HORNSAT, Model Checking, Verification and Games*

(Extended Abstract)

Sandeep K. Shukla[1] Harry B. Hunt III[1]
Daniel J. Rosenkrantz[1]

Department of Computer Science
University at Albany – State University of New York
Albany, NY 12222
Email: {sandeep,hunt,djr}@cs.albany.edu

Abstract. We develop a HORNSAT-based methodology for verification of finite state systems. This general methodology leads naturally to algorithms, that are *local* [25, 19], *on the fly* [28, 11, 13, 5] and *incremental* [24]. It also leads naturally to *diagnostic* behavioral relation checking [7] algorithms. Here we use it to develop model checking algorithms for various fragments of modal mu-calculus. We also use our methodology to develop a uniform game theoretic formulations of all the relations in the linear time/branching time hierarchy of [27]. As a corollary, we obtain natural sufficient conditions on a behavioral relation ρ, for ρ to be polynomial time decidable for finite state transition systems.

1 Introduction

We consider a number of problems related to the verification of finite state systems which include model checking for various fragments of modal mu-calculus [15] and checking behavioral relations [10] with diagnostic information. We outline a methodology for solving these problems, based upon efficient *local* reductions to satisfiability problems for simple variants of HORN formulas. We use our methodology to develop *local, on the fly* and *incremental* algorithms and to generate diagnostic information for these problems. Our algorithms are asymptotically as efficient as other specific algorithms in the literature for the problems considered. The desirability of *local, on the fly,* and *incremental* verification algorithms and algorithms for generating diagnostic information has been widely discussed [28, 5, 7, 13, 17, 18, 11, 10, 1, 24, 25, 8, 7]. However, previous algorithms proposed in the literature have only some of these advantages and only apply to some of the verification problems considered here. Our uniform methodology combines all these advantages in the same solution. Another advantage of our methodology is that efficient data structures and algorithms for the appropriate satisfiability problems for HORN formulas already exist in the literature [12, 3].

Our methodology is based upon efficient *local* reductions of the problems considered to the minimal and maximal satisfiability problems, for weakly positive

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

and weakly negative [21] Horn formulas. We call these satisfiability problems minimal-HORNSAT and maximal-NHORNSAT respectively. In fact, restricted forms of these Horn formulas are enough for some of the problems. In Sections 2-4, we outline our (N)HORNSAT-based algorithm for model checking, for the alternation-free modal mu-calculus. We show that this algorithm is a simplification of the algorithms in [19, 1] involving solutions of systems of Boolean equations. (Recall that [19] involves *consistent* and *factual* solutions of Boolean equation systems and [1] involves maximal and minimal fixed points of Boolean equation systems.)

In Section 5, we use our (N)HORNSAT-based methodology to define a class of games, that includes the *characteristic games* for each of the behavioral relations in the linear-time/branching-time hierarchy of [27]. As a corollary, we get natural sufficient conditions, for a behavioral relation on finite state processes to be polynomial time decidable.

In [22], we show in details, how our (N)HORNSAT-based methodology can be used to develop efficient algorithms for diagnostic behavioral relation checking and model checking for the modal mu-calculus.

The main advantages of our methodology may be summarized as follows. First, it shows that the underlying combinatorics for a number of verification problems and their proposed solutions is essentially very simple. Second, it turns out that an efficient verifier can be based on an implementation whose core consists of a solver for (N)HORNSAT which runs in linear time, which can run *on the fly* for space efficiency, and can run *incrementally* (e.g., using simple modifications of the incremental HORNSAT algorithms given in [3]). Third, the fact that efficient solutions for HORNSAT and its variants already exist in the literature [12, 3] and that many important verification problems are reducible to those variants of HORNSAT makes the implementation of verification tools easier. Moreover, it relieves the designer of the verifier from the obligation of reinventing complex data structures which already exist in the literature on HORNSAT. Many model checking algorithms in the literature involved inventing complex new data structures, whereas existing efficient data structures for solving variants of HORNSAT are sufficient to obtain the same efficiency. Moreover, this approach leads to modular design, because the efficient implementation of HORNSAT solver can be delegated to a different designer. In [16] a data structure for a linear time algorithm for determining functional dependencies in relational databases [4] was reused to obtain a model checking algorithm for CTL. It is interesting to note that functional dependency is also reducible to HORNSAT, and in [3, 2] the same kinds of data structures are used to solve them in linear time. [2] In [1] the model checking problem for mu-calculus was reduced to finding fixed points of system of Boolean equations; and complex graph-based data structures were invented for efficiency. Our results show that

[2] However, (N)HORNSAT captures the essence of these problems more directly and intuitively. Moreover, efficient data structures for solving (N)HORNSAT are easily implementable. Also, HORNSAT based methods are directly implementable in DATALOG.

the full power of Boolean equations are not needed to solve these problems. Fourth, we identified many easiness results in the area of model checking and verification as a consequence of the corresponding easy instances of NHORN-SAT. For example, after characterizing special cases of HORNSAT which have NC algorithms, we could strengthen the results in [29] by characterizing cases when the model checking problem is in NC.

2 Satisfiability Problem for (N)HORNSAT

We consider special instances of CNF satisfiability problems, namely HORNSAT, where each clause contains at most one positive literal, and NHORNSAT, where each clause contains at most one negative literal. We are interested in finding maximal and minimal satisfying assignment (if one exists) respectively.

An instance of the problem is presented as a pair (X, C), where $X = \{x_1, x_2, ..., x_n\}$, a finite set of propositional variables which take Boolean values, and $C = \{C_1, C_2, ..., C_m\}$, a set of clauses with one of the restrictions discussed above. Note that if an instance has a satisfying assignment, such an assignment can be represented as an element of an n-dimensional Boolean lattice $\{0,1\}^n$. If we consider $0 < 1$, then with a component-wise extension of the ordering, and a component-wise $\wedge$ and $\vee$ as *meet* and *join* operation, we get a complete lattice. For an instance of a satisfiability problem h, we denote the set of all satisfying assignments as $SAT(h) \subseteq \{0,1\}^n$. An element $x \in SAT(h)$ is *minimal*, if no other $y \in SAT(h)$ is less than x in the ordering of $\{0,1\}^n$. Dually, an element $x \in SAT(h)$ is *maximal*, if no other $y \in SAT(h)$ is greater than x in the ordering of $\{0,1\}^n$. We call the problem of finding the maximal satisfying assignment for an NHORNSAT instance as the maximal-NHORNSAT problem, and the problem of finding the minimal satisfying assignment for a HORNSAT instance as the minimal-HORNSAT problem.

A linear time algorithm for minimal-HORNSAT appears in [12]. Dually the maximal-NHORNSAT is also solvable in linear time.

In some of our applications we have a special type of HORNSAT or NHORN-SAT instances. Here we discuss that special type of NHORNSAT, called *rooted NHORNSAT*. The corresponding cases and algorithms for HORNSAT are very similar.

Definition 1. Given a clause C_k of the form $x_j \Rightarrow \bigvee_{i \in I} x_i$, where I is an index set possibly empty (note that the disjunction $\bigvee_{i \in I} x_i = true$ when $I = \phi$.), we call x_j the **head** of clause C_k, denoted as $head(C_k) = x_j$, and $\bigvee_{i \in I} x_i$ the **tail** of C_k. Any variable x_i appearing in $tail(C_k)$, is called a disjunct in the tail.

Note that for a clause of the form $C_k = \overline{x_j}$, $head(C_k) = x_j$ and $tail(C_k) = false$. Similarly, for a clause of the form $C_k = x_j$, $head(C_k) = true$ and $tail(C_k) = x_j$.

Definition 2. An instance of a *rooted NHORNSAT* problem is of the form (X, C, x_1) where (X, C) is an NHORNSAT instance and the clauses in C are ordered. Also. $C_1 = x_1$ (a single positive literal clause), where $x_1 \in X$. Furthermore, for each clause C_k, if $head(C_k) = x_j$ then there must be a clause $C_l (l < k)$ preceding C_k, such that x_j is a disjunct in $tail(C_l)$. Also for a single literal clause $C_k \doteq x_p$ $(k > 1)$, x_p must also be a disjunct in $tail(C_l)$ for some $l < k$. and x_p cannot be the head of any clause.

The correctness of our (N)HORNSAT based methodology for model checking can be demonstrated easily by showing the following. There is a *local* reduction (see the proof sketch of Theorem 3 below) between the (N)HORNSAT based methodology and the methodologies in [19, 1] based upon systems of simple Boolean equations. The (N)HORNSAT based approach has the advantage that efficient algorithms and data structures for (N)HORNSAT are already available in the literature [12, 3]. The soundness and completeness of our methodology follow easily from the following theorem and its extensions to the results in [1].

Theorem 3. *The* factuality problem *and the* consistency problem *of system of* simple *Boolean equations described in [19] and the class of minimal-HORNSAT and maximal-NHORNSAT problems we consider, are* locally *and efficiently interreducible.*

Proof sketch: [3] Given a system of *simple* Boolean equations, if we are interested in factuality [19], we replace

an equation of the form $x = true$ by a single literal clause x,
an equation of the form $x = false$ by a single negated literal clause $\overline{x}$,
an equation of the form $x = x_1 \wedge x_2$ by a clause $x \Leftarrow x_1 \wedge x_2$, and
an equation of the form $x = x_1 \vee x_2$ by two clauses $x \Leftarrow x_1$ and $x \Leftarrow x_2$.

It is easy to prove that the variables which are assigned a value 1 in the minimal satisfying assignment for this HORNSAT instance are the factual variables of the original Boolean equational system. Since, we are considering minimal-HORNSAT, the implications can replace the equalities. Given this, duality implies that the consistency problem of [19] can be reduced efficiently and locally to the maximal-NHORNSAT problem.

Similarly, the problems of finding the least and greatest fixed points of the Boolean equations of [1] can be reduced to minimal-HORNSAT and maximal-NHORNSAT respectively. Details are omitted due to lack of space.

3 On the Fly, Local and Incremental Model Checking

Local Model Checking : A *local* model checking algorithm does not explore all the states of the finite state system, if not required. It tries to explore only a

[3] Given a set E of Boolean equations over a set of Boolean variables in V, the *factuality problem* is to find $F \subseteq V$ such that $x \in F$ if and only if x is set to *true* in every model of E. The *consistency* problem is to find $C \subseteq V$, such that $x \in C$ if and only if there exists a model of E in which x is set to *true*.

minimal set of states and determines whether certain properties are true in those states in order to infer that a given property is true in a given state. The tableau based methods in [18, 25, 6] are examples of such *local* algorithms for model checking. Our (N)HORNSAT based method achieves this objectives naturally. Given a fix point formula Φ, and a state s^* of a finite transition system, suppose we want to determine if s^* satisfies Φ. We generate (N)HORN formulas roughly as follows: We use a Boolean variable Y_s^{ϕ}, and create clauses such that s satisfies ϕ if and only if Y_s^{ϕ} is *true* in the (maximal) minimal satisfying assignment of the (N)HORNSAT instance.

On the Fly Model Checking : In [28, 11, 5, 16, 13] *on the fly* model checking and behavioral relation checking have been emphasized. In an *on the fly* algorithm the state space is constructed on demand, hence the verification takes place together with the construction of the state space. In our (N)HORNSAT based approach, *on the fly* algorithm is obtained naturally because of the existing *on the fly* or *online* algorithms for (N)HORNSAT [3] and some minor improvements on them. Our reduction to (N)HORNSAT can be done in NLOGSPACE and *on the fly* algorithm for HORNSAT works in $O(q)$ amortized time, where q is the size of each new clause generated. Since the size of the (N)HORNSAT instance created is linear in the product of the size of the transition system and the specification in the case of model checking, and product of the sizes of the two transition systems in case of relational checking, we might use in the worst case, linear space and linear time in those measures. For on the fly behavioral relation checking this is an improvement over [13] which requires quadratic time in these measures for behavioral relation checking. However, in most cases, counter examples are found after constructing substantially less number of clauses.

Incremental Model Checking : In [24], an incremental algorithm for model checking alternation free mu-calculus was developed. The basic idea was the following. When transitions are added or deleted from the transition system, an incremental algorithm exploits the information available from the previous runs of the model checking algorithm. It carries out minimal computation so that the model checking problem with respect to the changed transition system is solved in time $O(\Delta)$, where Δ is a measure of changes in the transition system. It has been pointed out [24] that in the worst case, this may not be possible. However, in the best case and more importantly, in many pragmatic situations the incremental computation could be linear in the size of the modification. Since the online algorithm for HORNSAT [3] is incremental and since the modification in the transition system will be reflected in the changes in the corresponding (N)HORNSAT instance, we can now directly obtain incremental algorithms for all the problems considered in this paper.

Note: The equational syntax of modal mu-calculus used in the subsequent sections is taken from [10]. Due to lack of space, the syntax and semantics could not be discussed and the readers are referred to [10, 22].

4 Model Checking Fragments of Modal Mu-Calculus

Our methodology can be extended to apply to full Mu-Calculus [15, 6], by using the model checking algorithm for the alternation-free fragment as a subroutine, as in [9] with the same efficiency as in [9]. Here we, illustrate our methods through its application to the *unnested single fixed point* fragment (which is similar to the Hennessy-Milner Logic with recursion [17, 18]) and to the alternation-free mu-calculus, as discussed in [10].

Model Checking for Single Fix point Mu-Calculus to (N)HORNSAT
For each state $s \in \mathcal{S}$ of the given finite state system $\mathcal{T}$ and each variable X_i of the equational specification, we associate a boolean variable $Y_s^{X_i}$. Recall, in the single fixpoint calculus, there is a single block of equations which is either a *max* block or a *min* block.

We consider the case when the block is a max block $B = max\{E\}$ where $E = \{X_1 = \Phi_1, ..., X_n = \Phi_n\}$. A dualization will hold for *min* blocks.

Here, the model checking problem is to determine if $s^* \in \|X_i\|_{\|B\|_e}$, for a given transition system $\mathcal{T} = \langle \mathcal{S}, Act, \rightarrow \rangle$, for an initial environment e, and $s^* \in \mathcal{S}$.

The reduction proceeds as follows:

1. Create a variable $Y_{s^*}^{X_i}$ and put the variable $Y_{s^*}^{X_i}$ in a queue.
2. For each variable of the form $Y_s^{X_j}$ on the queue, such that X_j appears in the left-hand side of an equation $\hat{e}$ in B

   (i) If $\hat{e}$ is $X_j = A$ where A is atomic, then create a clause Y_s^A if A is true at s else create a clause $\overline{Y_s^A}$. (This information is obtained from the valuation map associated with the model.) Put the variable Y_s^A in the queue if this variable was never on the queue before.
   (ii) If $\hat{e}$ is $X_j = X_p \vee X_q$, then create the clause $Y_s^{X_j} \rightarrow Y_s^{X_p} \vee Y_s^{X_q}$ and put the variables $Y_s^{X_p}$ and $Y_s^{X_q}$ into the queue, if these variables were never on the queue before.
   (iii) If $\hat{e}$ is $X_j = X_p \wedge X_q$, then create two clauses $Y_s^{X_j} \rightarrow Y_s^{X_p}$ and $Y_s^{X_j} \rightarrow Y_s^{X_q}$ and put the variables $Y_s^{X_p}$ and $Y_s^{X_q}$ into the queue, if they were never on the queue before.
   (iv) If $\hat{e}$ is $X_j = \langle a \rangle X_p$, then create a clause of the form $Y_s^{X_j} \rightarrow \bigvee_{s' \in a(s)} Y_{s'}^{X_p}$ where $a(s) = \{s' \mid \exists s' : s \xrightarrow{a} s'\}$. When $a(s)$ is empty, the disjunction is equivalent to false. Put the variables $Y_{s'}^{X_p}$ on the queue if they were never on the queue before.
   (v) If $\hat{e}$ is $X_j = [a]X_p$, then create clauses of the form $Y_s^{X_j} \rightarrow Y_{s'}^{X_p}$ for each $s' \in a(s)$ where $a(s) = \{s' \mid \exists s' : s \xrightarrow{a} s'\}$. Put the variables $Y_{s'}^{X_p}$ on the queue if they were never on the queue before. When $a(s)$ is empty, create the single literal clause $Y_s^{X_j}$.
3. If $Y_s^{X_j}$ is in the queue and if X_j does not appear on the left hand side in B, then if $s \in e(X_j)$, add a single literal clause $Y_s^{X_j}$ else add the clause $\overline{Y_s^{X_j}}$.

This will produce an NHORNSAT instance, of the size linear in the product of the size of the transition system and equational block B. We now state the theorem stating the correctness of the reduction. The correctness of the model checking algorithm obtained this way follows from the discussions in section 2.

Let $s \in \mathcal{S}$ is a state in the given finite state transition system $\mathcal{T} = \langle \mathcal{S}, Act, \rightarrow \rangle$. Let X_i be a variable in the equational block used in specifying a property using the syntax of [10] and let the initial environment be e. Suppose the block specifying the formula is a max block, $B = max\{E\}$ where $E = \{X_1 = \Phi_1, ..., X_n = \Phi_n\}$.

Theorem 4. *If h is the instance of NHORNSAT produced by the algorithm described above from the given model checking problem (if $s^* \in ||X_i||_{||B||_e}$), then h is satisfiable and in the maximal satisfying assignment of h, $Y_{s^*}^{X_i} = 1$, if and only if $s^* \in ||X_i||_{||B||_e}$.*

The dual of the above theorem holds for min blocks. Which means that in the minimal solution of the HORNSAT instance produced in that case, $Y_{s^*}^{X_i} = 1$ if and only if $s^* \in ||X_i||_{||B||_e}$. This gives us a linear time algorithm for the problem.

Alternation free mu calculus : Now we generalize the algorithm in the previous section, to obtain a (N)HORNSAT based algorithm for the model checking of alternation free mu-calculus. A linear time algorithm for the same problem was presented in [10]. Their algorithm needed to invent an efficient data structure to obtain the linear time algorithm. Our method brings out the fact that the essential data structure necessary to obtain the linear time algorithm for model checking is in fact the same as in [12] for the linear time algorithm for HORNSAT/NHORNSAT

Given a Transition system $\mathcal{T}$, a valuation map v, an initial environment e, a blockset $\mathcal{B}$, the model checking problem is to decide if $s^* \in ||X_i||_{||\mathcal{B}||_e}$, for a given state s^* in the transition system and a given variable X_i appearing on the left hand side of some equation in some block B_l in $\mathcal{B}$.

Briefly, the steps in the (N)HORNSAT based version of the algorithm for model checking alternation free mu-calculus are as follows:

1. Create a variable $Y_{s^*}^{X_i}$ and put the variable $Y_{s^*}^{X_i}$ in the queue associated with the block B_l where X_i appears on the left hand side.
2. Expand the variables in the queue associated with each block, in the reverse topological order, [4] with the following rules:
If the block is a max block then use the methods described in the previous subsection and if the block is a min block use a dual approach. Keep the NHORN or HORN clauses for each block separated. If new variable $Y_s^{X_j}$ is generated and X_j belongs to a different block B , put that variable in the queue associated with block B.

[4] Given $\mathcal{B}$, the block set, topologically sort the blocks in B with respect to the variable dependency relation depicted in block graph. Let $B_1, B_2, ..., B_m$ be the set of blocks in the topologically sorted order.

If the a variable $Y_s^{X_j}$ in the queue for a block B is already expanded then remove it from the queue otherwise expand it.
3. Start solving the minimal-HORNSAT/maximal-NHORNSAT instances corresponding to each block in the topological order. Let h_B be the HORNSAT/ NHORNSAT instance corresponding to block B. Suppose a variable $Y_s^{X_j}$ was assigned a value 1 in the solution of a h_B (where X_j appears on the left hand side in B) then add a clause $Y_s^{X_j}$ in the (N)HORNSAT instances corresponding to the blocks which had to put this variable in the queue of the block B (This information can be read off the block graph also). If $Y_s^{X_j}$ was assigned a value 0 in the solution of a h_B (where X_j appears on the left hand side in B) then add a clause $\overline{Y_s^{X_j}}$ in the (N)HORNSAT instances corresponding to the blocks which put this variable in the queue of the block B. Then continue solving the next block HORNSAT instance.

Suppose the block B corresponding to X_i, is a max block. (A Dual strategy holds for the min blocks). The maximal-NHORNSAT instance for the block B is satisfiable and $Y_{s^*}^{X_i} = 1$, in the maximal satisfying assignment, if and only if $s^* \in \|X_i\|_{\|B\|_e}$.

Note that this algorithm produces a sequence of HORNSAT and NHORNSAT instances and it is local and it can be made into an On the fly algorithm by noting that one can use the on the fly algorithm for each HORNSAT instance. We state the theorem about the correctness and efficiency of the algorithm sketched above with out proof.

Theorem 5. *The algorithm for model checking alternation free mu-calculus obtained by reducing the problem to a sequence of minimal-HORNSAT and maximal-NHORNSAT problems runs in time linear in the product of the sizes of the transition system and the block set specifying the property. Hence the HORNSAT based algorithm is as efficient as the algorithm in [10].*

We also have developed HORNSAT based methods to capture the tableau based local model checking in [8] and [25]. Details will appear in a future version of this paper.

5 Game for rooted (N)HORNSAT and Stirling Games

In [23] we show that many relational problems are also directly, locally, and naturally reducible to rooted NHORNSAT. Hence, given a two-player game for *rooted* NHORNSAT, we can easily associate games to all these relations as well. However, our objective is to obtain a sufficient characterization of various process algebraic behavioral relations, which helps us identifying whether a particular relation ρ, between finite transition systems is polynomial time decidable. In what follows, through a game theoretic formulation (similar to [26] where a characteristic game for bisimulation was defined,) we fulfill this objective. Such a natural sufficient characterization is really useful in identifying a polynomial time decidable relation when the definitions of the relations are complicated. [5]

[5] In [14], J. F. Groote who originally defined 2-nested simulation and $k-$nested simulation conjectured that deciding these relations must be NP-hard. However, by our

Game for *rooted* NHORNSAT: Game for an instance of a rooted NHORNSAT instance $h = (X, C, x_1)$ is a two player game G_h in which player I (the *spoiler*) wants to show that the instance h is not satisfiable and Player II (the *duplicator*) wants to show otherwise. The game proceeds in rounds. The spoiler opens the game by choosing a clause C_i such that $head(C_i) = x_1$. Duplicator reciprocates by choosing x_{ij} such that x_{ij} is a disjunct in $tail(C_i)$. In subsequent rounds, the spoiler chooses a clause C_k such that $head(C_k) = x_{ij}$ where x_{ij} was the duplicator's choice in the previous round. The duplicator has to reciprocate by choosing a disjunct in the tail of C_k. The game continues until one of the player loses. The duplicator loses if it does not have such a disjunct to choose (i.e, when the spoiler has chosen a clause of the form $\overline{x_l}$ in its last move), the spoiler loses when the game continues for ever (which is not possible in a finite size NHORNSAT instance) or when the spoiler chooses a clause chosen earlier. The following theorem states that the game we defined above, is indeed characteristic for rooted-NHORNSAT.

Theorem 6. *Given an instance $h = (X, C, x_1)$ of the* rooted *NHORNSAT problem, the duplicator has a winning strategy* [6] *in the corresponding game if and only if h is satisfiable.*

Stirling Class of Games: Now we describe a class of two player games called the *Stirling Class*. In this class, player I (the *duplicator* or *prover*) and player II (the *spoiler* or *disprover*) plays on two Finite transition systems. Each game in the class has the following components:

> Two Finite Transition systems $T_1 = 1$ and $T_2 = 2$; Two languages $R_1 \subseteq A^*$ and $R_2 \subseteq A^*$; Two total relations $m_2 \subseteq R_1 \times A^*$ and $m_2 \subseteq R_2 \times A^*$; A set of (*winning positions*) $\Gamma \subseteq S_1 \times S_2$; A set of starting positions $\Sigma \subseteq \Gamma \subseteq S_1 \times S_2$; A set $M \subseteq \{1, 2\}$ which denotes the indices of the coordinate of a position that spoiler can play on. In each round the duplicator plays on the other coordinate; and, A positive integer r denoting the number of rounds allowed in the game. This is crucial for some of the games.

The game starts in a position $\langle s, t \rangle \in \Sigma$. A *play* of the game is a finite or infinite length sequence of the form $\langle s_0^1, s_0^2 \rangle, ..., \langle s_i^1, s_i^2 \rangle, ...$ The *spoiler* wants to show that there is a *difference* between the two transition systems (the kind of difference it wants to show depends on the relation the game corresponds to). The *duplicator* wants to show that such a distinction attempted by the *spoiler* is not possible. A *partial play* in a game is a prefix of a *play* of the game. Let π_j be a *partial play* $\langle s_0^1, s_0^2 \rangle, ..., \langle s_j^1, s_j^2 \rangle$. The next pair $\langle s_{j+1}^1, s_{j+1}^2 \rangle$ is determined by the following move rule:

- The Spoiler picks a triple $\langle i, x, u \rangle$ such that $i \in M$ and $x \in R_i$ and $s_j^i \overset{x}{\Longrightarrow}_i u$. and $u = s_{j+1}^i$. (Note that $\Longrightarrow_i$ denotes an extended step in the transition system T_i).

characterization it is easy to see that they are polynomial time decidable. Moreover, many other relations such as $\frac{m}{2}$−nested relations[20] were shown to be polynomial time decidable this way.

[6] For the definition of winning strategy, see next subsection

- Let the choice of the spoiler in the move be $\langle i, x, u\rangle$ and let $i' \neq i$. Then the Duplicator picks a pair $\langle y, u'\rangle$ such that $(x,y) \in m_{i'}$ and $s_j^{i'} \xrightarrow{y}_{i'} u'$ and $u' = s_{j+1}^{i'}$.

Extending a partial play π_j to π_{j+1} by the above move rule is called a *round* of the game. Hence a play can be thought of as a sequence of rounds.

The duplicator wins the game if either in the last position of the play, there is no further allowable move by none (when $M = \{1,2\}$) or there is no further allowable move by the spoiler(when $|M| = 1$), depending on the cardinality of the set M. Duplicator also wins, if in the play a position is repeated. In both cases, the spoiler has failed to expose a distinction between the transition systems. The spoiler wins, if in the last position of the play is not a winning position which means the spoiler has been able to force the duplicator to a non winning position of the game or if in the last position, the spoiler has an allowable move but the duplicator does not have a matching move. A *strategy* for a player is a set of rules which tells him/her how to make a move depending on the partial play and opponent's move so far.

A strategy is a *winning strategy* for a player, if playing with that strategy, that player wins against all possible strategies of the opponent.

Definition 7. A game G in Stirling class is called a characteristic game for a relation R between two finite state processes, if the following condition holds. Let the game G be played on two transition systems T_1 and T_2 and the duplicator has a history free winning strategy if and only if T_1 and T_2 are related by the relation R.

Here, we illustrate characteristic games for *bisimulation*, *weak bisimulation*, and *Failure equivalence*. We assume in the following that all the games are being played on $T_1 = 1$ and $T_2 = 2$. [7]

Characteristic Game for Bisimulation : $Bsim - game$ is a game in Stirling class with the following parameters: $R_1 = R_2 = A$, $m_1, m_2 = \iota$, $\Gamma = S_1 \times S_2$, $\Sigma = \{\langle s_1, s_2\rangle\}$, $M = \{1,2\}$, $r = \mid S_1 \mid * \mid S_2 \mid +1$.

Characteristic Game For Weak Bisimulation: $WeakBsim - game$ is a game in Stirling class with the following parameters: $R_1 = R_2 = \tau^* A \tau^*$, $m_1(a) = \tau^* a \tau^*, m_2(a) = \tau^* a \tau^* \forall a \in A$, $\Gamma = S_1 \times S_2$, $\Sigma = \{\langle s_1, s_2\rangle\}$, $M = \{1,2\}$, $r = \mid S_1 \mid * \mid S_2 \mid +1$.

Characteristic Game For Failure Equivalence: $Failure - game$ is a game in Stirling class with the following parameters: $R_1 = R_2 = A^*$, $m_1, m_2 = \iota$, $\Gamma = \{\langle s,t\rangle \mid s \in S_1, t \in S_2 \wedge Failures(s) = Failures(t)\}$, $\Sigma = \{\langle s_1, s_2\rangle\}$, $M = \{1,2\}$, $r = 1$.

For each relation R, in the linear-time/branching time hierarchy, and its characteristic game G_R, the following theorem can be proved easily.

Theorem 8. *Let T_1, T_2 be two transition systems and let G_R be the instance of the characteristic game for a relation R, such that the game is played on T_1 and*

[7] Note that ι denotes the identity relation.

T_2. The duplicator has a winning strategy for this instance of the game G_R if and only if R holds between the given two transitions systems.

For a certain subclass of Stirling class, the problem whether the duplicator has a winning strategy is directly reducible to rooted NHORNSAT problem. Hence, for any behavioral relation, whose characteristic game is in this subclass, the problem of checking that relation between two finite state transition systems is reducible to the rooted NHORNSAT problem. This leads to a polynomial time algorithm for the problem of checking that relation, provided one can create the instance of the game from the instance of the relational problem in polynomial time. For all the games in Stirling Class, given that the transition systems are represented as finite state systems, the transformation to game instance is polynomial time, provided that the winning positions can be decided in polynomial time. Hence, we get a sufficiency condition as to under what condition a behavioral relation between finite state processes is polynomial time decidable.

A Subclass of Stirling Class We now briefly give a sufficient characterization as to when a game in Stirling Class is reducible to an instance of *rooted* NHORNSAT in polynomial time.

1. R_1 and R_2 are finite and explicitly enumerated. For example, in bisimulation game $R_1 = R_2 = A$, where A is the set of action symbols.
2. The representation of the set of winning positions is either by an explicit listing or is a polynomial time decidable set.

Acknowledgements: We wish to thank Rajeev Alur, S. S. Ravi, and Moshe Vardi for helpful discussions.

References

1. H. R. Andersen. Model checking and boolean graphs. *Theoretical Computer Science*, 126(1):3–30, 1994.
2. G. Ausiello, A. D'Atri, and D. Sacca. Graph algorithms for functional dependency manipulation. *Journal of Association for Computing Machinery*, 30(4):752–766, Oct 1983.
3. G. Ausiello and G. F. Italiano. On-line algorithms for polynomially solvable satisfiability problems. *Journal of Logic Programming*, 10:69–90, 1991.
4. C. Beeri. On the membership problem for functional and multivalued dependencies in relational databases. *ACM Transactions on Database Systems*, 5:241–259, 1980.
5. G. Bhat, R. Cleaveland, and O. Grumberg. Efficient on-the-fly model checking for ctl. In *Proceedings of IEEE Symposium on Logic In Computer Science' 95*, 1995.
6. J. C. Bradfield. *Verifying Temporal Properties of Systems.* Birkhauser, 1992.
7. U. Celikkan and R. Cleaveland. Generating diagnostic information for behavioral preorders. In *Proceedings of Computer Aided Verification: 1992, Lecture Notes in Computer Science 663*, pages 370–383, 1992.
8. R. Cleaveland. Tableau-based model checking in the propositional mu-calculus. *Acta Informatica*, 27:725–747, 1990.
9. R. Cleaveland, M. Klein, and B. Steffen. Faster model checking for modal mu-calculus. In *Proceedings of Computer Aided Verification: 1992, Lecture Notes in Computer Science 663*, pages 410–422, 1992.

10. R. Cleaveland and B. Steffen. Computing behavioural relations, logically. In *ICALP*, pages 127–138, 1991.
11. C. Courcoubetis, M. Y. Vardi, P. Wolper, and M. Yannakakis. Memory efficient algorithms for the verification of temporal properties. *Formal Methods in System Design*, 1:275–288, 1992.
12. 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.
13. J. C. Fernandez and L. Mounier. On the fly verification of behavioral equivalences and preorders. In *The 3rd International Workshop on Computer Aided Verification 1991, Lecture Notes in Computer Science 575*, pages 181–191, 1991.
14. J. F. Groote. Private communications. 1996.
15. D. Kozen. Results on the propositional mu-calculus. *Theoretical Computer Science*, 27, 1983.
16. O. Kupferman, M. Y. Vardi, and P. Wolper. An automata-theoretic approach to branching time model checking. *Draft*, 1995.
17. K. G. Larsen. Proof systems for hennessy milner logic with recursion. In *CAAP'88 Lecture Notes in Computer Science 299*, 1988.
18. K. G. Larsen. Proof systems for satisfiability in hennessy-milner logic with recursion. *Theoretical Computer Science*, 72:265–288, 1990.
19. K. G. Larsen. Efficient local correctness checking. In *CAV 92, Lecture Notes in Computer Science 663*, pages 30–43, 1992.
20. X. Liu. Specification and decomposition in concurrency. Technical report, Department of Mathematics and Computer Science, Aalborg University, Denmark, 1992.
21. Thomas J. Schaefer. The complexity of satisfiability problems. In *Tenth Annual Symposium on Theory of Computing*, 1978.
22. S. K. Shukla, H. B. Hunt III, and D. J. Rosenkrantz. Hornsat, model checking, verification, and games. Research Report TR-95-8, Department of Computer Science, SUNY Albany, 1995.
23. S. K. Shukla, D. J. Rosenkrantz, H. B. Hunt III, and R. E. Stearns. A hornsat based approach to the polynomial time decidability of simulation relations for finite state processes. *DIMACS workshop on Satisfiability Problem: Theory and Practice*, 1996.
24. O. Sokolsky and S. A. Smolka. Incremental model checking in the modal mu-calculus. In *Proceedings of CAV'94*, 1994.
25. C. Stirling and D. Walker. Local model checking in the modal mu-calculus. *Theoretical Computer Science*, 89:161–177, 1991.
26. Colin Stirling. Modal and temporal logics for processes. In *Notes for Summer School in Logic Methods in Concurrency*, pages Department of Computer Science, Aarhus University, 1993.
27. 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.
28. M. Vardi and P. Wolper. An automata theoretic approach to automatic program verification. In *Proceedings of LICS 1986*, pages 332–344, 1986.
29. S. Zhang, O. Sokolsky, and S. A. Smolka. On the parallel complexity of model checking in the modal mu-calculus. In *Proceedings of LICS 1994*, 1994.