Planning under LTL Environment Specifications*

Benjamin Aminof
JKU Linz and TU Wien
Austria
aminof@forsyte.at

Giuseppe De Giacomo
Sapienza Univ. Roma
Rome, Italy
degiacomo@dis.uniroma1.it

Aniello Murano
Univ. Federico II
Naples, Italy
murano@unina.it

Sasha Rubin
Univ. Federico II
Naples, Italy
rubin@unina.it

Abstract

Planning domains represent what an agent assumes or believes about the environment it acts in. In the presence of nondeterminism, additional temporal assumptions, such as fairness, are often expressed as extra conditions on the domain. Here we consider environment specifications expressed in arbitrary LTL, which generalize many forms of environment specifications, including classical specifications of nondeterministic domains, fairness, and other forms of linear-time constraints on the domain itself. Existing literature typically implicitly or explicitly considers environment specifications as constraints on possible traces. In contrast, in spite of the fact that we use a linear-time formalism, we propose to consider environment specifications as specifications of environment strategies. Planning in this framework is the problem of computing an agent strategy that achieves its goal against all environment strategies satisfying the specification. We study the mathematical and computational properties of planning in this general setting. We observe that not all LTL formulas correspond to legitimate environment specifications, and formally characterize the ones that do. Moreover, we show that our notion of planning generalizes the classical notion of *Church's synthesis*, and that in spite this one can still solve it optimally using classical Church's synthesis.

1 Introduction

A foundational characteristic of planning is that, in devising plans, the agent takes advantage of a representation of its environment (McCarthy 1957; Green 1969; Lin and Levesque 1998; Reiter 2001; Ghallab, Nau, and Traverso 2004; Geffner and Bonet 2013), which corresponds to knowledge that the agent has of how its environment behaves.[1] In other words, the agent *assumes* that its environment works according to certain specifications, and *exploits these environment specifications in devising its plans*. Such environment specifications come in a variety of forms. Obviously, the planning domain itself (here we focus on nondeterministic fully observable domains (FONDs) with initial states) with its preconditions and effects is the typical form of environment specification: as long as the agent sticks to its preconditions, the environment brings about effects as described by the domain. So, the agent can exploit the effects of its actions in order to enforce its goal. Another common piece of environment specifications is to assume that the environment is *fair* in resolving its nondeterministic effects, so-called *fair FOND* (Daniele, Traverso, and Vardi 1999; Cimatti et al. 2003; Camacho et al. 2017; D'Ippolito, Rodríguez, and Sardiña 2018). In this case, the agent can exploit not only the effects but also the guarantee that, by repeatedly executing a given action from a given state, the environment will repeatedly respond with all its possible nondeterministic effects. More sophisticated pieces of environment specifications are LTL trajectories constraints over the domain, which have been proposed to model general assumptions on the environment other than fairness (Bonet and Geffner 2015; Bonet et al. 2017).

Planning domains, fairness of effects, and trajectory constraints can all be seen as linear-time temporal representations, i.e., LTL formulas. So a natural generalization is to consider general *environment specification expressed as arbitrary LTL formulas*. This is what we study in this paper.

Observe, however, that LTL formulas are interpreted over *traces*, e.g., domain specifications are thought of as sets of traces, fairness is expressed as a restriction on this set of traces, and trajectory constraints are also defined as restrictions on the traces in a certain domain. So, a crucial question is *in what sense can an LTL formula be considered a specification of the environment?* A specification should allow us to single out the objects that are denoted by it; but LTL models, the natural object denoted by the logic, are traces and not environment behaviors.

In this paper, we stipulate that environment specification single out *environment strategies*. This view is in line with the classical framework of *Church's synthesis* (aka reactive synthesis) (Church 1963; Pnueli and Rosner 1989), and, indeed, planning becomes a generalized form of Church's synthesis that is concerned with synthesizing strategies for the agent such that for *all* strategies of the environment *that satisfy the environment specification* expressed in LTL, the execution generated satisfies the goal expressed in LTL.

Still, we have to establish *in what sense an LTL formula can be taken as a specification of environment strategies*.

*Benjamin Aminof was supported by Austrian Science Fund (FWF): P 32021

[1] In this paper, we don't restrict plans to be sequences of actions, but rather allow them to be stateful controllers, also called *strategies*.

Again we resort to Church's synthesis: an LTL formula specifies all those environment strategies that are able to react to agent actions so as to enforce the LTL formula. In other words, an LTL environment specification denotes the environment strategies that enforce the LTL formula against all agent strategies.

Notice that not all LTL formulas can be considered legitimate environment specifications. For example, consider a formula expressing that eventually a certain possible action must be performed (and note that the agent may decide not to perform this action). Focusing on environment strategies allows us to give a precise and elegant account of which LTL formulas can be considered legitimate environment specifications. Specifically, to be legitimate, an environment specification needs to satisfy a basic *consistency* condition: *it needs to denote a nonempty set environment strategies.* Checking this form of consistency amounts to Church's synthesis from the point of view of the environment (i.e., checking if one can synthesize a strategy of the environment against the agent), and can be done in 2-EXPTIME. Eventhough, in many cases, checking this consistency is simpler, e.g., domain specifications and fairness constraints are trivially consistent, this bound is tight.

Once we have established what (consistent) environment specifications are, we study *how to solve* planning under LTL environment specifications. We resort again to the Verification literature, which take assumptions (our environment specifications) into account by resorting to Church's synthesis for the implication

$$EnvSpec \supset Goal$$

where both $EnvSpec$ and $Goal$ are expressed in LTL (Chatterjee and Henzinger 2007; Chatterjee, Henzinger, and Jobstmann 2008). *Does planning under LTL environment specification amount to synthesizing for the above implication?* Surprisingly, we show that this is *not the case*. An agent that synthesizes for the implication is too pessimistic: an agent strategy σ_{ag} for the implication has to ensure that $Goal$ holds, not only against environment strategies which enforce $EnvSpec$, but also against those that don't and yet when executed together with σ_{ag} make $EnvSpec$ true. In this way, the agent gives too much power to the environment, which can indeed break its specification.

On the other hand, surprisingly, we show that if there is an agent strategy enforcing $Goal$ under environment assumption $EnvSpec$, then there also exists one that enforces the implication (and vice-versa). Thus, even if the implication *cannot be used for characterizing planning under LTL environment specifications*, it *can be used to solve it*.

Exploiting this result, we devise techniques for planning under LTL environment specifications, and establish 2-EXPTIME-completeness in the general case. Then we look at special case that consist of classical FOND domains, with additional LTL assumptions. For this, we refine the analysis (in a non trivial way) and get a 1-EXPTIME bounds in the size of the domain (compactly represented) and 2-EXPTIME bound in the LTL assumptions and goals.

We conclude the paper by discussing related work in Verification and giving an outlook to future research.

2 Preliminaries

Linear-time Temporal logic

Linear-temporal logic (LTL) is a useful and natural logic for talking about sequences (later we will also use LTL to talk about strategies). *Formulas of LTL over Boolean variables* AP, or simply *LTL*, are generated by the following grammar: $\varphi ::= p \mid \varphi \vee \varphi \mid \neg\varphi \mid \mathsf{X}\varphi \mid \varphi \,\mathsf{U}\, \varphi$ where $p \in AP$. The *size* $|\varphi|$ of a formula φ is the number of symbols in it. LTL formulas are interpreted over infinite sequences $\tau \in (2^{AP})^\omega$, called *traces*, of valuations of the variables. The satisfaction relation $(\tau, n) \models \varphi$, stating that φ holds at step n of sequence τ, is defined as follows:

- $(\tau, n) \models p$ iff $p \in \tau_n$;
- $(\tau, n) \models \varphi_1 \vee \varphi_2$ iff $(\tau, n) \models \varphi_i$ for some $i \in \{1, 2\}$;
- $(\tau, n) \models \neg\varphi$ iff it is not the case that $(\tau, n) \models \varphi$;
- $(\tau, n) \models \mathsf{X}\,\varphi$ iff $(\tau, n+1) \models \varphi$;
- $(\tau, n) \models \varphi_1 \,\mathsf{U}\, \varphi_2$ iff there exists $i \geq n$ such that $(\tau, i) \models \varphi_2$ and for all $j \in [n, i)$, $(\tau, j) \models \varphi_1$.

We write $\tau \models \varphi$ if $(\tau, 0) \models \varphi$ and say that τ *satisfies* φ and τ is a *model* of φ. We use the usual abbreviations, $\varphi \supset \varphi' \doteq \neg\varphi \vee \varphi'$, $\mathsf{true} := p \vee \neg p$, $\mathsf{false} \doteq \neg\mathsf{true}$, $\mathsf{F}\,\varphi \doteq \mathsf{true}\,\mathsf{U}\,\varphi$, $\mathsf{G}\,\varphi \doteq \neg\,\mathsf{F}\,\neg\varphi$.

We remark that every result in this paper that mentions LTL also holds for LDL (linear dynamic logic) (Eisner and Fisman 2006).

Planning

Classical planning involves finding a sequence of actions that drives the initial state to a goal state. Fully observable non-deterministic (FOND) planning involves finding a plan which tells the agent what to do in every possible state, so that a goal state is reached no matter how the nondeterminism is resolved. Since we consider temporally extended goals expressed in LTL, rather than just achievement goals, planning involves finding strategies which tell the agent what to do from every possible history (rather than memoryless policies, i.e, plans which tell the agent what to do in every possible state) (Bacchus and Kabanza 2000).

FOND domains are typically represented compactly by taking states to be evaluations of a set of Boolean variables (Rintanen 2004; Geffner and Bonet 2013). Specifically, a *FOND domain specification* is a tuple $D = \langle F, A, init, ops\rangle$ where F is a set of Boolean variables called *fluents*, A is a set of *actions*, $init$ is a Boolean formula over F describing the *initial* states, and ops associates with each $a \in A$ a pair (pre_a, eff_a) where pre_a is a Boolean formula over F called the *precondition* of action a, and eff_a, the *effects* of action a, is specified as a non-empty set of pairs $E = (E^+, E^-)$ of disjoint sets of fluents. A *state* is an evaluation of the fluents. A state s is *initial* if $s \models init$. An action a is *possible* in state s if $s \models pre_a$. We assume that every state has at least one possible action (this can be ensured by adding a dummy "fail" state if needed). The *successor states* of a state s under effects eff_a is the set of states s' for which there exists $E = (E^+, E^-)$ in eff_a such that i) $s' \models f$ for all $f \in E^+$, ii) $s' \models \neg f$ for all $f \in E^-$, and

iii) $s' \models f$ iff $s \models f$, for all $f \notin E^+ \cup E^-$. If a is possible in s, and s' is a successor state of s under eff_a, then we call (s, a, s') a *transition*; we let T denote the set of transitions.

We now define traces. For a state s and an action a, let $[s, a]$ denote the set $s \cup \{a\}$. A *trace of* D is a finite or infinite sequence of the form $[s_0, a_0][s_1, a_1] \cdots$ where s_0 is an initial state and $(s_i, a_i, s_{i+1}) \in T$ for all i.[2] A trace of D is *fair for effects*, or simply *fair*, if for every $(s, a, s') \in T$ for which there are infinitely many i such that $s_i = s, a_i = a$ it also holds that there are infinitely many i such that $s_i = s, a_i = a, s_{i+1} = s'$ (Daniele, Traverso, and Vardi 1999; Cimatti et al. 2003). Note that finite traces are fair.

We now define strategies which tell the agent how to act given the history of states.[3] An *agent strategy* is a function $\sigma_{\mathsf{ag}} : S^+ \to A$, i.e., it maps the finite non-empty sequences of states (denoted by S^+) to actions. A finite or infinite sequence $[s_0, a_0][s_1, a_1] \cdots$ is *induced* by σ_{ag} if $\sigma_{\mathsf{ag}}(s_0 s_1 \dots s_i) = a_i$ for all i. An agent strategy is *executable* if for every such induced sequence that is also a trace of D, the strategy always selects possible actions, i.e., if $s_i \models \mathit{pre}_{a_i}$ for every i.

A *FOND problem with LTL goal* is a pair $P = (D, \gamma)$ where D is a FOND domain specification and γ is an LTL formula over $F \cup A$ called the *goal*. An executable agent strategy σ_{ag} is a *solution* to the FOND problem $P = (D, \gamma)$ if every infinite trace of D induced by σ_{ag} satisfies γ; it is a *solution* to the FOND problem P *under fairness* (*fair FOND problem* in the following) if every fair infinite trace of D induced by σ_{ag} satisfies γ (Camacho et al. 2017). Note that ordinary FOND problems amount to taking $\gamma \doteq \mathsf{F}\, \mathit{Goal}$ where Goal is a Boolean formula over the variables F. It is known that solving (fair) FOND problems with reachability goals, i.e., both with and without the fairness of effects, is 1EXPTIME-complete (Rintanen 2004).

Church's Synthesis

Church's Synthesis (aka Reactive Synthesis) is the problem of producing a module that satisfies a given property no matter how the environment behaves (Church 1963; Pnueli and Rosner 1989). Let X and Y be disjoint finite sets of Boolean variables. The idea is that the environment sets the variables in X, and the agent then responds by setting the variables in Y. A *reactive module* or *agent strategy* is a function $\sigma_{\mathsf{ag}} : (2^X)^+ \to 2^Y$, i.e., it tells the agent how to set its variables given the history of assignments of the environment. A *trace* is a sequence $(X_0 \cup Y_0)(X_1 \cup Y_1) \dots$ over the alphabet $2^{X \cup Y}$, i.e., $X_i \subseteq X, Y_i \subseteq Y$. An agent strategy *induces* a trace $(X_i \cup Y_i)_i$ if $\sigma_{\mathsf{ag}}(X_0 X_1 \cdots X_j) = Y_j$ for every $j \geq 0$.

In what follows we will also make use of strategies for the environment. Specifically, an *environment strategy* is a function $\sigma_{\mathsf{env}} : (2^Y)^* \to 2^X$, i.e., it captures how the environment starts as well as how it responds to the history of assignments of the agent. An environment strategy *induces* a trace $(X_i \cup Y_i)_i$ if $\sigma_{\mathsf{env}}(\epsilon) = X_0$ and $\sigma_{\mathsf{env}}(Y_0 Y_1 \cdots Y_j) = X_{j+1}$ for every $j \geq 0$. For an agent strategy σ_{ag} and an environment strategy σ_{env} let $\tau(\sigma_{\mathsf{ag}}, \sigma_{\mathsf{env}})$ denote the unique sequence over the alphabet $2^{X \cup Y}$ induced by both σ_{ag} and σ_{env}.

Let φ be an LTL formula. An agent strategy σ_{ag} *enforces*[4] φ, written $\sigma_{\mathsf{ag}} \rhd \varphi$, if for every environment strategy σ_{env}, the sequence $\tau(\sigma_{\mathsf{ag}}, \sigma_{\mathsf{env}})$ satisfies φ, i.e.,

$$\forall \sigma_{\mathsf{env}}. \tau(\sigma_{\mathsf{ag}}, \sigma_{\mathsf{env}}) \models \varphi.$$

In this case we say that φ is *agent enforceable*. Similarly, an environment strategy σ_{env} *enforces* φ, written $\sigma_{\mathsf{env}} \rhd \varphi$, if for every agent strategy σ_{ag}, the sequence $\tau(\sigma_{\mathsf{ag}}, \sigma_{\mathsf{env}})$ satisfies φ. In this case we say that φ is *environment enforceable*. Deciding if an LTL formula is agent enforceable (or environment enforceable), and computing a finite-state strategy[5] if one exists, is called the *LTL synthesis problem*. These problems are known to be 2EXPTIME-complete:

Theorem 1. *(Pnueli and Rosner 1989) Solving the LTL synthesis problem is* 2EXPTIME*-complete.*

In the rest of the paper, we let X denoted a set of fluents (i.e., $X = F$), and Y denote the bit representations of a set of actions (i.e., $A = 2^Y$).

3 Environment Specifications

Our work starts from the observation that we can conceive *environment specifications* as mechanisms for selecting environment strategies. In particular, an environment specification Σ denotes a *set of environment strategies*. This is different from a *goal* Γ which is a specification of a *set of traces* that are desirable. Under this view, the planning task itself can be see as a generalization of Church's synthesis: *find an agent strategy (a plan) such that for all environment strategies satisfying (i.e., denoted by) the environment specification we have that the resulting trace produced by the two strategies together satisfies (i.e., is denoted by) the goal.* In formulas: find an agent strategy σ_{ag} such that

$$\forall \sigma_{\mathsf{env}} \in \Sigma.\, \tau(\sigma_{\mathsf{ag}}, \sigma_{\mathsf{env}}) \in \Gamma$$

Note that this trivializes if Σ denotes an empty set, i.e., if the environment specification is inconsistent and rules out all environment strategies. Hence, we require that the environment specification is *consistent*, i.e., denotes at least one environment strategy.

It is easy to see that classical FOND domain specifications can be seen as environment specifications in the above sense. Let D be a FOND domain specification, e.g.,

[2] Sometimes traces are called possible executions, and defined to be alternating sequences of states and actions, i.e., $s_0 a_0 s_1 a_1 \cdots$, which start in an initial state and respect the transitions. Our definition is just a convenient notational variation.

[3] These are similar to history-based policies (Geffner and Bonet 2013).

[4] "Enforces" is often called "realizes" in the synthesis literature; cf. (Pnueli and Rosner 1989).

[5] A strategy is *finite state* if it is computed by a finite-state transducer, i.e., a deterministic finite-state machine that is fed symbols from an input alphabet, and symbol by symbol, it produces a symbol from an output alphabet, and changes its internal state. It turns out that if there is a strategy enforcing an LTL formula φ, then there is a finite-state one doing so.

a specification expressed in PDDL (Planning Domain Definition Language). Classically, D denotes a transition system, i.e., D compactly represents the set of states and how the agent actions and the environment reactions (resolving nondeterminism) drive the system from one state to another. However, we can also see D as specifying a set of *environment strategies* $\sigma_{\mathsf{env}} : A^* \rightarrow S$ which specify how the environment resolves the nondeterminism. An environment strategy σ_{env} is *specified by* D if i) $\sigma_{\mathsf{env}}(\epsilon)$ is an initial state, and ii) for every sequence $a_0 a_1 \cdots a_k$ of actions, if a_k is possible in state $\sigma_{\mathsf{env}}(a_0 a_1 \cdots a_{k-1})$ then $(\sigma_{\mathsf{env}}(a_0 a_1 \cdots a_{k-1}), a_k, \sigma_{\mathsf{env}}(a_0 a_1 \cdots a_k)) \in T$.

The following proposition shows that this new view is consistent with the classic view of FOND planning:

Proposition 2. *Let $P = (D, \gamma)$ be a FOND domain with an LTL goal, and let Σ_D be the set of environment strategies specified by D. An executable agent strategy σ_{ag} is a solution for P iff $\forall \sigma_{\mathsf{env}} \in \Sigma_D . \tau(\sigma_{\mathsf{ag}}, \sigma_{\mathsf{env}}) \models \gamma$.*

Note that FOND domains denote consistent sets, i.e., Σ_D is always non-empty.

Similarly, we can immediately handle fair FOND problems by further restricting the environment strategies σ_{env} to the *fair environment strategies*, i.e., those that together with any agent strategy generate a fair trace. Let $\Sigma_{D,fair}$ denote this set of environment strategies. The following proposition shows that this new view is consistent with the classic view of fair FOND planning:

Proposition 3. *Let $P = (D, \gamma)$ be a FOND domain with an LTL goal, and let $\Sigma_{D,fair}$ be the set of fair environment strategies specified by D. An executable agent strategy σ_{ag} is a fair solution for P iff $\forall \sigma_{\mathsf{env}} \in \Sigma_{D,fair} . \tau(\sigma_{\mathsf{ag}}, \sigma_{\mathsf{env}}) \models \gamma$.*

Note that fair FOND domains denote consistent sets, i.e., $\Sigma_{D,fair}$ is always non-empty.

In this paper, we generalize environment specifications from nondeterministic domains, possibly with fairness of effects, to arbitrary LTL formulas. In this way, for example, we may express that an agent doing a certain action now, results in having a certain effect *eventually* (as opposed to at the next state, as in typical planning domains).

Note that *LTL models are traces and not strategies*. Hence, we need to understand in what sense an LTL formula provides us with an environment specification. To do so, we observe that an LTL formula can be used to specify the set of strategies that describe how the *environment reacts to the agent's actions* (no matter what the agent does) in order to satisfy the LTL formula. That is, an LTL formula can be used to specify those environment strategies *enforce* the LTL formula, i.e., that fulfill the LTL formula against all possible agent strategies. Hence, *the environment specification in LTL denotes those environment strategies that enforce the LTL formula*.

Generalizing the ideas above, we can use arbitrary LTL formulas as specifications for sets of environment strategies.

Definition 1. *Let ε be an LTL formula. An environment strategy σ_{env} is* specified by *ε if σ_{env} enforces ε, in symbols $\sigma_{\mathsf{env}} \rhd \varepsilon$.*

On the basis of this general definition we now define planning under LTL environment specifications that generalizes Proposition 2.

Definition 2. *1. Let γ and ε be LTL formulas. An agent strategy σ_{ag}* enforces *γ* under the environment specification *ε if for all environment strategies σ_{env} specified by ε we have that $\tau(\sigma_{\mathsf{ag}}, \sigma_{\mathsf{env}}) \models \gamma$, in symbols:*

$$\forall \sigma_{\mathsf{env}} \rhd \varepsilon . \tau(\sigma_{\mathsf{ag}}, \sigma_{\mathsf{env}}) \models \gamma .$$

2. Planning under LTL environment specifications *is the problem of deciding whether there is an agent strategy that enforces γ under the environment specification ε, and returning such a strategy.*[6]

Let us illustrate these definitions with some notable examples. We can capture FOND specifications in LTL. Consider the following example.

Example 1. *In robot-action planning problems, typical environment specifications encode the physical space, e.g., "if robot is in Room 1 and does action $Move$ then in the next step it can only be in Rooms 1 or 4". Such specification are usually hardwired into the domain. However, they can also be captured by environment strategies σ_{env} satisfying the following property: for every trace $[s_1, a_1][s_2, a_2] \ldots$ generated by σ_{env} such that "the robot is in Room 1" holds in the state s_i, and a_i is the action $Move$, then either "the robot is in Room 1" or "the robot is in Room 4" holds in the state s_{i+1}. Note that this is a condition on traces that can be expessed in LTL by a formula of the form $\mathsf{G}(R_1 \wedge Move \supset \mathsf{X}(R_1 \vee R_4))$; here R_i is a fluent representing that the robot is in Room i.*

In general, we can capture any FOND domain specification D as an LTL formula:

$$\varepsilon_D \doteq \mathit{init} \wedge \bigwedge_{a \in A} \mathsf{G}((a \wedge \mathit{pre}_a) \supset (\bigvee_{E \in \mathit{eff}_a} \varphi_1 \wedge \varphi_2 \wedge \varphi_3))$$

where φ_1 is the formula $\bigwedge_{f \in E^+} \mathsf{X} f$, φ_2 is the formula $\bigwedge_{f \in E^-} \mathsf{X} \neg f$, and φ_3 is the formula $\bigwedge_{f \notin E^+ \cup E^-} f \leftrightarrow \mathsf{X} f$. This LTL formula says, of a sequence, that it starts in an initial state, and at every point in time, if a given action occurs and its precondition holds in the same step, then one of its effects holds in the next step. It is easy to see that the set of environment strategies specified by D are exactly those that *enforce* ε_D, i.e., those that make ε_D true no matter which strategy the agent adopts in selecting actions (this is formalized in Proposition 4, below).

Analogously we can capture fairness of effects in D by the following LTL formula:

$$\varepsilon_{D,fair} \doteq \varepsilon_D \wedge \bigwedge_{(s,a,s') \in T} (\mathsf{G}\,\mathsf{F}(s \wedge a) \supset \mathsf{G}\,\mathsf{F}(s \wedge a \wedge \mathsf{X} s')) .$$

This LTL formula says, of a sequence, that ε_D holds, and that if a state-action pair occurs infinitely often then it is infinitely often followed by every possible successor state.

[6] In fact, we will look for a finite-state strategies, since just as for ordinary LTL synthesis, it turns out that if there is a strategy then there is a finite-state strategy.

Similarly, the set of environment strategies legal for D under fairness are exactly those that realise $\varepsilon_{D,fair}$.

To summarize, we have the following proposition.

Proposition 4. *Let D be a FOND domain specification (respectively, fair FOND domain specification). The set of environment strategies specified by D (respectively, under fairness) are exactly those that enforce ε_D (respectively, $\varepsilon_{D,fair}$).*

Proof. Suppose σ enforces the formula ε_D. In particular, $\sigma(\epsilon)$ satisfies $init$ since every trace induced by σ (and there is at least one) satisfies ε_D (which logically implies $init$). Let $a_0 a_1 \cdots$ be an infinite sequence of actions. Write $\sigma(a_{\leq i})$ for $\sigma(a_0 a_1 \cdots a_i)$. Let k be such that a_k is possible in $\sigma(a_{\leq k-1})$. We will show that $(\sigma(a_{\leq k-1}), a_k, \sigma(a_{\leq k})) \in T$. Consider the infinite trace $\tau \doteq [\sigma(\epsilon), a_0][\sigma(a_0), a_1][\sigma(a_0 a_1), a_2] \cdots$. Since this trace is induced by σ, it satisfies ε_D. By the choice of k, both a_k and pre_{a_k} hold (in position $k+1$), and thus $\sigma(a_{\leq k})$ satisfies the effects (in position $k+2$), as required.

For the other direction, suppose σ is specified by D. Let $\tau = [s_0, a_0][s_1, a_1] \cdots$ be an infinite trace induced by σ. By assumption $\sigma(\epsilon)$ satisfies $init$. Let $k \geq 0$ be such that $a_k = a$, and s_k satisfies pre_a. Then, by assumption, $(s_k, a_k, s_{k+1}) \in T$, i.e., s_{k+1} satisfies the effects. Thus τ satisfies ε_D, as required.

The fair case is similar. □

It turns out that virtually all forms of planning (with linear-time temporally extended goals) in the literature are special cases of planning under LTL environment specifications (with LTL goals), i.e., the set of strategies that solve a given planning problem are exactly the set of strategies that solve the corresponding planning under environment specifications problem. In the following proposition, $Goal$ is a Boolean formula over F, and $Exec$ is the LTL formula $\mathsf{G} \bigwedge_{a \in A}(a \supset pre_a)$ expressing that if an action is done then its precondition holds.

Proposition 5. *1. FOND planning with reachability goals (Rintanen 2004) is a special case of planning under LTL environment specifications with environment specification $\varepsilon \doteq \varepsilon_D$ and goal $\gamma \doteq Exec \wedge \mathsf{F}\, Goal$.*

2. *FOND planning with LTL (temporally extended) goals γ (Bacchus and Kabanza 2000; Camacho et al. 2017) is a special case of planning under LTL environment specifications with environment specification $\varepsilon \doteq \varepsilon_D$ and goal $Exec \wedge \gamma$.*
3. *In planning, trace constraints, e.g., expressed in LTL, have been introduced for expressing temporally extended goals (Bacchus and Kabanza 2000; Gerevini et al. 2009). More recently, especially in the context of generalized planning, they have been used to describe restrictions on the environment as well (Bonet and Geffner 2015; De Giacomo et al. 2016; Bonet et al. 2017). FOND planning with LTL trace constraints ε and LTL (temporally extended) goals γ (Bonet and Geffner 2015; Bonet et al. 2017) is a special case of planning under LTL environment specifications with environment specification ε (though, we do need to check whether $\varepsilon \wedge \varepsilon_D$ is a consistent enviroment specification, see later) and goal $Exec \wedge \gamma$.*
4. *Fair FOND planning with reachability goals (Daniele, Traverso, and Vardi 1999; Geffner and Bonet 2013; D'Ippolito, Rodríguez, and Sardiña 2018) is a special case of planning under LTL environment specifications with environment specification $\varepsilon \doteq \varepsilon_{D,fair}$, and $\gamma \doteq Exec \wedge \mathsf{F}\, Goal$.*[7]
5. *Fair FOND planning with (temporally extended) goals γ (Patrizi, Lipovetzky, and Geffner 2013; Camacho et al. 2017) is a special case of planning under LTL environment specifications with environment specification $\varepsilon_{D,fair}$ and goal $Exec \wedge \gamma$.*
6. *Obviously adding LTL trace constraints ε to fair FOND planning with (temporally extended) goals is a special case of planning under LTL environment specifications (though one needs to check that $\varepsilon_{D,fair} \wedge \varepsilon$ is a consistent enviroment specification, see later).*

We conclude this section by observing that the definitions above can be immediately generalized to any other linear-time specifications, e.g., automata over infinite words. In a later section we study the complexity of FOND planning under environment specifications, and we will exploit this fact in order to provide optimal complexity, i.e., for FOND planning we will not use an LTL formula for the domain (cf. ε_D), but rather express the domain directly as an automaton.

4 Environment Specification Consistency

We now know how to use any LTL formula as an environment specification, i.e., to specify sets of environment strategies. However, as mentioned previously, not all formulas are legitimate environment specifications. This leads to the question: *Which formulas are legitimate environment specifications?*

We first discuss the intuition. When do we say that a logical specification is legitimate? When it is *consistent*, i.e., it admits at least one model. But in our setting we cannot simply use standard logical consistency, because, since the models of linear-time formulas are traces, standard consistency would only guarantee that the environment specification would allow at least one trace. To get such a trace we would need cooperation between the agent and the environment. Instead, under the view that environment specifications are specifications of environment strategies, the consistency requirement becomes: *admitting at least one environment strategy that fullfils the specification in spite of what the agent does.*

We can make such intuitions precise using Church's synthesis.

Definition 3. *An LTL formula ε is a* consistent environment specification *if it is environment enforceable, in symbols: $\exists \sigma_{\mathsf{env}}.\ \sigma_{\mathsf{env}} \rhd \varepsilon$.*

[7] Unfortunately, such a formula explicitly mentions the states s, s' and action a, and hence may be exponential in D (recall that D is represented compactly).

Note that FOND domain specifications, i.e., ε_D, and fair FOND domain specifications, i.e., $\varepsilon_{D,fair}$, are always environment enforceable, and hence consistent environment specifications. However, not every LTL formula is environment enforceable. For instance, if $a, b \in A$ then $\varepsilon = \mathsf{F}\, a$, which says that the agent eventually does action a, is not environment enforceable (indeed, for every environment strategy σ there is a trace in which all actions are b). Note, however, that one can decide if an LTL formula is a consistent environment specification, i.e., if it is environment enforceable, by using standard LTL synthesis (Theorem 1):

Theorem 6. *Deciding if a given LTL formula ε is a consistent environment specification is* 2EXPTIME*-complete.*

5 Planning under LTL environment specifications

In this section we show how to solve planning under LTL environment specifications. We show that we can actually reduce planning for an LTL goal γ under an LTL environment specification ε to Church's synthesis for the implication

$$\varepsilon \supset \gamma.$$

Notice that in this implication, if the environment specification ε is consistent, the agent will not be able to find a strategy (a plan) to make the implication true by falsifying the environment specification ε. The fact that ε is consistent, and hence enviroment enforceable, means that every strategy of the agent can be countered by the environment so as to fulfill ε. Hence, from now on, we focus on consistent environment specifications (in fact, the theorems in this section are trivial in the inconsistent case).

In spite of this, *understanding why the reduction works is not immediate.* After all we are moving from a problem of the form:

$$\text{find } \sigma_{\mathsf{ag}} \text{ such that } \forall \sigma_{\mathsf{env}} \rhd \varepsilon.\, \tau(\sigma_{\mathsf{ag}}, \sigma_{\mathsf{env}}) \models \gamma$$

to a problem of the form:

$$\text{find } \sigma_{\mathsf{ag}} \text{ such that } \sigma_{\mathsf{ag}} \rhd (\varepsilon \supset \gamma),$$

i.e. find σ_{ag} such that for all σ_{env}, if $\tau(\sigma_{\mathsf{ag}}, \sigma_{\mathsf{env}}) \models \varepsilon$ then $\tau(\sigma_{\mathsf{ag}}, \sigma_{\mathsf{env}}) \models \gamma$. In words, instead of requiring the agent to find a strategy that fulfills the goal γ against the environment strategies that satisfy the environment specification ε, we require the agent find one that fulfills the implication $\varepsilon \supset \gamma$ against *all* possible strategies of the environment.

In fact, one direction does hold on a strategy-by-strategy basis.

Theorem 7. *Let ε be an LTL environment specification and γ an LTL goal. Then every agent strategy that enforces $\varepsilon \supset \gamma$ (without environment specifications) also enforces γ under the environment specification ε.*

Proof. Let σ_{ag} be an agent strategy enforcing $\varepsilon \supset \gamma$, i.e., every trace induced by σ_{ag} satisfies $\varepsilon \supset \gamma$. To show that σ_{ag} enforces γ under the environment specification ε, let σ_{env} be an environment strategy enforcing ε, i.e., every trace induced by σ_{env} satisfies ε. In particular, the trace $\tau(\sigma_{\mathsf{ag}}, \sigma_{\mathsf{env}})$ induced by both σ_{ag} and σ_{env} satisfies γ, as required. □

However, the converse does not hold:

Theorem 8. *It is not the case that, for every LTL environment specification ε and LTL goal γ, every agent strategy that enforces γ under the environment specification ε also enforces $\varepsilon \supset \gamma$ (without environment specifications).*

Proof. We show one. Let $A \doteq \{a, b\}$ and $F \doteq \{f\}$, and let $\varepsilon \doteq f \supset a$ and $\gamma \doteq f \supset b$. First note that ε is a consistent LTL environment specification (indeed, the environment can enforce ε by playing $\neg f$ at the first step). Moreover, every environment strategy enforcing ε begins by playing $\neg f$ (since otherwise the agent could play b on its first turn and falsify ε). Thus, every agent strategy enforces γ under the environment specification ε (since the environment's first move is to play $\neg f$ which makes γ true no matter what the agent does). On the other hand, not every agent strategy enforces $\varepsilon \supset \gamma$ (indeed, the agent strategy which plays a on its first turn fails to satisfy the implication on the trace in which the environment plays f on its first turn). □

Notwithstanding the last theorem, the two problems are inter-reducible:

Theorem 9. *Suppose ε is an LTL environment specification. The following are equivalent:*

1. *There is an agent strategy enforcing $\varepsilon \supset \gamma$, in symbols*

$$\exists \sigma_{\mathsf{ag}}.\ \sigma_{\mathsf{ag}} \rhd (\varepsilon \supset \gamma).$$

2. *There is an agent strategy enforcing γ under environment specification ε, in symbols*

$$\exists \sigma_{\mathsf{ag}}.\ \forall \sigma_{\mathsf{env}} \rhd \varepsilon.\, \tau(\sigma_{\mathsf{ag}}, \sigma_{\mathsf{env}}) \models \gamma.$$

Proof. The previous lemma gives us $1 \rightarrow 2$. For the converse, suppose 1 does not hold, i.e., $\varepsilon \supset \gamma$ is not agent enforceable. Now, an immediate consequence of Martin's Borel Determinacy Theorem (Martin 1975) is that for every ϕ in any reasonable specification formalism (including LTL), ϕ is not agent enforceable iff $\neg\phi$ is environment enforceable. Thus, $\neg(\varepsilon \supset \gamma)$ is environment enforceable, i.e., $\exists \sigma_{\mathsf{env}} \forall \sigma_{\mathsf{ag}}.\tau(\sigma_{\mathsf{ag}}, \sigma_{\mathsf{env}}) \models \varepsilon \wedge \neg\gamma$. Note in particular that σ_{env} enforces ε. Now, suppose for a contradiction that 2 holds, and take σ_{ag} enforcing γ under environment specification ε. Then by definition of enforceability under environment specification and using the fact that σ_{env} enforces ε we have that $\tau(\sigma_{\mathsf{ag}}, \sigma_{\mathsf{env}}) \models \gamma$. On the other hand, we have already seen that $\tau(\sigma_{\mathsf{ag}}, \sigma_{\mathsf{env}}) \models \neg\gamma$, a contradiction. □

The above theorem, gives us a way to solve planning for and LTL goal γ under LTL environment specification ε in 2EXPTIME by using Theorem 1: solve classical Church's synthesis for $\varepsilon \supset \gamma$, and return a finite-state strategy (incase the formula is agent-enforceable).

Theorem 10. *Solving planning under LTL environment specifications is* 2EXPTIME*-complete. Moreover, if there is a strategy solving a given instance, then there is a finite-state strategy doing so, and the algorithm returns one such finite-state strategy.*

6 FOND domains with additional LTL assumptions

In this section we study planning under environment specifications that consist of a classical FOND domain with an additional LTL assumption. Such assumptions are, for example, used to specify trajectory constraints as in (Bonet et al. 2017). Notice that we could simply reduce FOND to LTL formulas and work as in the previous sections. However, we will follow a refined approach that will give us tight complexity bounds. To state the theorem we need the following terminology: if D is a FOND domain specification and ε is an LTL formula over variables $F \cup A$ then the *combined* environment specification is the set of environment strategies specified by D that also enforce ε.

Theorem 11. *1. Let D be a FOND domain specification, and ε an LTL formula. Deciding whether the combined environment specification is consistent is* 1EXPTIME*-complete in D and* 2EXPTIME*-complete in ε.*

2. Let D be a FOND domain specification, and ε an LTL formula such that the combined environment specification is consistent. Solving LTL planning under this environment specification for an LTL goal γ is 1EXPTIME*-complete in D and* 2EXPTIME*-complete in the goal γ, and* 2EXPTIME *in the LTL formula ε.*[8]

Note, as an immediate corollary, we establish the complexity of FOND planning with LTL goals (and no environment specifications). While this result was known for goals expressed in LTL on finite traces (LTLf) (De Giacomo and Rubin 2018), it was not known for LTL on infinite traces.

Corollary 12. *Let D be a FOND domain specification. Solving LTL planning under this environment specification for an LTL goal γ is* 1EXPTIME*-complete in D and* 2EXPTIME*-complete in the goal γ.*

Notice that the complexity for solving LTL planning under these environment specifications is the same for standard FOND planning for reachability goals (cf. Proposition 2).

To prove the theorem, first consider the lower-bounds. These follow from known results, i.e., taking reachability goals we get 1EXPTIME-hardness in D, see (Rintanen 2004); and by Theorem 1, taking the universal domain (i.e., every action is always possible, and its effects are all the states), we get 2EXPTIME-hardness in ε for part 1, and for 2EXPTIME-hardness for the goal γ for part 2.

Proving the upper bounds Here we focus on part 2 since part 1 is similar; full details are in the supplement. We make use of automata instead of LTL formulas representing the domain. This mimics a similar development for FOND for LTLf goals in which deterministic finite automata (DFA) and their corresponding games are used (De Giacomo and Rubin 2018). Since we deal with LTL (and not LTLf), we need deterministic automata over *infinite* words. We use deterministic parity word automata (DPW) and games on these.

Deterministic Parity Word automata (DPW) These automata are like (classic) deterministic finite word automata except that there is no set of final states; instead, there is a *priority* function $col : Q \to \mathbb{Z}$ that assigns an integer to every state. A run of a DPW is *accepting* if the infinite sequence of priorities of the states of the run satisfies the *parity condition*, i.e., the largest integer occuring infinitely often is even. The DPW *accepts an input word* if its run is accepting. The *size* of a DPW is the cardinality of its set Q of states, and the number of its priorities is the cardinality of $col(Q)$. DPW are effectively closed under Boolean operations: complementation is done by incrementing every priority by 1 (and thus causes no blowup in the size or number of priorities), while disjunction results in a DPW of size at most $(|Q_1| \times |Q_2|)d^2d!$ and with $O(d)$ many priorities, where $d = |col(Q_1)| + |col(Q_2)|$. It turns out that one can convert LTL formulas into equivalent DPW:

Theorem 13. *(Vardi 1995; Piterman 2007) For every LTL formula φ over variables $X \cup Y$ there is a DPW M_φ over alphabet $2^{X \cup Y}$ that accepts exactly the words satisfying φ. Moreover, the translation is effective, and can be done in* 2EXPTIME *and results in a DPW M_φ of size at most doubly-exponential in $|\varphi|$ and with at most exponentially in $|\varphi|$ many priorities.*

DPW games are played by two players who produce a run in the DPW; the agent chooses the actions and the environment chooses the fluents. The agent is trying to ensure that the run is accepting. To talk about winning the game, we recast the notion of strategy. A run $(q_i, X_i \cup Y_i, q_{i+1})_i$ is *induced* by an agent strategy $\sigma_{\mathsf{ag}} : (2^X)^+ \to 2^Y$ if $\sigma_{\mathsf{ag}}(X_0 X_1 \cdots X_j) = Y_j$ for all j. An agent strategy σ_{ag} *wins the DPW game* if every trace induced by σ_{ag} is accepting, i.e., the sequence of priorities $col(q_0)col(q_1)\cdots$ satisfies the parity condition. Similarly, an environment strategy *wins the DPW game* if every traced induced by it is not accepting. Moreover, since DPW games are determined (exactly one player has a winning strategy), if there is no agent strategy that wins the game then there is an environment strategy that wins the game (and vice versa). Deciding which of the two players (the agent or the environment) has a strategy that wins the game is called *solving* the game.

Theorem 14. *(Zielonka 1998) There is an algorithm for solving DPW games that works in time polynomial in the size $|Q|$ and exponential in the number of priorities $|col(Q)|$, i.e., in time $O(|Q|^{O(|col(Q)|)})$. Moreover, if there is an agent (resp. environment) strategy that wins the game, then there is a finite-state one, and this algorithm returns one such strategy.*

The relationship between DPW games and LTL synthesis is given by the following straightforward lemma:

Lemma 15. *Let φ be an LTL formula over variables $X \cup Y$, and let M be a DPW equivalent to φ (e.g., the DPW from Theorem 13). Then, an agent strategy σ_{ag} wins the game M iff σ_{ag} enforces φ.*

Proof. Let σ_{ag} be an agent strategy. Then, the following are equivalent:

- σ_{ag} wins the game M;
- σ_{ag} enforces the set $L(\mathcal{M})$ of all input words accepted by M;

[8] We do not know if the complexity is 2EXPTIME-hard in ε.

- σ_{ag} enforces the LTL formula φ.

The first equivalence holds because the automaton is deterministic; i.e., every input word has a unique run in M. The second equivalence holds because M accepts exactly the infinite words satisfied by φ. □

Solving LTL planning under combined environment specifications. Let D be a FOND domain specification, and ε an LTL formula such that the combined environment specification is consistent. Let γ be an LTL goal. To solve planning under the combined enviroment specification, it is enough to build a DPW M equivalent to $(\varepsilon_D \wedge \varepsilon) \supset \gamma$, and solve for M. Indeed, some agent strategy solves the DPW game M iff some agent strategy enforces $(\varepsilon_D \wedge \varepsilon) \supset \gamma$ iff some agent strategy enforces γ under the combined environment specification $\varepsilon_D \wedge \varepsilon$ (the first equivalence is the definition of solving the DPW game M, and the second equivalence is by Theorem 8). Moreover, if a specific agent-strategy σ_{ag} solves the DPW game M, then by Theorem 7, σ_{ag} also enforces γ under the environment specification $\varepsilon_D \wedge \varepsilon$.

Similarly, in order to decide if the combined environment specification D and ε is consistent, form the DPW for $\varepsilon_D \wedge \varepsilon$ of size 1exp in the size of D, and 2 in the size of ε, and with 1exp many priorities in the size of ε, and solve the DPW game. Note that if the agent does not have a strategy that wins this DPW game, then the environment does, and vice versa. This property of games is called Determinacy, and it holds for DPW games (Zielonka 1998).

The main problem, now, is to build the DPW M given D, ε and γ. If one naively applies the translation in Theorem 13 of the LTL formula $(\varepsilon_D \wedge \varepsilon) \supset \gamma$, and then solves the game using Theorem 14, one pays 2exp in the size of D, ε and γ. However, the promised complexity (Theorem 11) is 1exp in the size of D and 2exp in the formulas ε and γ. In order to achieve this complexity, it is enough to construct a DPW M equivalent to $(\varepsilon_D \wedge \varepsilon) \supset \gamma$ that has 1exp many states in the size of D, and 2exp many states and 1exp many priorities in the sizes of ε and γ. This can be done by translating ε and γ into DPWs using Theorem 13, and directly building a DPW M_D accepting the infinite traces of D of size single-exponential in D with two priorities, and then forming a DPW by taking Boolean combinations.

We end this sketch by explaining how to define the DPW M_D for D. It has state set $(S \times A) \cup \{q_{in}, q_{-}\}$ and works as follows: the initial state is q_{in}, and on reading the input $[e, a]$ it goes to state (e, a) if $e \in I$ and to q_{-} otherwise; from a state of the form (e, a) and on reading the input $[e', a']$ the DPW goes to state (e', a') if (a' is possible in e) implies (e' is a successor state of e under effects eff_a); and it goes to the rejecting sink q_{-} if a' is possible in e but e' is not a successor of state e under effects eff_a. This DPW requires just two priorities, i.e., the reject sink q_{-} has priority 1 and all states have priority 0.

This completes the sketch of the proof of Theorem 11. We mention that it is unknown if a similar technique can be applied to solve FOND fair planning (under combined environment specifications).

7 Related Work in Formal Methods

Our notion of environment specification is related to that of assumptions in the Formal Methods literature, and specifically in synthesis. The most well-known assumption is that of fairness. Synthesis under fairness was discussed and solved using automata-theoretic techniques in (Vardi 1995). More general assumptions appear in works that show how to use Church's synthesis to synthesize high-level robot controllers (Kress-Gazit, Fainekos, and Pappas 2009; Kress-Gazit, Wongpiromsarn, and Topcu 2011). The typical use of Church's Synthesis is to synthesize for the implication $Assumption \supset Goal$ (or some variant of this). Our work shows that one can reduce planning under environment specifications to Church's synthesis of such an implication formula (Theorem 9). However, this reduction does not hold on a strategy-by-strategy basis (Theorem 8). This clarifies a subtle point that has been missing from the Verification literature (Chatterjee and Henzinger 2007; Chatterjee, Henzinger, and Jobstmann 2008; Bloem, Ehlers, and Könighofer 2015; Brenguier, Raskin, and Sankur 2017). Also, the undesirable drawback of the agent being able to falsify an assumption when synthesizing $Assumption \supset Goal$ is well known, and it has been observed that it can be overcome by requiring the $Assumption$ to be environment enforceable (D'Ippolito et al. 2013; Chatterjee and Henzinger 2007). Our work (and Definition 3 in particular), clarifies that this requirement is a consistency condition for assumptions.

In fact, in the Verification literature, there are other approaches for synthesizing $Assumption \supset Goal$, especially in presence of multiple agents (the environment being one of them). These include identifying various "degrees of cooperation" required amongst agents to satisfy the assumptions (Chatterjee and Henzinger 2007; Bloem, Ehlers, and Könighofer 2015; Brenguier, Raskin, and Sankur 2017). We do not address multiple agent in this work; however, we believe that approaches similar to the present one may help in establishing principled foundations on which to ground assumptions in the multi-agent case.

8 Conclusion

This works gives a unified way to think about many kinds of assumptions that an agent makes about the environment. Thus, domain specifications, fairness constraints, trajectory constraints, etc., should all be viewed as the same type of object, i.e., as sets of environment strategies. We believe that this brings a principled clarity to the notion of "environment specification", which in literature has been formalized for specific cases but left at an intuitive level only in general.

Environment specifications could also be specified in branching-time formalisms such as CTL^*, μ-calculus, and tree automata (indeed, strategies are naturally viewed as infinite trees). The characteristic of such formalisms is that they allow one to describe possible choices that an agent or environment has. We leave this for future work.

References

Bacchus, F., and Kabanza, F. 2000. Using temporal logics to express search control knowledge for planning. *Artif. Intell.* 116(1-2):123–191.

Bloem, R.; Ehlers, R.; and Könighofer, R. 2015. Cooperative reactive synthesis. In *ATVA*, 394–410.

Bonet, B., and Geffner, H. 2015. Policies that generalize: Solving many planning problems with the same policy. In *IJCAI*, 2798–2804.

Bonet, B.; De Giacomo, G.; Geffner, H.; and Rubin, S. 2017. Generalized planning:non-deterministic abstractions and trajectory constraints. In *IJCAI*, 873–879.

Brenguier, R.; Raskin, J.; and Sankur, O. 2017. Assume-admissible synthesis. *Acta Inf.* 54(1):41–83.

Camacho, A.; Triantafillou, E.; Muise, C.; Baier, J. A.; and McIlraith, S. 2017. Non-deterministic planning with temporally extended goals: LTL over finite and infinite traces. In *AAAI*.

Chatterjee, K., and Henzinger, T. A. 2007. Assume-guarantee synthesis. In *TACAS*.

Chatterjee, K.; Henzinger, T. A.; and Jobstmann, B. 2008. Environment assumptions for synthesis. In *CONCUR*, 147–161.

Church, A. 1963. Logic, arithmetics, and automata. In *Proc. Int. Congress of Mathematicians, 1962*.

Cimatti, A.; Pistore, M.; Roveri, M.; and Traverso, P. 2003. Weak, strong, and strong cyclic planning via symbolic model checking. *Artif. Intell.* 1–2(147).

Daniele, M.; Traverso, P.; and Vardi, M. Y. 1999. Strong cyclic planning revisited. In *ECP*, 35–48.

De Giacomo, G., and Rubin, S. 2018. Automata-theoretic foundations of fond planning for LTL_f/LDL_f goals. In *IJCAI*.

De Giacomo, G.; Murano, A.; Rubin, S.; and Stasio, A. D. 2016. Imperfect-information games and generalized planning. In *IJCAI*, 1037–1043.

D'Ippolito, N.; Braberman, V. A.; Piterman, N.; and Uchitel, S. 2013. Synthesizing nonanomalous event-based controllers for liveness goals. *ACM Trans. Softw. Eng. Methodol.* 22(1):9:1–9:36.

D'Ippolito, N.; Rodríguez, N.; and Sardiña, S. 2018. Fully observable non-deterministic planning as assumption-based reactive synthesis. *J. Artif. Intell. Res.* 61:593–621.

Eisner, C., and Fisman, D. 2006. *A practical introduction to PSL*. Springer.

Geffner, H., and Bonet, B. 2013. *A Coincise Introduction to Models and Methods for Automated Planning*. Morgan & Claypool.

Gerevini, A.; Haslum, P.; Long, D.; Saetti, A.; and Dimopoulos, Y. 2009. Deterministic planning in the fifth international planning competition: PDDL3 and experimental evaluation of the planners. *Artif. Intell.* 173(5-6):619–668.

Ghallab, M.; Nau, D. S.; and Traverso, P. 2004. *Automated planning – Theory and Practice*. Elsevier.

Green, C. 1969. Theorem proving by resolution as basis for question-answering systems. In *Machine Intelligence*, volume 4. American Elsevier. 183–205.

Kress-Gazit, H.; Fainekos, G. E.; and Pappas, G. J. 2009. Temporal-logic-based reactive mission and motion planning. *IEEE Trans. Robotics* 25(6):1370–1381.

Kress-Gazit, H.; Wongpiromsarn, T.; and Topcu, U. 2011. Correct, reactive, high-level robot control. *IEEE Robotics Automation Magazine* 18(3):65–74.

Lin, F., and Levesque, H. J. 1998. What robots can do: Robot programs and effective achievability. *Artif. Intell.* 101(1-2):201–226.

Martin, D. A. 1975. Borel determinacy. *Annals of Mathematics* 363–371.

McCarthy, J. 1957. Programs with common sense. In *Proceedings of the Teddington Conference on the Mechanization of Thought Processes*, 756–791.

Patrizi, F.; Lipovetzky, N.; and Geffner, H. 2013. Fair LTL synthesis for non-deterministic systems using strong cyclic planners. In *IJCAI*.

Piterman, N. 2007. From nondeterministic büchi and streett automata to deterministic parity automata. *Logical Methods in Computer Science* 3(3).

Pnueli, A., and Rosner, R. 1989. On the synthesis of a reactive module. In *POPL*.

Reiter, R. 2001. *Knowledge in Action: Logical Foundations for Specifying and Implementing Dynamical Systems*. The MIT Press.

Rintanen, J. 2004. Complexity of planning with partial observability. In *ICAPS*.

Vardi, M. Y. 1995. An automata-theoretic approach to fair realizability and synthesis. In *CAV*, 267–278.

Zielonka, W. 1998. Infinite games on finitely coloured graphs with applications to automata on infinite trees. *Theor. Comput. Sci.* 200(1-2):135–183.