Towards a Unified View of AI Planning and Reactive Synthesis

Alberto Camacho
Department of Computer Science
University of Toronto
Vector Institute
Toronto, Canada
acamacho@cs.toronto.edu

Meghyn Bienvenu
CNRS
University of Bordeaux
Bordeaux, France
meghyn.bienvenu@labri.fr

Sheila A. McIlraith
Department of Computer Science
University of Toronto
Vector Institute
Toronto, Canada
sheila@cs.toronto.edu

Abstract

Automated planning and reactive synthesis are well-established techniques for sequential decision making. In this paper we examine a collection of AI planning problems with temporally extended goals, specified in Linear Temporal Logic (LTL). We characterize these so-called LTL planning problems as two-player games and thereby establish their correspondence to reactive synthesis problems. This unifying view furthers our understanding of the relationship between plan and program synthesis, establishing complexity results for LTL planning tasks. Building on this correspondence, we identify restricted fragments of LTL for which plan synthesis can be realized more efficiently.

1 Introduction

In this work we explore and expose the relationship between AI automated planning and reactive synthesis, a class of program synthesis problems. The two seemingly distinct sequential decision making frameworks share a common past in the seminal work on program and plan synthesis via theorem proving (e.g., (Green 1969; Nilsson 1969; Waldinger and Lee 1969; Fikes and Nilsson 1971)) but diverged as planning research focused on efficient algorithms for classical planning. With growing interest in non-classical planning problems that include non-deterministic actions and temporally extended goals, we reunite these frameworks through the lens of two-player games. This unified view advances our understanding of the relationship between plan and program synthesis, providing us with complexity results for Linear Temporal Logic (LTL) planning and leading to the identification of fragments of LTL for which plan synthesis is more efficient.

Program synthesis from logical specification follows from the classical synthesis problem originally proposed by Church (1957). *Reactive synthesis* is a class of program synthesis problems concerned with synthesizing a reactive module that responds to the environment in accordance with a temporal logic specification. Our concern in this paper is with so-called LTL *synthesis* – reactive synthesis of specifications expressed in LTL. The problem is commonly characterized as a two-player game and has been shown to be 2EXP-complete (Pnueli and Rosner 1989). More recently LTL synthesis has been extended to specifications in LTL interpreted over finite traces (aka $\mathsf{LTL_f}$) (e.g., (De Giacomo and Vardi 2013; 2015; Zhu et al. 2017; Camacho et al. 2018a; 2018b)), and the model has been extended to handle environment assumptions and quality measures (Camacho, Bienvenu, and McIlraith 2018).

The popularity of LTL is not restricted to program synthesis. Over the past 20 years there has been growing interest in planning with temporally extended goals and preferences specified in LTL. Initial work focused on planning with deterministic actions and LTL goals (and preferences) interpreted over finite traces (e.g., (Bacchus and Kabanza 2000; Doherty and Kvarnström 2001; Baier and McIlraith 2006a; 2006b; Triantafillou, Baier, and McIlraith 2015; Torres and Baier 2015)) and subsequently infinite traces (e.g., (Albarghouthi, Baier, and McIlraith 2009; Patrizi et al. 2011)). Other work focused on LTL preferences (e.g., (Edelkamp 2006; Baier, Bacchus, and McIlraith 2009; Coles and Coles 2011; Bienvenu, Fritz, and McIlraith 2011)) some of it in the PDDL3.0 (Planning Domain Definition Language) fragment of LTL (Gerevini et al. 2009). More recent work has examined LTL interpreted over both finite and infinite traces in fully-observable non-deterministic (FOND) planning settings (e.g., (De Giacomo, Patrizi, and Sardiña 2010; Patrizi, Lipovetzky, and Geffner 2013; Camacho et al. 2017; De Giacomo and Rubin 2018; D'Ippolito, Rodríguez, and Sardiña 2018)). We refer to this broad collection of planning problems loosely as LTL *Planning Problems*.

We are seeing increasing commonalities between the recent work on LTL synthesis and the work on LTL planning, particularly in the FOND setting. Both exploit automata representations of LTL for synthesizing solutions. Further the non-determinism in the FOND setting is closely related to the non-determinism exhibited by the environment in synthesis settings. Finally, we note the growing interest in using planning algorithms for the realization of $\mathsf{LTL_f}$ and LTL synthesis (Camacho et al. 2018a; 2018c; 2018d). The correspondence between planning and synthesis has been previously noted (e.g. (D'Ippolito, Rodríguez, and Sardiña 2018; Camacho et al. 2018b)), and even hybrid models have been proposed (e.g. (De Giacomo et al. 2016)). Opportunities abound from the cross-fertilization between these research areas, which motivated us to develop a unified view of LTL planning and reactive synthesis.

In Section 2 we review LTL and reactive LTL synthesis, and in Section 3 we outline a collection of LTL planning problems in a manner that is amenable to their formulation in terms of structured games. Next, we introduce game structures, providing a characterization of LTL synthesis in terms of structured games, re-establishing complexity results for LTL synthesis in this context, and highlighting the equivalence of winning strategies for these game and solutions to the corresponding LTL synthesis problems. In Section 5, we provide a clear bidirectional mapping between structured games and LTL FOND and show that $\mathsf{LTL_f}$ FOND can be reduced to LTL FOND and games. Our reductions allow us to obtain complexity results for LTL/$\mathsf{LTL_f}$ planning that replicate those for games, utilizing notions of combined complexity as well as domain complexity and goal complexity. While the domain complexity of LTL and $\mathsf{LTL_f}$ FOND planning is the same as for FOND with reachability goals, the overall complexity jumps to double-exponential time as result of the goal. In Section 6, we identify restricted fragments of LTL and $\mathsf{LTL_f}$ for which plan synthesis has the same worst-case complexity as FOND with reachability goals but which is richer in terms of its expressiveness. We first explore GR(1), a syntactically restricted fragment of LTL for which synthesis can be done in single exponential time. Then we move to the finite case, and revisit the syntax of PDDL3.0 in the context of $\mathsf{LTL_f}$ FOND planning. In both cases, we show that these compelling and expressive fragments of LTL can be employed without realizing the double exponential complexity of LTL FOND.

The main contributions of this paper are: (i) a unified view of LTL synthesis and various forms of LTL/$\mathsf{LTL_f}$ plan synthesis in terms of *game structures*; (ii) complexity results for various LTL/$\mathsf{LTL_f}$ plan synthesis tasks that decouple the goal and domain complexity, thereby exposing the barriers to efficient algorithms for LTL/$\mathsf{LTL_f}$ planning and synthesis; (iii) a structure-preserving correspondence between LTL synthesis and LTL/$\mathsf{LTL_f}$ planning in FOND settings; (iv) identification of fragments of LTL and $\mathsf{LTL_f}$ for which plan synthesis can be realized more efficiently, including that FOND planning for PDDL3.0 goals is in the same complexity class as FOND planning with final-state goals.

Our theoretical results open the door to combining the benefits of structured planning models and reactive synthesis to specify and perform controller synthesis. Similarly, existing techniques from one field (e.g. LTL synthesis for GR(1) objectives) may be leveraged for the other (e.g. FOND planning with GR(1) goals, for which no dedicated algorithm exists as of yet).

2 Preliminaries

2.1 LTL and Automata

Linear Temporal Logic (LTL) is a propositional modal logic commonly used to express temporally extended properties of state trajectories (Pnueli 1977). The syntax of LTL is defined over a set of propositional variables $p \in AP$, and includes the standard logical connectives and basic temporal operators *next* ($\bigcirc\varphi$) and *until* ($\varphi_1\mathcal{U}\varphi_2$).

$$\varphi := p \mid \top \mid \bot \mid \neg\varphi \mid \varphi_1 \vee \varphi_2 \mid \varphi_1 \wedge \varphi_2 \mid \bigcirc\varphi \mid \varphi_1\mathcal{U}\varphi_2$$

Other temporal operators are defined in terms of these basic operators, including *eventually* ($\Diamond\varphi := \top\mathcal{U}\varphi$), *always* ($\Box\varphi := \neg\Diamond\neg\varphi$), *release* ($\varphi_1\mathcal{R}\varphi_2 := \neg(\neg\varphi_1\mathcal{U}\neg\varphi_2)$), and *weak until* ($\varphi_1\mathcal{W}\varphi_2 := \Box\varphi_1 \vee (\varphi_1\mathcal{U}\varphi_2)$). We denote by $|\varphi|$ the size of φ, i.e., its total number of symbols.

LTL formulae are evaluated over *infinite traces*, i.e., infinite sequences $s_1 s_2 \ldots$ where each $s_i \subseteq AP$ defines a propositional valuation. Intuitively, formula $\bigcirc\varphi$ states that φ holds in the next timestep, while $\varphi_1\mathcal{U}\varphi_2$ states that φ_1 needs to hold until φ_2 holds. Formally, we say that an infinite trace $\pi = s_1 s_2 \cdots$ *satisfies an* LTL *formula* φ if $\pi, 1 \models \varphi$, where $\pi, i \models \varphi$ ($i \geq 1$) is defined recursively as follows:

- $\pi, i \models p$ iff $p \in AP$ and $p \in s_i$,
- $\pi, i \models \neg\varphi$ iff it is not the case that $\pi, i \models \varphi$,
- $\pi, i \models (\varphi_1 \wedge \varphi_2)$ iff $\pi, i \models \varphi_1$ and $\pi, i \models \varphi_2$,
- $\pi, i \models \bigcirc\varphi$ iff $\pi, i+1 \models \varphi$,
- $\pi, i \models \varphi_1\mathcal{U}\varphi_2$ iff there exists a $j \geq i$ such that $\pi, j \models \varphi_2$ and for every $k \in \{i, \ldots, j-1\}$, $\pi, k \models \varphi_1$.

We write $\pi \models \varphi$ when π satisfies φ and call π a *model* of φ.

Every LTL formula can be transformed into an equivalent automaton that accepts all and only the models of the formula. Property 1 formulates this result for *deterministic parity word automata* (DPW) (Piterman 2007). A DPW is a tuple $A = \langle \Sigma, Q, q_0, \delta, \alpha \rangle$, where Σ is a finite alphabet, Q is a finite set of states, $q_0 \in Q$ is the initial state, $\delta : Q \times \Sigma \to Q$ is the transition function, and $\alpha : Q \to \mathbb{N}$ is the labeling function. The number of different values (also called *colours*) that α can take is called the *index* of the automaton. A *run* of A on an infinite word $w = s_0 s_1 \cdots \in \Sigma^\omega$ is a sequence $q_0 q_1 \cdots \in Q^\omega$ where $q_{i+1} = \delta(q_i, s_i)$ for each $i \geq 0$. We say that a run $\rho = q_0 q_1 \cdots$ is *accepting* when the highest value that appears infinitely often in $\alpha(q_0)\alpha(q_1)\cdots$ is even. Finally, the language $\mathcal{L}(A)$ of A is the set of words that have an accepting run on A. LTL formulae can be transformed into DPW over the alphabet $\Sigma := 2^{AP}$ in double-exponential time.

Property 1. Given an LTL formula φ, a DPW A_φ that accepts all and only the models of φ can be constructed in worst-case double-exponential time in $|\varphi|$, with the index of A_φ being worst-case single-exponential in $|\varphi|$.

In this work, we also consider $\mathsf{LTL_f}$, a variant of LTL in which formulae are interpreted over *finite traces* $\pi = s_1 \cdots s_n$. $\mathsf{LTL_f}$ inherits the syntax of LTL. A macro $\mathsf{final} := \neg\bigcirc\top$ is commonly used to indicate the end of the trace. It is important to notice that in $\mathsf{LTL_f}$, $\neg\bigcirc\varphi \not\equiv \bigcirc\neg\varphi$, but instead we have $\neg\bigcirc\varphi \equiv \mathsf{final} \vee \bigcirc\neg\varphi$ (here $\equiv$ denotes equivalence for $\mathsf{LTL_f}$ semantics, i.e., being satisfied by the same finite traces). $\mathsf{LTL_f}$ can be transformed into finite-word automata (cf. (Baier and McIlraith 2006b)).

2.2 Reactive LTL Synthesis

Reactive synthesis is the problem of automatically constructing a reactive module – a system that *reacts* to the environment with the objective of *realizing* a given logical specification (Church 1957). In this paper, we are concerned with LTL synthesis, a form of reactive synthesis where the specification is expressed in LTL (Pnueli and Rosner 1989).

Definition 1. An LTL *specification* is a tuple $\langle \mathcal{X}, \mathcal{Y}, \varphi \rangle$, where $\mathcal{X}$ and $\mathcal{Y}$ are disjoint finite sets of variables, and φ is an LTL formula over $\mathcal{X} \cup \mathcal{Y}$.

A *strategy* for an LTL specification $\langle \mathcal{X}, \mathcal{Y}, \varphi \rangle$ is a function $\sigma : (2^{\mathcal{X}})^+ \to 2^{\mathcal{Y}}$ that maps finite sequences of subsets of $\mathcal{X}$ into subsets of $\mathcal{Y}$. For an infinite sequence $\mathbf{X} = X_1 X_2 \cdots \in (2^{\mathcal{X}})^\omega$, the *play* induced by strategy σ is the infinite sequence $\rho_{\sigma,\mathbf{X}} = (X_1 \cup \sigma(X_1))(X_2 \cup \sigma(X_1 X_2)) \cdots$. A play ρ is *winning* if $\rho \models \varphi$, and a strategy is winning when $\rho_{\sigma,\mathbf{X}} \models \varphi$ for all $\mathbf{X} \in (2^{\mathcal{X}})^\omega$. *Realizability* is the problem of deciding whether an LTL specification has a winning strategy, and *synthesis* is the problem of computing one. Both problems can be solved in double-exponential time (Pnueli and Rosner 1989):

Theorem 1. LTL *realizability is 2EXP-complete.*

3 Plan Synthesis with LTL Goals

Automated planning provides a framework for sequential decision-making in which the agent can interact with the environment according to a prescribed domain theory, and where the objective is to generate a plan whose execution is guaranteed to satisfy a prescribed goal. Planning problems differ in the assumptions they make about the form of the initial state (complete, partial), the transition system (deterministic, non-deterministic, stochastic), the nature of the goal condition, whether the state is (partially) observable during plan execution, the form of the solution or *plan* (sequence of actions, conditional plan, policy), and whether the execution of that plan is finite or infinite.

Our concern in this paper is with planning problems whose transition systems are non-deterministic and whose goals are temporally extended and as such take the form of constraints on valid executions of the plan. We specify such temporally extended goals in LTL. Plan executions may be finite or infinite. Unless otherwise noted, initial states are complete and plan execution is fully observable. In the rest of this section we review this class of so-called LTL *(resp.* LTL$_\mathsf{f}$*) Planning Problems* using notation that will facilitate subsequent characterization in terms of 2-player games.

Planning Domain Following (Ghallab, Nau, and Traverso 2016), we represent planning domains compactly using a symbolic representation of states, actions, and transitions:

Definition 2. A *planning domain* is a tuple $\langle \mathcal{F}, \mathcal{A}, \delta, \mathrm{Poss} \rangle$, where $\mathcal{F}$ is a finite set of fluent symbols, $\mathcal{A}$ is a finite set of action symbols, $\mathrm{Poss} \subseteq 2^{\mathcal{F}} \times \mathcal{A}$, and $\delta : 2^{\mathcal{F}} \times \mathcal{A} \to 2^{2^{\mathcal{F}}}$. The *size of the domain* is $|\mathcal{F}|$.

The fluents in $\mathcal{F}$ denote state properties; we employ a propositional STRIPS representation to parsimoniously represent states as subsets $s \subseteq \mathcal{F}$; the set of states is $S := 2^{\mathcal{F}}$. We use the term *search space* to refer to the set S. An action a is applicable in state s if $(s, a) \in \mathrm{Poss}$. The function δ describes state transitions, with $s' \in \delta(s, a)$ indicating that state s' is a possible result of applying action a in state s.

A *strategy* for a planning domain $\mathcal{D} = \langle \mathcal{F}, \mathcal{A}, \delta, \mathrm{Poss} \rangle$ is a function $\sigma : (2^{\mathcal{F}})^+ \to \mathcal{A} \cup \{\mathsf{end}\}$ that maps finite state histories to actions, or to end otherwise.[1] Here end is a special symbol, not contained in $\mathcal{D}$, that denotes the termination of a plan. We require strategies to be well defined, that is, for all $n \geq 0$, action $\sigma(s_0 \cdots s_n)$ is applicable in s_n. A strategy is *history-independent* if $\sigma(s_0 \cdots s_n) = \sigma(s_n)$ for each $s_0 \cdots s_n \in (2^{\mathcal{F}})^+$. In this case, we call σ a *policy*. An *execution* of σ from $s_0 \in S$ is a sequence of state-action pairs $\pi = (s_0, a_0)(s_1, a_1) \cdots$ such that, for each $n < |\pi|$: (i) $a_n = \sigma(s_0 \cdots s_n)$; (ii) if $n < |\pi| - 1$, then $a_n \in \mathcal{A}$ and $s_{n+1} \in \delta(s_n, a_n)$; and (iii) if $n = |\pi| - 1$, then $a_n = \mathsf{end}$. The execution *terminates* when $a_n = \mathsf{end}$. Terminating executions are finite sequences $\pi = (s_0, a_0) \cdots (s_n, a_n)$ where n is the first and only index i such that $a_i = \mathsf{end}$. Non-terminating executions are infinite, and we write $|\pi| = \infty$.

LTL **FOND Planning** An LTL *fully-observable non-deterministic (FOND) planning problem* is a tuple $\langle \mathcal{D}, \varphi_I, \varphi_G \rangle$, where $\mathcal{D} = \langle \mathcal{F}, \mathcal{A}, \delta, \mathrm{Poss} \rangle$ is a planning domain, φ_I is a propositional formula over $\mathcal{F}$ that describes a unique state called the *initial state* (typically given as a complete conjunction of literals), and φ_G is an LTL formula over $\mathcal{F} \cup \mathcal{A}$. A strategy σ is a *solution* to an LTL FOND planning problem $\langle \mathcal{D}, \varphi_I, \varphi_G \rangle$ if all executions of σ that commence in $s_0 \models \varphi_I$ are infinite and satisfy φ_G. In other words, the objective is to define a non-terminating process that guarantees achievement of the given temporally extended goal, whose satisfaction may depend on the entire execution, rather than just the final state.

LTL$_\mathsf{f}$ **FOND Planning** We can define LTL$_\mathsf{f}$ *FOND planning problems* in exactly the same manner, except that a strategy σ is deemed a *solution* if all executions of σ that commence in $s_0 \models \varphi_I$ *terminate* and satisfy φ_G (note that we use LTL$_\mathsf{f}$ semantics to evaluate φ_G over the induced finite trace). Thus, the aim is to generate terminating strategies that guarantee satisfaction of the goal.

FOND Planning As a special case of LTL$_\mathsf{f}$ FOND planning, we have (plain) FOND planning, in which the goal takes the form $\varphi_G = \Diamond(\theta_g \wedge \mathsf{final})$, with θ_g a propositional formula over fluents $\mathcal{F}$ (typically, a conjunction of fluents). Solutions to FOND planning are policies (rather than arbitrary strategies), but we observe that this restriction does not affect the existence of solutions (cf. Proposition 1).

Proposition 1. *Any strategy that solves a* LTL$_\mathsf{f}$ *FOND planning problem $\mathcal{P}$ with goal $\varphi_G = \Diamond(\theta_g \wedge \mathsf{final})$ can be transformed into a policy that is a solution to $\mathcal{P}$.*

Planning with Partially Specified Initial States A benefit of describing the initial state by a formula φ_I is that it allows us to *partially specify* the initial state. We extend the definitions of LTL / LTL$_\mathsf{f}$ FOND to allow for φ_I to describe a family of states, i.e., those that satisfy φ_I. Solutions to an LTL / LTL$_\mathsf{f}$ FOND problem $\langle \mathcal{D}, \varphi_I, \varphi_G \rangle$ with partially specified initial state φ_I are defined in the natural way. A strategy σ is a solution to $\langle \mathcal{D}, \varphi_I, \varphi_G \rangle$ if σ is a solution to an LTL / LTL$_\mathsf{f}$ FOND problem $\langle \mathcal{D}, \varphi'_I, \varphi_G \rangle$ for any *complete* formula φ'_I that describes a unique initial state, and such that

[1] History-dependent strategies are often represented compactly in form of *finite-state controllers* (cf. (Patrizi, Lipovetzky, and Geffner 2013; Camacho et al. 2017)).

$\varphi'_I \models \varphi_I$. Informally, σ solves all problems with initial state that satisfies φ_I.

Deterministic LTL / LTL$_f$ Planning The preceding problems can be restricted by only considering actions with deterministic effects, i.e., requiring that $|\delta(s, a)| = 1$ for each pair $(s, a) \in S \times \mathcal{A}$. We refer to this class of problems as *deterministic* LTL / LTL$_f$ *planning problems*.

Complexity of Planning When speaking about the complexity of planning, we focus on the decision problem of *plan existence*. There are different possible complexity measures depending on which parts of the problem are considered as fixed or varying. In addition to the standard *combined complexity* measure (in terms of the size of the whole problem), we will consider *domain complexity* (in terms of the size of the domain, with the goal formula treated as fixed) and *goal complexity* (in terms of the size of the goal formula, with the domain treated as fixed).

4 Game Structures

In the previous section, we presented a variety of LTL (resp. LTL$_f$) planning problems. In this section we review and elaborate on the notion of *game structures* which we employ to expose and highlight correspondences between these planning problems and LTL synthesis.

Piterman, Pnueli, and Sa'ar (2006) studied the problem of computing winning strategies for so-called *game structures* (see Definition 3). Intuitively, a game structure is similar to an LTL specification. The difference is that the moves of the players (system and environment) in a game structure are assumed to be constrained by the *rules* of the game. Solutions are system strategies that guarantee achievement of a *winning condition* φ, under the assumption that both players act as mandated by the rules described by θ_e, θ_s, ψ_e, and ψ_s.

Definition 3. A *game structure* is a tuple $G = \langle \mathcal{X}, \mathcal{Y}, \theta_e, \theta_s, \psi_e, \psi_s, \varphi \rangle$, where:

- $\mathcal{X}$ is a finite set of *environment* variables.
- $\mathcal{Y}$ is a finite set of *system* variables, disjoint with $\mathcal{X}$.
- θ_e is a propositional formula over $\mathcal{X}$.
- θ_s is a propositional formula over $\mathcal{X} \cup \mathcal{Y}$.
- ψ_e is a propositional formula over $\mathcal{X} \cup \mathcal{Y} \cup \bigcirc\mathcal{X}$.
- ψ_s is a propositional formula over $\mathcal{X} \cup \mathcal{Y} \cup \bigcirc\mathcal{X} \cup \bigcirc\mathcal{Y}$.
- φ is an LTL formula over $\mathcal{X} \cup \mathcal{Y}$ (the *winning condition*)

with $\bigcirc\mathcal{X} = \{\bigcirc x \mid x \in \mathcal{X}\}$ and $\bigcirc\mathcal{Y} = \{\bigcirc y \mid y \in \mathcal{Y}\}$.

The *environment* player moves first in each turn, and *inputs* a subset of the $\mathcal{X}$ variables. In response, the *system* player *outputs* a subset of $\mathcal{Y}$ variables. The moves $X_n \subseteq \mathcal{X}$ and $Y_n \subseteq \mathcal{Y}$ in the n-th turn constitute a game configuration, or *state* $s_n = X_n \cup Y_n$. We shall use the term *search space* to refer to the set of all such states and observe that its size is singly exponential in $|\mathcal{X}| + |\mathcal{Y}|$. The structure of the game constrains the players' moves. The LTL formula ψ_e relates the values of X_n, Y_n, and X_{n+1} and states how the environment is allowed to respond to a state s_n. Similarly, the LTL formula ψ_s relates the values of X_n, Y_n, X_{n+1}, and Y_{n+1}, and states how the system can react to the environment given the previous state. Formulas θ_e and θ_s have similar purposes, serving to constrain the players' initial moves.

A *play* of the game G is a maximal (possibly infinite) sequence of states, $\rho = s_0 s_1 \cdots$, such that: (i) s_0 is an *initial state*, i.e., $s_0 \models \theta_e \wedge \theta_s$; and (ii) for each $n > 0$, s_n is a *successor* of s_{n-1}, i.e. $(s_{n-1}, s_n) \models \psi_e \wedge \psi_s$. The system player aims to generate a play that satisfies the winning condition φ. Formally, a play ρ is *winning* for the system if either (i) $\rho = s_0 \cdots s_n$ is finite and there is no s_{n+1} such that $(s_n, s_{n+1}) \models \psi_e$ (i.e., the environment has no legal move); or (ii) ρ is infinite and $\rho \models \varphi$.

Game *strategies* are defined as for LTL synthesis. Namely, a strategy for the system player is a function $\sigma : (2^{\mathcal{X}})^+ \to 2^{\mathcal{Y}}$. A play $\rho = (X_0 \cup Y_0)(X_1 \cup Y_1) \cdots$ is *σ-compliant* if $Y_n = \sigma(X_0 \cdots X_n)$ at each $n < |\rho|$. We say σ is a *winning strategy* for the system player if all σ-compliant plays are winning. *Winner determination* is to decide whether such a winning strategy exists, and by *solving the game*, we mean the task of producing such a strategy.

Game Structures and LTL Synthesis The LTL synthesis and structured game problems are interreducible: structured games can be reduced to LTL synthesis where all the structure is embedded into the LTL objective (Theorem 2); conversely, any LTL specification formula can be converted into an equivalent game with no structural constraints (Proposition 2). We should emphasize that Theorem 2 was originally formulated for syntactically restricted formulae (in the GR(1) fragment, discussed in Section 6.1), but the result naturally extends to arbitrary LTL formulae. The *Temporal Logic Synthesis Format* (TLSF) is a high-level description format for LTL synthesis (Jacobs, Klein, and Schirmer 2016). TLSF specifications are described in terms of the components of a structured game. TLSF can be interpreted under two different semantics, so-called *strict* or alternatively *standard* semantics. The reduction in Theorem 2 corresponds to the *strict* semantics.

Theorem 2 (adapted[2] and extended from (Klein and Pnueli 2010; Bloem et al. 2012))**.** *The system wins the game $G = \langle \mathcal{X}, \mathcal{Y}, \theta_e, \theta_s, \psi_e, \psi_s, \varphi \rangle$ iff $\langle \mathcal{X}, \mathcal{Y}, \varphi_G \rangle$ is realizable, where:*

$$\varphi_G := (\theta_e \to \theta_s) \wedge (\theta_e \to (\psi_s \mathcal{W} \neg \psi_e)) \wedge ((\Box \psi_e \wedge \theta_e) \to \varphi))$$

Moreover, the transformation preserves winning strategies.

We now show how LTL synthesis corresponds to a special case of structured games:

Proposition 2. *A strategy is winning for the* LTL *synthesis problem $\langle \mathcal{X}, \mathcal{Y}, \varphi \rangle$ iff it is a winning strategy for the corresponding structured game $\langle \mathcal{X}, \mathcal{Y}, \top, \top, \top, \top, \varphi \rangle$*

Solving Structured Games As the reduction from Theorem 2 preserves winning strategies, it provides a way to solve games via reduction to LTL synthesis and to derive the following complexity results:

Theorem 3. *Structured games can be solved in double-exponential time, and the winner determination problem for structured games is 2EXP-complete.*

[2]The formula in (Bloem et al. 2012) makes uses future and past temporal operators. Here, we use an equivalent formula with only future temporal operators (cf. (Jacobs, Klein, and Schirmer 2016)).

In what follows, we present an alternative way to solve games that exploits their structure and allows us to obtain more fine-grained complexity results in terms of the size of the search space and winning condition (Theorem 4). These results extend similar results to games in a restricted fragment of LTL, which we will revisit in Section 6.1.

The classical approach to LTL realizability reduces the specification to a parity game, and constructs a strategy via fixpoint computation (e.g. (Pnueli and Rosner 1989)). This approach can be straightforwardly adapted to solve structured games $G = \langle \mathcal{X}, \mathcal{Y}, \theta_e, \theta_s, \psi_e, \psi_s, \varphi \rangle$. We revisit here the main steps. The interesting thing to notice is that the procedure decouples the role of the winning condition and the search space within the game complexity.

1. Transform φ into a DPW A_φ
2. Construct a parity game $G' = A_\varphi || G$
3. Compute a winning strategy for G'

The first step computes a DPW A_φ, in worst-case double exponential time in $|\varphi|$, that recognizes the models of φ. The index d of A_φ (i.e., number of colors) is, at most, exponential in $|\varphi|$. In the second step, a product *parity* game $G' = A_\varphi || G$ is constructed that integrates the dynamics of A_φ and G in parallel. In the context of this paper, a parity game may be viewed as a structured game with a parity winning condition, in place of an LTL formula, to evaluate which infinite plays are winning. Parity games are played on a finite graph whose nodes represent game states. The dynamics of the game alternates between system' and environment' nodes. The nodes associated with the turn of the environment player are labeled with tuples (q, X, Y), where q is a DPW state, $X \subseteq \mathcal{X}$, and $Y \subseteq \mathcal{Y}$. Intuitively, q keeps track of the run of A_φ on a partial play $\rho = s_0 s_1 \cdots s_n$, and X and Y are the components of state s_n. Similarly, the nodes associated with the turn of the system player are labeled with tuples (q, X, Y, X') that keep track of the run of A_φ on *ρ followed by the environment's move X'*. The colour of a node is the colour of the DBW state in the node's label, and the index of G' is the highest colour among all its nodes. The edges of the tree connect nodes in the natural way whenever both players have been following the rules of the structured game. Edges $(q, X, Y) \xrightarrow{X'} (q, X, Y, X')$ reflect an environment's move, X', and edges $(q, X, Y, X') \xrightarrow{Y'} (q', X', Y')$ reflect a system's move, Y', with $q' = \delta(q, X' \cup Y')$. Additional edges are included to handle illegal moves of the structured game G. Environment moves that violate ψ_e are captured by edges that point to a dummy node, $q_\top$. More precisely, $(q, X, Y) \xrightarrow{X'} q_\top$ whenever $(X \cup Y)(X') \not\models \psi_e$. Similarly, edges to a dummy node $q_\bot$ handle system moves that violate ψ_s, i.e., $(q, X, Y, X') \xrightarrow{Y'} q_\bot$ whenever $(X \cup Y)(X' \cup Y') \not\models \psi_s$. We shall assign colour 0 to $q_\top$, and 1 to $q_\bot$. Both $q_\top$ and $q_\bot$ are sink nodes, meaning that outgoing edges are all self-loops. The root node of the game is labeled with $(q_0, \_, \_)$, with q_0 being the initial state of the DBW. The root node has no play history associated to it. Its outgoing edges correspond to environment moves X and transition to nodes with labels of the form $(q_0, \_, \_, X)$ whenever $X \models \theta_e$, and to $q_\top$ otherwise. Similarly, nodes with labels of the form $(q_0, \_, \_, X)$ transition to nodes with labels of the form (q', X, Y) whenever $X \cup Y \models \theta_s$, and to $q_\bot$ otherwise. This construction preserves winner determination, and a winning strategy for G can be extracted from a winning strategy for G'.

In the third step, a winning strategy for G' can be computed in $O(e \cdot n^d)$, where e is the number of edges, n is the number of nodes, and d is the index of G' (Zielonka 1998). In the parity game G' described above, the number of nodes – as well as the number of edges – is polynomial on $|A_\varphi| \cdot 2^{O(|\mathcal{X}|+|\mathcal{Y}|)}$, and G' has the same index of A_φ, which is exponential in $|\mathcal{X}| + |\mathcal{Y}|$ (w.l.o.g., we assume A_φ has index greater than zero). The following result follows:

Theorem 4. *Structured games can be solved in double-exponential time in the size of the winning condition, and in polynomial time in the size of the search space.*

5 Plan Synthesis and Structured Games

The correspondence between planning and game structures has been noted in the past. De Giacomo et al. (2010) constructed a reduction of FOND planning with final-state goals into game structures. Here, we extend those results to LTL FOND and LTL$_f$ FOND planning.

We provide a clear bidirectional mapping between structured games and LTL FOND planning (Sections 5.1 and 5.2), and then show that LTL$_f$ FOND planning can be reduced to LTL FOND and games (Section 5.3). Our reductions allow us to obtain complexity results for LTL/LTL$_f$ planning that replicate those for games.

5.1 Structured Games as LTL FOND Planning

We begin by presenting a reduction of structured games to LTL FOND planning. Intuitively, we model the moves of the system by means of planning actions, and the response of the environment by means of non-deterministic action effects.

Let $G = \langle \mathcal{X}, \mathcal{Y}, \theta_e, \theta_s, \psi_e, \psi_s, \varphi \rangle$ be a game structure. We assume for now that θ_e is satisfiable and defines a complete truth assignment to the variables $\mathcal{X}$, returning later to the general case. We construct an LTL FOND planning problem $\mathcal{P}_G = \langle \mathcal{D}, \varphi_I, \varphi_G \rangle$ with domain $\mathcal{D} = \langle \mathcal{F}, \mathcal{A}, \delta, \mathrm{Poss} \rangle$ and

$$\mathcal{F} := \mathcal{X} \cup \mathcal{Y} \cup \mathcal{X}' \cup \{x_{init}, x_{end}, x'_{end}, a_{end}\}$$
$$\mathcal{A} := \{A' \mid A \subseteq \mathcal{Y}\} \cup \{a'_{end}\}$$
$$\varphi_I := x_{init} \wedge \neg x_{end} \wedge \neg x'_{end} \wedge \neg a_{end} \wedge \theta'_e \qquad \varphi_G := \Diamond x_{end} \vee \bigcirc\varphi$$

Note that we use $\mathcal{V}' := \{v' \mid v \in \mathcal{V}\}$ to denote a set of primed variables (and similarly, use θ'_e for θ_e with all variables now primed). Planning states keep track of the last game state. Intuitively, primed variables serve to indicate the moves performed by each player in the current turn, whereas unprimed variables indicate the last game state.

The set $\mathrm{Poss} \subseteq 2^{\mathcal{F}} \times \mathcal{A}$ consists of all pairs $(X_1 \cup A_1 \cup X'_2, A'_2)$ such that $X_1 \subseteq \mathcal{X} \cup \{x_{init}, x_{end}\}$, $X_2 \subseteq \mathcal{X} \cup \{x_{end}\}$, $A_1, A_2 \subseteq \mathcal{Y} \cup \{a_{end}\}$, and

- $X_1 \models x_{init} \wedge \neg x_{end}$ and $(X_2 \cup A_2) \models \theta_s$, or
- $X_1, X_2 \subseteq \mathcal{X}$, $A_1, A_2 \subseteq \mathcal{Y}$, and $(X_1 \cup A_1)(X_2 \cup A_2) \models \psi_s$, or
- $X_2 \models x_{end}$ and $A_2 = \{a_{end}\}$

The initial state φ_I simulates the initial move of the environment, determined by the (unique, by assumption) interpretation of $\mathcal{X}$ variables that make θ_e true. We now define how δ simulates the moves of the environment in response to the moves of the system. Intuitively, the non-determinism in δ allows the environment to make a legal move, or otherwise transition to $\{x_{end}\}$. Formally, if $X_1, X_2 \subseteq \mathcal{X}$, and $A_1, A_2 \subseteq \mathcal{Y}$, then δ is defined by:

$$\delta(X_1 \cup A_1 \cup X_2', A_2') := \{X_2 \cup A_2 \cup X_3' \mid X_3 = \{x_{end}\}\} \cup \{X_2 \cup A_2 \cup X_3' \mid X_3 \subseteq \mathcal{X} \text{ and } (X_2 \cup A_2)(X_3) \models \psi_e\}$$

We simulate that the environment makes an illegal move by assigning $X_3 = \{x_{end}\}$, and force the dynamics to enter a loop in which only action a_{end} can be applied: $\delta(X_1 \cup A_1 \cup \{x'_{end}\}, \{a_{end}\}) := \{\{x_{end}, a_{end}, x'_{end}\}\}$. Those simulated executions satisfy $\Diamond x_{end}$. Note that executions that satisfy $\bigcirc\varphi$ are in correspondence with plays that satisfy the winning condition, φ. The one-timestep offset in $\bigcirc\varphi$ is needed because φ is evaluated over unprimed variables, and those are delayed by one timestep in the construction of the problem.

In the general case, with generic θ_e, our reduction can be extended with a dummy action, only applicable in a new dummy initial state, that transitions non-deterministically into one of the states that satisfy $x_{init} \wedge \neg x_{end} \wedge \neg x'_{end} \wedge \neg a_{end} \wedge \theta'_e$, or x_{end}. The new goal becomes $\varphi_G := \Diamond x_{end} \vee \bigcirc\bigcirc\varphi$, with the additional *next* operator being introduced to reflect the extra initial timestep. The reduction is polynomial, as it transforms a game G into an LTL FOND problem $\mathcal{P}_G$ with a polynomial increase in the size of the search space. Winning strategies to G can be extracted directly from solutions to the reduced LTL FOND problem.

Theorem 5. *A structured game with winning condition φ can be reduced to an* LTL *FOND planning problem with goal $\varphi_G := \Diamond x_{end} \vee \bigcirc\bigcirc\varphi$, with a polynomial increase in the size of the search space.*

5.2 LTL FOND Planning as Structured Games

To complete the bidirectional mapping, we provide a reduction from LTL FOND planning into structured games. For an LTL FOND planning problem $\mathcal{P} = \langle \mathcal{D}, \varphi_I, \varphi_G \rangle$ with domain $\mathcal{D} = \langle \mathcal{F}, \mathcal{A}, \delta, \text{Poss} \rangle$ we construct a game $G_{\mathcal{P}} = \langle \mathcal{X}, \mathcal{Y}, \theta_e, \theta_s, \psi_e, \psi_s, \varphi_G \rangle$, where $\mathcal{X} := \mathcal{F}$, $\mathcal{Y} := \mathcal{A}$, and:

$$\theta_e := \varphi_I \qquad \psi_e := \bigvee_{s \in 2^{\mathcal{F}}} \bigvee_{a \in \mathcal{A}} \bigvee_{s' \in \delta(s,a)} (\tau(s) \wedge a \wedge \bigcirc(\tau(s')))$$

$$\theta_s := \left(\varphi^{one}_{\mathcal{A}} \wedge \bigvee_{(s,a) \in \text{Poss}} \tau(s) \wedge a \right) \quad \psi_s := \bigcirc\left(\varphi^{one}_{\mathcal{A}} \wedge \bigvee_{(s,a) \in \text{Poss}} \tau(s) \wedge a \right)$$

Consistent with our notation for planning, we use s for subsets of $\mathcal{F}$ and A for subsets of $\mathcal{A}$. We let $\tau(s) := \bigwedge_{f \in s} f \wedge \bigwedge_{f \in \mathcal{F} \setminus s} \neg f$ and set $\varphi^{one}_{\mathcal{A}} := \bigvee_{a \in \mathcal{A}} a \wedge \bigwedge_{a, a' \in \mathcal{A}, a' \neq a} \neg(a \wedge a')$. Note that in ψ_e and ψ_s, we use $\bigcirc$ in front of arbitrary propositional formulas, but by pushing $\bigcirc$ down to the fluents and actions, we obtain formulae of the required form.

The components of $G_{\mathcal{P}}$ mimic the dynamics of $\mathcal{P}$, according to the semantics discussed in Section 4. In the game $G_{\mathcal{P}}$, the environment controls the fluents ($\mathcal{X} := \mathcal{F}$), and the system controls the actions ($\mathcal{Y} := \mathcal{A}$). The formulas ψ_e and ψ_s reproduce the state transition model in $\mathcal{D}$. Formula ψ_e checks whether state s' in a sequence $(s \cup A)(s' \cup A')$ of two consecutive game states is a legal move of the environment player, relative to s and A – assuming that previous moves of each player respected the dynamics of the game. This occurs when $s' \in \delta(s, a)$. Formula ψ_s checks whether $A' \subseteq \mathcal{A}$ is a legal move of the system player, relative to s, A, and s' – assuming that previous moves of each player respected the dynamics of the game. This occurs when (i) A' contains a single action; and (ii) action $a \in A'$ is applicable in s'. Finally, formulas θ_e and θ_s are assertions over $\mathcal{X}$ and $\mathcal{X} \cup \mathcal{Y}$, respectively, and check that the initial move of the environment and system players is legal – that is, the environment sets the initial state to φ_I, and the system responds with an action that is applicable in the initial state. The auxiliary formula $\varphi^{one}_{\mathcal{A}}$ ensures that exactly one action symbol holds in each game position, making it possible to translate winning game strategies into solutions. Note that our reduction does not require φ_I to define a unique state.

Theorem 6. *Plan existence of an* LTL *FOND planning problem $\mathcal{P}$ can be reduced to winner determination of the game structure $G_{\mathcal{P}}$ and, by extension, to* LTL *realizability of*

$$\varphi_{\mathcal{P}} := (\theta_e \to \theta_s) \wedge (\theta_e \to (\psi_s \mathcal{W} \neg\psi_e)) \wedge ((\Box\psi_e \wedge \theta_e) \to \varphi_G))$$

Furthermore, the reduction is such that solutions can be directly extracted from winning strategies.

Encoding STRIPS Models When we represent planning problems with propositional STRIPS-like models, formulas ψ_e and ψ_s can be written more compactly, as follows:

$$\psi^{eff}_a := a \to \bigvee_{e \in Eff_a} \left(\bigwedge_{f \in Add_e} \bigcirc f \wedge \bigwedge_{f \in Del_e} \bigcirc\neg f \wedge \bigwedge_{f \in F \setminus Add_e \cup Del_e} f \leftrightarrow \bigcirc f \right)$$

$$\psi_e := \bigwedge_{a \in \mathcal{A}} \psi^{eff}_a \qquad \theta_s := \left(\varphi^{one}_{\mathcal{A}} \wedge \bigwedge_{a \in \mathcal{A}} \left(\neg a \vee \bigwedge_{p \in Pre_a} p \right) \right) \qquad \psi_s := \bigcirc\theta_s$$

Observe that ψ^{eff}_a captures both the positive and negative effects of non-deterministic action a, as well as frame assumptions – that the truth value of every other fluent, not affected by the action a, stays the same. In particular, formulae $\bigwedge_{f \in F \setminus Add_e \cup Del_e} f \leftrightarrow \bigcirc f$ capture the frame axioms for effect $e \in Eff_a$. We can write these as successor state axioms to realize a more parsimonious solution to the frame problem (Reiter 2001).

Complexity Analysis Our reduction of LTL FOND planning into structured games conveniently decouples the domain from the goal formula. This allows us to characterize the complexity of LTL FOND planning in terms of these two components and obtain analogous results to those in game structures (cf. Theorem 4), The same complexity results have been independently obtained very recently (available on arXiv, (Aminof et al. 2018)) with a different approach.

Theorem 7. *Plan existence in* LTL *FOND planning is EXP-complete in domain complexity, and 2EXP-complete for goal and combined complexity.*

Proof sketch. Membership follows from the reduction to games in Theorem 6 and results in Theorem 4. Hardness is obtained by reducing LTL_f FOND into LTL FOND (see Lemma 1), and then applying the hardness results for LTL_f FOND from (De Giacomo and Rubin 2018). □

Remark 1. The results in Theorems 6 and 7 also apply to LTL FOND planning with partially specified initial states.

5.3 $\mathsf{LTL_f}$ FOND Planning as Structured Games

It is also possible to reduce $\mathsf{LTL_f}$ FOND planning to structured games. We show this by reducing $\mathsf{LTL_f}$ FOND to LTL FOND, from which Theorem 6 can be applied. The reduction of an $\mathsf{LTL_f}$ FOND problem $\mathcal{P} = \langle \mathcal{D}, \varphi_I, \varphi_{fin} \rangle$ with domain $\mathcal{D} = \langle \mathcal{F}, \mathcal{A}, \delta, \mathrm{Poss} \rangle$ is an LTL FOND problem $\langle \mathcal{D}', \varphi'_I, \varphi_{inf} \rangle$ whose domain, $\mathcal{D}' := \langle \mathcal{F}', \mathcal{A}', \delta', \mathrm{Poss}' \rangle$, has symbols $\mathcal{F}' := \mathcal{F} \cup \{alive\}$ and $\mathcal{A}' := \mathcal{A} \cup \{a_{end}\}$. The dynamics of $\mathcal{D}'$ simulate plan executions in $\mathcal{D}$. Symbol a_{end} indicates that the simulated plan execution has terminated, and symbol *alive* indicates the opposite. Formally:

$$\mathrm{Poss}' := \{(s \cup \{alive\}, a) \mid (s, a) \in \mathrm{Poss}\} \cup \left\{(s, a_{end}) \mid s \in 2^{\mathcal{F}'}\right\}$$

$$\delta'(s, a) := \begin{cases} \{s' \cup \{alive\}\}_{s' \in \delta(s \setminus \{alive\}, a)} & \text{if } a \neq a_{end} \text{ and } s \models alive \\ \{\emptyset\} & \text{otherwise} \end{cases}$$

The new initial state is given by $\varphi'_I := \varphi_I \wedge (alive)$. Finally, LTL goal formula $\varphi_{inf} := \mathsf{tr}(\varphi_{fin}) \wedge \Diamond\neg(alive)$ is obtained by applying the following polynomial transformation to φ_{fin}:

$$\mathsf{tr}(p) = p \quad \mathsf{tr}(\neg\varphi) = \neg\mathsf{tr}(\varphi) \quad \mathsf{tr}(\varphi_1 \wedge \varphi_2) = \mathsf{tr}(\varphi_1) \wedge \mathsf{tr}(\varphi_2)$$
$$\mathsf{tr}(\bigcirc\varphi) = \bigcirc(alive \wedge \mathsf{tr}(\varphi)) \quad \mathsf{tr}(\varphi_1 \mathcal{U} \varphi_2) = \mathsf{tr}(\varphi_1)\mathcal{U}(alive \wedge \mathsf{tr}(\varphi_2))$$

Intuitively, an infinite trace π satisfies φ_{inf} iff *alive* is false at some point, and the finite prefix π' in which *alive* is true *simulates* a finite trace that satisfies φ_{fin}. The preceding transformation is based on the transformations found in (De Giacomo and Vardi 2013; Camacho, Bienvenu, and McIlraith 2018), but fixes some minor mistakes and makes some simplifications by leveraging the dynamics induced by a_{end}.

The transformation preserves plan existence (Lemma 1). Moreover, solutions σ_{fin} to the $\mathsf{LTL_f}$ FOND planning problem can be easily defined from solutions σ_{inf} to the LTL FOND planning reduction as $\sigma_{fin}(s_0 \cdots s_n) := \sigma_{inf}(s'_0 \cdots s'_n)$, where primed states are augmented with predicate *alive*, and execution terminates (end) whenever σ_{inf} outputs a_{end}.

Lemma 1. *Plan existence of an* $\mathsf{LTL_f}$ *FOND planning problem* $\langle \mathcal{D}, \varphi_I, \varphi_{fin} \rangle$ *can be polynomially reduced to plan existence of an* LTL *FOND planning problem* $\langle \mathcal{D}', \varphi'_I, \varphi_{inf} \rangle$.

Combining Lemma 1 and Theorem 6 yields:

Theorem 8. *Plan existence of an* $\mathsf{LTL_f}$ *FOND planning problem* $\mathcal{P}$ *can be reduced to winner determination of a game structure and, by extension, to* LTL *realizability.*

Encoding STRIPS Models With a propositional STRIPS-like representation of the planning problems, the translation of a domain $\mathcal{D}$ with symbols $\mathcal{F}, \mathcal{A}$ produces a new domain, $\mathcal{D}'$, with symbols $\mathcal{F}' := \mathcal{F} \cup \{alive\}$ and $\mathcal{A}' := \mathcal{A} \cup \{a_{end}\}$. Action a_{end} has preconditions $Pre_{a_{end}} := \{alive\}$ and a deterministic effect, $Eff_{a_{end}} = \{\langle \emptyset, \mathcal{F}' \rangle\}$, that deletes all fluents. Actions in $\mathcal{A}$ are updated to incorporate symbol *alive* in their preconditions.

Remark 2. The results in Lemma 1 and Theorem 8 apply to $\mathsf{LTL_f}$ FOND planning with partially specified initial states.

Complexity Analysis Complexity results for $\mathsf{LTL_f}$ FOND planning (with unique initial state) were recently published by De Giacomo and Rubin (2018). Membership can be easily deduced from our reductions. We extend their result to $\mathsf{LTL_f}$ FOND planning with partially specified initial states. Interestingly, the complexity of $\mathsf{LTL_f}$ FOND replicates that of LTL FOND. The same phenomenon occurs in LTL and $\mathsf{LTL_f}$ synthesis, both being 2EXP-complete problems (Pnueli and Rosner 1989; De Giacomo and Vardi 2015).

Theorem 9. *Plan existence for* $\mathsf{LTL_f}$ *FOND with partially specified initial states is EXP-complete in domain complexity, and 2EXP-complete in goal and combined complexity.*

Proof sketch. Hardness results follow from (De Giacomo and Rubin 2018). Membership follows from our reductions (cf. Lemma 1, Theorem 7, and Remarks 1 and 2). □

6 More Efficient LTL/$\mathsf{LTL_f}$ Plan Synthesis

In the previous sections, we formalized the connection between LTL synthesis, game structures, and various forms of planning. By doing so, we derived complexity results for planning that mimic those in LTL synthesis and games. While the domain complexity of LTL and $\mathsf{LTL_f}$ FOND planning is the same as for FOND with reachability goals, the overall complexity jumps to double-exponential time.

The aim of this section is to identify restricted fragments of LTL and $\mathsf{LTL_f}$ for which plan synthesis has the same worst-case complexity as FOND with reachability goals but which is richer in terms of its expressiveness. We first explore GR(1), a syntactically restricted fragment of LTL for which synthesis can be done in single-exponential time. Then we move to the finite case, and revisit the syntax of PDDL3.0 in the context of $\mathsf{LTL_f}$ FOND planning.

6.1 GR(1) Synthesis and Game Structures

Generalized Reactivity (1) (GR(1) for short) is a fragment of LTL that has been studied in the context of synthesis and games for its computational advantages. Inconsistent use of terminology in the literature has created some confusion regarding what the GR(1) fragment is, how GR(1) synthesis differs from GR(1) games, and the complexity of these tasks. We provide a brief historical overview, with the aim of clarifying each of these concepts as we understand them.

The GR(1) fragment of LTL was originally introduced by Piterman, Pnueli, and Sa'ar (2006), and comprised the fragment of LTL formulae in the syntactic form

$$\bigwedge_{i \in \mathcal{I}} \Box\Diamond\alpha_i \rightarrow \bigwedge_{j \in \mathcal{J}} \Box\Diamond\beta_j$$

where α_i and β_j are propositional formulae. *GR(1) synthesis* can be done with exponentially lower worst-case complexity than general LTL synthesis (Theorem 10).

Theorem 10 (Piterman et al. 2006)**.** *Synthesis of specifications* $\langle \mathcal{X}, \mathcal{Y}, \varphi \rangle$ *with GR(1) formula* φ *can be done in exponential time on the number of variables in* φ*.*

Piterman et al. studied the problem of computing winning strategies for so-called *GR(1) games*, that is, game structures with GR(1) winning condition. They provided a first reduction from GR(1) games into standard assume-guarantee LTL synthesis. That reduction contained a fundamental flaw that produced false unrealizable specifications. The flaw was detected and corrected by Klein and Pnueli (2010), and subsequently included in (Bloem et al. 2012) in the form given in Theorem 2, namely:

$$(\theta_e \rightarrow \theta_s) \wedge (\theta_e \rightarrow (\psi_s \mathcal{W} \neg\psi_e)) \wedge ((\Box\psi_e \wedge \theta_e) \rightarrow \varphi))$$

Similar to GR(1) synthesis, GR(1) games can be solved with exponentially lower hardness bounds than for general LTL winning conditions. Perhaps better known than Theorem 11 is the tractability result for GR(1) games in Corollary 1. It is worth noting that the polynomial time bounds are only valid *when the size of the game search space is fixed*.

Theorem 11. *[Bloem et al. 2012] GR(1) games with winning condition φ can be solved in polynomial time on the size of the search space, and exponential time in the number of variables in φ.*

Corollary 1. *For a fixed search space, GR(1) games can be solved in polynomial time.*

Despite the fact that GR(1) is a syntactically restricted fragment of LTL, many non-GR(1) properties can be captured by GR(1) games by adding extra *control* variables. Maoz and Ringert (2015) gave a method to reduce LTL realizability into winner determination for a GR(1) game. The approach works for any LTL formula that can be transformed into *deterministic* Büchi word automata (DBW). This is the case for 54 out of the 56 industrial patterns listed in (Dwyer, Avrunin, and Corbett 1999), highlighting the strong appeal of the GR(1) fragment for practical applications. The construction first takes an LTL description of the pattern and transforms it into a DBW. In a second step, an LTL formula is constructed with extra controllable variables – one per automaton state – that capture the Büchi acceptance condition. The syntactic form of the formula makes it possible to apply existing methods that reduce LTL realizability to winner determination of a GR(1) game.

6.2 Planning with GR(1) Goals

The advantageous computational properties of the GR(1) fragment for game structures also manifest in LTL FOND planning (Theorem 13). This is because there exists a bidirectional mapping between game structures and LTL planning *when the goal and winning condition is a GR(1) formula*: GR(1) games can be reduced to LTL planning with GR(1) goals and, conversely, LTL planning with GR(1) goals can be reduced to GR(1) LTL planning.

Theorem 12. *GR(1) games can be polynomially reduced to* LTL *FOND planning with GR(1) goals, and vice versa.*

Proof sketch. For the first direction, we employ the reduction from Theorem 5. Observe that $\varphi_G := \Diamond x_{end} \vee \bigcirc\bigcirc\varphi$ is equivalent to $\varphi'_G := \Diamond x_{end} \vee \varphi$ when $\varphi = \bigwedge_{i\in\mathcal{I}} \Box\Diamond\alpha_i \rightarrow \bigwedge_{j\in\mathcal{J}} \Box\Diamond\beta_j$ is a GR(1) formula. Further observe that, for the constructed planning problem, an execution satisfies $\Diamond x_{end}$ iff it satisfies $\neg\Box\Diamond\neg x_{end}$. This is because x_{end} cannot be made false once it has been made true. By means of syntactic manipulation, we can further show that an execution satisfies φ_G iff it satisfies the GR(1) formula $(\Box\Diamond\neg x_{end} \wedge \bigwedge_{i\in\mathcal{I}} \Box\Diamond\alpha_i) \rightarrow \bigwedge_{j\in\mathcal{J}} \Box\Diamond\beta_j$. For the second statement, we can reuse the reduction underlying Theorem 6, as it does not modify goal formulae. □

Theorem 13. *Plan existence for* LTL *FOND planning problems with GR(1) goals is EXP-complete in domain and combined complexity, and in PTIME w.r.t. goal complexity (i.e., when the domain is fixed).*

Proof sketch. Membership results are obtained by combining Theorems 11, 12, and Corollary 1. The EXP lower bound follows from the EXP-hardness of FOND and the fact that reachability goals can encoded using GR(1) formulae. □

The results in Theorem 13 can be extended to the finite case: plan synthesis of FOND problems with $\mathsf{LTL_f}$ goal in GR(1) syntactic form can be done in exponential time. Unfortunately, this result is not very interesting because $\mathsf{LTL_f}$ FOND planning with GR(1) goals is equivalent to FOND planning with final-state goals.

Property 2. The GR(1) syntactic fragment of $\mathsf{LTL_f}$ is equally expressive as the syntactic fragment of formulas of the form $\Diamond(\theta \wedge \mathsf{final})$ for some propositional formula θ.

Proof. Satisfaction of GR(1) $\mathsf{LTL_f}$ formula $\bigwedge_{i\in\mathcal{I}} \Box\Diamond\alpha_i \rightarrow \bigwedge_{j\in\mathcal{J}} \Box\Diamond\beta_j$ is equivalent to satisfaction of $\mathsf{LTL_f}$ formula $\Diamond(\theta \wedge \mathsf{final})$ with $\theta = \bigwedge_{i\in\mathcal{I}} \alpha_i \rightarrow \bigwedge_{j\in\mathcal{J}} \beta_j$. Conversely, satisfaction of $\mathsf{LTL_f}$ formula $\Diamond(\theta \wedge \mathsf{final})$ is equivalent to satisfaction of GR(1) $\mathsf{LTL_f}$ formula $\Box\Diamond\top \rightarrow \Box\Diamond\theta$. □

6.3 Request-Response Planning

We propose here the class of *request-response* formulas: a syntactic class that appear to be a reasonable counterpart to GR(1) in the finite case, as it captures similar patterns. Request-response formulas take the form $\Box(\alpha \rightarrow \Diamond\beta)$, for some propositional formulas α and β. Request-response LTL formulas can be transformed into GR(1) LTL formulas that capture the same patterns, possibly with introduction of two extra variables using Maoz and Ringert (2015)'s technique. Therefore, LTL games, synthesis, and planning for request-response LTL formulas can be polynomially reduced into their GR(1) counterparts. In contrast to GR(1), request-response $\mathsf{LTL_f}$ formulas are more expressive than final-state goals when interpreted over finite traces.

We omit the proof of Theorem 14, as it can be obtained from the more general Theorem 15.

Theorem 14. *Plan existence for* $\mathsf{LTL_f}$ *FOND with request-response goal formulas is EXP-complete for domain and combined complexity, and in PTIME for goal complexity.*

6.4 Planning with PDDL3.0 Goals

The *Planning Domain Definition Language* (PDDL) is a globally accepted language used to describe planning problems. Different variants of PDDL exist that allow for e.g. durative actions and conditional effects. PDDL3.0 (Gerevini et al. 2009), which we abbreviate to PDDL3, extends the language of PDDL with the modal temporal operators listed in Table 1, to be interpreted over finite traces. We included equivalent $\mathsf{LTL_f}$ formulae, as shown in (Gerevini et al. 2009; De Giacomo, Masellis, and Montali 2014). We provided corrected versions for the operators `hold-during` and `hold-after`. We also simplified the formula for `sometime-before` and `at-most-once`.

	modal operator	equivalent $\mathsf{LTL_f}$	size of DFA
	(`at-end` θ)	$\Diamond(\theta \wedge \mathsf{final})$	2
	(`always` θ)	$\Box\theta$	3
	(`sometime` θ)	$\Diamond\theta$	3
	(`sometime-after` θ_1 θ_2)	$\Box(\theta_1 \rightarrow \Diamond\theta_2)$	3
	(`sometime-before` θ_1 θ_2)	$\theta_2\mathcal{R}\neg\theta_1$	4
	(`at-most-once` θ)	$(\neg\theta)\mathcal{W}(\theta\mathcal{W}(\Box\neg\theta))$	5
	(`within` n θ)	$\bigvee_{0\le i\le n} \bigcirc^i\theta$	$n+3$
	(`always-within` n θ_1 θ_2)	$\Box(\theta_1 \rightarrow \bigvee_{0\le i\le n} \bigcirc^i\theta_2)$	$n+3$
*	(`hold-during` n_1 n_2 θ)	$\bigvee_{0\le i\le n_1} \bigcirc^i(\theta \wedge \mathsf{final})$ $\vee \bigwedge_{n_1<i\le n_2} \bullet^i(\theta)$	n_1+n_2+2
	(`hold-after` n θ)	$\bigcirc^{n+1}\theta$	$2n+5$
*	(`hold-after` n θ)	$\bigvee_{0\le i\le n+1} \bigcirc^i(\theta \wedge \mathsf{final})$	$2n+5$

Table 1: PDDL3 modal temporal operators, equivalent $\mathsf{LTL_f}$ formulae, and size of the corresponding minimal DFA. Nested operators are abbreviated with a superindex. Operator *weak-next* ($\bullet$) is defined as $\bullet\varphi := \mathsf{final} \vee \bigcirc\varphi$. We provide corrected $\mathsf{LTL_f}$ formulae for operators tagged with $*$.

The syntax of PDDL3 does not allow nesting of modal operators, and is limited to conjunctions of modal operators and literals. Despite its restricted syntax, PDDL3 temporal operators subsume GR(1) and request-response formulas. In this paper, all PDDL3 formulae apply PDDL3 modal operators to propositional formulae, but note that PDDL3 also allows for a lifted syntax.

Relation with Request-Response formulas Request-response formulas $\Box(p \rightarrow \Diamond q)$ correspond to PDDL3 operator (`sometime-after` p q).

Relation with GR(1) formulas When interpreted over finite traces, GR(1) formulas $\bigwedge_i \Box\Diamond\alpha_i \rightarrow \bigwedge_j \Box\Diamond\beta_j$ are equivalent to PDDL3 operator (`at-end` θ), with $\theta = \bigwedge_i \alpha_i \rightarrow \bigwedge_j \beta_j$.

Relation with FOND planning FOND planning with final-state goals is a special case of FOND planning with PDDL 3 goals in the form (`at-end` θ). Solutions in the form of a policy can be constructed from strategies (cf. Proposition 1).

Reduction to Structured Games FOND planning with PDDL3 goals can be reduced to $\mathsf{LTL_f}$ FOND by rewriting PDDL3 operators in $\mathsf{LTL_f}$, as indicated in Table 1.

Complexity Analysis The following theorem clarifies the complexity of FOND planning with PDDL3 goals, which to the best of our knowledge has not yet been considered in the literature. The formulation is a bit more involved than earlier results due to the use of numeric values in some of the PDDL3 temporal operators.

Theorem 15. *Assuming a unary encoding of numbers in PDDL3 modal operators, plan existence for FOND planning with PDDL3 goals is EXP-complete for domain and goal complexity, and the goal complexity is in PTIME provided that there is a bound on the number of conjuncts in the goal that use numeric temporal operators.*

Proof sketch. FOND planning with PDDL3 goals can be reduced to $\mathsf{LTL_f}$ FOND (cf. Table 1). $\mathsf{LTL_f}$ FOND planning can be reduced to solving a DFA game[3] following a similar technique as in Section 5.2. The game DFA is the product of (i) a DFA encoding the transition system (which is exponential, (ii) multiple DFA, one per PDDL3 operator in the goal formula. The former DFA is of exponential size in the domain, and each of the latter DFAs is of polynomial size (see third column of Table 1), assuming a unary representation of numeric values. The product DFA is thus of exponential size, and if the domain is fixed, the restriction on the number of numeric conjuncts suffices to obtain a polynomial-size DFA. We conclude by recalling that DFA games can be solved in polynomial time in the size of the DFA (see e.g. (Camacho, Bienvenu, and McIlraith 2018)). □

7 Summary and Discussion

Synthesizing strategies for software agents is a central problem in artificial intelligence. In this work, we made a step towards reconciling two different perspectives that address this problem: AI planning and LTL synthesis.

Our work contributes with principled, bidirectional mappings between so-called LTL FOND planning, two-player games, and reactive LTL synthesis. By means of these mappings, we were able to revisit well-known complexity results in LTL and $\mathsf{LTL_f}$ FOND planning, and derive new ones.

A further contribution was the identification of restricted forms of LTL and $\mathsf{LTL_f}$ FOND planning for which plan synthesis can be realized more efficiently than in the general case. For LTL FOND planning, we derived results for GR(1) goals that are analogous to so-called GR(1) games. For the finite-trace setting, we provided novel complexity results for $\mathsf{LTL_f}$ FOND planning with PDDL3.0 goals and observed that, over finite traces, its expressivity subsumes other fragments including GR(1). These results show that LTL FOND and $\mathsf{LTL_f}$ FOND plan synthesis can be performed with the same computational complexity as traditional FOND planning with final-state goals, but utilizing a richer class of temporally extended goal formulae.

Acknowledgements

We gratefully acknowledge funding from the Natural Sciences and Engineering Research Council of Canada (NSERC) and Microsoft Research.

References

Albarghouthi, A.; Baier, J. A.; and McIlraith, S. A. 2009. On the use of planning technology for verification. In *VVPS@ICAPS*.

Aminof, B.; De Giacomo, G.; Murano, A.; and Rubin, S. 2018. Planning and synthesis under assumptions. *CoRR* abs/1807.06777.

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

Baier, J., and McIlraith, S. 2006a. Planning with temporally extended goals using heuristic search. In *ICAPS*, 342–345.

[3] In the context of this paper, DFA games can be seen as structured games where the winning condition is given by the acceptance of DFA.

Baier, J. A., and McIlraith, S. A. 2006b. Planning with first-order temporally extended goals using heuristic search. In *AAAI*, 788–795.

Baier, J. A.; Bacchus, F.; and McIlraith, S. A. 2009. A heuristic search approach to planning with temporally extended preferences. *Artificial Intelligence* 173(5-6):593–618.

Bienvenu, M.; Fritz, C.; and McIlraith, S. A. 2011. Specifying and computing preferred plans. *Artificial Intelligence* 175(7–8):1308–1345.

Bloem, R.; Jobstmann, B.; Piterman, N.; Pnueli, A.; and Sa'ar, Y. 2012. Synthesis of reactive(1) designs. *Journal of Computer and System Sciences* 78(3):911–938.

Camacho, A.; Triantafillou, E.; Muise, C. J.; Baier, J. A.; and McIlraith, S. A. 2017. Non-deterministic planning with temporally extended goals: LTL over finite and infinite traces. In *AAAI*, 3716–3724.

Camacho, A.; Baier, J. A.; Muise, C. J.; and McIlraith, S. A. 2018a. Finite LTL synthesis as planning. In *ICAPS*, 29–38.

Camacho, A.; Baier, J. A.; Muise, C. J.; and McIlraith, S. A. 2018b. Synthesizing controllers: On the correspondence between LTL synthesis and non-deterministic planning. In *Canadian AI*, 45–59.

Camacho, A.; Muise, C. J.; Baier, J. A.; and McIlraith, S. A. 2018c. LTL realizability via safety and reachability games. In *IJCAI*, 4683–4691.

Camacho, A.; Muise, C. J.; Baier, J. A.; and McIlraith, S. A. 2018d. SynKit: LTL synthesis as a service. In *IJCAI*, 5817–5819.

Camacho, A.; Bienvenu, M.; and McIlraith, S. A. 2018. Finite LTL synthesis with environment assumptions and quality measures. In *KR*, 29–38.

Church, A. 1957. Applications of recursive arithmetic to the problem of circuit synthesis. *Summaries of the Summer Institute of Symbolic Logic, Cornell University 1957* 1:3–50.

Coles, A., and Coles, A. 2011. LPRPG-P: relaxed plan heuristics for planning with preferences. In *ICAPS*, 26–33.

De Giacomo, G., and Rubin, S. 2018. Automata-theoretic foundations of FOND planning for LTLf and LDLf goals. In *IJCAI*, 4729–4735.

De Giacomo, G., and Vardi, M. Y. 2013. Linear temporal logic and linear dynamic logic on finite traces. In *IJCAI*, 854–860.

De Giacomo, G., and Vardi, M. Y. 2015. Synthesis for LTL and LDL on finite traces. In *IJCAI*, 1558–1564.

De Giacomo, G.; Felli, P.; Patrizi, F.; and Sardiña, S. 2010. Two-player game structures for generalized planning and agent composition. In *AAAI*, 297–302.

De Giacomo, G.; Gerevini, A. E.; Patrizi, F.; Saetti, A.; and Sardiña, S. 2016. Agent planning programs. *Artificial Intelligence* 231:64–106.

De Giacomo, G.; Masellis, R. D.; and Montali, M. 2014. Reasoning on LTL on finite traces: Insensitivity to infiniteness. In *AAAI*, 1027–1033.

De Giacomo, G.; Patrizi, F.; and Sardiña, S. 2010. Generalized planning with loops under strong fairness constraints. In *KR*, 351–361.

D'Ippolito, N.; Rodríguez, N.; and Sardiña, S. 2018. Fully observable non-deterministic planning as assumption-based reactive synthesis. *Journal of Artificial Intelligence Research* 61:593–621.

Doherty, P., and Kvarnström, J. 2001. TALplanner: A temporal logic-based planner. *AI Magazine* 22(3):95–102.

Dwyer, M. B.; Avrunin, G. S.; and Corbett, J. C. 1999. Patterns in property specifications for finite-state verification. In *ICSE*, 411–420.

Edelkamp, S. 2006. On the compilation of plan constraints and preferences. In *ICAPS*, 374–377.

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

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

Ghallab, M.; Nau, D. S.; and Traverso, P. 2016. *Automated Planning and Acting*. Cambridge University Press.

Green, C. C. 1969. Application of theorem proving to problem solving. In *IJCAI*, 219–240.

Jacobs, S.; Klein, F.; and Schirmer, S. 2016. A high-level LTL synthesis format: TLSF v1.1. In *SYNT*, 112–132.

Klein, U., and Pnueli, A. 2010. Revisiting synthesis of GR(1) specifications. In *HVC*, 161–181.

Maoz, S., and Ringert, J. O. 2015. GR(1) synthesis for LTL specification patterns. In *ESEC/FSE*, 96–106.

Nilsson, N. J. 1969. A mobile automaton: An application of artificial intelligence techniques. In *IJCAI*, 509–520.

Patrizi, F.; Lipovetzky, N.; De Giacomo, G.; and Geffner, H. 2011. Computing infinite plans for LTL goals using a classical planner. In *IJCAI*, 2003–2008.

Patrizi, F.; Lipovetzky, N.; and Geffner, H. 2013. Fair LTL synthesis for non-deterministic systems using strong cyclic planners. In *IJCAI*, 2343–2349.

Piterman, N.; Pnueli, A.; and Sa'ar, Y. 2006. Synthesis of reactive(1) designs. In *VMCAI*, 364–380.

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

Pnueli, A., and Rosner, R. 1989. On the synthesis of a reactive module. In *POPL*, 179–190.

Pnueli, A. 1977. The temporal logic of programs. In *FOCS*, 46–57.

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

Torres, J., and Baier, J. A. 2015. Polynomial-time reformulations of LTL temporally extended goals into final-state goals. In *IJCAI*, 1696–1703.

Triantafillou, E.; Baier, J.; and McIlraith, S. 2015. A unifying framework for planning with ltl and regular expressions. In *MOCHAP@ICAPS*, 23–31.

Waldinger, R. J., and Lee, R. C. T. 1969. PROW: A step toward automatic program writing. In *IJCAI*, 241–252.

Zhu, S.; Tabajara, L. M.; Li, J.; Pu, G.; and Vardi, M. Y. 2017. Symbolic LTLf synthesis. In *IJCAI*, 1362–1369.

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