Planning and Synthesis Under Assumptions

Benjamin Aminof  
Technische Univ. Wien  
Vienna, 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

In Reasoning about Action and Planning, one synthesizes the agent plan by taking advantage of the assumption on how the environment works (that is, one exploits the environment’s effects, its fairness, its trajectory constraints). In this paper we study this form of synthesis in detail. We consider assumptions as constraints on the possible strategies that the environment can have in order to respond to the agent’s actions. Such constraints may be given in the form of a planning domain (or action theory), as linear-time formulas over infinite or finite runs, or as a combination of the two (e.g., FOND under fairness). We argue though that not all assumption specifications are meaningful: they need to be consistent, which means that there must exist an environment strategy fulfilling the assumption in spite of the agent actions. For such assumptions, we study how to do synthesis/planning for agent goals, ranging from a classical reachability to goal on traces specified in LTL and LTLf/LDLf, characterizing the problem both mathematically and algorithmically.

Introduction

Reasoning about actions and planning concern the representation of a dynamic system. This representation consists of a description of the interaction between an agent and its environment and aims at enabling reasoning and deliberation on the possible course of action for the agent (Reiter 2001). Planning in fully observable nondeterministic domains (FOND), say in Planning Domain Definition Language (PDDL), (Ghallab, Nau, and Traverso 2004; Geffner and Bonet 2013) exemplifies the standard methodology for expressing dynamic systems: it represents the world using finitely many fluents under the control of the environment and a finitely many actions under the control of the agent. Using these two elements a model of the dynamics of world is given. Agent goals, e.g., reachability objectives, or, say, temporally extended objectives written in LTL (Bacchus and Kabanza 2000; Camacho et al. 2017; De Giacomo and Rubin 2018), are expressed over such models in terms of such fluents and actions.

An important observation is that, in devising plans, the agent takes advantage of such a representation of the world. Such a representation corresponds to knowledge that the agent has of the world. In other words, the agent assumes that the world works in a certain way, and exploits such an assumption in devising its plans. A question immediately comes to mind:

Which kinds of environment assumptions can the agent make?

Obviously the planning domain itself (including the initial state) with its preconditions and effects is such an assumption. That is, as long as the agent sticks to its preconditions the environment acts as described by the domain. So, the agent can exploit the effect of its in order to reach a certain goal (state of affairs). Another common assumption is to assume the domain is fair, i.e., so-called fair FOND (Daniele, Traverso, and Vardi 1999; Pistore and Traverso 2001; 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 continuing to execute an action from a given state the environment will eventually respond with all its possible nondeterministic effects. More recently (Bonet and Geffner 2015; Bonet et al. 2017) trajectories constraints over the domain, expressed in LTL, have been proposed to model more general restrictions on the possible environment behavior. But is any kind of LTL formula on the fluents and actions of the domain a possible trajectory constrain for the environment? The answer is obviously not! To see this, consider a formula expressing that eventually a certain possible action must actually be performed (the agent may decide not to do it). But then

Which trajectory constraints are suitable as assumptions in a given domain?

In fact, we observe that the domain itself can be expressed in LTL, as can its fairness (as we illustrate later). So the question can be rephrased as:

Can any linear-time specification be used as an assumption for the environment?

So the ultimate question is

What is an environment assumption?

This is what we investigate in this paper. We take the view that environment assumptions, i.e., domains, fairness, LTL trajectory constraints, etc., are ways to talk about the set of
strategies the environment can enact. And when we synthesize the plan for a goal, i.e., the agent strategy for fulfilling the goal, such a strategy needs do so only against the strategies of the environment from the given set of environment strategies. We formalize this insight and define synthesis/planning under assumptions and the relationship between the two in a general linear-time setting. In particular, our definitions only allow linear-time properties to be assumptions if the environment can enforce them. In doing this we answer the above questions.

We also concretize the study and express goals and assumptions in LTL, automata over infinite words (deterministic parity word automata) (Grädel, Thomas, and Wilke 2002), as well as formalisms over finite traces, i.e., LTL/LDL (De Giacomo and Vardi 2013, De Giacomo and Rubin 2018) and finite word automata. This allows us to study how to solve synthesis/planning under assumptions problems. One may think that the natural way to solve such synthesis problems is to have the agent synthesize a strategy for the implication

\[ \text{Assumption} \supset \text{Goal} \]

where both Assumption and Goals are expressed, say, in LTL. A first problem with such an implication is that the agent should not devise strategies that make Assumption false, because in this case the agent would lose its model of the world without necessarily fulfilling its Goal. This undesirable situation is avoided by our very notion of environment assumption. A second issue is this:

Does synthesis/planning under assumptions amount to synthesizing for the above implication?

We show that this is not the case. Note that an agent that synthesizes for the implication is too pessimistic: the agent, having chosen a candidate agent strategy, considers as possible all environment strategies that satisfy Assumption against the specific candidate strategy it is analyzing. But, in this way the agent gives too much power to the environment, since, in fact, the environment does not know the agent’s chosen strategy. On the other hand, surprisingly, we show that if there is an agent strategy fulfilling Goal under Assumption, then also there exists one that indeed enforces the implication. Thus, even if the implication cannot be used for characterizing the problem of synthesis/planning under assumptions, it can be used to solve it. Exploiting this result, we devise techniques to solve synthesis/planning under assumptions, and study the worst case complexity of the problems when goals and assumptions are expressed in the logics and automata mentioned above.

Synthesis and Linear-time specifications

Synthesis is the problem of producing a module that satisfies a given property no matter how the environment behaves (Pnueli and Rosner 1989). Synthesis can be thought of in the terminology of games. Let Var be a finite set of Boolean variables (also called atoms), and assume it is partitioned into two sets: \( A \), those controllable by the agent, and \( E \), those controllable by the environment. Let \( A = 2^A \) be the set of actions and \( E = 2^E \) the set of environment states (note the symmetry: we think of \( A \) as a set of actions that are compactly represented as assignments of the variables in \( A \)). The game consists of infinitely many phases. In each phase of the game, both players assign values to their variables, with the environment going first. These assignments are given by strategies: an agent strategy \( \sigma_{ag} : E^+ \to A \) and an environment strategy \( \sigma_{env} : A^* \to E \). The resulting infinite sequence of assignments is denoted \( \pi_{\sigma_{ag},\sigma_{env}} \).

In classic synthesis the agent is trying to ensure that the produced sequence satisfies a given linear-time property. In what follows we write LT to denote a generic formalism for defining linear-time properties. Thus, the reader may substitute their favorite formalism for LT, e.g., LT = LTL. We use logical notation throughout, i.e., \( \phi \) refers to logical formulas/automata, and \( \phi_1 \land \phi_2 \) refers to conjunction/intersection, etc. If \( \phi \in \text{LT} \) write \( [\phi] \in (2^{\text{Var}})^\omega \) for the set it defines.

We say that \( \sigma_{ag} \) realizes \( \phi \) (written \( \sigma_{ag} \models \phi \)) if \( \forall \sigma_{env}, \pi_{\sigma_{ag},\sigma_{env}} \in [\phi] \).
Similarly, we say that \( \sigma_{env} \) realizes \( \phi \) (written \( \sigma_{env} \models \phi \)) if \( \forall \sigma_{ag}, \pi_{\sigma_{ag},\sigma_{env}} \in [\phi] \). We write \( \text{Str}_{\text{env}}(\phi) \) (resp. \( \text{Str}_{\text{ag}}(\phi) \)) for the set of environment (resp. agent) strategies that realize \( \phi \), and in case this set is non-empty we say that \( \phi \) is environment (resp. agent) realizable.

We write \( \text{Str}_{\text{env}}(\phi) \) (resp. \( \text{Str}_{\text{ag}}(\phi) \)) for the set of all environment (resp. agent) strategies.

Solving LT environment- (resp. agent-) synthesis asks, given \( \phi \in \text{LT} \) to decide if \( \phi \) is environment- (resp. agent-) realizable, and to return such a finite-state strategy (if one exists). In other words, realizability is the recognition problem associated to synthesis. We now recall two concrete specification formalisms, LTL and DPW, and then state that results about solving LTL/DPW synthesis.

Linear-temporal Logic (LTL) Formulas of LTL(Var), or simply LTL, are generated by the following grammar:

\[
\varphi ::= p \mid \varphi \lor \varphi \mid \neg \varphi \mid X\varphi \mid \varphi U \varphi
\]

where \( p \in \text{Var} \). The size \( |\varphi| \) of a formula \( \varphi \) is the number of symbols in it. LTL formulas are interpreted over infinite sequences \( \pi \in (2^{\text{Var}})^\omega \). Define the satisfaction relation \( |\pi| \models \varphi \) as follows:

\[
\begin{align*}
\pi, i &\models p \text{ if } p \in \pi, i \lor \pi, i &\models \varphi \lor \varphi \text{ if } i < n \text{ for some } i \in \{1, 2\} \lor \pi, i &\models \neg \varphi \text{ if it is not the case that } \pi, i &\models \varphi \lor \varphi \text{ if } i < n \text{ such that } \pi, i &\models \varphi \text{ and for all } i \leq j < n, \pi, j &\models \varphi \text{ Write } \pi, i &\models \varphi \text{ if } \pi, 0 \models \varphi \text{ and say that } \pi &\models \varphi \text{ is the set of descriptions of } \pi \models \varphi \text{ for the usual abbreviations, } \varphi \lor \varphi \models \varphi \lor \varphi \text{, true := } p \lor \neg p \text{, false := } \neg \text{true, F } &\models \text{true U } \varphi \text{, } G \varphi \models \neg \text{F } \varphi \text{. Write Bool}(\text{Var}) \text{ for the set of Boolean formulas over } \text{Var} \text{. We remark that every result in this paper that mentions LTL also holds for LDL (linear dynamic logic) (Vardi 2011) Eisner and Fisman 2006).}
\end{align*}
\]
Deterministic Parity Word Automata (DPW) A DPW over Var is a tuple $M = (Q, q_0, T, \omega)$ where $Q$ is a finite set of states, $q_0 \in Q$ is a set of initial states, $T : Q \times 2^{\text{Var}} \rightarrow Q$ is the transition function, and $\omega : Q \rightarrow \mathbb{Z}$ is the coloring. The run $\rho$ of $M$ on the input word $x_0x_1x_2 \cdots \in (2^{\text{Var}})^\omega$ is the infinite sequence of transitions $(q_0, x_0, q_1)(q_1, x_1, q_2)(q_2, x_2, q_3) \cdots$ such that $q_0 = q_0$. A run is successful if the largest color occurring infinitely often is even. In this case, we say that the input word is accepted. The DPW $M$ defines the set $\{[M]\}$ consisting of all input words it accepts. The size of $M$, written $|M|$, is the cardinality of $Q$. The number of colors of $M$ is the cardinality of $\omega(Q)$.

DPWs are effectively closed under Boolean operations, see e.g., (Grädel, Thomas, and Wilke 2002). Moreover, the size (resp. number of colors) of the resulting automata are polynomial in the sizes (resp. number of colors) of the input automata. Every LTL formula $\varphi$ can be translated into an equivalent DPW $M$, i.e., $[[\varphi]] = [[M]]$, see e.g. (Vardi 1995, Piterman 2007). Moreover, the cost of this and the size of $M$ are at most doubly exponential in the size of $\varphi$, and the number of colors of $M$ is at most singly exponential in the size of $\varphi$.

Finally, we state known results about solving synthesis:

Theorem 1 (Solving Synthesis).

1. Solving LTL environment (resp. agent) synthesis is $2\text{EXPTIME}$-complete (Pnueli and Rosner 1989).
2. Solving DPW environment (resp. agent) synthesis is $\text{PTIME}$ in the size of the automaton and $\text{EXPTIME}$ in the number of its colors (Finkbeiner 2007).

Synthesis under Assumptions

In this section we give core definitions of environment assumptions and synthesis under such assumptions. Intuitively, the assumptions are used to select the environment strategies that the agent considers possible, i.e., although the agent does not know the particular environment strategy it will encounter, it knows that it comes from such a set. We begin in the abstract, and then move to declarative specifications. Unless explicitly specified, we assume fixed sets $E$ and $A$ of environment and agent atoms.

Here are the main definitions of this paper:

Definition 1 (Environment Assumptions – abstract). We call any non-empty set $\Omega \subseteq \text{Str}_{\text{env}}$ of environment strategies an environment assumption.

Informally, the set $\Omega$ represents the set of environment strategies that the modeler considers possible.

Definition 2 (Agent Goals – abstract). We call any set $\Gamma$ of traces an agent goal.

Definition 3 (Synthesis under assumptions – abstract). Let $\Omega$ be an environment assumption and $\Gamma$ an agent goal. We say that an agent strategy $\sigma_{ag}$ realizes $\Gamma$ assuming $\Omega$ if

$$\forall \sigma_{\text{env}} \in \Omega, \pi_{\sigma_{ag}, \sigma_{\text{env}}} \in \Gamma$$

Remark 1 (On the non-emptiness of $\Omega$). Note that the requirement that $\Omega$ be non-empty is a consistency requirement; if it were empty then there would be no $\pi_{\sigma_{ag}, \sigma_{\text{env}}}$ to test for membership in $\Gamma$ and so synthesis under assumptions would trivialize and all agent strategies would realize all goals.

For the rest of this paper we will specify agent goals and environment assumptions as linear-time properties.

Let LT be a formalism for specifying linear-time properties over Var, e.g., LT = LTL or LT = DPW. How should $\omega \in \text{LT}$ determine an assumption $\Omega$? In general, $\omega$ talks about the interaction between the agent and the environment. However, we want that the agent can be guaranteed that whatever it does the resulting play satisfies $\omega$. Thus, a given $\omega$ induces the set $\Omega$ consisting of all environment strategies $\sigma_{\text{env}}$ such that for all agent strategies $\sigma_{ag}$ the resulting trace satisfies $\omega$. In particular, for $\Omega$ to be non-empty (as required for it to be an environment assumption) we must have that $\omega$ is environment realizable.

Definition 4 (Synthesis under Assumptions – linear-time).

1. We call $\omega \in \text{LT}$ an environment assumption if it is environment realizable.
2. We call any $\gamma \in \text{LT}$ an agent goal.
3. An LT synthesis under assumptions problem is a tuple $P = (E, A, \omega, \gamma)$ where $\omega \in \text{LT}$ is an environment assumption and $\gamma \in \text{LT}$ is an agent goal.
4. We say that an agent strategy $\sigma_{ag}$ realizes $\gamma$ assuming $\omega$, or that it solves $P$, if $\forall \pi_{\sigma_{ag}, \sigma_{\text{env}}} \in \Omega, \pi_{\sigma_{ag}, \sigma_{\text{env}}} \models \gamma$.
5. The corresponding decision problem is to decide, given $P$, if there is an agent strategy solving $P$.

For instance, solving LTL synthesis under assumptions means, given $P = (E, A, \omega, \gamma)$ with environment assumption $\omega \in \text{LTL}(E \cup A)$ and agent goal $\gamma \in \text{LTL}(E \cup A)$, to decide if there is an agent strategy solving $P$, and to return such a finite-state strategy (if one exists). We remark that solving LTL synthesis under assumptions is not immediate; we will provide algorithms in the next section. For now, we point out that deciding whether $\omega$ is an environment assumption amounts to checking if $\omega$ is environment realizable, itself a problem that can be solved by known results (i.e., Theorem 1).

Theorem 2. 1. Deciding if an LTL formula is an environment assumption is $2\text{EXPTIME}$-complete.
2. Deciding if a DPW is an environment assumption is in $\text{PTIME}$ in the size of the DPW and exponential in its number of colors.

We illustrate such notions with some examples.

Example 1. 1. The set $\Omega = \text{Str}_{\text{env}}$, definable in LTL by the formula $\omega \models \text{true}$, is an environment assumption. It captures the situation that the agent assumes that the environment will use any of the strategies in $\text{Str}_{\text{env}}$.
2. In robot-action planning problems, typical environment assumptions 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”. The set $\Omega$ of environment strategies that realize these properties is an environment
Solving Synthesis under Assumptions

In this section we show how to solve synthesis under assumptions when the environment assumptions and agent goals are given in LTL or by DPW. The general idea is to reduce synthesis under assumptions to ordinary synthesis, i.e., synthesis of the implication $\omega \supset \gamma$. Although correct, understanding why it is correct is not immediate.

**Lemma 3.** Let $\omega \in \text{LT}$ be an environment assumption and $\gamma \in \text{LT}$ an agent goal. Then, every agent strategy that realizes $\omega \supset \gamma$ also realizes $\gamma$ assuming $\omega$.

**Proof.** Let $\pi_{ag}$ be an agent strategy realizing $\omega \supset \gamma$ (a). To show that $\pi_{ag}$ realizes $\gamma$ assuming $\omega$ let $\pi_{env}$ be an environment strategy realizing $\omega$ (b). Now consider the trace $\pi = \pi_{ag} \pi_{env}$. We must show that $\pi$ satisfies $\gamma$. By (a) $\pi$ satisfies $\omega \supset \gamma$ and by (b) $\pi$ satisfies $\omega$. $\Box$

We now observe that the converse is not true. Consider $A = \{x\}$ and $E = \{y\}$, and let $\omega = y \supset x$ and $\gamma = y \supset \neg x$. First note that $\omega$ is an environment assumption formula (indeed, the environment can realize $\omega$ by playing $\neg y$ at the first step). Moreover, every environment strategy realizing $\omega$ begins by playing $\neg y$ (since otherwise the agent could play $\neg x$ on its first turn and falsify $\omega$). Thus, every agent strategy realizes $\gamma$ assuming $\omega$ (since the environment’s first move is to play $\neg y$ which makes $\gamma$ true no matter what the agent does). On the other hand, not every agent strategy realizes $\omega \supset \gamma$ (indeed, the strategy which plays $x$ on its first turn fails to satisfy the implication on the trace in which the environment plays $y$ on its first turn). In spite of the failure of the converse, the realizability problems are inter-reducible.

**Theorem 4.** Suppose $\omega \in \text{LT}$ is an environment assumption. The following are equivalent:

1. There is an agent strategy realizing $\omega \supset \gamma$.
2. There is an agent strategy realizing $\gamma$ assuming $\omega$.

**Proof.** The previous lemma gives us $1 \rightarrow 2$. For the converse, suppose 1 does not hold, i.e., $\omega \supset \gamma$ is not agent-realizable. Now, an immediate consequence of Martin’s Borel Determinacy Theorem (Martin 1975) is that for every $\phi$ in any reasonable specification formalism (including all the ones mentioned in this paper), $\phi$ is not agent-realizable iff $\neg \phi$ is environment realizable. Thus, $\neg (\omega \supset \gamma)$ is environment-realizable, i.e., $\exists \pi_{env} \exists \pi_{ag} \pi_{ag} \pi_{env} \models \omega \land \neg \gamma$. Note in particular that $\pi_{env}$ realizes $\omega$, i.e., $\pi_{env} \models \omega$. Now, suppose for a contradiction that 2 holds, and take $\pi_{ag}$ realizing $\gamma$ assuming $\omega$. Then by definition of realizability under assumptions and using the fact that $\pi_{env} \models \omega$ we have that $\pi_{ag} \pi_{env} \models \gamma$. On the other hand, we have already seen that $\pi_{ag} \pi_{env} \models \neg \gamma$, a contradiction. $\Box$

Moreover, we see that one can actually extract a strategy solving LTL synthesis by assumptions simply by extracting a strategy for solving the implication $\omega \supset \gamma$, which itself can be done by known results (i.e., Theorem 1).

**Theorem 5.** Solving LTL synthesis under assumptions is $2\text{EXPTIME}$-complete.

2. Solving DPW synthesis under assumptions is in PTIME in the size of the automaton and in EXPTIME in the number of colors of the automaton.

Planning under Assumptions

In this section we define planning under assumptions, that is, synthesis wrt a domain $D$. We begin with a representation of fully-observable non-deterministic (FOND) domains (Ghalabi, Nau, and Traverso 2004; Geffner and Bonet 2013). Our representation considers actions symmetrically to fluents, i.e., as assignments to certain variables.

A domain $D = (E, A, I, \text{Pre}, \Delta)$ consists of:

1. A non-empty set $E$ of environment Boolean variables, also called fluents; the elements of $E = 2^E$ are called environment states,
2. A non-empty set $A$ (disjoint from $E$) of action Boolean variables; the elements of $A = 2^A$ are called actions,
3. A non-empty set $I \subseteq E$ of initial environment states,
4. A relation $\text{Pre} \subseteq E \times A$ of available actions such that for every $s \in E$ there is an $a \in A$ with $(s, a) \in \text{Pre}$ (we say that $a$ is available in $s$), and
5. A relation $\Delta \subseteq E \times A \times E$ such that $(s, a, t) \in \Delta$ implies that $(s, a) \in \text{Pre}$.

As is customary in planning and reasoning about actions, we assume domains are represented compactly by tuples $(E, A, I, \text{init}, \text{pre}, \delta)$ where $\text{init} \in \text{Bool}(E)$, $\text{pre} \in \text{Bool}(E \cup A)$, and $\delta \in \text{Bool}(E \cup A \cup E')$ (here $E' = \{e' : e \in E\}$). This data induces the domain $(E, A, I, \text{Pre}, \Delta)$ where

1. $s \in I$ iff $s \models \text{init}$,
2. $(s, a) \in \text{Pre}$ iff $s \cup a \models \text{pre}$,
3. $(s, a, t) \in \Delta$ iff $s \cup a \cup \{e' : e \in t\} \models \delta$.

We emphasize that when measuring the size of $D$ we use this compact representation:

**Definition 5.** The size of $D$, written $|D|$, is $|E| + |A| + |\text{init}| + |\text{pre}| + |\delta|$.

$^2$Domains can be thought of as compact representations of the arenas in games on graphs (Grädel, Thomas, and Wilke 2002). The player chooses actions, also represented compactly, and the environment resolves the nondeterminism. In addition, not every action needs to be available in every vertex of the arena.
We remark that in PDDL action preconditions are declared using :precondition, conditional effects using the when operator, and nondeterministic outcomes using the oneof operator (note that we code actions with action variables).

**Example 2** (Universal Domain). Given $E$ and $A$ define the universal domain $U = (E,A,I,Pre,\Delta)$ where $I = E$, $Pre = E \times A$ and $\Delta = E \times A \times E$.

We now define the set of environment strategies induced by a domain. We do this by describing a property $\omega_D$, that itself can be represented in LTL and DPW, as shown below.

**Definition 6.** Fix a domain $D$. Define a property $\omega_D$ (over atoms $E \cup A$) as consisting of all traces $\pi = \pi_0\pi_1\ldots$ such that

1. $\pi_0 \in I$ and
2. for all $n \geq 1$, if $\pi_i \cap A$ is available in $\pi_i \cap E$ for every $i \in [0,n-1]$ then $(\pi_{i-1} \cap E, \pi_{i-1} \cap A, \pi_i \cap E) \in \Delta$.

Observe that $\omega_D$ is an environment assumption since, by the definition of domain, whenever an action is available in a state there is at least one possible successor state. Intuitively, an environment strategy $\sigma_{env} : A^* \to E$ is in $Str_{env}(\omega_D)$ if i) its first move is to pick an initial environment state, and ii) thereafter, if the current action $a$ is available in the current environment state $x$ (and the same holds in all earlier steps) then the next environment state $z \in E$ is constrained so that $(x,y,z) \in \Delta$. Notice that $\sigma_{env}$ is unconstrained the moment $a$ is not available in $x$, e.g., in PDDL these would be actions for which the preconditions are not satisfied. Intuitively, this means that it is in the interest of the agent to play available actions because otherwise the agent can’t rely on the trace coming from the domain.

**Remark 2.** The reader may be wondering why the above definition does not say i’) $\pi_0 \in I$ and ii’) for all $n \geq 1$, $(\pi_{n-1} \cap E, \pi_{n-1} \cap A, \pi_n \cap E) \in \Delta$. Consider the linear-time property $\omega_D'$ consisting of traces $\pi$ satisfying i’ and ii’. Observe that, in general, $\omega_D'$ is not environment realizable. Indeed, condition ii’ implies that $\pi_0 \cap A$ is available in $\pi_0 \cap E$. However, no environment strategy can force the agent to play an available action.

We now observe that one can express $\omega_D$ in LTL.

**Lemma 6.** For every domain $D$ there is an LTL formula equivalent to $\omega_D$. Furthermore, the size of the LTL formula is linear in the size of $D$.

To see this, say $D = (E,A,I,Pre,\Delta)$ is represented by $(E,A,init,pre,\delta)$. For the LTL formula, let $\delta'$ be the LTL$(E \cup A)$ formula formed from $\delta$ by replacing every term of the form $e'$ by $Xe$. Note that $(\pi, n) \models \delta'$ iff $(\pi_n \cap E, \pi_n \cap A, \pi_{n+1} \cap E) \in \Delta$. The promised LTL$(E \cup A)$ formula is

$$init \land (G\delta' \lor \delta' \land \neg pre).$$

One can also express $\omega_D$ directly by a DPW.

**Lemma 7.** For every domain $D$ there is a DPW $M_D$ equivalent to $\omega_D$. Furthermore, the size of the DPW is at most exponential in the size of $D$ and has two colors.

To do this we define the DPW directly rather than translate the LTL formula (which would give a double exponential bound). Define the DPW $M_D = (Q,q_0,T,col)$ over $E \cup A$ as follows. Introduce fresh symbols $q_{in},q_{+},q_-$. Let $q_{in}$ be the initial state. Define $Q = \{q_{in},q_{+},q_-\} \cup (E \times A)$, Define $col(q_{in}) = 1$, and $col(q) = 0$ for all $q \neq q_{in}$. For all $e,e' \in E, a,a' \in A$ the transitions are given in Table 1. Intuitively, on reading the input $e' \cup a'$ the DPW goes to the rejecting sink $q_-$ if $\Delta$ (resp. $I$) is not respected, it goes to the accepting sink $q_+$ if $\Delta$ (resp. $I$) is respected but $Pre$ is not, and otherwise it continues (and accepts).

| Transition | Description |
|------------|-------------|
| $q_{in} \rightarrow e' \cup a'$ | $q_-$ if $e' \notin I$ |
| $q_{in} \rightarrow e' \cup a'$ | $(e',a')$ if $e' \in I$ and $(e',a') \in Pre$ |
| $q_{in} \rightarrow e' \cup a'$ | $q_+$ if $(e',a') \notin Pre$ |
| $(e,a) \rightarrow e' \cup a'$ | $q_-$ if $(e,a,e') \notin \Delta$ |
| $(e,a) \rightarrow e' \cup a'$ | $(e',a')$ if $(e,a,e') \in \Delta$ and $(e',a') \in Pre$ |
| $(e,a) \rightarrow e' \cup a'$ | $q_+$ if $(e,a,e') \in \Delta$ and $(e',a') \notin Pre$ |

Table 1: Transitions for DPW for $\omega_D$

**Definition 7.** Let $D$ be a domain.

- A set $\Omega \subseteq Str_{env}$ is an environment assumption for the domain $D$ if $Str_{env}(\omega_D) \cap \Omega$ is non-empty.
- $\omega \in \text{LT}$ is an environment assumption for the domain $D$ if $Str_{env}(\omega_D) \cap Str_{env}(\omega)$ is non-empty, i.e., if $Str_{env}(\omega)$ is an environment assumption for the domain $D$.

We illustrate the notion with some examples.

**Example 3.**

1. $\omega = \text{true}$ is an environment assumption for $D$ since $\omega_D \land \omega = \omega_D$ is environment realizable.

2. Let $\omega_D,\omega_D,\text{fair}$ denote the following property: $\pi \in \omega_D,\text{fair}$ iff for all $(s,a) \in \text{Pre}$, if there are infinitely many $n$ such that $s = \pi_n \cap E, a = \pi_n \cap A$ then for every $t \in E$ with $(s,a,t) \in \Delta$ there are infinitely many $n$ such that $s = \pi_n \cap E, a = \pi_n \cap A$ and $t = \pi_n+1 \cap E$. In words this says that if a state-action pair occurs infinitely often, then infinitely often this is followed by every possible effect. Note that $\omega_D,\text{fair}$ is an environment assumption for domain $D$ since, e.g., the strategy that resolves the effects in a round-robin way realizes $\omega_D \land \omega_D,\text{fair}$. Later we will see that $\omega_D,\text{fair}$ is definable in LTL by a formula of size exponential in $D$, as well as by a DPW of size exponential in $D$.

3. In planning, trajectory 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 2013) (De Giacomo et al. 2016) (Bonet et al. 2017). However, not
We say that an agent strategy $\sigma$ solves the LTL planning under assumptions problem if $\exists \gamma \in \Omega$. Similar definitions apply to DPW planning under assumptions.

We answer by observing that there are translations between planning under assumptions with $\omega = \omega_D, \pi_{\text{fair}}$ and $\gamma = \text{F Goal for Goal} \in \text{Bool}(E \cup A)$. We can check if $\omega_D$ is an environment assumption for the domain $D$, converting it to a DPW $M_D$, and then checking if the DPW $M_D, \omega, \pi_{\text{fair}}$ is environment realizable. Hence we have:

**Theorem 8**. Deciding if an LTL formula $\omega$ is an environment assumption for the domain $D$ is $2\text{EXPTIME}$ complete. Moreover, it can be solved in $\text{EXPTIME}$ in the size of $D$ and $2\text{EXPTIME}$ in the size of $\omega$.

2. Deciding if a DPW $\omega$ is an environment assumption for the domain $D$ is in $\text{EXPTIME}$. Moreover, it can be solved in $\text{EXPTIME}$ in the size of $D$ and $\text{PTIME}$ in the size of $\omega$ and $\text{EXPTIME}$ in the number of colors of $\omega$.

For the lower bound take $D = U$ to be the universal domain and apply the lower bound from Theorem 1.

Now we turn to planning under assumptions.

**Definition 8** (Planning under Assumptions – abstract).

1. A planning under assumptions problem $P$ is a tuple $((D, \omega), \Gamma)$ where
   
   - $D$ is a domain,
   - $\omega \subseteq \text{Str}_\text{env}$ is an environment assumption for $D$, and
   - $\Gamma$ is an agent goal.

2. We say that an agent strategy $\sigma_{\text{ag}}$ solves $P$ if
   
   $$\forall \sigma_{\text{env}} \in \text{Str}_\text{env}(\omega_D) \cap \Omega. \pi_{\sigma_{\text{ag}}}, \sigma_{\text{env}} \in \Gamma$$

We can instantiate this definition to environment assumptions and agent goals definable in LT.

**Definition 9** (Planning under Assumptions – linear-time).

1. An LT planning under assumptions problem is a tuple $P = ((D, \omega), \gamma)$ where $\omega \in LT$ is an environment assumption for $D$ and $\gamma \in \Gamma$ is an agent goal.

2. We say that an agent strategy $\sigma_{\text{ag}}$ realizes $\gamma$ assuming $\omega$, or that it solves $P$, if
   
   $$\forall \sigma_{\text{env}} \triangleright (\omega_D \land \omega). \pi_{\sigma_{\text{ag}}, \sigma_{\text{env}}} \models \gamma$$

The corresponding decision problem asks, given an LT planning under assumptions problem $P$ to decide whether there is an agent strategy that solves $P$. For instance, LT planning under assumptions asks, given $P = ((D, \omega), \gamma)$ with $\omega, \gamma \in LT$, to decide if there is an agent strategy that solves $P$, and to return such a finite-state strategy (if one exists). Similar definitions apply to DPW planning under assumptions, etc.

We observe that typical forms of planning are captured by the above definition:

**Example 4.1**. FOND planning with reachability goals (Rintanen 2004) corresponds to LT planning under assumptions with $\omega = \text{true}$ and $\gamma = \text{F Goal}$ for $\text{Goal} \in \text{Bool}(E \cup A)$.

2. FOND planning with LTL (temporally extended) goals $\gamma$ (Bacchus and Kanban 2000; Pistore and Traverso 2001; Camacho et al. 2017) corresponds to LTL planning under assumptions with $\omega = \text{true}$.

3. FOND planning with LTL trajectory constraints $\omega$ and LTL (temporally extended) goals $\gamma$ (Bonet and Gelfond 2015; De Giacomo et al. 2016; Bonet et al. 2017) corresponds to LTL planning under assumptions.

4. Fair FOND planning with reachability goals (Daniele, Traverso, and Vardi 1999; Gelfner and Bonet 2013; D’Ippolito, Rodriguez, and Sarid 2018) corresponds to planning under assumptions with $\omega = \omega_D, \pi_{\text{fair}}$ and $\gamma = \text{F Goal for Goal} \in \text{Bool}(E \cup A)$.

5. Fair FOND planning with (temporally extended) goals $\gamma$ (Patrizi, Lipovetzky, and Gelfner 2013; Camacho et al. 2017) corresponds to planning under assumptions with $\omega = \omega_D, \pi_{\text{fair}}$.

6. Obviously adding LTL trajectory constraints $\omega_D$ to fair FOND planning with (temporally extended) goals corresponds to planning under assumptions with $\omega = \omega_D, \pi_{\text{fair}}$ and $\omega_{\text{ic}}$.

We also observe that the Fair FOND planning problems just mentioned can be captured by LTL planning under assumptions since $\omega_D, \pi_{\text{fair}}$ can be written in LTL as follows:

$$\exists s \in E \forall a \in A. (\text{GF}(s \land a) \supset \bigwedge_{s':(s, a, s') \in \Delta} \text{GF}(s \land a \land X s'))$$

Unfortunately such a formula mentions explicitly the states $s, s'$ and action $a$ hence may be exponential in $D$ (recall that $D$ is represented compactly). For this reason we later study a direct method for solving Fair FOND planning problems.

### Translating between planning and synthesis

In this section we ask the question if there is a fundamental difference between synthesis and planning in our setting (i.e., assumptions and goals given as linear-time properties).

We answer by observing that there are translations between them. The next two results follow immediately from the definitions:

**Theorem 9** (Synthesis to Planning). Let $(E, A, \omega, \gamma)$ be an LT Synthesis under Assumptions problem and let $P = ((U, \omega), \gamma)$ be the corresponding LT Planning under Assumptions problem where $U$ is the universal domain. Then, for every agent strategy $\sigma_{\text{ag}}$ we have that $\sigma_{\text{ag}}$ solves $P$ iff $\sigma_{\text{ag}}$ realizes $\gamma$ assuming $\omega$.

**Theorem 10** (Planning to Synthesis). Let $D = (E, A, \omega_D, \pi_{\text{fair}}, \Delta)$ be a domain and let $P = ((D, \omega), \gamma)$ be an LT Planning under Assumptions problem. Let $(E, A, \omega_D \land \omega, \gamma)$ be the corresponding LT Synthesis under Assumptions. Then, for every agent strategy $\sigma_{\text{ag}}$ we have that $\sigma_{\text{ag}}$ solves $P$ iff $\sigma_{\text{ag}}$ realizes $\gamma$ assuming $\omega_D \land \omega$.

Thus, we can solve LT planning under assumptions by reducing to LT synthesis under assumptions, which itself can be solved by known results (i.e., Theorem 9).

**Corollary 11**. Solving LTL planning under assumptions is $2\text{EXPTIME}$-complete.
However, this does not distinguish the complexity measured in the size of the domain from that in the size of the assumption and goal formulas. We take this up next.

**Solving Planning under Assumptions**

In this section we show how to solve Planning under Assumptions for concrete specification languages LT, i.e., LT = LTL and LT = DPW. We measure the complexity in two different ways: we fix the domain D and measure the complexity with respect to the size of the formulas/automata for the environment assumption and the agent goal, this is called goal/assumption complexity; and we fix the formulas/automata and measure the complexity with respect to the size of the domain, this is called the domain complexity.\(^4\)

We begin with LT = DPW and consider the following algorithm: Given \( P = ((D,\omega),\gamma) \) in which \( \omega \) is represented by a DPW \( M_\omega \) and \( \gamma \) is represented by a DPW \( M_\gamma \), perform the following steps:

**Alg 1. Solving DPW planning under assumptions**

Given domain \( D \), assumption \( M_\omega \), goal \( M_\gamma \),
1. Form DPW \( M_D \) equivalent to \( \omega_D \).
2. Form DPW \( M \) for \( (M_D \land M_\omega) \supset M_\gamma \).
3. Solve the parity game on \( M \).

The first step can be done in time exponential in the size of \( D \) with a constant number of colors (Lemma\(^7\)). The second step can be done in time polynomial in the sizes of the DPWs \( M_D, M_\omega \) and \( M_\gamma \). For the third step, the think of the DPW \( M \) as a parity game: play starts in the initial state, and at each step, if \( q \) is the current state of \( M \), first the environment picks \( s \in E \) and then the agent picks an action \( a \in A \), i.e., an evaluation of the action variables. The subsequence steps start in the state of \( M \) resulting from taking the unique transition from \( q \) labeled \( s \cup a \). This produces a run of the DPW which the agent is trying to ensure is successful (i.e., the largest color occurring infinitely often is even).

Formally, we say that an agent strategy \( \sigma ag \) is winning if for every environment strategy \( \sigma env \), the unique run of the DPW on input word \( \pi_{\sigma ag, \sigma env} \) is successful. Deciding if the player has a winning strategy, and returning a finite-state strategy (if one exists), is called solving the game. Parity games can be solved in time polynomial in the size of \( M \) and exponential in the number of colors of \( M \) (Grädel, Thomas, and Wilke 2002).

The analysis of the above algorithm shows the following.

**Theorem 12.**

1. The domain complexity of solving DPW planning under assumptions is in \( \text{EXPTIME} \).
2. The goal/assumption complexity of solving DPW planning under assumptions is in \( \text{PTIME} \). Moreover, the complexity with respect to the number of colors of the automata is in \( \text{EXPTIME} \).

Moreover, by converting LTL formulas to DPW with exponentially many colors and double-exponential many states (Vardi 1995; Piterman 2007), we get the upper bounds in the following:

**Theorem 13.** 1. The domain complexity of solving LTL planning under assumptions is \( \text{EXPTIME-complete} \).
2. The goal/assumption complexity of solving LTL planning under assumptions is \( \text{2EXPTIME-complete} \).

For the matching lower-bounds, we have that the domain complexity is \( \text{EXPTIME-hard} \) follows from the fact that planning with reachability goals and no assumptions is \( \text{EXPTIME-hard} \) (Rintanen 2004); to see that the goal/assumption complexity is \( \text{2EXPTIME-hard} \) note that LTL synthesis, known to be \( \text{2EXPTIME-hard} \) (Pnueli and Rosner 1989; Rosner 1992), is a special case (take \( \omega = \text{true} \) and \( D \) to be the universal domain).

Finally we turn to fair LTL planning under assumptions, i.e., planning under assumptions in which the agent goal is specified in LTL, and the environment assumption is of the form \( \omega \land \omega_{D,\text{fair}} \) for some \( \omega \in \text{LTL} \).

**Theorem 14.** 1. The domain complexity of solving Fair LTL planning under assumptions is \( \text{EXPTIME-complete} \).
2. The goal/assumption complexity of solving Fair LTL planning under assumptions is \( \text{2EXPTIME-complete} \).

For the upper bounds, it is enough to first convert the formulas to DPW, and then, using the fact that there is a DPW \( M_{D,\text{fair}} \) of size exponential and with exponentially many colors in the size of \( D \) such that \( Str_{\text{env}}(M_{D,\text{fair}}) = Str_{\text{env}}(\omega_{D,\text{fair}}) \), apply Algorithm 1. Finally, we describe how to build the DPW \( M_{D,\text{fair}} \). We build it up from smaller DPWs using Boolean operations. For \( (s,a,t) \in E \times A \times E \) let \( IO_{s,a} \) be a DPW of constant size equivalent to \( GF(s \land a) \), and let \( IO_{s,a,t} \) be a DPW of constant size equivalent to \( GF(s \land a \land X t) \). Thus \( \bigwedge_{s,a,t} \bigwedge_{(s,a,t) \in E}(\neg IO_{s,a} \lor \bigwedge_{s,a,t} IO_{s,a,t}) \) is a DPW of size polynomial in \( |E| \times |A| \) equivalent to \( \omega_{D,\text{fair}} \). For the lower bounds, that the domain complexity is \( \text{EXPTIME-hard} \) follows from the fact that fair planning with reachability goals and no assumptions is \( \text{EXPTIME-hard} \) (Rintanen 2004; Piterman 2007), we get the lower bounds in the following:

**Theorem 15.** 1. The domain complexity of solving Fair LTL planning under assumptions is \( \text{EXPTIME-complete} \).
2. The goal/assumption complexity of solving Fair LTL planning under assumptions is \( \text{2EXPTIME-complete} \).

**Focusing on finite traces**

In this section we revisit the definitions and results in case that assumptions and goals are expressed as linear-time properties over finite traces. There are two reasons to do this. First, in AI and CS applications executions of interest are often finite (De Giacomo and Vardi 2013). Second, the algorithms presented for the infinite-sequence case involve complex constructions on automata/games that are notoriously hard to optimize (Fogarty et al. 2013). Thus, we will not simply reduce the finite-trace case to the infinite-trace case (De Giacomo, Maselli, and Montali 2014). We begin by carefully defining the setting.
Synthesis and linear-time specifications over finite traces
We define synthesis over finite traces (cf. (De Giacomo and
Vardi 2015)) in a similar way to the infinite-trace case. The
main difference is that agent strategies $\sigma_{ag}$ : $\mathcal{E}^{*} \rightarrow \mathcal{A}$
can be partial. This represents the situation that the agent stops
the play. Environment strategies $\sigma_{env}$ : $\mathcal{A}^{*} \rightarrow \mathcal{E}$ are total
(as before). Thus, the resulting play $\pi_{\sigma_{ag},\sigma_{env}}$ may be finite,
if the agent chooses to stop, as well as infinite.\footnote{Formally, $\pi_{\sigma_{ag},\sigma_{env}}$ is redefined to be the longest trace (it
may be finite or infinite) that complies with both strategies.}

Objectives may be expressed in general specification formalisms LTF
for finite traces, e.g., LTF = LTf (LTL over finite traces)\footnote{All our results for LTf also hold for linear-dynamic logic over
finite traces (LDLf) (De Giacomo and Vardi 2013).}
LTf = DFA (deterministic finite word automata). For $\phi \in \text{LTf}$, we overload notation and write $[[\phi]]$ for the set of finite
traces $\phi$ defines.

We now define realizability in the finite-trace case:

Definition 10. Let $\phi \in \text{LTf}$.
1. We say that $\sigma_{ag}$ realizes $\phi$ (written $\sigma_{ag} \triangleright \phi$) if
   $\forall \sigma_{env}. \left( (\pi_{\sigma_{ag},\sigma_{env}} \text{ is finite and } \pi_{\sigma_{ag},\sigma_{env}} \in [[\phi]]) \right)$.
2. We say that $\sigma_{env}$ realizes $\phi$ (written $\sigma_{env} \triangleright \phi$) if
   $\forall \pi_{\sigma_{ag}}. \left( (\text{if } \pi_{\sigma_{ag},\sigma_{env}} \text{ is finite, then } \pi_{\sigma_{ag},\sigma_{env}} \in [[\phi]]) \right)$.

The asymmetry in the definition results from the fact that
stopping is controlled by the agent.

Duality still holds, and is easier to prove since it amounts
to determinacy of reachability games (Grädel, Thomas, and
Wilke 2002).

Lemma 15 (Duality). For every $\phi \in \text{LTf}$ we have that $\phi$ is not
agent realizable iff $\neg \phi$ is environment realizable.

Linear-temporal logic on finite traces (LTf) The logic
LTf has the same syntax as LTL but is interpreted on finite
traces $\pi \in (2^{\text{Var}})^{+}$. Formally, for $n \leq \text{len}(\pi)$ (the length
of $\pi$) we only reinterpret the temporal operators:
- $(\pi, n) \models \varphi$ if $n < \text{len}(\pi)$ and $(\pi, n+1) \models \varphi$;
- $(\pi, n) \models \varphi_{1} U \varphi_{2}$ if there exists $i$ with $n \leq i \leq \text{len}(\pi)$
such that $(\pi, i) \models \varphi_{2}$ and for all $i \leq j < n, (\pi, j) \models \varphi_{1}$.

Let $\tilde{X}$ denote the dual of $X$, i.e., $\tilde{X} \models \neg X \neg \varphi$. Semantically we have that
- $(\pi, n) \models \tilde{X} \varphi$ if $n < \text{len}(\pi)$ implies $(\pi, n+1) \models \varphi$.

Deterministic finite automata (DFA) A DFA over $\text{Var}$ is a
tuple $M = (Q, q_{init}, T, F)$ which is like a DPW except that
$col$ is replaced by a set $F \subseteq Q$ of final states. The run on
a finite input trace $\pi \in (2^{\text{Var}})^{+}$ is successful if it ends in
a final state. We recall that DFA are closed under Boolean
operations using classic algorithms (e.g., see Vardi 1995).
Also, LTf formulas $\varphi$ (and also LDLf formulas) can be
effectively translated into DFA. This is done in three clas-
sic simple steps that highlight the power of the automata-
theoretic approach: convert $\varphi$ to an alternating automaton
(poly), then into a nondeterministic finite automaton (exp),
and then into a DFA (exp). These steps are outlined in detail
in, e.g., (De Giacomo and Vardi 2013).

Solving Synthesis over finite traces LTf agent synthesis
is the problem, given $\phi \in \text{LTf}$, of deciding if the agent
can realize $\phi$. Now, solving DFA agent synthesis is $\text{PTIME}$-
complete: it amounts to solving a reachability game on the
given DFA $M$, which can be done with an algorithm that
captures how close the agent is to a final state, i.e., a least-
fixpoint of the operation. Finally, to solve LTf agent syn-
thesis first translate the LTf formula to a DFA and then
run the fixpoint algorithm (also, LTf agent synthesis
is $\text{2EXPTIME}$-complete) (De Giacomo and Vardi 2015).

Note that, by Duality, solving LTf environment real-
izability and solving LTf agent realizability are inter-
reducible (and thus the former is also $\text{2EXPTIME}$-complete).
Thus, to decide if $\phi$ is environment realizable we simply
negate the answer to whether $\neg \phi$ is agent realizable.
However, to extract an environment strategy, one solves the dual
safety game.

Synthesis under assumptions We say that $\omega \in \text{LTf}$ is an
environment assumption if $\omega$ is environment realizable.
Solving LTf synthesis under assumptions means to decide if
there is an agent strategy $\sigma_{ag}$ such that

$$\forall \sigma_{env} \triangleright \omega. \left( \pi_{\sigma_{ag},\sigma_{env}} \text{ is finite and } \pi_{\sigma_{ag},\sigma_{env}} \models \gamma \right).$$

We now consider the case that $\text{LTf} = \text{LTf}$. Checking if
$\omega \in \text{LTf}$ is an environment assumption is, by definition, the
problem of deciding if $\omega$ is environment realizable, as
just discussed. Hence we can state the following:

Theorem 16. 1. Deciding if an $\text{LTf}$ formula $\omega$ is an environment assumption
is $\text{2EXPTIME}$-complete.

2. Deciding if a DFA $\omega$ is an environment assumption
is $\text{PTIME}$-complete (cf. (Grädel, Thomas, and Wilke 2002)).

Turning to LTf synthesis under assumptions we have that
synthesis under assumptions and synthesis of the im-
lication are equivalent. Indeed, as before, the key point is
the duality which we have in Lemma 15.

Theorem 17. Suppose $\omega \in \text{LTf}$ is an environment assumption.
The following are equivalent:

1. There is an agent strategy realizing $\omega \supset \gamma$.
2. There is an agent strategy realizing $\gamma$ assuming $\omega$.

Hence to solve synthesis under assumptions we simply
solve agent synthesis for the implication. Hence we have:

Theorem 18. 1. Solving $\text{LTf}$ synthesis under assumptions is $\text{2EXPTIME}$-
complete.

2. Solving DFA synthesis under assumptions is $\text{PTIME}$-
complete.

Planning under assumptions Planning and fair planning
have recently been studied for LTf goals (De Giacomo and
Rubin 2018). Here we define and study how to add environ-
ment assumptions.

Recall that we represent a planning domain $D$ by the
linear-time property $\omega_{D}$ (Definition 5 which itself was
defined as those infinite traces satisfying two conditions. The
exact same conditions determine a set of finite traces, also
denoted $\omega_D$. Moreover, this $\omega_D$ is equivalent to an LTLf
formula of size linear in $D$ and a DFA of size at most ex-
ponential in $D$. To see this, replace $X$ by $X$ in the LTLf
formula from Lemma 6. That is, let $\delta'$ be the LTLf
formula formed from $\delta$ by replacing every term of the form
e' by $\tilde{X}e$. Note that if $n < \text{len}(\pi)$ then $(\pi, n) \models \delta''$ iff
$(\pi_n \cap E, \pi_n \cap A, \pi_n+1 \cap E) \in \Delta$, and if $n = \text{len}(\pi)$ then
$(\pi, n) \models \delta''$ iff $(\pi_n \cap E, \pi_n \cap A) \in \text{Pre}$. The promised
LTLf($E \cup A$) formula is $\text{init} \land [(G \delta' \land \delta' \land \text{pre})]$. Also,
similar to the DPW before there is a DFA of size at most ex-
ponential in the size of $D$ equivalent to $\omega_D$. To see this, take
the DPW $M_D = (Q, q_0, T, \text{col})$ from Lemma 7 and instead of $\text{col}$ define the set of final states to be the set $\text{col}^{-1} = \{0\}$.

As before, say that $\omega \in \text{LTLf}$ is an environment assump-
tion for the domain $D$ if $\omega_D \land \omega$ is environment realizable.
Define an LTLf planning under assumptions problem to be a
tuple $P = ((D, \omega), \gamma)$ with $\omega, \gamma \in \text{LTLf}$ such that $\omega$ is an
environment assumption for $D$. To decide if $\omega \in \text{LTLf}/DFA$ is an
environment assumption for $D$ we use the next algorithm:

Alg 2. Deciding if $\omega$ is an environment assumption for $D$
Given domain $D$, and DFA $M_\omega$.  
1. Convert $D$ into a DFA $M_D$ equivalent to $\omega_D$.  
2. Form the DFA $M$ for $(M_D \land M_\omega)$.  
3. Decide if $M$ is environment realizable.

Further, if $\omega$ is given as an LTLf formula, first convert it to a
DFA $M_\omega$ and then run the algorithm. We then have:

Theorem 19.

1. Deciding if LTLf formula $\omega$ is an environment assumption
   for the domain $D$ is $2\text{EXPTIME}$-complete. Moreover, it
can be solved in $\text{EXPTIME}$ in the size of $D$ and $2\text{EXPTIME}$
in the size of $\omega$.
2. Deciding if DFA $\omega$ is an environment assumption for the
domain $D$ is in $\text{EXPTIME}$. Moreover, it can be solved in
$\text{EXPTIME}$ in the size of $D$ and $\text{PTIME}$ in the size of $\omega$.

Solving Planning under Assumptions As before, there
are simple translations between LTL planning under assum-
ptions and LTLf synthesis under assumptions. And again, solv-
ing LTLf planning under assumptions via such a translation is not fine enough to analyze the complexity in the domain
vs the goal/assumption. To solve DFA/LTLf planning under
assumptions use the following simple algorithm:

Alg 3. Solving DFA planning under assumptions
Given domain $D$, assumption $M_\omega$, goal $M_\gamma$.
1. Convert $D$ into a DFA $M_D$ equivalent to $\omega_D$.  
2. Form the DFA $M$ for $(M_D \land M_\omega)$.  
3. Solve the reachability game on DFA $M$.

Further, if $\omega$ is given as an LTLf formula, first convert it to
a DFA $M_\omega$ and then run the algorithm. This gives the upper
bounds in the following:

Theorem 20.

1. The domain complexity of solving DFA (resp. LTLf) plan-
ing under assumptions is $\text{EXPTIME}$-complete.
2. The goal/assumption complexity of solving DFA (resp.
   LTLf) planning under assumptions is $\text{PTIME}$-complete (resp.
   $2\text{EXPTIME}$-complete).

For the lower bounds, setting $\omega \equiv \text{true}$ results in FOND
with reachability goals, known to be $\text{EXPTIME}$-hard \cite{tanen:2004}; and additionally taking the domain $D$ to be
the universal domain results in DFA (resp. LTLf) synthesis,
known to be $\text{PTIME}$-hard \cite{gradel:2002} (resp. $2\text{EXPTIME}$-hard \cite{degiacomo:2015}).

Finally, if $P = ((D, \omega), \gamma)$ is an LTLf planning under
assumptions problem, say that $\sigma_{ag}$ fairly solves $P$ if for ev-
evy $\sigma_{env} \triangleright \omega_D \land \omega$ we have that if $\sigma_{ag}, \sigma_{env} \in \{\omega_D, \omega\}$
then $\sigma_{ag, \sigma_{env}}$ (is finite and) satisfies $\gamma$ (here $\omega_D, \omega$ from
Example 3 is defined so that it now also includes all finite
traces). We remark that Alg 2 applies unchanged, while Alg
3 applies with step 3 replaced with solving fair reachability
games \cite{degiacomo:2018}.

Conclusion and Outlook

While we illustrate synthesis and planning under assum-
ptions expressed in linear-time specifications, our definitions
immediately apply to assumptions expressed in branching-
time specifications, e.g., CTL, $\mu$-calculus, and tree au-
tomata. As future work, it is of great interest to study synthe-
sis under assumptions in the branching time setting so as to
device restrictions on possible agent behaviors with certain
guarantees, e.g., remain in an area from where the agent can
force the ability to reach the recharging doc, whenever it
needs to, in the spirit of \cite{dal:2002}.

Although our work is in the context of reasoning about
actions and planning, we expect it can also provide in-
sights to verification and to multi-agent systems. In partic-
ular, the undesirable drawback of the agent being able to
falsify an assumption when synthesizing $\text{Assumption } \triangleright \gamma$
well known, and it has been observed that it can be
overcome when the $\text{Assumption}$ is environment realiz-
able \cite{ippolito:2013, brenguier:2017}. Our Theorem 4 provides the principle for such a so-
lution. Interestingly, various degrees of cooperation to fulfill
assumptions among adversarial agents has been considered,
e.g., \cite{chatterjee:2007, bloem:2015, brenguier:2017} and
we believe that a work like present one is needed to establish
similar principled foundations.

Turning to the multi-agent setting, there, agents in a com-
mon environment interact with each other and may have
their own objectives. Thus, it makes sense to model agents
don as hostile to each other, but as rational, i.e., agents
that act to achieve their own objectives. Rational synthe-
sis \cite{kupferman:2014} (as compared to classic synthesis) further requires that the strategy profile
chosen by the agents is in equilibrium (various notions of
equilibrium may be used). It would be interesting to inves-
tigate rational synthesis under environment assumptions, in
the sense that all agents also make use of their own assum-
ptions about their common environment. We believe that con-
sidering assumptions as sets of strategies rather than sets of
traces will serve as a clarifying framework also for the multi-
agent setting.
References

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

[Bloem, Ehlers, and Könighofer 2015] Bloem, R.; Ehlers, R.; and Könighofer, R. 2015. Cooperative reactive synthesis. In Proc. of ATVA 2015. 394–410.

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

[Camacho et al. 2017] 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, 873–879.

[Chatterjee and Henzinger 2007] Chatterjee, K., and Henzinger, T. A. 2007. Assume-guarantee synthesis. In Tools and Algorithms for the Construction and Analysis of Systems. 261–275.

[Cimatti et al. 2003] 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).

[De Giacomo and Rubin 2018] De Giacomo, G., and Rubin, S. 2018. Automata-theoretic foundations of fond planning for LTLf/LDLf goals. In IJCAI.

[De Giacomo and Vardi 2017] De Giacomo, G., and Vardi, M. Y. 2013. Linear temporal logic and linear dynamic logic on finite traces. In IJCAI.

[De Giacomo and Vardi 2015] De Giacomo, G., and Vardi, M. Y. 2015. Synthesis for LTL and LDL on finite traces. In IJCAI.

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

[De Giacomo, Masellis, and Montali 2014] De Giacomo, G.; Masellis, R. D.; and Montali, M. 2014. Reasoning on ltl on finite traces: Insensitivity to infiniteness. In AAAI.

[D’Ippolito et al. 2013] 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, Rodríguez, and Sardiña 2018] 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.

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

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

[Rosner 1992] Rosner, R. 1992. Modular synthesis of reactive systems. Ph.D. Dissertation, PhD thesis, Weizmann.

[Vardi 1995] Vardi, M. Y. 1995. An automata-theoretic approach to linear temporal logic. In Moller, F., and Birtwistle, G. M., eds., Logics for Concurrency - Structure versus Automata, volume 1043 of LNCS, 238–266. Springer.

[Vardi 2011] Vardi, M. Y. 2011. The rise and fall of linear time logic. In GandALF.