satisfiability of Horn formulas. Hence, they are all polynomial time decidable.*

6 A corollary

In this section we show that our reduction can be done in **NC** for *deterministic* transition systems and the instances of the CNF formula produced will be instances of **2-SAT**. It is well known that **2-SAT** has an NC-algorithm. Combining these two facts we prove that the bisimulation equivalence for deterministic transition systems is in **NC** which solves an open problem posed in [GHR95]. Also notice that all the equivalences between bisimulation and trace coincide for deterministic transition systems and hence all these equivalences are in **NC** for deterministic systems. This shows that it is possible to design fast parallel algorithms for deciding these equivalences when the transition systems are deterministic. In contrast, in case of nondeterministic systems the bisimulation equivalence is **P**-complete [ABGS91] which means that it is unlikely to have fast parallel algorithms for deciding bisimulation equivalence of nondeterministic transition systems.

The reason why our reduction algorithm is an NC algorithm follows from the fact that for deterministic transition system for each state there is only one transition on a particular action. We take $n \times m \times k$ processors (we are not optimising the number of processors here, which can be done) where n and m is the number of states in the two transition systems and k is the size of the action alphabet. Each processor correspond to a pair of states and a particular action. Let P^a_{pq} be the processor that correspond to states p of the first systems and state q of the second system and to the action a. This processor generates a clause of the form $X_{p,q} \rightarrow X_{s,t}$ if there is a transition (p, a, s) in the first transition system and correspondingly there is a transition (q, a, t) in the second system. 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 generated the clause $True$.

It is not difficult to see that each process takes constant time to generate appropriate clauses. Since the system is deterministic these clauses are either single literal or 2-literal clauses and hence the instance of CNF produced will be a $2-SAT$ instance. It is known as in [GHR95] that there is an NC algorithm for $2-SAT$ and hence by a combination of the NC reduction and an NC algorithm for $2-SAT$ we get an NC algorithm for the bisimulation equivalence of deterministic transition system.

Note that our reduction is NC even for nondeterministic transition systems but for simplicity we only show that for deterministic case.

Theorem 6.1 Bisimulation equivalence *of* deterministic *transition systems is in* **NC**.

Proof: Rigorous proof is omitted due to lack of space. However, the proof sketch is in the same line as the above discussion.□

References

[ABGS91] C. Alvarez, J.L Balcazar, J. Gabarro, and M Santha. Parallel complexity in the design and analysis of concurrent systems. In *PARLE91, Lecture Notes in Computer Science 505 Springer-Verlag*, 1991.

[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.

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

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

[Mil80] R. Milner. A calculus of communicating systems. *Lecture Notes in Computer Science*, 92, 1980.

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

[Par81] D. Park. Concurrency and automata on infinite sequences. In *Theoretical Computer Science, 5th GI. Conference, LNCS 104*, pages 167–183, 1981.

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

[PT87] Robert Paige and Robert E Tarjan. Three partition refinement algorithms. *SIAM Journal Of Computing*, 16:973–989, 1987.

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

[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.