The Planning Spectrum — One, Two, Three, Infinity

Marco Pistore*
University of Trento
pistore@dit.unitn.it

Moshe Y. Vardi†
Rice University
vardi@cs.rice.edu

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 to satisfy the formula for all the possible execution paths, and the* existential *interpretation $\mathcal{E}.\varphi$, which requires to satisfy the formula for some execution paths. 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 executions. 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 presents a planning algorithm for the new language, that is based on an automata-theoretic approach, and studies its complexity.*

1. Introduction

In automated task planning [16, 21], 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 languages, 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 (see, e.g., [2, 3, 6, 7, 10, 12, 19, 24]). Most of these approaches use *Linear Temporal Logic* (LTL) [15] 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* [9, 22, 27], 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 (e.g., robotics, autonomous controllers, etc.). 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 paths. 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 interpret the formula in an universal way, as $\mathcal{A}.\varphi$, while weak planning corresponds to interpret it in an existential way, as $\mathcal{E}.\varphi$.

Weak and strong plans are two very extreme ways of satisfiability of an LTL formula. In practical applications, it

*Supported in part by ASI project DOVES.
†Supported in part by NSF grants CCR-9988322, CCR-0124077, IIS-9908435, IIS-9978135, and EIA-0086264, by BSF grant 9800096, and by a grant from the Intel Corporation.

Proceedings of the 18th Annual IEEE Symposium on Logic in Computer Science (LICS'03)
1043-6871/03 $17.00 © 2003 IEEE

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* [8, 11] 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, [11] shows 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 to generic temporal goals the encoding of strong cyclic planning proposed in [11].

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. 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-players 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. During its turns, each player controls the outcome of action execution for a finite number of actions, and then passes the control to the other player.[1] 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 to 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 [13, 18]: 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 $\mathcal{A}.\varphi$, 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, on planning, and on temporal logics. In Section 3 we define $\mathcal{AE}$-LTL, the new logic of path quantifier, and study its basic properties. In Section 4 we present a planning algorithm for $\mathcal{AE}$-LTL, while in Section 5 we apply the new logic to the particular cases of reachability and maintainability goals. In Section 6 we make comparisons with related works and present some concluding remarks.

2. Preliminaries

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

[1] 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.

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 finite path p' is a *strict extension* of the finite path p if it is an extension and $p' \neq p$.

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* [14].

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* $\beta(r(x_i)) = h$ *for infinitely many* i *is even.*

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

In [14] it is 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. Planning domains and plans

A *(nondeterministic) planning domain* [9] can be expressed in terms of a set of *states*, one of which is designated as the *initial state*, of a set of *actions*, and of a *transition function* describing how (the execution of) an action leads from one state to possibly many different states.

Definition 4 (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 the transition relation is total, namely, for each $\sigma \in \Sigma$ *there are 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* $R(\sigma, a) = \{\sigma_1, \sigma_1, \ldots, \sigma_n\}$ *and* $\sigma_1 < \sigma_2 < \cdots < \sigma_n$*.*

A *plan* guides the evolution of a planning domain by issuing actions to be executed. In the case of nondeterministic domains, *conditional plans* [9, 24] 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 5 (plan) *A* plan *is a partial map* $\pi : \Sigma^+ \rightharpoonup A$ *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 all and only the finite paths $w \in \Sigma^+$ that can be reached executing the plan from the initial state of the domain.

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 6 (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* $a_0 = \pi(\sigma_0)$*;*
- *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* $x_n \cdot i \in \mathrm{dom}(\mathcal{T})$ *and* $\mathcal{T}(x_n \cdot i) = (\sigma'_i, a'_i)$ *with* $a'_i = \pi(\sigma_0 \cdot \sigma_1 \cdots \sigma_n \cdot \sigma'_i)$*, for every* $0 \leq i < d$*.*

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

2.3. Temporal logics

Formulas of *Linear Temporal Logic* (LTL) [15] 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 8 (LTL) *LTL formulas* φ *on* $Prop$ *are defined by the following grammar, where* $q \in Prop$*:*

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

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. A formal definition of the semantics of LTL can be found in [15].

CTL* [15] 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 9 (CTL*) *CTL* formulas* ψ *on* $Prop$ *are defined by the following grammar, where* $q \in Prop$*:*

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

CTL* formulas are interpreted over Σ-labelled trees. The definition of the semantics of CTL* can be found in [15].

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 can be found in [13].

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 all and only 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. 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 of the sets of paths that satisfy the LTL property. More precisely, 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 $\mathcal{A}$, that is used to denote that φ must hold on *all* the paths, and the path quantifier $\mathcal{E}$, that 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-players 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 by the tree if the 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.

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

3.1. 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 10 ($\mathcal{AE}$-LTL) *A $\mathcal{AE}$-LTL formula is a pair $g = \alpha.\varphi$, where φ is a 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 11 (semantics of $\mathcal{AE}$-LTL) *Let p be a finite path of a Σ-labelled tree τ. 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 $\tau(p') \models_{LTL} \varphi$.*
- *$p \models \mathcal{E}.\varphi$ if for some infinite extension p' of p it holds that $\tau(p') \models_{LTL} \varphi$.*

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

We now investigate when two path quantifiers are equivalent, i.e., they select the same sets of paths.

Definition 12 (equivalent path quantifiers) *Let α and α' be two path quantifiers. We say that α* implies *α', written $\alpha \rightsquigarrow \alpha'$, if for all the Σ-labelled trees τ and for all the LTL formulas φ, $\tau \models \alpha.\varphi$ implies $\tau \models \alpha'.\varphi$. We say that α is* equivalent *to α', written $\alpha \sim \alpha'$, if $\alpha \rightsquigarrow \alpha'$ and $\alpha' \rightsquigarrow \alpha$.*

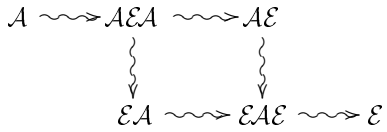
The following lemma describes the basic properties of path quantifiers.

Lemma 3 *Let α and α' be two finite path quantifiers.*

1. *$\alpha\mathcal{A}\mathcal{A}\alpha' \sim \alpha\mathcal{A}\alpha'$ and $\alpha\mathcal{E}\mathcal{E}\alpha' \sim \alpha\mathcal{E}\alpha'$.*
2. *$\alpha\mathcal{A}\alpha' \rightsquigarrow \alpha\alpha'$ and $\alpha\alpha' \rightsquigarrow \alpha\mathcal{E}\alpha'$.*
3. *$\alpha\mathcal{A}\alpha' \rightsquigarrow \alpha\mathcal{A}\mathcal{E}\mathcal{A}\alpha'$ and $\alpha\mathcal{E}\mathcal{A}\mathcal{E}\alpha' \rightsquigarrow \alpha\mathcal{E}\alpha'$.*
4. *$\alpha\mathcal{A}\mathcal{E}\mathcal{A}\mathcal{E}\alpha' \sim \alpha\mathcal{A}\mathcal{E}\alpha'$ and $\alpha\mathcal{E}\mathcal{A}\mathcal{E}\mathcal{A}\alpha' \sim \alpha\mathcal{E}\mathcal{A}\alpha'$.*

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 path quantifier α there is a canonical path quantifier $\alpha' \in \{\mathcal{A}, \mathcal{E}, \mathcal{AE}, \mathcal{EA}, \mathcal{AEA}, \mathcal{EAE}\}$ such that $\alpha \sim \alpha'$. Moreover, the following implications hold between the canonical path quantifiers:*



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. This consideration also applies to the case of infinite path quantifiers.

3.2. 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 $\tau \models \alpha.\varphi$, where α is an infinite game, if there is some strategy ξ for player E such that if p is an infinite paths of τ obtained according to α, by allowing player A to extend the path in an arbitrary way and by requiring that player E follows strategy ξ, then p satisfies the LTL formula φ.

Definition 13 (strategy) *A strategy for Σ-labelled tree τ is a mapping $\xi : P^*(\tau) \to P^*(\tau)$ that maps every finite path p to one of its finite, strict extensions $\xi(p)$.*

We require that path $\xi(p)$ is a strict extension of p in order to guarantee that, if player E takes control on the game, situations are avoided where the path is never extended.

Definition 14 (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 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 τ;*
- *if $\Pi_i = \mathcal{E}$ then $p_{i+1} = \xi(p_i)$;*
- *if $\Pi_i = \mathcal{A}$ then p_{i+1} is an arbitrary strict extension of p_i.*

The tree τ satisfies the $\mathcal{AE}$-LTL formula $g = \alpha.\varphi$, written $\tau \models g$, if there is some strategy ξ such that $\tau(p) \models_{LTL} \varphi$ for all paths p of τ that are possible outcomes of game α with strategy ξ.

In the next lemmas we extend to the case of infinite games the analysis of equivalence among path quantifiers.[2] 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 $\alpha(\mathcal{E})^\omega \sim \alpha\mathcal{E}$.*

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 6 *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 $\alpha \sim (\mathcal{EA})^\omega$.*

The next lemma contains other auxiliary results on path quantifiers.

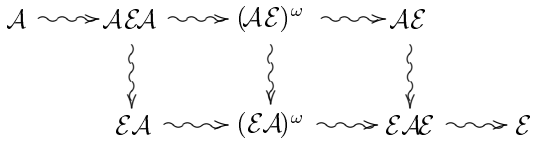
Lemma 7 *Let α be a finite path quantifier and α' be an infinite path quantifier.*

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 $\alpha\mathcal{E}\alpha' \rightsquigarrow \alpha(\mathcal{E})^\omega$.*

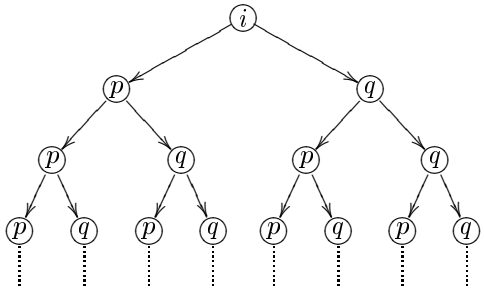
We can now complete the picture of Theorem 4: each finite or infinite path quantifier is equivalent to a *canonical path quantifier* that defines a game consisting of alternated moves of players A and E of length one, two, three, or infinity.

[2] The definitions of the implication and equivalence relations (Definition 12) also apply to the case of infinite path quantifiers.

Theorem 8 *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:*



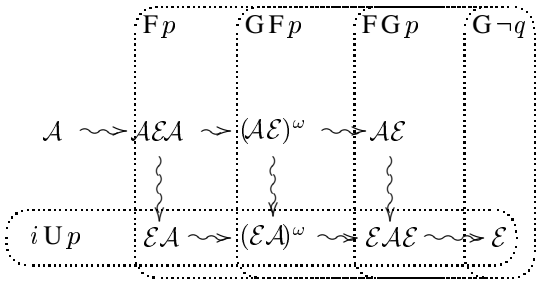
We conclude this section by showing that all the arrows in the diagram of Theorem 8 describe strict implications, namely, the eight canonical path quantifiers are all different. Let us consider the following $\{i, p, q\}$-labelled binary tree, where the root is labelled by i and each node has two children 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.
- $i\ \mathrm{U}\ p$: 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:



4. 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.

4.1. Tree automata and $\mathcal{AE}$-LTL formulas

In [5] it is 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 [20]. A tree can be obtained for an $\mathcal{AE}$-LTL formula using this reduction and Theorem 2. In this paper we describe a simpler reduction that is better suited for our algorithmic purposes.

A Σ-labelled tree $\mathcal{T}$ satisfies a formula $\alpha.\varphi$ if a 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 subsets of paths that are suitable for α, we assume to have a w-marking[3] of the tree $\mathcal{T}$, and we use the labels w to define the selected paths. More precisely, 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 15 ($\mathcal{AE}$-LTL and CTL*) *Let $\alpha.\varphi$ be an $\mathcal{AE}$-LTL formula. The CTL* formula $[[\alpha.\varphi]]$ is defined as follows:*

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

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 $\mathrm{F\,G}\,w$), that is, from a certain node on we follow the path marked with 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 9 *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 $[[\alpha.\varphi]]$.*

In [17] an extension of CTL* with existential quantification over atomic propositions (EGCTL*) is defined and the complexity of model checking and satisfiability for the new logic is examined. We remark that $\mathcal{AE}$-LTL can be seen as a subset of EGCTL*. Indeed, according to Theorem 9, 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 16 *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 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 10 *Let A and $A_{\exists w}$ be two parity tree automata as in Definition 16. $A_{\exists w}$ accepts all and only the Σ-labelled trees that have some w-marking which is accepted by A.*

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 17 (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.*

[3] 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 $\mathcal{T}_w(x) = (\sigma, \overline{w})$.

The parity tree automaton $A^{\mathcal{D}}_{\alpha.\varphi}$ is obtained by applying the transformation described in Definition 16 to the parity automaton $A^{\mathcal{D}}_{[[\alpha.\varphi]]}$ built according to Theorem 2.

Theorem 11 *The parity tree automaton $A^{\mathcal{D}}_{\alpha.\varphi}$ accepts all and only the Σ-labelled $\mathcal{D}$-trees that satisfy formula $\alpha.\varphi$.*

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

4.2. 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. 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 18 (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 6, 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 19 (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 6 and 19 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 13 *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 plan 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 tree $\mathcal{T}$.*

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 in obtained by combining in a suitable way the tree automaton for $\mathcal{AE}$-LTL formula g (Definition 17) and the tree automaton for domain D (Definition 18).

Definition 20 (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 lemma shows 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 14 *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 plan π is a solution to planning problem (D, g). Conversely, let π be a solution to planning problem (D, g). Then the execution tree of π is accepted by automaton $A^{\mathcal{D}}_{D,g}$.*

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 15 *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, tree automaton $A^{\mathcal{D}}_{D,g}$ is nonempty.*

Proposition 16 *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 $2^{O(|\varphi|)}$.*

4.3. Complexity

We now study the time complexity of the planning algorithm defined in Subsection 4.2.

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 17 *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 $2^{2^{O(|\varphi|)}}$.*

The doubly exponential time bound is tight. Indeed, the *realizability problem* for an LTL formula φ, that is known to be 2EXPTIME-complete [25], can be reduced to a planning problem for goal the $\mathcal{A}.\varphi$.

Theorem 18 *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.*

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 goal $\mathcal{E}.\varphi$ 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 [26]. We conjecture that, for all the canonical path quantifiers α except $\mathcal{E}$, the doubly exponential bound of Theorem 18 is tight.

Some remark 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[4], and the latter problem is known to be PSPACE-complete [26]. A similar argument holds also for validity. We leave for the full version of the paper the study of the complexity of the model checking problem for $\mathcal{AE}$-LTL.

5. Reachability and maintainability goals

In this section we consider two basic classes of goals, namely the *reachability goals* corresponding to the LTL formula $\mathrm{F}\,q$ and the *maintainability goals* corresponding to the LTL formula $\mathrm{G}\,q$.

Let us start from the case of the "classical" reachability goals $\mathrm{F}\,q$, where q is a propositional formula. In this case, as soon as player E takes control, the reachability goal can be achieved, if possible; therefore, the only relevant cases are $\mathcal{A}.\,\mathrm{F}\,q$, $\mathcal{E}.\,\mathrm{F}\,q$, and $\mathcal{AE}.\,\mathrm{F}\,q$.

Lemma 19 *Let $\mathcal{T}$ be a labelled tree. Then $\mathcal{T} \models \mathcal{E}.\,F\,q$ iff $\mathcal{T} \models \mathcal{EA}.\,F\,q$ iff $\mathcal{T} \models \mathcal{EAE}.\,F\,q$ iff $\mathcal{T} \models (\mathcal{EA})^{\omega}.\,F\,q$. Moreover $\mathcal{T} \models \mathcal{AE}.\,F\,q$ iff $\mathcal{T} \models \mathcal{AEA}.\,F\,q$ iff $\mathcal{T} \models (\mathcal{AE})^{\omega}.\,F\,q$.*

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 [11].

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 one of reachability goals. In this case, as soon as player A takes control, the maintainability goal can be violated, if possible; therefore, the only relevant cases are $\mathcal{A}.\,\mathrm{G}\,q$, $\mathcal{E}.\,\mathrm{G}\,q$, and $\mathcal{EA}.\,\mathrm{G}\,q$.

Lemma 20 *Let $\mathcal{T}$ be a labelled tree. Then $\mathcal{T} \models \mathcal{A}.\,G\,q$ iff $\mathcal{T} \models \mathcal{AE}.\,G\,q$ iff $\mathcal{T} \models \mathcal{AEA}.\,G\,q$ iff $\mathcal{T} \models (\mathcal{AE})^{\omega}.\,G\,q$. Moreover $\mathcal{T} \models \mathcal{EA}.\,G\,q$ iff $\mathcal{T} \models \mathcal{EAE}.\,G\,q$ iff $\mathcal{T} \models (\mathcal{EA})^{\omega}.\,G\,q$.*

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 require 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 ($\mathcal{EA}.\,\mathrm{G}\,p$).

6. Related works and concluding remarks

In this paper we have defined $\mathcal{AE}$-LTL, a new temporal logics 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, and we have studied its time complexity.

In the field of planning, several works use temporal logics for defining goals. Most of these approaches [2, 3, 6, 7, 12, 19] 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 in [23, 24] and in [10]: in [23, 24], CTL is used as goal language, while in [10] a new branching time logic is defined that allows for expressing temporally extended goals that can deal explicitly with failure and recovery in goal achievement. In the goal languages of [10, 23, 24], 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) [1] 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 in [1]. Our conjecture is that our logic and ATL* are of incomparable expressiveness.

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

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

for nesting of path quantifiers, or for goals that combine $\mathcal{AE}$-LTL formulas with CTL and CTL* formulas. Another direction for future investigations is the extension of the approach proposed in this paper to the case of planning under partial observability [12], 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 (see, e.g., [4, 23]).

Acknowledgments. We would like to thank Erich Grädel for his comments on the reduction of $\mathcal{AE}$-LTL formulas to CTL* formulas.

References

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

[2] F. Bacchus and F. Kabanza. Planning for temporally extended goals. *Ann. of Mathematics and Artificial Intelligence*, 22:5–27, 1998.

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

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

[5] D. Berwanger, E. Grädel, and S. Kreutzer. Once upon a time in the west (a note on path games), 2003. Private communication.

[6] D. Calvanese, G. de Giacomo, and M. Vardi. 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)*, pages 593–602, 2002.

[7] S. Cerrito and M. Mayer. 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)*, volume 1397 of *LNAI*, pages 124–140. Springer Verlag, 1998.

[8] A. Cimatti, M. Roveri, and P. Traverso. Automatic OBDD-based generation of universal plans in non-deterministic domains. In *Proc. of 15th National Conf. on Artificial Intelligence (AAAI'98)*, pages 875–881. AAAI Press, 1998.

[9] A. Cimatti, M. Roveri, and P. Traverso. Strong planning in non-deterministic domains via model checking. In *Proc. of 4th Int. Conf. on Artificial Intelligence Planning Systems (AIPS-98)*, pages 36–43. AAAI-Press, 1998.

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

[11] M. Daniele, P. Traverso, and M. Vardi. Strong cyclic planning revisited. In *Proc. of 5th European Conf. in Planning (ECP'99)*, volume 1809 of *LNAI*, pages 35–48. Springer Verlag, 1999.

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

[13] E. Emerson and C. Jutla. The complexity of tree automata and logics of programs. In *Proc. of 29th IEEE Symp. on Foundations of Computer Science*, pages 328–337, 1988.

[14] E. Emerson and C. Jutla. Tree automata, μ-calculus and determinacy. In *Proc. of 32nd IEEE Symp. on Foundations of Computer Science*, pages 368–377, 1991.

[15] E. A. Emerson. Temporal and modal logic. In J. van Leeuwen, editor, *Handbook of Theoretical Computer Science, Volume B: Formal Models and Semantics*. Elsevier, 1990.

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

[17] O. Kupferman. Augmenting branching temporal logics with existential quantification over atomic propositions. In *Proc. 8th Int. Conf. on Computer Aided Verification (CAV'95)*, volume 939 of *LNCS*, pages 325–338. Springer-Verlag, 1995.

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

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

[20] F. Moller and A. Rabinovich. On the expressive power of CTL*. In *Proc. of 4th Annual IEEE Symposium on Logic in Computer Science (LICS'99)*, pages 360–369. IEEE Computer Science Press, 1999.

[21] J. Penberthy and D. Wed. 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)*, 1992.

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

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

[24] M. Pistore and P. Traverso. 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, 2001.

[25] A. Pnueli and R. Rosner. Distributed reactive systems are hard to synthesize. In *Proc. of 31st IEEE Symp. on Foundation of Computer Science*, pages 746–757, 1990.

[26] A. Sistla and E. Clarke. The complexity of propositional linear temporal logic. *Journal ACM*, 32:733–749, 1985.

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