Deterministic Generators and Games for LTL Fragments *

Rajeev Alur
University of Pennsylvania &
Bell Labs
alur@cis.upenn.edu

Salvatore La Torre †
University of Pennsylvania &
Università degli Studi di Salerno
latorre@seas.upenn.edu

Abstract

Deciding infinite two-player games on finite graphs with the winning condition specified by a linear temporal logic (LTL) formula, is known to be 2EXPTIME-complete. In this paper, we identify LTL fragments of lower complexity. Solving LTL games typically involves a doubly-exponential translation from LTL formulas to *deterministic* ω-automata. First, we show that the *longest distance* (length of the longest simple path) of the generator is also an important parameter, by giving an $O(d \log\ n)$-space procedure to solve a Büchi game on a graph with n vertices and longest distance d. Then, for the LTL fragment with only eventualities and conjunctions, we provide a translation to deterministic generators of exponential size and linear longest distance, show both of these bounds to be optimal, and prove the corresponding games to be PSPACE-complete. Introducing *next* modalities in this fragment, we provide a translation to deterministic generators still of exponential size but also with exponential longest distance, show both of these bounds to be optimal, and prove the corresponding games to be EXPTIME-complete. For the fragment resulting by further adding disjunctions, we provide a translation to deterministic generators of doubly-exponential size and exponential longest distance, show both of these bounds to be optimal, and prove the corresponding games to be EXPSPACE. Finally, we show tightness of the double-exponential bound on the size as well as the longest distance for deterministic generators for LTL even in the absence of *next* and *until* modalities.

*This research was partially supported by NSF Career award CCR97-34115, NSF award CCR99-70925, SRC award 99-TJ-688, and Alfred P. Sloan Faculty Fellowship.

†Partially supported by the M.U.R.S.T. in the framework of project TOSCA.

1 Introduction

Linear temporal logic (LTL) is a popular choice for specifying correctness requirements of reactive systems [14, 13]. An LTL formula is built from state predicates, boolean connectives, and temporal modalities such as *next*, *eventually*, *always*, and *until*, and is interpreted over infinite sequences of states modeling computations of reactive programs. The most studied decision problem concerning LTL is *model checking*: given a finite-state abstraction G of a reactive system and an LTL formula φ, do all infinite computations of G satisfy φ? The first step of the standard solution to model checking involves translating a given LTL formula to a (non-deterministic) Büchi automaton that accepts all of its satisfying models [12, 21]. Such a translation is central to solving the satisfiability problem for LTL also. The translation can be exponential in the worst case, and in fact, both model checking and satisfiability are PSPACE-complete [18].

The standard interpretation of LTL over infinite computations is the natural one for closed systems, where a *closed system* is a system whose behavior is completely determined by the state of the system. However, the compositional modeling and design of reactive systems requires each component to be viewed as an open system, where an *open system* is a system that interacts with its environment and whose behavior depends on the state of the system as well as the behavior of the environment. In the setting of open systems, the key decision problem is to compute the winning strategies in infinite two-player games. In the satisfiability game, we are given an LTL formula φ and a partitioning of atomic propositions into inputs and outputs, and we wish to determine if there is a strategy to produce outputs so that no matter which inputs are supplied, the resulting computation satisfies φ. This problem has been formulated in different contexts such as *synthesis* of reactive modules [15], *realizability* of liveness specifications [4], and *receptiveness* [5]. In the model-checking game, we are given an

0-7695-1281-X/01 $10.00 © 2001 IEEE

LTL specification φ, and a game graph G whose states are partitioned into system states and environment states. We wish to determine if the protagonist has a strategy to ensure that the resulting computation satisfies φ in the infinite game in which the protagonist chooses the successor in all system states and the adversary chooses the successor state in all environment states. This problem appears in contexts such as *module checking* and its variants [9, 10], and the definition of *alternating temporal logic* [2]. Such game-based model checking for restricted formulas such as "always p" has already been implemented in the software MOCHA [3], and shown to be useful in construction of the most-general environments for automating assume-guarantee reasoning [1].

We focus on the game version of model checking: given a game graph G and an LTL formula φ, what is the complexity of deciding whether a given player has a winning strategy starting from a given initial state (game version of satisfaction is a special case, and similar bounds apply). It is known that the complexity of this problem is doubly-exponential in the size of the LTL formula, and the problem is 2EXPTIME-complete [15]. Note that the complexity is much lower for formulas of specific form: generalized Büchi games (formulas of the form $\wedge_i \Box\Diamond p_i$) are solvable in polynomial time, and Streett games (formulas of the form $\wedge_i(\Box\Diamond p_i \rightarrow \Box\Diamond q_i)$) are coNP-complete (the dual, Rabin games are NP-complete) [16, 7]. It is worth mentioning that, in the standard model checking, while full LTL is PSPACE-complete, the fragment which allows only *eventually* and *always* operators (but no *next* or *until*) has a small model property and is NP-complete [18] (see also [6] for complexity results on simpler fragments of LTL). This motivated us to consider the problem addressed in this paper: are there fragments of LTL for which games have complexity lower than 2EXPTIME?

The standard approach to solving games for LTL is by reduction to a game on the product of the game graph and a *deterministic* automaton that accepts all the models of the given formula. The winning condition in this reduced game corresponds to the type of the acceptance condition (e.g. Büchi or Rabin) for the deterministic generator [1]. To obtain a deterministic generator, the standard approach is to first build a nondeterministic generator and then determinize it. Each of these steps costs an exponential, and it is known that there are LTL formulas whose deterministic generators have to be doubly-exponential [11].

In this paper, we give a comprehensive study of deterministic generators and game complexities of various LTL fragments. We use the notation $\text{LTL}(op_1, \ldots, op_k)$ to denote the fragment of LTL given by top-level boolean combination of formulas which use only the boolean connectives and the temporal operators in the list $op_1, \ldots, op_k$. Our first result is a construction of a singly-exponential deterministic Büchi automaton for the fragment $\text{LTL}(\Diamond, \wedge)$. This construction is different from the standard tableau-based construction, and builds the automaton for a formula in a modular way from the automata for its subformulas. This immediately gives a single exponential bound for $\text{LTL}(\Diamond, \wedge)$ games by using the standard algorithm for Büchi games. However, the deterministic generators have the property that the longest simple path is at most linear in the size of the formula. We show that this property can be exploited to reduce space requirement. In fact, we show a general result: in a game graph with n vertices and *longest distance* d (that is, length of longest simple path), a Büchi game can be solved in space $O(d \log\ n)$ (the conventional algorithm uses $O(n)$ space). This leads us to the result that $\text{LTL}(\Diamond, \wedge)$ games can be solved in PSPACE, and we show a matching lower bound. Note that the fragment $\text{LTL}(\Diamond, \wedge)$ contains boolean combinations of invariant ("always p") and termination ("eventually q") properties, and thus includes many of the commonly used specifications.

Combining *next* modalities with the eventualities raises the complexity. For any formula in $\text{LTL}(\Diamond, \bigcirc, \wedge)$, we show how to construct a deterministic Büchi generator with both states and longest distance of exponential size. The construction is optimal since there exists an $\text{LTL}(\Diamond, \bigcirc, \wedge)$ formula for which all deterministic generators must have exponential longest distance. This construction leads to an EXPTIME algorithm for solving games in $\text{LTL}(\Diamond, \bigcirc, \wedge)$, and we show a matching lower bound.

Adding disjunctions to $\text{LTL}(\Diamond, \bigcirc, \wedge)$ raises complexity. Given an $\text{LTL}(\Diamond, \bigcirc, \wedge, \vee)$ formula, we show how to construct a corresponding deterministic Büchi automaton with doubly-exponential states and singly-exponential longest distance. The construction is optimal since we show that there is an $\text{LTL}(\Diamond, \wedge, \vee)$ formula whose deterministic generator must be doubly-exponential with singly-exponential longest distance. Our construction leads to an EXPSPACE algorithm for

[1] In the automata-theoretic formulation of the problem [20], the game graph can be viewed as a tree automaton that generates all the strategies of one of the players. From the formula φ, we can construct a tree automaton that accepts precisely those trees all of whose paths satisfy φ, take product with the game tree automaton, and test for emptiness. This approach has the same computational essence, and requires determinization.

solving games in LTL($\Diamond$,$\bigcirc$,$\wedge$,$\vee$). A matching lower bound remains an open problem.

The nesting of eventually and always modalities causes a further increase in the complexity. We prove that there exists a formula in LTL($\Box, \Diamond, \wedge, \vee$) whose deterministic generator must be doubly-exponential with doubly-exponential longest distance, that matches the upper bound for the full LTL. This is in sharp contrast to the fact that the longest distance of nondeterministic generators for LTL($\Box, \Diamond, \wedge, \vee$) formulas is only linear, and becomes exponential only by addition of next or until modalities.

2 Definitions

2.1 Linear Temporal Logic

We first recall the syntax and the semantics of linear temporal logic. We will define temporal logics by assuming that the atomic formulas are state predicates, that is, boolean combinations of atomic propositions. Given a set of atomic propositions, a *linear temporal logic* (LTL) formula is composed of state predicates, the boolean connectives *conjunction* ($\wedge$) and *disjunction* ($\vee$), the temporal operators *Next* ($\bigcirc$), *Eventually* ($\Diamond$), *Always* ($\Box$), and *Until* ($\mathcal{U}$). Formulas are built up in the usual way from these operators and connectives, according to the following grammar

$$\varphi := p \mid \varphi \wedge \varphi \mid \varphi \vee \varphi \mid \bigcirc\varphi \mid \Diamond\varphi \mid \Box\varphi \mid \varphi\,\mathcal{U}\,\varphi.$$

An *ω-word* over a given alphabet Σ is a mapping from $\mathbb{N}$ into Σ, that is, an infinite sequence of symbols over Σ. LTL formulas are interpreted on an ω-word $w = w_0 w_1 w_2 \ldots$ over the alphabet $\Sigma = 2^P$ and the satisfaction relation $w \models \varphi$ is defined in the standard way. In the following, we will use the notation LTL($op_1, \ldots, op_k$) to denote the fragment of LTL which contains boolean combination of basic formulas which use only the boolean connectives and the temporal operators in the list $op_1, \ldots, op_k$.

2.2 Finite automata on ω-words

Automata on ω-words have been extensively studied in relation to temporal logic [8]. In this section, we will recall the definition of Büchi automata and the results relating them to LTL as *generators* of models.

A *nondeterministic transition graph* is a 4-tuple (Σ, S, S_0, Δ), where Σ is an alphabet, S is a finite set of states, $S_0 \subseteq S$ is the set of initial states, and Δ is a subset of $S \times \Sigma \times S$. A transition graph is *deterministic* if $|S_0| = 1$ and Δ defines a total function δ from $S \times \Sigma$ into S. In the following, when we consider deterministic transition graphs, we will define directly this function δ instead of the transition relation Δ. The behavior of a transition graph on a word is captured by the concept of a *run*. Let $A = (\Sigma, S, S_0, \Delta)$ be a transition graph and w be an ω-word, a run of A on w is a mapping $r : \mathbb{N} \longrightarrow S$ such that $r(0) \in S_0$ and for all $i \in \mathbb{N}$, $(r(i), w(i), r(i+1)) \in \Delta$. Given a run r on a word w, we denote with $Inf(r)$ the set of states appearing infinitely often in r. A clear property of deterministic transition graphs is that they have exactly one run for each word.

Given a transition graph we define an automaton by specifying the acceptance conditions. A *nondeterministic (resp. deterministic) Büchi automaton* is a 5-tuple $A = (\Sigma, S, S_0, \Delta, F)$, where (Σ, S, S_0, Δ) is a nondeterministic (resp. deterministic) transition graph and $F \subseteq S$ is the set of the *accepting* states. An ω-word w is accepted by a Büchi automaton A iff there exists a run r of A on w such that $Inf(r) \cap F \neq \emptyset$. The language accepted by A, denoted by $L(A)$, is defined to be the set $\{w \mid w \text{ is accepted by } A\}$.

For our results, besides the size, another characterizing measure of an automaton A is the length of the longest simple directed path connecting two states in the transition graph. We will refer to this measure as the *longest distance* of A.

For every LTL formula φ, it is possible to construct an automaton on ω-words accepting all models of it. We will denote such an automaton as A_φ and we will refer to it as a *generator* of models for φ. A deterministic generator for an LTL formula of size $O(exp(exp(|\varphi|)))$ can be obtained in the following way: from the formula φ, by the tableau construction, it is possible to construct a nondeterministic Büchi generator of size $O(exp(|\varphi|))$ [12, 21]; this automaton can then be determinized so that we obtain a deterministic Rabin automaton of size $O(exp(exp(|\varphi|)))$ [17]. Notice that in general, for a given formula φ, a deterministic Büchi generator may not exist but, when this exists, it has been proved that the translation from LTL formulas to deterministic Büchi automata is doubly-exponential [11], and thus, the above construction is asymptotically optimal.

2.3 Game graphs

In this section we will introduce the notation concerning two-player games. A two-player game is modeled by a *game graph* and a *winning condition*. A game graph is a tuple $G = (V, V_0, V_1, \Sigma, \gamma)$ where V

is a finite or countable set of vertices, V_0 and V_1 define a partition of V, Σ is a finite set of actions and $\gamma : V \times \Sigma \to V$ is a partial function. For $i = 0, 1$, the vertices in V_i are those from which only $Player_i$ can move and the allowed moves are given by the function γ. A winning condition is a predicate over ω-words of vertices, and depending on its type, we can have different kinds of games. In this paper we will consider only Büchi and LTL games. In a Büchi game, the winning condition is given by a set of vertices $F \subseteq V$ with the requirement that at least a state in F must repeat infinitely often. In an LTL game, the winning condition is instead an LTL formula.

A *play* of a game G is constructed as a sequence of vertices corresponding to the actions taken by the two players. Formally, a play starting at x_0 is a sequence $x_0 x_1 \dots x_h$ in V^* with the property that there exists a sequence of actions $a_1, \dots, a_h \in \Sigma$ such that $\gamma(x_{j-1}, a_j) = x_j$, for $j = 1, \dots, h$. Starting from a vertex u, a game G can be seen as the ω-tree $T_{(G,u)}$, called a *game tree*, which is obtained by unwinding G from u. Each node of this tree corresponds to a play starting at u: the root corresponds to u and, if a node v corresponds to a play $x_1 \dots x_h$, then each of its children corresponds to a possible continuation of the play $x_0 \dots x_h$, i.e. to a play $x_0 \dots x_h x_{h+1}$ such that $\gamma(x_h, a) = x_{h+1}$ for an action $a \in \Sigma$. A *strategy* for $Player_i$ gives an allowed move to continue each play ending at a vertex in V_i. More formally, a strategy for $Player_i$ is a total function $f : V^* V_i \to V$ mapping a node in the function domain into one of its successors in the game tree. A strategy then corresponds to a tree obtained from the game tree $T_{(G,u)}$ by pruning all the subtrees containing plays that are not constructed according to f. When a strategy depends only on the last vertex of a play, it is called a *memoriless strategy*.

Given a game G and a winning condition W, a strategy f is said to be a *winning strategy* if the requirement expressed by W holds on all the paths of the tree corresponding to f. In a two-player game, given a game G and a winning condition W, we consider the decision problem: "Is there a strategy for $Player_i$ satisfying the winning condition W?" We remark that while Büchi games admit memoriless winning strategies and can be solved in quadratic time, LTL games in general do not have a memoryless winning strategy and are decidable in time polynomial in $|G|$ and doubly-exponential in $|\varphi|$ [15].

3 Deterministic generators

We begin this section by introducing a proper subclass of deterministic Büchi automata whose transition function defines a partial order over the states. To emphasize this property, we call an automaton in this class a *partially-ordered deterministic Büchi automaton* (PODB). Then, we will show that, for formulas in some fragments of LTL, it is possible to construct a deterministic generator which is a PODB.

A PODB is a deterministic Büchi automaton whose transition graph is a directed acyclic graph except for the self-loops. Obviously, the longest distance of a PODB is the longest distance between the initial state and a sink state, where an initial and a sink state are respectively a minimal and a maximal state with respect to the partial order induced by the transition function of the PODB. PODBs are closed under boolean operations.

Proposition 3.1 *For $i = 1, 2$, let A_i be PODBs of size n_i and longest distance d_i. There exists a PODB $A_1 \cap A_2$ (resp. $A_1 \cup A_2$) accepting the language $L(A_1) \cap L(A_2)$ (respectively, $L(A_1) \cup L(A_2)$), and such that its size is $O(n_1 n_2)$ and its longest distance is not greater than $d_1 + d_2$. Moreover, for $i = 1, 2$, there exists a PODB $\overline{A_i}$ of size n_i and longest distance d_i accepting $\Sigma^\omega \setminus L(A_i)$.*

Note that to prove the above proposition, the construction for intersection does not require the introduction of a counter as in the case of general deterministic Büchi automata. Moreover, the above results on intersection and union are naturally extended to a tuple of automata $A_1, \dots, A_k$ and we will denote the corresponding automata with $A_1 \cap \dots \cap A_k$ and $A_1 \cup \dots \cup A_k$, respectively.

The following automaton construction will be used in the next sections to build the generator for $\Diamond(p \wedge \varphi)$ given the generator for φ. Let $A = (\Sigma, S, s_0, \delta, F)$ be a Büchi automaton and p be a predicate over Σ. Given a $s'_0 \notin S$, we define the (deterministic) Büchi automaton $A^{\Diamond(p,A)}$ as $(\Sigma, S \cup \{s'_0\}, s'_0, \delta', F)$ where:

- $\delta'(s, a) = \delta(s, a)$ for $s \in S$,
- $\delta'(s'_0, a) = \delta(s_0, a)$ for a satisfying p, and
- $\delta'(s'_0, a) = s'_0$, otherwise.

The construction is illustrated in Figure 1.

Proposition 3.2 *Let $A = (\Sigma, S, s_0, \delta, F)$ be a (deterministic) Büchi automaton of size n and longest distance d such that $\Sigma\, L(A) \subseteq L(A)$, and p be a predicate over Σ. The (deterministic) automaton $A^{\Diamond(p,A)}$ has size $O(n)$, longest distance $d + 1$ and accepts the language $\Sigma^* [p] L(A)$, where $[p] = \{a \in \Sigma \mid a \text{ satisfies } p\}$.*

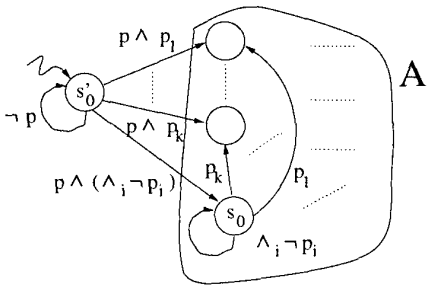

Figure 1: Graphical representation of the automaton $A^{\Diamond(p,A)}$.

Moreover, if A is a PODB then $A^{\Diamond(p,A)}$ is a PODB also.

3.1 Generators for LTL($\Diamond, \wedge$)

The fragment LTL($\Diamond, \wedge$) contains boolean combinations of formulas built from state predicates using eventualities and conjunctions. Thus, negations and disjunctions are allowed only at the top-level and at the atomic level. By definition, LTL($\Diamond, \wedge$) is equivalent to LTL($\Box, \vee$). A sample formula of this fragment is $\Box p \vee \Diamond(q \wedge \Diamond r)$. This fragment includes combinations of typical invariants and termination properties.

Let us consider the formula $\varphi = \Diamond p_1 \wedge \ldots \wedge \Diamond p_n$, where $p_i \in P$ for $i = 1, \ldots, n$. Obviously, φ is in LTL($\Diamond, \wedge$). This formula asserts that each one of $p_1, \ldots, p_n$ has to be true sometimes. Then, a deterministic generator A_φ for φ has to keep track only of the set of atomic propositions which have been already fulfilled. The size of A_φ is $O(2^n)$ and its longest distance is the cardinality of the maximal totally ordered set of states with respect to the subset relation, that is, n. We proceed to show that all the LTL($\Diamond, \wedge$) formulas have a deterministic generator which is a PODB of exponential size and linear longest distance, but first, we introduce a characterization of the formulas in the considered fragment. A formula φ in LTL($\Diamond, \wedge$) is a boolean combination of formulas defined inductively by the following rules:

- φ is a state predicate over P or,
- for $k \geq 0$, φ is $p \wedge \Diamond\varphi_1 \wedge \ldots \wedge \Diamond\varphi_k$ where p is a state predicate over P and $\varphi_1, \ldots, \varphi_k$ are formulas in LTL($\Diamond, \wedge$) that do not contain negations and disjunctions at the top-level.

Theorem 3.3 *There exists a deterministic Büchi automaton A accepting all the models of a formula φ in* LTL($\Diamond, \wedge$) *such that A is a PODB of $O(exp(|\varphi|))$ size and $O(|\varphi|)$ longest distance.*

Proof. We inductively define a deterministic Büchi automaton A accepting all the models of a given formula $\Diamond\varphi$ in LTL($\Diamond, \wedge$) such that A is a PODB of exponential size and linear longest distance in $|\varphi|$, and then by Proposition 3.1 this result is extended to a general formula in LTL($\Diamond, \wedge$). For a state predicate p, we define A_p and $A_{\Diamond p}$ as the minimal deterministic generator for p and $\Diamond p$, respectively. Clearly, A_p and $A_{\Diamond p}$ are PODBs and $A_{\Diamond p}$ is such that $\Sigma^* L(A_{\Diamond p}) \subseteq L(A_{\Diamond p})$. Now, let ψ be the formula $\Diamond(p \wedge \Diamond\psi_1 \wedge \ldots \wedge \Diamond\psi_k)$ and, for a formula $\gamma \in \{\psi_1, \ldots, \psi_k\}$, $A_{\Diamond\gamma}$ be a PODB accepting all the models of $\Diamond\gamma$. By inductive hypothesis we have that size of $A_{\Diamond\gamma}$ is $O(exp(|\Diamond\gamma|))$ and longest distance of $A_{\Diamond\gamma}$ is $O(|\Diamond\gamma|)$. Obviously, $\Sigma^* L(A_{\Diamond\gamma}) \subseteq L(A_{\Diamond\gamma})$ also holds. Then, by Proposition 3.1, $A' = A_{\Diamond\psi_1} \cap \ldots \cap A_{\Diamond\psi_k}$ is a PODB of $O(exp(|\Diamond\psi_1| + \ldots + |\Diamond\psi_k|))$ size, $O(|\Diamond\psi_1| + \ldots + |\Diamond\psi_k|)$ longest distance, and such that $\Sigma^* L(A') \subseteq L(A')$. Thus, from Proposition 3.2, we have that $A_\psi = A^{\Diamond(p,A')}$ is the generator for ψ. ∎

The previous result is optimal in the sense that we may not have a smaller generator for some formula in LTL($\Diamond, \wedge$), as shown in the following theorem.

Theorem 3.4 *There exists a formula φ in* LTL($\Diamond, \wedge$) *such that all generators of φ have $\Omega(exp(|\varphi|))$ size and $\Omega(|\varphi|)$ longest distance.*

Proof. Consider the formula $\varphi = \Diamond p_1 \wedge \ldots \wedge \Diamond p_n$, where $p_i \in P$ for $i = 1, \ldots, n$ and $n \geq 2$. Clearly, $|\varphi| = O(n)$. The first assertion can be easily proved by contradiction showing that the initial state of a φ generator must have at least $2^n - 1$ successors. The second assertion can be proved by contradiction by showing that if a generator A_φ for φ has longest distance less than n, from the φ model $w = \{p_1\}\{p_2\}\ldots\{p_n\}^\omega$, we can derive another word which is not a model of φ but is accepted by A_φ. ∎

3.2 Generators for LTL($\Diamond, \bigcirc, \wedge$)

In this section we use the notation $\bigcirc^n$ as a shorthand for n nested next modalities. We therefore consider size of $\bigcirc^n \varphi$ to be $|\varphi| + n$. Let us consider the formula $\varphi = \Diamond(p \wedge \bigcirc^n q)$, where $p, q \in P$. This formula asserts that p has to be fulfilled at a position i and q at a position $i + n$ for some $i \in \mathbb{N}$. A deterministic generator for φ has to keep track of the truth values of p in the previous n positions. This can be done by running n copies of the deterministic generators for $(p \wedge \bigcirc^n q)$. Such a generator requires exponentially many states and has exponential longest distance. We prove that this upper bound holds for all LTL($\Diamond, \bigcirc, \wedge$) formulas:

Theorem 3.5 *There exists a deterministic Büchi automaton A accepting all the models of a formula φ in* LTL($\Diamond$,$\bigcirc$,$\wedge$) *such that A has both size and longest distance at most exponential in $|\varphi|$.*

Proof. The construction is done inductively on the structure of formulas in LTL($\Diamond$,$\bigcirc$,$\wedge$). We observe that given a formula ψ, the next operators in ψ can be pushed inside so that we can obtain an equivalent formula ψ' having only state predicates in the scope of a finite sequence of next operators, and such that $\psi' = O(|\psi|^2)$. As a consequence most of the cases are handled as for the construction of a deterministic generator for LTL($\Diamond$,$\wedge$) formulas. The interesting case is to construct a deterministic generator for $\varphi = \Diamond(p \wedge \bigcirc^k q \wedge \varphi')$ given a deterministic generator $A_{\varphi'}$ for φ' of both size and longest distance exponential in $|\varphi|$, and such that $\Sigma^* L(A_{\varphi'}) \subseteq L(A_{\varphi'})$. A deterministic generator A_φ for φ can be obtained by running in parallel k copies of $A_{\varphi'}$ and checking for the fulfillment of $(p \wedge \bigcirc^k q)$. At every position i of the input word a copy of $A_{\varphi'}$ is started and if $i > k$ and $(p \wedge \bigcirc^k q)$ is not true at position $(i-k)$ then the copy started at position $(i-k)$ is dismissed. As soon as $(p \wedge \bigcirc^k q)$ becomes true, A_φ dismisses all copies of $A_{\varphi'}$ but the one started at the position where $(p \wedge \bigcirc^k q)$ is true, and continues as $A_{\varphi'}$. The size of A_φ is thus $O(exp(k\,|P|)|A_{\varphi'}|)$ and hence exponential in $|\varphi|$. Its longest distance is $O(exp(k) + d')$, where d' is the longest distance of $A_{\varphi'}$, and thus is exponential in $|\varphi|$. ∎

The previous result is optimal in the sense that we may not have a smaller generator for some formula in LTL($\Diamond$,$\bigcirc$,$\wedge$), as shown in the following theorem.

Theorem 3.6 *There exists a formula φ in* LTL($\Diamond$,$\bigcirc$,$\wedge$) *such that all generators of φ have $\Omega(exp(|\varphi|))$ size and $\Omega(exp(|\varphi|))$ longest distance.*

Proof. Consider the formula $\varphi = \Box(p \rightarrow \bigcirc^n q)$, where $p, q \in P$ and $n \geq 2$. Clearly, $|\varphi| = O(n)$. Since LTL($\Diamond$,$\wedge$) is a fragment of LTL($\Diamond$,$\bigcirc$,$\wedge$), we only need to prove that all generators for φ have a simple path of length at least 2^n. Assume that $A_\varphi = (2^P, S, s_0, \Delta, F)$ is a generator for φ. Consider words $w = a_1 \ldots a_n$ and $w' = a'_1 \ldots a'_n$ such that $w, w' \in (2^P)^*$, and $p \notin a_i$ and $p \in a'_i$ for some i. Let $y \in (2^P)^\omega$ be such that $y = b_1 \ldots b_h \ldots$, $q \notin b_i$, and xwy is a model of φ for some $x \in (2^P)^*$. We have that $xw'y$ is not a model of φ. Thus a generator A_φ cannot enter the same state after reading xw and xw', since it must accept xwy and reject $xw'y$. Clearly we can prove this for any pair of words w, w' of length n that differs with respect to the truth of p at least in a position. Since we can determine 2^n words $w_1, \ldots, w_{2^n}$ which are pairwise different with respect to truth values of p, there are 2^n pairwise disjoint sets of states each of them contains the states which are reached on all runs of A_φ by reading a prefix of a model for φ ending in w_i. To conclude this proof we just need to prove that there exists a word that forces A_φ to visit a state from each of these sets without reentering any of them before reading at least one state from each set. But this is equivalent to prove that there is an exponentially long word w in $\{0,1\}^*$ such that any two subwords of w of length n differ at least in a position, and thus we are done. ∎

3.3 Generators for LTL($\Diamond$,$\wedge$,$\vee$) and LTL($\Diamond$,$\bigcirc$,$\wedge$,$\vee$)

The fragment LTL($\Diamond$,$\bigcirc$,$\wedge$,$\vee$) contains boolean combinations of formulas built from state predicates using eventualities, next, disjunctions, and conjunctions. This fragment includes combinations of safety and guarantee properties, and belongs to the class of syntactic obligation properties [13].

Let us consider the formula $\varphi = \Diamond \bigwedge_{i=1}^{n} (p_i \vee \Diamond q_i)$, where $p_i, q_i \in P$, for $i = 1, \ldots, n$ and $n \geq 2$. Obviously φ is an LTL($\Diamond$,$\wedge$,$\vee$) formula. This formula asserts that at a same position in the model all the clauses $(p_i \vee \Diamond q_i)$ have to be satisfied. Since the fulfillment of a clause at a position implies either $p_i \vee q_i$ at that position or q_i at a later position, a nondeterministic generator for φ is the one that nondeterministically guesses the first position at which all the clauses are satisfied and, then, check for their fulfillment. Such a generator has an exponential size and a linear longest distance. We can determinize this strategy to obtain a deterministic generator for φ with $O(2^{2^n})$ states and $O(2^n)$ longest distance. It is possible to prove that this result indeed holds for all LTL($\Diamond$,$\bigcirc$,$\wedge$,$\vee$) formulas, as stated by the following theorem.

Theorem 3.7 *There exists a deterministic Büchi automaton A accepting all the models of a formula φ in* LTL($\Diamond$,$\bigcirc$,$\wedge$,$\vee$) *such that A has size doubly-exponential in $|\varphi|$ and longest distance exponential in $|\varphi|$.*

Proof. To construct a deterministic generator for LTL($\Diamond$,$\bigcirc$,$\wedge$,$\vee$) formulas we first transform them into a "layered" conjunctive normal form where we have either LTL($\Diamond$,$\bigcirc$,$\wedge$) formulas or formulas of type $\psi = \Diamond \bigvee_i (p_i \wedge \bigcirc^k q_i \wedge \psi_i)$. This translation may cause an exponential blow-up in the size of the formula. The results obtained for LTL($\Diamond$,$\bigcirc$,$\wedge$) then give the upper bound on the size of the deterministic genera-

tor for LTL($\Diamond, \bigcirc, \wedge, \vee$) formulas. An accurate analysis of the longest distance in the construction given for LTL($\Diamond, \bigcirc, \wedge$) gives an $O(exp(k\,|P|) + |\psi|)$ upper bound, where k is the largest number of nested next modalities in the starting formula. Since the transformation into the layered CNF does not increase this parameter, given an LTL($\Diamond, \bigcirc, \wedge, \vee$) formula we get that the longest distance of the deterministic generator obtained by the given construction is exponential in $|\varphi|$. ∎

The following theorem shows that the above result is optimal also in the case of LTL($\Diamond, \wedge, \vee$) formulas.

Theorem 3.8 *There exists a formula φ in* LTL($\Diamond, \wedge, \vee$) *such that all the deterministic generators of φ have $\Omega(exp(exp(|\varphi|)))$ size and $\Omega(exp(|\varphi|))$ longest distance.*

Proof. Consider the formula $\varphi = \Diamond \bigwedge_{i=1}^{n} (p_i \vee \Diamond q_i)$, where $p_i, q_i \in P$ for $i = 1, \ldots, n$ and $n \geq 2$. Obviously, $|\varphi| = O(n)$. Denote with P_p the set $\{p_1, \ldots, p_n\}$ and P_q the set $\{q_1, \ldots, q_n\}$. We prove that a minimal deterministic generator for φ has $2^{2^{\Omega(n)}}$ states. With a similar argument it is also possible to show that all the deterministic generators for φ have a simple path of length $2^{\Omega(n)}$. Assume that $A_\varphi = (2^P, S, s_0, \delta, F)$ is a deterministic generator for φ. Given a subset b of P_p, define $q(b)$ as the set $\{q_i \,|\, p_i \notin b\}$. Define Σ_k as the set of P_p subsets of cardinality k, that is, $\Sigma_k = \{a \subseteq P_p \,|\, |a| = k\}$. The cardinality of Σ_k is $\binom{n}{k}$. If we choose $k = \lceil \frac{n}{2} \rceil$, then $|\Sigma_k| = 2^{\Omega(n)}$. Observe that for $w, w' \in \Sigma_k^*$ such that $w = \sigma_0 \sigma_1 \ldots \sigma_m$, $w' = \sigma'_0 \sigma'_1 \ldots \sigma'_{m'}$ and $\cup_{i=1}^{m} \{\sigma_i\} \neq \cup_{i=1}^{m'} \{\sigma'_i\}$, it must hold that $\delta(s_0, w) \neq \delta(s_0, w')$. In fact, we can suppose without loss of generality that there is a $\sigma \in \cup_{i=1}^{m} \{\sigma_i\}$ such that $\sigma \notin \cup_{i=1}^{m'} \{\sigma'_i\}$. Thus, for any $w'' \in (2^P)^\omega$, the word $wq(\sigma)w''$ is a model of φ and $w'q(\sigma)\emptyset \ldots \emptyset \ldots$ is not. Since A_φ accepts all and only the models of φ, and there is an accepting run for any word $wq(\sigma)w''$, if $\delta(s_0, w) = \delta(s_0, w')$ then A_φ accepts also $w'q(\sigma)\emptyset \ldots \emptyset \ldots$, and this contradicts the hypothesis A_φ being a generator of models for φ. Since the number of subsets of Σ_k is $2^{|\Sigma_k|}$, A_φ must have at least $2^{|\Sigma_k|}$ states. Thus, for $k = \lceil \frac{n}{2} \rceil$, this means $2^{2^{\Omega(n)}}$ states. ∎

3.4 Generators for LTL($\Box, \Diamond, \wedge, \vee$)

In section 2.2 we recalled the results concerning the construction of a deterministic generator for a given formula in LTL. In this section we prove that a matching lower bound to that construction even in absence of next and until modalities.

Theorem 3.9 *There exists a formula φ in* LTL($\Box, \Diamond, \wedge, \vee$) *such that all the deterministic generators of φ have an $\Omega(exp(exp(|\varphi|)))$ longest distance.*

Proof. Consider the formula

$$\Box(\Diamond \bigwedge_{i=1}^{n} (a_i \vee \Diamond b_i) \rightarrow \Diamond \bigwedge_{i=1}^{n} (c_i \vee \Diamond d_i)),$$

where $a_i, b_i, c_i, d_i \in P$ for $i = 1, \ldots, n$ and $n \geq 2$. Assume that $A_\varphi = (2^P, S, s_0, \delta, F)$ is a deterministic generator for φ. Denote by P_x the set $\{x_1, \ldots, x_n\}$. Moreover, denote by p_j a subset of P_a and by q_j a subset of P_c. By arguments similar to those used in the proof of Theorem 3.8, it is possible to prove that: 1) a deterministic generator for φ has to keep track of the p_j's that have been fulfilled and for each p_j the list of q_h's which have been fulfilled starting at the position where p_j was true the last time; 2) we may need to store exponentially many p_j's and exponentially many q_j's, to check the fulfillment of $\Diamond \bigwedge_{i=1}^{n} (a_i \vee \Diamond b_i)$ and $\Diamond \bigwedge_{i=1}^{n} (c_i \vee \Diamond d_i)$, respectively. Thus for $k = \Omega(2^n)$, let $p_1, \ldots, p_k$ and $q_1, \ldots, q_k$ such sets. We observe that only one among all p_j's (respectively, q_j's) can be true at each position. Every time a p_j is true at a position i, A resets the list for p_j with only the q_h which is true at position i. Every time a q_j is true, A adds q_j to all lists. To conclude the proof it is sufficient to show that there exists a word w in $(P_p \cup P_q \cup \{p_j \cup q_h \,|\, p_j \in P_p, q_h \in Q_p\})^*$ of length 2^k such that the A run on w is such that $r(i) \neq r(j)$ for any $i \neq j$. To see this, we map each state s of A into a binary k-tuple $(x_1, \ldots, x_k)$ such that $x_i = 1$ if and only if q_i is in the list for p_i. Clearly, if two states s and s' are mapped into two different tuples then $s \neq s'$. Moreover, by the above observations, if neither q_i or p_i is true at the current position the i-th bit of the tuple associated to the next A state is the i-th bit of the current state, while if q_i true then the i-th bit becomes 1, otherwise if p_i is true the i-th bit becomes 0. Since at most a p_i and a q_j are true at each position, the tuples of two consecutive states in a run may differ for at most 2 bits. Since it is possible to list all the 2^k binary tuples in such a way two consecutive tuples differs in exactly 1 or 2 bits, we have proved that any deterministic generator for φ has $\Omega(2^k) = \Omega(2^{2^n})$ longest distance. ∎

4 Büchi games

In this section we present a new decision algorithm for Büchi games, which mainly performs a depth-first

traversal of a portion of the game tree and is space-efficient when the longest distance is $O(\frac{n}{\log n})$. Standard techniques to solve Büchi games involve fix-point computation [19], and requires space $O(n)$ no matter what the longest distance is. An interesting aspect of our algorithm is that it can be applied to all the games in which the winning condition can be translated into a deterministic Büchi automaton, as for the formulas in the fragments of LTL we have studied in sections 3.1, 3.2 and 3.3. Then we combine this algorithm with the results on LTL generators from the previous section and study the complexity of the obtained solutions.

In this section we search for winning strategies of $Player_0$, while $Player_1$ will be our adversary. Consider a game graph G and a subset F of G vertices. We denote by Π the set of plays whose last state is the first state which repeats, that is, plays of the form $x_0 \dots x_h$ such that $x_h = x_i$ for some $0 \leq i < h$, and for all $0 \leq i, j < h$, $x_i \neq x_j$. We have that any long-enough play in G has a prefix which is in Π, and each of the plays from Π is constituted by an acyclic prefix followed by a loop. Moreover, we denote by Π_F the set of plays in Π containing a state from F in their loop, and by Π_f the set of plays from Π which can be constructed using the strategy f. We define a game $(G, F)_{fin}$ as the game where $Player_0$ wins from a state u if there is a strategy f from u such that $\Pi_f \subseteq \Pi_F$. Since Büchi games are memoryless, we have:

Lemma 4.1 *There exists a winning strategy for $Player_0$ from a vertex u in a Büchi game (G, F) if and only if there exists a winning strategy for $Player_0$ from u in $(G, F)_{fin}$.*

Directly from the definition of a winning strategy in a game $(G, F)_{fin}$, we have the following lemma.

Lemma 4.2 *Any winning strategy f for $Player_0$ in a game $(G, F)_{fin}$ is such that the lenght of a play in Π_f is $O(d)$, where d is the longest distance of G.*

By the above lemmas, there is a decision algorithm for Büchi games which explores a tree whose height is the longest distance of the game graph.

Theorem 4.3 *Given a game graph G with m vertices and longest distance d, the Büchi game (G, F) is decidable in space $O(d \log m)$.*

Given a game (G, W), if the winning condition W can be translated to a deterministic Büchi automaton, it is possible to use the algorithm by Lemmas 4.1 and 4.2 to decide it. In particular, let A be a deterministic Büchi automaton equivalent to winning condition W, in the sense that the language accepted by A is the language of the ω-words satisfying W. Define $G \times A$ as the game graph whose vertices $V \times Q$, where Q is the set of A states, are partitioned according to the V partition, and from a vertex (v, q) it is possible to reach a vertex (v', q') by taking an action a if and only if A enters q' from q by reading the subset of atomic propositions true at v and in G it is possible to move from v to v' taking the action a. Let F and s_0 be the set of final states and the initial state of A, respectively, then there is a winning strategy in the Büchi game $(G \times A, V \times F)$ starting at a vertex (u, s_0) if and only if there is a winning strategy in (G, W) starting at u.

As a consequence of the results from section 3 and the above argument, Theorem 4.3 applies to games with winning condition expressed by formulas in the LTL fragments we have considered so far. In fact, the following theorems hold.

Theorem 4.4 LTL($\Diamond, \wedge$) *games are* PSPACE*-complete.*

Proof. Membership in PSPACE is a consequence of Theorems 3.3 and 4.3. To prove PSPACE-hardness, we can reduce the satisfiability of quantified boolean formulas in conjunctive normal form to deciding the existence of a winning strategy in an LTL($\Diamond, \wedge$) game. This also shows that LTL($\Box, \vee$) games are PSPACE-hard. Let $\varphi = A_1 x_1 \dots A_n x_n . \bigwedge_{i=1}^{m} c_i$ be a quantified boolean formula over the variables $x_1, \dots, x_n$. Consider the LTL($\Diamond, \wedge$) formula $\varphi' = \bigwedge_{i=1}^{m} \Diamond c_i$ over the atomic propositions $\{c_1, \dots, c_m\}$. The game graph G is defined in such a way that each literal corresponds only to a vertex, a path of the game tree corresponds to the assignment given by assuming true the literals corresponding to its vertices, each vertex is labeled with the conjuncts which contain the corresponding literal, and a strategy corresponds to a selection of paths fulfilling the requirements of quantifiers $A_1, \dots, A_n$. We have that φ is satisfiable if and only if there is a winning strategy in the game (G, φ'). ∎

Theorem 4.5 LTL($\Diamond$,$\bigcirc$,$\wedge$) *games are* EXPTIME*-complete.*

Proof. By Theorem 3.5, LTL($\Diamond$,$\bigcirc$,$\wedge$) has exponentially-sized deterministic generators, and hence, membership in EXPTIME follows. For the lower bound, we reduce the halting problem for alternating linear bounded automata. We briefly sketch the construction. Consider a Turing machine M that uses n tape positions over a tape alphabet Γ, and let Q be the set of control states that are partitioned into Q_0 and Q_1 corresponding to the two players. The transitions of the machine are of the form $\langle q, \sigma, q', \sigma', L/R \rangle$ meaning that if control state is q and current symbol is σ, then the machine can

overwrite the current cell with σ', update control state to q', and move left (L) or right (R). If multiple transitions are applicable, then depending on whether the current control state belongs to Q_0 or Q_1, one of the two players gets to choose the transition. The problem of deciding whether $Player_0$ has a strategy to reach a specified control state, say q_h, is EXPTIME-complete. Given such a machine M, we build a game graph G_M as follows. For every tape symbol σ and position i, G_M has a vertex $v_{\sigma,i}$ belonging to V_1. For every control state q, tape symbol σ and position i, G_M has a vertex $v_{q,\sigma,i}$ belonging to V_0 if q is in Q_0 and to V_1 otherwise. For every control state q, and symbol σ, G_M has a vertex $v_{q,\sigma,L}$ and a vertex $v_{q,\sigma,R}$, both belonging to V_1. For $i < n$, there is an edge from $v_{\sigma,i}$ to every $v_{\sigma',i+1}$. There is an edge from $v_{\sigma,n}$ to every $v_{q,\sigma',i}$. For every transition $\langle q, \sigma, q', \sigma', L/R\rangle$ of M, there is an edge from every $v_{q,\sigma,i}$ to $v_{q',\sigma',L/R}$. Finally, every $v_{q,\sigma,L/R}$ has an edge to every $v_{\sigma',1}$. The intuition is that $Player_1$ chooses a sequence of vertices $v_{\sigma_1,1}, \ldots v_{\sigma_n,n}$, denoting the tape content, followed by a vertex $v_{q,\sigma,i}$, meaning that current control is in state q with head reading symbol σ in position i. The next vertex of the form $v_{q',\sigma',L/R}$ indicates the choice of the transition (and hence, new control state and new symbol in position i, and movement of the head), and is determined by one of the players depending on whether q belongs to Q_0 or Q_1. $Player_0$ wins if either the control state q_h is encountered or $Player_1$ does not make the choices for encoding the configuration according to the intended interpretation. Assume that there are enough propositions to identify each vertex uniquely by a state predicate. Then, the winning condition for $Player_0$ is a top-level disjunction of several formulas that use only eventualities and conjunctions. For instance, a mistake in the encoding of the content of i-th tape position is described by the formula

$$\vee \Diamond(v_{\sigma,i} \wedge \bigcirc^{n-i+1} v_{q,\sigma',i'\neq i} \wedge \bigcirc^{n+2} v_{\sigma''\neq\sigma,i})$$

$$\vee \Diamond(v_{\sigma,i} \wedge \overset{n-i+1}{\bigcirc} v_{q,\sigma,i} \wedge \overset{n-i+2}{\bigcirc} v_{q',\sigma',L/R} \wedge \overset{n+2}{\bigcirc} v_{\sigma''\neq\sigma',i})$$

∎

Theorem 4.6 LTL($\Diamond, \bigcirc, \wedge, \vee$) *games are* EXPSPACE.

Proof. Directly from Theorems 3.7 and 4.3. ∎

5 Conclusions

For the problem of solving infinite games with the winning condition specified by an LTL formula, we have studied the impact of different connectives on the complexity. In the same way as model checking (or satisfiability) is related to translation from LTL to nondeterministic ω-automata, solving games is related to translation from LTL to deterministic ω-automata. We have established that the longest distance, besides the size, of the automaton produced by the translation is an important parameter. The results are summarized in the table of Figure 2 for various fragments [2]. As the table indicates the sources of complexity for games are different from the ones for model checking. The matching lower bounds for the games in the LTL fragments LTL($\Diamond, \wedge, \vee$), LTL($\Diamond, \bigcirc, \wedge, \vee$), and LTL($\Box, \Diamond, \wedge, \vee$) are open problems, while the results on the corresponding deterministic generators are tight with respect to both the size and the longest distance. We observe that LTL($\Box, \Diamond, \wedge, \vee$) and thus LTL, formulas may not have deterministic Büchi generators, but it is known that they have doubly-exponential deterministic Streett generators.

Besides the classification of complexity of games for various fragments, the constructions of this paper can be used to solve synthesis problems for certain kinds of formulas more efficiently. In particular, the fragments LTL($\Diamond, \wedge$) and LTL($\Diamond, \wedge, \vee$) contains many commonly occuring specifications that are boolean combinations of safety and guarantee properties, and for these, we have provided a direct construction of deterministic generators in a modular manner.

References

[1] R. Alur, L. de Alfaro, T. Henzinger, and F. Mang. Automating modular verification. In *CONCUR'99: Concurrency Theory, Tenth Int. Conference*, LNCS 1664, pages 82–97, 1999.

[2] R. Alur, T.A. Henzinger, and O. Kupferman. Alternating-time temporal logic. In *Proc. of the 38th IEEE Symposium on Foundations of Computer Science*, pages 100 - 109, 1997.

[3] R. Alur, T. Henzinger, F. Mang, S. Qadeer, S. Rajamani, and S. Tasiran. MOCHA: Modularity in model checking. In *Proc. of the Tenth Int. Conference on Computer Aided Verification*, LNCS 1427, pages 521 - 525. Springer-Verlag, 1998.

[4] M. Abadi, L. Lamport, and P. Wolper. Realizable and unrealizable specifications of reac-

[2] Some of the entries in the table concerning nondeterministic generators and model checking are not explicitly stated in the literature, and will be explained in detail in the full paper.

	Nondet. Generators		Det. Generators		Model Checking	Games
	Size	Long. Distance	Size	Long. Distance		
LTL($\Diamond, \wedge$)	Θ(EXP)	Θ(LINEAR)	Θ(EXP)	Θ(LINEAR)	NP-complete	PSPACE-complete
LTL($\Diamond, \bigcirc, \wedge$)	Θ(EXP)	Θ(EXP)	Θ(EXP)	Θ(EXP)	PSPACE-complete	EXPTIME-complete
LTL($\Diamond, \wedge, \vee$)	Θ(EXP)	Θ(LINEAR)	Θ(2EXP)	Θ(EXP)	NP-complete	EXPSPACE
LTL($\Diamond, \bigcirc, \wedge, \vee$)	Θ(EXP)	Θ(EXP)	Θ(2EXP)	Θ(EXP)	PSPACE-complete	EXPSPACE
LTL($\Box, \Diamond, \wedge, \vee$)	Θ(EXP)	Θ(LINEAR)	Θ(2EXP)	Θ(2EXP)	NP-complete	2EXPTIME
LTL	Θ(EXP)	Θ(EXP)	Θ(2EXP)	Θ(2EXP)	PSPACE-complete	2EXPTIME-complete

Figure 2: Complexity results in LTL.

tive systems. In *Proc. of the 16th Int. Colloquium on Automata, Languages and Programming, ICALP'89*, LNCS 372, pages 1–17, 1989.

[5] D.L. Dill. *Trace Theory for Automatic Hierarchical Verification of Speed-independent Circuits.* ACM Distinguished Dissertation Series. MIT Press, 1989.

[6] S. Demri and Ph. Schnoebelen. The complexity of propositional linear temporal logics in simple cases. In *Proc. of the 15th Annual Symposium on Theoretical Aspects of Computer Science, STACS'98*, LNCS 1373, pages 61 – 72. Springer-Verlag, 1998.

[7] E.A. Emerson and C.S. Jutla. The complexity of tree automata and logics of programs. In *Proc. of the 29th IEEE-CS Symposium on Foundations of Computer Science*, pages 328 – 337, 1988.

[8] E.A. Emerson. Temporal and modal logic. In *Handbook of Theoretical Computer Science*, volume B, pages 995 – 1072. Elsevier Science Publishers, 1990.

[9] O. Kupferman and M.Y. Vardi. Module checking. In *Computer Aided Verification, Proc. Eighth Int. Workshop*, LNCS 1102, pages 75 – 86. Springer-Verlag, 1996.

[10] O. Kupferman and M.Y. Vardi. Module checking revisited. In *Proc. of the Ninth Int. Conference on Computer Aided Verification, CAV'97*, LNCS 1254, pages 36 –47, 1997.

[11] O. Kupferman and M.Y. Vardi. Freedom, weakness, and determinism: From linear-time to branching-time. In *Proc. of the 13th IEEE Symposium on Logic in Computer Science*, pages 81 – 92, 1998.

[12] O. Lichtenstein and A. Pnueli. Checking that finite-state concurrent programs satisfy their linear specification. In *Proc. of the 12th ACM Symposium on Principles of Programming Languages*, pages 97 – 107, 1985.

[13] Z. Manna and A. Pnueli. *The temporal logic of reactive and concurrent systems: Specification.* Springer-verlag, 1991.

[14] A. Pnueli. The temporal logic of programs. In *Proc. of the 18th IEEE Symposium on Foundations of Computer Science*, pages 46 – 77, 1977.

[15] A. Pnueli and R. Rosner. On the synthesis of a reactive module. In *Proc. of the 16th ACM Symposium on Principles of Programming Languages*, pages 179 – 190, 1989.

[16] M.O. Rabin. Automata on infinite objects and Church's problem. *Trans. Amer. Math. Soc.*, 1972.

[17] S. Safra. On the complexity of ω-automata. In *Proc. of the 29th IEEE Symposium on Foundations of Computer Science*, pages 319 – 327, 1988.

[18] A.P. Sistla and E.M. Clarke. The complexity of propositional linear temporal logics. *The Journal of the ACM*, 32:733 – 749, 1985.

[19] W. Thomas. On the synthesis of strategies in infinite games. In *12th Annual Symposium on Theoretical Aspects of Computer Science, STACS'95*, LNCS 900, pages 1 – 13. Springer-Verlag, 1995.

[20] M.Y. Vardi. Verification of concurrent programs: the automata-theoretic framework. In *Proc. of the Second IEEE Symposium on Logic in Computer Science*, pages 167 – 176, 1987.

[21] M.Y. Vardi and P. Wolper. Reasoning about infinite computations. *Information and Computation*, 115:1 – 37, 1994.