UNIVERSITY AT ALBANY
STATE UNIVERSITY OF NEW YORK

Computer Science Department

HORNSAT, Model Checking, Verification, and Games

TR 95-8

S K Shukla, H B HuntIII, D J Rosenkrantz
State University of New York, Albany
sandeep@cs.albany.edu, hunt@cs.albany.edu, djr@cs.albany.edu

HORNSAT, Model Checking, Verification and Games *

(Abstract For Category A)

Sandeep K. Shukla *Harry B. Hunt III* *Daniel J. Rosenkrantz*
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 methodology based upon HORNSAT for model checking and for checking behavioral relations between finite state processes. This methodology has a number of advantages, previously only obtained in different solutions of some of these problems. For example, our methodology can be used to generate diagnostic information [CC92] efficiently. It can be used to do model checking efficiently, for various fragments of modal mu-calculus. It is naturally *local* [SW91, Lar92]; and it can be made to run both *on the fly* [VW86, CVWY92, FM91, BCG95] and *incrementally* [SS94].

Our results show that previous methodologies involving systems of Boolean equations [Lar92, And94] can be simulated by finding maximal and minimal solutions of weakly positive and weakly negative Horn formulas. Since efficient algorithms for finding minimal and maximal satisfying assignments for HORNSAT and its variant NHORNSAT can be easily obtained [DG84, AI91], we use them to develop efficient algorithms for model checking and other verification problems.

We also develop uniform game theoretic formulations of all the relations in the linear time/branching time hierarchy of [vG90]. We define a class of games such that every relation in the hierarchy has its characteristic game in this class. This class which we call *Stirling class*, includes the *bisimulation* game of [Sti93]. We also show that our HORNSAT based methodology implies the existence of such a class of games, since there is a natural game associated with the kind of HORNSAT instances created using our methodology. As a corollary, we obtain natural sufficient conditions on any behavioral relation ρ, for ρ to be polynomial time decidable for finite state transition systems.

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, Results and Related Work

We consider a number of problems related to the verification of finite state systems. These problems include model checking for various logics including alternation free mu calculus and checking behavioral relations between finite state processes. We outline a methodology, for solving these problems, based upon efficient reductions to the satisfiability problems for HORN formulas. The advantages of this methodology include the following:

(i) it yields efficient solutions,
(ii) it yields *local* and *on the fly* solutions naturally,
(iii) it can be used to generate diagnostic information efficiently,
(iv) it can be modified easily to yield incremental solutions,
and (v) data structures and algorithms for the efficient solutions of the required HORN satisfiability problems already exist in the literature [DG84, AI91]. (See Appendices D and E.)

The desirability of (i) through (iv) for verification algorithms has been widely discussed [VW86, BCG95, CC92, FM91, Lar88, Lar90, CVWY92, CS91, And94, SS94, SW91, Cle90]. However, the solutions proposed in the literature have only some of the advantages of (i) through (iv) and have only been applied to some of the verification problems considered here. Our uniform methodology combines all these advantages in the same solution.

Our methodology is based upon efficient reductions of the problems considered to the minimal and maximal satisfiability problems, for weakly positive and weakly negative [Sch78] Horn formulas. We call these satisfiability problems, minimal-HORNSAT and maximal-NHORNSAT respectively. In fact, a restricted form of these Horn formulas is enough for some of the problems. We call this restricted form of HORNSAT and NHORNSAT *rooted* (N)HORNSAT. In Section 2, we outline how our algorithm for model checking for modal mu-calculus is a simplification of the ones in [Lar92, And94]. (Recall that [Lar92] involves *consistent* and *factual* solutions of Boolean equation systems and [And94] involves maximal and minimal fixed points of Boolean equation systems.) We further illustrate our (N)HORNSAT based methodology by outlining our efficient algorithms

(i) to generate diagnostic information for the prebisimulation relation between finite state systems,
and (ii) for model checking alternation free mu calculus.

We discuss why our algorithms are naturally *local* and *on the fly* in the sense of [VW86, Lar88, SW91, FM91, CVWY92, BCG95]; and we observe that they can be modified easily to run *incrementally* in the sense of [SS94]. Moreover, the (N)HORNSAT based algorithms for these problems are as efficient as the algorithms for these problems presented in [FM91, CC92, CS91, And94, SS94, Lar88, SW91].

Consequently, 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, (we discuss such algorithms in the appendix), which has the option to run *on the fly* for space efficiency, and an option to run *incrementally* (e.g., using simple modifications of the incremental HORNSAT algorithms given in [AI91]). The fact that efficient solutions for HORNSAT and its variants are already existent in the literature [DG84, AI91] 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 are already existent in the literature on HORNSAT. Our results show that many model checking algorithms in the literature invented complex data structures where as existing efficient data structures for solving variants of HORNSAT are sufficient to obtain the same efficiency. Moreover, this leads to modular design, because the efficient implementation of HORNSAT solver can be delegated to a different designer.

However, in [KVW95] a data structure for a linear time algorithm of functional dependency in relational databases [Bee80] was reused to obtain model checking algorithm for CTL. It is interesting to note that functional dependency and HORNSAT are interreducible and in [AI91, ADS83] one can find that same kinds of data structures are needed to solve them in linear time. [1] In [And94] the model checking problem for mu-calculus was reduced to solving fix points of Boolean equations and for efficiency, complex graph based data structures were

[1] 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 PROLOG.

invented. Our results show that the full power of boolean equations are not needed to solve these problems. In fact, a kind of implication based methodology embodied in (N)HORNSAT is enough.

Not only from implementational point of view, but also theoretically, our results bring out the underlying principles of a number of verification methodology. Understanding the different methodologies from an unifying framework has been emphasized also in [KVW95].

Lastly extending a result in [Sti93], we define a general class of games and show that this class includes the *characteristic games*, for each of the behavioral relations in the linear time /branching time hierarchy [vG90]. We also show that the existence of such a class of characteristic games also follows from our *rooted* NHORNSAT based methodology for behavioral relations since,

(i) we show that there is a natural two player game associated with rooted NHORNSAT problem, and
(ii) we show that all the relations in linear time/branching time hierarchy [vG90] can be reduced to rooted NHORNSAT.

Then we recall from [SRHS96] that many polynomial time decidable relations in [vG90] are reducible to rooted NHORNSAT in polynomial time. We also observe that existence of winning strategies in the games corresponding to these relations are also reducible to the NHORNSAT problem. A generalization of these observations leads to a sufficiency condition for these relations to be polynomial time decidable for finite state systems.

1.2 Organization of the Paper

We assume that the reader is familiar with the basic definitions of various behavioral relations in linear time/branching time hierarchy [vG90]. (Otherwise, Appendix A has some of the relevant definitions.) Section 2 defines basic notions of (N)HORNSAT and discusses its relationship with Boolean equation based methodologies in [Lar92, And94]. Sketches of efficient data structures and algorithms to solve the satisfiability problem of (N)HORNSAT are given in the Appendix D and E. In Section 3, we discuss the desirability of *on the fly*, *local* and *incremental* properties of verification algorithms and point out that these properties come naturally in our (N)HORNSAT based methodology. Section 4, discusses the use of our methodology to obtain diagnostic information efficiently. Relevant definitions such as definitions of intuitionistic Hennessy-Milner logic [CC92] and prebisimulation are provided in Appendix B. In Section 5, we develop algorithms for model checking alternation free mu calculus and Hennessy-Milner logic with recursion using (N)HORNSAT based method. Definitions related to modal mu-calculus and alternation free mu-calculus are provided in the Appendix C. Section 6, defines *Stirling class* as a class of games and gives a sufficient condition for polynomial time decidability of behavioral relations between finite state processes.

Note that due to lack of space, no correctness proofs have been included.

2 Satisfiability Problem for (N)HORNSAT

Although most of the definitions related to behavioral relations, transition systems etc., are given in the appendix, definitions of satisfiability problems for HORNSAT and its variants are discussed below. We include these definitions in the main text because we have reduced all our problems to (N)HORNSAT and hence it is crucial. Some of the other relevant definitions related to mu-calculus and model checking are in the corresponding sections and appendices as appropriate.

Let us consider an instance of a propositional CNF satisfiability problem, which is a conjunction of clauses and each clause is a disjunction of positive and negative literals with either of the following two restrictions.

1. Each clause is a disjunction of literals with at most one negative literal.
2. Each clause is a disjunction of literals with at most one positive literal.

When the first restriction applies, we call the problem NHORNSAT, and when the second restriction applies, we call it HORNSAT [DG84]. 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 the corresponding 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 pointwise extension of the ordering, and a pointwise $\wedge$ and $\vee$ as *meet* and *join* operation, we get a complete lattice. Now, for an instance of a satisfiability problem h, let us denote the set of all satisfying assignment 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 $\{0,1\}^n$. So now we have the following two problems:

1. Problem maximal-NHORNSAT: Given an NHORNSAT instance, find a maximal satisfying assignment, if there is one. Other wise, conclude that the instance is not satisfiable.
2. Problem minimal-HORNSAT: Given a HORNSAT instance, find a minimal satisfying assignment, if there is one. Otherwise, conclude that the instance is not satisfiable.

A linear time algorithm for minimal-HORNSAT appears in [DG84](See Appendix D). It follows that maximal-NHORNSAT is also solvable in linear time (See Appendix E).

In some of our applications we have a special type of HORNSAT or NHORNSAT instances. Here we discuss that special type of NHORNSAT called *rooted NHORNSAT* and the corresponding cases and algorithms for HORNSAT is very similar.

Definition 2.1 *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 the 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 $tail(C_k) = true$. Similarly, for a clause of the form $C_k = x_j$, $head(C_k) = true$ and $tail(C_k) = x_j$.

Definition 2.2 *A instance of a* rooted NHORNSAT *problem is of the form (X, C, x_1) where (X, C) is an NHORNSAT instance and $x_1 \in X$ is such that $C_1 = x_1$ (containing a single positive literal). And the clauses in C are ordered in such a way that if $head(C_k) = x_j$ then, there must be a clause $C_l (l < k)$ preceding it, such that x_j is a disjunct in $tail(C_l)$. Also for a single literal clause $C_k = x_p$ $(k > 1)$, x_p must also be a disjunct in $tail(C_l)$ for some $l < k$. and x_p cannot be head of any clause. Note that if C_2 is an implication clause then $head(C_2) = x_1$.*

The correctness of our (N)HORNSAT based methodology for model checking can be easily demonstrated by showing the following. The (N)HORNSAT based methodology may replace the methodologies in [Lar92, And94] based upon systems of simple Boolean equations. The advantage of this lies in the fact that efficient algorithms and data structures for (N)HORNSAT are already available in the literature [DG84, AI91]. The soundness and completeness of our methodology easily follow from the following theorem and its extensions to the results in [And94]..

Theorem 2.3 *The* factuality problem *and the* consistency problem *of system of* simple *Boolean equations described in [Lar92] can be efficiently reduced to the minimal-HORNSAT and maximal-NHORNSAT problems respectively.*

The reason why this theorem holds is as follows: Given a system of *simple* Boolean equations, if we are interested in factuality [Lar92], 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. A dualization of this will show that the consistency problem of [Lar92] can be reduced efficiently to the maximal-NHORNSAT problem.

Similarly, the problems of finding the least and greatest fixed points of the Boolean equations of [And94] 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

In this section, we assume that the reader is familiar with modal mu-calculus and the ideas of model checking. Otherwise, brief discussions may be found in Section 5.

We discuss here the advantages of *local*, *on the fly* and *incremental* algorithms for model checking and checking behavioral relations. We also sketch how our (N)HORNSAT based methodology achieves these desirable goals.

Local Model Checking : The idea of finite state model checking is to decide if a given state of a finite state system satisfies a given specification. The specification is expressed in a suitable logic, such as modal mu-calculus. However, the original algorithms [EL86] for finite state model checking were "global" in the following sense. These *global* algorithms use a fix point approximation technique for computing sets of states which satisfy a fixpoint formula. However, in many cases, this involves many unnecessary computations as discussed in [Lar90, SW91, Bra92]. Hence, in [SW91] a tableau based algorithm for model checking was introduced. They appealed to an implicit fixpoint induction rather than iterative approximation and our (N)HORNSAT construction also appeals to an implicit fixpoint induction. 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 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 [Lar90, SW91, Bra92] 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, such that s satisfies the property expressed in ϕ if and only if Y_s^ϕ is *true* in the satisfying assignment of the maximal/minimal (N)HORNSAT instance. If Φ is a maximal fixpoint formula, then we generate a maximal-NHORNSAT instance, if Φ is a minimal fix point formula, we generate a minimal-HORNSAT instance, in case of nested fix points it is more subtle. However, in the next sections, it will be clear, that we explore only those states which are necessary to be explored in our method. Hence, our model checking algorithms are as local as any other local model checking algorithm in the literature.

On the Fly Model Checking : Traditional model checking algorithms such as [EL86] require the whole state space to be constructed in memory before they can be applied. However, in many applications, one can find counter examples much before exploring the whole state space. More over, when the finite state systems are described succinctly, for example, using parallel composition operator, the actual state space may be exponentially large and hence, it might be impractical to construct the whole state space in memory. As a result in [VW86, CVWY92, BCG95, KVW95, FM91] *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 [AI91] 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 [FM91] 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. Details of our on line algorithm for (N)HORNSAT and another depth first search based on the fly algorithm for rooted (N)HORNSAT will be presented in a fuller version of this abstract.

Incremental Model Checking : In [SS94], an incremental algorithm for model checking alternation free mu-calculus was developed. The basic idea behind the algorithm was as follows: Suppose, a model checking algorithm is run on a transition system and a formula and the information regarding satisfiability of subformulas at different states are available. Now, suppose, there are some changes in the specification of the transition system, so that some new transitions are added and some transitions are 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 [SS94] that in the worst case, this may not be possible. One can construct examples, such that one has to spend as much time in the incremental algorithm as required in the model checking from the scratch. However, in the best case and more importantly, in many pragmatic situations the incremental computation could be linear in the size of the modification. It can be shown that minor modifications of the online algorithm for HORNSAT [AI91] will give an incremental algorithm for (N)HORNSAT. Hence, with addition and deletion of clauses, the amortized time in incrementally solving the modified (N)HORNSAT problem will be linear in the size of the modification. Since, 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 including the behavioral relation checking and model checking which have amortized time complexity linear in the size of the modifications.

4 Generating Diagnostic Information Efficiently via HORNSAT

We briefly outline how we can efficiently generate diagnostic information in the sense of [CC92] when two transition systems are not related by a behavioral relation. In [CC92], a method was presented to generate a formula of *intuitionistic Henessy Milner logic* which distinguishes the two processes in the sense of [Wal88]. In the first subsection we explain the problem following [CC92] and in the next subsection we review the HORNSAT based method for checking behavioral relations between finite state processes presented in [SRHS96]. Then we extend this method to generate diagnostic information much more efficiently than the methods presented in [CC92]. Although our method is general enough to apply to all the preorders and equivalences considered in [SRHS96], we illustrate our technique only on *prebisimulation preorder* which was also the illustrative example used in [CC92].

4.1 Prebisimulation and Intuitionistic Henessy Milner Logic

Definition 4.1 *[CC92, Wal88] An* **Extended Labelled Transition System(elts)** *is a quadruple* $\langle S, Act, \rightarrow, \uparrow\rangle$ *where* $\langle S.Act, \rightarrow\rangle$ *is a labelled transition system and* $\uparrow \subseteq S \times Act$ *is an underdefinedness relation.*

The relation $\uparrow$ represents *underdefinedness.* If $(p, a) \in \uparrow$ then the behavior of p in response to action a is not completely specified. Other $a-$transitions may be added later.

A process is a pair (T, s) where T is an extended transition system and s is a state in that transition system. **notation:** We use $p \uparrow a$ in place of $(p, a) \in \uparrow$ and $p \downarrow a$ in place of $\neg(p \uparrow a)$.

Given a labelled transition system(*lts*), one can easily generate the corresponding extended transition system (*elts*). Moreover, given a finite *lts*, one can use a transitive closure algorithm in polynomial time to generate the corresponding *elts*. The size of the generated *elts* is linear in the size of the *lts* [CC92]. Hence, for simplicity we assume henceforth that we are given elts for checking behavioral relations.

In Appendix B we define a particular relation on *elts* called **prebisimulation** preorder and it can be easily seen [CC92] that many other simulation relations and equivalences can be shown as special cases of this preorder. We denote the preorder by $\sqsubseteq$. If two processes $\mathcal{P}$ and $\mathcal{Q}$ are in the prebisimulation preorder, we write $\mathcal{P} \sqsubseteq \mathcal{Q}$. We also define Intuitionistic Hennessy-Milner Logic (IHML) which is a characteristic logic for prebisimulation in Appendix B, following [Sti87].

The logical characterization of prebisimulation preorder [Sti87] says that if $\mathcal{P} \sqsubseteq \mathcal{Q}$ then the set of IHML formulas satisfied by $\mathcal{P}$ is a subset of the set of IHML formulas satisfied by $\mathcal{Q}$.

Definition 4.2 *We call an IHML formula* Φ *a* **diagnostic formula** *for two processes* $\mathcal{P} = (\langle P, Act, \rightarrow, \uparrow\rangle, p_0)$ *and* $\mathcal{Q} = (\langle Q, Act, \rightarrow, \uparrow\rangle, q_0)$*, if* $\mathcal{P} \models \Phi$ *but* $\mathcal{Q} \not\models \Phi$*. (In which case we say that* Φ *distinguishes* $\mathcal{P}$ *from* $\mathcal{Q}$*.)*

So given two processes, $\mathcal{P} = (\langle P, Act, \rightarrow, \uparrow\rangle, p_0)$ and $\mathcal{Q} = (\langle Q, Act, \rightarrow, \uparrow\rangle, q_0)$, we are interested in constructing a diagnostic formula for them if $\mathcal{P} \not\sqsubseteq \mathcal{Q}$. In [CC92] an algorithm for this problem is presented. The complexity of their algorithm is quite high. Although in [CS91] an efficient algorithm for computing behavioral relations via model checking is presented, the method in [CC92] for producing diagnostic traces does not trivially apply to that algorithm.

4.2 HORNSAT based checking of Prebisimulation Preorder

We now recall from [SRHS96], how to reduce the prebisimulation problem to *rooted* NHORNSAT.

Given two finite state processes $\mathcal{P} = (\langle P, Act, \rightarrow, \uparrow\rangle, p_0)$ and $\mathcal{Q} = (\langle Q, Act, \rightarrow, \uparrow\rangle, q_0)$, we outline an algorithm for checking if $\mathcal{P} \sqsubseteq \mathcal{Q}$. We give an efficient reduction to an NHORNSAT instance (see appendix) and since there is linear time algorithm to check the satisfiability of NHORNSAT, that gives us an efficient algorithm for prebisimulation preorder checking.

Our reduction of the prebisimulation problem to an NHORNSAT instance f is 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 (p, q) is required to be in the prebisimulation relation we construct this type of clause. We always create X_{p_0,q_0} because (p_0, q_0) requires to be in the prebisimulation relation for the relation to hold between the two transition systems. Also, if there is a pair (p, q) such that p, q have no out going transition and $p \uparrow a$ for all $a \in Act$, or if p, q have no out going transition and $p \downarrow a$ as well as $q \downarrow a$ for all $a \in Act$, then we generate a clause $X_{p,q}$.
   (b) A single negated literal $\overline{X_{p,q}}$. Such a clause is constructed to indicate that (p, q) cannot be in any prebisimulation relation. We create such a clause when one of the followings is true:
      i. When there is an $a \in Act$ such that $p \xrightarrow{a} p'$ for some p' but there is no q' such that $q \xrightarrow{a} q'$. We mark such a clause with a **clause number and** $< a > true$ to indicate that q does not have an a action whereas p has one.
      ii. For some $a \in Act$, $p \downarrow a$ but $q \uparrow a$. We mark this clause with the **clause number and** $[a]_\downarrow true$ to denote that p satisfies $[a]_\downarrow true$ but q does not.
      iii. For some $a \in Act$, $p \downarrow a$ and $q \downarrow a$ but there is q' such that $q \xrightarrow{a} q'$ but there is no p' such that $p \xrightarrow{a} p'$. Mark this clause with a **clause number and** $[a]_\downarrow false$ to denote that $[a]_\downarrow false$ is satisfied by p but not by q. (Note that $[a]_\downarrow false$ can be satisfied by a state p if and only if there is no a action out of it and $p \downarrow a$).
   (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 prebisimulation relation one of the (i, j)'s must also be in the prebisimulation relation. We generate these clauses in the following cases:
      i. For each action $a \in Act$, for each transition $p \xrightarrow{a} p'$. in $\mathcal{P}$, we create an implication clause in the following manner:
      Let $S(q, a) = \{q_i \mid q \xrightarrow{a} q_i\}$. Then we generate a clause $X_{p,q} \Rightarrow \bigvee_{r \in S(q,a)} X_{p',r}$ and mark this clause with a **clause number and "**$a1$**"** to denote that it corresponds to obligation that an a action in the first process needs to be matched by a similar one in the second process.
      ii. If for all $a \in Act$, both $p \downarrow a$ and $q \downarrow a$, then for each $a \in Act$, for each transition $q \xrightarrow{a} q'$. in $\mathcal{Q}$, we create an implication clause in the following manner:
      Let $S(p, a) = \{p_i \mid p \xrightarrow{a} p_i\}$. Then we generate a clause $X_{p,q} \Rightarrow \bigvee_{r \in S(p,a)} X_{r,q'}$ and mark this clause with a **clause number and "**$a2$**"** to denote that it corresponds to the obligation that an a action in the second process needs to be matched by a similar one in the first.

Now we state without a proof a theorem that states that the NHORNSAT instance produced by the above sketched algorithm is satisfiable if and only if $\mathcal{P} \sqsubseteq \mathcal{Q}$. Also the size of the NHORNSAT instance is $O(|\mathcal{P}| * |\mathcal{Q}|)$ and hence a linear time NHORNSAT solver based on [DG84] (see appendix) combined with the above reduction will give as efficient an algorithm for prebisimulation checking.

Theorem 4.3 *the NHORNSAT instance produced by the algorithm described above, is satisfiable if and only if* $\mathcal{P} \sqsubseteq \mathcal{Q}$. *The running time of the prebisimulation checking algorithm obtained this way is* $O(|\mathcal{P}| * |\mathcal{Q}|)$.

4.3 Generating Diagnostic Formula

Now we show how to obtain diagnostic IHML formula without increasing the asymptotic complexity, in case $\mathcal{P} \not\sqsubseteq \mathcal{Q}$.

Recall that the linear time algorithm for HORNSAT satisfiability presented in [DG84] first builds a graph representation of the instance and then do pebbling on the graph. The on the fly algorithm for HORNSAT in [AI91] builds this graph incrementally on demand basis. In the appendix, we have outlined, how to adapt this pebbling to NHORNSAT. This pebbling helps us to generate the diagnostic IHML formula without any extra overhead.

Note that when the two systems are not related by prebisimulation preorder, there is a pebbling from $\{false\}$ to $true$ in the graph. (See appendix). Now, since (p_0, q_0) are not related, there is a pebbling through X_{p_0,q_0}. We can find such a pebbling in linear time and once such a pebbling has been found we construct the IHML formula as follows:

$\triangleright$ Recall we marked each clause with a clause number and a certain fixed size information whose size depends on the size of Act. During the graph building while solving the NHORNSAT instance, we labelled the edges of the graph by clause numbers as well as the markings of the clauses.

The distinguishing formula for $(p, q) \in P \times Q$ is obtained by the following rules: We are assuming that the NHORNSAT instance produced by the algorithm in the previous section is unsatisfiable and thus there is a pebbling from $\{false\}$ to $true$ through X_{p_0,q_0}.

$\triangleright$ The distinguishing formula is not represented directly but as a set of equations in variables of the form $d_{p,q}$ such that the value of the variable d_{p_0,q_0}, if the equations are solved, will give the distinguishing formula for the two transition systems. The value of $d_{p,q}$ gives the distinguishing formula for the states p, q[2].

1. If there is an edge directly from the node $false$ in the graph to $X_{p,q}$ (which also means that there is a clause $C_i = \overline{X_{p,q}}$.) Obviously that edge must be marked by (i, m) where m is either of the following three symbols. $< a > true$, $[a]_\downarrow true$ and $[a]_\downarrow false$. Then create the equation $d_{p,q} = m$.
2. Let $X_{p,q}$ be pebbled and the pebbling is via edges marked with $(i, a1)$ for some clause C_i, for some $a \in Act$, and the clause C_i is of the form $X_{p,q} \Rightarrow \bigvee_{r \in S(q,a)} X_{p',r}$ where $S(q, a) = \{q_i \mid q \xrightarrow{a} q_i\}$. Then create a new equation $d_{p,q} =< a > (\bigwedge_{r \in S(q,a)} d_{p',r})$.
3. Let $X_{p,q}$ be pebbled via edges marked with $(i, a2)$ for some clause C_i, for some $a \in Act$, and the clause C_i is of the form $X_{p,q} \Rightarrow \bigvee_{r \in S(p,a)} X_{r,q'}$ where $S(p, a) = \{p_i \mid p \xrightarrow{a} p_i\}$. Then create a new equation $d_{p,q} = [a]_\downarrow(\bigvee_{r \in S(q,a)} d_{r,q'})$.

The following theorem states the correctness of the above method and the proof is by induction on the pebbling distance [DG84] which is omitted due to lack of space.

Theorem 4.4 *The method outlined above produces a set of propositional equations with propositional variables of the form $d_{p,q}$ such that the value of d_{p_0,q_0} (when the substitutions are made according to the equation set,) is a diagnostic IHML formula for the processes $\mathcal{P}$ and $\mathcal{Q}$ when $\mathcal{P} \not\sqsubseteq \mathcal{Q}$.*

Moreover, the size of this equational representation of the diagnostic formula is at the worst $O(|P| * |Q| * max(|P|, |Q|))$

However, it is easy to see that the size of the NHORNSAT instance will be $O(|P| * |Q| * (max(|P|, |Q|))^2)$ and using the linear time implementation of the pebbling and writing the distinguishing formula during the pebbling itself will provide an algorithm that runs in $O(|P| * |Q| * (max(|P|, |Q|))^2)$ time which is $O(|\mathcal{P}| * |\mathcal{Q}|)$ [3]and which has the property that it decides if $\mathcal{P} \sqsubseteq \mathcal{Q}$ and in case $\mathcal{P} \not\sqsubseteq \mathcal{Q}$, it produces an equational representation of a diagnostic IHML formula without any extra cost in the asymptotic complexity.

We also generate diagnostic traces for model checking in a very similar way by marking the clauses with special information of fixed size. The details of diagnostic information generation for model checking is not discussed in this abstract.

5 Model Checking Fragments of Modal Mu-Calculus

Although our methodology can be extended to apply to general Mu-Calculus [Koz83, Bra92], we illustrate our methods through two well discussed fragments of modal mu-calculus. One is the *unnested single fixed point* fragment. This is similar to the Hennessy-Milner Logic with recursion [Lar88, Lar90]. The other is alternation-free mu-calculus, as discussed in [CS91]. We recall their definitions from [CS91] in Appendix C.

5.1 Model Checking to (N)HORNSAT

Now we illustrate how to reduce the model checking problem for the above mentioned fragments of modal mu-calculus to (N)HORNSAT.

[2]We can construct examples of transition systems $\mathcal{P}$ and $\mathcal{Q}$ such that $\mathcal{P} \not\sqsubseteq \mathcal{Q}$ and the distinguishing IHML formula is exponential in the size of the description of $\mathcal{P}$ and $\mathcal{Q}$. This lower bound on the distinguishing formula size justifies our use of the equational representation for the distinguishing formula.

[3]Note that $|\mathcal{P}| = O(|P|^2)$

5.1.1 Reduction of Model Checking 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 $s^* \in \mathcal{S}$. (See Appendix C for clarification on the notations). 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 [CS91] 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 5.1 *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.

5.1.2 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 [CS91]. 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 could also be obtained by noting the fact that crucial data structuring [DG84] gives 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.

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. (Dual holds for 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 5.2 *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 [CS91].*

We also have developed HORNSAT based methods to capture the tableau based local model checking in [Cle90] and [SW91]. Details will appear in a future version of this paper.

In the next section, we go on to see that there is a game associated to rooted HORNSAT and in fact, that implies existence of the characteristic games for all the behavioral relations in the linear time /branching time hierarchy[vG90].

6 Game for rooted (N)HORNSAT and Stirling Games

We now describe a game for *rooted* NHORNSAT. We showed in Section 4, how to reduce the problem of checking prebisimulation relation between finite state processes to *rooted* NHORNSAT. Note that in [CS91, Ste89], it was shown that most equivalences and preorders including failure equivalence, trace equivalence etc., are reducible to prebisimulation problem(via process transformation). Hence, all those relations that are reducible to a prebisimulation problem are also reducible to rooted NHORNSAT. Given a two-player game for *rooted* NHORNSAT, we can easily associate games to all these relations as well. Now recall that Colin Stirling in [Sti93] defined a characteristic game for bisimulation. Our results show that such a game formulation is very natural given the game for rooted NHORNSAT.

However, we develop here a different class of games such that all relational preorders and equivalences between finite state processes in linear time /branching time hierarchy [vG90] have their characteristic games in this class. In particular Stirling's bisimulation game is one game in our class. Henceforth we call our class of games **Stirling Class** of games.

[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.

6.1 Game for *rooted* NHORNSAT

Recall the definition of *rooted HORNSAT* from Section 2. Game for an instance of a rooted NHORNSAT instance $h = (X, C, x_1)$ is a two player game G_h as follows:
Player I is called a *spoiler* who wants to show that the instance h is not satisfiable and Player II is called a *duplicator* who wants to show that the instance h is satisfiable. The game proceeds in rounds. In the first round, the spoiler opens the game by choosing a clause C_i such that $head(C_i) = x_1$. Duplicator has to reciprocate by choosing x_{ij} such that x_{ij} is a disjunct in $tail(C_i)$. In the subsequent rounds, the spoiler chooses a clause C_k such that $head(C_k) = x_{ij}$ where x_{ij} was chosen by the duplicator 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.

It is easy to prove the following theorem by recalling the pebbling based algorithm for solving NHORNSAT discussed in the Appendix E.

Theorem 6.1 *Given an instance $h = (X, C, x_1)$ of the* rooted *NHORNSAT problem, the duplicator has a winning strategy* [5] *in the corresponding game if and only if h is satisfiable.*

6.2 Stirling Class of Games

Now we are ready to describe *Stirling Class* of games. Each game in this class also has two players. One player is called *duplicator* or *prover* and the other is called *spoiler* or *disprover*. Each game in the class has the following components:

1. Two Finite Transition systems $T_1 = \langle S_1, A, \rightarrow_1, s_1\rangle$ and $T_2 = \langle S_2, A, \rightarrow_2, s_2\rangle$.
2. Two languages $R_1 \subseteq A^*$ and $R_2 \subseteq A^*$.
3. Two total relations $m_2 \subseteq R_1 \times A^*$ and $m_2 \subseteq R_2 \times A^*$.
4. A set of (*winning positions*) $\Gamma \subseteq S_1 \times S_2$.
5. A set of starting positions $\Sigma \subseteq \Gamma \subseteq S_1 \times S_2$.
6. 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.
7. 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 $\langle s^1_0, s^2_0\rangle, ..., \langle s^1_i, s^2_i\rangle, ...$ 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* $\langle s^1_0, s^2_0\rangle, ..., \langle s^1_j, s^2_j\rangle$. The next pair $\langle s^1_{j+1}, s^2_{j+1}\rangle$ 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 $s^i_j \stackrel{x}{\Longrightarrow}_i u$. and $u = s^i_{j+1}$. (Note that $\Longrightarrow_i$ denotes an extended step in the transition system T_i).

- 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 $s^{i'}_j \stackrel{y}{\Longrightarrow}_{i'} u'$ and $u' = s^{i'}_{j+1}$.

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. Each round consists of two moves. The first move of each round is a move by the spoiler and the second move is by the duplicator. If in a round, after the spoiler has made its move, the duplicator can also make a move according to the moves described above, then we say that the duplicator has a *matching move* in that round.

Now we describe the winning conditions of a game of this kind. The game may continue until one of the players wins. The winning conditions for each players is as follows.

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

- **Duplicator wins:**
  1. The play is $\langle s_0^1, s_0^2\rangle, ..., \langle s_n^1, s_n^2\rangle$ and there is no available transitions from s_n^i and $M = \{i\}$. In case $M = \{1, 2\}$, the corresponding condition is that s_n^1 and s_n^2 both have no available transitions.
  2. A partial play is $\langle s_0^1, s_0^2\rangle, ..., \langle s_n^1, s_n^2\rangle$ and for some $i < n$, $s_i^1 = s_n^1$ and $s_i^2 = s_n^2$.
- **Spoiler wins:**
  1. The play is $\langle s_0^1, s_0^2\rangle, ..., \langle s_n^1, s_n^2\rangle$ and $\langle s_n^1, s_n^2\rangle \notin \Gamma$.
  2. $M = \{1, 2\}$ and a partial play is $\langle s_0^1, s_0^2\rangle, ..., \langle s_n^1, s_n^2\rangle$. For some $i \in M$, $\exists x \in R_i.(\exists s^i \in S_i.(s_n^i \xRightarrow{x}_i s^i)$ but for $i' \in M - \{i\}$, forall y such that $((x, y) \in m_{i'}$ and for all $s_{i'} \in S_{i'}$, there is no transition $s_n^{i'} \xRightarrow{y}_{i'} s_{i'}$.
  3. $M = \{i\}(i \in \{1, 2\})$ and a partial play is $\langle s_0^1, s_0^2\rangle, ..., \langle s_n^1, s_n^2\rangle$, $\exists x \in R_i.(\exists s^i \in S_i.(s_n^i \xRightarrow{x}_i s^i)$ but for $i' \in M - \{i\}$, forall y such that $((x, y) \in m_{i'}$ and for all $s_{i'} \in S_{i'}$, there is no transition $s_n^{i'} \xRightarrow{y}_{i'} s_{i'}$.

So 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.

6.3 Simulations and Equivalences as games in the Stirling Class

We now define what is meant by a *characteristic game* for a particular relation or equivalence relation between finite state processes.

Definition 6.2 *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.*

Now we illustrate the characteristic games for the following relations between finite state transition systems.

1. Bisimulation Game($Bsim - game$)
2. Weak Bisimulation Game($Weakbsim - game$)
3. Simulation Game($Sim - game$)
4. Ready Simulation Game($Rsim - game$)
5. Failure Equivalence Game($Failure - game$)

In the Appendix F we have discussed the characteristic games for some other relations such as Forward Simulation Game($Fsim - game$), Trace Equivalence Game($Trace - game$), Readiness Equivalence Game($Readiness - game$), and the 2-nested Equivalence Game ($2 - nested - game$)

Many other equivalences considered in the literature may be shown to have a characteristic game in the Stirling Class. Here, we list the restrictions on various parameters of the Stirling class of games which make the games a characteristic game for the particular relations. Note that ι denotes the identity relation in the subsequent paragraphs. We also assume in the following that all the games are being played on $T_1 = \langle S_1, A, \rightarrow_1, s_1\rangle$ and $T_2 = \langle S_2, A, \rightarrow_2, s_2\rangle$.

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

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

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

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 6.3 *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 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 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. So far as we know, this is the first time such a sufficient characterization of polynomial time decidable behavioral relations between finite state transition systems is given.

This is useful, because, when ever a new relation is defined, if that relation satisfies this set of conditions, it is guaranteed that the relation is polynomial time decidable for finite state transition systems.

6.3.1 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 position is either by an explicit listing or such that determining if a position of the game is a winning position is polynomial time decidable.

As a corollary we get the following result:

Theorem 6.4 *Given a game G in Stirling class satisfying the conditions listed above, whether the duplicator has a winning strategy for G, can be decided in polynomial time.*

Hence the corollary is :

Corollary 6.4.1 *Any behavioral relation between two finite state transition systems, whose characteristic game satisfy the conditions listed above, is decidable in polynomial time. Hence, strong and weak bisimulation equivalence, forward simulation, simulation equivalence, ready simulation and equivalence, prebisimulation, k-nested simulation for any fixed k are all polynomial time decidable relations for finite state transition systems.*

Acknowledgements: We thank Moshe Vardi, Pierre Wolper and Rajeev Alur for helpful communications. We thank S. S. Ravi for helpful discussions.

References

[ADS83] 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.

[AI91] G. Ausiello and G. F. Italiano. On-line algorithms for polynomially solvable satisfiability problems. *Journal of Logic Programming*, 10:69–90, 1991.

[And94] H. R. Andersen. Model checking and boolean graphs. *Theoretical Computer Science*, 126(1):3–30, 1994.

[BCG95] 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.

[Bee80] C. Beeri. On the membership problem for functional and multivalued dependencies in relational databases. *ACM Transactions on Database Systems*, 5:241–259, 1980.

[Bra92] J. C. Bradfield. *Verifying Temporal Properties of Systems.* Birkhauser, 1992.

[CC92] U. Celikkan and R. Cleaveland. Generating diagnostic information for behavioral preorders. In *Proceedings of Computer Aided Verification: 1992, LNCS 663*, pages 370–383, 1992.

[Cle90] R. Cleaveland. Tableau-based model checking in the propositional mu-calculus. *Acta Informatica*, 27:725–747, 1990.

[CS91] R. Cleveland and B. Steffen. Computing behavioural relations, logically. In *ICALP*, pages 127–138, 1991.

[CVWY92] 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.

[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.

[EL86] E. A. Emerson and C. L. Lei. Efficient model checking in fragments of the propositional modal mu-calculus. In *Proceedings of LICS 1986*, pages 267–278, 1986.

[FM91] 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, LNCS 575*, pages 181–191, 1991.

[GV92] J.F. Groote and F.W. Vaandrager. Structured operational semantics and bisimulation as a congruence. *Information and Computation*, 100(2):202–260, Oct 1992.

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

[Koz83] D. Kozen. Results on the propositional mu-calculus. *Theoretical Computer Science*, 27, 1983.

[KVW95] O. Kupferman, M. Y. Vardi, and P. Wolper. An automata-theoretic approach to branching time model checking. *Draft*, 1995.

[Lar88] K. G. Larsen. Proof systems for hennessy milner logic with recursion. In *CAAP'88 LNCS 299*, 1988.

[Lar90] K. G. Larsen. Proof systems for satisfiability in hennessy-milner logic with recursion. *Theoretical Computer Science*, 72:265–288, 1990.

[Lar92] K. G. Larsen. Efficient local correctness checking. In *CAV 92, LNCS 663*, pages 30–43, 1992.

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

[SRHS96] 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. *To be presented at: DIMACS workshop on Satisfiability Problem: Theory and Practice*, 1996.

[SS94] O. Sokolsky and S. A. Smolka. Incremental model checking in the modal mu-calculus. In *Proceedings of CAV'94*, 1994.

[Ste89] B. U. Steffen. Characteristic formulae for ccs with divergence. In *Proceedings of ICALP 89, LNCS 372*, pages 723–733, 1989.

[Sti87] C. Stirling. Modal logics for communicating systems. *Theoretical Computer Science*, 49:311–347, 1987.

[Sti93] 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.

[SW91] C. Stirling and D. Walker. Local model checking in the modal mu-calculus. *Theoretical Computer Science*, 89:161–177, 1991.

[Tar55] A. Tarski. A lattice theoretic fixpoint theorem and its applications. *Pacific Journal of Mathematics*, 5, 1955.

[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.

[VW86] M. Vardi and P. Wolper. An automata theoretic approach to automatic program verification. In *Proceedings of LICS 1986*, pages 332–344, 1986.

[Wal88] D. Walker. Bisimulation and divergence. In *Proceedings of the Third Annual Symposium on Logic in Computer Science*, pages 186–192, 1988.

Appendix

A Transition Systems, Simulations and Equivalences

In this section we define transition systems, some of the equivalences in the linear time/branching time hierarchy [vG90], and a few simulation relations between transition systems. For an exhaustive study of these relations reader is referred to [vG90].

Definition A.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 $\mathcal{T}$.*

A transition system may be represented graphically by an edge labelled directed multi graph with a special vertex denoting the starting state.

In some discussions, a transition system $\mathcal{T}$ is represented by $\langle \mathcal{S}, Act, \rightarrow \rangle$, where $\mathcal{S}$ is the set of states, *Act* is the set of action symbols and $\rightarrow$ is the transition relation. However, we discuss the representation of a transition system in the relevant section to avoid confusion.

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

Definition A.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 $init(p) = \{a \in Act \mid \exists t \in S((p, a, t) \in D)\}$.*

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 A.3 *Let $R \subseteq S \times T$ be a binary relation between S and T. R is a* **simulation** *if*

$$\forall (s,t) \in R \quad (\forall a \in Act, \forall s' \in S \quad ((s, a, s') \in D_1 \Rightarrow (\exists t' \in T \ ((t, a, t') \in D_2 \wedge (s', t') \in R)))).$$

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 A.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 $(s_1, t_1) \in R$. $\mathcal{T}_1$ 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 $\mathcal{T}_2 \preceq_{rsim} \mathcal{T}_1$.*

Definition A.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. $\forall (r,s) \in B, w \in (Act - \{\tau\})^* :$
if $\exists \gamma : \hat{\gamma} = w \wedge r \stackrel{\gamma}{\Rightarrow} r'$ *then* $\exists s' \exists \beta : \hat{\beta} = w \wedge s \stackrel{\beta}{\Rightarrow} s' \wedge (r', s') \in B$.
and if $\exists \beta : \hat{\beta} = w \wedge s \stackrel{\beta}{\Rightarrow} s'$ *then* $\exists r' \exists \gamma : \hat{\gamma} = w \wedge r \stackrel{\gamma}{\Rightarrow} r' \wedge (r', s') \in B$.

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 A.6 *Let* $\mathcal{T}_1 = \langle S, D_1, s_1 \rangle$ *and* $T_2 = \langle T, D_2, t_1 \rangle$ *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** *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 that 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 that by* $(\mathcal{T}_1 \sim_{trace} \mathcal{T}_2)$.

Definition A.7 *The* failure set *of a state* s *in a transition system* T*, denoted by* $Failures(s)$*, is defined by* $Failures(s) = \{(x, Z) \in (Act - \{\tau\})^* \times 2^{(Act - \{\tau\})} \mid \exists q \in S : s \stackrel{x}{\Longrightarrow} q$ *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.

For definitions of other relations in the linear-time/branching time hierarchy see [vG90].

B Prebisimulation and Intuitionistic Hennessy-Milner Logic

Definition B.1 *[CC92] Let* $\mathcal{P} = (\langle P, Act, \rightarrow, \uparrow \rangle, p_0)$ *and* $\mathcal{Q} = (\langle Q, Act, \rightarrow, \uparrow \rangle, q_0)$ *be two processes. A relation* $R \subseteq P \times Q$ *is a* **prebisimulation** *preorder between* $\mathcal{P}$ *and* $\mathcal{Q}$ *if* $(p,q) \in R$ *implies that for all* $a \in Act$ *the following holds:*

1. $p \stackrel{a}{\rightarrow} p' \Rightarrow \exists q'.q \stackrel{a}{\rightarrow} q' \wedge (p', q') \in R$
2. $p \downarrow a \rightarrow (q \downarrow a \wedge (q \stackrel{a}{\rightarrow} q' \Rightarrow \exists p'.p \stackrel{a}{\rightarrow} p' \wedge (p', q') \in R))$

We say that $\mathcal{P} \sqsubseteq \mathcal{Q}$ *if there is a prebisimulation* R *with* $(p_0, q_0) \in R$.

Recall that we use $p \uparrow a$ in place of $(p, a) \in \uparrow$ and $p \downarrow a$ in place of $\neg(p \uparrow a)$.

Intuitionistic Hennessy-Milner Logic We now recall the logical characterization of the prebisimulation preorder in terms of *Intuitionistic Hennessy Milner* Logic (IHML) as presented in [Sti87]. The abstract syntax of an IHML formula is given as

$$\Phi ::= true \mid false \mid \Phi \wedge \Phi \mid \Phi \vee \Phi \mid < a > \Phi \mid [a]_{\downarrow} \Phi$$

where $a \in Act$.

The formal semantics of IHML is given in terms of the following set of rules that relates the states of a process $\mathcal{P} = (\langle P, Act, \rightarrow, \uparrow \rangle, p_0)$ to formulas. Formally $\models$ is the smallest relation satisfying the following set of rules, where $p \in P$:

$p \models true$
$p \models \Phi_1 \wedge \Phi_2$ if $p \models \Phi_1$ and $p \models \Phi_2$
$p \models \Phi_1 \vee \Phi_2$ if $p \models \Phi_1$ or $p \models \Phi_2$
$p \models < a > \Phi$ if $\exists q.p \stackrel{a}{\rightarrow} q$ and $q \models \Phi$
$p \models [a]_{\downarrow} \Phi$ if $p \downarrow a$ and $\forall q$ if $p \stackrel{a}{\rightarrow} q$ then, $q \models \Phi$

We say that $\mathcal{P} \models \Phi$ if $p_0 \models \Phi$.

C Definitions of Alternation Free Modal Mu-Calculus

C.1 Syntax and Semantics of Basic Formulas

Let *Var* be a countable set of variables. $\mathcal{A}$ a set of atomic propositions, and *Act* a set of actions. The abstract syntax of *basic formulas* is given by the following BNF style notation.

$$\Phi ::= A \mid X \mid \Phi \wedge \Phi \mid \Phi \vee \Phi \mid \langle a \rangle \Phi \mid [a]\Phi$$

where $X \in Var$, $A \in \mathcal{A}$ and $a \in Act$.

Formulas are interpreted with respect to a fixed labelled transition system $\langle \mathcal{S}, Act, \rightarrow \rangle$, a valuation $v : \mathcal{A} \rightarrow 2^{\mathcal{S}}$, and an environment $e : Var \rightarrow 2^{\mathcal{S}}$. The valuation map associates states with atomic propositions, the environment associates states with variables. Thus intuitively, the semantics of a formula Φ with respect an environment e, a transition system $\mathcal{T}$, and a valuation v, denoted as $||\Phi||_e$, means the set of states in which Φ is true. The formal semantics can now be given as follows:

$$||A||_e = v(A)$$

$$||X||_e = e(X)$$

$$||\Phi_1 \wedge \Phi_2||_e = ||\Phi_1||_e \cap ||\Phi_2||_e$$

$$||\Phi_1 \vee \Phi_2||_e = ||\Phi_1||_e \cup ||\Phi_2||_e$$

$$||\langle a \rangle \Phi||_e = \{s \mid \exists s'.s \xrightarrow{a} s' \wedge s' \in ||\Phi||_e\}$$

$$||[a]\Phi||_e = \{s \mid \forall s'.s \xrightarrow{a} s' \Rightarrow s' \in ||\Phi||_e\}$$

C.2 Syntax of Equational Blocks

Basic formulas are not much expressive by themselves and hence there have been enhancements of this logic by adding fixpoint operators. However, the fixpoint operator is not directly added to the syntax as in [Koz83, Bra92]. Instead, *equational blocks* are used following [CS91] to express the fixed points.

A block of equations have one of the two forms, $max\{E\}$ or $min\{E\}$ where E is a list of equations $\{X_1 = \Phi_1, X_2 = \Phi_2, ..., X_n = \Phi_n\}$, and each Φ_i is a basic formula and X_i are all distinct.

If we are restricted to use a single such block to define a formula, then we are restricted to the fragment of modal mu-calculus which correspond to the Hennessy-Milner Logic with recursion [Lar88, Lar90]. We also call this fragment *single fixed point* fragment of modal mu-calculus.

In the case of alternation free modal mu-calculus, several such equational blocks may be used. [CS91] gives a syntactic characterization of alternation-free modal mu-calculus in terms of *block graphs*.

Definition C.1 *[CS91] Let B be a set of blocks. Then the* block graph *of B is defined as follows.*

1. *The nodes are the elements (blocks) of B.*
2. *The edges are defined by: $B_i \rightarrow B_j$, if B_i and B_j are distinct and a left-hand-side variable in B_i appears in a right-hand-side expression of B_j.*

If we are restricted to using blocks such that the corresponding *block graph* is acyclic, then we are restricted to the fragment of modal mu-calculus, called alternation-free modal mu-calculus.

C.3 Semantics of Equational Blocks

Following [CS91], we first define the semantics of a single block $B = m\{E\}$ where m could be *max* or *min* and $E = \{X_1 = \Phi_1, ..., X_n = \Phi_n\}$.

Given a fixed environment e, we can build a semantic function $f_E^e : (2^{\mathcal{S}})^n \rightarrow (2^{\mathcal{S}})^n$ as follows: Let $\overline{S} = \langle S_1, ..., S_n \rangle \in (2^{\mathcal{S}})^n$, and let $e_{\overline{S}} = e[X_1 \mapsto S_1, ..., X_n \mapsto S_n]$ be the environment that results from e by updating the bindings of X_i to S_i. Then

$$f_E^e(\overline{S}) = \langle ||\Phi_1||_{e_{\overline{S}}}, ..., ||\Phi_n||_{e_{\overline{S}}} \rangle$$

.

Note that $(2^{\mathcal{S}})^n$ forms a complete lattice, where the ordering, join and meet are the pointwise extensions of the set theoretic inclusion, union and intersection. Moreover, f_E^e is monotonic with respect to this lattice for any equation system E and given environment e. Hence, by Tarski fixed-point theorem [Tar55], it has both the *greatest* and the *least* fixed points, denoted as νf_E^e and μf_E^e respectively. By [Tar55], $\nu f_E^e = \bigcup\{\overline{S} \mid \overline{S} \subseteq f_E^e(\overline{S})\}$ and $\nu f_E^e = \bigcap\{\overline{S} \mid f_E^e(\overline{S}) \subseteq \overline{S}\}$.

When the labelled transition systems are finite state, f_E^e is continuous and hence both the greatest and the least fixed points have iterative characterization. Let

$$f_0 = \langle \mathcal{S}, ..., \mathcal{S} \rangle$$
$$\hat{f_0} = \langle \phi, ..., \phi \rangle$$
$$f_{i+1} = f_E^e(f_i)\forall_{i \geq 0}$$
$$\hat{f_{i+1}} = f_E^e(\hat{f_i})\forall_{i \geq 0}$$

. Then $\nu f_E^e = \bigcap_{i=0}^{\infty} f_i$ and $\mu f_E^e = \bigcup_{i=0}^{\infty} \hat{f_i}$

Now, blocks $max\{E\}$ and $min\{E\}$ are interpreted as *environments* as follows:

$$||max\{E\}||_e = e_{\nu f_E^e}$$
$$||min\{E\}||_e = e_{\mu f_E^e}$$

. So $max\{E\}(min\{E\})$ represents the greatest (least) fixed point of E.

Now, one can give the semantics of a finite set of blocks. Let $\mathcal{B} = \{B_1, ..., B_n\}$ be a topological sorting of the blocks in B according to the relation $\rightarrow$ defined above. The syntactic restriction on the alternation free fragment of modal mu-calculus makes sure that a variable appearing on the right-hand-side of a block B_j, can only appear in the left-hand side of a block B_i with $i \leq j$, if they appear on the left side at all.

Given a starting environment e, we define the following sequence of environment as in [CS91].

$$e_1 = ||B_1||_e, ..., e_m = ||B_m||_{e_{m-1}}$$

. Then $||\mathcal{B}||_e = e_m$

Now, one can talk about the semantics of a formula Φ, whose variables are bound by a set of equations.

Given a basic formula Φ whose variables are bound by a set of equational blocks $\mathcal{B}$, we define the semantics with respect to an initial environment e as $||\Phi||_{(||\mathcal{B}||_e)}$.

The alternation free fragment of modal mu-calculus is denoted by $L\mu_1$ [EL86]. The **expressivity theorem** in [CS91] states that every formula in $\Gamma \in L\mu_1$ can be translated in linear time to a block set $\mathcal{B}$ with $||\Gamma||_e = ||X||_{(||\mathcal{B}||_e)}$ for some left hand side variable X. Similarly, for every block set $\mathcal{B}$ and variable X, there is a formula Γ in $L\mu_1$, with $||X||_{(||\mathcal{B}||_e)} = ||\Gamma||_e$.

They also show that with a linear blow up in size, one can make all the right hand sides *simple* formulas in linear time in a semantics preserving transformation.

Definition C.2 *[CS91, And94] A formula is simple if it is of the form A, $X_i \vee X_j$, $X_i \wedge X_j$, $\langle a \rangle X_i$ or $[a]X_i$, where A is atomic and X_i, X_j are variables.*

Hence we assume that the equational blocks have only simple formulas on the right hand side.

D Linear time solvability of HORNSAT

The problem minimal-HORNSAT has been solved in [DG84]. The idea there consists in representing the instance as a labelled directed graph with $n + 2$ nodes, n nodes corresponding to the n propositional variables in the instance and two special nodes designated $true$ and $false$. If clause C_i in the instance is of the form x_1, where x_1 is a propositional variable, then there is an edge labelled i, from the node $true$ to the node corresponding to x_1. If clause C_j is of the form $\overline{x_{i1}} \vee \overline{x_{i2}} \vee ... \vee \overline{x_{ik}}$, then there are edges marked j from each of nodes corresponding to $x_{i1}, x_{i2}, ..., x_{ik}$ to the node marked $false$. If clause C_l is of the form $x_{i1} \wedge x_{i2} \wedge ... \wedge x_{ik} \Rightarrow x_a$, then there are edges labelled l, from each of the nodes corresponding to $x_{i1}, x_{i2}, ..., x_{ik}$, to the node corresponding to x_a.

Given this graph representation, they defined a notion of pebbling as follows.

Definition D.1 *[DG84] Let $G = (V, E, L)$ be an edge labelled directed graph. There is a* **pebbling** *of a node $Q \in V$ from a set $X \subseteq V$ if either Q belongs to X or, for some label i, there are pebblings for $P_1, P_2, ..., P_q$ from X, where $P_1, ..., P_q$ are the sources of* **all** *incoming edges to Q labelled i.*

Note that pebbling is equivalent to hypergraph reachability.
Given this definition, [DG84] proves the following :

Proposition D.2 *[DG84] Given a HORNSAT instance h, let $G_h = (V, E, L)$ be its corresponding graph.*

1. *h is satisfiable if and only if there is no pebbling of* **false** *from* $\{$**true**$\}$.
2. *If h is satisfiable, the truth assignment $(v(x_1), v(x_2), ..., v(x_n)$ such that $v(x_i) = 1$ if and only if there is a pebbling of the node corresponding to x_i from* $\{$**true**$\}$ *and $v(x_i) = false$ other wise, is the least element in $SAT(h)$.*

Recall that $SAT(h)$ is the subset of the Boolean lattice $\{0, 1\}^n$ representing all the satisfying assignments of h.

Using this result and by using convenient data structure, they implemented this basic pebbling algorithm to obtain a linear time algorithm for problem minimal-HORNSAT.

In the next section we show that a dualization of the ideas in [DG84] gives a linear time algorithm for problem maximal-NHORNSAT.

E Solution for Maximal-NHORNSAT

In this section we show that a simple dualization of the methods in [DG84] gives a linear time algorithm for the maximal-NHORNSAT problem.

Recall that an NHORNSAT instance (X, C) has clauses of the following three forms.

1. Disjunction of positive literals only (e.g., $x_{i1} \vee x_{i2} \vee ... \vee x_{ik}$).
2. An implication of the form $x_a \Rightarrow x_{i1} \vee x_{i2} \vee ... \vee x_{ik}$
3. A single negated literal (e.g., $\overline{x_j}$).

Now given an instance of NHORNSAT $h = (X, C)$, where $X = \{x_1, x_2, ..., x_n\}$ and $C = \{C_1, C_2, ..., C_m\}$, we construct a graph $G_h = (V, E, L)$, where $V = \{x_1, x_2, ..., x_n\} \cup \{true, false\}$. The label set is $\{1, 2, ..., m\}$. The edges can be described as follows:

1.If the clause C_i is of the form $x_{i1} \vee x_{i2} \vee ... \vee x_{ik}$, there is an edge labelled i, from each of the nodes x_{ij} to the node marked $true$.
2. If C_i is an implication clause of the form $x_a \Rightarrow x_{i1} \vee x_{i2} \vee ... \vee x_{ik}$, then there is an edge labelled i from each of the nodes x_{ij} to the node x_a.
3. If C_i is a single negated literal, $\overline{x_j}$, then there is an edge labelled i from the node marked $false$ to x_j .

Now we can prove a theorem similar to the Proposition D.2 using a dual argument which will show that an instance h, is satisfiable if and only if there is no pebbling in G_h, of $true$ from $\{false\}$. And also, in that process of checking for satisfaction, a pebbling algorithm will produce a maximal satisfying assignment if h is satisfiable.

Theorem E.1 *Given a NHORNSAT instance h, let $G_h = (V, E, L)$ be its corresponding graph.*

1. *h is satisfiable if and only if there is no pebbling of* **true** *from* $\{$**false**$\}$.
2. *If h is satisfiable, the truth assignment $(v(x_1), v(x_2), ..., v(x_n)$ such that $v(x_i) = 0$ if and only if there is a pebbling of the node corresponding to x_i from* $\{$**false**$\}$ *and $v(x_i) = true$ other wise, is the greatest element in $SAT(h)$.*

Now we can use the refinements similar to [DG84] of the pebbling based algorithm to obtain a linear time algorithm for the problem maximal-NHORNSAT.

F Some more Characteristic Games

F.1 Characteristic Game For Forward simulation preorder

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

F.2 Characteristic Game For Trace Equivalence

$Trace - game$ is a game in Stirling class with the following parameters: $R_1 = R_2 = A^*$, $m_1, m_2 = \iota$, $\Gamma = S_1 \times S_2$, $\Sigma = \{\langle s_1, s_2 \rangle\}$, $M = \{1, 2\}$, $r = 1$,

F.3 Characteristic Game For Readiness Equivalence

The *ready set* of a state s in a transition system T, denoted by $Readies(s)$, is defined by
$Readies(s) = \{(x, Z) \in (Act - \{\tau\})^* \times 2^{(Act - \{\tau\})} \mid \exists q \in S : s \stackrel{x}{\Longrightarrow} q$ 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.
$Readiness - game$ is a game in Stirling class with the following parameters:
$R_1 = R_2 = A^*$, $m_1, m_2 = \iota$, $\Gamma = \{\langle s, t \rangle \mid s \in S_1, t \in S_2 \wedge Readies(s) = Readies(t)\}$, $\Sigma = \{\langle s_1, s_2 \rangle\}$, $M = \{1, 2\}$, $r = 1$.

F.4 Characteristic Game for 2-nested Simulation Relation

Given a Transition system $T = \langle S, D, z \rangle$ over Act, where S is the set of states, $D \subseteq S \times Act \times S$ is the transition relation and z is a starting state, any state of T can be thought of as a process. A state p is a process whose transition system $T_p = \langle S, D, p \rangle$. So any relation that we define over states, can also be thought of as being defined over processes. So, below, we are going to define various binary relations over processes.

Definition F.1 *A relation R between processes is a* simulation *iff whenever $(p, q) \in R$, then for each $a \in Act$, for each $p' \in S$, $p \xrightarrow{a} p'$ implies $\exists q' : q \xrightarrow{a} q' \wedge (p', q') \in R$.*

A process p is simulated *by a process q, denoted by $p \preceq_{sim} q$, iff there is a simulation relation R, such that $(p, q) \in R$.*

Two processes p and q are simulation equivalent, *denoted by $p \sim_{sim} q$, iff $p \preceq_{sim} q$ and $q \preceq_{sim} p$.*

Now we define *$n-$nested simulation equivalence* between processes. The notion of $n-$nested simulation equivalence was introduced by Groote and Vaandrager in [GV92].

Definition F.2 *For all $n \in \aleph$, $n-$nested simulation, written $\preceq^n$, is inductively defined as follows.*

- *$p \preceq^0 q$ for all processes p, q.*
- *$p \preceq^{n+1} q$ iff there is a simulation $R \subseteq (\preceq^n)^{-1}$ with $(p, q) \in R$.*

Two processes p and q are $n-$nested simulation equivalent, denoted $p \sim_n q$, iff $p \preceq^n q$ and $q \preceq^n p$.

Note that $1-$nested simulation is just *simulation* and hence $1-$nested simulation equivalence is same as *simulation equivalence.* In other words $\preceq_{sim} = \preceq^1$.

Although the definition of 2-nested simulation relation should be clear from Definition F.2, we define it explicitly for the sake of clarity.

Definition F.3 *$2-$nested simulation, written $\preceq^2$, is defined as follows.*
$p \preceq^2 q$ iff there is a simulation $R \subseteq (\preceq_{sim})^{-1}$ with $(p, q) \in R$.

Two processes p and q are $2-$nested simulation equivalent, denoted $p \sim_2 q$, iff $p \preceq^2$ and $q \preceq^2 p$.

We now prove the following fact about 2-nested simulation.

Definition F.4 *Given two transitions systems $T_1 = \langle S_1, \rightarrow_1, s_1 \rangle$ and $T_2 = \langle S_2, \rightarrow_2, s_2 \rangle$, we say that T_1 is 2-nested simulated by T_2, written $T_1 \preceq^2 T_2$, if and only if $s_1 \preceq^2 s_2$.*

Lemma F.5 *Given two transitions systems $T_1 = \langle S_1, \rightarrow_1, s_1 \rangle$ and $T_2 = \langle S_2, \rightarrow_2, s_2 \rangle$, the following are equivalent:*

1. *T_1 is 2-nested simulated by T_2 (i.e., $T_1 \preceq^2 T_2$)*
2. *There is a simulation relation $R \subseteq S_1 \times S_2$, such that $(s_1, s_2) \in R$, and for all $(x, y) \in R, x \sim_{sim} y$.*

Proof: First we prove (1) $\Rightarrow$ (2). Recall Definition F.3 and Definition F.4. $T_1 \preceq^2 T_2$ implies that $s_1 \preceq^2 s_2$. That implies that there is a simulation relation R containing (s_1, s_2) with the property that for any $(x, y) \in R$, $y \preceq_{sim} x$ because by Definition F.3 $R \subseteq (\preceq_{sim})^{-1}$. Now since $(x, y) \in R$ and R is a simulation relation, obviously $x \preceq_{sim} y$, hence we get by Definition F.1 $x \sim_{sim} y$.

We now prove that (2) $\Rightarrow$ (1). By conditions in 2, there is a simulation relation R that relates s_1 and s_2 and it has the property that for each $(x, y) \in R$, $x \sim_{sim} y$ and hence $y \preceq_{sim} x$ and that means that $R \subseteq (\preceq_{sim})^{-1}$. Hence by Definition F.3 $s_1 \preceq^2 s_2$ and thus by Definition F.4 T_1 is 2-nested simulated by T_2. □

We now define the characteristic game for 2-nested simulation relation. $2nested - game$ is a game in Stirling class with the following parameters: $R_1 = R_2 = A$, $m_1, m_2 = \iota$, $\Gamma = \{(x, y) \in S_1 \times S_2 \mid x \sim_{sim} y\}$, $\Sigma = \{\langle s_1, s_2 \rangle\}$, $M = \{1, 2\}$, $r = \mid S_1 \mid\mid S_2 \mid +1$,