Journal of Artificial Intelligence Research 30 (2007) 101-132 Submitted 10/05; published 9/07

The Planning Spectrum — One, Two, Three, Infinity

Marco Pistore PISTORE@DIT.UNITN.IT
Department of Information and Communication Technology
University of Trento
Via Sommarive 14, 38050 Povo (Trento), Italy

Moshe Y. Vardi VARDI@CS.RICE.EDU
Department of Computer Science
Rice University
6100 S. Main Street, Houston, Texas

Abstract

Linear Temporal Logic (LTL) is widely used for defining conditions on the execution paths of dynamic systems. In the case of dynamic systems that allow for nondeterministic evolutions, one has to specify, along with an LTL formula φ, which are the paths that are required to satisfy the formula. Two extreme cases are the *universal* interpretation $\mathcal{A}.\varphi$, which requires that the formula be satisfied for all execution paths, and the *existential* interpretation $\mathcal{E}.\varphi$, which requires that the formula be satisfied for some execution path.

When LTL is applied to the definition of goals in planning problems on nondeterministic domains, these two extreme cases are too restrictive. It is often impossible to develop plans that achieve the goal in all the nondeterministic evolutions of a system, and it is too weak to require that the goal is satisfied by some execution.

In this paper we explore alternative interpretations of an LTL formula that are between these extreme cases. We define a new language that permits an arbitrary combination of the $\mathcal{A}$ and $\mathcal{E}$ quantifiers, thus allowing, for instance, to require that each finite execution can be extended to an execution satisfying an LTL formula ($\mathcal{AE}.\varphi$), or that there is some finite execution whose extensions all satisfy an LTL formula ($\mathcal{EA}.\varphi$). We show that only eight of these combinations of path quantifiers are relevant, corresponding to an alternation of the quantifiers of length one ($\mathcal{A}$ and $\mathcal{E}$), two ($\mathcal{AE}$ and $\mathcal{EA}$), three ($\mathcal{AEA}$ and $\mathcal{EAE}$), and infinity ($(\mathcal{AE})^\omega$ and $(\mathcal{EA})^\omega$). We also present a planning algorithm for the new language that is based on an automata-theoretic approach, and study its complexity.

1. Introduction

In automated task planning (Fikes & Nilsson, 1971; Penberthy & Weld, 1992; Ghallab, Nau, & Traverso, 2004), given a description of a dynamic domain and of the basic actions that can be performed on it, and given a goal that defines a success condition to be achieved, one has to find a suitable plan, that is, a description of the actions to be executed on the domain in order to achieve the goal. "Classical" planning concentrates on the so called "reachability" goals, that is, on goals that define a set of final desired states to be reached. Quite often practical applications require plans that deal with goals that are more general than sets of final states. Several planning approaches have been recently proposed, where *temporal logic* formulas are used as goal language, thus allowing for goals that define conditions on the whole plan execution paths, i.e., on the sequences of states resulting from the execution of plans (Bacchus & Kabanza, 1998, 2000; Calvanese, de Giacomo, & Vardi, 2002; Cerrito &

Mayer, 1998; Dal Lago, Pistore, & Traverso, 2002; de Giacomo & Vardi, 1999; Kvarnström & Doherty, 2001; Pistore & Traverso, 2001). Most of these approaches use *Linear Temporal Logic* (LTL) (Emerson, 1990) as the goal language. LTL allows one to express reachability goals (e.g., $\mathrm{F}\, q$ — reach q), maintainability goals (e.g., $\mathrm{G}\, q$ — maintain q), as well as goals that combine reachability and maintainability requirements (e.g., $\mathrm{F}\,\mathrm{G}\, q$ — reach a set of states where q can be maintained), and Boolean combinations of these goals.

In *planning in nondeterministic domains* (Cimatti, Pistore, Roveri, & Traverso, 2003; Peot & Smith, 1992; Warren, 1976), actions are allowed to have different outcomes, and it is not possible to know at planning time which of the different possible outcomes will actually take place. Nondeterminism in action outcome is necessary for modeling in a realistic way several practical domains, ranging from robotics to autonomous controllers to two-player games.[1] For instance, in a realistic robotic application one has to take into account that actions like "pick up object" might result in a failure (e.g., if the object slips out of the robot's hand). A consequence of nondeterminism is that the execution of a plan may lead to more than one possible execution path. Therefore, one has to distinguish whether a given goal has to be satisfied by all the possible execution paths (in this case we speak of "strong" planning), or only by some of the possible execution paths ("weak" planning). In the case of an LTL goal φ, strong planning corresponds to interpreting the formula in a universal way, as $\mathcal{A}.\varphi$, while weak planning corresponds to interpreting it in an existential way, as $\mathcal{E}.\varphi$.

Weak and strong plans are two extreme ways of satisfying an LTL formula. In nondeterministic planning domains, it might be impossible to achieve goals in a strong way: for instance, in the robotic application it might be impossible to fulfill a given task if objects keep slipping from the robot's hand. On the other hand, weak plans are too unreliable, since they achieve the goal only under overly optimistic assumptions on the outcomes of action executions.

In the case of reachability goals, *strong cyclic planning* (Cimatti et al., 2003; Daniele, Traverso, & Vardi, 1999) has been shown to provide a viable compromise between weak and strong planning. Formally, a plan is strong cyclic if each possible partial execution of the plan can always be extended to an execution that reaches some goal state. Strong cyclic planning allows for plans that encode iterative trial-and-error strategies, like "pick up an object until succeed". The execution of such strategies may loop forever only in the case the action "pick up object" continuously fails, and a failure in achieving the goal for such an unfair execution is usually acceptable. Branching-time logics like CTL and CTL* allow for expressing goals that take into account nondeterminism. Indeed, Daniele et al. (1999) show how to encode strong cyclic reachability goals as CTL formulas. However, in CTL and CTL* path quantifiers are interleaved with temporal operators, making it difficult to extend the encoding of strong cyclic planning proposed by Daniele et al. (1999) to generic temporal goals.

In this paper we define a new logic that allows for exploring the different degrees in which an LTL formula φ can be satisfied that exist between the strong goal $\mathcal{A}.\varphi$ and the weak goal $\mathcal{E}.\varphi$. We consider logic formulas of the form $\alpha.\varphi$, where φ is an LTL formula and α is a path quantifier that generalizes the $\mathcal{A}$ and $\mathcal{E}$ quantifiers used for strong and weak planning.

1. See the work of Ghallab et al. (2004) for a deeper discussion on the fundamental role of nondeterminism in planning problems and in practical applications.

A path quantifier is a (finite or infinite) word on alphabet $\{\mathcal{A}, \mathcal{E}\}$. The path quantifier can be seen as the definition of a two-player game for the selection of the outcome of action execution. Player A (corresponding to symbol $\mathcal{A}$) chooses the action outcomes in order to make goal φ fail, while player E (corresponding to symbol $\mathcal{E}$) chooses the action outcomes in order to satisfy the goal φ. At each turn, the active player controls the outcome of action execution for a finite number of actions and then passes the control to the other player.[2] We say that a plan satisfies the goal $\alpha.\varphi$ if the player E has a winning strategy, namely if, for all the possible moves of player A, player E is always able to build an execution path that satisfies the LTL formula φ.

Different path quantifiers define different alternations in the turns of players A and E. For instance, with goal $\mathcal{A}.\varphi$ we require that the formula φ is satisfied independently of how the "hostile" player A chooses the outcomes of actions, that is, we ask for a strong plan. With goal $\mathcal{E}.\varphi$ we require that the formula φ is satisfied for some action outcomes chosen by the "friendly" player E, that is, we ask for a weak plan. With goal $\mathcal{AE}.\varphi$ we require that every plan execution led by player A can be extended by player E to a successful execution that satisfies the formula φ; in the case of a reachability goal, this corresponds to asking for a strong cyclic solution. With goal $\mathcal{EA}.\varphi$ we require that, after an initial set of actions controlled by player E, we have the guarantee that formula φ will be satisfied independently of how player A will choose the outcome of the following actions. As a final example, with goal $(\mathcal{AE})^\omega.\varphi = \mathcal{AEAEA}\cdots.\varphi$ we require that formula φ is satisfied in all those executions where player E has the possibility of controlling the action outcome an infinite number of times.

Path quantifiers can define arbitrary combinations of the turns of players A and E, and hence different degrees in satisfying an LTL goal. We show, however, that, rather surprisingly, only a finite number of alternatives exist between strong and weak planning: only eight "canonical" path quantifiers give rise to plans of different strength, and every other path quantifier is equivalent to a canonical one. The canonical path quantifiers correspond to the games of length one ($\mathcal{A}$ and $\mathcal{E}$), two ($\mathcal{AE}$ and $\mathcal{EA}$), and three ($\mathcal{AEA}$ and $\mathcal{EAE}$), and to the games defining an infinite alternation between players A and E ($(\mathcal{AE})^\omega$ and $(\mathcal{EA})^\omega$). We also show that, in the case of reachability goals $\varphi = \mathrm{F}\,q$, the canonical path quantifiers further collapse. Only three different degrees of solution are possible, corresponding to weak ($\mathcal{E}.\,\mathrm{F}\,q$), strong ($\mathcal{A}.\,\mathrm{F}\,q$), and strong cyclic ($\mathcal{AE}.\,\mathrm{F}\,q$) planning.

Finally, we present a planning algorithm for the new goal language and we study its complexity. The algorithm is based on an automata-theoretic approach (Emerson & Jutla, 1988; Kupferman, Vardi, & Wolper, 2000): planning domains and goals are represented as suitable automata, and planning is reduced to the problem of checking whether a given automaton is nonempty. The proposed algorithm has a time complexity that is doubly exponential in the size of the goal formula. It is known that the planning problem is 2EXPTIME-complete for goals of the form $\mathcal{A}.\varphi$ (Pnueli & Rosner, 1990), and hence the complexity of our algorithm is optimal.

The structure of the paper is as follows. In Section 2 we present some preliminaries on automata theory and on temporal logics. In Section 3 we define planning domains and plans. In Section 4 we define $\mathcal{AE}$-LTL, our new logic of path quantifier, and study its basic

2. If the path quantifier is a finite word, the player that has the last turn chooses the action outcome for the rest of the infinite execution.

properties. In Section 5 we present a planning algorithm for $\mathcal{AE}$-LTL, while in Section 6 we apply the new logic to the particular cases of reachability and maintainability goals. In Section 7 we make comparisons with related works and present some concluding remarks.

2. Preliminaries

This section introduces some preliminaries on automata theory and on temporal logics.

2.1 Automata Theory

Given a nonempty alphabet Σ, an infinite word on Σ is an infinite sequence $\sigma_0, \sigma_1, \sigma_2, \ldots$ of symbols from Σ. Finite state automata have been proposed as finite structures that accept sets of infinite words. In this paper, we are interested in *tree* automata, namely in finite state automata that recognize trees on alphabet Σ, rather than words.

Definition 1 (tree) *A* (leafless) tree τ *is a subset of* $\mathbb{N}^*$ *such that:*

- $\epsilon \in \tau$ *is the root of the tree;*
- *if* $x \in \tau$ *then there is some* $i \in \mathbb{N}$ *such that* $x \cdot i \in \tau$*;*
- *if* $x \cdot i \in \tau$*, with* $x \in \mathbb{N}^*$ *and* $i \in \mathbb{N}$*, then also* $x \in \tau$*;*
- *if* $x \cdot (i{+}1) \in \tau$*, with* $x \in \mathbb{N}^*$ *and* $i \in \mathbb{N}$*, then also* $x \cdot i \in \tau$*.*

The arity *of* $x \in \tau$ *is the number of its children, namely* $arity(x) = |\{i : x \cdot i \in \tau\}|$*. Let* $\mathcal{D} \subseteq \mathbb{N}$*. Tree* τ *is a* $\mathcal{D}$-tree *if* $arity(x) \in \mathcal{D}$ *for each* $x \in \tau$*. A* Σ-labelled tree *is a pair* $(\tau, \mathcal{T})$*, where* τ *is a tree and* $\mathcal{T} : \tau \to \Sigma$*. In the following, we will denote* Σ*-labelled tree* $(\tau, \mathcal{T})$ *as* $\mathcal{T}$*, and let* $\tau = \mathrm{dom}(\mathcal{T})$*.*

Let $\mathcal{T}$ be a Σ-labelled tree. A *path* p of $\mathcal{T}$ is a (possibly infinite) sequence $x_0, x_1, \ldots$ of nodes $x_i \in \mathrm{dom}(\mathcal{T})$ such that $x_{k+1} = x_k \cdot i_{k+1}$. In the following, we denote with $P^*(\mathcal{T})$ the set of finite paths and with $P^\omega(\mathcal{T})$ the set of infinite paths of $\mathcal{T}$. Given a (finite or infinite) path p, we denote with $\mathcal{T}(p)$ the string $\mathcal{T}(x_0) \cdot \mathcal{T}(x_1) \cdots$, where $x_0, x_1, \ldots$ is the sequence of nodes of path p. We say that a finite (resp. infinite) path p' is a finite (resp. infinite) *extension* of the finite path p if the sequence of nodes of p is a prefix of the sequence of nodes of p'.

A tree automaton is an automaton that accepts sets of trees. In this paper, we consider a particular family of tree automata, namely *parity tree automata* (Emerson & Jutla, 1991).

Definition 2 (parity tree automata) *A* parity tree automaton *with parity index* k *is a tuple* $A = \langle \Sigma, \mathcal{D}, Q, q_0, \delta, \beta \rangle$*, where:*

- Σ *is the finite, nonempty alphabet;*
- $\mathcal{D} \subseteq \mathbb{N}$ *is a finite set of arities;*
- Q *is the finite set of states;*
- $q_0 \in Q$ *is the initial state;*

- $\delta : Q \times \Sigma \times \mathcal{D} \to 2^{Q^*}$ *is the transition function, where* $\delta(q, \sigma, d) \in 2^{Q^d}$*;*
- $\beta : Q \to \{0, \ldots, k\}$ *is the parity mapping.*

A tree automaton accepts a tree if there is an accepting run of the automaton on the tree. Intuitively, when a parity tree automaton is in state q and it is reading a d-ary node of the tree that is labeled by σ, it nondeterministically chooses a d-tuple $\langle q_1, \ldots, q_d \rangle$ in $\delta(q, \sigma, d)$ and then makes d copies of itself, one for each child node of the tree, with the state of the i-th copy updated to q_i. A run of the parity tree automaton is accepting if, along every infinite path, the minimal priority that is visited infinitely often is an even number.

Definition 3 (tree acceptance) *The parity tree automaton* $A = \langle \Sigma, \mathcal{D}, Q, q_0, \delta, \beta \rangle$ accepts *the* Σ*-labelled* $\mathcal{D}$*-tree* $\mathcal{T}$ *if there exists an* accepting run r *for* $\mathcal{T}$*, namely there exists a mapping* $r : \tau \to Q$ *such that:*

- $r(\epsilon) = q_0$*;*
- *for each* $x \in \tau$ *with arity*$(x) = d$ *we have* $\langle r(x \cdot 0), \ldots r(x \cdot (d{-}1)) \rangle \in \delta(r(x), \mathcal{T}(x), d)$*;*
- *along every infinite path* $x_0, x_1, \ldots$ *in* $\mathcal{T}$*, the minimal integer* h *such that* $\beta(r(x_i)) = h$ *for infinitely many nodes* x_i *is even.*

The tree automaton A *is* nonempty *if there exists some tree* $\mathcal{T}$ *that is accepted by* A*.*

Emerson and Jutla (1991) have shown that the emptiness of a parity tree automaton can be decided in a time that is exponential in the parity index and polynomial in the number of states.

Theorem 1 *The emptiness of a parity tree automaton with* n *states and index* k *can be determined in time* $n^{O(k)}$*.*

2.2 Temporal Logics

Formulas of *Linear Temporal Logic* (LTL) (Emerson, 1990) are built on top of a set *Prop* of atomic propositions using the standard Boolean operators, the unary temporal operator X (next), and the binary temporal operator U (until). In the following we assume to have a fixed set of atomic propositions *Prop*, and we define $\Sigma = 2^{Prop}$ as the set of subsets of *Prop*.

Definition 4 (LTL) *LTL formulas* φ *on Prop are defined by the following grammar, where* $q \in$ *Prop:*

$$\varphi ::= q \mid \neg\varphi \mid \varphi \wedge \varphi \mid \mathrm{X}\,\varphi \mid \varphi\,\mathrm{U}\,\varphi$$

We define the following auxiliary operators: $\mathrm{F}\,\varphi = \top\,\mathrm{U}\,\varphi$ (eventually in the future φ) and $\mathrm{G}\,\varphi = \neg\,\mathrm{F}\,\neg\varphi$ (always in the future φ). LTL formulas are interpreted over infinite words on Σ. In the following, we write $w \models_{\mathrm{LTL}} \varphi$ whenever the infinite word w satisfies the LTL formula φ.

Definition 5 (LTL semantics) *Let* $w = \sigma_0, \sigma_1, \ldots$ *be an infinite word on* Σ *and let* φ *be an LTL formula. We define* $w, i \models_{LTL} \varphi$*, with* $i \in \mathbb{N}$*, as follows:*

- $w, i \models_{LTL} q$ *iff* $q \in \sigma_i$*;*
- $w, i \models_{LTL} \neg\varphi$ *iff it does not hold that* $w, i \models_{LTL} \varphi$*;*
- $w, i \models_{LTL} \varphi \wedge \varphi'$ *iff* $w, i \models_{LTL} \varphi$ *and* $w, i \models_{LTL} \varphi'$*;*
- $w, i \models_{LTL} \mathrm{X}\,\varphi$ *iff* $w, i{+}1 \models_{LTL} \varphi$*;*
- $w, i \models_{LTL} \varphi\,\mathrm{U}\,\varphi'$ *iff there is some* $j \geq i$ *such that* $w, k \models_{LTL} \varphi$ *for all* $i \leq k < j$ *and* $w, j \models_{LTL} \varphi'$*.*

We say that w *satisfies* φ*, written* $w \models_{LTL} \varphi$*, if* $w, 0 \models_{LTL} \varphi$*.*

CTL* (Emerson, 1990) is an example of "branching-time" logic. Path quantifiers A ("for all paths") and E ("for some path") can prefix arbitrary combinations of linear time operators.

Definition 6 (CTL*) *CTL* formulas* ψ *on Prop are defined by the following grammar, where* $q \in$ *Prop:*

$$\begin{array}{lcl} \psi & ::= & q \mid \neg\psi \mid \psi \wedge \psi \mid \mathrm{A}\,\varphi \mid \mathrm{E}\,\varphi \\ \varphi & ::= & \psi \mid \neg\varphi \mid \varphi \wedge \varphi \mid \mathrm{X}\,\varphi \mid \varphi\,\mathrm{U}\,\varphi \end{array}$$

CTL* formulas are interpreted over Σ-labelled trees. In the following, we write $\mathcal{T} \models_{\mathrm{CTL}^*} \psi$ whenever $\mathcal{T}$ satisfies the CTL* formula ψ.

Definition 7 (CTL* semantics) *Let* $\mathcal{T}$ *be a* Σ*-labelled tree and let* ψ *be a CTL* formula. We define* $\mathcal{T}, x \models_{CTL^*} \psi$*, with* $x \in \tau$*, as follows:*

- $\mathcal{T}, x \models_{CTL^*} q$ *iff* $q \in \mathcal{T}(x)$*;*
- $\mathcal{T}, x \models_{CTL^*} \neg\psi$ *iff it does not hold that* $\mathcal{T}, x \models_{CTL^*} \psi$*;*
- $\mathcal{T}, x \models_{CTL^*} \psi \wedge \psi'$ *iff* $\mathcal{T}, x \models_{CTL^*} \psi$ *and* $\mathcal{T}, x \models_{CTL^*} \psi'$*;*
- $\mathcal{T}, x \models_{CTL^*} \mathrm{A}\,\varphi$ *iff* $\mathcal{T}, p \models_{CTL^*} \varphi$ *holds for all infinite paths* $p = x_0, x_1, \ldots$ *with* $x_0 = x$*;*
- $\mathcal{T}, x \models_{CTL^*} \mathrm{E}\,\varphi$ *iff* $\mathcal{T}, p \models_{CTL^*} \varphi$ *holds for some infinite path* $p = x_0, x_1, \ldots$ *with* $x_0 = x$*;*

where $\mathcal{T}, p \models_{CTL^*} \phi$*, with* $p \in P^\omega(\mathcal{T})$*, is defined as follows:*

- $\mathcal{T}, p \models_{CTL^*} \psi$ *iff* $p = x_0, x_1, \ldots$ *and* $\mathcal{T}, x_0 \models_{CTL^*} \psi$*;*
- $\mathcal{T}, p \models_{CTL^*} \neg\varphi$ *iff it does not hold that* $\mathcal{T}, p \models_{CTL^*} \varphi$*;*
- $\mathcal{T}, p \models_{CTL^*} \varphi \wedge \varphi'$ *iff* $\mathcal{T}, p \models_{CTL^*} \varphi$ *and* $\mathcal{T}, p \models_{CTL^*} \varphi'$*;*
- $\mathcal{T}, p \models_{CTL^*} \mathrm{X}\,\varphi$ *iff* $\mathcal{T}, p' \models_{CTL^*} \varphi$*, where* $p' = x_1, x_2, \ldots$ *if* $p = x_0, x_1, x_2, \ldots$*;*
- $\mathcal{T}, p \models_{CTL^*} \varphi\,\mathrm{U}\,\varphi'$ *iff there is some* $j \geq 0$ *such that* $\mathcal{T}, p_k \models_{CTL^*} \varphi$ *for all* $0 \leq k < j$ *and* $\mathcal{T}, p_j \models_{CTL^*} \varphi'$*, where* $p_i = x_i, x_{i+1}, \ldots$ *if* $p = x_0, x_1, \ldots$*.*

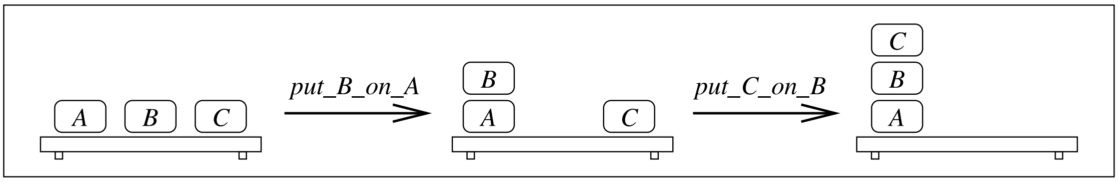


Figure 1: A possible scenario in the blocks-world domain.

We say that $\mathcal{T}$ satisfies the CTL formula ψ, written $\mathcal{T} \models_{CTL^*} \psi$, if $\mathcal{T}, \epsilon \models_{CTL^*} \psi$.*

The following theorem states that it is possible to build a tree automaton that accepts all the trees satisfying a CTL* formula. The tree automaton has a number of states that is doubly exponential and a parity index that is exponential in the length of the formula. A proof of this theorem has been given by Emerson and Jutla (1988).

Theorem 2 *Let ψ be a CTL* formula, and let $\mathcal{D} \subseteq \mathbb{N}^*$ be a finite set of arities. One can build a parity tree automaton $A^{\mathcal{D}}_{\psi}$ that accepts exactly the Σ-labelled $\mathcal{D}$-trees that satisfy ψ. The automaton $A^{\mathcal{D}}_{\psi}$ has $2^{2^{O(|\psi|)}}$ states and parity index $2^{O(|\psi|)}$, where $|\psi|$ is the length of formula ψ.*

3. Planning Domains and Plans

A *(nondeterministic) planning domain* (Cimatti et al., 2003) can be expressed in terms of a set of *states*, one of which is designated as the *initial state*, a set of *actions*, and a *transition function* describing how (the execution of) an action leads from one state to possibly many different states.

Definition 8 (planning domain) *A* planning domain *is a tuple $D = \langle \Sigma, \sigma_0, A, R \rangle$ where:*

- *Σ is the finite set of states;*
- *$\sigma_0 \in \Sigma$ is the initial state;*
- *A is the finite set of actions;*
- *$R : \Sigma \times A \to 2^{\Sigma}$ is the transition relation.*

We require that for each $\sigma \in \Sigma$ there is some $a \in A$ and some $\sigma' \in \Sigma$ such that $\sigma' \in R(\sigma, a)$. We assume that states Σ are ordered, and we write $R(\sigma, a) = \langle \sigma_1, \sigma_2, \ldots, \sigma_n \rangle$ whenever $R(\sigma, a) = \{\sigma_1, \sigma_2, \ldots, \sigma_n\}$ and $\sigma_1 < \sigma_2 < \cdots < \sigma_n$.

Example 1 *Consider a blocks-world domain consisting of a set of blocks, which are initially on a table, and which can be stacked on top of each other in order to build towers (see Figure 1).*

The states Σ of this domain are the possible configurations of the blocks: in the case of three blocks there are 13 states, corresponding to all the blocks on the table (1 configuration), a 2-block tower and the remaining block on the table (6 configurations), and a 3-block tower (6 possible configurations). We assume that initially all blocks are on the table.

The actions in this domain are put_X_on_Y, put_X_on_table, and wait, where X and Y are two (different) blocks. Actions put_X_on_Y and put_X_on_table are possible only if there are no blocks on top of X (otherwise we could not pick up X). In addition, action put_X_on_Y requires that there are no blocks on top of Y (otherwise we could not put X on top of Y).

We assume that the outcome of action put_X_on_Y is nondeterministic: indeed, trying to put a block on top of a tower may fail, in which case the tower is destroyed. Also action wait is nondeterministic: it is possible that the table is bumped and that all its towers are destroyed.

A *plan* guides the evolution of a planning domain by issuing actions to be executed. In the case of nondeterministic domains, *conditional plans* (Cimatti et al., 2003; Pistore & Traverso, 2001) are required, that is, the next action issued by the plan may depend on the outcome of the previous actions. Here we consider a very general definition of plans: a plan is a mapping from a sequence of states, representing the past history of the domain evolution, to an action to be executed.

Definition 9 (plan) *A* plan *is a partial function $\pi : \Sigma^+ \rightharpoonup A$ such that:*

- *if $\pi(w \cdot \sigma) = a$, then $\sigma' \in R(\sigma, a)$ for some σ';*
- *if $\pi(w \cdot \sigma) = a$, then $\sigma' \in R(\sigma, a)$ iff $w \cdot \sigma \cdot \sigma' \in \mathrm{dom}(\pi)$;*
- *if $w \cdot \sigma \in \mathrm{dom}(\pi)$ with $w \neq \epsilon$, then $w \in \mathrm{dom}(\pi)$;*
- *$\pi(\sigma)$ is defined iff $\sigma = \sigma_0$ is the initial state of the domain.*

The conditions in the previous definition ensure that a plan defines an action to be executed for exactly the finite paths $w \in \Sigma^+$ that can be reached executing the plan from the initial state of the domain.

Example 2 *A possible plan for the blocks-world domain of Example 1 is represented in Figure 2. We remark the importance of having plans in which the action to be executed depends on the whole sequence of states corresponding to the past history of the evolution. Indeed, according to the plan if Figure 2, two different actions put_C_on_A and put_C_on_table are performed in the state with block B on top of A, depending on the past history.*

Since we consider nondeterministic planning domains, the execution of an action may lead to different outcomes. Therefore, the execution of a plan on a planning domain can be described as a $(\Sigma{\times}A)$-labelled tree. Component Σ of the label of the tree corresponds to a state in the planning domain, while component A describes the action to be executed in that state.

Definition 10 (execution tree) *The* execution tree *for domain D and plan π is the $(\Sigma{\times}A)$-labelled tree $\mathcal{T}$ defined as follows:*

- *$\mathcal{T}(\epsilon) = (\sigma_0, a_0)$ where σ_0 is the initial state of the domain and $a_0 = \pi(\sigma_0)$;*

w	$\pi(w)$
A B C	*put_B_on_A*
A B C · B A C	*put_C_on_B*
A B C · B A C · C B A	*put_C_on_table*
A B C · B A C · C B A · B A C	*put_B_on_table*
A B C · B A C · C B A · B A C · A B C	*wait*
any other history	*wait*

Figure 2: A plan for the blocks-world domain.

- *if* $p = x_0, \ldots, x_n \in P^*(\mathcal{T})$ *with* $\mathcal{T}(p) = (\sigma_0, a_0) \cdot (\sigma_1, a_1) \cdots (\sigma_n, a_n)$*, and if* $R(\sigma_n, a_n) = \langle \sigma'_0, \ldots, \sigma'_{d-1} \rangle$*, then for every* $0 \leq i < d$ *the following conditions hold:* $x_n \cdot i \in \mathrm{dom}(\mathcal{T})$ *and* $\mathcal{T}(x_n \cdot i) = (\sigma'_i, a'_i)$ *with* $a'_i = \pi(\sigma_0 \cdot \sigma_1 \cdots \sigma_n \cdot \sigma'_i)$.

A *planning problem* consists of a planning domain and of a goal g that defines the set of desired behaviors. In the following, we assume that the goal g defines a set of execution trees, namely the execution trees that exhibit the behaviors described by the goal (we say that these execution trees satisfy the goal).

Definition 11 (planning problem) *A* planning problem *is a pair* (D, g)*, where* D *is a planning domain and* g *is a* goal*. A* solution *to a planning problem* (D, g) *is a plan* π *such that the execution tree for* π *satisfies the goal* g*.*

4. A Logic of Path Quantifiers

In this section we define a new logic that is based on LTL and that extends it with the possibility of defining conditions on the sets of paths that satisfy the LTL property. We start by motivating why such a logic is necessary for defining planning goals.

Example 3 *Consider the blocks-world domain introduced in the previous section. Intuitively, the plan of Example 2 is a solution to the goal of building a tower consisting of blocks A, B, C and then of destroying it. This goal can be easily formulated as an LTL* *formula:*

$$\varphi_1 = \mathrm{F}\ ((C\_on\_B \wedge B\_on\_A \wedge A\_on\_table) \wedge \mathrm{F}\ (C\_on\_table \wedge B\_on\_table \wedge A\_on\_table)).$$

Notice however that, due to the nondeterminism in the outcome of actions, this plan may fail to satisfy the goal. It is possible, for instance, that action put_C_on_B fails and the tower is destroyed. In this case, the plan proceeds performing wait actions, and hence the tower is never finished. Formally, the plan is a solution to the goal which requires that there is some *path in the execution structure that satisfies the LTL formula φ_1.*

Clearly, there are better ways to achieve the goal of building a tower and then destroying it: if we fail building the tower, rather than giving up, we can restart building it and keep trying until we succeed. This strategy allows for achieving the goal in "most of the paths": only if we keep destroying the tower when we try to build it we will not achieve the goal. As we will see, the logic of path quantifiers that we are going to define will allow us to formalize what we mean by "most of the paths".

Consider now the following LTL formula:

$$\varphi_2 = \mathrm{F}\ \mathrm{G}\ ((C\_on\_B \wedge B\_on\_A \wedge A\_on\_table).$$

The formula requires building a tower and maintaining it. In this case we have two possible ways to fail to achieve the goal. We can fail to build the tower; or, once built, we can fail to maintain it (remember that a wait action may nondeterministically lead to a destruction of the tower). Similarly to the case of formula ϕ_1, a planning goal that requires satisfying the formula ϕ_2 in all *paths of the execution tree is unsatisfiable. On the other hand, a goal that requires satisfying it on* some *paths is very weak; our logic allows us to be more demanding on the paths that satisfy the formula.*

Finally, consider the following LTL formula:

$$\varphi_3 = \mathrm{G}\ \mathrm{F}\ ((C\_on\_B \wedge B\_on\_A \wedge A\_on\_table).$$

It requires that the tower exists infinitely many time, i.e., if the tower gets destroyed, then we have to rebuild it. Intuitively, this goal admits plans that can achieve it more often, i.e., on "more paths", than φ_2. Once again, a path logic is needed to give a formal meaning to "more paths".

In order to be able to represent the planning goals discussed in the previous example, we consider logic formulas of the form $\alpha.\varphi$, where φ is an LTL formula and α is a path quantifier and defines a set of infinite paths on which the formula φ should be checked. Two extreme cases are the path quantifier $\mathcal{A}$, which is used to denote that φ must hold on *all* the paths, and the path quantifier $\mathcal{E}$, which is used to denote that φ must hold on *some* paths. In general, a path quantifier is a (finite or infinite) word on alphabet $\{\mathcal{A}, \mathcal{E}\}$ and defines an alternation in the selection of the two modalities corresponding to $\mathcal{E}$ and $\mathcal{A}$. For instance, by writing $\mathcal{AE}.\varphi$ we require that all finite paths have some infinite extension that satisfies φ, while by writing $\mathcal{EA}.\varphi$ we require that all the extensions of some finite path satisfy φ.

The path quantifier can be seen as the definition of a two-player game for the selection of the paths that should satisfy the LTL formula. Player A (corresponding to $\mathcal{A}$) tries to build a path that does not satisfy the LTL formula, while player E (corresponding to $\mathcal{E}$) tries to build the path so that the LTL formula holds. Different path quantifiers define different alternations in the turns of players A and E. The game starts from the path consisting only of the initial state, and, during their turns, players A and E extend the path by a finite number of nodes. In the case the path quantifier is a finite word, the player that moves last in the game extends the finite path built so far to an infinite path. The formula is satisfied if player E has a winning strategy, namely if, for all the possible moves of the player A, it is always able to build a path that satisfies the LTL formula.

Example 4 *Let us consider the three LTL formulas defined in Example 3, and let us see how the path quantifiers we just introduced can be applied.*

In the case of formula φ_1, the plan presented in Example 2 satisfies requirement $\mathcal{E}.\varphi_1$: there is a path on which the tower is built and then destroyed. It also satisfies the "stronger" requirement $\mathcal{EA}.\varphi_1$ that stresses the fact that, in this case, once the tower has been built and destroyed, we can safely give the control to player A. Formula φ_1 can be satisfied in a stronger way, however. Indeed, the plan that keeps trying to build the tower satisfies the requirement $\mathcal{AE}.\varphi_1$, as well as the requirement $\mathcal{AEA}.\varphi_1$: player A cannot reach a state where the satisfaction of the goal is prevented.

Let us now consider the formula φ_2. In this case, we can find plans satisfying $\mathcal{AE}.\varphi_2$, but no plan can satisfy requirement $\mathcal{AEA}.\varphi_2$. Indeed, player A has a simple strategy to win, if he gets the control after we built the tower: bump the table. Similar considerations hold also for formula φ_3. Also in this case, we can find plans for requirement $\mathcal{AE}.\varphi_3$, but not for requirement $\mathcal{AEA}.\varphi_3$. In this case, however, plans exist also for requirement $\mathcal{AEAEAE}\cdots.\varphi_3$: if player E gets the control infinitely often, then it can rebuild the tower if needed.

In the rest of the section we give a formal definition and study the basic properties of this logic of path quantifiers.

4.1 Finite Games

We start considering only games with a finite number of moves, that is path quantifiers corresponding to finite words on $\{\mathcal{A}, \mathcal{E}\}$.

Definition 12 ($\mathcal{AE}$-LTL) *An $\mathcal{AE}$-LTL formula is a pair $g = \alpha.\varphi$, where φ is an LTL formula and $\alpha \in \{\mathcal{A}, \mathcal{E}\}^+$ is a path quantifier.*

The following definition describes the games corresponding to the finite path quantifiers.

Definition 13 (semantics of $\mathcal{AE}$-LTL) *Let p be a finite path of a Σ-labelled tree $\mathcal{T}$. Then:*

- $p \models \mathcal{A}\alpha.\varphi$ *if for all finite extensions p' of p it holds that $p' \models \alpha.\varphi$.*
- $p \models \mathcal{E}\alpha.\varphi$ *if for some finite extension p' of p it holds that $p' \models \alpha.\varphi$.*
- $p \models \mathcal{A}.\varphi$ *if for all infinite extensions p' of p it holds that $\mathcal{T}(p') \models_{LTL} \varphi$.*
- $p \models \mathcal{E}.\varphi$ *if for some infinite extension p' of p it holds that $\mathcal{T}(p') \models_{LTL} \varphi$.*

We say that the Σ-labelled tree $\mathcal{T}$ satisfies the $\mathcal{AE}$-LTL formula g, and we write $\mathcal{T} \models g$, if $p_0 \models g$, where $p_0 = \epsilon$ is the root of $\mathcal{T}$.

$\mathcal{AE}$-LTL allows for path quantifiers consisting of an arbitrary combination of $\mathcal{A}$s and $\mathcal{E}$s. Each combination corresponds to a different set of rules for the game between A and E. In Theorem 4 we show that all this freedom in the definition of the path quantifier is not needed. Only six path quantifiers are sufficient to capture all the possible games. This result is based on the concept of *equivalent path quantifiers*.

Consider formulas $\mathcal{A}.\,\mathrm{F}\,p$ and $\mathcal{AE}.\,\mathrm{F}\,p$. It is easy to see that the two formulas are equi-satisfiable, i.e., if a tree $\mathcal{T}$ satisfies $\mathcal{A}.\,\mathrm{F}\,p$ then it also satisfies $\mathcal{AE}.\,\mathrm{F}\,p$, and vice-versa. In this case, path quantifiers $\mathcal{A}$ and $\mathcal{AE}$ have the same "power", but this depends on the fact that we use the path quantifiers in combination with the LTL formula $\mathrm{F}\,p$. If we combine the two path quantifiers with different LTL formulas, such as $\mathrm{G}\,p$, it is possible to find trees that satisfy the latter path quantifier but not the former. For this reason, we cannot consider the two path quantifiers equivalent. Indeed, in order for two path quantifiers to be equivalent, they have to be equi-satisfiable for *all* the LTL formulas. This intuition is formalized in the following definition.

Definition 14 (equivalent path quantifiers) *Let α and α' be two path quantifiers. We say that α* implies *α', written $\alpha \rightsquigarrow \alpha'$, if for all Σ-labelled trees $\mathcal{T}$ and for all LTL formulas φ, $\mathcal{T} \models \alpha.\varphi$ implies $\mathcal{T} \models \alpha'.\varphi$. We say that α is* equivalent *to α', written $\alpha \sim \alpha'$, if $\alpha \rightsquigarrow \alpha'$ and $\alpha' \rightsquigarrow \alpha$.*

The following lemma describes some basic properties of path quantifiers and of the equivalences among them. We will exploit these results in the proof of Theorem 4.

Lemma 3 *Let $\alpha, \alpha' \in \{\mathcal{A}, \mathcal{E}\}^*$. The following implications and equivalences hold.*

1. $\alpha\mathcal{AA}\alpha' \sim \alpha\mathcal{A}\alpha'$ *and* $\alpha\mathcal{EE}\alpha' \sim \alpha\mathcal{E}\alpha'$.
2. $\alpha\mathcal{A}\alpha' \rightsquigarrow \alpha\alpha'$ *and* $\alpha\alpha' \rightsquigarrow \alpha\mathcal{E}\alpha'$*, if* $\alpha\alpha'$ *is not empty.*
3. $\alpha\mathcal{A}\alpha' \rightsquigarrow \alpha\mathcal{AEA}\alpha'$ *and* $\alpha\mathcal{EAE}\alpha' \rightsquigarrow \alpha\mathcal{E}\alpha'$.
4. $\alpha\mathcal{AEAE}\alpha' \sim \alpha\mathcal{AE}\alpha'$ *and* $\alpha\mathcal{EAEA}\alpha' \sim \alpha\mathcal{EA}\alpha'$.

Proof. In the proof of this lemma, in order to prove that $\alpha\alpha' \rightsquigarrow \alpha\alpha''$ we prove that, given an arbitrary tree $\mathcal{T}$ and an arbitrary LTL formula φ, $p \models \alpha'.\varphi$ implies $p \models \alpha''.\varphi$ for every finite path p of $\mathcal{T}$. Indeed, if $p \models \alpha'.\varphi$ implies $p \models \alpha''.\varphi$ for all finite paths p, then it is easy to prove, by induction on α, that $p \models \alpha\alpha'.\varphi$ implies $p \models \alpha\alpha''.\varphi$ for all finite paths p. In the following, we will refer to this proof technique as prefix induction.

1. We show that, for every finite path p, $p \models \mathcal{AA}\alpha'.\varphi$ if and only if $p \models \mathcal{A}\alpha'.\varphi$: then the equivalence of $\alpha\mathcal{AA}\alpha'$ and $\alpha\mathcal{A}\alpha'$ follows by prefix induction.

   Let us assume that $p \models \mathcal{AA}\alpha'.\varphi$. We prove that $p \models \mathcal{A}\alpha'.\varphi$, that is, that $p' \models \alpha'.\varphi$ for every finite[3] extension p' of p. Since $p \models \mathcal{AA}\alpha'.\varphi$, by Definition 13 we know that,

3. We assume that α' is not the empty word. The proof in the case α' is the empty word is similar.

for every finite extension p' of p, $p' \models \mathcal{A}\alpha'.\varphi$. Hence, again by Definition 13, we know that for every finite extension p'' of p', $p'' \models \alpha'.\varphi$. Since p' is a finite extension of p', we can conclude that $p' \models \alpha'.\varphi$. Therefore, $p' \models \alpha'.\varphi$ holds for all finite extensions p' of p.

Let us now assume that $p \models \mathcal{A}\alpha'.\varphi$. We prove that $p \models \mathcal{A}\mathcal{A}\alpha'.\varphi$, that is, for all finite extensions p' of p, and for all finite extensions p'' of p', $p'' \models \alpha'.\varphi$. We remark that the finite path p'' is also a finite extension of p, and therefore $p'' \models \alpha'.\varphi$ holds since $p \models \mathcal{A}\alpha'.\varphi$.

This concludes the proof of the equivalence of $\alpha\mathcal{A}\mathcal{A}\alpha'$ and $\alpha\mathcal{A}\alpha'$. The proof of the equivalence of $\alpha\mathcal{E}\mathcal{E}\alpha'$ and $\alpha\mathcal{E}\alpha'$ is similar.

2. Let us assume first that α' is not an empty word. We distinguish two cases, depending on the first symbol of α'. If $\alpha' = \mathcal{A}\alpha''$, then we should prove that $\alpha\mathcal{A}\mathcal{A}\alpha'' \rightsquigarrow \alpha\mathcal{A}\alpha''$, which we already did in item 1 of this lemma. If $\alpha' = \mathcal{E}\alpha''$, then we show that, for every finite path p, if $p \models \mathcal{A}\mathcal{E}\alpha''.\varphi$ then $p \models \mathcal{E}\alpha''.\varphi$: then $\alpha\mathcal{A}\alpha' \rightsquigarrow \alpha\alpha'$ follows by prefix induction. Let us assume that $p \models \mathcal{A}\mathcal{E}\alpha''.\varphi$. Then, for all finite extensions p' of p there exists some finite[4] extension p'' of p' such that $p'' \models \alpha'.\varphi$. Let us take $p' = p$. Then we know that there is some finite extension p'' of p such that $p'' \models \alpha'.\varphi$, that is, according to Definition 13, $p \models \mathcal{E}\alpha'.\varphi$.

   Let us now assume that α' is the empty word. By hypothesis, $\alpha\alpha' \neq \epsilon$, so α is not empty. We distinguish two cases, depending on the last symbol of α. If $\alpha = \alpha''\mathcal{A}$, then we should prove that $\alpha''\mathcal{A}\mathcal{A} \rightsquigarrow \alpha''\mathcal{A}$, which we already did in item 1 of this lemma. If $\alpha = \alpha''\mathcal{E}$, then we prove that for every finite path p, if $p \models \mathcal{E}\mathcal{A}.\varphi$ then $p \models \mathcal{E}.\varphi$: then $\alpha''\mathcal{E}\mathcal{A} \rightsquigarrow \alpha''\mathcal{E}$ follows by prefix induction. Let us assume that $p \models \mathcal{E}\mathcal{A}.\varphi$. By Definition 13, there exists some finite extension p' of p such that, for every infinite extension p'' of p' we have $\mathcal{T}(p'') \models_{\mathrm{LTL}} \varphi$. Let p'' be any infinite extension of p'. We know that p'' is also an infinite extension of p, and that $\mathcal{T}(p'') \models_{\mathrm{LTL}} \varphi$. Then, by Definition 13 we deduce that $p \models \mathcal{E}.\varphi$.

   This concludes the proof that $\alpha\mathcal{A}\alpha' \rightsquigarrow \alpha\alpha'$. The proof that $\alpha\alpha' \rightsquigarrow \alpha\mathcal{E}\alpha'$ is similar.

3. By item 1 of this lemma we know that $\alpha\mathcal{A}\alpha' \rightsquigarrow \alpha\mathcal{A}\mathcal{A}\alpha'$ and by item 2 we know that $\alpha\mathcal{A}\mathcal{A}\alpha' \rightsquigarrow \alpha\mathcal{A}\mathcal{E}\mathcal{A}\alpha'$. This concludes the proof that $\alpha\mathcal{A}\alpha' \rightsquigarrow \alpha\mathcal{A}\mathcal{E}\mathcal{A}\alpha'$. The proof that $\alpha\mathcal{E}\mathcal{A}\mathcal{E}\alpha' \rightsquigarrow \alpha\mathcal{E}\alpha'$ is similar.

4. By item 3 of this lemma we know that $(\alpha\mathcal{A})\mathcal{E}\mathcal{A}\mathcal{E}\alpha' \rightsquigarrow (\alpha\mathcal{A})\mathcal{E}\alpha'$. Moreover, again by item 3, we know that $\alpha\mathcal{A}(\mathcal{E}\alpha') \rightsquigarrow \alpha\mathcal{A}\mathcal{E}\mathcal{A}(\mathcal{E}\alpha')$. Therefore, we deduce $\alpha\mathcal{A}\mathcal{E}\alpha' \sim \alpha\mathcal{A}\mathcal{E}\mathcal{A}\mathcal{E}\alpha'$. The proof that $\alpha\mathcal{E}\mathcal{A}\alpha' \sim \alpha\mathcal{E}\mathcal{A}\mathcal{E}\mathcal{A}\alpha'$ is similar. □

We can now prove the first main result of the paper: each finite path quantifier is equivalent to a *canonical path quantifier* of length at most three.

Theorem 4 *For each finite path quantifier α there is a canonical finite path quantifier*

$$\alpha' \in \{\mathcal{A}, \mathcal{E}, \mathcal{A}\mathcal{E}, \mathcal{E}\mathcal{A}, \mathcal{A}\mathcal{E}\mathcal{A}, \mathcal{E}\mathcal{A}\mathcal{E}\}$$

4. We assume that α'' is not the empty word. The proof in the case where α'' is empty is similar.

such that $\alpha \sim \alpha'$. Moreover, the following implications hold between the canonical finite path quantifiers:

$$\begin{tikzcd} \mathcal{A} \arrow[r, rightsquigarrow] & \mathcal{AEA} \arrow[r, rightsquigarrow] \arrow[d, rightsquigarrow] & \mathcal{AE} \arrow[d, rightsquigarrow] & \\ & \mathcal{EA} \arrow[r, rightsquigarrow] & \mathcal{EAE} \arrow[r, rightsquigarrow] & \mathcal{E} \end{tikzcd} \tag{1}$$

Proof. We first prove that each path quantifier α is equivalent to some canonical path quantifier α'. By an iterative application of Lemma 3(1), we obtain from α a path quantifier α'' such that $\alpha \sim \alpha''$ and α'' does not contain two adjacent $\mathcal{A}$ or $\mathcal{E}$. Then, by an iterative application of Lemma 3(4), we can transform α'' into an equivalent path quantifier α' of length at most 3. The canonical path quantifiers in (1) are precisely those quantifiers of length at most 3 that do not contain two adjacent $\mathcal{A}$ or $\mathcal{E}$.
For the implications in (1):

- $\mathcal{A} \rightsquigarrow \mathcal{AEA}$ and $\mathcal{EAE} \rightsquigarrow \mathcal{E}$ come from Lemma 3(3);
- $\mathcal{AEA} \rightsquigarrow \mathcal{EA}$ and $\mathcal{AE} \rightsquigarrow \mathcal{EAE}$ come from Lemma 3(2);
- $\mathcal{AEA} \rightsquigarrow \mathcal{AE}$ and $\mathcal{EA} \rightsquigarrow \mathcal{EAE}$ come from Lemma 3(2). □

We remark that Lemma 3 and Theorem 4 do not depend on the usage of LTL for formula φ. They depend on the general observation that $\alpha \rightsquigarrow \alpha'$ whenever player E can select for game α' a set of paths which is a subset of those selected for game α.

4.2 Infinite Games

We now consider infinite games, namely path quantifiers consisting of infinite words on alphabet $\{\mathcal{A}, \mathcal{E}\}$. We will see that infinite games can express all the finite path quantifiers that we have studied in the previous subsection, but that there are some infinite games, corresponding to an infinite alternation of the two players A and E, which cannot be expressed with finite path quantifiers.

In the case of infinite games, we assume that player E moves according to a strategy ξ that suggests how to extend each finite path. We say that $\mathcal{T} \models \alpha.\varphi$, where α is an infinite game, if there is some winning strategy ξ for player E. A strategy ξ is winning if, whenever p is an infinite path of $\mathcal{T}$ obtained according to α — i.e., by allowing player A to play in an arbitrary way and by requiring that player E follows strategy ξ — then p satisfies the LTL formula φ.

Definition 15 (strategy) *A strategy for a Σ-labelled tree $\mathcal{T}$ is a mapping $\xi : P^*(\mathcal{T}) \to P^*(\mathcal{T})$ that maps every finite path p to one of its finite extensions $\xi(p)$.*

Definition 16 (semantics of $\mathcal{AE}$-LTL) *Let $\alpha = \Pi_0\Pi_1 \cdots$ with $\Pi_i \in \{\mathcal{A}, \mathcal{E}\}$ be an infinite path quantifier. An infinite path p is a* possible outcome *of game α with strategy ξ if there is a* generating sequence *for it, namely, an infinite sequence $p_0, p_1, \ldots$ of finite paths such that:*

- *p_i are finite prefixes of p;*

- *$p_0 = \epsilon$ is the root of tree $\mathcal{T}$;*
- *if $\Pi_i = \mathcal{E}$ then $p_{i+1} = \xi(p_i)$;*
- *if $\Pi_i = \mathcal{A}$ then p_{i+1} is an (arbitrary) extension of p_i.*

We denote with $\mathcal{P}_{\mathcal{T}}(\alpha, \xi)$ the set of infinite paths of $\mathcal{T}$ that are possible outcomes of game α with strategy ξ. The tree $\mathcal{T}$ satisfies the $\mathcal{AE}$-LTL formula $g = \alpha.\varphi$, written $\mathcal{T} \models g$, if there is some strategy ξ such that $\mathcal{T}(p) \models_{LTL} \varphi$ for all paths $p \in \mathcal{P}_{\mathcal{T}}(\alpha, \xi)$.

We remark that it is possible that the paths in a generating sequence stop growing, i.e., that there is some p_i such that $p_i = p_j$ for all $j \geq i$. In this case, according to the previous definition, all infinite paths p that extend p_i are possible outcomes.

In the next lemmas we extend the analysis of equivalence among path quantifiers to infinite games.[5] The first lemma shows that finite path quantifiers are just particular cases of infinite path quantifiers, namely, they correspond to those infinite path quantifiers that end with an infinite sequence of $\mathcal{A}$ or of $\mathcal{E}$.

Lemma 5 *Let α be a finite path quantifier. Then $\alpha(\mathcal{A})^\omega \sim \alpha\mathcal{A}$ and $\alpha(\mathcal{E})^\omega \sim \alpha\mathcal{E}$.*

Proof. We prove that $\alpha(\mathcal{A})^\omega \sim \alpha\mathcal{A}$. The proof of the other equivalence is similar.
First, we prove that $\alpha(\mathcal{A})^\omega \rightsquigarrow \alpha\mathcal{A}$. Let $\mathcal{T}$ be a tree and φ be an LTL formula such that $\mathcal{T} \models \alpha(\mathcal{A})^\omega.\varphi$. Moreover, let ξ be any strategy such that all $p \in \mathcal{P}_{\mathcal{T}}(\alpha(\mathcal{A})^\omega, \xi)$ satisfy φ. In order to prove that $\mathcal{T} \models \alpha\mathcal{A}.\varphi$ it is sufficient to use the strategy ξ in the moves of player E, namely, whenever we need to prove that $p \models \mathcal{E}\alpha'.\varphi$ according to Definition 13, we take $p' = \xi(p)$ and we move to prove that $p' \models \alpha'.\varphi$. In this way, the infinite paths selected by Definition 13 for $\alpha\mathcal{A}$ coincide with the possible outcomes of game $\alpha(\mathcal{A})^\omega$, and hence satisfy the LTL formula φ.
This concludes the proof that $\alpha(\mathcal{A})^\omega \rightsquigarrow \alpha\mathcal{A}$. We now prove that $\alpha\mathcal{A} \rightsquigarrow \alpha(\mathcal{A})^\omega$. We distinguish three cases.

- **Case $\alpha = (\mathcal{A})^n$, with $n \geq 0$.**

  In this case, $\alpha\mathcal{A} \sim \mathcal{A}$ (Lemma 3(1)) and $\alpha(\mathcal{A})^\omega = (\mathcal{A})^\omega$. Let $\mathcal{T}$ be a tree and φ be an LTL formula. Then $\mathcal{T} \models \mathcal{A}.\varphi$ if and only if all the paths of $\mathcal{T}$ satisfy formula φ. It is easy to check that also $\mathcal{T} \models (\mathcal{A})^\omega.\varphi$ if and only if all the paths of $\mathcal{T}$ satisfy formula φ. This is sufficient to conclude that $(\mathcal{A})^n\mathcal{A} \sim (\mathcal{A})^n(\mathcal{A})^\omega$.

- **Case $\alpha = \mathcal{E}\alpha'$.**

  In this case, $\alpha\mathcal{A} \sim \mathcal{EA}$. Indeed, $\alpha\mathcal{A}$ is an arbitrary path quantifier that starts with $\mathcal{E}$ and ends with $\mathcal{A}$. By Lemma 3(1), we can collapse adjacent occurrences of $\mathcal{A}$ and of $\mathcal{E}$, thus obtaining $\alpha\mathcal{A} \sim (\mathcal{EA})^n$ for some $n > 0$. Moreover, by Lemma 3(4) we have $(\mathcal{EA})^n \sim \mathcal{EA}$.

  Let $\mathcal{T}$ be a tree and φ be an LTL formula. Then $\mathcal{T} \models \mathcal{EA}.\varphi$ if and only if there is some finite path $\bar{p}$ of $\mathcal{T}$ such that all the infinite extensions of $\bar{p}$ satisfy φ. Now, let

5. The definitions of the implication and equivalence relations (Definition 14) also apply to the case of infinite path quantifiers.

ξ be any strategy such that $\xi(\epsilon) = \bar{p}$. Then every infinite path $p \in \mathcal{P}_{\mathcal{T}}(\mathcal{E}\alpha'(\mathcal{A})^\omega, \xi)$ satisfies φ. Indeed, since player E has the first turn, all the possible outcomes are infinite extensions of $\xi(\epsilon) = \bar{p}$.

This concludes the proof that $\mathcal{E}\alpha'\mathcal{A} \rightsquigarrow \mathcal{E}\alpha'(\mathcal{A})^\omega$.

- **Case** $\alpha = (\mathcal{A})^n\mathcal{E}\alpha'$**, with** $n > 0$.

  Reasoning as in the proof of the previous case, it is easy to show that $\alpha\mathcal{A} \sim \mathcal{AEA}$.

  Let $\mathcal{T}$ be a tree and φ be an LTL formula. Then $\mathcal{T} \models \mathcal{AEA}.\varphi$ if and only if for every finite path p of $\mathcal{T}$ there is some finite extension p' of p such that all the infinite extensions of p' satisfy the formula φ. Let ξ be any strategy such that $p' = \xi(p)$ is a finite extension of p such that all the infinite extensions of p' satisfy φ. Then every infinite path $p \in \mathcal{P}_{\mathcal{T}}((\mathcal{A})^n\mathcal{E}\alpha'(\mathcal{A})^\omega, \xi)$ satisfies φ. Indeed, let $p_0, p_1, \ldots, p_n, p_{n+1}, \ldots$ be a generating sequence for p. Then $p_{n+1} = \xi(p_n)$ and p is an infinite extension of p_{n+1}. By construction of ξ we know that p satisfies φ.

  This concludes the proof that $(\mathcal{A})^n\mathcal{E}\alpha'\mathcal{A} \rightsquigarrow (\mathcal{A})^n\mathcal{E}\alpha'(\mathcal{A})^\omega$.

Every finite path quantifier α falls in one of the three considered cases. Therefore, we can conclude that $\alpha\mathcal{A} \rightsquigarrow \alpha(\mathcal{A})^\omega$ for every finite path quantifier α. $\square$

The next lemma defines a sufficient condition for proving that $\alpha \rightsquigarrow \alpha'$. This condition is useful for the proofs of the forthcoming lemmas.

Lemma 6 *Let α and α' be two infinite path quantifiers. Let us assume that for all Σ-labelled trees and for each strategy ξ there is some strategy ξ' such that $\mathcal{P}_{\mathcal{T}}(\alpha', \xi') \subseteq \mathcal{P}_{\mathcal{T}}(\alpha, \xi)$. Then $\alpha \rightsquigarrow \alpha'$.*

Proof. Let us assume that $\mathcal{T} \models \alpha.\varphi$. Then there is a suitable strategy ξ such that all $p \in \mathcal{P}_{\mathcal{T}}(\alpha, \xi)$ satisfy the LTL formula φ. Let ξ' be a strategy such that all $\mathcal{P}_{\mathcal{T}}(\alpha', \xi') \subseteq \mathcal{P}_{\mathcal{T}}(\alpha, \xi)$. By hypothesis, all possible outcomes for game α' and strategy ξ' satisfy the LTL formula φ, and hence $\mathcal{T} \models \alpha'.\varphi$. This concludes the proof that $\alpha \rightsquigarrow \alpha'$. $\square$

In the next lemma we show that all the games where players A and E alternate infinitely often are equivalent to one of the two games $(\mathcal{AE})^\omega$ and $(\mathcal{EA})^\omega$. That is, we can assume that each player extends the path only once before the turn passes to the other player.

Lemma 7 *Let α be an infinite path quantifier that contains an infinite number of $\mathcal{A}$ and an infinite number of $\mathcal{E}$. Then $\alpha \sim (\mathcal{AE})^\omega$ or $\alpha \sim (\mathcal{EA})^\omega$.*

Proof. Let $\alpha = (\mathcal{A})^{m_1}(\mathcal{E})^{n_1}(\mathcal{A})^{m_2}(\mathcal{E})^{n_2}\cdots$ with $m_i, n_i > 0$. We show that $\alpha \sim (\mathcal{AE})^\omega$.
First, we prove that $(\mathcal{AE})^\omega \rightsquigarrow \alpha$. Let ξ be a strategy for the tree $\mathcal{T}$ and let p be an infinite path of $\mathcal{T}$. We show that if $p \in \mathcal{P}_{\mathcal{T}}(\alpha, \xi)$ then $p \in \mathcal{P}_{\mathcal{T}}((\mathcal{AE})^\omega, \xi)$. By Lemma 6 this is sufficient for proving that $(\mathcal{AE})^\omega \rightsquigarrow \alpha$.
Let $p_0, p_1, \ldots$ be a generating sequence for p according to α and ξ. Moreover, let $p'_0 = \epsilon$, $p'_{2i+1} = p_{m_1+n_1+\cdots+m_{i-1}+n_{i-1}+m_i}$ and and $p'_{2i+2} = p_{m_1+n_1+\cdots+m_{i-1}+n_{i-1}+m_i+1}$. It is easy to check that $p'_0, p'_1, p'_2, \ldots$ is a valid generating sequence for p according to game $(\mathcal{AE})^\omega$ and strategy ξ. Indeed, extensions $p'_0 \rightarrow p'_1$, $p'_2 \rightarrow p'_3$, $p'_4 \rightarrow p'_5$, $\ldots$ are moves of player A,

and hence can be arbitrary. Extensions $p'_1 \to p'_2$, $p'_3 \to p'_4$, ... correspond to extensions $p_{m_1} \to p_{m_1+1}$, $p_{m_1+n_1+m_2} \to p_{m_1+n_1+m_2+1}$, ..., which are moves of player E and hence respect strategy ξ.
We now prove that $\alpha \rightsquigarrow (\mathcal{A}\mathcal{E})^\omega$. Let ξ be a strategy for the tree $\mathcal{T}$. We define a strategy $\bar{\xi}$ such that if $p \in \mathcal{P}_\mathcal{T}((\mathcal{A}\mathcal{E})^\omega, \bar{\xi})$, then $p \in \mathcal{P}_\mathcal{T}(\alpha, \xi)$. By Lemma 6 this is sufficient for proving that $\alpha \rightsquigarrow (\mathcal{A}\mathcal{E})^\omega$.
Let $\bar{p}$ be a finite path. Then $\bar{\xi}(\bar{p}) = \xi^{k_{\bar{p}}}(\bar{p})$ with $k_{\bar{p}} = \sum_{i=1}^{|\bar{p}|} n_i$. That is, strategy $\bar{\xi}$ on path $\bar{p}$ is obtained by applying $k_{\bar{p}}$ times strategy ξ. The number of times strategy ξ is applied depends on the length $|\bar{p}|$ of path $\bar{p}$.
We show that, if p is a possible outcome of the game α with strategy $\bar{\xi}$, then p is a possible outcome of the game $(\mathcal{A}\mathcal{E})^\omega$ with strategy ξ. Let $p_0, p_1, \ldots$ be a generating sequence for p according to $(\mathcal{A}\mathcal{E})^\omega$ and $\bar{\xi}$. Then

$$p_0, \underbrace{p_1, ..., p_1}_{m_1 \text{ times}}, \underbrace{\xi(p_1), \xi^2(p_1), ..., \xi^{n_1}(p_1)}_{n_1 \text{ times}}, \underbrace{p_3, ..., p_3}_{m_2 \text{ times}},$$
$$\underbrace{\xi(p_3), \xi^2(p_3), ..., \xi^{n_2}(p_3)}_{n_2 \text{ times}}, \underbrace{p_5, ..., p_5}_{m_3 \text{ times}}, \cdots$$

is a valid generating sequence for p according to α and ξ. The extensions corresponding to an occurrence of symbol $\mathcal{E}$ in α consist of an application of the strategy ξ and are hence valid for player E. Moreover, extension $\xi^{n_i}(p_{2i-1}) \to p_{2i+1}$ is a valid move for player A because p_{2i+1} is an extension of $\xi^{n_i}(p_{2i-1})$. Indeed, $\xi^{n_i}(p_{2i-1})$ is a prefix of p_{2i} (and hence of p_{2i+1}) since $p_{2i} = \bar{\xi}(p_{2i-1}) = \xi^{k_{p_{2i-1}}}(p_{2i-1})$ and $k_{p_{2i-1}} = \sum_{x=1}^{|p_{2i-1}|} n_x \geq n_i$, since $|p_{2i-1}| \geq i$. The other conditions of Definition 16 can be easily checked.
This concludes the proof that $\alpha \sim (\mathcal{A}\mathcal{E})^\omega$ for $\alpha = (\mathcal{A})^{m_1}(\mathcal{E})^{n_1}(\mathcal{A})^{m_2}(\mathcal{E})^{n_2}\cdots$. The proof that $\alpha \sim (\mathcal{E}\mathcal{A})^\omega$ for $\alpha = (\mathcal{E})^{m_1}(\mathcal{A})^{n_1}(\mathcal{E})^{m_2}(\mathcal{A})^{n_2}\cdots$ is similar. $\square$

The next lemma contains other auxiliary results on path quantifiers.

Lemma 8 *Let α be a finite path quantifier and α' be an infinite path quantifier.*

1. *$\alpha\mathcal{A}\alpha' \rightsquigarrow \alpha\alpha'$ and $\alpha\alpha' \rightsquigarrow \alpha\mathcal{E}\alpha'$.*
2. *$\alpha(\mathcal{A})^\omega \rightsquigarrow \alpha\mathcal{A}\alpha'$ and $\alpha\mathcal{E}\alpha' \rightsquigarrow \alpha(\mathcal{E})^\omega$.*

Proof.

1. We prove that $\alpha\mathcal{A}\alpha' \rightsquigarrow \alpha\alpha'$. Let ξ be a strategy for tree $\mathcal{T}$ and let p be an infinite path of $\mathcal{T}$. We show that if $p \in \mathcal{P}_\mathcal{T}(\alpha\alpha', \xi)$ then $p \in \mathcal{P}_\mathcal{T}(\alpha\mathcal{A}\alpha', \xi)$. Let $p_0, p_1, \ldots$ be a generating sequence for p according to $\alpha\alpha'$ and ξ. Then it is easy to check that $p_0, p_1, \ldots, p_{i-1}, p_i, p_i, p_{i+1}, \ldots$, where i is the length of α, is a valid generating sequence for p according to $\alpha\mathcal{A}\alpha'$ and ξ. Indeed, the extension $p_i \to p_i$ is a valid move for player A. This concludes the proof that $\alpha\mathcal{A}\alpha' \rightsquigarrow \alpha\alpha'$.

   Now we prove that $\alpha\alpha' \rightsquigarrow \alpha\mathcal{E}\alpha'$. If $\alpha' = (\mathcal{E})^\omega$, then $\alpha\mathcal{E}\alpha' = \alpha\mathcal{E}(\mathcal{E})^\omega = \alpha(\mathcal{E})^\omega = \alpha\alpha'$, and $\alpha\mathcal{E}\alpha' \rightsquigarrow \alpha\alpha'$ is trivially true. If $\alpha' \neq (\mathcal{E})^\omega$, we can assume, without loss of generality, that $\alpha' = \mathcal{A}\alpha''$. In this case, let ξ be a strategy for tree $\mathcal{T}$ and let p be a path of $\mathcal{T}$. We show that if $p \in \mathcal{P}_{\mathcal{T}}(\alpha\mathcal{E}\alpha', \xi)$ then $p \in \mathcal{P}_{\mathcal{T}}(\alpha\alpha', \xi)$. Let $p_0, p_1, \ldots$ be a generating sequence for p according to $\alpha\mathcal{E}\alpha'$ and ξ. Then it is easy to check that $p_0, p_1, \ldots, p_i, p_{i+2}, \ldots$, where i is the length of α, is a valid generating sequence for p according to $\alpha\alpha'$ and ξ. Indeed, extension $p_i \to p_{i+2}$ is valid, as it corresponds to the first symbol of α' and we have assumed it to be symbol $\mathcal{A}$. This concludes the proof that $\alpha\alpha' \rightsquigarrow \alpha\mathcal{E}\alpha'$.

2. We prove that $\alpha(\mathcal{A})^\omega \rightsquigarrow \alpha\alpha'$. The proof that $\alpha\alpha' \rightsquigarrow \alpha(\mathcal{E})^\omega$ is similar.

   Let ξ be a strategy for tree $\mathcal{T}$ and let p be an infinite path of $\mathcal{T}$. We show that if $p \in \mathcal{P}_{\mathcal{T}}(\alpha(\mathcal{A})^\omega, \xi)$ then $p \in \mathcal{P}_{\mathcal{T}}(\alpha\alpha', \xi)$. Let $p_0, p_1, \ldots$ be a generating sequence for p according to $\alpha\alpha'$ and ξ. Then it is easy to check that $p_0, p_1, \ldots$ is a valid generating sequence for p according to $\alpha(\mathcal{A})^\omega$ and ξ. In fact, $\alpha(\mathcal{A})^\omega$ defines less restrictive conditions on generating sequences than $\alpha\alpha'$.

   This is sufficient to conclude that $\alpha(\mathcal{A})^\omega \rightsquigarrow \alpha\alpha'$. □

We can now complete the picture of Theorem 4: each finite or infinite path quantifier is equivalent to a *canonical path quantifier* that defines a game consisting of alternated moves of players A and E of length one, two, three, or infinity.

Theorem 9 *For each finite or infinite path quantifier α there is a canonical path quantifier*

$$\alpha' \in \{\mathcal{A}, \mathcal{E}, \mathcal{AE}, \mathcal{EA}, \mathcal{AEA}, \mathcal{EAE}, (\mathcal{AE})^\omega, (\mathcal{EA})^\omega\}$$

such that $\alpha \sim \alpha'$. Moreover, the following implications hold between the canonical path quantifiers:

$$\begin{tikzcd}
\mathcal{A} \arrow[r, rightsquigarrow] & \mathcal{AEA} \arrow[r, rightsquigarrow] \arrow[d, rightsquigarrow] & (\mathcal{AE})^\omega \arrow[r, rightsquigarrow] \arrow[d, rightsquigarrow] & \mathcal{AE} \arrow[d, rightsquigarrow] & \\
 & \mathcal{EA} \arrow[r, rightsquigarrow] & (\mathcal{EA})^\omega \arrow[r, rightsquigarrow] & \mathcal{EAE} \arrow[r, rightsquigarrow] & \mathcal{E}
\end{tikzcd} \tag{2}$$

Proof. We first prove that each path quantifier is equivalent to a canonical path quantifier. By Theorem 4, this is true for the finite path quantifiers, so we only consider infinite path quantifiers.

Let α be an infinite path quantifier. We distinguish three cases:

- α contains an infinite number of $\mathcal{A}$ and an infinite number of $\mathcal{E}$: then, by Lemma 7, α is equivalent to one of the canonical games $(\mathcal{AE})^\omega$ or $(\mathcal{EA})^\omega$.

- α contains a finite number of $\mathcal{A}$: in this case, α ends with an infinite sequence of $\mathcal{E}$, and, by Lemma 5, $\alpha \sim \alpha''$ for some finite path quantifier α''. By Theorem 4, α'' is equivalent to some canonical path quantifier, and this concludes the proof for this case.

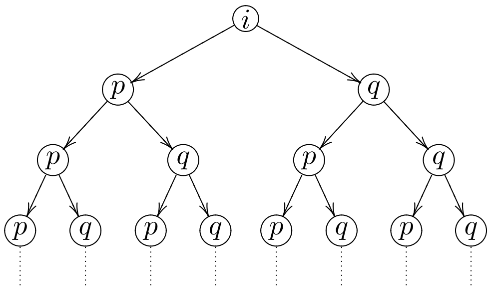
- α contains a finite number of $\mathcal{E}$: this case is similar to the previous one.

For the implications in (2):

- $(\mathcal{A}\mathcal{E})^\omega \rightsquigarrow (\mathcal{E}\mathcal{A})^\omega$ comes from Lemma 8(1), by taking the empty word for α and $\alpha' = (\mathcal{A}\mathcal{E})^\omega$.
- $\mathcal{A}\mathcal{E}\mathcal{A} \rightsquigarrow (\mathcal{A}\mathcal{E})^\omega$, $(\mathcal{A}\mathcal{E})^\omega \rightsquigarrow \mathcal{A}\mathcal{E}$, $\mathcal{E}\mathcal{A} \rightsquigarrow (\mathcal{E}\mathcal{A})^\omega$, and $(\mathcal{E}\mathcal{A})^\omega \rightsquigarrow \mathcal{E}\mathcal{A}\mathcal{E}$ come from Lemmas 5 and 8(2).
- The other implications come from Theorem 4. □

4.3 Strictness of the Implications

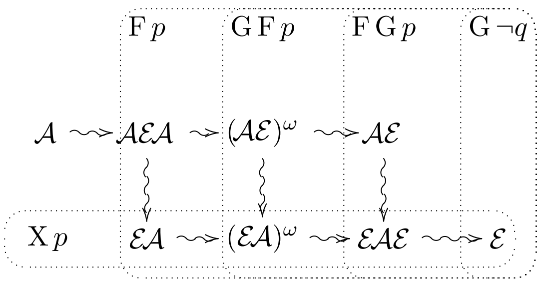
We conclude this section by showing that all the arrows in the diagram of Theorem 9 describe strict implications, namely, the eight canonical path quantifiers are all different. Let us consider the following $\{i, p, q\}$-labelled binary tree, where the root is labelled by i and each node has two children labelled with p and q:



Let us consider the following LTL formulas:

- $\mathrm{F}\,p$: player E can satisfy this formula if he moves at least once, by visiting a p-labelled node.
- $\mathrm{G}\,\mathrm{F}\,p$: player E can satisfy this formula if he can visit an infinite number of p-labelled nodes, that is, if he has the final move in a finite game, or if he moves infinitely often in an infinite game.
- $\mathrm{F}\,\mathrm{G}\,p$: player E can satisfy this formula only if he takes control of the game from a certain point on, that is, only if he has the final move in a finite game.
- $\mathrm{G}\,\neg q$: player E can satisfy this formula only if player A never plays, since player A can immediately visit a q-labelled node.
- $\mathrm{X}\,p$: player E can satisfy this formula by playing the first turn and moving to the left child of the root node.

The following graph shows which formulas hold for which path quantifiers:



5. A Planning Algorithm for $\mathcal{AE}$-LTL

In this section we present a planning algorithm for $\mathcal{AE}$-LTL goals. We start by showing how to build a parity tree automaton that accepts all the trees that satisfy a given $\mathcal{AE}$-LTL formula. Then we show how this tree automaton can be adapted, so that it accepts only trees that correspond to valid plans for a given planning domain. In this way, the problem of checking whether there exists some plan for a given domain and for an $\mathcal{AE}$-LTL goal is reduced to the emptiness problem on tree automata. Finally, we study the complexity of planning for $\mathcal{AE}$-LTL goals and we prove that this problem is 2EXPTIME-complete.

5.1 Tree Automata and $\mathcal{AE}$-LTL Formulas

Berwanger, Grädel, and Kreutzer (2003) have shown that $\mathcal{AE}$-LTL formulas can be expressed directly as CTL* formulas. The reduction exploits the equivalence of expressive power of CTL* and monadic path logic (Moller & Rabinovich, 1999). A tree automaton can be obtained for an $\mathcal{AE}$-LTL formula using this reduction and Theorem 2. However, the translation proposed by Berwanger et al. (2003) has an upper bound of non-elementary complexity, and is hence not useful for our complexity analysis. In this paper we describe a different, more direct reduction that is better suited for our purposes.

A Σ-labelled tree $\mathcal{T}$ satisfies a formula $\alpha.\varphi$ if there is a suitable subset of paths of the tree that satisfy φ. The subset of paths should be chosen according to α. In order to characterize the suitable subsets of paths, we assume to have a w-marking of the tree $\mathcal{T}$, and we use the labels w to define the selected paths.

Definition 17 (w-marking) *A w-marking of the Σ-labelled tree $\mathcal{T}$ is a $(\Sigma\times\{w,\overline{w}\})$-labelled tree $\mathcal{T}_w$ such that* $\mathrm{dom}(\mathcal{T}) = \mathrm{dom}(\mathcal{T}_w)$ *and, whenever $\mathcal{T}(x) = \sigma$, then $\mathcal{T}_w(x) = (\sigma, w)$ or $\mathcal{T}_w(x) = (\sigma, \overline{w})$.*

We exploit w-markings as follows. We associate to each $\mathcal{AE}$-LTL formula $\alpha.\varphi$ a CTL* formula $[[\alpha.\varphi]]$ such that the tree $\mathcal{T}$ satisfies the formula $\alpha.\varphi$ if and only if there is a w-marking of $\mathcal{T}$ that satisfies $[[\alpha.\varphi]]$.

Definition 18 ($\mathcal{AE}$-LTL and CTL*) *Let $\alpha.\varphi$ be an $\mathcal{AE}$-LTL formula. The CTL* formula $[[\alpha.\varphi]]$ is defined as follows:*

$$
\begin{aligned}
[[\mathcal{A}.\varphi]] &= \mathrm{A}\,\varphi \\
[[\mathcal{E}.\varphi]] &= \mathrm{E}\,\varphi \\
[[\mathcal{EA}.\varphi]] &= \mathrm{EF}\,w \wedge \mathrm{A}(\mathrm{F}\,w \rightarrow \varphi) \\
[[\mathcal{AEA}.\varphi]] &= \mathrm{AG}\,\mathrm{EF}\,w \wedge \mathrm{A}(\mathrm{F}\,w \rightarrow \varphi) \\
[[\mathcal{AE}.\varphi]] &= \mathrm{AG}\,\mathrm{EXG}\,w \wedge \mathrm{A}(\mathrm{F}\,\mathrm{G}\,w \rightarrow \varphi) \\
[[\mathcal{EAE}.\varphi]] &= \mathrm{EF}\,\mathrm{AG}\,\mathrm{EXG}\,w \wedge \mathrm{A}(\mathrm{F}\,\mathrm{G}\,w \rightarrow \varphi) \\
[[(\mathcal{AE})^\omega.\varphi]] &= \mathrm{AG}\,\mathrm{EF}\,w \wedge \mathrm{A}(\mathrm{G}\,\mathrm{F}\,w \rightarrow \varphi) \\
[[(\mathcal{EA})^\omega.\varphi]] &= \mathrm{EF}\,\mathrm{AG}\,\mathrm{EF}\,w \wedge \mathrm{A}(\mathrm{G}\,\mathrm{F}\,w \rightarrow \varphi)
\end{aligned}
$$

In the case of path quantifiers $\mathcal{A}$ and $\mathcal{E}$, there is a direct translation into CTL* that does not exploit the w-marking. In the other cases, the CTL* formula $[[\alpha.\varphi]]$ is the conjunction of two sub-formulas. The first one characterizes the good markings according to the path quantifier α, while the second one guarantees that the paths selected according to the marking satisfy the LTL formula φ. In the case of path quantifiers $\mathcal{EA}$ and $\mathcal{AEA}$, we mark with w the nodes that, once reached, guarantee that the formula φ is satisfied. The selected paths are hence those that contain a node labelled by w (formula $\mathrm{F}\,w$). In the case of path quantifiers $\mathcal{AE}$ and $\mathcal{EAE}$, we mark with w all the descendants of a node that define an infinite path that satisfies φ. The selected paths are hence those that, from a certain node on, are continuously labelled by w (formula $\mathrm{F\,G}\,w$). In the case of path quantifiers $(\mathcal{AE})^\omega$ and $(\mathcal{EA})^\omega$, finally, we mark with w all the nodes that player E wants to reach according to its strategy before passing the turn to player A. The selected paths are hence those that contain an infinite number of nodes labelled by w (formula $\mathrm{G\,F}\,w$), that is, the paths along which player E moves infinitely often.

Theorem 10 *A Σ-labelled tree $\mathcal{T}$ satisfies the $\mathcal{AE}$-LTL formula $\alpha.\varphi$ if and only if there is some w-marking of $\mathcal{T}$ that satisfies formula $[[\alpha.\varphi]]$.*

Proof. In the proof, we consider only the cases of $\alpha = \mathcal{AEA}$, $\alpha = \mathcal{AE}$ and $\alpha = (\mathcal{AE})^\omega$. The other cases are similar.
Assume that a tree $\mathcal{T}$ satisfies $\alpha.\varphi$. Then we show that there exists a w-marking $\mathcal{T}_w$ of $\mathcal{T}$ that satisfies $[[\alpha.\varphi]]$.

- **Case** $\alpha = \mathcal{AEA}$**.** According to Definition 13, if the tree $\mathcal{T}$ satisfies $\mathcal{AEA}.\varphi$, then every finite path p of $\mathcal{T}$ can be extended to a finite path p' such that all the infinite extensions p'' of p' satisfy φ. Let us mark with w all the nodes of $\mathcal{T}_w$ that correspond to the extension p' of some path p. By construction, the marked tree satisfies $\mathrm{AG\,EF}\,w$. It remains to show that the marked tree satisfies $\mathrm{A}(\mathrm{F}\,w \to \varphi)$.

  Let us consider any path p'' in the tree that satisfies $\mathrm{F}\,w$, and let us show that p'' also satisfies φ. Since p'' satisfies $\mathrm{F}\,w$, we know that it contains nodes marked with w. Let p' be the finite prefix of path p'' up to the first node marked by w. By construction, there exists a finite path p such that p' is a finite extension of p and all the infinite extensions of p' satisfy φ. As a consequence, also p'' satisfies φ.

- **Case** $\alpha = \mathcal{AE}$**.** According to Definition 13, if the tree $\mathcal{T}$ satisfies $\mathcal{AE}.\varphi$, then for all the finite paths p there is some infinite extension of p that satisfies φ. Therefore, we can define a mapping $m : P^*(\mathcal{T}) \to P^\omega(\mathcal{T})$ that associates to a finite path p an infinite extension $m(p)$ that satisfies φ. We can assume, without loss of generality, that, if p' is a finite extension of p and is also a prefix of $m(p)$, then $m(p') = m(p)$. That is, as far as p' extends the finite path p along the infinite path $m(p)$ then m associates to p' the same infinite path $m(p)$.

  For every finite path p, let us mark with w the node of $\mathcal{T}_w$ that is the child of p along the infinite path $m(p)$. By construction, the marked tree satisfies $\mathrm{AG\,EXG}\,w$. It remains to show that the marked tree satisfies $\mathrm{A}(\mathrm{F\,G}\,w \to \varphi)$.

  Let us consider a path p'' in the tree that satisfies $\mathrm{F\,G}\,w$, and let us show that p'' also satisfies φ. Since p'' satisfies $\mathrm{F\,G}\,w$, we know that there is some path p such that all the descendants of p along p'' are marked with w. In order to prove that p'' satisfies φ

we show that $p'' = m(p)$. Assume by contradiction that $m(p) \neq p''$ and let p' be the longest common prefix of $m(p)$ and p''. We observe that p is a prefix of p', and hence $m(p) = m(p')$. This implies that the child node of p' along p'' is not marked with w, which is absurd, since by definition of p all the descendants of p along p'' are marked with w.

- **Case** $\alpha = (\mathcal{A}\mathcal{E})^\omega$**.** According to Definition 16, if the tree $\mathcal{T}$ satisfies $(\mathcal{A}\mathcal{E})^\omega.\varphi$, then there exists a suitable strategy ξ for player E so that all the possible outcomes of game α with strategy ξ satisfy φ. Let us mark with w all the nodes in $\mathcal{T}_w$ that correspond to the extension $\xi(p)$ of some finite path p. That is, we mark with w all the nodes that are reached after some move of player E according to strategy ξ. The marked tree satisfies the formula $\mathrm{AG}\,\mathrm{EF}\,w$, that is, every finite path p can be extended to a finite path p' such that the node corresponding to p' is marked with w. Indeed, by construction, it is sufficient to take $p' = \xi(p'')$ for some extension p'' of p. It remains to show that the marked tree satisfies $\mathrm{A}(\mathrm{G}\,\mathrm{F}\,w \rightarrow \varphi)$.

  Let us consider a path p in the tree that satisfies $\mathrm{G}\,\mathrm{F}\,w$, and let us show that p also satisfies φ. To this purpose, we show that p is a possible outcome of game α with strategy ξ. We remark that, given an arbitrary finite prefix p' of p it is always possible to find some finite extension p'' of p' such that $\xi(p'')$ is also a prefix of p. Indeed, the set of paths $P = \{\bar{p} : \xi(\bar{p}) \text{ is a finite prefix of } p\}$ is infinite, as there are infinite nodes marked with w in path p.

  Now, let $p_0, p_1, p_2, \ldots$ be the sequence of finite paths defined as follows: $p_0 = (\epsilon)$ is the root of the three; p_{2k+1} is the shortest extension of p_{2k} such that $\xi(p_{2k+1})$ is a prefix of p; and $p_{2k+2} = \xi(p_{2k+1})$. It is easy to check that $p_0, p_1, p_2, \ldots$ is a generating sequence for p according to $(\mathcal{A}\mathcal{E})^\omega$ and ξ. Hence, by Definition 16, the infinite path p satisfies the LTL formula φ.

This concludes the proof that if $\mathcal{T}$ satisfies $\alpha.\varphi$, then there exists a w-marking of $\mathcal{T}$ that satisfies $[[\alpha.\varphi]]$.
Assume now that there is a w-marked tree $\mathcal{T}_w$ that satisfies $[[\alpha.\varphi]]$. We show that $\mathcal{T}$ satisfies $\alpha.\varphi$.

- **Case** $\alpha = \mathcal{A}\mathcal{E}\mathcal{A}$**.** The marked tree satisfies the formula $\mathrm{AG}\,\mathrm{EF}\,w$. This means that for each finite path p (AG) there exists some finite extension p' such that the final node of p' is marked by w ($\mathrm{EF}\,w$) . Let p'' be any infinite extension of such a finite path p'. We show that p'' satisfies the LTL formula φ. Clearly, p'' satisfies the formula $\mathrm{F}\,w$. Since the tree satisfies the formula $\mathrm{A}(\mathrm{F}\,w \rightarrow \varphi)$, all the infinite paths that satisfy $\mathrm{F}\,w$ also satisfy φ. Therefore, p'' satisfies the LTL formula φ.

- **Case** $\alpha = \mathcal{A}\mathcal{E}$**.** The marked tree satisfies the formula $\mathrm{AG}\,\mathrm{EXG}\,w$. Then, for each finite path p (AG) there exists some infinite extension p' such that, from a certain node on, all the nodes of p' are marked with w ($\mathrm{EXG}\,w$). We show that, if p' is the infinite extension of some finite path p, then p' satisfies the LTL formula φ. Clearly, p' satisfies the formula $\mathrm{F}\,\mathrm{G}\,w$. Since the tree satisfies the formula $\mathrm{A}(\mathrm{F}\,\mathrm{G}\,w \rightarrow \varphi)$, all the infinite paths that satisfy $\mathrm{F}\,\mathrm{G}\,w$ also satisfy φ. Therefore, p' satisfies the LTL formula φ.

- **Case $\alpha = (\mathcal{AE})^\omega$.** Let ξ be any strategy so that, for every finite path p, the node corresponding to $\xi(p)$ is marked with w. We remark that it is always possible to define such a strategy. In fact, the marked tree satisfies the formula $\mathrm{AG\,EF}\,w$, and hence, each finite path p can be extended to a finite path p' such that the node corresponding to p' is marked with w.

  Let p be a possible outcome of game α with strategy ξ. We should prove that p satisfies the LTL formula φ. By Definition 16, the infinite path p contains an infinite set of nodes marked by w: these are all the nodes reached after a move of player E. Hence, p satisfies the formula $\mathrm{G\,F}\,w$. Since the tree satisfies the formula $\mathrm{A}(\mathrm{G\,F}\,w \rightarrow \varphi)$, all the infinite paths that satisfy $\mathrm{G\,F}\,w$ also satisfy φ. Therefore, path p satisfies the LTL formula φ.

This concludes the proof that, if there exists a w-marking of tree $\mathcal{T}$ that satisfies $[[\alpha.\varphi]]$, then $\mathcal{T} \models \alpha.\varphi$. $\square$

Kupferman (1999) defines an extension of CTL* with existential quantification over atomic propositions (EGCTL*) and examines complexity of model checking and satisfiability for the new logic. We remark that $\mathcal{AE}$-LTL can be seen as a subset of EGCTL*. Indeed, according to Theorem 10, a Σ-labelled tree satisfies an $\mathcal{AE}$-LTL formula $\alpha.\varphi$ if and only if it satisfies the EGCTL* formula $\exists w.[[\alpha.\varphi]]$.

In the following definition we show how to transform a parity tree automaton for the CTL* formula $[[\alpha.\varphi]]$ into a parity tree automaton for the $\mathcal{AE}$-LTL formula $\alpha.\varphi$. This transformation is performed by abstracting away the information on the w-marking from the input alphabet and from the transition relation of the tree automaton.

Definition 19 *Let $A = \langle \Sigma \times \{w, \overline{w}\}, \mathcal{D}, Q, q_0, \delta, \beta\rangle$ be a parity tree automaton. The parity tree automaton $A_{\exists w} = \langle \Sigma, \mathcal{D}, Q, q_0, \delta_{\exists w}, \beta\rangle$, obtained from A by abstracting away the w-marking, is defined as follows: $\delta_{\exists w}(q, \sigma, d) = \delta(q, (\sigma, w), d) \cup \delta(q, (\sigma, \overline{w}), d)$.*

Lemma 11 *Let A and $A_{\exists w}$ be two parity tree automata as in Definition 19. $A_{\exists w}$ accepts exactly the Σ-labelled trees that have some w-marking which is accepted by A.*

Proof. Let $\mathcal{T}_w$ be a $(\Sigma \times \{w, \overline{w}\})$-labelled tree and let $\mathcal{T}$ be the corresponding Σ-labelled tree, obtained by abstracting away the w-marking. We show that if $\mathcal{T}_w$ is accepted by A, then $\mathcal{T}$ is accepted by $A_{\exists w}$. Let $r : \tau \rightarrow Q$ be an accepting run of $\mathcal{T}_w$ on A. Then r is also an accepting run of $\mathcal{T}$ on $A_{\exists w}$. Indeed, if $x \in \tau$, $\mathit{arity}(x) = d$, and $\mathcal{T}_w(x) = (\sigma, m)$ with $m \in \{w, \overline{w}\}$, then we have $\langle r(x \cdot 0), \ldots, r(x \cdot d{-}1)\rangle \in \delta(r(x), (\sigma, m), d)$. Then $\mathcal{T}(x) = \sigma$, and, by definition of $A_{\exists w}$, we have $\langle r(x \cdot 0), \ldots, r(x \cdot d{-}1)\rangle \in \delta_{\exists w}(r(x), \sigma, d)$.
Now we show that, if the Σ-labelled tree $\mathcal{T}$ is accepted by $A_{\exists w}$, then there is a $(\Sigma \times \{w, \overline{w}\})$-labelled tree $\mathcal{T}_w$ that is a w-marking of $\mathcal{T}$ and that is accepted by A. Let $r : \tau \rightarrow Q$ be an accepting run of $\mathcal{T}$ on $A_{\exists w}$. By definition of run, we know that if $x \in \tau$, with $\mathit{arity}(x) = d$ and $\mathcal{T}(x) = \sigma$, then $\langle r(x \cdot 0), \ldots, r(x \cdot d{-}1)\rangle \in \delta_{\exists w}(r(x), \sigma, d)$. By definition of $\delta_{\exists w}$, we know that $\langle r(x \cdot 0), \ldots, r(x \cdot d{-}1)\rangle \in \delta(r(x), (\sigma, w), d) \cup \delta(r(x), (\sigma, \overline{w}), d)$. Let us define $\mathcal{T}_w(x) = (\sigma, w)$ if $\langle r(x \cdot 0), \ldots, r(x \cdot d{-}1)\rangle \in \delta(r(x), (\sigma, w), d)$, and $\mathcal{T}_w(x) = (\sigma, \overline{w})$ otherwise. It is easy to check that r is an accepting run of $\mathcal{T}_w$ on A. $\square$

Now we have all the ingredients for defining the tree automaton that accepts all the trees that satisfy a given $\mathcal{AE}$-LTL formula.

Definition 20 (tree automaton for $\mathcal{AE}$-LTL) *Let $\mathcal{D} \subseteq \mathbb{N}^*$ be a finite set of arities, and let $\alpha.\varphi$ be an $\mathcal{AE}$-LTL formula. The parity tree automaton $A^{\mathcal{D}}_{\alpha.\varphi}$ is obtained by applying the transformation described in Definition 19 to the parity automaton $A^{\mathcal{D}}_{[[\alpha.\varphi]]}$ built according to Theorem 2.*

Theorem 12 *The parity tree automaton $A^{\mathcal{D}}_{\alpha.\varphi}$ accepts exactly the Σ-labelled $\mathcal{D}$-trees that satisfy the formula $\alpha.\varphi$.*

Proof. By Theorem 2, the parity tree automaton $A^{\mathcal{D}}_{[[\alpha.\varphi]]}$ accepts all the $\mathcal{D}$-trees that satisfy the CTL* formula $[[\alpha.\varphi]]$. Therefore, the parity tree automaton $A^{\mathcal{D}}_{\alpha.\varphi}$ accepts all the $\mathcal{D}$-trees that satisfy the formula $\alpha.\varphi$ by Lemma 11 and Theorem 10. $\square$

The parity tree automaton $A^{\mathcal{D}}_{\alpha.\varphi}$ has a parity index that is exponential and a number of states that is doubly exponential in the length of formula φ.

Proposition 13 *The parity tree automaton $A^{\mathcal{D}}_{\alpha.\varphi}$ has $2^{2^{O(|\varphi|)}}$ states and parity index $2^{O(|\varphi|)}$.*

Proof. The construction of Definition 19 does not change the number of states and the parity index of the automaton. Therefore, the proposition follows from Theorem 2. $\square$

5.2 The Planning Algorithm

We now describe how the automaton $A^{\mathcal{D}}_{\alpha.\varphi}$ can be exploited in order to build a plan for goal $\alpha.\varphi$ on a given domain.

We start by defining a tree automaton that accepts all the trees that define the valid plans of a planning domain $D = \langle \Sigma, \sigma_0, A, R \rangle$. We recall that, according to Definition 8, transition relation R maps a state $\sigma \in \Sigma$ and an action $a \in A$ into a tuple of next states $\langle \sigma_1, \sigma_2, \ldots, \sigma_n \rangle = R(\sigma, a)$.

In the following we assume that $\mathcal{D}$ is a finite set of arities that is compatible with domain D, namely, if $R(\sigma, a) = \langle \sigma_1, \ldots, \sigma_d \rangle$ for some $\sigma \in \Sigma$ and $a \in A$, then $d \in \mathcal{D}$.

Definition 21 (tree automaton for a planning domain) *Let $D = \langle \Sigma, \sigma_0, A, R \rangle$ be a planning domain and let $\mathcal{D}$ be a set of arities that is compatible with domain D. The tree automaton $A^{\mathcal{D}}_D$ corresponding to the planning domain is $A^{\mathcal{D}}_D = \langle \Sigma \times A, \mathcal{D}, \Sigma, \sigma_0, \delta_D, \beta_0 \rangle$, where $\langle \sigma_1, \ldots, \sigma_d \rangle \in \delta_D(\sigma, (\sigma, a), d)$ if $\langle \sigma_1, \ldots, \sigma_d \rangle = R(\sigma, a)$ with $d > 0$, and $\beta_0(\sigma) = 0$ for all $\sigma \in \Sigma$.*

According to Definition 10, a $(\Sigma \times A)$-labelled tree can be obtained from each plan π for domain D. Now we show that also the converse is true, namely, each $(\Sigma \times A)$-labelled tree accepted by the tree automaton $A^{\mathcal{D}}_D$ induces a plan.

Definition 22 (plan induced by a tree) *Let $\mathcal{T}$ be a $(\Sigma \times A)$-labelled tree that is accepted by automaton $A^{\mathcal{D}}_D$. The* plan π induced by $\mathcal{T}$ *on domain D is defined as follows: $\pi(\sigma_0, \sigma_1, \ldots, \sigma_n) = a$ if there is some finite path p in $\mathcal{T}$ with $\mathcal{T}(p) = (\sigma_0, a_0) \cdot (\sigma_1, a_1) \cdots (\sigma_n, a_n)$ and $a = a_n$.*

The following lemma shows that Definitions 10 and 22 define a one-to-one correspondence between the valid plans for a planning domain D and the trees accepted by automaton $A^{\mathcal{D}}_{D}$.

Lemma 14 *Let $\mathcal{T}$ be a tree accepted by automaton $A^{\mathcal{D}}_{D}$ and let π be the corresponding induced plan. Then π is a valid plan for domain D, and $\mathcal{T}$ is the execution tree corresponding to π. Conversely, let π be a plan for domain D and let $\mathcal{T}$ be the corresponding execution structure. Then $\mathcal{T}$ is accepted by automaton $A^{\mathcal{D}}_{D}$ and π is the plan induced by $\mathcal{T}$.*

Proof. This lemma is a direct consequence of Definitions 10 and 22. □

We now define a parity tree automaton that accepts only the trees that correspond to the plans for domain D and that satisfy goal $g = \alpha.\varphi$. This parity tree automaton is obtained by combining in a suitable way the tree automaton for $\mathcal{AE}$-LTL formula g (Definition 20) and the tree automaton for domain D (Definition 21).

Definition 23 (instrumented tree automaton) *Let $\mathcal{D}$ be a set of arities that is compatible with planning domain D. Let also $A^{\mathcal{D}}_{g} = \langle \Sigma, \mathcal{D}, Q, q_0, \delta, \beta \rangle$ be a parity tree automaton that accepts only the trees that satisfy the $\mathcal{AE}$-LTL formula g. The parity tree automaton $A^{\mathcal{D}}_{D,g}$ corresponding to planning domain D and goal g is defined as follows: $A^{\mathcal{D}}_{D,g} = \langle \Sigma{\times}A, \mathcal{D}, Q{\times}\Sigma, (q_0, \sigma_0), \delta', \beta' \rangle$, where $\langle (q_1, \sigma_1), \ldots, (q_d, \sigma_d) \rangle \in \delta'((q, \sigma), (\sigma, a), d)$ if $\langle q_1, \ldots, q_d \rangle \in \delta(q, \sigma, d)$ and $\langle \sigma_1, \ldots, \sigma_d \rangle = R(\sigma, a)$ with $d > 0$, and where $\beta'(q, \sigma) = \beta(q)$.*

The following lemmas show that solutions to planning problem (D, g) are in one-to-one correspondence with the trees accepted by the tree automaton $A^{\mathcal{D}}_{D,g}$.

Lemma 15 *Let $\mathcal{T}$ be a $(\Sigma{\times}A)$-labelled tree that is accepted by automaton $A^{\mathcal{D}}_{D,g}$, and let π be the plan induced by $\mathcal{T}$ on domain D. Then the plan π is a solution to planning problem (D, g).*

Proof. According to Definition 11, we have to prove that the execution tree corresponding to π satisfies the goal g. By Lemma 14, this amounts to proving that the tree $\mathcal{T}$ satisfies g. By construction, it is easy to check that if a $(\Sigma{\times}A)$-labeled tree $\mathcal{T}$ is accepted by $A^{\mathcal{D}}_{D,g}$, then it is also accepted by $A^{\mathcal{D}}_{g}$. Indeed, if $r_{D,g} : \tau \to Q \times \Sigma$ is an accepting run of $\mathcal{T}$ on $A^{\mathcal{D}}_{D,g}$, then $r_g : \tau \to Q$ is an accepting run of $\mathcal{T}$ on $A^{\mathcal{D}}_{g}$, where $r_g(x) = q$ whenever $r_{D,g} = (q, \sigma)$ for some $\sigma \in \Sigma$. □

Lemma 16 *Let π be a solution to planning problem (D, g). Then the execution tree of π is accepted by automaton $A^{\mathcal{D}}_{D,g}$.*

Proof. Let $\mathcal{T}$ be the execution tree of π. By Lemma 14 we know that $\mathcal{T}$ is accepted by $A^{\mathcal{D}}_{D}$. Moreover, by definition of solution of a planning problem, we know that $\mathcal{T}$ is accepted also by $A^{\mathcal{D}}_{g}$. By construction, it is easy to check that if a $(\Sigma{\times}A)$-labeled tree $\mathcal{T}$ is accepted by $A^{\mathcal{D}}_{D}$ and by $A^{\mathcal{D}}_{g}$, then it is also accepted by $A^{\mathcal{D}}_{D,g}$. Indeed, let $r_D : \tau \to \Sigma$ be an accepting run of $\mathcal{T}$ on $A^{\mathcal{D}}_{D}$ and let $r_g : \tau \to Q$ be an accepting run of $\mathcal{T}$ on $A^{\mathcal{D}}_{g}$. Then $r_{D,g} : \tau \to Q \times \Sigma$ is an accepting run of $\mathcal{T}$ on $A^{\mathcal{D}}_{D,g}$, where $r_{D,g}(x) = (q, \sigma)$ if $r_D(x) = \sigma$ and $r_g(x) = q$. □

As a consequence, checking whether goal g can be satisfied on domain D is reduced to the problem of checking whether automaton $A^{\mathcal{D}}_{D,g}$ is nonempty.

Theorem 17 *Let D be a planning domain and g be an $\mathcal{AE}$-LTL formula. A plan exists for goal g on domain D if and only if the tree automaton $A^{\mathcal{D}}_{D,g}$ is nonempty.*

Proposition 18 *The parity tree automaton $A^{\mathcal{D}}_{D,g}$ for domain $D = (\Sigma, \sigma_0, A, R)$ and goal $g = \alpha.\varphi$ has $|\Sigma| \cdot 2^{2^{O(|\varphi|)}}$ states and parity index $2^{O(|\varphi|)}$.*

Proof. This is a consequence of Proposition 13 and of the definition of automaton $A^{\mathcal{D}}_{D,g}$. $\square$

5.3 Complexity

We now study the time complexity of the planning algorithm defined in Subsection 5.2.

Given a planning domain D, the planning problem for $\mathcal{AE}$-LTL goals $g = \alpha.\varphi$ can be decided in a time that is doubly exponential in the size of the formula φ by applying Theorem 1 to the tree automaton $A^{\mathcal{D}}_{D,g}$.

Lemma 19 *Let D be a planning domain. The existence of a plan for $\mathcal{AE}$-LTL goal $g = \alpha.\varphi$ on domain D can be decided in time $2^{2^{O(|\varphi|)}}$.*

Proof. By Theorem 17 the existence of a plan for goal g on domain D is reduced to the emptiness problem on parity tree automaton $A^{\mathcal{D}}_{D,g}$. By Proposition 18, the parity tree automaton $A^{\mathcal{D}}_{D,g}$ has $2^{2^{O(|\varphi|)}} \times |\Sigma|$ states and parity index $2^{O(|\varphi|)}$. Since we assume that domain D is fixed, by Theorem 1, the emptiness of automaton $A^{\mathcal{D}}_{D,g}$ can be decided in time $2^{2^{O(|\varphi|)}}$. $\square$

The doubly exponential time bound is tight. Indeed, the *realizability problem* for an LTL formula φ, which is known to be 2EXPTIME-complete (Pnueli & Rosner, 1990), can be reduced to a planning problem for the goal $\mathcal{A}.\varphi$. In a realizability problem one assumes that a program and the environment alternate in the control of the evolution of the system. More precisely, in an execution $\sigma_0, \sigma_1, \ldots$ the states σ_i are decided by the program if i is even, and by the environment if i is odd. We say that a given formula φ is realizable if there is some program such that all its executions satisfy φ independently on the actions of the environment.

Theorem 20 *Let D be a planning domain. The problem of deciding the existence of a plan for $\mathcal{AE}$-LTL goal $g = \alpha.\varphi$ on domain D is 2EXPTIME-complete.*

Proof. The realizability of formula φ can be reduced to the problem of checking the existence of a plan for goal $\mathcal{A}.\varphi$ on planning domain $D = \big(\{init\} \cup (\Sigma \times \{p, e\}), init, \Sigma \cup \{e\}, R\big)$, with:

$$\begin{aligned} R(init, \sigma') &= \{(\sigma', e)\} & R(init, e) &= \emptyset \\ R((\sigma, p), \sigma') &= \{(\sigma', e)\} & R((\sigma, p), e) &= \emptyset \\ R((\sigma, e), \sigma') &= \emptyset & R((\sigma, e), e) &= \{(\sigma', p) : \sigma' \in \Sigma\} \end{aligned}$$

for all $\sigma, \sigma' \in \Sigma$.
States (σ, p) are those where the program controls the evolution through actions $\sigma' \in \Sigma$. States (σ, e) are those where the environment controls the evolution; only the nondeterministic action e can be performed in this state. Finally, state *init* is used to assign the initial move to the program.
Since the realizability problem is 2EXPTIME-complete in the size of the LTL formula (Pnueli & Rosner, 1990), the planning problem is 2EXPTIME-hard in the size of the goal $g = \alpha.\varphi$. The 2EXPTIME-completeness follows from Lemma 19. $\square$

We remark that, in the case of goals of the form $\mathcal{E}.\varphi$, an algorithm with a better complexity can be defined. In this case, a plan exists for $\mathcal{E}.\varphi$ if and only if there is an infinite sequence $\sigma_0, \sigma_1, \ldots$ of states that satisfies φ and such that $\sigma_{i+1} \in R(\sigma_i, a_i)$ for some action a_i. That is, the planning problem can be reduced to a model checking problem for LTL formula φ, and this problem is known to be PSPACE-complete (Sistla & Clarke, 1985). We conjecture that, for all the canonical path quantifiers α except $\mathcal{E}$, the doubly exponential bound of Theorem 20 is tight.

Some remarks are in order on the complexity of the *satisfiability* and *validity problems* for $\mathcal{AE}$-LTL goals. These problems are PSPACE-complete. Indeed, the $\mathcal{AE}$-LTL formula $\alpha.\varphi$ is satisfiable if and only if the LTL formula φ is satisfiable[6], and the latter problem is known to be PSPACE-complete (Sistla & Clarke, 1985). A similar argument holds also for validity.

The complexity of the *model checking problem* for $\mathcal{AE}$-LTL has been recently addressed by Kupferman and Vardi (2006). Kupferman and Vardi introduce mCTL*, a variant of CTL*, where path quantifiers have a "memoryful" interpretation. They show that memoryful quantification can express (with linear cost) the semantics of path quantifiers in our $\mathcal{AE}$-LTL. For example, the $\mathcal{AE}$-LTL formula $\mathcal{AE}.\varphi$ is expressed in mCTL* by the formula $\mathrm{AG}\,\mathrm{E}\,\varphi$. Kupferman and Vardi show that the model checking problem for the new logic is EXPSPACE-complete, and that this result holds also for the subset of mCTL* that corresponds to formulas $\mathcal{AE}.\varphi$. Therefore, the model checking problem for $\mathcal{AE}$-LTL with finite path quantifiers is also EXPSPACE-complete. To the best of our knowledge the complexity of model checking $\mathcal{AE}$-LTL formulas $(\mathcal{AE})^\omega.\varphi$ and $(\mathcal{EA})^\omega.\varphi$ is still an open problem.

6. Two Specific Cases: Reachability and Maintainability Goals

In this section we consider two basic classes of goals that are particularly relevant in the field of planning.

6.1 Reachability Goals

The first class of goals are the *reachability goals* corresponding to the LTL formula $\mathrm{F}\,q$, where q is a propositional formula. Most of the literature in planning concentrates on this class of goals, and there are several works that address the problem of defining plans of different strength for this kind of goals (see, e.g., Cimatti et al., 2003 and their citations).

6. If a tree satisfies $\alpha.\varphi$ then some of its paths satisfy φ, and a path that satisfies φ can be seen also as a tree that satisfies $\alpha.\varphi$.

In the context of $\mathcal{AE}$-LTL, as soon as player E takes control, it can immediately achieve the reachability goal if possible at all. The fact that the control is given back to player A after the goal has been achieved is irrelevant. Therefore, the only significant path quantifiers for reachability goals are $\mathcal{A}$, $\mathcal{E}$, and $\mathcal{AE}$.

Proposition 21 *Let q be a propositional formula on atomic propositions Prop. Then, the following results hold for every labelled tree $\mathcal{T}$. $\mathcal{T} \models \mathcal{E}.\,\mathrm{F}\,q$ iff $\mathcal{T} \models \mathcal{EA}.\,\mathrm{F}\,q$ iff $\mathcal{T} \models \mathcal{EAE}.\,\mathrm{F}\,q$ iff $\mathcal{T} \models (\mathcal{EA})^\omega.\,\mathrm{F}\,q$. Moreover $\mathcal{T} \models \mathcal{AE}.\,\mathrm{F}\,q$ iff $\mathcal{T} \models \mathcal{AEA}.\,\mathrm{F}\,q$ iff $\mathcal{T} \models (\mathcal{AE})^\omega.\,\mathrm{F}\,q$.*

Proof. We prove that $\mathcal{T} \models \mathcal{AE}.\,\mathrm{F}\,q$ iff $\mathcal{T} \models \mathcal{AEA}.\,\mathrm{F}\,q$ iff $\mathcal{T} \models (\mathcal{AE})^\omega.\,\mathrm{F}\,q$. The other cases are similar.
Let us assume that $\mathcal{T} \models \mathcal{AE}.\,\mathrm{F}\,q$. Moreover, let p be a finite path of $\mathcal{T}$. We know that p can be extended to an infinite path p' such that $\mathcal{T}(p') \models \mathrm{F}\,q$. According to the semantics of LTL, $\mathcal{T}(p') \models \mathrm{F}\,q$ means that there is some node x in path p' such that $q \in \mathcal{T}(x)$. Clearly, all infinite paths of $\mathcal{T}$ that contain node x also satisfy the LTL formula $\mathrm{F}\,q$. Therefore, there is a finite extension p'' of p such that all the infinite extensions of p'' satisfy the LTL formula $\mathrm{F}\,q$: it is sufficient to take as p'' an finite extension of p that contains node x. Since this property holds for every finite path p, we conclude that $\mathcal{T} \models \mathcal{AEA}.\,\mathrm{F}\,q$.
We have proven that $\mathcal{T} \models \mathcal{AE}.\,\mathrm{F}\,q$ implies $\mathcal{T} \models \mathcal{AEA}.\,\mathrm{F}\,q$. By Theorem 9 we know that $\mathcal{AEA} \rightsquigarrow (\mathcal{AE})^\omega \rightsquigarrow \mathcal{AE}$, and hence $\mathcal{T} \models \mathcal{AEA}.\,\mathrm{F}\,q$ implies $\mathcal{T} \models (\mathcal{AE})^\omega.\,\mathrm{F}\,q$ implies $\mathcal{T} \models \mathcal{AE}.\,\mathrm{F}\,q$. This concludes the proof. $\square$

The following diagram shows the implications among the significant path quantifiers for reachability goals:

$$\mathcal{A} \rightsquigarrow \mathcal{AE} \rightsquigarrow \mathcal{E} \tag{3}$$

We remark that the three goals $\mathcal{A}.\,\mathrm{F}\,q$, $\mathcal{E}.\,\mathrm{F}\,q$, and $\mathcal{AE}.\,\mathrm{F}\,q$ correspond, respectively, to the strong, weak, and strong cyclic planning problems of Cimatti et al. (2003).

6.2 Maintainability Goals

We now consider another particular case, namely the maintainability goals $\mathrm{G}\,q$, where q is a propositional formula. Maintainability goals have properties that are complementary to the properties of reachability goals. In this case, as soon as player A takes control, it can violate the maintainability goal if possible at all. The fact that player E can take control after player A is hence irrelevant, and the only interesting path quantifiers are $\mathcal{A}$, $\mathcal{E}$, and $\mathcal{EA}$.

Proposition 22 *Let q be a propositional formula on atomic propositions Prop. Then, the following results hold for every labelled tree $\mathcal{T}$. Then $\mathcal{T} \models \mathcal{A}.\,\mathrm{G}\,q$ iff $\mathcal{T} \models \mathcal{AE}.\,\mathrm{G}\,q$ iff $\mathcal{T} \models \mathcal{AEA}.\,\mathrm{G}\,q$ iff $\mathcal{T} \models (\mathcal{AE})^\omega.\,\mathrm{G}\,q$. Moreover $\mathcal{T} \models \mathcal{EA}.\,\mathrm{G}\,q$ iff $\mathcal{T} \models \mathcal{EAE}.\,\mathrm{G}\,q$ iff $\mathcal{T} \models (\mathcal{EA})^\omega.\,\mathrm{G}\,q$.*

Proof. The proof is similar to the proof of Proposition 21. $\square$

The following diagram shows the implications among the significant path quantifiers for maintainability goals:

$$\mathcal{A} \rightsquigarrow \mathcal{EA} \rightsquigarrow \mathcal{E}$$

The goals $\mathcal{A}.\,\mathrm{G}\,q$, $\mathcal{E}.\,\mathrm{G}\,q$, and $\mathcal{EA}.\,\mathrm{G}\,q$ correspond to maintainability variants of strong, weak, and strong cyclic planning problems. Indeed, they correspond to requiring that condition q is maintained for all evolutions despite nondeterminism ($\mathcal{A}.\,\mathrm{G}\,q$), that condition q is maintained for some of the evolutions ($\mathcal{E}.\,\mathrm{G}\,q$), and that it is possible to reach a state where condition q is always maintained despite nondeterminism ($\mathcal{EA}.\,\mathrm{G}\,p$).

7. Related Works and Concluding Remarks

In this paper we have defined $\mathcal{AE}$-LTL, a new temporal logic that extends LTL with the possibility of declaring complex path quantifiers that define the different degrees in which an LTL formula can be satisfied by a computation tree. We propose to use $\mathcal{AE}$-LTL formulas for expressing temporally extended goals in nondeterministic planning domains. We have defined a planning algorithm for $\mathcal{AE}$-LTL goals that is based on an automata-theoretic framework: the existence of a plan is reduced to checking the emptiness of a suitable parity tree automaton. We have studied the time complexity of the planning algorithm, proving that it is 2EXPTIME-complete in the length of the $\mathcal{AE}$-LTL formula.

In the field of planning, several works use temporal logics for defining goals. Most of these approaches (Bacchus & Kabanza, 1998, 2000; Calvanese et al., 2002; Cerrito & Mayer, 1998; de Giacomo & Vardi, 1999; Kvarnström & Doherty, 2001) use linear temporal logics as the goal language, and are not able to express conditions on the degree in which the goal should be satisfied with respect to the nondeterminism in the execution. Notable exceptions are the works described by Pistore, Bettin, and Traverso (2001), Pistore and Traverso (2001) and by Dal Lago et al. (2002). Pistore et al. (2001) and Pistore and Traverso (2001) use CTL as goal language, while Dal Lago et al. (2002) define a new branching time logic that allows for expressing temporally extended goals that can deal explicitly with failure and recovery in goal achievement. In these goal languages, however, path quantifiers are interleaved with the temporal operators, and are hence rather different from $\mathcal{AE}$-LTL.

In the field of temporal logics, the work on alternating temporal logic (ATL) (Alur, Henzinger, & Kupferman, 2002) is related to our work. In ATL, the path quantifiers in CTL and CTL* are replaced by game quantifiers. Nevertheless, there is no obvious way to expressed formulas of the form $\alpha.\varphi$, where α is a path quantifier and φ is an LTL formula in ATL*, which is the most expressive logic studied by Alur et al. (2002). Our conjecture is that our logic and ATL* are of incomparable expressiveness.

Some comments are in order on the practical impact of the 2EXPTIME complexity of the planning algorithm. First of all, in many planning problems we expect to have very complex and large domains, but goals that are relatively simple (see, e.g., the experimental evaluation performed by Pistore et al. (2001) in the case of planning goals expressed as CTL formulas). In these cases, the doubly exponential complexity of the algorithm in the size of the formula may not be a bottleneck. For larger $\mathcal{AE}$-LTL goals, a doubly exponential time complexity may not be feasible, but it should be noted that this is *worst-case* complexity. We also note that improved algorithms for plan synthesis is an active research area, including the analysis of simpler LTL goals (Alur & La Torre, 2004) and the development of improved automata-theoretic algorithms (Kupferman & Vardi, 2005).

The automata-theoretic framework that we have used in the paper is of wider applicability than $\mathcal{AE}$-LTL goals. An interesting direction for future investigations is the application of the framework to variants of $\mathcal{AE}$-LTL that allow for nesting of path quantifiers, or for goals that combine $\mathcal{AE}$-LTL with propositional or temporal operators. This would allow, for instance, to specify goals which compose requirements of different strength. A simple example of such goals is $(\mathcal{AE}.\,\mathrm{F}\,p) \wedge (\mathcal{A}.\,\mathrm{G}\,p)$, which requires to achieve condition p in a strong cyclic way, maintaining condition q in a strong way. The impossibility to define such kind of goals is, in our opinion, the strongest limitation of $\mathcal{AE}$-LTL with respect to CTL and CTL*.

Another direction for future investigations is the extension of the approach proposed in this paper to the case of planning under partial observability (de Giacomo & Vardi, 1999), where one assumes that the agent executing the plan can observe only part of the state and hence its choices on the actions to execute may depend only on that part.

We also plan to explore implementation issues and, in particular, the possibility of exploiting BDD-based symbolic techniques in a planning algorithm for $\mathcal{AE}$-LTL goals. In some cases, these techniques have shown to be able to deal effectively with domains and goals of a significant complexity, despite the exponential worst-case time complexity of the problems (Bertoli, Cimatti, Pistore, Roveri, & Traverso, 2001; Pistore et al., 2001).

Acknowledgments

A shorter version of this paper, without proofs, has been published by Pistore and Vardi (2003). The authors would like to thank Erich Grädel for his comments on the reduction of $\mathcal{AE}$-LTL formulas to CTL* formulas.

References

Alur, R., Henzinger, T., & Kupferman, O. (2002). Alternating-time temporal logic. *Journal of the ACM*, *49*(5), 672–713.

Alur, R., & La Torre, S. (2004). Deterministic generators and games for LTL fragments. *ACM Trans. Comput. Log.*, *5*(1), 1–25.

Bacchus, F., & Kabanza, F. (1998). Planning for temporally extended goals. *Ann. of Mathematics and Artificial Intelligence*, *22*, 5–27.

Bacchus, F., & Kabanza, F. (2000). Using temporal logic to express search control knowledge for planning. *Artificial Intelligence*, *116*(1-2), 123–191.

Bertoli, P., Cimatti, A., Pistore, M., Roveri, M., & Traverso, P. (2001). MBP: A Model Based Planner. In *Proc. of IJCAI'01 workshop on Planning under Uncertainty and Incomplete Information.*

Berwanger, D., Grädel, E., & Kreutzer, S. (2003). Once upon a time in the West - Determinacy, definability, and complexity of path games. In *Prof. of 10th Int. Conf on Logic for Programming, Artificial Intelligence, and Reasoning (LPAR'03)*, pp. 229–243.

Calvanese, D., de Giacomo, G., & Vardi, M. (2002). Reasoning about actions and planning in LTL action theories. In *Proc. of 8th Int. Conf. on the Principles of Knowledge Representation and Reasoning (KR'02)*, pp. 593–602.

Cerrito, S., & Mayer, M. (1998). Bounded model search in linear temporal logic and its application to planning. In *Proc. of 2nd Int. Conf. on Analytic Tableaux and Related Methods (TABLEAUX'98)*, Vol. 1397 of *LNAI*, pp. 124–140. Springer Verlag.

Cimatti, A., Pistore, M., Roveri, M., & Traverso, P. (2003). Weak, strong, and strong cyclic planning via symbolic model checking.. *Artificial Intelligence*, *147*(1-2), 35–84.

Dal Lago, U., Pistore, M., & Traverso, P. (2002). Planning with a language for extended goals. In *Proc. of 18th National Conf. on Artificial Intelligence (AAAI'02)*. AAAI Press.

Daniele, M., Traverso, P., & Vardi, M. (1999). Strong cyclic planning revisited. In *Proc. of 5th European Conf. in Planning (ECP'99)*, Vol. 1809 of *LNAI*, pp. 35–48. Springer Verlag.

de Giacomo, G., & Vardi, M. (1999). Automata-theoretic approach to planning with temporally extended goals. In *Proc. of 5th European Conf. in Planning (ECP'99)*, Vol. 1809 of *LNAI*, pp. 226–238. Springer Verlag.

Emerson, E. A. (1990). Temporal and modal logic. In van Leeuwen, J. (Ed.), *Handbook of Theoretical Computer Science, Volume B: Formal Models and Semantics*. Elsevier.

Emerson, E., & Jutla, C. (1988). The complexity of tree automata and logics of programs. In *Proc. of 29th IEEE Symp. on Foundations of Computer Science*, pp. 328–337.

Emerson, E., & Jutla, C. (1991). Tree automata, μ-calculus and determinacy. In *Proc. of 32nd IEEE Symp. on Foundations of Computer Science*, pp. 368–377.

Fikes, R., & Nilsson, N. (1971). STRIPS: A new approach to the application of theorem proving to problem solving. *Artificial Intelligence*, *2*(3-4), 189–208.

Ghallab, M., Nau, D., & Traverso, P. (2004). *Automated Planning: Theory and Practice*. Elsevier.

Kupferman, O. (1999). Augmenting branching temporal logics with existential quantification over atomic propositions. *Journal of Logic and Computation*, *9*(2), 135–147.

Kupferman, O., & Vardi, M. (2005). Safraless decision procedures. In *Proc. of 46th IEEE Symp. on Foundations of Computer Science (FOCS'05)*, pp. 531–542. IEEE Computer Society.

Kupferman, O., & Vardi, M. (2006). Memoryful branching-time logic. In *Proc. of the 21th IEEE Symposium on Logic in Computer Science (LICS 2006)*. IEEE Computer Society.

Kupferman, O., Vardi, M., & Wolper, P. (2000). An automata-theoretic approach to branching time model checking. *Journal of the ACM*, *47*(2).

Kvarnström, J., & Doherty, P. (2001). TALplanner: A temporal logic based forward chaining planner. *Ann. of Mathematics and Artificial Intelligence*, *30*, 119–169.

Moller, F., & Rabinovich, A. (1999). On the expressive power of CTL*. In *Proc. of 14th Annual IEEE Symposium on Logic in Computer Science (LICS'99)*, pp. 360–369. IEEE Computer Science Press.

Penberthy, J., & Weld, D. (1992). UCPOP: A sound, complete, partial order planner for ADL. In *Proc. of 3rd Int. Conf. on the Principles of Knowledge Representation and Reasoning (KR'92)*.

Peot, M., & Smith, D. (1992). Conditional nonlinear planning. In *Proc. of 1st Int. Conf. on AI Planning Systems (AIPS'92)*, pp. 189–197. Morgan Kaufmann Publisher.

Pistore, M., Bettin, R., & Traverso, P. (2001). Symbolic techniques for planning with extended goals in non-deterministic domains. In *Proc. of 6th European Conf. in Planning (ECP'01)*.

Pistore, M., & Traverso, P. (2001). Planning as model checking for extended goals in non-deterministic domains. In *Proc. of 17th Int. Joint Conf. on Artificial Intelligence (IJCAI'01)*. AAAI Press.

Pistore, M., & Vardi, M. (2003). The planning specturm — one, two, three, infinity. In *Proc. of the 18th IEEE Symposium on Logic in Computer Science (LICS 2003)*, pp. 234–243.

Pnueli, A., & Rosner, R. (1990). Distributed reactive systems are hard to synthesize. In *Proc. of 31st IEEE Symp. on Foundation of Computer Science*, pp. 746–757.

Sistla, A., & Clarke, E. (1985). The complexity of propositional linear temporal logic. *Journal ACM*, *32*, 733–749.

Warren, D. (1976). Generating conditional plans and programs. In *Proc. of the Summer Conf. on Artificial Intelligence and Simulation of Behaviour (AISB'76)*, pp. 344–354.