The Complexity of Tree Automata and Logics of Programs†

E. Allen EMERSON[1,2] and *Charanjit S. JUTLA*[3]

1. Department of Computer Sciences, The University of Texas at Austin, USA
2. Mathematics and Computing Science Department, Eindhoven University of Technology, The Netherlands
3. IBM T.J. Watson Research Center, Yorktown Heights, NY, USA

Abstract: The complexity of testing nonemptiness of finite state automata on infinite trees is investigated. It is shown that for tree automata with the pairs (or complemented pairs) acceptance condition having m states and n pairs, nonemptiness can be tested in deterministic time $(mn)^{O(n)}$; however, it is shown that the problem is in general NP-complete (or co-NP-complete, respectively). The new nonemptiness algorithm yields exponentially improved, essentially tight upper bounds for numerous important modal logics of programs, interpreted with the usual semantics over structures generated by binary relations. For example, it follows that satisfiability for the full branching time logic CTL* can be tested in deterministic double exponential time. Another consequence is that satisfiability for Propositional Dynamic Logic with a repetition construct (PDL-delta) and for the Propositional Mu-calculus ($L\mu$) can be tested in deterministic single exponential time.

1. Introduction

There has been a resurgence of interest in automata on infinite objects, due to their intimate relation with temporal and modal logics of programs. They provide an important and uniform approach to the development of decision procedures for testing satisfiability of the propositional versions of these logics ([VW84], [St81]). Such logics and their corresponding decision procedures are not only of inherent mathematical interest, but are also potentially useful in the specification, verification, and synthesis of concurrent programs (cf. [Pn77], [EC82], [MW84], [LPZ85]).

In the case of branching time temporal logic, the standard paradigm nowadays for testing satisfiability is the reduction to the nonemptiness problem for finite state automata on infinite trees; i.e., one builds a tree automaton which accepts essentially all models of the candidate formula and then tests nonemptiness of the tree automaton. Thus in order to improve the complexity there are two issues: (1) the size of the tree automaton, and (2) the complexity of testing nonemptiness of the tree automaton.

In this paper we obtain new, improved and essentially tight bounds on testing nonemptiness of tree automata that allow us to close an exponential gap which has existed between the upper and lower bounds of the satisfiability problem of numerous important Modal Logics of Programs. These logics include CTL^*- the full Branching Time Logic [EH83], PDL-delta - Propositional Dynamic Logic with an infinite repetition construct [St81], and the propositional Mu-calculus ($L\mu$) - a language for characterizing temporal correctness properties in terms of extremal fixpoints of predicate transformers [Ko83] (cf. [EC80]).

To obtain these improvements, we focus on the complexity of testing nonemptiness of tree automata. We first note, however, that the size of an automaton has two parameters: the number of states in the automaton's transition diagram and the number of pairs in its acceptance condition. We next make the following important observation: for most logics of programs, the number of pairs is logarithmic in the number of states.

We go on to analyze the complexity of testing nonemptiness of pairs tree automata [Ra69] and show that it is NP-complete. However, a multi-parameter analysis shows that there is an algorithm that runs in time $(mn)^{O(n)}$ which is polynomial in the number of states m exponential in the number n of pairs in the acceptance condition of the automaton. The algorithm is based on a type of "pseudo-model checking" for certain restricted

†This work was supported in part by NSF grant CCR-941-5496, ONR URI contract N00014-86-K-0763, and Netherlands NWO grant nf-3/nfb 62-500.

Mu-calculus formulae. Moreover, since the problem is NP-complete, it is unlikely to have a better algorithm which is polynomial in both parameters. The previous best known algorithm was in NP ([Em85], [VS85]).

The above nonemptiness algorithm now permits us to obtain a deterministic double exponential time decision procedure for CTL^*, by using the reduction from CTL^* to tree automata obtained in [ESi84], in which the size of the automaton is double exponential in the length of the formula and the number of pairs is only exponential in the length of the formula. The bound follows by simple arithmetic, since a double exponential raised to a single exponential power is still a double exponential.

This amounts to an exponential improvement over the best previously known algorithm which was in non-deterministic double exponential time ([Em85], [VS85]) i.e., three exponentials when determinized. It is also essentially tight, since CTL^* was shown to be double-exponential time hard [VS85]; thus CTL^* is deterministic double exponential time complete.

The above result has been obtained using only the classical pairs tree automata of Rabin [Ra69]. However, we also consider the complemented pairs tree automata of Streett [St81], which were specifically introduced to facilitate formulation of temporal decision procedures. We show that the nonemptiness problem of complemented pairs automata is co-NP-complete, by reducing the complement of the problem to nonemptiness of the pairs automata and vice-versa. The reduction employs the fact that infinite Borel Games are determinate (Martin's Theorem [Ma75]). This reduction also gives a deterministic algorithm which is polynomial in the number of states and exponential in the number of pairs. We can thus re-establish the above upper bound for CTL^* using complemented pairs automata as well.

Using the recent single exponential general McNaughton [Mc66] construction of Safra [Sa88] (i.e., construction for determinizing a Büchi finite automaton on infinite strings), our new nonemptiness algorithm also gives us a deterministic single exponential time decision procedure for both PDL-delta and the Mu-Calculus, since the Safra construction allows us to reduce satisfiability of these logics to testing nonemptiness of a tree automaton with exponentially many states and polynomially many pairs. This represents an exponential improvement over the best known deterministic algorithms for these logics, which took deterministic double exponential time, corresponding to the nondeterministic exponential time upper bounds of [VS85]. The bounds are essentially tight also, since the exponential time lower bound follows from that established for ordinary PDL (Propositional Dynamic Logic) by Fischer-Ladner [FL79].

It is interesting to note that, for most logics (including PDL-delta and the Mu-Calculus but excluding CTL^*), our nonemptiness algorithm(s) and Safra's construction both play a crucial role. Each is independent of the other. Moreover, each is needed and neither alone suffices. Our algorithm improves the complexity of testing nonemptiness by an exponential factor, while Safra's construction independently applies to reduce the size of the automaton by an exponential factor. For example, the "traditional" result of Streett [St81] gave a deterministic triple exponential algorithm for PDL-delta. Our algorithm alone improves it to deterministic double exponential time. Alternatively, Safra's construction alone improves it to deterministic double exponential time. As shown in this paper, the constructions can be applied together to get a cumulative double exponential speedup for PDL-delta. In the case of CTL^*, we already had the effect of Safra's construction, because [ESi84] gave a way to determinize with only a single exponential blowup the Büchi string automaton corresponding to a linear Temporal Logic formula, using the special structure of such automata (unique accepting run). So Safra's construction provides no help for the complexity of the CTL^* logic.

The remainder of the paper is organized as follows: in section 2 we give preliminary definitions and terminology. In section 3 we establish a "Small Model Theorem" for (pairs) tree automata. In section 4 we give the main technical results on testing nonemptiness of pairs tree automata. In section 5 we give the main results on nonemptiness of complemented pairs tree automata. Applications of the algorithms to testing satisfiability of modal logics of programs, including CTL^*, PDL-delta, and the Mu-calculus are described in section 6. Some concluding remarks are given in section 7.

2 Preliminaries

2.1 Logics of Programs

Full Branching Time Logic

The *Full Branching Time Logic* CTL^* [EH83] derives its expressive power from the freedom of combining modalities which quantify over paths and modalities which quantify states along a particular path. These modalities are A, E, F, G, X_s, and U_w ("for all futures", "for some future", "sometime", "always", "strong nexttime", and "weak until", respectively), and they are allowed to appear in virtually arbitrary combinations. Formally, we inductively define a class of state formulae (true or false of states) and a class of path formulae (true or false of paths):

(S1) Any atomic proposition P is a state formula.

(S2) If p, q are state formulae, then so are $p \wedge q$, $\neg p$.

(S3) If p is a path formula then Ep is a state formula.

(P1) Any state formula p is also a path formula.

(P2) If p, q are path formulae, then so are $p \wedge q$, $\neg p$.

(P3) If p, q are path formulae then so are $X_s p$ and $p U_w q$.

The semantics of a formula are defined with respect to a structure $M = (S, R, L)$, where S is a nonempty set of states, R is a nonempty binary relation on S, and L is a labeling which assigns to each state a set of atomic propositions true in the state. A *fullpath* $(s_1, s_2, ...)$ is a maximal sequence of states such that $(s_i, s_{i+1}) \in R$ for all i. A fullpath is infinite unless for some s_k there is no s_{k+1} such that $(s_k, s_{k+1}) \in R$. We write $M, s \models p$ $(M, x \models p)$ to mean that state formula p (path formula p) is true in structure M at state s (of fullpath x, resp.). When M is understood, we write simply $s \models p$ $(x \models p)$. We define $\models$ inductively using the convention that $x = (s_1, s_2, ...)$ denotes a fullpath and x^i denotes the suffix fullpath $(s_i, s_{i+1}, ...)$, provided $i \leq |x|$, where $|x|$, the length of x, is ω when x is infinite and k when x is finite and of the form $(s_1, \ldots, s_k)$; otherwise x^i is undefined. For a state s:

(S1) $s \models P$ iff $P \in L(s)$ for atomic proposition P

(S2) $s \models p \wedge q$ iff $s \models p$ and $s \models q$,
$s \models \neg p$ iff not $(s \models p)$

(S3) $s \models Ep$ iff for some fullpath x starting at s, $x \models p$

For a fullpath $x = (s_1, s_2, ...)$:

(P1) $x \models p$ iff $s_1 \models p$, for any state formula p

(P2) $x \models p \wedge q$ iff $x \models p$ and $x \models q$,
$x \models \neg p$ iff not $(x \models p)$

(P3) $x \models X_s p$ iff x^2 is defined and $x^2 \models p$,
$x \models (p\, U_w\, q)$ iff for all $i \in [1 : |x|]$, if for all $j \in [1 : i]$ $x^j \models \neg q$ then $x^i \models p$

We say that state formula p is *valid*, and write $\models p$, if for every structure M and every state s in M, $M, s \models p$. We say that state formula p is *satisfiable* iff for some structure M and some state s in M, $M, s \models p$. In this case we also say that M defines a *model* of p. We define validity and satisfiability for path formulae similarly.

We write $f \doteq g$ to mean that formula f abbreviates formula g. Other connectives can then be defined as abbreviations in the usual way: $p \vee q \doteq \neg(\neg p \wedge \neg q)$, $p \Rightarrow q \doteq \neg p \vee q$, $p \Leftrightarrow q \doteq (p \Rightarrow q) \wedge (q \Rightarrow p)$, $Ap \doteq \neg E \neg p$, $Gp \doteq p\, U_w\, false$, and $Fp \doteq \neg G \neg p$. Further operators may also be defined as follows:
$X_w p \doteq \neg X_s \neg p$ is the weak nexttime,
$p U_s q \doteq (p U_w q) \wedge Fq$ is the strong until,
$\overset{\infty}{F} p \doteq GFX_s p$ means infinitely often p,
$\overset{\infty}{G} p \doteq FGX_s p$ means almost everywhere p,
$inf \doteq GX_s true$ means the path is infinite, and
$fin \doteq FX_w false$ means the path is finite.

Propositional Dynamic Logic plus Repeat

As opposed to CTL^*, in which the models represent behaviors of the programs, in *PDL-delta*, Propositional Dynamic Logic plus Repetition, the programs are explicit in the models. The modalities in PDL-delta quantify the states reachable by programs explicitly stated in the modality. Thus, for a program B (which is obtained from atomic programs and tests using regular expressions), $< B > p$ ($[B]p$) states that there is an execution of B leading to p (after all executions of B, p holds). Also included is the infinite repetition construct $delta(\triangle)$ which makes PDL-delta much more expressive than PDL. $\triangle B$ states that it is possible to execute B repetitively infinitely many times. PDL-delta formulae are interpreted over structures $M = (S, R, L)$, where S is a set of states, $R : Prog \to 2^{S \times S}$ is a transition relation, $Prog$ is the set of atomic programs, and L is a labeling of S with propositions in $Prop$. For more details see [St81].

Propositional Mu-Calculus

A *least fixpoint* construct can be used to increase the power of simple Modal Logics. Thus, by adding this construct to PDL, we get $L\mu$, the *Propositional Mu-calculus* [Ko83], a logic which subsumes PDL-delta. A variant formulation of the Mu-calculus, which we use here, adds the least fixpoint construct to a simple subset of CTL* including just nexttime (AX_s), the boolean connectives, and propositions (cf. [EC80]). The least fixpoint construct has the syntax $\mu Y.f(Y)$, where $f(Y)$ is any formula syntactically monotone in the propositional variable Y, i.e. all occurrences of Y in $f(Y)$ fall under an even number of negations. It is interpreted as the smallest set S of states such that $S = f(S)$. By the well known Tarski-Knaster Theorem $\mu Y.\ f(Y) = \bigcup_i\ f^i(\text{false})$ where i ranges over all ordinals and f^i (intuitively) denotes the i-fold composition of f with itself; when the domain is finite we may take i as ranging over just the natural numbers. Its dual, the greatest fixpoint, is denoted $\nu Y.f(Y)$ $(\equiv \neg \mu Y. \neg f(\neg Y))$. Thus, e.g, $\mu Y.[A]Y$ is equivalent to $\neg \triangle A$ of PDL-delta. Similarly, using temporal logic, $\mu Y.P \vee AX_s Y$ is equivalent to AFP (i.e. along all paths eventually P holds). Many correctness properties of concurrent programs can be characterized in terms of the Mu-calculus, including all those expressible in CTL* and PDL-delta. For more details, see, for example, [EC80],[Ko83],[StE84],[EL86].

The *formulae* of the (Propositional) Mu-Calculus are:
(1) Propositional constants $P, Q, ...$
(2) Propositional variables $Y, Z, ...$
(3) $\neg p$, $p \vee q$, and $p \wedge q$, where p and q are any formulae
(4) $EX_s p$ and $AX_s p$, where p is any formula
(5) $\mu Y.f(Y)$ and $\nu Y.f(Y)$, where $f(Y)$ is any formula syntactically monotone in the propositional variable Y, i.e. all occurrences of Y in $f(Y)$ fall under even number of negations.

In the sequel, we will use σ as a generic symbol for μ or ν. In a fixed point expression $\sigma Y.f(Y)$, we say that each occurrence of Y is *bound* to σY. If an occurrence of Y is not bound then it is *free*. A *sentence* (or *closed formula*) is a formula containing no free propositional variables, i.e. no variables unbound by a μ or a ν operator.

Sentences are interpreted over structures $M = (S, R, L)$ as for CTL*. As usual we will write $M, s \models p$ to mean that in structure M at state s sentence p holds true. To give the technical definition of $\models$ we need some preliminaries.

The power set of S, 2^S, may be viewed as the complete lattice $(2^S, S, \phi, \subseteq, \cup, \cap)$. Intuitively, we identify a proposition with the set of states which make it true. Thus, $false$, which corresponds to the empty set is the bottom element, $true$, which corresponds to S, is the top element, $\cup$ is join, $\cap$ is meet, and implication $(\forall s \in S(P(s) \Rightarrow Q(s)))$, which corresponds to simple set-theoretic containment $(P \subseteq Q)$, provides the partial ordering on the lattice.

Let $\tau : 2^S \rightarrow 2^S$ be given; then we say that τ is *monotonic* provided $P \subseteq Q$ implies $\tau(P) \subseteq \tau(Q)$. A monotonic functional τ always has both a least fixpoint $\mu X.\tau(X)$ and a greatest fixpoint $\nu X.\tau(X)$.

For a formula or function $p(Y)$, we write $p^0(Y) = false$, $p^1(Y) = p(Y)$, $p^{i+1}(Y) = p(p^i)(Y)$, for successor ordinal $i+1$, and $p^j(Y) = \bigcup_{k<j} p^k(Y)$ for limit ordinal j.

Theorem 2.1 (Tarski-Knaster). Let $\tau : 2^S \rightarrow 2^S$ be a given monotonic functional. Then,
(a) $\mu Y.\tau(Y) = \bigcap\{Y : \tau(Y) = Y\} = \bigcap\{Y : \tau(Y) \subseteq Y\}$,
(b) $\nu Y.\tau(Y) = \bigcup\{Y : \tau(Y) = Y\} = \bigcup\{Y : \tau(Y) \supseteq Y\}$,
(c) $\mu Y.\tau(Y) = \bigcup_{i \leq |S|} \tau^i(false)$, and
(d) $\nu Y.\tau(Y) = \bigcap_{i \leq |S|} \tau^i(true)$.

A formula p with free variables $Y_0, Y_1, ...Y_n$ is thus interpreted as a mapping p^M from $(2^S)^{n+1}$ to 2^S, i.e., it is interpreted as a predicate transformer. We write $p(Y_0, Y_1, ...Y_n)$ to denote that all free variables of p are among $Y_0, Y_1, ...Y_n$. Let $V_0, V_1, ...V_n$ be subsets of S; then a *valuation* $\Upsilon = V_0, V_1, ...V_n$ is an assignment of $V_0, V_1, ...V_n$ to the free variables $Y_0, Y_1, ...Y_n$, respectively. $\Upsilon[Y_i \leftarrow V_i']$ denotes the valuation identical to Υ except that Y_i is assigned V_i'. We use $p^M(\Upsilon)$ to denote the value of p in structure M on the arguments $V_0, V_1, ...V_n$. We drop M when it is understood from context. We then let

$M, s \models p(\Upsilon)$ iff $s \in p^M(\Upsilon)$, and we define $\models$ inductively as follows:

(1) $s \models P(\Upsilon)$ iff $P \in L(s)$

(2) $s \models Y(\Upsilon)$ iff $s \in \Upsilon(Y)$

(3) $s \models (\neg p)(\Upsilon)$ iff $s \not\models p(\Upsilon)$

$s \models (p \vee q)(\Upsilon)$ iff $s \models p(\Upsilon)$ or $s \models q(\Upsilon)$
$s \models (p \wedge q)(\Upsilon)$ iff $s \models p(\Upsilon)$ and $s \models q(\Upsilon)$

(4) $s \models (EX_s p)(\Upsilon)$ iff $\exists t(s,t) \in R$ and $t \models p(\Upsilon)$
$s \models (AX_s p)(\Upsilon)$ iff (a) $\exists u\ (s,u) \in R$ and $u \models p$ and (b) $\forall t\ (s,t) \in R$ implies $t \models p(\Upsilon)$

(5) $s \models (\mu Y.f(Y))(\Upsilon)$ iff $s \in \bigcap\{S' \subseteq S | S' = \{t : t \models f(Y)(\Upsilon[Y \leftarrow S'])\}\}$,
$s \models (\nu Y.f(Y))(\Upsilon)$ iff $s \in \bigcup\{S' \subseteq S | S' = \{t : t \models f(Y)(\Upsilon[Y \leftarrow S'])\}\}$.

Conventions.

To avoid a proliferation of unnecessary parentheses, we order the connectives from greatest to lowest *binding power* as follows: $\neg$ binds tighter than $F, G, X_w, X_s, \overset{\infty}{F}, \overset{\infty}{G}$ which bind tighter than $\wedge$ which binds tighter than $\vee$ which binds tighter than $\Rightarrow$ which binds tighter than U_w, U_s which bind tighter than A, E which bind tighter than μ, ν which bind tighter than $\Leftrightarrow$.

If we write $M, s \models p$ it is implicit that s is a state of M. That is, $s \in S$ where $M = (S, R, L)$. A convenient abuse of notation is to write $s \in M$ in some places.

If p is a formula then p^M denotes $\{s : M, s \models p\}$, the set of states s in M at which p is true.

We can write $M, s_1, \ldots, s_k \models p$ to abbreviate $M, s_1 \models p$ and ... and $M, s_k \models p$.

We write $p \equiv q$ for $\models p \Leftrightarrow q$. In the context of an structure M, we can also write $p \equiv_M q$ for $p^M = q^M$. If M is understood then we can drop the M and write just $p \equiv q$. It should be clear from context whether equivalence over all structures or over M is meant. We use $p_1 \equiv p_2 \equiv \ldots \equiv p_k$ as shorthand for $p_1 \equiv p_2$ and ... and $p_{k-1} \equiv p_k$.

Technically, we distinguish between an atomic proposition symbol P and the associated set, viz. P^M, of states which are labelled with it in structure M. It is often convenient notation to use the uppercase sans serif symbol P corresponding to P to denote the set of states that are labelled with P.

Remark: We note the following identities:
$EX_w false \equiv AX_w false$ and asserts that a state has no successors.
$EX_s true \equiv AX_s true$ and asserts that a state has one or more successors.

2.2 Automata on Infinite Trees

We consider finite automata on labelled, infinite binary trees.[1] The set $\{0,1\}^*$ may be viewed as an infinite binary tree, where the empty string λ is the root node and each node u has two successors: the 0-successor $u0$ and the 1-successor $u1$. A finite (infinite) path through the tree is a finite (resp., infinite) sequence $x = u_0, u_1, u_2, \ldots$ such that each node u_{i+1} is a successor of node u_i. If Σ is an alphabet of symbols, an infinite binary Σ-*tree* is a labelling L which maps $\{0,1\}^* \longrightarrow \Sigma$.

A *finite automaton* $\mathcal{A}$ on infinite binary Σ-trees consists of a tuple $(\Sigma, Q, \delta, q_0, \Phi)$ where
Σ is the finite, nonempty input alphabet labeling the nodes of the input tree,
Q is the finite, nonempty set of states of the automaton,
$\delta : Q \times \Sigma \rightarrow 2^{Q \times Q}$ is the non-deterministic transition function,

[1] We consider here only binary trees to simplify the exposition, and for consistency with the classical theory of tree automata. CTL^* and the other logics we study, have the property that their models can be unwound into an infinite tree. In particular, in [ESi84] it was shown that a CTL^* formula of length k is satisfiable iff it has an infinite tree model with finite branching bounded by k, i.e. iff it is satisfiable over a k-ary tree. Our results on tree automata apply to such k-ary trees as explained at the end of the proof of Theorem 4.1.

$q_0 \in Q$ is the start state of the automaton, and
Φ is an acceptance condition described subsequently.

A *run* of $\mathcal{A}$ on the input Σ-tree L is a function $\rho : \{0,1\}^* \to Q$ such that for all $v \in \{0,1\}^*$, $(\rho(v0), \rho(v1)) \in \delta(\rho(v), L(v))$ and $\rho(\lambda) = q_0$. We say that $\mathcal{A}$ *accepts* input tree L iff there exists a run ρ of $\mathcal{A}$ on L such that for all infinite paths x starting at the root of L if $r = \rho \circ x$ is the sequence of states $\mathcal{A}$ goes through along path x, then the acceptance condition Φ holds along r.

For a *pairs* automaton (cf. [McN66], [Ra69]) acceptance is defined in terms of a finite list $((\mathsf{RED_1}, \mathsf{GREEN_1}), \ldots, (\mathsf{RED_k}, \mathsf{GREEN_k}))$ of pairs of sets of automaton states (which may be thought of as pairs of colored lights where $\mathcal{A}$ flashes the red light of the first pair upon entering any state of the set $\mathsf{RED_1}$, etc.): r satisfies the pairs condition iff there exists a pair $i \in [1..k]$ such that $\mathsf{RED_i}$ flashes finitely often and $\mathsf{GREEN_i}$ flashes infinitely often. We assume the pairs acceptance condition is given formally by a temporal logic formula $\Phi = \bigvee_{i \in [1..k]} (GF\ \mathrm{GREEN}_i \wedge \neg GF\ \mathrm{RED}_i)$.[2] Similarly, a *complemented pairs* (cf. [St81]) automaton has the negation of the pairs condition as its acceptance condition; i.e., for all pairs $i \in [1..k]$, GREEN_i flashes infinitely often implies that RED_i flashes infinitely often too. The complemented pairs acceptance condition is given formally by a temporal logic formula $\Phi = \bigwedge_{i \in [1:k]} GF\ \mathrm{GREEN}_i \Rightarrow GF\ \mathrm{RED}_i$.

3 Small Model Theorems

3.1 Tree Automata Running on Graphs

Note that an infinite binary tree L' may be viewed as a "binary" structure $M = (S, R, L)$ where $S = \{0,1\}^*$, $R = R_0 \cup R_1$ with $R_0 = \{(s, s0) : s \in S\}$ and $R_1 = \{(s, s1) : s \in S\}$, and $L = L'$. We could alternatively write $M = (S, R_0, R_1, L)$.

We can also define a notion of a tree automaton running on certain appropriately labelled binary, directed graphs, that are not binary trees. Such graphs, if accepted, are witnesses to the nonemptiness of tree automata. We make the following definitions.

A *binary structure* $M = (S, R_0, R_1, L)$ consists of a state set S and labelling L as before, plus a transition relation $R_0 \cup R_1$ decomposed into two partial functions: $R_0 : S \longrightarrow S$, where $R_0(s)$, when defined, specifies the 0-successor of s, and $R_1 : S \longrightarrow S$, where $R_1(s)$, when defined, specifies the 1-successor of s. We say that M is a *full binary structure* iff R_0 and R_1 are total.

A *run* of automaton $\mathcal{A}$ on binary structure $M = (S, R_0, R_1, L)$, if it exists, is a mapping $\rho : S \to Q$ such that $\forall s \in S$, $(\rho(R_0(s)), \rho(R_1(s))) \in \delta(\rho(s), L(s))$ and $\rho(s_0) = q_0$. Intuitively, a run is a labeling of M with states of $\mathcal{A}$ consistent with the local structure of $\mathcal{A}$'s transition diagram. It will turn out that if an automaton accepts some binary tree, there does exist some finite binary graph on which there is a run that is *accepting*: all of the paths trhough the graph define state sequences of the automaton meeting its acceptance condition.

3.2 The Transition Diagram of a Tree Automaton

The transition diagram of $\mathcal{A}$ can be viewed as an AND/OR-graph where the set Q of states of $\mathcal{A}$ comprises the set of OR-nodes, while the AND-nodes define the allowable moves of the automaton. Intuitively, OR-nodes indicate that a non-deterministic choice has to be made (depending on the input label), while the AND-nodes force the automaton along all directions. For example, suppose that for automaton $\mathcal{A}$, $\delta(s, a) = \{(t_1,\ u_1), \ldots,\ (t_m, u_m)\}$ and $\delta(s, b) = \{(v_1,\ w_1),\ \ldots,\ (v_n,\ w_n)\}$ then the transition diagram contains the portion shown in Figure 3.1.

[2] We are assuming that each proposition symbol, such as GREEN_i, of formula Φ is associated with exactly the set of states of the corresponding name, in this case GREEN_i.

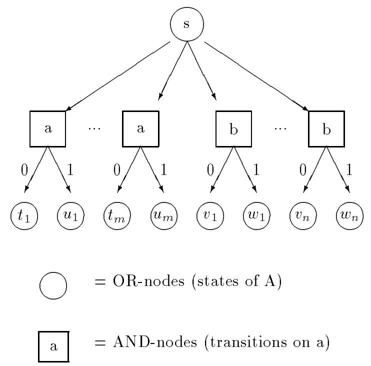


Figure 1

Formally, given a tree automaton $\mathcal{A} = (\Sigma, Q, \delta, q_0, \Phi)$, with transition function $\delta : Q \times \Sigma \longrightarrow 2^{Q \times Q} : (q, a) \longmapsto \{(r_1, s_1), \ldots, (r_k, s_k)\}$ we may view it as defining a transition diagram T, which is an AND/OR-graph $(D, C, R_{DC}, R_{CD0}, R_{CD1}, L)$ where

$D = Q$, the set of OR-nodes,

$C = \bigcup_{q \in D} \bigcup_{a \in \Sigma} \{((q, a), (r, s)) : (r, s) \in \delta(q, a)\}$ is the set of AND-nodes. Each AND-node corresponds to a transition. If $\delta(q, a) = \{(r_1, s_1), \ldots, (r_k, s_k)\}$ then the corresponding AND-nodes are essentially the pairs $(r_1, s_1), \ldots, (r_k, s_k)$. However, since each transition is associated with a unique current state/input symbol pair (q, a), we formally define the corresponding AND-nodes to be pairs of pairs: $((q, a), (r_1, s_1)), \ldots, ((q, a), (r_k, s_k))$.

$R_{DC} \subseteq D \times C$ specifies the AND-node successors of each OR-node. For each $q \in D$ as above, $R_{DC}(q) = \bigcup_{a \in \Sigma} \{((q, a), (r, s)) : (r, s) \in \delta(q, a)\}$.[3]

$R_{CD0}, R_{CD1} : C \longrightarrow D$ are partial[4] functions giving the 0-successor and 1-successor states, respectively: $R_{CD0}(((q, a), (r, s))) = r$ and $R_{CD1}(((q, a), (r, s))) = s$.

L is a labelling of nodes. For an AND-node $L(((q, a), (r, s))) = \{a\}$ where $a \in \Sigma$. For an OR-node, the labelling assigns propositions associated with the acceptance condition Φ, so that all OR-nodes in the set $\mathsf{GREEN_i}$ (resp., $\mathsf{RED_i}$) are labelled with the corresponding proposition GREEN_i (resp., RED_i).

Thus, we may write a tree automaton A in the form (T, d_0, Φ) where T is the diagram, d_0 is the start state, and Φ is an acceptance condition.

[3] We identify a relation such as $R_{DC} \subseteq D \times C$ with the corresponding function $R'_{DC} : D \longrightarrow 2^C$ defined by $R'_{DC}(d) = \{c \in C : (d, c) \in R_{DC}\}$ for each $d \in D$.

[4] For classically defined tree automata, the functions R_{CD0}, R_{CD1} are total so that there is always a 0-successor and 1-successor automaton state. For technical reasons it is convenient to allow R_{CD0}, R_{CD1} to be partial.

3.3 One Symbol Alphabets

For purposes of testing nonemptiness, without loss of generality, we can restrict our attention to tree automata over a single letter alphabet, and, thereby, subsequently ignore the input alphabet. Let $\mathcal{A} = (Q, \Sigma, \delta, q_0, \Phi)$ be a tree automaton over input alphabet Σ. Let $\mathcal{A}' = (Q, \Sigma', \delta', q_0, \Phi)$ be the tree automaton over one letter input alphabet $\Sigma' = \{c\}$ obtained from $\mathcal{A}$ by, intuitively, taking the same transition diagram but now making all transitions on symbol c. Formally, $\mathcal{A}'$ is identical to $\mathcal{A}$ except that the input alphabet is Σ' and the transition function δ' is defined by $\delta'(q,c) = \bigcup_{a\in\Sigma}\delta(q,a)$.

Observation 3.1 The set accepted by $\mathcal{A}$ is nonempty iff the set accepted by $\mathcal{A}'$ is nonempty.

Henceforth, we shall therefore assume that we are dealing with tree automata over a one symbol alphabet.

3.4 Generation and Containment

It is helpful to reformulate the notion of run to take advantage of the AND/OR-graph organization of the transition diagram of an automaton. Intuitively, there is a run of diagram T on structure M provided M is "generated" from T by unwinding T, so that that each state of M is a copy of an OR-node of T.

Formally, we say that a binary structure $M = (S, R_0, R_1, L)$ is a *generated by* transition diagram $T = (D, C, R_{DC}, R_{CD0}, R_{CD1}, L_T)$ (*starting at* $s_0 \in S$ *and* $d_0 \in D$) iff
$\exists$ a total function $h : S \longrightarrow D$ such that $\forall s \in S$
if s has any successors in M then
$\exists c \in R_{DC}(h(s))$
$\forall i \in \{0,1\}$ $R_{CDi}(c)$ is defined iff $R_i(s)$ is defined, and
$R_{CDi}(c) = h(R_i(s))$ when both are defined
and $L(s) = L_T(h(s))$
(such that $h(s_0) = d_0$).

We say that a binary structure $M = (S, R_0, R_1, L)$ is *contained in* transition diagram T (*starting at* $s_0 \in M$ *and* $q_0 \in T$) provided M is generated by T (starting at $s_0 \in M$ and $q_0 \in T$) where the generation function h is the natural injection $h : S \longrightarrow D : s \in S \longmapsto s \in D$, so that all states of M are OR-nodes of T.

Note that if M is a structure generated by (respectively, contained in) T, there is an associated AND/OR-graph H generated by (respectively, contained in) T obtained from M by inserting between each state and its successors in M a copy of the AND-node that determines the successors of state via the generation function for M. We write $H = ao(M)$. Conversely, if H is an AND/OR-graph generated by (respectively, contained in) T there is an associated structure M generated by (respectively, contained in) T obtained from H by eliding AND-nodes. Here we write $M = o(H)$.

3.5 Linear Size Model Theorems

The following theorem (cf. [Em85]) is the basis of our method of testing nonemptiness of pairs automata. It shows that there is a small binary structure contained in the transition diagram and accepted by the automaton.

Theorem 3.2 (Linear Size Model Theorem)

Let $\mathcal{A}$ be a tree automaton over a one symbol alphabet with pairs acceptance condition $\Phi = \bigvee_{i\in[1:k]}(\overset{\infty}{F}Q_i \wedge \overset{\infty}{G}P_i)$.
Then automaton $\mathcal{A}$ accepts some tree T iff $\mathcal{A}$ accepts some binary model M, of size linear in the size of $\mathcal{A}$, which is a structure contained in the transition diagram of $\mathcal{A}$.

Proof. ($\Rightarrow$) For Φ a pairs condition, by Hossley-Rackoff ([HR72]) finite model theorem, if $\mathcal{A}$ accepts some tree M_0, then it accepts some finite binary model M_1 starting at some state $s_0 \in M_1$. Thus, $M_1, s_0 \models A\Phi$ and M_1 is a structure generated by $\mathcal{A}$.

Given any such finite structure M_1 of $A\Phi$ generated by $\mathcal{A}$ we can obtain a finite structure M contained in $\mathcal{A}$as follows. Let h be the generation function. If two distinct nodes s and t of M_1 have the same labelling with states of $\mathcal{A}$, i,e, $h(s) = h(t)$, then we can eliminate one of them as follows. Attempt to delete s by redirecting all its predecessors u to have t as a successor instead. More precisely, delete all edges of the form (u, s) and replace them by edges of the form (u,t). If the resulting structure, call it M^t, is a model of $A\Phi$, we have reduced the number of "duplicates", such as s and t by one. If not, try replacing t by s instead. If M^s, the resulting structure is a model of $A\Phi$ we are done.

However, if both these replacements fail to yield a model of $A\Phi$, each must introduce a (not necessarily simple) bad cycle where the acceptance condition Φ fails. In M^t the bad cycle is of the form (where u is a predecessor of s in M_1) $u{\to}t{\to}...{\to}u$, where except for the first transition (u,t) the suffix path from t to u is in the original M_1. In M^s the bad cycle is of the form (where v is a predecessor of t in M_1) $v{\to}s{\to}...{\to}v$, where except for the first transition (v, s) the suffix path from s to v is in the original M_1.

But, these two suffix paths in M_1 together with the edges (u, s), and (v,t) in M_1 form a bad cycle in M_1: $u{\to}s{\to}...{\to}v{\to}t{\to}...{\to}u$. This contradicts that M_1 was a model of $A\Phi$.

By repeatedly eliminating duplicates in this way we eventually get the desired model M contained in $\mathcal{A}$.

($\Leftarrow$) Any model M contained in $\mathcal{A}$ such that $M, s_0 \models A\Phi$ can plainly be unwound into a tree that is accepted by $\mathcal{A}$ $\square$

A helpful generalization follows:

Theorem 3.3 (Generalized Linear Sized Model Theorem)

Let $s_0, s_1, ...s_k$ ($k \geq 0$) be distinct elements of T, and formula $g = P \wedge A(\Phi \vee FR)$, where $\Phi = \bigvee_{i\in[1:k]}(\overset{\infty}{F}Q_i \wedge \overset{\infty}{G}P_i)$ is a pairs acceptance condition and P, R are atomic propositions. Then,

$\exists M'$ generated by T such that $M', s_0, s_1, ...s_k \models g$
iff
$\exists M$ contained in T such that $M, s_0, s_1, ...s_k \models g$.

Proof. ($\Rightarrow$) Assume the existence of M'. We first show that this implies the existence of M'' generated by T such that
$M'', s_0, s_1, ...s_k \models P \wedge A(\Phi \vee FR)$ and $\forall s \in M''$, $M'', s \models A(\Phi \vee FR)$.

Intuitively, M'' is obtained by deleting spurious nodes from M', retaining just those nodes which are in the "cone" from some s_i until some R- node, if any, is encountered. Technically, let $S = \{t \in M'$: there exists a path from some s_i to t in M' all of whose nodes except (possibly) t itself satisfy $\neg R\}$. Let $S' = M' - S$.

Thus, S' consists of all nodes that can only be reached from any s_i by going through some R-node. Now, delete all successor arcs of R-nodes, and then delete all nodes no longer reachable from some s_i. Note that the deleted nodes are exactly S'. Call the resulting substructure of M' so obtained M''. Note that the nodes retained in M'' are exactly S, and that M'' is still generated by T.

Now observe that $A(\Phi \vee FR) \equiv A(A(\Phi \vee FR)\, U_w\, R)$. Therefore, for all $t \in S$, $M', t \models A(\Phi \vee FR)$. Hence, for all $t \in S$, $M'', t \models A(\Phi \vee FR)$, as the only nodes deleted in M'' are the ones which are reached from any s_i by going through some R-node. Thus, $M'', s_0, s_1, ...s_k \models A(\Phi \vee FR) \wedge P$, and $\forall s \in M''$, $M'', s \models A(\Phi \vee FR)$.

Any fullpath starting at any node in M'' is either
(i) infinte and satisfies Φ, or
(ii) is finite, terminating in an R-node.

We can now argue, just as in the Linear Size model Theorem, that duplicates can be eliminated, If u and v are duplicates, we can attempt to replace u by v. The only problem is that it may introduce a "bad" cycle satisfying $\neg\Phi$. In that case, it must be possible to replace v by u, for otherwise there would be a bad cycle, i.e. an infinite full path satisfying $\neg\Phi$ in the original M''. Continuing in this fashion, we get a structure M with no duplicates. It is isomorphic to a structure contained in T with the desired properties, i.e.
$M, s_0, s_1, ...s_k \models P \wedge A(\Phi \vee FR)$.

($\Leftarrow$) By definition, any model M contained in T is also generated by T. $\square$

3.6 Psuedo-Models

The Linear Size Model Theorem also suggests the following:

Definition. Let $T = (D, C, R_{DC}, R_{CD0}, R_{CD1}, L_T)$ be the transition diagram of a tree automaton $\mathcal{A}$, let s be a state of $\mathcal{A}$ and let p be a formula.

- We say that diagram T at state s is a *psuedo-model* of p (or that p is *satisfiable* in T at s) and write $T, s \,\|\!-_{con}\, p$ iff there exists a structure M contained in T such that $M, s \models p$. We also write $p^{T,con}$ for $\{s \in T : T, s \|\!-_{con}\, p\}$.
- We say p is *generable* in T at s and write $T, s \|\!-_{gen}\, p$ iff there exists a structure M generated by T such that $M, s \models p$. We also write $p^{T,gen}$ for $\{s \in T : T, s \|\!-_{gen}\, p\}$.
- We say p is *true* (or *modelled*) in T at s, considered simply as a structure with state set $D \cup C$ and transition relation $R_{CD} \cup R_{DC0} \cup R_{DC1}$, iff $T, s \models p$. We also write $p^{T,mod}$ for $\{s \in T : T, s \models p\}$.

Remark: For a proposition symbol P we have $P^{T,mod} = P^{T,con} = P^{T,gen}$.

Our overall approach to testing nonemptiness can now be summarized in the following rephrasing of the Linear Size Model Theorem.

Theorem 3.4 Automaton $\mathcal{A}$ with diagram T and pairs acceptance condition Φ is nonempty iff $q_0 \in A\Phi^{T,con}$.

We will check nonemptiness by calculating $A\Phi^{T,con}$ by a process of "psuedo-model checking" (cf. [CE81], [CES86]).

4 Complexity of Pairs Tree Automata

In this section we prove that nonemptiness of pairs tree automata can be tested in time polynomial in the number of states and exponential in the number of pairs, even though, as we show, the problem is NP-complete in general.

Theorem 4.1: Non-emptiness of a pairs automaton $\mathcal{A}$ having m states and n pairs can be tested in deterministic time $(mn)^{O(n)}$.

Proof. Let $\mathcal{A}$ be a tree automaton with transition diagram T of size m states and pairs acceptance condition $\Phi_\Gamma = \bigvee_{\gamma\in\Gamma} (GFQ_\gamma \wedge FGP_\gamma)$ where Γ is the index set $[1:n]$ of pairs. Here, Q_γ and P_γ stand for GREEN_γ and $\neg\text{RED}_\gamma$ resp. When Γ is understood from context we can drop it and write just Φ for Φ_Γ. We also let $\Phi_{-\gamma}$ denote $\Phi_{\Gamma\setminus\{\gamma\}}$.

The basic idea of the algorithm is to inductively compute the set of states in T at which $A\Phi$ is satisfiable, viz. $A\Phi^{T,con} = \{s \in T : T, s\|\!-_{con}\, A\Phi\}$. For this purpose, we use the fixpoint characterization $\mu Y.\tau(Y)$ of $A\Phi$ from Lemma 4.2 below, and evaluate it iteratively using the Tarski-Knaster Theorem specialized to psuedo-model checking in Lemma 4.3 below. To compute each $Y^{i+1} = \tau(Y^i)$ we evaluate the body $\tau(Y)$ compositionally[5] using Lemmas 4.2 through 4.8 as appropriate. This in turn entails the recursive calculation of (essentially) $A\Phi_{-\gamma}$ with one fewer pair. The recursion terminates when $\Phi_{-\gamma} \equiv false$ has 0 pairs, in which case Lemma 4.4 is used instead of Lemma 4.3. For technical reasons, we actually psuedo-model check a modified pairs condition of the form $A(\Phi \vee FR)$, which specializes to the ordinary pairs conditions when $R \equiv false$.

If each OR-node in T had a unique successor then the pseudo-model checking of $A(\Phi \vee FR)$ would simply amount to model checking in the Mu-calculus [EL86]. But in general, each OR-node has more than one successor. Therefore, our algorithm must simultaneously (i) exhaust the search space of all Structures contained in T and (ii) check if one of these structures is a model of the pairs-condition. The Lemmas 4.2-4.8 below permit both steps to be done together, thereby performing the desired pseudo-model checking. The proofs of the Lemmas are given in the appendix A.

Lemma 4.2 (Fixpoint Characterization for Modified Pairs Condition)
Let $\Phi = \vee_\gamma(\overset{\infty}{G}P_\gamma \wedge \overset{\infty}{F}Q_\gamma)$ denote the pairs condition where $\Gamma = [1:n]$; let R be propositional.
Then $A(\Phi \vee FR) \equiv \mu Y.\tau(Y)$ where
$\tau(Y) = R \vee \vee_\gamma AX_s A(g_\gamma U_w R)$ and
$g_\gamma = A(\Phi_{-\gamma} \vee F(R \vee Q_\gamma)) \wedge (P_\gamma \vee Y) \wedge AX_s true$.

Lemma 4.3 (Psuedo-Model Checking via Tarski-Knaster Approximation) [6]
Set $\mathsf{Y}^0 := false^{T,con}$; set $\mathsf{Y}^{i+1} := \tau(Y^i)^{T,con}$.
Then $A(\Phi \vee FR)^{T,con} = \mu Y.\tau(Y)^{T,con} = \bigcup_{i\le|T|} \mathsf{Y}^i$.

Lemma 4.4 (Psuedo-Model Checking Inevitability)
$AFR^{T,con} = (\mu V.R \vee EX_s AX_s V)^{T,mod}$

Observation 4.5 (Psuedo-Model Checking of Disjunctions)
$(R \vee \bigvee_\gamma (AX_s A[g_\gamma U_w R]))^{T,con} = R^{T,con} \cup \bigcup_\gamma (AX_s A[g_\gamma U_w R])^{T,con}$.

[5] i.e., by induction on formula structure
[6] It is to be understood below that "set $\mathsf{Y}^{i+1} := \tau(Y^i)^{T,con}$", for example, means to assign to the set Y^{i+1} exactly the set of states in $\tau(Y^i)^{T,con}$ and label the resulting states in the set Y^{i+1} with the symbol Y^{i+1}.

Observation 4.6 (Psuedo-Model Checking of Nexttime)
$(AX_s A[g_\gamma U_w R])^{T,con} = \{s \in T : T, s \models EX_s AX_s W\}$
where $\mathsf{W} = A[g_\gamma U_w R]^{T,con}$.

Definition. Let U be a transition diagram for an automaton and let Z be a set of OR-nodes of U. We define the AND/OR-graph denoted $U|\mathsf{Z}$, called U *restricted to* Z, to be the result of deleting from U all OR-nodes not in Z and incident arcs, and then deleting all AND-nodes which do not have all successors in Z, along with all incident arcs. By a convenient abuse of notation, we shall write Z for $U|\mathsf{Z}$ when it is clear that an AND/OR-subgraph of U is intended. In particular, if f is a temporal formula, we write $f^{\mathsf{Z},con}$ for $f^{U|\mathsf{Z},con}$.

Lemma 4.7 (Psuedo-Model Checking of Weak Until)
Set $\mathsf{Z}^0 := true^{T,con}$; set $\mathsf{Z}^{i+1} := ((g_\gamma \wedge AX_s Z^i) \vee R)^{\mathsf{Z}^i,con}$.
Then for some least k, $\mathsf{Z}^k = \mathsf{Z}^{k+1}$ is the fixpoint of the descending chain $\mathsf{Z}^i \supseteq \mathsf{Z}^{i+1}$
and $A[g_\gamma U_w R]^{T,con} = \mathsf{Z}^k$.

Lemma 4.8 (Psuedo-Model Checking of Conjunctions in g_γ)
$g_\gamma^{T,con} = A(\Phi_{-\gamma} \vee F(R \vee Q_\gamma))^{T,con} \cap (P_\gamma \vee Y)^{T,con} \cap AX_s true^{T,con}$.

Finally, we analyze the complexity as follows: Let $Com(T,f)$ stand for the complexity of computing $\{s|T, s \Vdash f\}$. For size of the transition diagram $|T| = m$, and number of pairs $|\Gamma| = n$, let $C(m,n)$ denote $Com(T, A(\Phi_\Gamma \vee R))$.

$$\begin{aligned}
Com(T, A(\Phi_\Gamma \vee FR)) &\leq O(m) \cdot Com(T, Y^i) \\
Com(T, Y^i) &\leq n \cdot Com(T, R \vee AX_s A(g_\gamma U_w R)) \\
Com(T, R \vee AX_s A(g_\gamma U_w R)) &\leq O(m) + O(m) \cdot Com(Z^i, g_\gamma \wedge AX_s Z^i) \\
Com(Z^i, g_\gamma \wedge AX_s Z^i) &\leq O(m) + Com(U, A(\Phi_{\Gamma/\{\gamma\}} \vee FR'))
\end{aligned}$$

where $|Z^i|, |U| \leq |T|$. Thus, for some constant $k > 0$ we have

$$\begin{aligned}
C(m,n) &\leq kmn(km + km(km + C(m, n-1))) \\
&= kmn(km + k^2m^2 + kmC(m, n-1)) \\
&= k^2m^2n + k^3m^3n + k^2m^2nC(m, n-1) \\
&\leq 2k^3m^3n + k^2m^2nC(m, n-1).
\end{aligned}$$

The above accounts for the cost for 1 or more pairs ($n > 0$). Lemma 4.4 implies $n = 0$ pairs can be handled in $O(m)$ time. Hence, for some sufficiently large constant $c \geq 2k^3$ we have

$$\begin{aligned}
C(m,n) &\leq cm^3n + cm^2nC(m, n-1) \\
C(m,0) &\leq cm.
\end{aligned}$$

We must thus solve a recurrence of the form

$$\begin{aligned}
C(m,n) &= x + yC(m, n-1) \\
C(m,0) &= z.
\end{aligned}$$

which is readily expanded to show that its solution is $x(y^n - 1) + y^n z \leq y^n(x + z)$. It follows that $C(m,n)$ is at most

$$\begin{aligned}
&= (cm + cm^3n)(cm^2n)^n \\
&\leq (dm^3n)(dm^2n)^n, \text{ for some constant } d > 2c \\
&= d^{n+1}m^{2n+3}n^{n+1} \\
&\leq n^{n+1}m^{2n+3}n^{n+1}, \text{ for all } n \geq d \\
&\leq (mn)^{2n+3} \\
&= (mn)^{O(n)}, \text{ for all } n, m \geq 1.
\end{aligned}$$

Note that m is the size of the transition diagram, which for an automaton on binary trees can be cubic in the number of states, m_0, i.e. $m = O(m_0^3)$. As a function of the number of states, however we still get the same order of growth, i.e. $(mn)^{O(n)} = (m_0^3 n)^{O(n)} = (m_0 n)^{3 \cdot O(n)} = (m_0 n)^{O(n)}$. In general, for k-ary trees, the size of the transition diagram m can be $O(m_0^{k+1})$. Thus we get time $(m_0 n)^{(k+1)O(n)}$. This is still $(m_0 n)^{O(n)}$ for fixed k. Moreover, if k grows linearly with n the bound is still $(m_0 n)^{O(n^2)}$ which is sufficient for obtaining the complexity bounds on logics of programs of Section 6. □

Theorem 4.9 Pairs tree automaton non-emptiness is NP-complete.
Proof. Pairs automaton non-emptiness was shown to be in NP in [Em85]: a nondeterministic Turing machine can guess a linear size model M contained in the automaton diagram and verify that it satisfies the acceptance condition using the model checking alogithm for fairness of [EL85] (cf. [VS85]).

To show NP-hardness we reduce from 3-SAT. Let f be a 3-CNF formula with m clauses $C_1, C_2, \ldots, C_m$ over n variables $v_1, \ldots, v_n$. We will reduce satisfiability of f to the nonemptiness problem of a pairs tree automaton $\mathcal{A}$ with number of states and pairs polynomial in $|f|$.

We will describe $\mathcal{A}$ in terms of its AND/OR-graph transition diagram as usual and pairs acceptance condition.[7] Since we are only concerned about the nonemptiness problem, $\mathcal{A}$ is assumed to have a one symbol alphabet. Corresponding to each clause C_i of f, $\mathcal{A}$ will have an OR-node of the same name, C_i. For each possible literal x, i.e., for each variable v_j and its negation $\neg v_j$, there is an AND-node of the same name, x. If clause C_i contains literal x then there is an edge $C_i \to x$ in the diagram. If clause C_i contains the literal $\neg x$ then there is an edge $x \to C_i$ in the diagram. Here we identify $\neg\neg x$ with x. We also let S_0 be the start state OR-node with a single AND-node successor which has as its successors $C_1, \ldots, C_m$. Finally, for each literal x there is a pair of lights $(GREEN_x, RED_x)$ such that $GREEN_x$ colors AND-node x and RED_x colors AND-node $\neg x$. Let Φ be the corresponding pairs condition.

The basic idea is that a structure M contained in (the diagram of) $\mathcal{A}$ specifies a choice of literal for each clause. From these literals we can try to recover a satisfying truth assignment for f, and conversely. However, there may be conflicts with one clause using x, another $\neg x$. The following argument shows that there is a conflict-free choice of literals when there is an M satisfying $A\Phi$.

The following claims are equivalent:

(i) $\mathcal{A}$ is nonempty

(ii) There is a structure M contained in $\mathcal{A}$, accepted by $\mathcal{A}$ so that $M, S_0 \models A\Phi$.

[7] We will actually describe the diagram of an automaton on trees of arity at most m, where, moreover, the propositions (or "colors") defining acceptance label AND-nodes rather than OR-nodes. It is not difficult to show that it can be converted into an "equivalent, ordinary" automaton. Here "equivalent" means that the original automaton accepts iff the derived automaton accepts. Conversion to a binary tree automaton is effected by inserting "dummy" nodes in the transition diagram to reduce fan-in m to fan-in 2. This causes a linear blowup in size. The acceptance condition colors can be moved to the OR-nodes by having each derived node be of the form: (OR-node, AND-node predecessor causing transition). This can cause a quadratic blowup in size.

(iii) There exists $H = ao(M)$, where M is contained in $\mathcal{A}$, such that in H, each C_j has an edge to some x_l, and no other C_k has an edge to the negation of x_l.

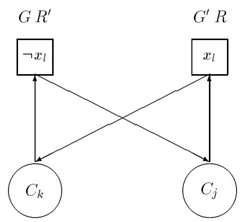
(iv) f is satisfiable.

Claims (i) and (ii) are equivalent by the Linear Size Model Theorem. The equivalence of (iii) and (iv) is immediate from the construction of $\mathcal{A}$. The equivalence of (ii) and (iii) follows from a slightly sharper equivalence. Let $H = ao(M)$ where M is contained in $\mathcal{A}$. Then these are equivalent:

(ii)' $M, S_0 \models A\Phi$

(iii)' In H, each C_j has an edge to some x_l, and no other C_k has an edge to the negation of x_l.

Condition (iii)' implies that for every green light labelling a node in H its corresponding red light does not occur in H. Hence, along every infinite path some green light appears infinitely often but its corresponding red light never appears, and Φ holds along the path. Condition (ii)' follows. Now assume (ii)'. Then M is total is is H. It must be that (iii)' holds of H. Otherwise, there is a C_j with an edge to x_l, and a C_k with an edge to $\neg x_l$. Then we have the following subgraph in H



Here $(G,\ R)$ and $(G',\ R')$ are the pairs corresponding to the literals x_l and $\neg x_l$ resp. The arcs $x_l \rightarrow C_k$ and $\neg x_l \rightarrow C_j$ exist by definition of $\mathcal{A}$, considering that $\neg x_l$ is in C_k and x_l is in C_j. The above subgraph generates a path which has for all green lights flashing infintely often, its corresponding red light flashing infinitely often too. But then there would be a "bad" path in M violating Φ, contrary to our assumption of (ii)'. The equivalence of (ii) and (iii) follows. This completes the reduction. $\square$

5 Complexity of Complemented Pairs Tree Automata

In this section, we will show that there is also a deterministic algorithm to test nonemptiness of a complemented pairs tree automaton (cf. [ST81]) which runs in time $(mn)^{O(n)}$, where m is the number of states of the automaton, and n is the number of pairs in its acceptance condition. Moreover, we will show that testing nonemptiness of such automata is coNP-complete.

The key idea is that for a tree automaton $\mathcal{A}$ over a one symbol alphabet, we can define its dual tree automaton $\tilde{\mathcal{A}}$ such that $\mathcal{A}$ is nonempty iff $\tilde{\mathcal{A}}$ is empty. The dual is essentially obtained by swapping AND-nodes with OR-nodes and complementing the acceptance condition. Since we view the transition diagram of a nondeterministic tree automaton as a bipartite AND-OR graph, the dual is also a bipartite AND-OR graph, and hence can be viewed as

a nondeterministic automaton. Moreover, the dual of a pairs automaton is a complemented pairs automaton, and vice-versa. Muller and Schupp also define a dual automaton in [MS84], but for a nondeterministic automaton their dual automaton is in general an alternating automaton. For the purpose of checking nonemptiness of automata over one symbol alphabets, the dual defined in this new way suffices.

For an automaton $\mathcal{A}$ and its dual $\tilde{\mathcal{A}}$, we are interested in showing that $\mathcal{A}$ is non-empty iff $\tilde{\mathcal{A}}$ is empty. In other words, we must show that

$$\exists \text{ Run } \rho \text{ of } \mathcal{A}\ \forall \text{ paths in } \rho \quad \Phi_{\mathcal{A}} \tag{1}$$

$$\equiv \quad \neg\ \exists \text{ Run } \rho \text{ of } \tilde{\mathcal{A}}\ \forall \text{ paths in } \rho \quad \Phi_{\tilde{\mathcal{A}}} \tag{2}$$

where $\Phi_{\mathcal{A}}$ is the acceptance condition of $\mathcal{A}$ and $\Phi_{\tilde{\mathcal{A}}} \equiv \neg\Phi_{\mathcal{A}}$ is the acceptance condition of $\tilde{\mathcal{A}}$. We can argue this informally as follows. By expanding (1) we get that

$$(1) \quad \equiv \quad \forall! d_0 \exists c_0 \forall d_1 \exists c_1 \forall d_2 \exists c_2 \ldots\ d_0 c_0 d_1 c_1 d_2 c_2 \ldots \models \Phi_{\mathcal{A}}$$

meaning that for the unique start OR-node d_0 of $\mathcal{A}$ there exists an AND-node successor c_0 such that for all OR-node successors d_1 of it there exists an AND-node successor c_1, etc., *ad infinitum* such that $d_0 c_0 d_1 c_1 d_2 c_2 \ldots \models \Phi_{\mathcal{A}}$. Similarly, by expanding (2) we get that

$$(2) \quad \equiv \quad \neg\forall! e_0 \exists! d_0 \forall c_0 \exists d_1 \forall c_1 \exists d_2 \forall c_2 \ldots\ e_0 d_0 c_0 d_1 c_1 d_2 c_2 \ldots \models \neg\Phi_{\mathcal{A}}$$

where e_0 is the unique "dummy" start state of $\tilde{\mathcal{A}}$ all of whose successors are d_0. (This is explained in more detail below.) Applying the well known quantifier negation law to the infinite string of quantifiers for (2) we get after driving the negation inside that

$$(2) \quad \equiv \quad \exists! e_0 \forall! d_0 \exists c_0 \forall d_1 \exists c_1 \forall d_2 \exists c2 \ldots\ e_0 d_0 c_0 d_1 c_1 d_2 c_2 \ldots \models \Phi_{\mathcal{A}}$$

Since the truth of $\Phi_{\mathcal{A}}$ is oblivious to the initial e_0 we should be able to conclude that $(1) \equiv (2)$ as desired. But the quantifier negation law for infinite strings of quantifiers is known to contradict the Axiom of Choice in general. So to formally prove the above, we show that the set of paths $\Phi_{\mathcal{A}}$ and its complement are "nice", in the sense that they are finite automaton definable (and hence Borel), and conclude the desired equivalence using Martin's theorem on determinacy of infinite Borel games (cf. [Ma75]).[8]

The above notions are formalized in the sequel.

Definition. Let λ denote the empty sequence. For an infinite sequence $x_0 x_1 \ldots$ let x^i be the suffix $x_i x_{i+1} \ldots$. In particular, $tail(x)$ stands for x^1. If $x = x_0 x_1 \ldots$ and $y = y_0 y_1 \ldots$ are two infinite sequences then let $x\hat{\ }y$ ("*zip*") denote the sequence $x_0 y_0 x_1 y_1 \ldots$ We let $\lambda^\omega = \lambda^n = \lambda$, and $x\lambda = x$. Then we can define zip of two finite/infinite sequences by appending finite sequences with λ^ω.

Definition. We are given an automaton $\mathcal{A}$ over a one symbol alphabet with acceptance condition $\Phi_{\mathcal{A}}$. Without loss of generality, we may stipulate that each node of its transition diagram has exactly two successors.[9] Thus we may assume $\mathcal{A}$ is of the form (D, C, R, d_0, L) with OR-node set D, AND-node set C, start OR-node d_0, labelling L, and transition relation $R = R_{DC} \cup R_{CD0} \cup R_{CD1}$. Since each node has exactly two successors we may view R as a function so that $R(d, i) = c$ indicates that c is the AND-node successor of OR-node d of index $i \in \{0, 1\}$

[8] Since $\Phi_{\mathcal{A}}$ is finite automaton definable, a weaker result of Büchi-Landweber [BL69], showing determinacy of such games, will do.

[9] The argument generalizes in a straightforward way to k successors.

and $R(c, j) = \mathrm{d}$ indicates that d is the OR-node successor of AND-node c in direction $j \in \{0, 1\}$. We also assume that the labelling L assigning "colors" associated with $\Phi_{\mathcal{A}}$ to each node has been extended to AND-nodes, so that each AND-node is colored exactly as is its unique OR-node parent.[10]

We shall define the dual automaton $\tilde{\mathcal{A}}$ essentially by swapping OR-nodes and AND-nodes in $\mathcal{A}$. However, we shall have to do a bit more to ensure that the result meets the technical definition of being an automaton as we have defined them to be.

We first define $\hat{\mathcal{A}}$ with diagram of the form $(\hat{D}, \hat{C}, \hat{R}, \hat{d}_0, \hat{L})$ where $\hat{D} = C \cup \{e_0\}$, $\hat{C} = D$, $\hat{R}(e_0, 0) = \hat{R}(e_0, 1) = d_0$ with $\hat{R}(b, i) = R(b, i)$ for all $b \in C \cup D$ and $i \in [0, 1]$ and $\hat{L}$ is the same as L but it includes e_0 in its domain, assigning it the empty set of color propositions. We see that $\hat{\mathcal{A}}$ is the result of swapping OR-nodes and AND-nodes from $\mathcal{A}$ and adding a new dummy start state e_0 both of whose successors are d_0, start state of $\mathcal{A}$. $\hat{\mathcal{A}}$ is almost what we want for the dual except that its AND-nodes, which were OR-nodes of $\mathcal{A}$, are not guaranteed to have unique predecessors. Technically, this violates the definition of an automaton.

Therefore, to get $\tilde{\mathcal{A}}$ from $\hat{\mathcal{A}}$ we must create duplicates of the AND-nodes in $\hat{C}$. In other words, we locally unravel $\hat{\mathcal{A}}$: if $\hat{d} \to \hat{c}$ and $\hat{c} \to \hat{d}'$ appear in $\hat{\mathcal{A}}$ then $\hat{d} \to (\hat{d}, \hat{c})$ and $(\hat{d}, \hat{c}) \to \hat{d}'$ appear in $\tilde{\mathcal{A}}$, with the AND-node $(\hat{d}, \hat{c})$ of $\tilde{\mathcal{A}}$ labelled exactly as is AND-node $\hat{c}$ of $\hat{ca}$. The result is the underlying diagram of $\tilde{\mathcal{A}}$. Formally, we let $\tilde{A}$ have diagram $(\tilde{D}, \tilde{C}, \tilde{R}, \tilde{d}_0, \tilde{L})$. Here $\tilde{D} = \hat{D}$ and $\tilde{d}_0 = \hat{d}_0$. We define $\tilde{C} \subseteq \hat{D} \times \hat{C}$ such that if AND-node $\hat{c}$ has OR-node predecessors $\hat{d}_1, \ldots, \hat{d}_k$ in $\hat{\mathcal{A}}$ then AND-nodes $(\hat{d}_1, \hat{c}), \ldots, (\hat{d}_k, \hat{c})$ are in $\tilde{C}$. If $\hat{R}(\hat{d}, i) = \hat{c}$ then $\tilde{R}(\hat{d}, i) = (\hat{d}, \hat{c})$, for $i \in \{0, 1\}$. If $\hat{R}(\hat{c}, j) = \hat{d}$ then $\tilde{R}((\hat{d}_l, \hat{c}), j) = \hat{d}$ for $j \in \{0, 1\}$ and where $\hat{d}_l$ is any predecessor of $\hat{c}$ in $\hat{\mathcal{A}}$. Finally, $\tilde{L}(\hat{d}) = \hat{L}(\hat{d})$ and $\tilde{L}((\hat{d}_l, \hat{c})) = \hat{L}(\hat{c})$. This transformation of $\hat{\mathcal{A}}$ into $\tilde{\mathcal{A}}$ can cause at worst a quadratic blowup in size.

Definition. Given an infinite string $z \in \{0, 1\}^\omega$, and automaton $\mathcal{A}$ with diagram (D, C, R, d_0, L) we can associate with z, by starting at d_0 and following the arc labels spelling out z, a unique infinite path through the diagram of $\mathcal{A}$ and vice-versa:

Formally, we define $path_{\mathcal{A}} : \{0, 1\}^\omega \to (DC)^\omega$
such that for $z = x\hat{\ } y$, $path_{\mathcal{A}}(z) = s\hat{\ }\, t$, where $x = x_0 x_1 \ldots$ $y = y_0 y_1 \ldots$ $s = s_0 s_1 \ldots$ $t = t_0 t_1 \ldots$,
$s_0 = d_0$ is the start node in D, $t_n = R(s_n, x_n)$, and $s_{n+1} = R(t_n, y_n)$.

Observation 5.1. $\tilde{\pi} = path_{\tilde{\mathcal{A}}}(z)$ is the same as $\hat{\pi} = path_{\hat{\mathcal{A}}}(z)$, except that each AND-node $\tilde{c}$ along $\tilde{\pi}$ is a duplicate of the corresponding AND-node $\hat{c}$ along $\hat{\pi}$ of the form $(\hat{d}, \hat{c})$ where $\hat{d}$ is the predecessor of $\hat{c}$ along $\hat{\pi}$. Therefore, $\tilde{\pi} \models \Phi_{\mathcal{A}}$ iff $\hat{\pi} \models \Phi_{\mathcal{A}}$.

Observation 5.2. $path_{\tilde{\mathcal{A}}}(x\hat{\ }y) = e_0 \cdot path_{\mathcal{A}}(y\hat{\ }tail(x))$.

This follows because, all successors of e_0 are d_0, and hence x_0 is redundant.

Definition. We now define a *two player infinite game* $\mathcal{G}$ associated with $\mathcal{A}$. There are two players I and II. I goes first and picks $x_0 \in \{0, 1\}$, then II picks some $y_0 \in \{0, 1\}$, then I picks some $x_1 \in \{0, 1\}$, and so on alternatively. The resulting infinite string $z = x_0 y_0 x_1 y_1 \ldots$ is a *play* of the game. Player I wins this particular play z if z is in the *winning set* Γ, which is defined by letting $\Gamma = \{x : x \in \{0, 1\}^\omega\ path_G(x)$ satisfies $\Phi_{\mathcal{A}}\}$; otherwise II wins the play.

A *strategy* for I is a function $f_I\ \{0, 1\}^* \to \{0, 1\}$, and a strategy for II is a function $f_{II} : \{0, 1\}^+ \to \{0, 1\}$. The family of all such strategy function will be denoted $StratI$ and $StratII$, respectively, with typical elements denoted f_I and f_{II}, respectively.

[10] Thus, if x is a path through $\mathcal{A}$, we have $x \models \Phi_{\mathcal{A}}$ iff $x|D \models \Phi_{\mathcal{A}}$ iff $x|C \models \Phi_{\mathcal{A}}$.

Any function $f\ \{0, 1\}^* \to \Sigma$, induces a map $\overline{f} : \{0, 1\}^\omega \to \Sigma^\omega$. If $z = z_0 z_1 \ldots$, and $x = x_0 x_1 \ldots$, $\overline{f}(x) = z$ such that, for all n, $z_n = f(x_0 x_1 \ldots x_{n-1})$, and $z_0 = f(\lambda)$. Similarly, a function $f : \{0, 1\}^+ \to \Sigma$, induces a map $\overline{f} : \{0, 1\}^\omega \to \Sigma^\omega$, such that $z_n = f(x_0 x_1 \ldots x_n)$.

We say that player I follows strategy f_I, if it chooses $f_I(y_0 y_1 \ldots y_{n-1})$ when II has chosen $y_0 y_1 \ldots y_{n-1}$. Similarly for Player II. If II builds up y during a play, and I follows f_I, then the resulting play is $\overline{f_I}(y)\hat{\ }y$. Similarly, if I builds up x, and II follows f_{II}, the resulting play is $x\hat{\ }\overline{f_{II}}(x)$.

We say that f_I is a *winning strategy* for I if $\forall y \in \{0, 1\}^\omega\ \overline{f_I}(y)\hat{\ }y \in \Gamma$. Similalrly, f_{II} is a winning strategy for II if: $\forall y \in \{0, 1\}^\omega\ y\hat{\ }\overline{f_{II}}(y) \notin \Gamma$.

Remark. When the transition diagram is presented as in this section, the definition of a run ρ is formulated as:

$\rho(\lambda) = d_0$
$\forall y \in \{0, 1\}^*\ \exists i \in \{0, 1\}\ \exists c \in C\ c = R(\rho(y), i)$ and $\forall j \in \{0, 1\}\ \rho(yj) = R(b, j)$

The first condition asserts that ρ annotates the root node with the start state. The second condition asserts that each node y of the tree has its successors $y0, y1$ annotated in a manner consistent with the diagram because there is an AND-node c from OR-node $\rho(y)$ to OR-nodes $\rho(y0), \rho(y1)$ in the diagram. Note also that the condition that "along all paths of a run ρ, $\Phi_{\mathcal{A}}$ holds", is now formulated as "$\forall y \in \{0, 1\}^\omega\ \ \overline{\rho}(y)$ satisfies $\Phi_{\mathcal{A}}$", since y is here spelling out a path through the tree in terms of edge labels.

Lemma 5.3. Let $\mathcal{A}$ be an automaton with acceptance condition $\Phi_{\mathcal{A}}$. Then,

$\exists$ run ρ of $\mathcal{A}$ $\forall y \in \{0, 1\}^\omega\ \overline{\rho}(y)$ satisfies $\Phi_{\mathcal{A}}$

iff

$\exists f_I\ \forall y \in \{0, 1\}^\omega\ path_{\mathcal{A}}(\overline{f_I}(y)\hat{\ }y)$ satisfies $\Phi_{\mathcal{A}}$

.

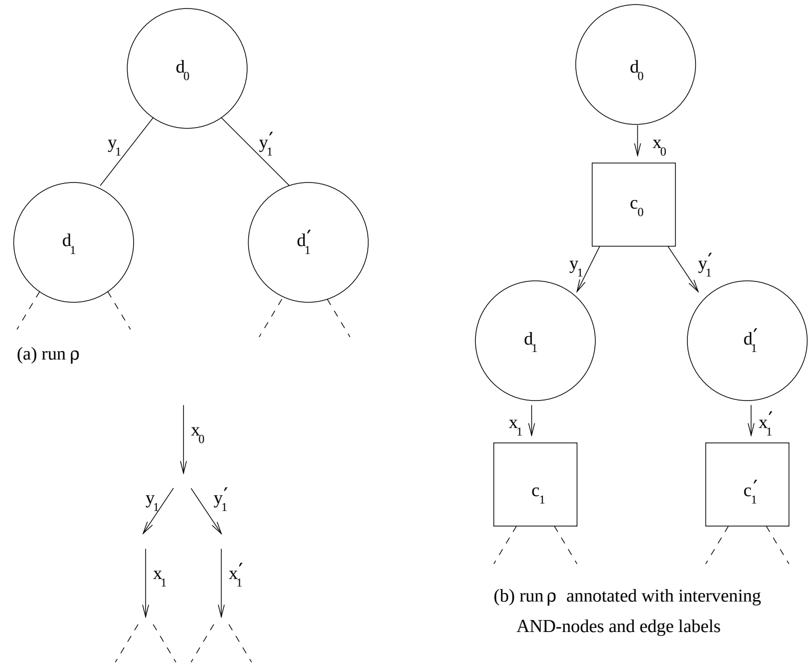
This lemma says that an accepting run defines a corresponding winning strategy and conversely. Intuitively, given a run as shown in Figure 5.1 (a), we can extend it to indicate the intermediate AND-nodes and edges, indexed by 0 or 1, between nodes, as shown in Figure 5.1 (b). These edges indicating that c_n is the x_n-successor of d_n, that d_{n+1} is the y_{n+1}-successor of c_n, and so forth, constitute an edge-labelled tree defining the strategy, as shown in Figure 5.1 (c). Conversely, given the strategy, we can view it as an edge-labelled tree, and given the start node d_0, infer the corresponding run. This argument is formalized below.

Proof. Given ρ, let $f_I(y) = \min\ \{i : \exists b\ b = R(\rho(y), i)$ and $\forall j\ \rho(yj) = R(b, j)\}$, so that f_I picks, for the sake of definiteness, the AND-node successsor of least index. The above definition is well-defined by the definition of a run.

We show that $\overline{\rho}(y)$ is exactly s, where $s\hat{\ }t = path_{\mathcal{A}}(\overline{f_I}(y)\hat{\ }x)$. By the definition of $path_{\mathcal{A}}$, $s_{n+1} = R(t_n, y_n)$, and $t_n = R(s_n, x_n)$, where $x = \overline{f_I}(y)$. We show by induction that $\rho(y_0 y_1 \ldots y_n) = s_{n+1}$.

The base case is trivial, because $\rho(\lambda) = s_0$. By induction hypothesis, $\rho(y_0 y_1 \ldots y_{n-1}) = s_n$. By definition of f_I, $t_n = g(\rho(y_0 y_1 \ldots y_{n-1}), x_n) = b$, and $\rho(y_0 y_1 \ldots y_n) = g(b, y_n) = g(t_n, y_n) = s_{n+1}$.

For the other direction, given f_I, we define ρ:
$\rho(\lambda) = s_0$,
$\rho(yj) = R(R(\rho(y), f_I(y)), j)$.
It follows from this definition of ρ and the definition of $path_{\mathcal{A}}$ that , if $s\hat{\ }t = path_{\mathcal{A}}(\overline{f_I}(y)\hat{\ }y)$, then $s = \overline{\rho}(y)$.



(a) run ρ

(b) run ρ annotated with intervening AND-nodes and edge labels

(c) corresponding strategy f_I viewed as an edge labelled tree

Figure 5.1

In the definition of $\tilde{\mathcal{A}}$, we required that the proposition labels be the same on the OR-nodes and their AND-node successors. For the dual automaton, it follows that the labels on the OR-nodes are the same as on their predecessor AND-nodes. Thus, $path_{\mathcal{A}}(\overline{f_I}(y)\hat{\ }x)$ satisfies $\Phi_{\mathcal{A}}$ iff $\overline{\rho}(y)$ satisfies $\Phi_{\mathcal{A}}$. □

Lemma 5.4. Let $\tilde{\mathcal{A}}$ be the dual automaton, which has acceptance condition $\neg\Phi_{\mathcal{A}}$. Then,

$\exists$ run ρ of $\tilde{\mathcal{A}}$ $\forall y \in \{0,1\}^{\omega}$ $\overline{\rho}(y)$ satisfies $\neg\Phi_{\mathcal{A}}$

iff

$\exists f_{II}$ $\forall x \in \{0,1\}^{\omega}$ $path_{\tilde{\mathcal{A}}}(x\hat{\ }\overline{f_{II}}(x))$ satisfies $\neg\Phi_{\mathcal{A}}$

Proof. We argue as follows:

$\exists f_{II}$ $\forall x \in \{0,1\}^{\omega}$ $path_{\mathcal{A}}(x\hat{\ }\overline{f_{II}}(x))$ satisfies $\neg\Phi_{\mathcal{A}}$
iff (since $\Phi_{\mathcal{A}}$ is oblivious to finite prefixes being altered)
$\exists f_{II}$ $\forall x \in \{0,1\}^{\omega}$ $e_0 \cdot path_{\mathcal{A}}(x\hat{\ }\overline{f_{II}}(x))$ satisfies $\neg\Phi_{\mathcal{A}}$
iff (taking f_I to be the same as f_{II}, except $f_I(\lambda)$ is defined arbitrarily)
$\exists f_I$ $\forall x \in \{0,1\}^{\omega}$ $e_0 \cdot path_{\mathcal{A}}(x\hat{\ }tail(\overline{f_I}(x)))$ satisfies $\neg\Phi_{\mathcal{A}}$
iff (by Observation 1)
$\exists f_I$ $\forall x \in \{0,1\}^{\omega}$ $path_{\tilde{\mathcal{A}}}(\overline{f_I}(x)\hat{\ }x)$ satisfies $\neg\Phi_{\mathcal{A}}$
iff (by Observation 5.2)
$\exists f_I$ $\forall x \in \{0,1\}^{\omega}$ $path_{\tilde{\mathcal{A}}}(\overline{f_I}(x)\hat{\ }x)$ satisfies $\neg\Phi_{\mathcal{A}}$
iff (by Lemma 5.3)
$\exists$ run ρ of $\tilde{A}$ $\forall x \in \{0,1\}^{\omega}$ $\overline{\rho}(x)$ satisfies $\neg\Phi_{\mathcal{A}}$

This completes the proof. □

Lemma 5.5. Γ is Borel.
Proof. We show that Γ is accepted by a deterministic string automata $\bar{A}$, with acceptance condition $\Phi_{\mathcal{A}}$. $\bar{A}$ has as its alphabet $\{0,1\}$. Let the states of $\bar{A}$ be from $S \cup T$ The transition function $\overline{\delta}$ is g itself, i.e.
$\overline{\delta}(u,i) = R(u,i)$. A routine inspection shows that $\bar{A}$ accepts Γ.

Landweber [69] showed that every set accepted by a deterministic Muller string automaton is in the Borel class $G_{\delta\omega} \cap F_{\omega\delta}$, with the usual product topology on the set of all infinite binary sequences. Since, we are interested in $\Phi_{\mathcal{A}}$ being either pairs, or complemented pairs acceptance condition, the $\omega-$string automaton with such $\Phi_{\mathcal{A}}$ are well known to be equivalent to deterministic Muller string automaton (see e.g. [La69]). □

Now, we are ready to prove the main Lemma.

Lemma 5.6. $\mathcal{A}$ is nonempty iff $\tilde{\mathcal{A}}$ is empty.

Proof. By Lemma 5.5, the acceptance condition of $\mathcal{A}$, Γ, is Borel, and hence the game $\mathcal{G}$ associated with Γ is determined by Martin's Theorem [Ma75]. Thus, I has a winning strategy iff II does not have a winning strategy. By the Lemma 5.3, Player I has a winning strategy iff $\mathcal{A}$ is nonempty. By Lemma 5.4, Player II has a winning strategy iff $\tilde{\mathcal{A}}$ is nonempty. It immediately follows then that $\mathcal{A}$ is nonempty iff $\tilde{\mathcal{A}}$ is empty. □

Given a complemented pairs automaton $\mathcal{A}$ we can construct with at most a quadratic blowup its pairs dual automaton $\tilde{\mathcal{A}}$ such that $\mathcal{A}$ is nonempty iff $\tilde{\mathcal{A}}$. Since by Theorem 4.9 of the previous section testing nonemptiness of pairs automata is NP-complete, we immediately get

Corollary 5.7. Testing nonemptiness of complemented pairs automata is co-NP-complete.

Since the quadratic reduction is deterministic, we also get

Corollary 5.8. There is a deterministic algorithm to test nonemptiness of complemented pairs automata with m states and n pairs that runs in time $(mn)^{O(n)}$.

6. Applications to Logics of Programs

Using existing reductions and our new algorithms for tree automata, we give essentially optimal algorithms for CTL^*, PDL-delta, and the Mu-calculus. The "standard" algorithms for each of these logics are of triple exponential complexity (cf. [ES84], [St81], [StE84]). The existing reductions to which we appeal, reduce satisfiability of these logics to nonemptiness of tree automata. These reductions employ determinization of a nondeterministic Büchi automaton on ω-strings. With pre-existing algorithms for testing nonemptiness of (pairs) tree automata of [Em85] (cf. [VS85]) and the McNaughton determinization construction [Mc66],[11] we can get a nondeterministic double exponential algorithm for testing (un)satisfiability of PDL-delta and the Mu-calculus. Using the new ω-string automaton determinization construct of Safra [Sa88], we can get a nondeterministic single exponential time algorithm for these two logics. As described below, using our new nonemptiness testing algorithms we get a deterministic single exponential time algorithm. The best previously existing upper bounds for these logics, viz., nondeterministic single exponential time (which amounts to deterministic double exponential time in practice), employed "Hybrid-automata" (cf. [VS85]). The Safra construction alone does not help reduce the complexity using the Hybrid Automata. For CTL^*, the existing reductions [ESi84], which use a special structure of the ω-string automata associated with the logic to determinize with only a single exponential blowup, Safra's construction provides no additional help. The previous nonemptiness algorithms give a nondeterministic double exponential time algorithm. Our new nonemptiness algorithm reduces the upper bound to deterministic double exponential time. In the sequel $exp(n)$ denotes the class of functions bounded above by $2^{q(n)}$ for some polynomial $q(n)$.

Theorem 6.1[ESi84]. There is a exp(exp($|p|$)) time reduction of a CTL^* formula p into a pairs automaton $\mathcal{A}_p$ on infinite trees with exp(exp($|p|$)) states and exp($|p|$) pairs such that p is satisfiable iff $\mathcal{A}_p$ is non-empty. □

Theorem 6.2. The satisfiability problem for CTL^* is complete for deterministic double exponential time.
Proof. Applying the nonemptiness algorithm (Theorem 4.1) to the pairs tree automaton obtained by Theorem 6.1 it follows directly that CTL^* can be decided in deterministic double exponential time. CTL^* was shown hard for double exponential time in [VS85]. □

Theorem 6.3.: Satisfiability of the Mu-calculus is in deterministic exponential time.
Proof. For a Mu-calculus formula p a complemented pairs automaton $\mathcal{A}_p$ of size exp($|p|$) and number of pairs polynomial in $|p|$ can be constructed in time exp($|p|$), such that p is satisfiable iff $\mathcal{A}_p$ is nonempty. This reduction follows by the technique described in [StE84], but using Safra's construction instead of McNaughton's construction. The upper bound follows by using the nonemptiness algorithm for complemented pairs automata (Corollary 5.8). □

PDL-delta has a linear blowup translation into the Mu-calculus ([EL86], [Ko83]) (provided we allow consolidation of common subformulae). Hence, we have

Theorem 6.4. Satisfiability of PDL-delta is in deterministic exponential time.

[11] Here we also use our Lemma 5.6.

Remark. A direct algorithm, similar to the one mentioned for the Mu-calculus, can also be given (See [St81], [VS85]).

Because of the known deterministic exponential time lower bound for PDL (FL[79]), it follows that the above algorithms for the Mu-calculus and PDL-delta are essentially optimal (i.e. upto polynomial blowup). Moreover, we have

Corollary 6.5 Satisfiability of the Mu-calculus and PDL-delta are complete for deterministic exponential time.

7. Conclusion

We have investigated the complexity of testing nonemptiness of finite state automata on infinite trees. For the classical pairs acceptance condition of Rabin [Ra69] we show the problem is NP-complete, while for the complemented pairs condition of Streett [St81] we show the problem is co-NP-complete. In both cases, we are still able to give a deterministic algorithm that runs in time polynomial in the number of states but exponential in the number of pairs. These nonemptiness algorithms improve previous results in the literature and permit us to give exponentially improved, essentially tight, upper bounds on the complexity of testing satisfiability for a number of important modal logics of programs including CTL*, PDL-delta, and the propositional mu-calculus. Moreover, we believe that the technique of psuedo-model checking may be useful in connection with other types of automata (cf. [EJ91]).

Among related work we mention the following. Historically, Rabin [Ra70] gave an exponential time algorithm for pairs tree automaton nonemptiness, but did not perform a multi-parameter analysis or provide a lower bound. Hossley and Rackoff [HR72] gave an elegant reduction to the nonemptiness problem for automata on finite trees but the complexity stated for their algorithm was triple exponential. More recently, subsequent to the appearance of the preliminary version of this work in [EJ88], Pnueli and Rosner [PR89a] independently developed a different nonemptiness algorithm for pairs tree automata providing essentially the same upper bound as ours and intended for program synthesis applications; however, they did not consider lower bounds. Their work was extended to a corresponding upper bound for complemented pairs tree automata in [PR89b]. A control-theoretic view of tree automata is given in [Thi91] and [TW91], while an authoritative survey of automata on infinite objects is presented in [Tho90].

8 Appendix A

Proof of Lemma 4.2. We will establish the dual claim. Note that given a pairs condition $\Phi = \bigvee_\gamma(\overset{\infty}{G}P_\gamma \wedge \overset{\infty}{F}Q_\gamma)$, the corresponding "complemented" pairs condition $\hat{\Phi} = \wedge_\gamma(\overset{\infty}{F}P_\gamma \vee \overset{\infty}{G}Q_\gamma)$ is intended to capture the dual property. Interpreted over fullpaths in total structures, which must be infinite paths, $\hat{\Phi} \equiv \tilde{\Phi}$, the actual dual of Φ, i.e. $\neg\Phi(\neg P_1, ...\neg P_\gamma, \neg Q_1, ...\neg Q_\gamma)$. However, over partial structures, as we are permitting, finite fullpaths are allowed, and we do not necessarily have the equivalence of $\hat{\Phi}$ and $\tilde{\Phi}$. Rather,
$\Phi \equiv \Phi \wedge inf \equiv \vee_\gamma(FGP_\gamma \wedge GFQ_\gamma) \wedge inf$.
So,
$\tilde{\Phi} \equiv \wedge_\gamma(GFP_\gamma \vee FGQ_\gamma) \vee fin \equiv (\wedge_\gamma(GFP_\gamma \vee FGQ_\gamma) \wedge inf) \vee fin \equiv \hat{\Phi} \vee fin$. We thus have that $A(\Phi \vee FR) \equiv A((\Phi \wedge inf) \vee FR)$, so that the dual $E(\tilde{\Phi} \wedge GR) \equiv E((\hat{\Phi} \vee fin) \wedge GR)$.

We must show that $E(\tilde{\Phi} \wedge GR) \equiv \nu Y.\tilde{\tau}(Y)$
where $\tilde{\tau}(Y) = R \wedge \wedge_\gamma EX_w E[RU_s R \wedge (E(\tilde{\Phi}_{-\gamma} \wedge G(R \wedge Q_\gamma)) \vee (P_\gamma \wedge Y) \vee AX_w false)]$. In the sequel, let M be an arbitrary structure, which is understood, but not explicitly mentioned.

We first show that $E(\tilde{\Phi} \wedge GR)$ is a fixpoint of $\tilde{\tau}(Y)$, i.e., $E(\tilde{\Phi} \wedge GR) \equiv \tilde{\tau}(E(\tilde{\Phi} \wedge GR))$ is valid. For the forward ($\Rightarrow$) direction, assume $s_0 \models E(\tilde{\Phi} \wedge GR)$. Then there is a fullpath x starting at s_0 satisfying $\tilde{\Phi} \wedge GR$, which is equivalent to $(\hat{\Phi} \vee fin) \wedge GR$. By virtue of x and by definition of $\tilde{\Phi}$, for each γ, we also have $s_0 \models R \wedge EX_w E[RU_s R \wedge (AX_w false \vee \ (P_\gamma \wedge E(\tilde{\Phi} \wedge GR)) \vee \ E(\tilde{\Phi}_{-\gamma} \wedge G(R \wedge Q_\gamma)))]$, where the $AX_w false$ disjunct ultimately obtains if the fullpath x is finite, the $(P_\gamma \wedge E(\tilde{\Phi} \wedge GR))$ disjunct ultimately obtains if P_γ occurs along the fullpath x, and the $E(\tilde{\Phi}_{-\gamma} \wedge G(R \wedge Q_\gamma))$ disjunct ultimately obtains otherwise, since if P_γ never occurs then eventually Q_γ must always hold. Hence, s_0 satisfies all conjuncts of $\tilde{\tau}(E(\tilde{\Phi} \wedge R))$, and $s_0 \models \tilde{\tau}(E(\tilde{\Phi} \wedge GR))$. For the converse ($\Leftarrow$) direction, assume $s_0 \models \tilde{\tau}(E(\tilde{\Phi} \wedge GR))$. Pick an arbitrary γ. We have by definition of $\tilde{\tau}(Y)$ that $s_0 \models R \wedge EX_w E[RU_s R \wedge (AX_w false \vee \ (P_\gamma \wedge E(\tilde{\Phi} \wedge GR)) \vee \ E(\tilde{\Phi}_{-\gamma} \wedge G(R \wedge Q_\gamma)))]$. Whichever disjunct ultimately obtains, it follows that $s_0 \models E(\tilde{\Phi} \wedge GR)$. Thus $E(\tilde{\Phi} \wedge GR)$ is a fixpoint of $\tilde{\tau}(Y)$ as desired.

We now show that $E(\tilde{\Phi} \wedge GR)$ is the greatest fixpoint of $\tilde{\tau}(Y)$. Suppose $Z \equiv \tilde{\tau}(Z)$ is an arbitrary fixpoint. We will show that $Z \Rightarrow E(\tilde{\Phi} \wedge GR)$ is valid.

For any state s such that $s \models Z$, since Z is a fixpoint, we have two cases by analyzing $Z \equiv \tilde{\tau}(Z)$.

(a) For some index γ,
$s \models R \wedge EX_w E[RU_s R \wedge (E(\Phi_{-\gamma} \wedge G(R \wedge Q_\gamma)) \vee AX_w false)]$ or

(b) For all indices γ,
$s \models R \wedge EX_w E[RU_s R \wedge (P_\gamma \wedge Z)]$.

If (b) holds then we either have the strengthened

(b)' For all indices γ,
$s \models R \wedge EX_s E[RU_s R \wedge (P_\gamma \wedge Z)]$, or

(b)" For some index γ,
$s \models R \wedge EX_w false$.

But case (b)" implies case (a). So either (a) or (b)' must obtain.

Observation: if state s can reach state t by a finite path y where R holds everywhere along y, then if case (a) applies to t, it applies to s also.

Now let s_0 be an arbitrary state such that $s_0 \models Z$. If case (a) applies to s_0 then $s_0 \models E((\hat{\Phi} \vee fin) \wedge GR)$ and we are done. Otherwise, case (b)' applies. We will show how to use the recursion $Z \equiv \tilde{\tau}(Z)$ infinitely many times to construct an infinite path satisfying $E(\tilde{\Phi} \wedge GR)$.

Since (b)' applies to s_0, there exists a path from s_0 to some state s_1 of the form $s_0 = t_0, t_1, \ldots, t_k = s_1$ $(k \geq 0)$ such that $s_1 \models P_1 \wedge Z$ and for all $i \in [0:k]$, $t_i \models R$.

We now do case analysis on s_1. By the Observation, we see that case (a) cannot apply to s_1, for if it did, it would apply to s_0 also. So case (b)' applies to s_1, and there exists a path from s_1 to some state s_2 of the form $s_1 = u_0, u_1, \ldots, u_l$ $(l \geq 0)$ such that $s_2 \models P_2 \wedge Z$ and for all $j \in [0:l]$, $u_j \models R$.

Continuing in this fashion , we construct an infinite path x of the form $s_0 \ldots s_1 \ldots s_2 \ldots s_i \ldots$ along which R always holds and which successively visits states where $P_1, P_2, \ldots, P_m, P_1, P_2, \ldots, P_m, \ldots$, etc. hold in succession. Technically, we have $x \models GR$ and for each i, $s_i \models P_{i \,\mathbf{modd}\, m}$ where i **modd** m is defined to be i **mod** m (the remainder of i on division by m) if $i \,\mathbf{mod}\, m \neq 0$ and to be m if $i \,\mathbf{mod}\, m = 0$. □

Proof of Lemma 4.3. We have $\bigcup_i \tau^i(false)^{T,con} = A(\Phi \vee FR)^{T,con} = A(\Phi \vee FR)^{T,gen} = \bigcup_i \tau^i(false)^{T,gen}$, where the first and third equalities follow from the Tarksi-Knaster theorem together with the disjunctivity of con and gen, while the second equality follows from the Generalized Linear Size Model Theorem.

It will thus be sufficient to show that for all i,

$$(*) \quad \tau^i(false)^{T,con} \subseteq \mathsf{Y}^{\mathsf{i}} \subseteq \tau^i(false)^{T,gen}$$

We do this by induction on i.

The base case $i = 0$ is immediate since $\mathsf{Y}^{\mathsf{0}} = false^{T,con}$.

For the induction step, we assume that (*) holds for i and argue that it holds for $i + 1$.

To establish the first containment $\tau^{i+1}(false)^{T,con} \subseteq \mathsf{Y}^{\mathsf{i+1}}$, assume $s \in \tau(\tau^i(false)^{T,con}$. Then $M, s \models \tau(\tau^i(false))$ for some M contained in T. It follows, as justified below, that $M, s \models \tau(Y^i)$, from which we conclude that $s \in \tau(Y^i)^{T,con} = \mathsf{Y}^{\mathsf{i+1}}$.

The key step to be justified is that $\tau^i(false)^M \subseteq (Y^i)^M$. Note that for any temporal formula f and for any M contained in T, $f^M \subseteq f^{T,con} \cap M$. And we have by induction hypothesis that $\tau^i(false)^{T,con} \subseteq (Y^i)^{T,con}$. So $\tau^i(false)^M \subseteq \tau^i(false)^{T,con} \cap M \subseteq (Y^i)^{T,con} \cap M = (Y^i)^M$.

To establish the second containment $\mathsf{Y}^{\mathsf{i+1}} \subseteq \tau^{i+1}(false)^{T,gen}$, assume $s \in \mathsf{Y}^{\mathsf{i+1}} = \tau(Y^i)^{T,con}$. There exists M contained in T such that $M, s \models \tau(Y^i)$. We will construct from M a new structure M' that will be generated by T and will satisfy $\tau^{i+1}(false)$ at s.

For each state $y \in M \cap \mathsf{Y}^{\mathsf{i}}$, let M_y be a structure generated by T rooted at $\hat{y}$ (a copy of y) such that $M_y, \hat{y} \models \tau^i(false)$. Such an M_y is guaranteed to exist by the induction hypothesis. Let M' be the structure obtained from M by replacing each y by M_y, i.e., redirecting edges from predecessors of y into $\hat{y}$.

It follows that M' is generated by T by virtue of its construction from structures generated by T. It is also the case that $M', s \models \tau(Y^i)$ since each $\hat{y}$ has the same labelling with Y^i as y. Moreover, each $\hat{y}$ is the root of M_y, so $M', s \models \tau^{i+1}(false)$ and $s \in \tau^{i+1}(false)^{T,gen}$. Hence, $s \in \tau^{i+1}(false)^{T,gen}$ establishing the second containment. □

Remark. In the construction above, there may be multiple copies of nodes from T in M'; since we do not have a Generalized Linear Size Model Theorem for $\tau^j(false)$ for $j \geq 2$, it is not in general possible to get a structure contained in T. For this reason $Y^i \neq \tau^i(false)^{T,con}$. However, a more involved argument than that above can be given to show that $Y^i = \tau^i(false)^{T,gen}$.

Proof of Lemma 4.4. We use the fixpoint characterization $\mu V.(R \vee AX_s V)$ for AFR. Let $q(V) = R \vee AX_s V$ and $q'(V) = R \vee EX_s AX_s V$.

Now, $T, s \Vdash AFR$ iff $\exists$ a structure M contained in T such that $M, s \models AFR$ iff $\exists j > 0$ $\exists$ a structure M contained in T such that $M, s \models q^j(false)$.

We will argue by induction on j that

$$\exists \text{ a structure } M \text{ contained in } T \text{ such that } M, s \models q^j(false) \text{ iff } T, s \models (q')^j(false) \quad (*)$$

which will, by taking the disjunction over j, establish the Lemma.

The base case $j = 1$ is immediate since R is a proposition. So assume (*) holds for j and show that it holds for $j+1$.

($\Rightarrow$) Assume $\exists$ a structure M contained in T such that $M, s \models q^{j+1}(false)$. Then $M, s \models R \vee AX_s q^j(false)$. If $M, s \models R$ we are done, since R is propositional. If $M, s \models AX_s q^j(false)$ then all successors t_i of s in M, (there is at least one) are such that $M, t_i \models q^j(false)$. Hence, $T, t_i \|\!- q^j(false)$ as M is contained in T, and $T, t_i \models (q')^j(false)$ by induction hypothesis. Also, there exists an AND-node c_s from s to the t_i's in T. So $T, s \models EX_s AX_s (q')^j(false)$ and $T, s \models (q')^{j+1}(false)$ as desired.

($\Leftarrow$) Suppose $T, s \models (q')^{j+1}(false)$. Then $T, s \models R \vee EX_s AX_s (q')^j(false)$. If $T, s \models R$, then because R is propositional, $T, s \|\!- R$ and we are done. If $T, s \models EX_s AX_s (q')^j(false)$ then there is an AND-node successor c_s from s to OR-nodes t_0, t_1 such that each $T, t_i \models (q')^j(false)$. By induction hypothesis, for each t_i, $T, t_i \|\!- q^j(false)$, and $\exists$ a structure M_i contained in T such that $M_i, t_i \models q^j(false)$.

We let M_0', M_1' be copies [12] of M_0, M_1 respectively, and graft them onto s by letting t_0', t_1' (the copies of t_0, t_1 respectively) be the successors of s. Then the resulting M' is a structure generated by T such that $M', s \models q^{j+1}(false)$. Now we can get a structure $\hat{M}$ contained in T such that $\hat{M}, s \models q^{j+1}(false)$, as follows.

Suppose nodes u, u' of M' are copies of the same OR-node of T, i.e., map to the same OR-node under the generation function. There is a smallest $k \leq j$ and a smallest $k' \leq j$ such that $M', u \models q^k(false)$ and $M', u' \models q^{k'}(false)$, respectively. If $k \leq k'$ then let u replace u' by redirecting all arcs going from predecessors of u' into u' so that they go from predecessors of u' into u instead. Thus u' is no longer accessible and may be deleted. Similarly, if $k > k'$ then let u' replace u. Call the resulting structure, which has one of u, u' "chopped out", M''. Note that for every node v common to M' and M'', if $M', v \models q^l(false)$ then $M'', v \models q^l(false)$. Accordingly, we have that the "q-rank" of nodes does not increase in going from M' to M''. Moreover, M'' is still a structure generated by T, with one fewer pair of duplicates than M', such that $M'', s \models q^j(false)$. This process can be repeated until all duplicates are eliminated. Call the resulting final structure $\hat{M}$. Then $\hat{M}, s \models q^{j+1}(false)$ and is (a copy of) a structure contained in T. $\square$

Proof of Lemma 4.7. The argument depends on the following
Merging Property. If for all $s \in \mathsf{Z}^k$ there exists M contained in Z^k such that $M, s \models g_\gamma \wedge AX_s Z^k \vee R$, then there exists a single M_0 contained in Z^k such that for all $s \in \mathsf{Z}^k$, $M_0, s \models g_\gamma \wedge AX_s Z^k \vee R$.

Proof of property: Let $s_1, \ldots, s_n$ be an enumeration without repetitions of $\mathsf{Z}^k \setminus R$. We will show that we can repeatedly apply the Generalized Linear Size Model Theorem to get M contained in Z^k such that $M, s_1, \ldots, s_n \models g_\gamma \wedge AX_s Z^k$.

For s_1 there exists M_1 contained in Z^k such that $M_1, s_1 \models g_\gamma \wedge AX_s Z^k$. A similar M_2 for s_2 exists. Let M_1' be a copy of M_1 that is disjoint from M_1 except that the original node s_1 is retained. Similarly, let M_2' be a copy of M_2 that is also disjoint from M_1'. Then, defining the union of two disjoint structures in the obvious way, the structure $M_{12}' = M_1' \cup M_2'$ is generated by Z^k, and we have $M_{12}', s_1, s_2 \models g_\gamma \wedge AX_s Z^k$.

By the Generalized Linear Size Model Theorem, we can collapse out duplicates yielding M_{12}'' contained in Z^k such that $M_{12}'', s_1'', s_2'' \models g_\gamma$, where M_{12}'' contains no duplicates and s_1'', s_2'' are (possibly copies of) s_1, s_2, respectively.

Note that for any node t in M_{12}'', if t had successors in M_{12}', it also has successors in M_{12}'', by the nature of the method of chopping out duplicates. Hence, $M_{12}'', s_1'', s_2'' \models g_\gamma \wedge AX_s Z^k$. Of course, M_{12}'' is isomorphic to a structure M_{12} contained in Z^k with s_1'', s_2'' corresponding to s_1, s_2, respectively, and $M_{12}, s_1, s_2 \models g_\gamma \wedge AX_s Z^k$.

[12] We say M' is a copy of M if it is an isomorphic structure, i.e., labelled graph, with a fresh set of nodes disjoint from those of M.

We now continue with s_3 and use M_{12}, M_3 to get M_{123} contained in Z^k such that $M_{123}, s_1, s_2, s_3 \models g_\gamma \wedge AX_s Z^k$. Continue this process until $\mathsf{Z}^k \setminus R$ is exhausted, yielding $M_{1\ldots k}$ contained in Z such that $M_{1\ldots k}, s_1, \ldots, s_k \models g_\gamma \wedge AX_s Z^k$. Now let $M_0 = M_{1\ldots k} \cup N$ where N is the structure formed by just those single nodes t in Z^k satisfying R such that a copy of t does not appear in $M_{1\ldots k}$. Then M_0 is contained in Z^k and for all $s \in \mathsf{Z}^k$ we have $M_0, s \models g_\gamma \wedge AX_s Z^k \vee R$.

This completes the proof of the merging property.

To establish soundness of the the calculation, i.e. $\mathsf{Z}^k \subseteq A(g_\gamma U_w R)^{T,con}$, we first note that $\mathsf{Z}^k = (g \wedge AX_s \mathsf{Z}^k \vee R)^{\mathsf{Z}^k}$ implies, by definition of the f^{Z^k} notation, for each $s \in \mathsf{Z}^k$ that, $\mathsf{Z}^k, s \|\!-_{con} g \wedge AX_s \mathsf{Z}^k \vee R$. By the Merging Property, this in turn implies that there exists a single M_0 contained in Z^k such that for every $s \in \mathsf{Z}^k \setminus R$, $M_0, s \models g$. It follows that for each $s_0 \in M_0$, $M_0, s_0 \models A(g_\gamma U_w R)$. To see this, let $x = s_0, s_1, s_2, \ldots$ be a fullpath in M_0. For each s_i, $M_0, s_i \models g$ or $M_0, s_i \models R$. So $M_0, x \models (g_\gamma U_w R)$ as desired.

By virtue of M_0, for every $s_0 \in \mathsf{Z}^k \setminus R$, we now have that $\mathsf{Z}^k, s_0 \|\!-_{con} A(g_\gamma U_w R)$. Since M_0 is contained in Z^k and $\mathsf{Z}^k \subseteq T$, we also have that M_0 is contained in T and for each $s_0 \in \mathsf{Z}^k \setminus R$, we get $T, s_0 \|\!-_{con} A(g_\gamma U_w R)$. Since for all $s_0 \in R^{T,con}$, we have $T, s_0 \|\!-_{con} A(g_\gamma U_w R)$, we get for all $s_0 \in \mathsf{Z}^k$ that, $T, s_0 \|\!-_{con} A(g_\gamma U_w R)$. Thus, $\mathsf{Z}^k \subseteq A(g_\gamma U_w R)^{T,con}$.

In order to show completeness, i.e., $A(g_\gamma U_w R)^{T,con} \subseteq \mathsf{Z}^k$, we argue that for any i and for any state s_0,
$(*)$ $\quad \mathsf{Z}^i, s_0 \|\!-_{con} A(g_\gamma U_w R)$ implies $\mathsf{Z}^{i+1}, s_0 \|\!-_{con} A(g_\gamma U_w R)$.
Assume that $\mathsf{Z}^i, s_0 \|\!-_{con} A(g_\gamma U_w R)$. Thus for some M contained in Z^i we have $M, s_0 \models A(g_\gamma U_w R)$. Without loss of generality, we may assume that every state t of M is reachable from s_0 and $M, t \models A(g_\gamma U_w R)$.

We claim that M is contained in Z^{i+1}. For all $t \in M$, since $M, t \models A(g_\gamma U_w R)$, it follows that $M, t \models (g \wedge AX_s Z^i) \vee R$, and thus $t \in \mathsf{Z}^{i+1} = \{u \in \mathsf{Z}^i : \mathsf{Z}^i, u \|\!-_{con} (g \wedge AX_s Z^i) \vee R\}$. Hence, $M \subseteq \mathsf{Z}^{i+1}$. Since M is contained in Z^i and $M \subseteq \mathsf{Z}^{i+1} \subseteq \mathsf{Z}^i$, it follows by the definition of containment that M is contained in Z^{i+1}.

Thus, $M, s_0 \models A(g_\gamma U_w R)$ and M is contained in Z^{i+1}, so $\mathsf{Z}^{i+1}, s_0 \|\!-_{con} A(g_\gamma U_w R)$ as desired. (Note that by definition of the $\|\!-_{con}$ notation, $\mathsf{Z}^{i+1}, s_0 \|\!-_{con} A(g_\gamma U_w R)$ implies $s_0 \in \mathsf{Z}^{i+1}$, and the above argument guarantees this.)

Since $\mathsf{Z}^0 = T$, for any $s \in A(g_\gamma U_w R)^{T,con}$, $\mathsf{Z}^0, s \|\!-_{con} A(g_\gamma U_w R)$ and by induction on i using $(*)$, we have that for all i, $\mathsf{Z}^i, s \|\!-_{con} A(g_\gamma U_w R)$. In particular, $\mathsf{Z}^k, s \|\!-_{con} A(g_\gamma U_w R)$. Since s was an arbitrary member of $A(g_\gamma U_w R)^{T,con}$, we conclude $A(g_\gamma U_w R)^{T,con} \subseteq \mathsf{Z}^k$, as desired.

To implement the calculation, we see by using disjunctivity of con that $((g_\gamma \wedge AX_s \mathsf{Z}^i) \vee R)^{\mathsf{Z}^i,con} = (g_\gamma \wedge AX_s \mathsf{Z}^i)^{\mathsf{Z}^i,con} \cup R^{\mathsf{Z}^i,con}$. Now $(g_\gamma \wedge AX_s \mathsf{Z}^i)^{\mathsf{Z}^i,con}$
$= (g_\gamma \wedge AX_s true)^{\mathsf{Z}^i,con}$ within the scope of Z^i, con
$= g_\gamma^{\mathsf{Z}^i,con}$ since g_γ already has $AX_s true$ as a conjunct.
Now apply Lemma 4.8. $\square$

Proof of Lemma 4.8. The $\subseteq$ direction is immediate.

For the $\supseteq$ direction, pick an arbitrary element s of the right hand side. Because s is in the first term, it must be that there exsits an M contained in T such that $M, s \models A(\Phi_{-\gamma} \vee F(R \vee Q_\gamma))$. Since s is also in the second term, it follows that $M, s \models A(\Phi_{-\gamma} \vee F(R \vee Q_\gamma)) \wedge (P_\gamma \vee Y)$. If s has successors in M then $M, s \models A(\Phi_{-\gamma} \vee F(R \vee Q_\gamma)) \wedge (P_\gamma \vee Y) \wedge AX_s true$ and we are done. If s has no successors in M, then since we still have $M, s \models A(\Phi_{-\gamma} \vee F(R \vee Q_\gamma))$ it must be that $M, s \models R \vee Q_\gamma$. Attach to s the successors t, u it must have by virtue of member in the third term and call the resulting structure M'. We have $M', s \models$

$A(\Phi_{-\gamma} \vee F(R \vee Q_\gamma)) \wedge (P_\gamma \vee Y) \wedge AX_s true$, the first conjunct holding because s satisfies $R \vee Q_\gamma$ in M and M', the second conjunct holding because s satisfies $P_\gamma \vee Y$ in M and M', abd the 3rd conjunct holding by virtue of t, u. Again, $s \in g_\gamma^{T,con}$ and we are done. □

9 References

[Bu77] J.R. Büchi, " Using Determinacy to eliminate quantifiers", *Fundamentals of Computation Theory*, LNCS 56, 367-378.

[BB86] B. Banieqbal and H. Barringer, "A Study of an Extended Temporal Language and a Temporal Fixed Point Calculus", Univ. of Manchester Tech. Report UMCS-86-10-2.

[BL69] J.R. Büchi, and L.H. Landweber, " Solving sequential conditions by finite finite-state strategies", Trans. Amer. Math. Soc. 138, 295-311, 1969.

[CE81] E.M. Clarke, E.A. Emerson, "Design and synthesis of synchronization skeletons using branching time temporal logic", Proc. IBM Workshop on Logics of Programs, *Lecture Notes in Computer Science* No. 131, Springer-Verlag.

[CVW86] C. Courcoubetis, M.Y. Vardi, P. Wolper, "Reasoning about Fair Concurrent Programs", *Proc. 16th ACM STOC* , 1986.

[Em85] E.A. Emerson, "Automata,tableaux, and temporal logics", *Proc. Workshop on Logics of Programs*, Brooklyn, June 85.

[EC80] E. A. Emerson and E. M. Clarke, "Characterizing Correctness Properties of Prallel Programs using Fixpoints", Seventh International Conference on Automata, Languages, and Programming, Springer LNCS no. 85, pp. 169–182, July, 1980.

[EC82] E.A. Emerson, E.M. Clarke, "Using Branching Time Logic to Synthesize Synchronization Skeletons", *Science of Computer Programming*, 2, pp. 241-266, 1982.

[EH82] E.A. Emerson, J.Y. Halpern, "Decision procedures and Expressiveness in the Temporal Logic of Branching Time", *Proc. 14th ACM STOC*, San Francisco, 1982, pp. 169-180.

[EJ88] E.A. Emerson, C.S. Jutla, "Complexity of Tree Automata and Modal Logics of Programs", *Proc. 29th IEEE Foundations of Computer Sc., 1988*

[EJ91] E. A. Emerson, C. S. Jutla "Tree Automata, Mu-Calculus, and Determinacy", *Proc. 33rd IEEE Symp. on Found. of Comp Sci., 1991*

[EL86] E.A. Emerson, C.L. Lei, "Efficient Model Checking in Fragments of the Propositional Mu-Calculus ", *IEEE Symposium on Logics in Computer Science*, 1986, pp.267-278.

[ESi84] E.A. Emerson, A.P. Sistla, "Deciding Branching Time Logic", *Information and Control*, vol. 61, no. 3., pp. 175-201, June 1984.

[FL79] M.J. Fisher, R.E. Ladner, "Propositional Dynamic Logic of Regular Programs", *J. Computer and System Sciences*, 18(2), 1979, pp. 194-211.

[Ga76] D.M. Gabbay, " Investigation in Modal and Tense Logic", Reidel, 1976.

[GH82] Y. Gurevich, L. Harrington, "Trees, Automata, and Games", *14th ACM STOC*, 1982.

[Ha82] J.Y. Halpern, "Deterministic Process Logic is Elementary", *23rd IEEE FOCS*, 1982.

[HKP82] D. Harel, D. Kozen, R. Parikh, " Process Logic: Expressiveness, Decidability, Complexity", *JCSS*, 25, pp. 144-170, 1982.

HR72 Hossley, R., and Rackoff, C., The Emptiness Problem for Automata on Infinite Trees, Switching and Automata Theory Symposium, pp. 121-124, IEEE, Oct. 1972.

[Ko83] D. Kozen, "Results on the Propositional Mu-Calculus", *Theoretical Computer Science* , 27, 1983, pp. 333-354.

[La69] L.H. Landweber, "Decision problems for $\omega-$automata", *Math. Syst. Theory*, 3, 376-384.

[LPZ85] Lichtenstein, O., Pneuli, A., Zuck, L., "The Glory of the Past", *International Colloq. on Automata, Languages, and Programming*, Lecture Notes in Computer Science, 1985, Springer- Verlag.

[Ma75] D.A. Martin, "Borel Determinacy", *Annals of Mathematics*, 1975, 102, 363-371.

[Mc66] R. McNaughton, "Testing and Generating Infinite Sequences by a finite Automaton", *Information and Control*, 9, 521-530, 1966.

[MS84] D.E. Muller, P.E. Schupp, "Alternating Automata on Infinite Objects, Determinacy and Rabin's Theorem", *Automata on Infinite Words*, 1984, Lecture Notes in Computer Science, 192.

[MW84] Z. Manna, P. Wolper, "Synthesis of Communicating Processes from Temporal Logic Specifications", *ACM TOPLAS*, Vol. 6, No.1, pp 68-93, 1984.

[Pa83] R. Parikh, "Propositional Game Logic", *25th IEEE FOCS*, 1983, pp. 195-200

[Pn77] A. Pneuli, "The Temporal Logic of Programs", *19th IEEE FOCS*, 1977.

[PR89a] A. Pnueli and R. Rosner, On the Synthesis of a Reactive Module, 16th Annual ACM Symp. on Principles of Programing Languages, pp. 179-190, Jan. 1989.

[PR89b] A. Pnueli and R. Rosner, On the Synthesis of an Asynchronous Reactive Module, Proc. 16th Int'l Colloq. on Automata, Languages, and Programming, Stresa, Italy, July, 1989, pp. 652-671, Springer-Verlag LNCS no. 372.

[Ra69] M.O. Rabin, "Decidability of Second Order Theories and Automata on

[Ra72] M. O. Rabin, Automata on Infinite Objects and Church's Problem, Conf. Board. of Math. Sciences, Regional Series in Math. no. 13, Amer. Math. Soc., Providence, Rhode Island, 1972. Infinite Trees", *Trans. AMS*, 141(1969), pp. 1-35.

[Sa88] S. Safra, "On Complexity of ω-automata", Proc. 29th IEEE FOCS, 1988.

[St81] R.S. Streett, "A Propositional dynamic Logic of Looping and Converse", *MIT LCS* Technical Reeport TR-263.

[SC85] A.P. Sistla, E.M. Clarke, "The Complexity of Propositional Linear Temporal Logic", *JACM*, vol. 32, no. 3, pp 733-749, July 1985

[StE84] R.S.Streett, E.A. Emerson, "An Elementary Decision Procedure for the Mu-calculus", *Proc. 11th Int. Colloq. on Automata, languages and Programming*, 1984, Lecture Notes in Computer Science, Springer-Verlag.

[Thi91] Thistle, J. G., Control of Infinite Behavior of Discrete Event Systems, PhD Dissertation, University of Toronto, 1991.

[TW91] Thistle, J., and Wonham, W., "Control of ω-Automata", Church's Problem, and the Emptiness Problem for Tree ω-Automata", Computer Science Logic '91.

[Tho90] Thomas, W., Automata on Infinite Objects, in Handbook of Theoretical Computer Science, vol. B., J. van Leeuwen, ed., Elsevier/MIT Press, 1990.

[V87] M.Y. Vardi, "Verification of Concurrent Programs: The Automata-Theoretic Framework", *Logics in Computer Science*, 1987.

[V88] M. Y. Vardi, A Temporal Fixpoint Calculus, POPL88

[VS85] M. Vardi, L. Stockmeyer, " Improved Upper and Lower Bounds for Modal logics of Programs", *17th ACM STOC*, 1985, pp.240-251.

[VW83] M. Y. Vardi , P. L. Wolper, Yet Another Process Logic, CMU Workshop on Logics of Programs, Springer LNCS no. 164, June 1983.

[VW84] M.Y. Vardi, P. L. Wolper, "Automata-Theoretic Techniques for Modal Logics of Programs", *Proc. 14th ACM STOC* , 1984.