Deterministic Generators and Games for LTL Fragments

RAJEEV ALUR
University of Pennsylvania
and
SALVATORE LA TORRE
University of Pennsylvania and Università degli Studi di Salerno

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 of the boolean combinations of formulas obtained only by 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 give 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. We also show tightness of the double exponential bound on the size as well as the longest distance for deterministic generators of LTL formulas without *next* and *until* modalities. Finally, we identify a class of deterministic Büchi automata corresponding to a fragment of LTL with restricted use of *always* and *until* modalities, for which deciding games is PSPACE-complete.

Categories and Subject Descriptors: F.4.1 [**Mathematical logic and Formal Languages**]: Mathematical Logic—*Temporal logic*; F.3.1 [**Logics and Meanings of Programs**]: Specifying,Verifying and Reasoning about Programs—*Logics of programs.*

General Terms: Theory, Verification.

Additional Key Words and Phrases: Games, Temporal Logic, Automata.

A preliminary version of this paper appeared in *LICS'01: Proceedings of the 16th Annual IEEE Symposium on Logic in Computer Science*, IEEE Computer Society Press, pp. 291 - 300.
This research was partially supported by NSF Career award CCR97-34115, NSF award CCR99-70925, NSF award ITR/SY 0121431, SRC award 99-TJ-688, and Alfred P. Sloan Faculty Fellowship. The second author was also supported in part by the M.U.R.S.T. in the framework of project TOSCA.
Authors' address: R. Alur and S. La Torre: Dept. of Computer and Information Science, University of Pennsylvania, 200 South 33rd St., Philadelphia, PA 19104, email: `alur@cis.upenn.edu`, `latorre@seas.upenn.edu`.

1. INTRODUCTION

Linear temporal logic (LTL) is a popular choice for specifying correctness requirements of reactive systems [Pnu77; MP91]. 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 (nondeterministic) Büchi automaton that accepts all of its satisfying models [LP85; VW94]. 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 [SC85].

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 [PR89], *realizability* of liveness specifications [ALW89], and *receptiveness* [Dil89]. In the model-checking game, we are given an 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 in all environment states. This problem appears in contexts such as *module checking* and its variants [KV96; KV97], and the definition of *alternating temporal logic* [AHK97]. Such game-based model checking for restricted formulas such as "always p" has already been implemented in the software MOCHA [AHM+98], and shown to be useful in construction of the most-general environments for automating assume-guarantee reasoning [AdAHM99].

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 from a given initial state? The 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 [PR89; Ros92]. 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) [Rab72; EJ88]. 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 [SC85] (see also [DS98] 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 need to be doubly exponential [KV98].

In this paper, we initiate 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 the 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 polynomial space, 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 exponentially many states as well as exponential longest distance. 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 exponential-time 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 $\text{LTL}(\Diamond, \bigcirc, \wedge, \vee)$

[1] In the automata-theoretic formulation of the problem [PR89], 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.

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 LTL($\Diamond, \wedge, \vee$) formula whose deterministic generator must be doubly exponential with singly exponential longest distance. Our construction leads to an algorithm for solving games in LTL($\Diamond, \bigcirc, \wedge, \vee$) that uses exponential space. A matching lower bound has been recently proved in [MT02]. It is worth noticing that if the next modality is not allowed in this fragment, we do not get better lower bounds on both the size and the longest distance of corresponding deterministic Büchi automata. On the other hand, games in the positive fragment of LTL($\Diamond, \wedge, \vee$) (that is, the fragment of LTL($\Diamond, \wedge, \vee$) where negation is allowed only at the level of the atomic propositions) can be solved using polynomial space [MT02]. A matching lower bound or a better upper bound for games in the full LTL($\Diamond, \wedge, \vee$) remains an open problem.

The nesting of eventually and always modalities causes a further increase of the complexity. We prove that there exists a formula in LTL($\Box, \Diamond$) 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($\Box, \Diamond$) formulas is only linear, and becomes exponential only by addition of next or until modalities. Our construction leads to an algorithm for solving LTL($\Box, \Diamond$) games in doubly exponential time. A matching lower bound or a better upper bound for these games remains an open problem.

The automata that we construct for LTL($\Diamond, \wedge$) and LTL($\Diamond, \wedge, \vee$) formulas have a special form: the underlying graph is acyclic modulo the self-loops. We define the class of *partially-ordered deterministic Büchi automata* (PODB) as the class of deterministic Büchi automata with such a structure. The class PODB is closed under all boolean operations, strictly more expressive than LTL($\Diamond, \wedge, \vee$), and incomparable to LTL($\Diamond, \bigcirc, \wedge$). We show that PODBs are as much expressive as the LTL fragment obtained by imposing the following two restrictions: in *always* formulas $\Box\varphi$, φ must be a state predicate, and in *until* formulas $\varphi \,\mathcal{U}\, \psi$, φ must be a state predicate and $\neg\varphi$ must be a conjunct of ψ. Deciding games for this fragment is PSPACE-complete.

The rest of this paper is organized as follows. In Section 2 we give the notation and recall some known results we will use throughout this paper. In Section 3 we introduce a class of deterministic automata that we call PODB, and we show that some interesting fragments of LTL admit deterministic generators in this class of automata. In Section 4, we study the problem of constructing the deterministic generators for larger fragments of LTL. In Section 5 we give a new algorithm to solve Büchi games that we use along with the deterministic generators to solve the games in the LTL fragments we consider. We also prove matching lower bounds for some of them. Finally, we give a few conclusions in Section 6.

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

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

where p is a state predicate. 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. Let $w = w_0 w_1 w_2 \ldots$ be an ω-word, with w^i we denote the subsequence of w starting at position i, that is, the mapping defined by $w^i_n = w_{i+n}$. LTL formulas are interpreted on an ω-word w over the alphabet $\Sigma = 2^P$ and the satisfaction relation $w \models \varphi$ is defined in the standard way:

—if φ is a state predicate, then $w \models \varphi$ if and only if the assignment of atomic propositions specified by w_0 satisfies φ;

—$w \models \varphi \wedge \psi$ if and only if $w \models \varphi$ and $w \models \psi$;

—$w \models \varphi \vee \psi$ if and only if $w \models \varphi$ or $w \models \psi$;

—$w \models \bigcirc \varphi$ if and only if $w^1 \models \varphi$;

—$w \models \Diamond \varphi$ if and only if $\exists i : w^i \models \varphi$;

—$w \models \Box \varphi$ if and only if $\forall i : w^i \models \varphi$;

—$w \models \varphi \,\mathcal{U}\, \psi$ if and only if $\exists i : w^i \models \varphi$ and $w^j \models \psi$ for all j such that $0 \leq j < i$.

Obviously, the eventually modality is a restricted form of until (i.e., $\Diamond \varphi \equiv \textsc{True}\, \mathcal{U}\, \varphi$ holds), and the always modality expresses the logical negation of the eventually modality (i.e., $\Diamond \varphi \equiv \neg(\Box \neg \varphi)$ holds).

2.2 Fragments of LTL

In the rest of the paper we consider some fragments of LTL. We denote by $\text{LTL}_+(op_1, \ldots, op_k)$ the logic of formulas built from the state predicates by using only the boolean connectives and the temporal operators in the list $op_1, \ldots, op_k$. Furthermore, we denote by $\text{LTL}(op_1, \ldots, op_k)$ the logic of formulas obtained as boolean combinations of $\text{LTL}_+(op_1, \ldots, op_k)$ formulas. As an example of this notation consider the LTL fragment $\text{LTL}(\Diamond, \wedge)$. The logic $\text{LTL}_+(\Diamond, \wedge)$ is defined by the grammar

$$\varphi := p \,|\, \varphi \wedge \varphi \,|\, \Diamond \varphi,$$

where p is a state predicate, and the fragment $\text{LTL}(\Diamond, \wedge)$ is defined by the grammar

$$\varphi := \psi \,|\, \neg\, \varphi \,|\, \varphi \wedge \varphi \,|\, \varphi \vee \varphi,$$

where ψ is a formula of $\text{LTL}_+(\Diamond, \wedge)$. Thus, $\text{LTL}(\Diamond, \wedge)$ is the fragment of LTL containing the boolean combinations of formulas built from state predicates using only eventualities and conjunctions. Notice that negations and disjunctions are allowed only at the top-level and at the atomic level, and by definition, $\text{LTL}(\Diamond, \wedge)$ is equivalent to $\text{LTL}(\Box, \vee)$. A sample formula of this fragment is $\Box\, p \vee \Diamond(q \wedge \Diamond\, r)$.

2.3 Finite automata on ω-words

Automata on ω-words have been extensively studied in relation to temporal logic [Eme90]. 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 a 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 $\{w \,|\, w \text{ is accepted by } A\}$.

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 φ of doubly exponential size 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 $2^{O(|\varphi|)}$ [LP85; VW94]; we recall that a Büchi automaton of size n can be determinized and the resulting deterministic Rabin automaton has $2^{O(n \log n)}$ states and n pairs [Saf88]; thus we determinize the Büchi generator for φ so obtaining a deterministic Rabin generator of doubly exponential size with exponentially many pairs. 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 [KV98], and thus, the above construction is asymptotically optimal.

2.4 Game graphs

In this section we will introduce the notation concerning Büchi and LTL games. A *game graph* is a tuple $G = (V, V_0, V_1, \gamma)$ where V is a finite or countable set of vertices, V_0 and V_1 define a partition of V, and $\gamma : V \to 2^V$ is a function giving for each vertex $u \in V$ the set of its *successors* in G. 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 *play* of a game graph G is constructed as a sequence of vertices selected by the two players. Formally, a play starting at x_0 is a sequence $x_0x_1 \dots x_h$ in V^* such that $x_j \in \gamma(x_{j-1})$, for $j = 1, \dots, h$.

In this paper, we give an asymmetric definition of games, and we focus on $Player_0$, the *protagonist*, while $Player_1$ will be the *adversary* (or *antagonist*). A *strategy* from a vertex u is a total function f mapping a play starting from u and ending in V_0 into V, i.e., it gives the moves of $Player_0$ in any play ending in V_0. A play $x_0 \dots x_n$ is constructed according to f if for any $x_j \in V_0$, $j \in \{0, \dots, n-1\}$, $f(x_0 \dots x_j) = x_{j+1}$. When a strategy depends only on the last vertex of a play, it is called a *memoryless strategy*. A strategy f from u defines an ω-tree (*strategy tree*), where each node corresponds to one of the plays constructed according to f: 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 legal continuation of the play $x_0 \dots x_h$, i.e., to a play $x_0 \dots x_h x_{h+1}$ such that $x_{h+1} \in \gamma(x_h)$, if $x_h \in V_1$, and $f(x_0 \dots x_h) = x_{h+1}$, otherwise. A node of a strategy tree, corresponding to a play $x_1 \dots x_h$, is labeled with the last vertex of the play, that is, x_h.

A *Büchi game* is a pair (G, F) where G is a game graph and F is a subset of G vertices. Given a Büchi game (G, F), a *winning strategy* from u is a strategy f such that, on every path of the strategy tree corresponding to f, there is a vertex from F that repeats infinitely often (*Büchi winning condition*).

To define LTL games, we introduce the concept of labeled game graph. A Σ-*labeled game graph* is a game graph G along with a function μ labeling each vertex of G with a subset of Σ. An LTL *game* is a pair (G, φ) where G is a P-labeled game graph and φ is an LTL formula over the set of atomic propositions P. For strategy trees in LTL games, a node corresponding to a play $x_1 \dots x_h$ is labeled with $\mu(x_h)$. Given an LTL game (G, φ), a *winning strategy* from u is a strategy f such that φ is satisfied on all paths of the strategy tree corresponding to it.

Given a game (G, 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 memoryless 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 $|\varphi|$ [PR89].

3. PARTIALLY-ORDERED DETERMINISTIC GENERATORS

We begin this section by introducing a proper subclass of deterministic Büchi automata whose transitive closure of the 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.

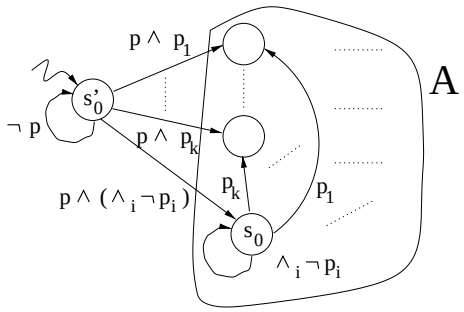

Fig. 1. Graphical representation of the automaton $A^{\Diamond(p,A)}$.

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, $L(A_1) \cup L(A_2)$), whose size is $O(n_1 \cdot n_2)$ and 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 $\Sigma^\omega \setminus L(A_i)$.*

PROOF. For $i = 1, 2$, let F_i be the set of accepting states of A_i. Consider first the closure of PODBs under the union. The automaton A accepting $L(A_1) \cup L(A_2)$ is obtained by the usual cross-product construction for deterministic Büchi automata. To prove that A is a PODB, we observe that if there exists a cycle of A which is not a self-loop then immediately from the construction of A we get that either for $i = 1$ or $i = 2$, there exists a cycle of A_i which is not a self-loop. To complete the proof for the union, we need to show that the longest distance of A is not greater than $d_1 + d_2$. For this purpose, consider a simple path $\pi = (a_1, b_1) \dots (a_m, b_m)$ of A. Since π is simple, $a_i \neq a_{i+1}$ or $b_i \neq b_{i+1}$ must hold for every $i = 1, \dots, m-1$. Being A_1 and A_2 both PODBs, the number of times "a_i is not equal to a_{i+1}" cannot be larger than d_1 and the number of times "b_i is not equal to b_{i+1}" cannot be larger than d_2. Thus the longest distance of A is not greater than $d_1 + d_2$.

For the intersection we can use similar arguments. We just observe that differently from the usual construction for deterministic Büchi automata, the automaton accepting $L(A_1) \cap L(A_2)$ is given by the cross-product of A_1 and A_2 along with the set of accepting states $F_1 \times F_2$. This simple construction works for PODBs since for this class of automata along any infinite run only one state can repeat infinitely often and thus the conditions "a state repeats infinitely often" and "a state eventually repeats forever" are equivalent. For the same reason, the complementation of a PODB is obtained by simply complementing the set of accepting states, as in the case of deterministic automata on finite words. □

The above results on intersection and union are naturally extended to k-tuples 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.

Let φ be a formula and L be the set of models of φ. Suppose that $\Sigma L \subseteq L$ holds. If a deterministic generator for φ is known, we can easily construct a deterministic generator for $\Diamond(p \wedge \varphi)$, where p is a state predicate. In fact, since $\Sigma L \subseteq L$, to accept all the models of $\Diamond(p \wedge \varphi)$, it is sufficient to start a generator for φ

as soon as p becomes true. Such a construction is clearly deterministic and is formally described as follows. Let $A = (\Sigma, S, s_0, \delta, F)$ be a (deterministic) Büchi automaton and $s'_0 \notin S$, we define the (deterministic) Büchi automaton $A^{\Diamond(p,A)}$ as $(\Sigma, S \cup \{s'_0\}, s'_0, \delta', F)$ where:

(1) $\delta'(s, a) = \delta(s, a)$ for $s \in S$,

(2) $\delta'(s'_0, a) = \delta(s_0, a)$ for a satisfying p, and

(3) $\delta'(s'_0, a) = s'_0$, otherwise.

The above 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 $[p] = \{a \in \Sigma \mid a \text{ satisfies } p\}$. Moreover, if A is a PODB then $A^{\Diamond(p,A)}$ is also a PODB.*

PROOF. It is easy to verify that $A^{\Diamond(p,A)}$ has size $O(n)$ and longest distance $d+1$, and that the above construction preserves the determinism and the PODB property of A (see Figure 1).

To complete the proof we need to show that $L(A^{\Diamond(p,A)}) = \Sigma^*\,[p]\,L(A)$. Let $v \in \Sigma^*\,[p]\,L(A)$. Since $\Sigma L(A) \subseteq L(A)$, we can assume that $v = w_1 a w_2$ where $w_1 \in (\Sigma \setminus [p])^*$, $a \in [p]$, and $w_2 \in L(A)$ (i.e., a is the first occurrence in v of a symbol from $[p]$). By rule 3, $A^{\Diamond(p,A)}$ reads w_1 staying in the starting state s'_0, then by rule 2 enters a state of A on a, and behaves as A on w_2. Since $w_2 \in L(A)$, we get that $v \in L(A^{\Diamond(p,A)})$. The proof of the inclusion $L(A^{\Diamond(p,A)}) \subseteq \Sigma^*\,[p]\,L(A)$ is analogous. □

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. Negations and disjunctions are allowed only at the top-level and at the atomic level, thus 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($\Diamond, \wedge$) and it asserts that each one of $p_1, \ldots, p_n$ has to be true at least once. 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. We observe that each formula φ in LTL$_+$($\Diamond, \wedge$) is 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$_+$($\Diamond, \wedge$).

Fig. 2. PODBs for p and $\Diamond p$.

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 size exponential in $|\varphi|$ and longest distance linear in $|\varphi|$.*

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 any formula in LTL$(\Diamond, \wedge)$. For a state predicate p, we define A_p and $A_{\Diamond p}$ as in Figure 2. It is easy to verify that A_p (respectively $A_{\Diamond p}$) is a PODB accepting all the models of p (respectively $\Diamond p$). Clearly, $\Sigma^* L(A_{\Diamond p}) \subseteq L(A_{\Diamond p})$ also holds. 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 $A_{\Diamond\gamma}$ has $O(2^{|\Diamond\gamma|})$ size and $O(|\Diamond\gamma|)$ longest distance. Thus, by Proposition 3.1, $A' = A_{\Diamond\psi_1} \cap \ldots \cap A_{\Diamond\psi_k}$ is a PODB of $O(2^{|\Diamond\psi_1|+\ldots+|\Diamond\psi_k|})$ size and $O(|\Diamond\psi_1| + \ldots + |\Diamond\psi_k|)$ longest distance, accepting all the models of $\Diamond\psi_1 \wedge \ldots \wedge \Diamond\psi_k$. Clearly, since $\Sigma^* L(A_{\Diamond\psi_i}) \subseteq L(A_{\Diamond\psi_i})$ trivially holds for every $i = 1, \ldots, k$, we have that $\Sigma^* L(A') \subseteq L(A')$ also holds. Thus, from Proposition 3.2, we get that $A_\psi = A^{\Diamond(p,A')}$ is a PODB accepting all models of ψ such that the size of A_ψ is exponential in $|\psi|$ and the longest distance of A_ψ is linear in $|\psi|$. □

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

THEOREM 3.4. *There exists a formula φ in* LTL$(\Diamond, \wedge)$ *such that all generators of φ have $2^{\Omega(|\varphi|)}$ size and $\Omega(|\varphi|)$ longest distance.*

PROOF. Consider a formula $\varphi = \Diamond p_1 \wedge \ldots \wedge \Diamond p_n$, where $P = \{p_1, \ldots, p_n\}$ and $n \geq 2$. Clearly, $|\varphi| = O(n)$ holds. First we prove that a generator for φ has at least 2^n states. Let $A_\varphi = (2^P, S, s_0, \Delta, F)$ be a generator for φ. For any $c \in 2^P$, we denote by $\overline{c}$ the complement of c with respect to P, by w_c the ω-word $\overline{c}\emptyset \ldots \emptyset \ldots$, and by s_c a state of A_φ such that $(s_0, c, s_c) \in \Delta$ and there exists an accepting run of A_φ on cw_c starting with the transition (s_0, c, s_c). Consider $a, b \in 2^P$ such that $a \neq b$. We have that bw_a is not a model of φ, if $a \not\subseteq b$, and aw_b is not a model of φ, if $b \not\subseteq a$. Being A_φ a generator of models for φ, we get that A_φ does not accept w_a starting from s_b, in the first case, and w_b starting from s_a, in the second case. Thus for $a \neq b$ we get that $s_a \neq s_b$ and hence we have proved that A_φ has at least 2^n states.

For the second assertion, consider the ω-word $w = \{p_1\}\{p_2\} \ldots \{p_n\}^\omega$. Clearly,

w is a model of φ. Suppose that A_φ is a generator of φ with longest distance less than n and let r be an accepting run on w. Thus there exist $0 \leq i < j < n$ such that $r(i) = r(j)$. Then, $r(0) \dots r(i-1) r(j) r(j+1) \dots$ is an accepting run of A_φ on $w' = \{p_1\} \dots \{p_{i-1}\}\{p_j\}\{p_{j+1}\} \dots \{p_n\}^\omega$, but w' is not a model of φ, and this contradicts the hypothesis that A_φ is a generator for φ. □

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

The fragment LTL($\Diamond, \wedge, \vee$) contains boolean combinations of formulas built from state predicates using eventualities, disjunctions, and conjunctions.

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 $i = 1, \dots, n$. Obviously φ is an LTL($\Diamond, \wedge, \vee$) formula. This formula asserts that all the clauses $(p_i \vee \Diamond q_i)$ have to be fulfilled at the same position in the model. 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 exponential size and linear longest distance. To obtain a deterministic generator for φ we can determinize this strategy. A way to do this is to store in each state the family of sets of q_i's whose fulfillment at later positions can guarantee the fulfillment of φ. Then, the corresponding deterministic automaton has 2^{2^n} states (that is, not more than the set of parts of $\{q_1, \dots, q_n\}$). Clearly, the fulfillment of all propositions in a set A implies the fulfillment of all propositions in any subset of A. Thus we do not need to store in the next state a set of q_i's which contains one of the sets stored in the current state and if a q_i is fulfilled it can be deleted by any stored set. As a consequence the longest distance of this automaton is at most 2^n. It is possible to prove that this result indeed holds for all LTL($\Diamond, \wedge, \vee$) formulas, as stated by the following theorem.

THEOREM 3.5. *There exists a deterministic Büchi automaton A accepting all the models of a formula φ in* LTL($\Diamond, \wedge, \vee$) *such that A is a PODB of size doubly exponential in $|\varphi|$ and longest distance exponential in $|\varphi|$.*

PROOF. We observe that an LTL($\Diamond, \wedge, \vee$) formula φ can be translated into an equivalent LTL($\Diamond, \wedge$) formula φ' such that $|\varphi'| = O(2^{|\varphi|})$. This can be done by pushing outside all the "or" connectives in φ using the following equivalences: $\psi_1 \wedge (\psi_2 \vee \psi_3) \equiv (\psi_1 \wedge \psi_2) \vee (\psi_1 \wedge \psi_3)$, $(\psi_1 \vee \psi_2) \wedge \psi_3 \equiv (\psi_1 \wedge \psi_3) \vee (\psi_2 \wedge \psi_3)$, $\Diamond(\psi_1 \vee \psi_2) \equiv \Diamond \psi_1 \vee \Diamond \psi_2$, and $\Diamond(\Diamond \psi) \equiv \Diamond \psi$. By Theorem 3.3 there exists a PODB A accepting all the models of φ' such that its size is $O(2^{|\varphi'|})$ and its longest distance is $O(|\varphi'|)$. Thus by $\varphi \equiv \varphi'$ and $|\varphi'| = O(2^{|\varphi|})$, the statement of the theorem is proved. □

The following theorem shows that for some LTL($\Diamond, \wedge, \vee$) formula, we may not be able to have a deterministic generator smaller than that given by the above construction.

THEOREM 3.6. *There exists a formula φ in* LTL($\Diamond, \wedge, \vee$) *such that all the deterministic generators of φ have $2^{2^{\Omega(|\varphi|)}}$ size and $2^{\Omega(|\varphi|)}$ longest distance.*

PROOF. Consider a formula $\varphi = \Diamond \bigwedge_{i=1}^{n} (p_i \vee \Diamond q_i)$, where $p_i, q_i \in P$ for $i = 1, \dots, n$ and $n \geq 2$. Obviously, $|\varphi| = O(n)$. Denote by P_p the set $\{p_1, \dots, p_n\}$ and

by P_q the set $\{q_1, \ldots, q_n\}$. Let $A_\varphi = (2^P, S, s_0, \delta, F)$ be a deterministic generator for φ. First we prove that A_φ has $2^{2^{\Omega(n)}}$ states, then we show that it has a simple path of length $2^{\Omega(n)}$.

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 any $w, w' \in \Sigma_k^*$ 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, without loss of generality we can assume 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.

For the second assertion, consider a word $w' = b_1 b_2 \ldots b_{m(k)}$, where $\{b_1, \ldots, b_{m(k)}\}$ is the set Σ_k defined above. It holds that $w_i = w'q(b_i)w$ is a model of φ for all $w \in (2^P)^\omega$ and $i = 1, \ldots, m(k)$. Denote by r_i the run of A_φ on w_i. Since A_φ is deterministic, for any $i, j \in \{1, \ldots, m(k)\}$, we get that $r_i(h) = r_j(h)$ for all $h = 0, \ldots, m(k)$. Now suppose that A_φ has longest distance less than $m(k)$. There exist h and l such that $0 \leq h < l < n$ and $r_i(h) = r_j(l)$ for all $i, j \in \{1, \ldots, m(k)\}$. As a consequence, the run obtained from r_h by replacing the cycle $r_h(h) \ldots r_h(l)$ by $r_h(l)$, is an accepting run of A_φ on $w'' = b_1 \ldots b_{h-1} b_l \ldots b_{m(k)} q(b_h)\emptyset \ldots \emptyset$, but w'' is clearly not a model of φ and this contradicts the above hypothesis on the longest distance of A_φ. Moreover, since $m(\lceil \frac{n}{2} \rceil) = 2^{\Omega(n)}$, we are done. □

3.3 Characterization of PODB in LTL

In this section we introduce an LTL fragment, denoted by LTL^{PODB}, which is equivalent to PODBs, in the sense that given a formula φ in LTL^{PODB} there is a PODB which is a generator of models for φ and, vice-versa, given a PODB A there is formula φ in LTL^{PODB} such that $L(A)$ is the set of all the models of φ. We prove that the translations from LTL^{PODB} formulas into PODBs and vice-versa may be exponential. We also show that LTL^{PODB} is strictly more expressive than $\text{LTL}(\Diamond, \wedge, \vee)$.

The syntax of LTL^{PODB} is given by the following grammar:

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

The formula $p \,\mathcal{U}' \, \varphi$ asserts that p has to be true until φ becomes true and at the same position p becomes false. The semantics of this temporal operator $\mathcal{U}'$ is then given by the equivalence

$$p \,\mathcal{U}' \, \varphi \equiv p \,\mathcal{U}\, (\neg p \wedge \varphi).$$

The logic LTL^{PODB} is closed under logical negation.

PROPOSITION 3.7. *For any* LTL^{PODB} *formula* φ *there exists an* LTL^{PODB} *formula* φ' *such that* $\varphi' \equiv \neg\varphi$ *and* $|\varphi'| = O(|\varphi|)$.

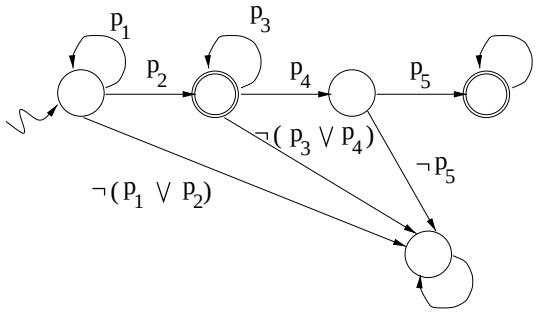

Fig. 3. Graphical representation of the PODB A.

PROOF. The following equivalences hold:

$$\begin{aligned} \neg \bigcirc p &\equiv \bigcirc \neg p \\ \neg \Box p &\equiv \neg(\neg p \; \mathcal{U}' \; p) \\ \neg(p \; \mathcal{U}' \; \varphi) &\equiv \Box p \vee (\neg p \; \mathcal{U}' \; \neg\varphi) \end{aligned}$$

□

Consider the PODB A in Figure 3. A corresponds to the LTL^{PODB} formula

$$p_1 \; \mathcal{U}' \; (p_2 \wedge \bigcirc(\Box \, p_3 \vee (p_3 \; \mathcal{U}' \; (p_4 \wedge \bigcirc p_5)))).$$

The following theorem holds.

THEOREM 3.8. *There exists a deterministic Büchi automaton A accepting all the models of a formula φ in* LTL^{PODB} *such that A is a PODB of size exponential in $|\varphi|$ and longest distance linear in $|\varphi|$.*

PROOF. We prove this result by induction on the structure of a formula in LTL^{PODB}. The base cases p and $\Box \, p$ are trivial. By Proposition 3.1 the induction directly holds for formulas of type $\varphi \wedge \varphi'$ and $\varphi \vee \varphi'$. Suppose now by induction that for a formula φ, A_φ is a PODB of size exponential in $|\varphi|$ and longest distance linear in $|\varphi|$ generating all the models of φ. A PODB for $\bigcirc \varphi$ can be obtained by adding a new state s to A_φ along with transitions on any input to the initial state of A_φ, and choosing s as the new initial state. This new PODB obviously has size exponential in $|\varphi|$ and longest distance linear in $|\varphi|$. Since $(p \; \mathcal{U}' \; \varphi)$ is equivalent to $\Diamond(\neg p \wedge \Diamond \varphi)$, a PODB for $(p \; \mathcal{U}' \; \varphi)$ is $A^{\Diamond(\neg p, A_\varphi)}$, and clearly satisfies the stated property on the size and longest distance. □

By the equivalence $\Diamond \, p \equiv (\neg p \; \mathcal{U}' \; p)$, the formula $\varphi = \Diamond \, p_1 \wedge \ldots \wedge \Diamond \, p_n$, where $p_i \in P$ for $i = 1, \ldots, n$, is equivalent to an LTL^{PODB} formula of size linear in $|\varphi|$. We recall that a generator for this formula needs at least 2^n states and its longest distance is at least n (see proof of Theorem 3.4). Therefore, the result stated in Theorem 3.8 is optimal in the sense that there exists an LTL^{PODB} formula for which we cannot find a smaller generator.

The proof of the claimed equivalence between PODBs and LTL^{PODB} formulas is completed by the following theorem.

THEOREM 3.9. *Given a PODB A there exists an* LTL^{PODB} *formula φ such that A is a generator of φ models and $|\varphi| = O(2^{|A|})$.*

PROOF. We prove this result by induction on the number of states of a PODB. The base case is a PODB having only one state. Since it is deterministic it has also a self-loop. If the only state is also an accepting one, then the corresponding formula is TRUE, and it is FALSE, otherwise. Suppose now that any PODB B with at most n states can be translated into an LTL^{PODB} formula of size $O(2^{|B|})$, and let A be a PODB with $n+1$ states. Denote by s_0 the starting state of A. For $i = 1, \ldots, k$, let (s_0, p_i, s_i) be all the transitions from s_0 for $s_i \neq s_0$ and (s_0, p_0, s_0) be the self-loop on s_0 in case there is one. Since A is deterministic, $p_i \wedge p_j$ does not hold for any $i \neq j$. Moreover, denote by A_i the automaton obtained from A by removing all the states which are not reachable from s_i and considering s_i as the starting state. Since A is a PODB s_0 is not a state of A_i, therefore A_i has at most n states and $|A_i| < |A| - k$ (we remove at least k transitions from A). By inductive hypothesis, A_i is a generator for an LTL^{PODB} formula φ_i of size $O(2^{|A_i|})$. There are three different cases depending on whether s_0 has a self-loop or not, and in the positive case whether it is also an accepting state or not. If s_0 does not have a self-loop, then the formula corresponding to A is $\varphi = \bigvee_{i=1}^{k}(p_i \wedge \bigcirc \varphi_i)$. Suppose now that s_0 has a self-loop and is not final. Thus A is equivalent to the formula $\varphi = p_0 \,\mathcal{U}'\, (\bigvee_{i=1}^{k}(p_i \wedge \bigcirc \varphi_i))$, where p_0 is the state predicate of the self-loop on s_0. In the last case, the formula corresponding to A is $\varphi = \Box\, p_0 \vee (p_0 \,\mathcal{U}'\, (\bigvee_{i=1}^{k}(p_i \wedge \bigcirc \varphi_i)))$. In all the three cases, the size of φ is $O(k\, 2^{|A|-k})$ and thus $O(2^{|A|})$. □

We conclude this section by comparing the expressiveness of the LTL fragments LTL^{PODB} and LTL$(\Diamond, \wedge, \vee)$. In the next lemma we show that LTL$(\Diamond, \wedge)$ formulas can be translated to LTL^{PODB} formulas with at most a linear blow-up of the size.

LEMMA 3.10. *Given a formula φ in* LTL$(\Diamond, \wedge)$*, there exists an equivalent formula φ' in* LTL^{PODB} *such that $|\varphi'| = O(|\varphi|)$.*

PROOF. We show the lemma by induction on the structure of LTL$(\Diamond, \wedge)$ formulas. State predicates are already LTL^{PODB} formulas, and the equivalence $\Diamond p \equiv (\neg p \,\mathcal{U}'\, p)$ trivially holds. Suppose by induction that given the LTL$_+(\Diamond, \wedge)$ formulas $\Diamond \psi_1, \ldots, \Diamond \psi_n$ there exist LTL^{PODB} formulas $\psi'_1, \ldots, \psi'_n$ such that ψ'_i is equivalent to $\Diamond \psi_i$ and ψ'_i has size $O(|\Diamond \psi_i|)$. Consider the formula $\Diamond(p \wedge \Diamond \psi_1 \wedge \ldots \wedge \Diamond \psi_n)$ where p is a state predicate. Clearly, $\psi' = \psi'_1 \wedge \ldots \wedge \psi'_n$ is an LTL^{PODB} formula which is equivalent to $\psi = \Diamond \psi_1 \wedge \ldots \wedge \Diamond \psi_n$ and whose size is $O(|\Diamond \psi_1| + \ldots + |\Diamond \psi_n|) = O(|\psi|)$. Moreover, we observe that the equivalence $\Diamond(p \wedge \psi) \equiv (\neg p \,\mathcal{U}'\, \psi')$ trivially holds. By Proposition 3.7, and since top-level conjunction and disjunction are allowed by the syntax of LTL^{PODB}, we can conclude that each LTL$(\Diamond, \wedge)$ formula φ can be translated into into an equivalent LTL^{PODB} formula of $O(|\varphi|)$ size. □

THEOREM 3.11. LTL^{PODB} *is strictly more expressive than* LTL$(\Diamond, \wedge, \vee)$.

PROOF. By pushing out all the disjunctions, each LTL$(\Diamond, \wedge, \vee)$ formula φ can be transformed into an equivalent LTL$(\Diamond, \wedge)$ formula φ'. Thus by Lemma 3.10 every LTL$(\Diamond, \wedge, \vee)$ formula can be translated into an equivalent LTL^{PODB} formula. To conclude the proof, we observe that it is known that the next modality cannot be

expressed using only until modalities and boolean connectives [Eme90], and thus $\bigcirc p$ cannot be expressed in $\text{LTL}(\Diamond, \wedge, \vee)$. □

Using the construction sketched in the above proof, it is possible to translate a formula φ of $\text{LTL}(\Diamond, \wedge, \vee)$ into an equivalent formula φ' of LTL^{PODB} such that $|\varphi'|$ is exponential in $|\varphi|$. This translation is also optimal, since a sub-exponential translation along with the result from Theorem 3.8 would contradict the lower bound from Theorem 3.6.

4. DETERMINISTIC GENERATORS FOR OTHER LTL FRAGMENTS

In this section we consider some proper fragments of LTL which contains formulas that do not have a generator which is also a PODB. We first give the results on the deterministic generators for these fragments, then compare their expressiveness with LTL^{PODB}. We use the notation $\bigcirc^n$ as a shorthand for n nested next modalities, and therefore, consider size of $\bigcirc^n \varphi$ to be $|\varphi| + n$.

4.1 Generators for $\text{LTL}(\Diamond, \bigcirc, \wedge)$

Let us consider the formula $\varphi = \Diamond(p \wedge \bigcirc^n q)$, where $p, q \in P$. This formula asserts that in an ω-word satisfying φ, there exists a position $i \in \mathbb{N}$ where p is fulfilled and q is fulfilled at position $i+n$. Thus, to decide if $p \wedge \bigcirc^n q$ is satisfied at position i we need to see the next n positions. A way "to delay" decisions of n steps is to keep record of the last n inputs. Therefore, we can have a deterministic generator for φ by augmenting a deterministic generator for $(p \wedge \bigcirc^n q)$ with a record containing the last n inputs. Clearly, the total construction requires $O(2^n)$ states and longest distance. If a formula φ has several distinct subformulas of type $\bigcirc^n p$, it is sufficient to store in the states of a deterministic generator for φ only a record of the last N inputs, where N is the nesting depth of the next modalities of φ. For more complex formulas, we can construct a deterministic generator that stores the last inputs and the subformulas that still need to be satisfied in order to accept the input word. Consider for example a formula $\varphi = \Diamond(p_1 \wedge \bigcirc^{k_1} q_1 \wedge \bigcirc^{k_2} q_2 \wedge \Diamond(p_2 \wedge \bigcirc^{k_3} q_3))$ such that $k_1 \geq k_j$ for $j = 2, 3$. A deterministic generator for φ, A_φ, needs only to store the last k_1 inputs to check the fulfillment of subformulas $\bigcirc^{k_j} q_j$ for $j = 1, 2, 3$. Suppose that w, $|w| \leq k_1$, is the sequence currently stored in the A_φ state and a is the input symbol. If $p_1 \wedge \bigcirc^{k_1} q_1 \wedge \bigcirc^{k_2} q_2$ is fulfilled on wa, A_φ also checks whether $p_2 \wedge \bigcirc^{k_3} q_3$ is fulfilled on wa. If this is the case, then A_φ accepts the current word independently from its suffix, otherwise A_φ enters a state where the fulfillment of $\Diamond(p_2 \wedge \bigcirc^{k_3} q_3)$ still needs to be checked. Clearly, $|A_\varphi|$ is $O\left(2^{|\varphi|\, k_1}\right)$. Since the total size of the formulas stored in a state of A_φ decreases along a run, we have that the longest distance of A_φ is $O(|\varphi|\, 2^{k_1})$.

In the following theorem, we extend this construction to all $\text{LTL}(\Diamond, \bigcirc, \wedge)$ formulas and prove an upper bound on the size and the longest distance of deterministic generators for formulas in this LTL fragment.

THEOREM 4.1. *There exists a deterministic Büchi automaton A accepting all the models of a formula φ in $\text{LTL}(\Diamond, \bigcirc, \wedge)$ such that A has $O\left(2^{k\,|P|+|\varphi|^2}\right)$ size and $O\left(|\varphi|^2\, 2^{k\,|P|}\right)$ longest distance, where k is the nesting depth of next modalities in φ.*

PROOF. We observe that given a formula ψ, the next operators in ψ can distribute over the boolean connectives and the eventually operators 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 $\psi' = O(|\psi|^2)$. Clearly, the nesting depth of next modalities is the same in ψ and ψ'. To complete the proof we only need to show that we can construct a deterministic generator for ψ' of $O\left(2^{k\,|P|+|\psi'|}\right)$ size and $O\left(|\psi'|\,2^{k\,|P|}\right)$ longest distance, where k is the nesting depth of next modalities in ψ'. To prove this claim we sketch the construction of a deterministic generator for formulas $\Diamond\varphi$ from $\text{LTL}_+(\Diamond,\bigcirc,\wedge)$ and then we observe how to extend it to boolean combinations of these formulas. We leave the details of the constructions to the reader.

We start giving some notation. Given a $\Diamond$-formula $\Diamond\varphi$ from $\text{LTL}_+(\Diamond,\bigcirc,\wedge)$, we have that $\Diamond\varphi = \Diamond(p \wedge \bigcirc^{k_1} q_1 \wedge \ldots \wedge \bigcirc^{k_h} q_h \wedge \Diamond\varphi_1 \wedge \ldots \wedge \Diamond\varphi_l)$, where $h \geq 0$, $l \geq 0$, and $p, q_1, \ldots, q_h$ are state predicates. Since we can rewrite each subformula $\Diamond\psi$ of $\Diamond\varphi$ with no nested eventually modalities as $\Diamond(\psi \wedge \Diamond \text{TRUE})$, we can assume that every $\Diamond$-subformula of $\Diamond\varphi$ has either a nested $\Diamond$-subformula or is $\Diamond\,\text{TRUE}$. We inductively define the conjuncts of a subformula as follows. For a formula $\Diamond\varphi = \Diamond(p \wedge \bigcirc^{k_1} q_1 \wedge \ldots \wedge \bigcirc^{k_h} q_h \wedge \Diamond\varphi_1 \wedge \ldots \wedge \Diamond\varphi_l)$, the formula $c = (p \wedge \bigcirc^{k_1} q_1 \wedge \ldots \wedge \bigcirc^{k_h} q_h)$ is defined as the conjunct of $\Diamond\varphi_i$, $i = 1, \ldots, l$, with respect to $\Diamond\varphi$. If $\Diamond\varphi'$ is a $\Diamond$-subformula of $\Diamond\varphi_i$, $i = 1, \ldots, l$, and c' is its conjunct with respect to $\Diamond\varphi_i$, then $c \wedge c'$ is defined as the conjunct of $\Diamond\varphi''$ with respect to $\Diamond\varphi'$. Given a sequence w and a set of formulas Ψ, the set of maximal formulas from Ψ satisfied on w is the set of formulas $\psi \in \Psi$ such that: ψ is satisfied on w, and for every formula $\psi' \in \Psi$ such that ψ is a subformula of ψ', ψ' is not satisfied on w.

Let $\Diamond\varphi$ be a formula from $\text{LTL}_+(\Diamond,\bigcirc,\wedge)$, we construct a deterministic Büchi generator $A_{\Diamond\varphi}$ for $\Diamond\varphi$ as follows. The states of $A_{\Diamond\varphi}$ are pairs given by a set of $\Diamond$-subformulas of $\Diamond\varphi$ and a sequence containing the last k input symbols (*last k digits record*). The starting state of $A_{\Diamond\varphi}$ is $(\{\Diamond\varphi\}, \varepsilon)$ where ε is the empty word. The automaton $A_{\Diamond\varphi}$ from a state (Ψ, ax) and on an input b moves to a state $(\Psi', x'b)$ where:

—$x' = x$, if $|ax| = k$, and $x' = ax$, otherwise;
—Ψ' contains all the $\Diamond$-subformulas φ' of $\Diamond\varphi$ such that φ' is a subformula of $\varphi'' \in \Psi$ and its conjunct with respect to φ'' is maximal among the conjuncts with respect to φ'' that are satisfied on axb.

Clearly, if no conjuncts are satisfied on axb then $\Psi = \Psi'$. The acceptance condition is given by the set $\{(\Psi, w) \mid \Psi = \{\Diamond\,\text{TRUE}\}\}$.

It is easy to verify that the size of A_φ is $O(2^{k\,|P|+|\varphi|})$. Since the size of the set of $\Diamond$ subformulas of φ decreases monotonically along a run, the longest distance of A_φ is $O(|\varphi|\,2^{k\,|P|})$. To complete the proof we observe that the construction for $\neg\varphi$ is dual. Moreover, positive boolean combinations of formulas of type $\Diamond\varphi$ or $\neg\Diamond\varphi$ from $\text{LTL}_+(\Diamond,\bigcirc,\wedge)$ can be handled by a modified cross product construction, where we use as last digits record the largest one among those of the composing automata. Clearly, this construction achieves the claimed upper bounds on the size and the longest distance of the deterministic generator for an $\text{LTL}(\Diamond,\bigcirc,\wedge)$ formula. □

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 stated by the following theorem.

THEOREM 4.2. *There exists a formula φ in* LTL($\Diamond, \bigcirc, \wedge$) *such that all generators of φ have $2^{\Omega(|\varphi|)}$ size and $2^{\Omega(|\varphi|)}$ longest distance.*

PROOF. Since LTL($\Diamond, \wedge$) is a fragment of LTL($\Diamond, \bigcirc, \wedge$), we only need to prove that there exists a formula φ such that all generators for φ have a simple path of length at least 2^n. Consider the formula $\varphi = \Box(p \rightarrow \bigcirc^n q)$, where $p, q \in P$ and $n \geq 2$. Clearly, $|\varphi| = O(n)$. Assume that $A_\varphi = (2^P, S, s_0, \Delta, F)$ is a generator for φ. Consider words $w = a_1 \dots a_n$ and $w' = a'_1 \dots a'_n$ such that $w, w' \in \left(2^P\right)^*$, and for some i, $p \notin a_i$ and $p \in a'_i$. Let $y \in \left(2^P\right)^\omega$ be such that $y = b_1 \dots b_h \dots$, $q \notin b_i$, and xwy is a model of φ for some $x \in \left(2^P\right)^*$. We have that $xw'y$ is not a model of φ since $(p \rightarrow \bigcirc^n q)$ is not fulfilled on $xw'y$ starting at a'_i. 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 there exists 2^n words $w_1, \dots, w_{2^n}$ which are pairwise distinguishable with respect to truth values of p, there are at least 2^n pairwise disjoint sets of states $S_1, \dots, S_{2^n}$ such that A_φ, by reading a prefix of a model for φ ending in w_i, reaches the states in S_i. To conclude this proof we just need to prove that there exists a word on which A_φ reaches in turn a state from each S_i and if a state in S_i has been reached, it does not visit another state in S_i until a state from each of the sets S_j has been reached. By the above arguments this is equivalent to prove that there is an exponentially long word w in $\{0,1\}^*$ such that any two of its subwords of length n differ at least in a position. Since such a word exists (for example, it can be determined starting from $w = 0^n$ and iteratively repeating the following step: append 1 to w, if the suffix of $w1$ of length n is different from any subword of length n of w, and append 0, otherwise), we are done. □

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

The fragment LTL($\Diamond, \bigcirc, \wedge, \vee$) contains boolean combinations of formulas built from state predicates using eventualities, next modalities, disjunctions, and conjunctions. This fragment includes combinations of safety and guarantee properties, and belongs to the class of syntactic obligation properties [MP91].

In Section 3.2 we have considered the LTL fragment LTL($\Diamond, \wedge, \vee$) and proved that each formula in this fragment has a deterministic generator of doubly exponential size and exponential longest distance. Indeed this result holds also for LTL($\bigcirc, \Diamond, \wedge, \vee$) which strictly subsumes LTL($\Diamond, \wedge, \vee$), as stated by the following theorem.

THEOREM 4.3. *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. We first observe that as for LTL($\Diamond, \wedge, \vee$) formulas, we can translate an LTL($\Diamond, \bigcirc, \wedge, \vee$) formula φ into an equivalent LTL($\Diamond, \bigcirc, \wedge$) formula φ' with at most an exponential blow-up of the size. By Theorem 4.1 there exists a deterministic Büchi automaton A accepting all the models of φ' such that A has $O\left(2^{k\,|P|+|\varphi'|^2}\right)$

size and $O\left(|\varphi'|^2\, 2^{k\,|P|}\right)$ longest distance, where k is the nesting depth of next modalities in $|\varphi'|$. Since $k = O(|\varphi|)$ and $|\varphi'| = O(2^{|\varphi|})$, we have that A is a generator for φ of size doubly exponential in $|\varphi|$ and longest distance exponential in $|\varphi|$. □

Since LTL$(\Diamond, \wedge, \vee)$ is a proper fragment of LTL$(\Diamond, \bigcirc, \wedge, \vee)$, by Theorem 3.6 we may not have a smaller deterministic generator for some formula in LTL$(\Diamond, \bigcirc, \wedge, \vee)$.

4.3 Generators for LTL$(\Box, \Diamond)$

In Section 2.3 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 4.4. *There exists a formula φ in* LTL$(\Box, \Diamond)$ *such that all the deterministic generators of φ have $2^{2^{\Omega(|\varphi|)}}$ 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 $\{x_1, \ldots, x_n\}$, for $x \in \{a, b, c, d\}$, by p_j a subset of P_a, and by q_j a subset of P_c. By arguments similar to those used to prove Theorem 3.6, it is possible to show that: 1) a deterministic generator for φ needs 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 $k = 2^{\Omega(n)}$, let $p_1, \ldots, p_k$ and $q_1, \ldots, q_k$ be such sets, and denote by P_p the set $\{p_1, \ldots, p_k\}$, and by P_q the set $\{q_1, \ldots, q_k\}$. We observe that exactly 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 replaces the list for p_j with the only q_h which is true at that position. Every time a q_j is true, A adds it to all lists. To conclude the proof it is sufficient to show that there exists a word w in $(P_p \cup P_q)^*$ of length 2^k such that the only run r of A on w satisfies $r(i) \neq r(j)$ 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 of s. 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, then 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. As either a p_i or a q_j is true at each position, the tuples of two consecutive states in a run may differ in at most a bit. We observe that it is possible to list all the tuples of k bits in such a way that each tuple appears exactly once and any two consecutive tuples differ exactly on a bit. A way to generate such a sequence is the following. Let σ be a sequence of k tuples, we denote by σ^R the reverse sequence, and by $a\sigma$ the sequence of $k+1$ tuples obtained from σ by expanding each k tuple t to a $k+1$ tuple having a as the leftmost bit and t as the remaining bits. We determine a sequence of $k+1$ tuples by a given sequence of k tuples σ, by appending 1σ to $0\sigma^R$. If we start this process from the sequence 01 ($k = 1$), it is easy to verify by induction that for any $k \geq 1$ the obtained sequence of k tuples lists all the k tuples over $\{0, 1\}$ and satisfies the above property. Thus we have proved that the longest distance of any deterministic

generator for φ is at least $2^k = 2^{2^{\Omega(n)}}$. □

4.4 Expressiveness

In the following theorem we compare the expressiveness of the logics considered in this section with LTL^{PODB}.

THEOREM 4.5. *The expressiveness of* LTL^{PODB} *is not comparable to the expressiveness of either* $\text{LTL}(\Box, \Diamond)$ *or* $\text{LTL}(\Diamond, \bigcirc, \wedge)$.

PROOF. Consider again the $\text{LTL}(\Diamond, \bigcirc, \wedge)$ formula $\varphi = \Diamond(p \wedge \bigcirc^n q)$, where $p, q \in P$ and $n \geq 2$. We claim that any deterministic generator for φ has a cycle which is not a self-loop, and thus by Theorem 3.8, there is no LTL^{PODB} formula which is equivalent to φ. We recall that a model of φ is any ω-word such that there exists a position i with the property that p is true at i and q is true at $i+k$. Any generator for φ must have cycles since a position at which the above property is satisfied can be any $i \in \mathbb{N}$. Thus we only need to show that a cycle of any deterministic generator for φ is not a self-loop. To prove this, consider for $i = 0, \ldots, n-1$ the sequences $x_i = p\,(\neg p)^i$ and $y_i = (\neg p)^{n-i-1}\, q$. For $h \in \mathbb{N}$ and $i = 0, \ldots, n-1$ denote by $w_{h,i}$ the word $x_{n-1}^h x_i$. Clearly, $w_{h,i} y_j \emptyset \ldots \emptyset \ldots$ is a model of φ if and only if $i = j$. This implies that, for any $h, h' \in \mathbb{N}$, the state entered by A after reading $w_{h,i}$ must be different from that entered after reading $w_{h',j}$ if $i \neq j$. Thus A does not enter self-loops reading x_{n-1}^ω, and the above claim is proved.

To prove that LTL^{PODB} is not as expressive as $\text{LTL}(\Box, \Diamond)$, consider the $\text{LTL}(\Box, \Diamond)$ formula $\varphi' = \Box(\bigwedge_{i=1}^n \Diamond p_i)$, where the set of atomic propositions is $P = \{p_1, \ldots, p_n\}$. Let $P_i = P \setminus \{p_i\}$, and A' be a deterministic generator for φ'. It is possible to prove, using arguments analogous to those used for the above claim, that the run of A' on the word x^ω, where $x = P_1 \ldots P_n$, is a sequence of states where two consecutive states cannot be the same. Thus, since A' has a finite number of states, A' must have a cycle and that there is no PODB generating models of φ'.

To conclude the proof we claim that for a formula $(p\, \mathcal{U}'\, q)$ there is not a formula in either $\text{LTL}(\Diamond, \bigcirc, \wedge)$ or $\text{LTL}(\Box, \Diamond)$ which is equivalent to it [Eme90]. □

Since $\text{LTL}(\Diamond, \bigcirc, \wedge)$ and $\text{LTL}(\Diamond, \bigcirc, \wedge, \vee)$ are equivalently expressive, the above result extends to $\text{LTL}(\Diamond, \bigcirc, \wedge, \vee)$.

5. 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 $O\left(\frac{n}{\log n}\right)$. Standard techniques to solve Büchi games involve fix-point computation [Tho95], and require 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, 4.1 and 4.2. In the second part of this section, we combine this algorithm with the results on LTL generators from the previous sections and study the complexity of the obtained solutions.

We consider games as defined in Section 2.4. Let (G, F) be a Büchi game. Given a play $x_0 \ldots x_n$ of (G, F) we say that it *ends in a loop* if $x_i \neq x_j$ for any $i \neq j$,

$0 \leq i, j < n$, and there exists an $0 \leq i < n$ such that $x_i = x_n$. Moreover, given a play $x_0 \dots x_n$ which ends in a loop, and let i be such that $0 \leq i < n$ and $x_i = x_n$, we say that $x_0 \dots x_n$ is *winning* if $x_h \in F$ for some $h \in [i, n]$. We define a game $(G, F)_{fin}$ as the game where $Player_0$ wins from a vertex u if there exists a strategy f in (G, F) starting from u such that any play, constructed according to f and which ends in a loop, is also a winning play.

The following lemma holds.

LEMMA 5.1. *There exists a winning strategy from a vertex u in a Büchi game (G, F) if and only if there exists a winning strategy from u in $(G, F)_{fin}$.*

PROOF. Consider first the forward direction. We recall that Büchi games admit a memoryless solution, and clearly any memoryless winning strategy f in (G, F) is also a winning strategy in $(G, F)_{fin}$. To see this it is sufficient to observe that all plays, constructed according to f and ending in a loop, must be also winning, otherwise there must be a path of the ω-tree corresponding to f on which all states of F does not repeat infinitely often, and thus f is not a winning strategy. Consider now the converse direction. Assume that there exists a winning strategy f from u in $(G, F)_{fin}$, let L be the set of winning plays from u constructed according to f, and define a graph G' such that the set of vertices $pre(L)$ is the set of all proper prefixes of L, and there is an edge from w to w' if and only if: for $x \in V$ either $w' = wx$ or $wx \in L$ and $w' = w''x$ is a prefix of w. We define a new strategy f' that corresponds to the unwinding of this graph in an obvious way (we omit the tedious details of a formal definition). Since f' coincides with f on all the plays ending in a loop, we get that f' is also a winning strategy from u in $(G, F)_{fin}$. To complete the proof we claim that f' is indeed also a winning strategy from u in the Büchi game (G, F). To see this assume by contradiction that there is a path of the tree corresponding to f' where none of the states in F repeats infinitely often. Observe that any of such paths is obtained by unwinding cycles of G corresponding to cycles of G'. Thus there must be a cycle of G' which corresponds to a cycle of G without a state in F, this is a contradiction since f' is a winning strategy of $(G, F)_{fin}$ and is defined by G'. □

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

LEMMA 5.2. *Any winning strategy f in a game $(G, F)_{fin}$ is such that the length 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 5.3. *Given a game graph G with m vertices and longest distance d, the Büchi game (G, F) is decidable in space $O(d \log m)$.*

Consider a Σ-labeled game graph G with labeling function μ and a predicate W over words of set of atomic propositions contained in Σ. If the winning condition W can be translated into a deterministic Büchi automaton, it is possible to use the algorithm derived by Lemmas 5.1 and 5.2 to decide it. In particular, let A be a deterministic Büchi automaton equivalent to the winning condition W, in the

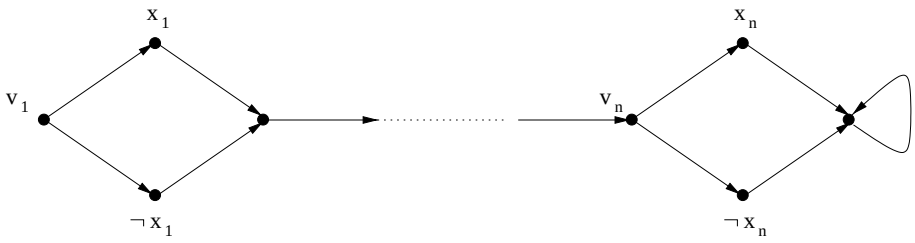

Fig. 4. Reduction from QBF satisfiability: the game graph.

sense that the language accepted by A is the language of the ω-words satisfying W. Define $G \times A$ as the game graph given by:

—the set of vertices $V \times Q$, where Q is the set of A states;

—the partition of $V \times Q$ reflecting the partition of V;

—the successor relation defined as: a vertex (v', q') is a successor of a vertex (v, q) if and only if v' is a successor of v in G and there is a transition of A from q to q' on $\mu(v)$.

Let F and s_0 be respectively the set of final states and the initial state of A, 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 4 and the above construction, Theorem 5.3 applies to games with winning condition expressed by formulas in the LTL fragments which we have considered so far. The following theorems hold.

THEOREM 5.4. *Deciding* LTL$(\Diamond, \wedge)$ *games is* PSPACE*-complete.*

PROOF. Membership in PSPACE is a consequence of Theorems 3.3 and 5.3. 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. Consider a quantified boolean formula, over the variables $x_1, \ldots, x_n$, $\varphi = A_1 x_1 \ldots A_n x_n . \bigwedge_{i=1}^{m} c_i$. Let $\varphi' = \bigwedge_{i=1}^{m} \Diamond d_i$ be the LTL$(\Diamond, \wedge)$ formula over the atomic propositions $\{d_1, \ldots, d_m\}$ such that d_i corresponds to the clause c_i. The game graph G (see Figure 4) has a vertex for each literal, a vertex for each quantifier and an extra vertex. We denote by the corresponding literal any vertex in the first set, and by v_i the vertex corresponding to A_i for $i = 1, \ldots, n$. The extra vertex has, except for a self-loop, no exiting transitions. In our reduction, the vertices corresponding to literals along with the extra vertex can be either $Player_0$ or $Player_1$ vertices since from them there is only one possible move. Vertex v_i is a $Player_0$ vertex, if A_i is the existential quantifier, and a $Player_1$ vertex, otherwise. For a literal l, we label the corresponding vertex by any d_i such that l is a literal of the corresponding clause c_i. In all the other vertices all the atomic propositions are false. Thus a path in the game tree corresponds in a natural way to an assignment of the variables in φ and a strategy in the game (G, φ') corresponds to a selection of paths fulfilling the requirements of quantifiers $A_1, \ldots, A_n$. Hence, we have that φ is satisfiable if and only if there is a winning strategy in the game (G, φ'). □

THEOREM 5.5. *Deciding* LTL^{PODB} *games is* PSPACE*-complete.*

PROOF. Membership in PSPACE is a consequence of Theorems 3.8 and 5.3. To prove PSPACE-hardness, we observe that by Lemma 3.10 every LTL($\Diamond, \wedge$) formula φ can be translated into an equivalent LTL^{PODB} formula of $O(|\varphi|)$ size. Thus we can reduce in linear time LTL($\Diamond, \wedge$) games to LTL^{PODB} games, and hence PSPACE-hardness follows from Theorem 5.4. □

THEOREM 5.6. *Deciding* LTL($\Diamond, \bigcirc, \wedge$) *games is* EXPTIME*-complete.*

PROOF. By Theorem 4.1, 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 eventually and next modalities along with conjunctions. For instance, a mistake in the encoding of the content of i-th tape position is described by the formula

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

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

THEOREM 5.7. *Deciding* LTL($\Diamond, \bigcirc, \wedge, \vee$) *games is* EXPSPACE.

PROOF. Directly from Theorems 4.3 and 5.3. □

	Nondet. Generators		Det. Generators	
	Size	Long. Distance	Size	Long. Distance
LTL($\Diamond, \wedge$)	Θ(EXP)	Θ(LINEAR)	Θ(EXP)	Θ(LINEAR)
LTL($\Diamond, \bigcirc, \wedge$)	Θ(EXP)	Θ(EXP)	Θ(EXP)	Θ(EXP)
LTL($\Diamond, \wedge, \vee$)	Θ(EXP)	Θ(LINEAR)	Θ(2EXP)	Θ(EXP)
LTL($\Diamond, \bigcirc, \wedge, \vee$)	Θ(EXP)	Θ(EXP)	Θ(2EXP)	Θ(EXP)
LTL($\Box, \Diamond$)	Θ(EXP)	Θ(LINEAR)	Θ(2EXP)	Θ(2EXP)
LTL	Θ(EXP)	Θ(EXP)	Θ(2EXP)	Θ(2EXP)

Fig. 5. Complexity of generators for LTL fragments.

	Model-checking	Games
LTL($\Diamond, \wedge$)	NP-complete	PSPACE-complete
LTL($\Diamond, \bigcirc, \wedge$)	PSPACE-complete	EXPTIME-complete
LTL($\Diamond, \wedge, \vee$)	NP-complete	EXPSPACE
LTL($\Diamond, \bigcirc, \wedge, \vee$)	PSPACE-complete	EXPSPACE-complete
LTL($\Box, \Diamond$)	NP-complete	2EXPTIME
LTL	PSPACE-complete	2EXPTIME-complete

Fig. 6. Complexity of model-checking and games in LTL fragments.

6. 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 tables of Figures 5 and 6 for various fragments. 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($\Diamond, \wedge, \vee$) and LTL($\Box, \Diamond$) 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($\Box, \Diamond$), 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

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

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.

R. Alur, T. Henzinger, F. Mang, S. Qadeer, S. Rajamani, and S. Tasiran. MOCHA: Modularity in model checking. In *Proc. of the 10th International Conference on Computer Aided Verification*, LNCS 1427, pages 521 – 525. Springer-Verlag, 1998.

M. Abadi, L. Lamport, and P. Wolper. Realizable and unrealizable specifications of reactive systems. In *Proc. of the 16th Intern. Colloquium on Automata, Languages and Programming, ICALP'89*, LNCS 372, pages 1–17, 1989.

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

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.

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.

E.A. Emerson. Temporal and modal logic. In J. van Leeuwen, editor, *Handbook of Theoretical Computer Science*, volume B, pages 995 – 1072. Elsevier Science Publishers, 1990.

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

O. Kupferman and M.Y. Vardi. Module checking revisited. In *Proc. of the 9th Intern. Conference on Computer Aided Verification, CAV'97*, LNCS 1254, pages 36 –47, June 1997.

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, June 1998.

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.

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

J. Marcinkowski and T. Truderung. Optimal complexity bounds for positive LTL games. In *Proc. of the 16th Annual Conference of the European Association for Computer Science Logic, CSL'02*. Springer-verlag, 2002.

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

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.

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

R. Rosner. *Modular Synthesis of Reactive Systems*. PhD thesis, Weizmann Institute of Science, 1992.

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

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

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

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

Received: September 2001; revised: June 2002; accepted: June 2002.