LTL_f and LDL_f Synthesis Under Partial Observability

Giuseppe De Giacomo
Sapienza Università di Roma
Roma, Italy
degiacomo@dis.uniroma1.it

Moshe Y. Vardi
Rice University,
Houston, TX, USA
vardi@cs.rice.edu

Abstract

In this paper, we study synthesis under partial observability for logical specifications over finite traces expressed in LTL_f/LDL_f. This form of synthesis can be seen as a generalization of planning under partial observability in nondeterministic domains, which is known to be 2EXPTIME-complete. We start by showing that the usual "belief-state construction" used in planning under partial observability works also for general LTL_f/LDL_f synthesis, though with a jump in computational complexity from 2EXPTIME to 3EXPTIME. Then we show that the belief-state construction can be avoided in favor of a direct automata construction which exploits projection to hide unobservable propositions. This allow us to prove that the problem remains 2EXPTIME-complete. The new synthesis technique proposed is effective and readily implementable.

1 Introduction

LTL_f, *linear temporal logic on finite traces*, has been extensively used in AI [Bacchus and Kabanza, 2000; Gerevini *et al.*, 2009; De Giacomo and Vardi, 2013], as well as in other areas of CS such as in business process modeling [Pesic and van der Aalst, 2006; Sun *et al.*, 2012; De Giacomo *et al.*, 2014]. In [De Giacomo and Vardi, 2013], LTL_f has been extended to LDL_f, *linear dynamic logic over finite traces*, which fully captures monadic second-order logic on finite traces (vs. first-order logic captured by LTL_f). LDL_f includes regular expressions in the temporal operators, which can be used to express, e.g., procedural constraints on executions. Interestingly, reasoning in LTL_f/LDL_f can be done through manipulation of finite state automata (on finite words). In [De Giacomo and Vardi, 2015] LTL_f/LDL_f synthesis under full observability has been studied. This is a general form of adversarial synthesis, related to the classical Church realizability problem [Church, 1963; Vardi, 1996], that has been thoroughly investigated in the infinite setting, starting from [Pnueli and Rosner, 1989]. From an AI perspective LTL_f/LDL_f synthesis is a generalized form of conditional planning under full observability in nondeterministic domains, where the agent controls the choice of actions, while the environment control their nondeterministic effect (notice the devilish, or adversarial, nature of such nondeterminism) [Rintanen, 2004]. The worst case complexity of LTL_f/LDL_f synthesis under full observability is 2EXPTIME-complete [De Giacomo and Vardi, 2015], and it lowers to EXPTIME when we do not mix angelic nondeterminism in the specification (analogous to the ability of guessing the right path towards a final state of an NFA, which makes the specification more succinct) with the devilish nondeterminism due to the adversarial choice. This is exactly what happens in conditional planning under full observability, in which the specification (the planning domain, initial state and reachability goal) does not generate angelic nondeterminism, and the only nondeterminism is the devilish one corresponding to the adversarial choice by the environment of the effects of the action selected by the agent. Indeed conditional planning under full observability is EXPTIME-complete [Rintanen, 2004].

In this paper we study LTL_f/LDL_f synthesis under partial observability, which in turn is a generalized form of conditional planning under partial observability, known to be 2EXPTIME-complete [Rintanen, 2004]. In particular, we consider propositions partitioned into two sets: the first under the control of the agent and the second under the control of the environment. The key point is that only some of the environment variables are observable. The specification consists of an LTL_f/LDL_f formula (typically expressed as a conjunction of a finite set of formulas), which expresses how environment's and agent's propositions should jointly evolve over time. The problem of interest is checking whether there exist strategies for the agent to set the controllable propositions over time, depending only on the history of the *observable* environment propositions, so that regardless of the values assumed by the unobservable propositions, the LTL_f/LDL_f formula is fulfilled. If such strategies exist, it is of interest to compute one. To set the analogy with planning, consider actions (represented as propositions) as the agent's controllable propositions and fluents as the environment's ones. Only some fluents are observable. Then strategies above are generalizations of conditional plans under partial observability.

Inspired by the analogy with planning, we first look into the usual "belief-state construction" [Goldman and Boddy, 1996], which is (sometimes implicitly) at the base of most results in planning under partial observability [Geffner and Bonet, 2013; Bonet and Geffner, 2000; Hoffmann and Brafman, 2005; Bertoli *et al.*, 2006; Bryce *et al.*, 2006; Albore *et al.*, 2009; Maliah *et al.*, 2014]. We show that also

for LTL$_f$/LDL$_f$ synthesis such construction works. Unfortunately, it yields a 3EXPTIME technique, which reduces to 2EXPTIME only when the specification does not induce forms of angelic nondeterminism, as in the case of conditional planning under partial observability.

We then show that we can avoid the belief-state construction altogether in favor of a direct automata construction based on projecting out the unobservable propositions and complementation. This new technique gives us a 2EXPTIME upper-bound for LTL$_f$/LDL$_f$ synthesis under partial observability, which matches the 2EXPTIME-hardness of conditional planning under partial observability [Rintanen, 2004]. Hence we get that, at least from the worst-case complexity point of view, the generalization from conditional planning to full LTL$_f$/LDL$_f$ synthesis under partial observability is for free. The new technique proposed is easily implementable with standard automata operations, and works for a variety of linear time specifications formalisms, ranging from explicit DFA's to arbitrary LTL$_f$/LDL$_f$ formulas.

2 LTL$_f$ and LDL$_f$

LTL on finite traces, or LTL$_f$, has essentially the same syntax as LTL on infinite traces [Pnueli, 1977], namely, given a set $\mathcal{P}$ of propositional symbols, LTL$_f$ formulas φ are as follows:

$$\varphi \quad ::= \quad \phi \mid \neg\varphi \mid \varphi_1 \wedge \varphi_2 \mid \circ\varphi \mid \varphi_1 \,\mathcal{U}\, \varphi_2$$

where ϕ is a propositional formula over $\mathcal{P}$, $\circ$ is the *next* operator and $\mathcal{U}$ is the *until* operator.

We use the usual abbreviations such as $\varphi_1 \vee \varphi_2 \doteq \neg(\neg\varphi_1 \wedge \neg\varphi_2)$; *eventually* as $\Diamond\varphi \doteq \mathit{true}\,\mathcal{U}\,\varphi$; *always* as $\Box\varphi \doteq \neg\Diamond\neg\varphi$; and $\bullet\varphi \doteq \neg\circ\neg\varphi$. (Note that on finite traces $\neg\circ\varphi \not\equiv \circ\neg\varphi$.)

LTL$_f$ is as expressive as FO (first order logic) over finite traces and star-free regular expressions, so strictly less expressive than regular expressions, which in turn are as expressive as MSO (monadic second order logic) over finite traces. On the other hand, regular expressions are not convenient for expressing temporal specifications, since, for example, they miss direct constructs for negation and for conjunction.

For this reason [De Giacomo and Vardi, 2013] introduced[1] LDL$_f$ (*linear dynamic logic on finite traces*), which merges LTL$_f$ with regular expressions through the syntax of the well-known logic of programs PDL, *propositional dynamic logic* [Fischer and Ladner, 1979; Harel *et al.*, 2000], but adopting a semantics based on finite traces. Formally, LDL$_f$ formulas φ are built as follows:

$$\begin{array}{lcl} \varphi & ::= & \phi \mid \neg\varphi \mid \varphi_1 \wedge \varphi_2 \mid \langle\rho\rangle\varphi \\ \rho & ::= & \phi \mid \varphi? \mid \rho_1 + \rho_2 \mid \rho_1;\rho_2 \mid \rho^* \end{array}$$

where ϕ is a propositional formula over $\mathcal{P}$; ρ denotes path expressions, which are regular expressions over propositional formulas ϕ with the addition of the test construct $\varphi?$ typical of PDL. As in PDL, we use the abbreviation $[\rho]\varphi \doteq \neg\langle\rho\rangle\neg\varphi$.

Intuitively, $\langle\rho\rangle\varphi$ states that, from the current step in the trace, there exists an execution satisfying the regular expression ρ such that its last step satisfies φ, while $[\rho]\varphi$ states that, from the current step, all executions satisfying the regular expression ρ are such that their last step satisfies φ. Tests are used to insert into the execution path checks for satisfaction of additional LDL$_f$ formulas.

LDL$_f$ is indeed as expressive as MSO over finite words, and easily captures LTL$_f$ by seeing *next* and *until* as abbreviations: $\circ\varphi \doteq \langle \mathit{true}\rangle\varphi$ and $\varphi_1\,\mathcal{U}\,\varphi_2 \doteq \langle(\varphi_1?;\mathit{true})^*\rangle\varphi_2$.

The semantics of LDL$_f$ is given in terms of *finite traces*, i.e., finite words, denoting a finite, nonempty, sequence π of consecutive steps over the alphabet $2^{\mathcal{P}}$. We use the notation $\mathit{length}(\pi)$ and $\pi(i)$, and in addition we denote by $\pi(i,j)$ the segment of the trace π starting at the i-th step and ending at the j-th step. If $j > \mathit{length}(\pi)$, we get the segment from the i-th step to the end. The satisfaction relation is defined by simultaneous induction on formulas and path expressions as follows: given a finite trace π, an LDL$_f$ formula φ *is true* at a step i, with $1 \le i \le \mathit{length}(\pi)$, in symbols $\pi, i \models \varphi$, if:

- $\pi, i \models \phi$ iff $\pi(i) \models \phi$ (ϕ propositional);
- $\pi, i \models \neg\varphi$ iff $\pi, i \not\models \varphi$;
- $\pi, i \models \varphi_1 \wedge \varphi_2$ iff $\pi, i \models \varphi_1$ and $\pi, i \models \varphi_2$;
- $\pi, i \models \langle\rho\rangle\varphi$ iff there exists $i \le j \le \mathit{length}(\pi)$ such that $\pi(i,j) \in \mathcal{L}(\rho)$ and $\pi, j \models \varphi$;

where the relation $\pi(i,j) \in \mathcal{L}(\rho)$ is as follows:

- $\pi(i,j) \in \mathcal{L}(\phi)$ if $j{=}i{+}1$, $j \le \mathit{length}(\pi)$, and $\pi, i \models \phi$ (ϕ propositional);
- $\pi(i,j) \in \mathcal{L}(\varphi?)$ if $j = i$ and $\pi, i \models \varphi$;
- $\pi(i,j) \in \mathcal{L}(\rho_1 + \rho_2)$ if $\pi(i,j) \in \mathcal{L}(\rho_1)$ or $\pi(i,j) \in \mathcal{L}(\rho_2)$;
- $\pi(i,j) \in \mathcal{L}(\rho_1;\rho_2)$ if there exists k, with $i \le k \le j$, such that $\pi(i,k) \in \mathcal{L}(\rho_1)$ and $\pi(k,j) \in \mathcal{L}(\rho_2)$;
- $\pi(i,j) \in \mathcal{L}(\rho^*)$ if $j = i$ or there exists k, with $i \le k \le j$, such that $\pi(i,k) \in \mathcal{L}(\rho)$ and $\pi(k,j) \in \mathcal{L}(\rho^*)$.

We say that a trace π satisfies an LTL$_f$/LDL$_f$ formula φ, written $\pi \models \varphi$, if $\pi, 1 \models \varphi$. Also, sometimes we denote by $\mathcal{L}(\varphi)$ the set of traces that satisfy φ: $\mathcal{L}(\varphi) = \{\pi \mid \pi \models \varphi\}$.

We can associate with LDL$_f$ formulas φ a (possibly exponentially larger) NFA A_φ that accepts exactly the traces that satisfy φ. To do so (*i*) we translate φ into *alternating automaton on words* (AFW) (whose number of states is polynomial in the size of φ) that accepts exactly the traces that satisfy φ, and then (*ii*) transform the AFW into a NFA [De Giacomo and Vardi, 2013] (whose number of states is exponential in that of the AFW, and hence in the size of φ —this exponential blowup is unavoidable). These two steps can be combined into a simple direct algorithm for computing the NFA corresponding to an LDL$_f$ formula [De Giacomo and Vardi, 2015].

LTL$_f$/LDL$_f$ synthesis under full observability. Synthesis under full observability [Vardi, 1996; Pnueli and Rosner, 1989] has been studied for LTL$_f$/LDL$_f$ in [De Giacomo and Vardi, 2015]. Essentially it is as follows. We partition the set $\mathcal{P}$ of propositions into two disjoint sets $\mathcal{X}$ and $\mathcal{Y}$. We assume to have *no control* on the truth value of the propositions in $\mathcal{X}$, while we can control those in $\mathcal{Y}$. The problem of interest is: *can we control the values of $\mathcal{Y}$ in such a way that for all possible values of $\mathcal{X}$ the LTL$_f$/LDL$_f$ specification φ remains true?* More precisely, traces now assume the form $\pi = (X_0, Y_0)(X_1, Y_1)(X_2, Y_2)\cdots(X_n, Y_n)$, where (X_i, Y_i) is the propositional interpretation at the i-th position in π, now partitioned in the propositional interpretation

[1] An adaptation of LDL interpreted over infinite traces, in turn introduced in [Vardi, 2011].

X_i for $\mathcal{X}$ and Y_i for $\mathcal{Y}$. Let us denote by $\pi_{\mathcal{X}}|_i$ the trace π projected only on $\mathcal{X}$ and truncated at the i-th element (included), i.e., $\pi_{\mathcal{X}}|_i = X_0 X_1 \cdots X_i$. *Synthesis* consists in computing a function $f : (2^{\mathcal{X}})^* \rightarrow 2^{\mathcal{Y}}$ such that for all π with $Y_i = f(\pi_{\mathcal{X}}|_i)$, we have that π satisfies the formula φ. *Realizability* is the corresponding recognition problem, i.e., checking that such a function exists. As usual we will blur the distinction between the two and talk about complexity of synthesis, to intend the complexity of the associated recognition problem, i.e., realizability.

Observe that in synthesis we have no way of constraining the value assumed by the propositions in $\mathcal{X}$: the function we are looking for only acts on propositions in $\mathcal{Y}$.

DFA games. The basic technique introduced in [De Giacomo and Vardi, 2015] to solve synthesis under full observability is to reduce the LTL$_f$/LDL$_f$ specification to a DFA automata and then play over it the so called DFA game. DFA *games* are games between the agent and the environment. A *round* of the game consists of both the agent and the environment setting the values of the propositions they control. A (complete) *play* is a word in $(2^{\mathcal{X}} \times 2^{\mathcal{Y}})^*$ describing how the agent and environment set their propositions at each round till the game stops. The *specification* of the game is given by a DFA $\mathcal{G}$ of the form $\mathcal{G} = (2^{\mathcal{X}} \times 2^{\mathcal{Y}}, S, s_0, \delta, F)$, where:

- $2^{\mathcal{X}} \times 2^{\mathcal{Y}}$ is the alphabet of the game;
- S are the states of the game;
- s_0 is the initial state of the game;
- $\delta : S \times 2^{\mathcal{X}} \times 2^{\mathcal{Y}} \rightarrow S$ is the transition function of the game: given the current state s and a choice of propositions X and Y for the enviroment and the agent, $\delta(s, (X, Y)) = s'$ is the resulting state of the game;
- F are the final states of the game, where the game can be considered terminated.

A play is *winning* for the agent if such a play leads from the initial to a final state. A *strategy* for the agent is a function $f : (2^{\mathcal{X}})^* \rightarrow 2^{\mathcal{Y}}$ that, given a history of choices from the environment, decides which propositions $\mathcal{Y}$ to set to true/false next. A *winning strategy* is a strategy $f : (2^{\mathcal{X}})^* \rightarrow 2^{\mathcal{Y}}$ such that for all π with $Y_i = f(\pi_{\mathcal{X}}|_i)$ we have that π leads to a final state of $\mathcal{G}$. Notice that in the DFA game, in spite of the name, we do have *devilish nondeterminism*, corresponding to uncontrollability of the propositions $\mathcal{X}$. What has been removed is the *angelic nondeterminism* coming from the LTL$_f$/LDL$_f$ specification, which can be translated (in exponential time) into an NFA, which, if not already a DFA, requires a further determination (another exponential step).

To better grasp the game from a planning point of view, consider $\mathcal{X}$ to be *fluents* and $\mathcal{Y}$ to be *actions*. Then the DFA game is akin to an explicit state representation of a conditional planning under full observability problem, with the final states of the DFA game acting as the goal.

To actually compute the strategy, we start by defining the *controllable preimage* $PreC(\mathcal{E})$ of a set $\mathcal{E}$ of states of $\mathcal{G}$ as the set of states s such that there exists a choice of values for propositions $\mathcal{Y}$ such that for all choices of values for propositions $\mathcal{X}$, game $\mathcal{G}$ progresses to states in $\mathcal{E}$. Formally:

$$PreC(\mathcal{E}) = \{s \in S \mid \text{exists } Y \in 2^{\mathcal{Y}} \text{ such that for all } X \in 2^{\mathcal{X}} \text{ we have } \delta(s, (X, Y)) \in \mathcal{E}\}$$

Then, we define the set $Win(\mathcal{G})$ of winning states of the DFA game $\mathcal{G}$, i.e., the set formed by the states from which the agent can win $\mathcal{G}$, as a least-fixpoint, making use of approximates $Win_i(\mathcal{G})$ denoting all states where the controller wins in at most i steps:

- $Win_0(\mathcal{G}) = F$ (the final states of $\mathcal{G}$);
- $Win_{i+1}(\mathcal{G}) = Win_i(\mathcal{G}) \cup PreC(Win_i(\mathcal{G}))$.

Then, $Win(\mathcal{G}) = \bigcup_i Win_i(\mathcal{G})$. Notice that computing $Win(\mathcal{G})$ requires *linear time* in the number of states in $\mathcal{G}$. Indeed, after at most a linear number of steps $Win_{i+1}(\mathcal{G}) = Win_i(\mathcal{G}) = Win(\mathcal{G})$. A DFA game $\mathcal{G}$ admits a winning strategy iff $s_0 \in Win(\mathcal{G})$.

Then, we define a *strategy generator* based on the winning sets $Win_i(\mathcal{G})$. This is a nondeterministic transducer, where nondeterminism is of the kind "don't-care": all nondeterministic choices are equally good. The strategy generator $\mathcal{T}_{\mathcal{G}} = (2^{\mathcal{X}} \times 2^{\mathcal{Y}}, S, s_0, \varrho, \omega)$ is as follows:

- $2^{\mathcal{X}} \times 2^{\mathcal{Y}}$ is the alphabet of the trasducer;
- S are the states of the trasducer;
- s_0 is the initial state;
- $\varrho : S \times 2^{\mathcal{X}} \rightarrow 2^S$ is the transition function such that

$$\varrho(s, X) = \{s' \mid s' = \delta(s, (X, Y)) \text{ and } Y \in \omega(s)\};$$

- $\omega : S \rightarrow 2^{\mathcal{Y}}$ is the output function such that

$$\omega(s) = \{Y \mid \text{ if } s \in Win_{i+1}(\mathcal{G}) - Win_i(\mathcal{G}) \text{ then } \forall X.\delta(s, (X, Y)) \in Win_i(\mathcal{G})\}.$$

The transducer $\mathcal{T}_{\mathcal{G}}$ generates strategies in the following sense: for every way of restricting $\omega(s)$ to return only one of its values (chosen arbitrarily), we get a strategy.

To obtain the DFA game given an LTL$_f$/LDL$_f$ specification we perform some preprocessing, to get from a compact representation in logic to an explicit representation in terms of game states, and to get rid of the angelic nondeterminism. As a result, the complexity of LTL$_f$/LDL$_f$ synthesis is 2EXPTIME-complete in general and EXPTIME-complete in the case of LTL$_f$/LDL$_f$ specifications that do not give rise to angelic nondeterminism, as for conditional planning.

3 Synthesis Under Partial Observability

Now, we introduce synthesis for LTL$_f$/LDL$_f$ specification under partial observability. As before, we partition the set $\mathcal{P}$ of propositions into two disjoint sets $\mathcal{X}$ and $\mathcal{Y}$. As before, we assume to have *no control* on the truth value of the propositions in $\mathcal{X}$, while we can control those in $\mathcal{Y}$. In addition, we assume that only a subset Obs of the uncontrollable propositions $\mathcal{X}$ are actually observable.

The synthesis problem intuitively is: *can we choose suitably the values of* $\mathcal{Y}$ on the basis of what observed *so far in such a way that for all possible values of* $\mathcal{X}$ *the* LTL$_f$/LDL$_f$ *specification* φ *remains true?* More precisely, as before, traces have the form $\pi = (X_0, Y_0)(X_1, Y_1)(X_2, Y_2) \cdots (X_n, Y_n)$, where (X_i, Y_i) is the propositional interpretation at the i-th position in π, partitioned in the propositional interpretation X_i for $\mathcal{X}$

and Y_i for $\mathcal{Y}$. But now, given a propositional interpretation X_i for $\mathcal{X}$, we denote by $obs(X_i)$ the projection of X_i on the propositions in Obs. Given a trace $\pi = (X_0, Y_0)(X_1, Y_1)(X_2, Y_2) \cdots (X_n, Y_n)$, we define the corresponding *observable trace* ω as: $obs(\pi) = (obs(X_1), Y_1)(obs(X_2), Y_2) \cdots (obs(X_n), Y_n)$. Vice versa, given an observable trace ω we define the set $full(\omega)$, denoting all traces giving rise to the same observable trace ω, as: $full(\omega) = \{\pi \mid obs(\pi) = \omega\}$

As before, we denote by $\pi_{\mathcal{X}}|_i$ the trace π projected only on $\mathcal{X}$ and truncated at the i-th element (included), i.e., $\pi_{\mathcal{X}}|_i = X_0 X_1 \cdots X_i$. Similarly, we denote by $\pi_{Obs}|_i$ the observable trace $obs(\pi)$ projected only on Obs and truncated at the i-th element. The *synthesis problem under partial observability* consists in computing a partial function $f : (2^{Obs})^* \to 2^{\mathcal{Y}}$ such that for all π with $Y_i = f(\pi_{Obs}|_i)$ (note that this implies that it is defined), we have that π satisfies formula φ.

To solve the synthesis problem, we can try to proceed as in the case of full observability. We can reduce our LTL$_f$/LDL$_f$ specification to a DFA A over the alphabet $2^{\mathcal{X}} \times 2^{\mathcal{Y}}$.

Over such DFA, however this time, we need to play the DFA *game under partial observability*. A *round* of the game consists of both the agent and the environment setting the values of the propositions they control. A (complete) *play* is a word in $(2^{\mathcal{X}} \times 2^{\mathcal{Y}})^*$ describing how the agent and the environment set their propositions at each round till the game stops.

A play is *winning* for the agent if such a play leads from the initial to a final state. A *partial-observability-strategy* for the agent is a partial function $f_A : (2^{Obs})^* \to 2^{\mathcal{Y}}$ that, given a history of choices from the environment, decides which propositions $\mathcal{Y}$ to set to true/false next based only on the observable history of the environment. A *winning partial-observability-strategy* is a strategy $f_A : (2^{Obs})^* \to 2^{\mathcal{Y}}$ such that for all π with $Y_i = f_A(\pi_{Obs}|_i)$ we have that π leads to a final state of the DFA A. The *synthesis* problem consists of computing a winning strategy.

We show below that one possible way to solve DFA games under partial observability is to use the belief-state construction typically used in conditional planning under partial observability [Goldman and Boddy, 1996; Rintanen, 2004; Bertoli *et al.*, 2006].

4 Belief-States Construction

We define *belief-states* DFA *game* $\mathcal{G}_A^{Obs}$, or simply $\mathcal{G}$, associated with a DFA game $\mathcal{A} = (2^{\mathcal{X}} \times 2^{\mathcal{Y}}, S, s_0, \delta, F)$ as the following DFA game: $\mathcal{G} = (2^{\mathcal{X}} \times 2^{\mathcal{Y}}, \mathcal{B}, B_0, \partial, \mathcal{F})$, where:

- $2^{\mathcal{X}} \times 2^{\mathcal{Y}}$ is the alphabet of the partial observability game;
- $\mathcal{B} = 2^S$ are the states of the partial observability game, which are called *belief states*, and correspond to sets of the states of the original game [Goldman and Boddy, 1996];
- $B_0 = \{s_0\}$ is the initial state of the partial observability game, which corresponds to that of the original game;
- $\partial : \mathcal{B} \times 2^{\mathcal{X}} \times 2^{\mathcal{Y}} \to \mathcal{B}$ is the transition function of the game: given the current state B and a choice of propositions X and Y, respectively for the enviroment and the agent, the transition function is defined as:

$$\partial(B, (X, Y)) = \{s' \mid \text{exists } s, X' \text{ s.t. } s \in B \text{ and } obs(X') = obs(X) \text{ and } \delta(s, (X', Y)) = s'\}$$

- $\mathcal{F} = 2^F$ are the final states of the game, where the game can be considered terminated.

Observe that for X_1 and X_2 such that $obs(X_1) = obs(X_2)$, we have $\partial(B, (X_2, Y)) = \partial(B, (X_1, Y))$, for all B and Y. In other words the partial observability game cannot distinguish between assignments of the uncontrolled propositions that are observationally equivalent.

A crucial observation in the finite trace setting is the following: a strategy reaches a final state if and only if it reaches a final belief state. This observation has been first made in the context of conditional planning under partial observability: a plan reaches the goal iff it reaches a belief state where the goal is satisfied, see Theorem 3.5 in [Bertoli *et al.*, 2006].

Notice that $\mathcal{G}$ is a standard fully observable DFA game, though in the belief state space.

Theorem 1. *Let A be* DFA *game under partial observability, and $\mathcal{G}$ the corresponding belief-state* DFA *game. Then A admits a winning strategy iff $\mathcal{G}$ does.*

Proof (sketch). If-Direction. Form the observation that for X_1 and X_2 such that $obs(X_1) = obs(X_2)$, we have $\partial(B, (X_2, Y)) = \partial(B, (X_1, Y))$, for all B and Y, It is immediate to see that every strategy for $\mathcal{G}$ is also a partial-observability-strategy for A. Indeed, $f_{\mathcal{G}}(\pi_{\mathcal{X}}|_i) = f_{\mathcal{G}}(\pi'_{\mathcal{X}}|_i)$ as long as $\pi_{Obs}|_i = \pi'_{Obs}|_i$. Moreover if $\partial(B_0, \pi) \in \mathcal{F}$ then $\partial(B_0, \hat{\pi}) \in \mathcal{F}$ for every $\hat{\pi} \in full(obs(\pi))$ and hence for $\delta(s_0, \hat{\pi}) \in F$ for every $\hat{\pi} \in full(obs(\pi))$.

Only-if-Direction. Suppose we have a strategy for $f_A : (2^{Obs})^* \to 2^{\mathcal{Y}}$ for A that is winning, i.e., such that for every π with $Y_i = f_A(\pi_{Obs}|_i)$ we have that π leads to a final state of A. Then $f_{\mathcal{G}} : (2^{\mathcal{X}})^* \to 2^{\mathcal{Y}}$ for $\mathcal{G}$ defined as $f_{\mathcal{G}}(\pi_{\mathcal{X}}|_i) = f_A(\pi_{Obs}|_i)$ is a winning strategy for $\mathcal{G}$ since it leads to $\mathcal{F}$. □

Hence we can concentrate on the belief-state DFA Game $\mathcal{G}$, which is a standard DFA game and use it to solve the corresponding DFA game under partial observability.

The construction above requires an exponential blow-up in the original number of states to get the belief-state DFA game, which in turn can be solved polynomially. In fact it can be shown that DFA games with partial information can solve two-player reachability games which are indeed EXPTIME-complete [Reif, 1984].[2] Hence we get:

Theorem 2. *Let A be* DFA *game under partial observability. Then deciding whether A admits a winning strategy is EXPTIME-complete.*

LTL$_f$ /LDL$_f$ synthesis via belief-state construction. To do synthesis in LTL$_f$ or LDL$_f$, we translate an LTL$_f$/LDL$_f$ specification φ into (an AFW and then into) an NFA A_φ. This is an exponential step. Then, we transform the resulting NFA into a DFA A_φ^d, e.g., using the standard determinization algorithm based on the subset construction [Rabin and Scott, 1959]. This costs us another exponential. Then we build the

[2] This paper introduces a construction analogous to the belief-state construction used in planning, see also [Raskin *et al.*, 2007].

belief-state DFA game $\mathcal{G}_{\varphi}^{Obs}$ corresponding to A_{φ}^{d}. This costs us another exponential. At this point we solve the DFA game $\mathcal{G}_{\varphi}^{Obs}$ by computing $Win(\mathcal{G}_{\varphi}^{Obs})$ and the corresponding strategy generator $\mathcal{T}_{\mathcal{G}_{\varphi}^{Obs}}$. This is a linear step.

Considering the cost of each of the steps above, we get the following worst-case computational complexity upper bound.

Theorem 3. *Synthesis under partial observability in* LTL_f/LDL_f *can be solved in 3EXPTIME.*

A 2EXPTIME lower-bound comes from the fact that LTL_f/LDL_f synthesis under full observability is 2EXPTIME-complete [De Giacomo and Vardi, 2015], but also by considering that conditional planning under partial observability, which is indeed 2EXPTIME-complete [Rintanen, 2004], is a special case.

Theorem 4. *Synthesis under partial observability in* LTL_f/LDL_f *is 2EXPTIME-hard.*

It turns out that we can close the above gap between membership and hardness, however we have to give up the belief-state construction. This is what we do next.

5 Projection-based Construction

We now devise a technique to solve synthesis under partial observability which avoids the belief-state construction. In fact, we study such a technique starting from a variety of specifications for the join admissible traces of environment and agent. Namely we consider specifications given as:

- DFA, which is a case of particular interest since conditional planning problems under partial observability can be seen as compactly (logarithmically) represented DFA specifications (notice these allow one to model devilish nondeterminism of nondeterministic planning domains);
- NFA, which, while possibly hard to imagine as an actual specification for a planning/synthesis problem, is of interest from a technical point of view;
- AFW, which generalizes both a DFA and NFA and is tightly related to LTL_f/LDL_f synthesis;
- LTL_f/LDL_f formula, which is the problem of interest in this paper.

The various procedures above, along with complexity results, are summarized in the table in Figure 1 (In the case of DFA we consider also compact representation which essentially corresponds to conditional planning under partial observability).

From DFA specification. We consider first the case in which the specification is given directly as a DFA (over $\mathcal{X}$ and $\mathcal{Y}$, not directly over Obs and $\mathcal{Y}$).

Let the DFA A be $A = (2^{\mathcal{X}} \times 2^{\mathcal{Y}}, S, s_0, \delta, F)$, where:

- $2^{\mathcal{X}} \times 2^{\mathcal{Y}}$ is the alphabet of the DFA;
- S are the states of the DFA;
- s_0 is the initial state of the DFA;
- $\delta : S \times 2^{\mathcal{X}} \times 2^{\mathcal{Y}} \to S$ is the transition function of the DFA: given the current state s and a choice of propositions X and Y, respectively for the enviroment and the agent, $\delta(s, (X, Y)) = s'$ is the resulting state of the DFA;
- F are the final states of the DFA.

Step 1: first complementation. Consider the complement $\overline{A}$ of A. Since A is a DFA we can obtain its complement $\overline{A}$ in linear time by simply switching in A the final states with the non-final ones: $\overline{A} = (2^{\mathcal{X}} \times 2^{\mathcal{Y}}, S, s_0, \delta, \overline{F})$, where $\overline{F} = S - F$. The DFA $\overline{A}$ accepts all joint traces for enviroment and agent that *do not satisfy* the specification A.

Step 2: projection. We project out the propositions that are hidden, getting a new automaton $\Pi(\overline{A}) = (2^{Obs} \times 2^{\mathcal{Y}}, S, s_0, \delta_{\Pi}, \overline{F})$ where:

$$\delta_{\Pi}(s, (O, Y)) = \{s' \mid \delta(s, (X, Y)) = s' \text{ and } obs(X){=}O\}$$

Notice that this automaton $\Pi(\overline{A})$ accepts an observable trace ω if there *exists* a full trace $\pi \in full(\omega)$ that is accepted by $\overline{A}$, i.e., a full trace π that violates the specification A. Notice also that $\Pi(\overline{A})$ is an NFA.

Step 3: second complementation. Consider the complement $\overline{\Pi(\overline{A})}$ of $\Pi(\overline{A})$. This is an exponentially larger DFA, obtained through the usual subset construction [Hopcroft *et al.*, 2001], which accepts a trace ω if *all* full traces in $full(\omega)$ satisfy the specification A.

The DFA $\overline{\Pi(\overline{A})}$ represents the specification for a synthesis problem with full observability whose solutions are strategies (depending only on the observable history) which are solutions for the original synthesis problem under partial observability. To solve it we consider $\overline{\Pi(\overline{A})}$ as a DFA game under full observability and compute its winning strategies (which require a polynomial fixpoint computation). Thus we get an EXPTIME procedure for synthesis under partial observability from DFA specification.

Theorem 5. *Synthesis under partial observability from a* DFA *specification is EXPTIME-complete.*

Proof (sketch). Membership is obtained by using the procedure above. Hardness follows from EXPTIME-hardness of DFA games under partial observability, see Theorem 2. □

Note that the technique above, although it avoids the belief-state construction is analogous to it at Step 3. Indeed, the transition function generated at Step 3 is the result of the subset construction for determinization, which is based on the power set of the original states as the belief-state construction is.

It is of particular interest to consider the case where the DFA specification is represented compactly (i.e., logarithmically), as for example in conditional planning problems under partial observability, where the planning domain and the reachability goal are represented compactly, e.g., as logical formulas [Rintanen, 2004]. Then the technique above becomes 2EXPTIME.

Theorem 6. *Synthesis under partial observability from a* DFA *specification given in a compact (logarithmic) representation is 2EXPTIME-complete.*

Proof (sketch). Membership is obtained by using the procedure above. Hardness comes from 2EXPTIME-completeness of conditional planning under partial observability. □

From NFA specification. If we start from a specification given as an NFA, the procedure above still applies, though in this case the first complementation requires an exponential

DFA spec	NFA spec	AFW spec	LTL$_f$/LDL$_f$ spec
1. Complement DFA spec A, getting DFA $\overline{A}$ *(poly)* 2. Project out unobservable propositions from $\overline{A}$, getting NFA $\Pi(\overline{A})$ *(poly)* 3. Complement NFA $\Pi(\overline{A})$, getting DFA $\overline{\Pi(\overline{A})}$ *(exp)* 4. Solve DFA game *(poly)*	1. Complement NFA spec A, getting DFA $\overline{A}$ *(exp)* 2. Project out unobservable propositions from $\overline{A}$, getting NFA $\Pi(\overline{A})$ *(poly)* 3. Complement NFA $\Pi(\overline{A})$, getting DFA $\overline{\Pi(\overline{A})}$ *(exp)* 4. Solve DFA game *(poly)*	0. Complement AFW spec A_{afw}, getting AFW $\overline{A}_{afw}$ *(poly)* 1. Transform AFW $\overline{A}_{afw}$ into NFA, getting NFA $\overline{A}$ *(exp)* 2. Project out unobservable propositions from $\overline{A}$, getting NFA $\Pi(\overline{A})$ *(poly)* 3. Complement NFA $\Pi(\overline{A})$, getting DFA $\overline{\Pi(\overline{A})}$ *(exp)* 4. Solve DFA game *(poly)*	0. Compute AFW $A_{\neg\Phi}$ for $\neg\Phi$ *(poly)* 1. Transform AFW $A_{\neg\Phi}$ into NFA, getting NFA $\overline{A}$ *(exp)* 2. Project out unobservable propositions from $\overline{A}$, getting NFA $\Pi(\overline{A})$ *(poly)* 3. Complement NFA $\Pi(\overline{A})$, getting DFA $\overline{\Pi(\overline{A})}$ *(exp)* 4. Solve DFA game *(poly)*
EXPTIME-complete *(2EXPTIME-complete starting from compact representations)*	2EXPTIME	2EXPTIME-complete	2EXPTIME-complete

Figure 1: Summary of techniques and results for synthesis under partial observability

blow up, thus making the procedure 2EXPTIME. It remains open whether the problem is indeed 2EXPTIME-hard. Indeed, we conjecture it is.

Theorem 7. *Synthesis under partial observability from an* NFA *specification is in 2EXPTIME.*

From AFW **specification.** If we start from a specification given as AFW then the first complementation can be done in polynomial time, but then we need to transform the resulting AFW into an NFA for projection, and this costs an exponential. So performing the rest of the steps above we get a 2EXPTIME procedure, as for the case of NFA specification.

Theorem 8. *Synthesis under partial observability from an* AFW *specification is 2EXPTIME-complete.*

Proof (sketch). Membership is obtained by using the procedure above. Hardness follows from that for LTL$_f$/LDL$_f$ specifications (Theorem 9), which can indeed be translated polynomially into AFW. □

From LTL$_f$/LDL$_f$ **specification.** Finally, if we start from specifications Φ given as an LTL$_f$/LDL$_f$ formula (or as a finite conjunctions of formulas), then complementation trivially reduces to negating the formula, getting $\neg\Phi$, and the translation of the formula into an NFA can be done in exponential time. Then, as before we proceed by projection, complementation and solving the resulting DFA game. The procedure in this case is 2EXPTIME in the size of the specification formula Φ. This is the same complexity of conditional planning under partial information which is 2EXPTIME-complete already.

Theorem 9. *Synthesis under partial observability from* LTL$_f$/LDL$_f$ *specification is 2EXPTIME-complete.*

Proof (sketch). Membership is obtained by using the procedure above. Hardness follows from 2EXPTIME-hardness of conditional planning under partial observability [Rintanen, 2004], which is a special case. □

We observe that it is quite odd and interesting that for the synthesis problem remains of the same complexity starting from such different forms of specification, cf. Figure 1, although for different reasons in each case.

6 Conclusion

We have studied LTL$_f$/LDL$_f$ synthesis under partial observability, showing that the problem is 2EXPTIME-complete, as for the cases of LTL$_f$/LDL$_f$ synthesis under full observability [De Giacomo and Vardi, 2015] and conditional planning under partial observability [Rintanen, 2004]. Unlike in [Rintanen, 2004], the classical belief-state construction, while applicable, does not give us a procedure that is optimal wrt worst-case computational complexity. This non-optimality is due to the need of, on one hand, removing angelic nondeterminism from the specification, leaving only the adversarial nondeterminism, and on other hand, dealing with the lack of knowledge coming from partial observability. The belief-state construction handles these two aspects separately, requiring an exponential blow up for each of them. This, combined with the fact that logic allows for compact representations, gives us a 3EXPTIME procedure in the case of synthesis from general LTL$_f$/LDL$_f$ specification. The belief-state construction gets back 2EXPTIME only when the LTL$_f$/LDL$_f$ specification does not give rise to angelic nondeterminism, as in the notable case of conditional planning (under partial observability). The new technique based on projection and complementation proposed here is able to combine the elimination of angelic nondeterminism with the elimination of partial observation, thus getting to 2EXPTIME.

Interestingly also in the case of infinite traces LTL synthesis under partial observability is 2EXPTIME-complete [Vardi, 1995; Kupferman and Vardi, 1997]. Nevertheless, the techniques available in this case, are not easily implementable, since they involve determinization of ω-automata that still resists good algorithms [Fogarty *et al.*, 2013]. The construction proposed here, instead, although as expensive in the worst-case, includes only steps for which good algorithms are available, so effective tools can indeed be developed.

One final observation is that in the end we reduce our problem to solving DFA games, which are based on reachability. This is an area where the research in planning excels [Geffner and Bonet, 2013]. It would be interesting to understand to what extent planning technologies can actually be pushed to solve efficiently these forms of synthesis, see [Torres and Baier, 2015] for some recent results.

Acknowledgements. This research was partially supported by the Sapienza project "Immersive Cognitive Environments", by NSF grants CCF-1319459 and IIS-1527668, by NSF Expeditions in Computing project "ExCAPE: Expeditions in Computer Augmented Program Engineering", and by BSF grant 9800096. Part of this work was done while the second author was visiting the Israeli Institute for Advanced Studies.

References

[Albore *et al.*, 2009] Alexandre Albore, Héctor Palacios, and Hector Geffner. A translation-based approach to contingent planning. In *Proc. of IJCAI*, 2009.

[Bacchus and Kabanza, 2000] Fahiem Bacchus and Froduald Kabanza. Using temporal logics to express search control knowledge for planning. *Artificial Intelligence*, 116(1-2), 2000.

[Bertoli *et al.*, 2006] Piergiorgio Bertoli, Alessandro Cimatti, Marco Roveri, and Paolo Traverso. Strong planning under partial observability. *Artificial Intelligence*, 170(45):337 – 384, 2006.

[Bonet and Geffner, 2000] Blai Bonet and Hector Geffner. Planning with incomplete information as heuristic search in belief space. In *Proc. of AIPS*, pages 52–61, 2000.

[Bryce *et al.*, 2006] Daniel Bryce, Subbarao Kambhampati, and David E. Smith. Planning graph heuristics for belief space search. *J. Artif. Intell. Res. (JAIR)*, 26, 2006.

[Church, 1963] Alonzo Church. Logic, arithmetics, and automata. In *Proc. International Congress of Mathematicians, 1962*. institut Mittag-Leffler, 1963.

[De Giacomo and Vardi, 2013] Giuseppe De Giacomo and Moshe Y. Vardi. Linear temporal logic and linear dynamic logic on finite traces. In *Proc. of IJCAI*, 2013.

[De Giacomo and Vardi, 2015] Giuseppe De Giacomo and Moshe Y. Vardi. Synthesis for LTL and LDL on finite traces. In *Proc. of IJCAI*, 2015.

[De Giacomo *et al.*, 2014] Giuseppe De Giacomo, Riccardo De Masellis, Marco Grasso, Fabrizio Maria Maggi, and Marco Montali. Monitoring business metaconstraints based on LTL and LDL for finite traces. In *Proc. of BPM*, 2014.

[Fischer and Ladner, 1979] Michael J. Fischer and Richard E. Ladner. Propositional dynamic logic of regular programs. *Journal of Computer and System Sciences*, 18, 1979.

[Fogarty *et al.*, 2013] Seth Fogarty, Orna Kupferman, Moshe Y. Vardi, and Thomas Wilke. Profile trees for Büchi word automata, with application to determinization. In *Proc. of GandALF*, 2013.

[Geffner and Bonet, 2013] Hector Geffner and Blai Bonet. *A Concise Introduction to Models and Methods for Automated Planning*. Morgan & Claypool Publishers, 2013.

[Gerevini *et al.*, 2009] Alfonso Gerevini, Patrik Haslum, Derek Long, Alessandro Saetti, and Yannis Dimopoulos. Deterministic planning in the fifth international planning competition: PDDL3 and experimental evaluation of the planners. *Artificial Intelligence*, 173(5-6), 2009.

[Goldman and Boddy, 1996] Robert P. Goldman and Mark S. Boddy. Expressive planning and explicit knowledge. In *Proc. of AIPS*, 1996.

[Harel *et al.*, 2000] David Harel, Dexter Kozen, and Jerzy Tiuryn. *Dynamic Logic*. MIT Press, 2000.

[Hoffmann and Brafman, 2005] Jörg Hoffmann and Ronen I. Brafman. Contingent planning via heuristic forward search witn implicit belief states. In *Proc. of ICAPS*, 2005.

[Hopcroft *et al.*, 2001] John E Hopcroft, Rajeev Motwani, and Jeffrey D Ullman. *Introduction to Automata Theory, Languages, and Computation (2nd Edition)*. Addison Wesley, 2 edition, July 2001.

[Kupferman and Vardi, 1997] Orna Kupferman and Moshe Vardi. Synthesis with Incomplete Informatio. *Proc. ICTL*, 1997.

[Maliah *et al.*, 2014] Shlomi Maliah, Ronen I. Brafman, Erez Karpas, and Guy Shani. Partially observable online contingent planning using landmark heuristics. In *Proc. of ICAPS*, 2014.

[Pesic and van der Aalst, 2006] Maja Pesic and Wil M. P. van der Aalst. A declarative approach for flexible business processes management. In *Proc. of the BPM 2006 Workshops*, volume 4103 of *LNCS*. Springer, 2006.

[Pnueli and Rosner, 1989] Amir Pnueli and Roni Rosner. On the synthesis of a reactive module. In *Proc. of POPL*, 1989.

[Pnueli, 1977] Amir Pnueli. The temporal logic of programs. In *Proc. of FOCS*, 1977.

[Rabin and Scott, 1959] Michael O. Rabin and Dana Scott. Finite automata and their decision problems. *IBM J. Res. Dev.*, 3(2), April 1959.

[Raskin *et al.*, 2007] Jean-François Raskin, Krishnendu Chatterjee, Laurent Doyen, and Thomas A. Henzinger. Algorithms for omega-regular games with imperfect information. *Logical Methods in Computer Science*, 3(3), 2007.

[Reif, 1984] John H. Reif. The complexity of two-player games of incomplete information. *Journal of Computer and System Sciences*, 29(2):274 – 301, 1984.

[Rintanen, 2004] Jussi Rintanen. Complexity of planning with partial observability. In *Proc. of ICAPS*, 2004.

[Sun *et al.*, 2012] Yutian Sun, Wei Xu, and Jianwen Su. Declarative choreographies for artifacts. In *Proc. of ICSOC*, 2012.

[Torres and Baier, 2015] Jorge Torres and Jorge A. Baier. Polynomial-time reformulations of LTL temporally extended goals into final-state goals. In *Proc. IJCAI*, 2015.

[Vardi, 1995] Moshe Y. Vardi. An automata-theoretic approach to fair realizability and synthesis. In *Proc. of CAV*, 1995.

[Vardi, 1996] Moshe Y. Vardi. An automata-theoretic approach to linear temporal logic. In *Logics for Concurrency: Structure versus Automata*, volume 1043 of *LNCS*. Springer, 1996.

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