Strong Fully Observable Non-Deterministic Planning with LTL and LTLf Goals

Alberto Camacho and **Sheila A. McIlraith**
Department of Computer Science, University of Toronto, Toronto, Canada
Vector Institute, Toronto, Canada
{acamacho, sheila}@cs.toronto.edu

Abstract

We are concerned with the synthesis of strategies for sequential decision-making in non-deterministic dynamical environments where the objective is to satisfy a prescribed temporally extended goal. We frame this task as a *Fully Observable Non-Deterministic* planning problem with the goal expressed in *Linear Temporal Logic* (LTL), or LTL interpreted over finite traces ($\mathsf{LTL_f}$). While the problem is well-studied theoretically, existing algorithmic solutions typically compute so-called strong-cyclic solutions, which are predicated on an assumption of fairness. In this paper we introduce novel algorithms to compute so-called strong solutions, that guarantee goal satisfaction even in the absence of fairness. Our strategy generation algorithms are complemented with novel mechanisms to obtain proofs of unsolvability. We implemented and evaluated the performance of our approaches in a selection of domains with LTL and $\mathsf{LTL_f}$ goals.

1 Introduction

We are concerned with the synthesis of strategies in discrete, dynamical environments that are *fully observable* and where the outcomes of actions are *non-deterministic* (FOND). FOND planning provides a computational core for many applications and sequential decision-making problems including two-player games, web-service composition, and the synthesis of controllers for devices and systems. In FOND planning, two classes of solutions are commonly considered: *strong* solutions guarantee eventual satisfaction of the goal, whereas in *strong-cyclic* solutions such guarantee is predicated on the assumption that the dynamics of the system is *fair* (e.g., [Cimatti *et al.*, 2003]).

The focus of this paper is on the development of algorithms for the synthesis of solutions to FOND planning with temporally extended goals – in particular, strong solutions. Different languages have been used for the specification of temporally extended goals in planning. In this paper, we follow on recent work in so-called LTL FOND and $\mathsf{LTL_f}$ FOND planning that use *Linear Temporal Logic* (LTL) and LTL interpreted over finite traces ($\mathsf{LTL_f}$), respectively (e.g. [Patrizi *et al.*, 2013; Camacho *et al.*, 2017; De Giacomo and Rubin, 2018; Camacho *et al.*, 2019]). Our techniques complement early work on FOND planning with temporally extended goals expressed in CTL [Pistore *et al.*, 2001]. Note, CTL and LTL are non-equivalent languages, although both can be transformed into automata. Our techniques exploit automata transformations of the goal specification, and can therefore be extended for use with any goal specification language that can be transformed into automata – e.g. PSL, CTL, Past LTL (PLTL), and (finite) Linear Dynamic Logic ($\mathsf{LDL_f}$) [De Giacomo and Vardi, 2013], as well as PDDL3.0 temporal operators [Gerevini *et al.*, 2009].

	Type of solutions	
	strong-cyclic	strong
$\mathsf{LTL_f}$ FOND	Camacho *et al.* [2017]	This paper
LTL FOND	Camacho *et al.* [2017] Patrizi *et al.* [2011]	This paper

Table 1: Existing approaches to FOND planning with $\mathsf{LTL_f}$ and LTL goals. Patrizi et al.'s approach is limited to a subset of LTL.

The complexity of LTL FOND and $\mathsf{LTL_f}$ FOND planning is now well understood [De Giacomo and Rubin, 2018; Camacho *et al.*, 2019]. The latter paper provides a clear correspondence with LTL synthesis and game structures, and identifies fragments of LTL and $\mathsf{LTL_f}$ for which planning can be done more efficiently. Despite increasing interest in the area, the number of algorithmic approaches to LTL FOND and $\mathsf{LTL_f}$ FOND planning remains limited. To date, existing tools have focused on the computation of strong-cyclic solutions whose validity is predicated on the fairness assumption (cf. Table 1). The synthesis of strong solutions had remained elusive, and is the focus of this paper.

We provide the first algorithmic approach to strong LTL FOND and $\mathsf{LTL_f}$ FOND planning. Our approach exploits automata transformations of the goal formula, and compiles the problem into a standard goal-oriented FOND planning problem that can be solved with off-the-shelf planners. We further provide an analysis of the different sources of algorithmic complexity. Whereas we adopt LTL and $\mathsf{LTL_f}$ as goal specification languages, our techniques can be extended to any language that can be transformed into automata. This includes PSL, CTL, PLTL, $\mathsf{LDL_f}$, and PDDL3.0 temporal operators.

Our algorithms have been implemented and evaluated empirically. Experiments demonstrate the effectiveness of our approach to computing strong solutions. They also suggest that assuming fairness and computing strong-cyclic solutions via existing approaches can ease the burden of computation.

2 Preliminaries

2.1 Linear Temporal Logic

Linear Temporal Logic (LTL) over a finite set of atomic propositions AP extends propositional logic with temporal operators *next* ($\bigcirc$) and *until* ($\mathcal{U}$). Namely,

$$\varphi ::= p \mid \varphi \wedge \varphi \mid \varphi \vee \varphi \mid \neg\varphi \mid \bigcirc\varphi \mid \varphi\mathcal{U}\varphi$$

where $p \in AP$. Other operators are usually considered, such as *eventually* ($\Diamond\varphi := \top\mathcal{U}\varphi$) and *always* ($\Box\varphi := \neg\Diamond\neg\varphi$). LTL was initially conceived for model checking of properties of logical circuits [Pnueli, 1977], and over the years it has been used in a myriad of applications: from specification language for the synthesis of programs [Pnueli and Rosner, 1989], to goal specification language for automated planning [Bacchus and Kabanza, 1998]. In automated planning in particular, variants of LTL interpreted over *finite* traces have been widely studied. Here, we use $\mathsf{LTL_f}$. The syntax of $\mathsf{LTL_f}$ is the same as for LTL, and differs in the semantics.

A finite trace $\rho = s_1 \cdots s_n$ is a model of $\mathsf{LTL_f}$ formula (and we write $\rho \models \varphi$) φ if $\rho, 1 \models \varphi$, where

- $\rho, i \models p$ if $p \in AP$ and $p \in s_i$
- $\rho, i \models \varphi$ if φ is a propositional formula and $s_i \models \varphi$
- $\rho, i \models \neg\varphi$ if $\rho, i \not\models \varphi$, i.e., it is not the case that $\rho, i \models \varphi$
- $\rho, i \models \varphi \wedge \varphi_2$ if $\rho, i \models \varphi_1$ and $\rho, i \models \varphi_2$
- $\rho, i \models \bigcirc\varphi$ if $i + 1 \leq |\rho|$ and $\rho, i + 1 \models \varphi$
- $\rho, i \models \varphi_1\mathcal{U}\varphi_2$ if $\rho, j \models \varphi_2$ for some $i \leq j \leq |\rho|$ and $\rho, k \models \varphi_1$ for every $i \leq k \leq j$

The semantics of LTL is similar, with the exception that formulae are evaluated over infinite traces (i.e., $|\rho| = \infty$). In the sequel, we refer to the *size* of an LTL or $\mathsf{LTL_f}$ formula φ as the number of temporal and logical operators, plus the number of atomic propositions in φ.

2.2 LTL and Automata

LTL and $\mathsf{LTL_f}$ have a well-known correspondence with automata, that we review here (cf. [Baier and McIlraith, 2006b; Kupferman and Vardi, 2005]). An *automaton* is a tuple $\mathcal{A} = \langle Q, \Sigma, q_0, T, \alpha\rangle$, where Q is a finite set of automaton states, Σ is a finite alphabet of input symbols (in this paper, $\Sigma = 2^{AP}$), $q_0 \in Q$ is the initial state, and $T \subseteq Q \times \Sigma \times Q$ is the transition relation. The transition relation is *deterministic* when a unique triplet $(q, s, q') \in T$ exists for each pair $(q, s) \in Q \times \Sigma$. A *run* of $\mathcal{A}$ on an infinite word $w = s_0 s_1 \cdots \in \Sigma^\omega$ is a (possibly infinite) sequence $q_0 q_1 \cdots$ where $(q_i, s_i, q_{i+1}) \in T$ for each $i \geq 0$. The set $\alpha \subseteq Q$ serves to characterize the set of runs that are *accepting*. Different types of automata exist according to their acceptance condition. Here, we are interested in *Universal Co-Büchi Word* (UCW) and *Non-deterministic Finite Word* (NFW) automata.

A UCW automaton accepts an infinite word w if *all* the runs on w hit a *finite* number of states in α. In contrast, a NFW automaton accepts a finite word w if *some* run on w has the same length as w and finishes in one of the states in α. NFW automata with deterministic transition relation are called *Deterministic Finite Word* (DFW) automata. We say that automaton $\mathcal{A}$ is a *transformation* of an LTL (resp. $\mathsf{LTL_f}$) formula φ if $\mathcal{A}$ accepts all and only the models of φ.

Property 1. LTL *(resp.* $\mathsf{LTL_f}$*) formulae can be transformed into UCW (resp. NFW) automata in worst-case exponential time, and so that the number of states is worst-case exponential in the size of the formula.*

Property 2. $\mathsf{LTL_f}$ *formulae can be transformed into DFW automata in worst-case double exponential time, and so that the number of states is worst-case double exponential in the size of the formula.*

3 Planning with Temporally Extended Goals

Automated planning is a popular model-based approach to sequential decision-making. The *world* dynamics – i.e., the dynamics of the system under observation – is assumed to be given to the agent by means of a *planning domain* that describes how the world reacts in response to the agent's actions. The objective in a planning problem is to synthesize an agent strategy that guarantees satisfaction of a prescribed goal. We formalize these concepts below.

3.1 Planning Domains

In this paper we focus on planning domains that are *fully observable and non-deterministic* (FOND). We represent FOND domains compactly (Definition 1), following the notation in Ghallab *et al.* [2016]. The set of *fluents*, $\mathcal{F}$, represents properties of the world. A planning state is a complete truth assignment over the set of fluent variables, and is usually represented with the subset of fluents $s \subseteq \mathcal{F}$ that hold true in such state. Regarding the dynamics, an action $a \in \mathcal{A}$ is *applicable* in a state s if $(s, a) \in \text{Poss}$. The *result* of applying action a in state s is a state $s' \in \delta(s, a)$. We say an action a is *deterministic* when $|\delta(s, a)| = 1$ in all states s for which a is applicable. Note that, in general, the result of applying an action is non-deterministic and cannot be predicted in advance.

Definition 1. *A* FOND planning domain *is a tuple* $\langle \mathcal{F}, \mathcal{A}, \delta, \text{Poss}\rangle$*, where* $\mathcal{F}$ *is a finite set of fluent symbols,* $\mathcal{A}$ *is a finite set of action symbols,* $\text{Poss} \subseteq 2^{\mathcal{F}} \times \mathcal{A}$*, and* $\delta : 2^{\mathcal{F}} \times \mathcal{A} \to 2^{2^{\mathcal{F}}}$*. The* size of the domain *is* $|\mathcal{F}|$.

Fairness over Executions. *Executions* of a planning domain $\mathcal{D} = \langle \mathcal{F}, \mathcal{A}, \delta, \text{Poss}\rangle$ from state s_0 are (possibly infinite) sequences of state-action pairs $(s_0, a_0)(s_1, a_1)\cdots$ such that $(s_n, a_n) \in \text{Poss}$ and $s_{n+1} \in \delta(s_n, a_n)$. It is common in FOND planning to presume *fairness* over executions, that is, that all the non-determinism of the actions manifests infinitely in the limit (cf. [Cimatti *et al.*, 2003]). An execution π is *unfair* if $|\pi| = \infty$ and there exists s, a and $s' \in \delta(s, a)$ such that the pair (s, a) appears infinitely often in π, but the number of occurrences of consecutive pairs $(s_1, a_1)(s_2, a_2)$ with $s_1 = s$, $a_1 = a$, and $s_2 = s'$ is finite. Fair executions are those that are not unfair.

3.2 Planning Problems

A FOND planning problem is a tuple $\langle \mathcal{D}, \varphi_I, \varphi_G \rangle$, where: $\mathcal{D}$ is a FOND domain; φ_I is a formula that describes the (complete) *initial state* of the world[1]; and φ_G is a formula that describes the objective of the agent.

Compact Problem Representations. Planning problems are commonly encoded with a compact representation of states, actions, and state transitions. For the reader not familiar with STRIPS-like languages, such as PDDL, a planning problem is a tuple $\langle F, I, O, G \rangle$, where F is the set of fluents, I is the initial state, O is the set of operators, and G is the goal condition. For final-state goals, G is usually the set of literals that hold true in a goal state. States are sets of fluents that represent what holds true in the world. For a fluent f, we write $s \models f$ if $f \in s$, and $s \models \neg f$ if $f \notin s$. For a set of literals Φ, we write $s \models \Phi$ if $s \models l$ for each $l \in \Phi$. An action $a \in O$ is defined in terms of its *preconditions* (Pre_a) and *effects* (Eff_a). The preconditions are a set of literals that need to hold true in a state s for a to be applied, that is, $s \models Pre_a$. The result of applying action a in s is a state s' characterised non-deterministically by one of the Eff_a^i in $\mathit{Eff}_a = \{\mathit{Eff}_a^1, \ldots, \mathit{Eff}_a^n\}$. Each Eff_a^i contains a set of conditional effects that add and delete fluents from s. More formally, each Eff_a^i contains pairs $c \to e$. A fluent f holds in s' iff: (i) $s \models c$ for some $c \to e$ in Eff_a^i and $f \in c$; or (ii) $s \models f$ and for all $c \to e$ in Eff_a^i such that $s \models c$, $\neg f \notin c$. Sometimes we abuse notation and write $\mathit{Eff}_a = \{c_j \to e_j\}_{j \in J} \cup \mathit{oneof}\, \{\mathit{Eff}_a^1, \ldots, \mathit{Eff}_a^n\}$ to denote that each $c_j \to e_j$ is a conditional effect in all Eff_a^i, and we omit writing it in their explicit description. We also omit writing c in $c \to e$ when $c = \emptyset$. An execution finishing in state s satisfies a final-state goal G if $s \models G$.

3.3 Strong and Strong-Cyclic Solutions

In this paper, solutions to a FOND planning problem take the form of strategies. Following Camacho *et al.* [2019], a *strategy* for a FOND planning domain $\mathcal{D} = \langle \mathcal{F}, \mathcal{A}, \delta, \mathrm{Poss} \rangle$ is a function $\sigma : (2^{\mathcal{F}})^+ \to \mathcal{A} \cup \{\mathsf{end}\}$. For a strategy to be well-defined, we require actions $\sigma(s_0 \cdots s_n)$ be applicable in s_n. An *execution* of strategy σ in domain $\mathcal{D}$ from initial state $s_0 \models \varphi_I$ is a sequence $\pi = (s_0, a_0)(s_1, a_1) \cdots$ with the following properties: (i) $a_n = \sigma(s_0 \cdots s_n)$ for each $n < |\pi|$; (ii) for each $n < |\pi| - 1$, $a_n \in \mathcal{A}$ is an action applicable in s_n and $s_{n+1} \in \delta(s_n, a_n)$ is a result of applying action a_n in s_n; (iii) if $n = |\pi| - 1$, then $a_n = \mathsf{end}$. Intuitively, end is a special action symbol that indicates the *termination* of the execution. Non-terminating executions are infinite, and we write $|\pi| = \infty$. Strategies can be implemented compactly as *finite-state controllers* (FSCs) (cf. [Geffner and Bonet, 2013]). Goal formulae are evaluated with respect to *execution traces*, which are simply rewritings of executions $\pi = (s_0, a_0)(s_1, a_1) \cdots$ in the form of sequences of sets $\rho_\pi = (s_0 \cup \{a_0\})(s_1 \cup \{a_1\}) \cdots$.

[1]If φ_I is only partially defined, the problem can be polynomially reduced to FOND planning with a (completely defined) dummy initial state s_I and dummy action a_I, only applicable at state s_I, that non-deterministically maps s_I to some state s (and henceforth φ_I) that models the original φ_I.

Definition 2 (Strong Solutions)**.** *A strategy σ is a* strong solution *to FOND planning problem $\langle \mathcal{D}, \varphi_I, \varphi_G \rangle$ with temporally extended goal formula φ_G if all execution traces of σ that commence in $s_0 \models \varphi_I$ satisfy φ_G.*

Definition 3 (Strong-Cyclic Solutions)**.** *A strategy σ is a* strong-cyclic solution *to FOND planning problem $\langle \mathcal{D}, \varphi_I, \varphi_G \rangle$ with temporally extended goal formula φ_G if all fair execution traces of σ that commence in $s_0 \models \varphi_I$ satisfy φ_G.*

Our definitions for the classes of strong and strong-cyclic solutions to FOND planning with temporally extended goals extend those defined by Cimatti *et al.* [2003] for FOND planning with final-state goals. We highlight three differences worth noting. First, we define solutions to be strategies, whereas solutions as for Cimatti *et al.* [2003]'s paper take the more restricted form of *policies*, or *memoryless* strategies σ such that $\sigma(s_0 \cdots s_n) = \sigma(s_n)$ for each $s_0 \cdots s_n$. Second, execution of solutions to FOND planning with temporally extended goals may not terminate. In contrast, (fair) executions of solutions to FOND planning with final-state goals always terminate upon achievement of a goal state. Finally, strong policies that are solution to a FOND planning problem with final-state goal are acyclic. This property no longer holds, in general, for temporally extended goals.

3.4 FOND Planning with LTL and LTLf Goals

Throughout the paper, we use LTL and $\mathsf{LTL_f}$ as goal specification languages and refer to the subsequent problems as LTL FOND and $\mathsf{LTL_f}$ FOND, respectively. Nevertheless, our techniques can be extended to any formal language that can be transformed into automata (e.g. PSL, CTL, PLTL, and $\mathsf{LDL_f}$). For simplicity, we limit the algorithms presented here to temporal formulas defined over the set of propositional variables $AP = \mathcal{F}$. Note, our techniques can be naturally extended to handle propositional variables $\mathcal{F} \cup \mathcal{A}$, allowing goal formulae to describe state-action execution traces by means of the addition of new fluents $\mathrm{occ}(a)$ that are made true after each action a is executed (cf. [Bienvenu *et al.*, 2011]).

$\mathsf{LTL_f}$ **FOND Planning.** The model of $\mathsf{LTL_f}$ FOND planning is concerned with the synthesis of *terminating* strategies whose executions satisfy a prescribed temporally extended goal expressed with an $\mathsf{LTL_f}$ formula over fluent variables.

LTL **FOND Planning.** The objective in LTL FOND planning is to synthesize *non-terminating* strategies whose executions satisfy a prescribed temporally extended goal expressed with an LTL formula over fluent variables.

3.5 Related Work

Existing tools to solve FOND planning with $\mathsf{LTL_f}$ and LTL goals were limited to the computation of strong-cyclic solutions (cf. Table 1). In the context of planning with *deterministic* actions, a variety of planners exist for temporally extended goals expressed in $\mathsf{LTL_f}$ (e.g. [Baier and McIlraith, 2006b; Baier and McIlraith, 2006a; Torres and Baier, 2015])), LTL (e.g. [Patrizi *et al.*, 2011]), and with temporal operators in the syntax of PDDL3.0, a standard language for modeling planning problems [Gerevini *et al.*, 2009]. The aforementioned approaches follow a similar schema that we also adopt.

Goal	Solution	FOND planner	Automaton (transf.)	Compilation	Size of search space wrt... domain	autom.	goal	Complete?
$\mathsf{LTL_f}$	strong / str.-cyclic	strong / str.-cyclic	AFW (LIN)	Camacho *et al.* [2017]	EXP	EXP	EXP	No
$\mathsf{LTL_f}$	strong / str.-cyclic	strong / str.-cyclic	NFW (EXP)	Camacho *et al.* [2017]	EXP	EXP	2EXP	Yes
$\mathsf{LTL_f}$	strong / str.-cyclic	strong / str.-cyclic	DFW (2EXP)	Camacho *et al.* [2017]	EXP	LIN	2EXP	Yes
LTL	str.-cyclic	str.-cyclic	ABW (LIN)	Camacho *et al.* [2017]	EXP	EXP	EXP	No
LTL	str.-cyclic	str.-cyclic	NBW (EXP)	Camacho *et al.* [2017]	EXP	EXP	2EXP	Yes
LTL	str.-cyclic	str.-cyclic	DBW (2EXP)	Camacho *et al.* [2017]	EXP	LIN	2EXP	No
LTL	str.-cyclic	str.-cyclic	DBW (2EXP)	Patrizi *et al.* [2013]	EXP	LIN	2EXP	No
LTL	strong	str.-cyclic	UCW (EXP)	This paper	EXP	EXP	2EXP	Yes

Table 2: Summary of existing compilation-based approaches to $\mathsf{LTL_f}$ FOND and LTL FOND. For each type of goal formula ($\mathsf{LTL_f}$ or LTL) and solution sought (strong or strong-cyclic), we list the type of automaton used by different compilations, the off-the-shelf planner that has to be used to obtain the desired solution, the sources of computational complexity, and whether the approach is complete. Interestingly, NFW-based (resp. NBW-based) compilations in Camacho et al. (2017) for $\mathsf{LTL_f}$ FOND (resp. LTL FOND) planning can be also used with deterministic (DFW) automata without affecting the domain, goal, and overall worst-case computational complexity of the approach.

(1) Transform the goal formula φ_G into automata

(2) Construct a new planning problem $\mathcal{P}'$ by augmenting $\mathcal{P}$ with the states and dynamics of the automata

(3) Solve $\mathcal{P}'$ with an off-the-shelf planner

(4) Extract a solution to $\mathcal{P}$ from a solution to $\mathcal{P}'$

$\mathsf{LTL_f}$ FOND and LTL FOND planning have been well studied theoretically. Strong and strong-cyclic plan existence are EXP-complete in the size of the domain, and 2EXP-complete in the size of the goal formula [De Giacomo and Rubin, 2018; Camacho *et al.*, 2019]. FOND planning with final-state goals is EXP-complete, and can be solved in polynomial time in the size of the search space [Rintanen, 2004]. Motivated by these recent results, we conducted an analysis of the sources of computational complexity of existing compilation-based approaches to $\mathsf{LTL_f}$ FOND and LTL FOND planning. Table 2 reports the size of the search space in the compiled FOND planning problems in terms of the sizes of the original domain, the automaton transformation, and the goal formula. We determined that existing complete approaches can compute solutions in exponential time in the size of the original domain, and double exponential time in the size of the goal formula – matching the complexity of the decision problems.

4 Algorithms for LTLf FOND Planning

The approach taken by Camacho *et al.* [2017] to *strong-cyclic* $\mathsf{LTL_f}$ FOND planning takes an $\mathsf{LTL_f}$ FOND planning problem $\mathcal{P} = \langle \mathcal{D}, \varphi_I, \varphi_G \rangle$ and constructs a final-state FOND problem, $\mathcal{P}'$. Strong-cyclic solutions to $\mathcal{P}$ can be extracted from strong-cyclic solutions to $\mathcal{P}'$. In essence, the compilation does a cross-product of the original domain and the automaton transformation of the formula. Figure 1 shows the details – we introduced minor changes that do not affect correctness. We convey that *strong* solutions to $\mathcal{P}$ can be also obtained from the same compilation, $\mathcal{P}'$, by simply searching for strong solutions. We elaborate on the approach below.

Step 1: Transformation of φ_G into NFW or DFW. In the first step, a CNF representation of the $\mathsf{LTL_f}$ goal formula $\varphi_G = \varphi_G^{(1)} \wedge \cdots \wedge \varphi_G^{(n)}$ is transformed into multiple NFW $\mathcal{A}^{(i)}$, one per each clause $\varphi_G^{(i)}$. The traces that satisfy φ_G are in correspondence with the traces that are accepted by all the $\mathcal{A}^{(i)}$, $i = 1..n$. Automata decompositions of φ_G are optional, and enable to scale to larger goal formulas. The compilation also works with DFW transformations of the $\mathsf{LTL_f}$ formula, as DFW automata are a particular type of NFW automata.

Step 2: Construction of a new FOND problem. In the second step, a problem $\mathcal{P}'$ is constructed that integrates the dynamics of the NFW obtained in the first step. The details of the compilation are shown in Figure 1. In a nutshell, planning states in $\mathcal{P}'$ keep track of automaton runs on the partial plan being simulated. To this end, execution of the actions in $\mathcal{P}$ are followed by actions that synchronize the automaton states.

Step 3: Solving the compiled problem. An off-the-shelf strong or strong-cyclic planner is used to compute a solution to $\mathcal{P}'$. Strong (resp. strong-cyclic) solutions to $\mathcal{P}$ can be extracted from strong (resp. strong-cyclic) solutions to $\mathcal{P}'$.

Step 4: Extraction of a strategy. A strategy σ that is solution to $\mathcal{P}$ is constructed from the solution σ' found in Step 3. Actions $\sigma(s_0 \cdots s_n)$ are obtained by unfolding σ' in $\mathcal{P}'$.

Correctness. Theorem 1 states the soundness and completeness of the approach. We omit the proof for space reasons, but it follows from the following properties: (i) State-action plans in $\mathcal{P}'$ simulate state-action plans in $\mathcal{P}$; (ii) State-action plans in $\mathcal{P}'$ do bookkeeping of (some, or all) the runs of the automata; and (iii) An state-action plan in $\mathcal{P}$ satisfies φ iff the counterpart state-action plan in $\mathcal{P}'$ that simulates it and does bookkeeping of all the automata runs achieves the goal.

Theorem 1. *An $\mathsf{LTL_f}$ FOND problem $\mathcal{P}$ has a strong (resp. strong-cyclic) solution iff the compiled FOND problem $\mathcal{P}'$ described in Figure 1 has a strong (resp. strong-cyclic) solution.*

Computational complexity. Our compilation-based ap-

Components of the compilation:

$$F' := F \cup \{\texttt{sync}, \texttt{world}\} \cup \{\texttt{oldF(q)}, \texttt{newF(q)}\}_{q \in Q} \cup \bigcup_i \{\texttt{goal}^{(i)}\}$$

$$I' := I \cup \{\texttt{sync}\} \cup \bigcup_i \{\texttt{oldF(q}_0^{(\texttt{i})}\texttt{)}\}$$

$$O' := \{\texttt{a}' \mid \texttt{a} \in O\} \cup \{\texttt{trans(t)}\}_{t \in T} \cup \{\texttt{continue}\}$$

$$G' := \bigcup_i \{\texttt{goal}^{(i)}\}$$

Stage 1: World actions:

$$Pre_{\texttt{a}'} := Pre_{\texttt{a}} \cup \{\texttt{world}\}$$

$$Eff_{\texttt{a}'} := Eff_{\texttt{a}} \cup \{\neg\texttt{world}, \texttt{sync}\}$$

Stage 2: Actions that simulate transitions $t = (q, \Phi, q') \in T$:

$$Pre_{\texttt{trans(t)}} := \{\texttt{sync}, \texttt{oldF(q)}\} \cup \Phi$$

$$Eff_{\texttt{trans(t)}} := \begin{cases} \{\texttt{newF(q')}, \texttt{goal}^{(i)}\} & \text{if } q' \in \alpha^{(i)} \\ \{\texttt{newF(q')}\} & \text{if } q' \notin \alpha^{(i)} \end{cases}$$

Action that reestablishes the dynamics of the problem:

$$Pre_{\texttt{continue}} := \{\texttt{sync}\}$$

$$Eff_{\texttt{continue}} := \{\texttt{world}, \neg\texttt{sync}\} \cup \{\neg\texttt{newF(q)}\}_{q \in Q} \cup \bigcup_i \{\neg\texttt{goal}^{(i)}\} \cup \{\texttt{newF(q)} \rightarrow \texttt{oldF(q)}\}_{q \in Q} \cup \{\neg\texttt{newF(q)} \rightarrow \neg\texttt{oldF(q)}\}_{q \in Q}$$

Figure 1: Components of the compiled FOND problem $\mathcal{P}' = \langle F', I', O', G' \rangle$ for an $\mathsf{LTL_f}$ FOND problem $\mathcal{P} = \langle F, I, O, \varphi_G \rangle$ with $\varphi_G = \varphi_G^{(1)} \wedge \cdots \wedge \varphi_G^{(n)}$ and $\mathcal{A}_i = \langle Q^{(i)}, 2^{\mathcal{F}}, q_0^{(i)}, T^{(i)}, \alpha^{(i)} \rangle$. For convenience, we write $Q := \bigcup_i Q^{(i)}$ and $T := \bigcup_i T^{(i)}$.

proach to strong and strong-cyclic $\mathsf{LTL_f}$ FOND planning can compute solutions in worst-case time that matches the complexity of the decision problem (Theorem 2). The result follows by looking at the size of the search space in $\mathcal{P}'$ with respect of the size of the search space in $\mathcal{P}$ and the size of automata transformations of the goal formula. Interestingly, the worst-case computational complexity does not depend on whether NFW or DFW automata transformations are used (Corollary 1).[2] This is because the extra exponential cost incurred by DFW automata transformations is compensated with an exponentially reduced size of the search space in the compiled problem (w.r.t. compact domain representations).

Theorem 2. *A strong (resp. strong-cyclic) solution to an* $\mathsf{LTL_f}$ *FOND planning problem* $\mathcal{P} = \langle \mathcal{D}, \varphi_I, \varphi_G \rangle$ *can be computed in worst-case exponential time in the size of the* $\mathcal{D}$*, and double-exponential time in the size of* φ_G *via the compilations to FOND planning described in Figure 1.*

Proof. The size of the search space in the compiled problem, $\mathcal{P}'$, is linear in the size of the search space of $\mathcal{P}$ and exponential in the size of NFW automata transformations of the goal formula. The desired result follows by observing that NFW automata transformations are worst-case exponential in the size of φ_G, and FOND planning can be solved in polynomial time in the size of the search space [Rintanen, 2004]. DFW automata transformations are worst-case double-exponential. To obtain the desired result, we need to observe that with *deterministic* automata the size of the search space in $\mathcal{P}'$ augments only linearly (not exponentially) with the automata sizes – because in $\mathcal{P}'$ only one automaton state fluent per DFW automaton can be made true at a time. □

[2] We thank Moshe Vardi for a question that inspired this result.

Corollary 1. *The worst-case domain and goal complexity of the compilation-based approach to strong and strong-cyclic* $\mathsf{LTL_f}$ *FOND planning does not depend on whether the* $\mathsf{LTL_f}$ *goal formula is transformed into NFW or DFW automata.*

5 Bounded LTL FOND Planning

To date, no algorithms for computing strong solutions to LTL FOND planning had been studied. Unlike with $\mathsf{LTL_f}$ FOND, we cannot simply take Camacho *et al.* [2017]'s approach to strong-cyclic LTL FOND planning and replace the strong-cyclic planner by a strong planner. The reason is complex, and is related to the type of automaton that was used – *Non-deterministic Büchi Word* (NBW) automata. In essence, all approaches to LTL synthesis rely, in one way or another, to automata determinization. The problem with using NBW is that it cannot be determinized, in general. Because LTL synthesis can be reduced to strong LTL FOND planning [Camacho *et al.*, 2019], we can expect similar determinization requirements in algorithms for strong LTL FOND planning

Noteworthy, we use a type of automaton that is particularly new to the planning community: UCW automata. In what follows, we say that the *co-Büchi index* of a word w (wrt a UCW automaton $\mathcal{A} = \langle Q, \Sigma, q_0, \delta, \alpha \rangle$) is the maximum k for which there exists a run of $\mathcal{A}$ on w that hits k states in α. Theorem 3 sets bounds on the co-Büchi index required to guarantee existence of solutions. In contrast to analogous results for *bounded* LTL *synthesis*, the bounds also depend on the domain size. In the next section, we will design algorithms for *bounded plan synthesis* in strong LTL FOND planning.

Lemma 1. *Strategy executions traces of a strong solution to an* LTL *FOND planning problem yield accepting runs of UCW transformations of the goal formula.*

Theorem 3 (Bounded plan synthesis). *Let* $\mathcal{P} = \langle \mathcal{D}, \varphi_I, \varphi_G \rangle$ *be an* LTL *FOND planning problem, and let* $\mathcal{A}_G$ *be a UCW transformation of* φ_G*. If* $\mathcal{P}$ *has a strong solution, then there exists a strategy* σ *that is a strong solution to* $\mathcal{P}$ *such that all executions traces of* σ *yield accepting runs of* $\mathcal{A}_G$ *with co-Büchi index bounded by some* $k < \infty$ *exponential in the size of* $\mathcal{A}_G$*, and exponential in the size of* $\mathcal{D}$*.*

Proof. From the reductions of LTL FOND into game structures, and further reductions into parity games, we know that a finite-state controller M that implements a strategy σ that is a strong solution to $\mathcal{P}$ can be synthesized in exponential time in the size of $\mathcal{D}$, and exponential time in the size of $\mathcal{A}_G$ (cf. [Camacho *et al.*, 2019]). The size of such controller, $|M|$, is worst-case exponential in the size of $\mathcal{D}$, and exponential in the size of $\mathcal{A}_G$. It must be the case that the runs of $\mathcal{A}_G$ on executions traces of M have finite co-Büchi index – because M implements a solution (Lemma 1). Moreover, the co-Büchi index must be bounded by $k = |M| \cdot |\mathcal{A}_G|$.

Algorithm 1 Strong solutions to LTL FOND planning

```
Input: $\mathcal{P} = \langle \mathcal{D}, \varphi_I, \varphi_G \rangle$
1: Transform $\varphi_G$ into UCW $\mathcal{A}_\varphi$. Fix $k = 1$
2: Construct $\mathcal{P}'_k = \text{XPRODUCT}(\mathcal{P}, \mathcal{A}_\varphi, k)$
3: If $\mathcal{P}'_k$ has no solution, increment $k$ by one and go to 2
4: return solution
```

Otherwise, a loopy behaviour could be generated that yields executions traces with infinite co-Büchi index. The bound k is exponential in the sizes of $\mathcal{A}_G$ and $\mathcal{D}$. $\square$

6 Algorithms for LTL FOND Planning

Our approach to compute a strong solution f to a LTL FOND planning problem $\mathcal{P}$ compiles the problem into a series of standard FOND planning problems $\mathcal{P}'_k$ with final-state goal. The compilations simulate the dynamics of $\mathcal{P}$, and keep track of the runs of a UCW automata transformations of the goal formula. The parameter k in $\mathcal{P}'_k$ bounds the maximum co-Büchi index allowed on the executions traces of f. Solutions to $\mathcal{P}$ can be directly extracted from solutions to $\mathcal{P}'_k$.

Algorithm 1 summarizes the steps of our approach, that we detailed below. We combine ideas from strong-cyclic LTL FOND planning [Camacho *et al.*, 2017] and LTL synthesis via strong-cyclic FOND planning [Camacho *et al.*, 2018b].

Step 1: Transformation of φ_G into UCW. A CNF representation of the LTL formula $\varphi_G = \varphi_G^1 \wedge \cdots \wedge \varphi_G^n$ is transformed into multiple UCW $\mathcal{A}_i$, one for each clause φ_G^i. The traces that satisfy φ_G are in correspondence with the traces that are accepted by *every* $\mathcal{A}_i$, $i = 1..n$. As usual, automata decompositions serve to scale to larger goal formulas.

Step 2: Construction of a new FOND problem. In the second step, we construct a FOND problem $\mathcal{P}'_k$ that integrates the dynamics of the UCW within the original LTL FOND problem, $\mathcal{P}$. Details are shown in Figure 2. The resulting problem is constructed in a way that enforces that strategy execution traces yield UCW runs with co-Büchi index no greater than input parameter k, starting with $k = 1$. The non-determinism induced by action *continue* is a recourse used to generate non-terminating plans. It is important to note that the dynamics simulate *all* the runs of the UCW, and fluents $\texttt{newCnt}(\texttt{q}, \texttt{m})$ do bookkeeping of the maximum co-Büchi index among all the runs that finish in $q \in Q$. The dynamics of the FOND domain enforces that those runs never hit α more than k times. Certainly, a deadend is induced whenever one run hits α exactly $k + 1$ times. This is because actions $\texttt{syncF}(\texttt{q}, \texttt{m})$ can only synchronize automaton states with associated co-Büchi index not greater than k. Hence, automaton state fluents $\texttt{newF}(\texttt{q})$ with associated co-Büchi index $\texttt{oldCnt}(\texttt{q}, \texttt{k} + \texttt{1})$ cannot be made false, preventing the action $\texttt{continue}$ from being applied.

Step 3: Solving the compiled problem. An off-the-shelf planner is used to compute a *strong-cyclic* solution to the compiled FOND problem. If no solution exists, parameter k is incremented by one and the algorithm goes to Step 2.

Step 4: Extraction of a strategy. Similar to the strategy extraction procedure for $\mathsf{LTL_f}$ FOND, a strong solution to $\mathcal{P}$ can be obtained by unfolding a strong-cyclic solution to $\mathcal{P}'$.

Components of the compilation:
$F' := F \cup \{\texttt{oldF}(\texttt{q}), \texttt{newF}(\texttt{q})\}_{q \in Q}$ $\cup \{\texttt{sync}, \texttt{aut}, \texttt{world}, \texttt{goal}\}$ $\cup \{\texttt{oldCnt}(\texttt{q},\texttt{m}), \texttt{newCnt}(\texttt{q},\texttt{m})\}_{q \in Q, 0 \le m \le k+1}$
$I' := I \cup \{\texttt{aut}\} \cup \bigcup_i \{\texttt{oldF}(\texttt{q}_0^{(\texttt{i})}), \texttt{oldCnt}(\texttt{q}_0^{(\texttt{i})}, \texttt{0})\}$
$O' := \{\texttt{a}' \mid \texttt{a} \in O\} \cup \{\texttt{startSync}, \texttt{continue}\}$ $\cup \{\texttt{trans}(\texttt{q},\texttt{m}), \texttt{syncF}(\texttt{q},\texttt{m})\}_{q \in Q, 0 \le m \le k}$
$G' := \{\texttt{goal}\}$
World actions:
$Pre_{\texttt{a}'} := Pre_{\texttt{a}} \cup \{\texttt{world}\}$
$Eff_{\texttt{a}'} := Eff_{\texttt{a}} \cup \{\neg\texttt{world}, \texttt{aut}\}$
Actions that simulate UkCW transitions:
$Pre_{\texttt{trans(q,m)}} := \{\texttt{aut}, \texttt{oldF}(\texttt{q}), \texttt{oldCnt}(\texttt{q},\texttt{m})\}$
$Eff_{\texttt{trans(q,m)}} := \{\neg\texttt{oldF}(\texttt{q})\}$ $\cup\{\Phi \rightarrow \{\texttt{newF}(\texttt{q}'), \texttt{newCnt}(\texttt{q}', \texttt{m} + \mathbb{1}(\texttt{q}' \in \alpha))\}\}_{(q,\Phi,q') \in T}$
Actions that synchronize UkCW states:
$Pre_{\texttt{startSync}} := \{\texttt{aut}\} \cup \{\neg\texttt{oldF}(\texttt{q})\}_{q \in Q}$
$Eff_{\texttt{startSync}} := \{\texttt{sync}, \neg\texttt{aut}\}$ $\cup \{\neg\texttt{oldCnt}(\texttt{q},\texttt{m})\}_{q \in Q, 0 \le m \le k}$
$Pre_{\texttt{syncF(q,m)}} := \{\texttt{sync}, \texttt{newF}(\texttt{q}), \texttt{newCnt}(\texttt{q},\texttt{m})\}$ $\cup \{\neg\texttt{newCnt}(\texttt{q},\texttt{n})\}_{m < n \le k}$
$Eff_{\texttt{syncF(q,m)}} := \{\texttt{oldF}(\texttt{q}), \neg\texttt{newF}(\texttt{q}), \texttt{oldCnt}(\texttt{q},\texttt{m})\}$ $\cup \{\neg\texttt{newCnt}(\texttt{q},\texttt{n})\}_{0 \le n \le k}$
Action that reestablishes the dynamics of the problem:
$Pre_{\texttt{continue}} := \{\texttt{sync}\} \cup \{\neg\texttt{newF}(\texttt{q})\}_{q \in Q}$
$Eff_{\texttt{continue}} := \textit{oneof}(\{\texttt{goal}\}, \{\texttt{world}, \neg\texttt{sync}\})$

Figure 2: Components of the compiled FOND problem $\mathcal{P}'_k = \langle F', I', O', G' \rangle$ for an LTL FOND problem $\mathcal{P} = \langle F, I, O, \varphi_G \rangle$ with $\varphi_G = \varphi_G^{(1)} \wedge \cdots \wedge \varphi_G^{(n)}$ and $\mathcal{A}_i = \langle Q^{(i)}, 2^{\mathcal{F}}, q_0^{(i)}, T^{(i)}, \alpha^{(i)} \rangle$. For convenience, we write $Q := \bigcup_i Q^{(i)}$ and $T := \bigcup_i T^{(i)}$.

Correctness. Theorem 4 establishes the correctness of our compilation-based approach to computing strong solutions to LTL FOND planning. Note that a *strong-cyclic* planner is needed to obtain *strong* solutions.

Theorem 4. *An* LTL *FOND problem* $\mathcal{P} = \langle \mathcal{D}, \varphi_I, \varphi_G \rangle$ *has a strong solution iff the compiled FOND problem* $\mathcal{P}'_k$ *described in Figure 2 has a strong-cyclic solution for some* $k < \infty$ *double exponential in the size of* φ_G*, and exponential in the size of* $\mathcal{D}$*.*

Proof sketch. We will exploit the following property: a strategy f is a strong-cyclic solution to a goal-oriented FOND planning problem iff the goal condition can be reached from any state that is reachable by f from the initial state.

Suppose $\mathcal{P}$ has a strong solution. By Theorem 3, there exists a strong solution f and $k < \infty$ worst-case double exponential in the size of φ_G and $\mathcal{D}$ so that execution traces of f generate automata runs with co-Büchi index not greater than k. Let f' be a strategy for $\mathcal{P}'_k$ that *simulates* f, i.e., that outputs the same actions as f in response to the plans being simulated. The goal of $\mathcal{P}'_k$ is reachable from all states that are reachable by f' from the initial state. I.e., f' is a strong-cyclic solution to $\mathcal{P}'_k$ (in the form of a strategy). Finally, if a FOND planning problem with final-state goal has a solution in the form of a strategy, then it also has one in the form of a policy.

In the other direction, we will see that a strong-cyclic solution π to $\mathcal{P}'_k$ yields a strong solution f to $\mathcal{P}$. Certainly, consider f to be the strategy that results from unfolding π on the simulated plans, and choosing the non-deterministic effect of action `continue` that does not lead to the goal. It is easy to see that f yields infinite-length plans with co-Büchi index not greater than k, and therefore is a strong solution to $\mathcal{P}$. The bounds follow from Theorem 3. □

Computational complexity. Our compilations can be used to compute strong solutions to an LTL FOND problem in worst-case time that matches the complexity of the decision problem (Theorem 5).

Theorem 5. *A strong solution to an* LTL *FOND problem* $\mathcal{P} = \langle \mathcal{D}, \varphi_I, \varphi_G \rangle$ *can be computed in worst-case exponential time in the size of the size of* $\mathcal{D}$*, and double exponential time in the size of* φ_G *via the compilations to strong-cyclic FOND planning described in Figure 2 and Algorithm 1.*

Proof. The goal formula φ_G can be transformed into an UCW automaton $\mathcal{A}_G$ in exponential time, with a number of states $|Q|$ that is exponential in the size of φ_G and $\mathcal{D}$ (cf. Property 1). A strong solution to $\mathcal{P}$ can be extracted from a strong-cyclic solution to $\mathcal{P}'_k$ in linear time. The size of the search space in $\mathcal{P}'_k$ is exponential in the size of $\mathcal{D}$, and polynomial in $k^{|Q|}$ (cf. Figure 2). The desired result follows by exploiting the bounds on the co-Büchi index stated in Theorem 3, and observing that FOND planning can be solved in polynomial time in the size of the search space, and the bounded number of iterations in Algorithm 1. □

6.1 Proofs of Unsolvability

Whereas planning algorithms have been traditionally focused on the search for solutions, detecting when a planning instance is unsolvable is a topic of growing interest (e.g. [Eriksson *et al.*, 2017]). The absence of strong solutions to an LTL FOND problem can be assessed when the compiled FOND problem $\mathcal{P}_k$ obtained with k exponential in Q has no strong-cyclic solution (cf. Theorem 4). Unfortunately, the bounds on k are prohibitively large to assess unsolvability in practice.

We present an alternative method to obtain a proof that an LTL FOND problem has no strong solution. We follow on recent work by [Camacho *et al.*, 2018a] to check unrealizability of specifications for LTL synthesis via reduction to reachability games, and adapt their techniques to LTL FOND planning LTL FOND can be interpreted as a two-player game between the agent – that decides on actions – and the environment – that decides on the outcome of the actions [Camacho *et al.*, 2019]. The agent has the objective to satisfy φ_G. Observe that $\mathcal{P}$ is unsolvable iff the environment has a strategy to defeat the agent, that is, a strategy to satisfy $\neg\varphi_G$. With this in mind, we reason on the dualization of the problem. We transform $\neg\varphi_G$ into UCW automata, and compute an agent strategy that forces a run with co-Büchi index greater than k, for some hyperparameter k. If no such strategy exists, we can conclude that the game is winning for the environment. In other words, $\mathcal{P}$ has no strong solution. Note, we do *not* obtain a environment counter-strategy, but a proof that one exists. The details of our approach follow below.

Algorithm 2 Proof unsolvability in LTL FOND planning

```
Input: $\mathcal{P} = \langle \mathcal{D}, \varphi_I, \varphi_G \rangle$
1: Transform $\neg\varphi_G$ into UCW $\mathcal{A}_\varphi$. Fix $k = 1$
2: Construct $\overline{\mathcal{P}_k} = \text{XPRODUCTUNSOL}(\mathcal{P}, \mathcal{A}_\varphi, k)$
3: If $\overline{\mathcal{P}_k}$ has strong solution, increment $k$ by one and go to 2
4: return unsolvable
```

Step 1: Transformation of $\neg\varphi_G$ into UCW. The negation of the goal formula is transformed into one or multiple UCW.

Step 2: Construction of a new FOND problem. The construction of a problem $\overline{\mathcal{P}_k} = \text{XPRODUCTUNSOL}(\mathcal{P}, \mathcal{A}_\varphi, k)$ follows that of Figure 2, with three exceptions: (i) the UCW automata used in the compilation are transformations of $\neg\varphi_G$; (ii) the goal of $\overline{\mathcal{P}_k}$ is achieved when one of the fluents $\texttt{newCnt(q,m)}$ with $m = k$ is achieved; and (iii) the action to restablish the world dynamics of the problem is replaced by:

$$Pre_{\texttt{continue}} := \{\texttt{sync}\} \cup \{\neg\texttt{newF(q)}\}_{q \in Q}$$

$$\mathit{Eff}_{\texttt{continue}} := \{\texttt{world}, \neg\texttt{sync}\}$$

Step 3: Solving the compiled problem. A FOND planner is used to search for *strong* solutions to $\overline{\mathcal{P}_k}$. If no solution exists, the LTL FOND problem is deemed unsolvable. Otherwise, k is incremented by one and Step 2 is repeated.

Correctness. Theorem 6 states the correctness of our compilation-based approach to determining the existence of strong solutions to LTL FOND planning. Note that a *strong* planner is needed to search for solutions to $\mathcal{P} = \langle \mathcal{D}, \varphi_I, \varphi_G \rangle$. The proof follows the mechanics of the proofs of Theorems 3 and 4, and we omit it here for space reasons.

Theorem 6. *An* LTL *FOND problem* $\mathcal{P} = \langle \mathcal{D}, \varphi_I, \varphi_G \rangle$ *has no strong solution iff for some* $k < \infty$ *double exponential in the size of* φ_G*, and exponential in the size of* $\mathcal{D}$*, the compiled FOND problem* $\overline{\mathcal{P}_k}$ *has no strong solution*

Computational complexity. Our compilation-based approach to determine the existence of solutions to LTL FOND problem described in Algorithm 2 runs in worst-case time that matches the complexity of the decision problem (Theorem 7).

Theorem 7. *Non-existence of strong solutions to an* LTL *FOND problem* $\mathcal{P} = \langle \mathcal{D}, \varphi_I, \varphi_G \rangle$ *can be proved in worst-case exponential time in the size of the size of* $\mathcal{D}$*, and double exponential time in the size of* φ_G *via the compilations to strong FOND planning* $\overline{\mathcal{P}_k}$ *and Algorithm 2.*

Proof sketch. Analogous to the proof of Theorem 5. □

7 Experiments

To the best of our knowledge, ours are the first algorithmic approaches to compute strong solutions to LTL_f FOND and LTL FOND planning. Our compilations are implemented in Python, and take (and produce) PDDL files. We used *Spot* [Duret-Lutz *et al.*, 2016] to transform LTL to UCW, and Baier and McIlraith [2006b]'s code to transform LTL_f to NFW.

We conducted a series of experiments to evaluate the efficiency of our methods in the *Clerk, Lift*, and *Waldo* domains from [Patrizi *et al.*, 2013], detailed below. Goal formulas were interpreted as $\mathsf{LTL_f}$ (resp. LTL) for $\mathsf{LTL_f}$ FOND (resp. LTL FOND) planning. Strong policies to the compiled problems were computed with myND planner [Mattmüller *et al.*, 2010], and PRP planner was used to compute strong-cyclic policies [Muise *et al.*, 2012]. Because myND does not support conditional effects, we replaced action *continue* in Figure 1 with a cascade of actions without conditional effects. Our experiments ran in Ubuntu machines with an Intel(R) Xeon(R) 2.30GHz CPU. Runtime was limited to 30 minutes, and memory usage never exceeded 1GB.

Clerk. The agent has to process orders of different types of packages (n in problem p_n). Each package can be served or stored in a designated location, with n locations in total. The agent can request orders, in which one of the n products is non-deterministically requested to be served, or put on hold for restock. The objective of the agent is to continuously complete orders. which we encoded with the following formula:

$$\Box\Diamond(\texttt{active\_request})\wedge$$
$$\Box((\texttt{active\_request}) \to \Diamond((\texttt{pkg\_served}) \vee (\texttt{pkg\_stored}))$$

Lift. The agent controls a lift in a building with a number of floors (n floors in problem p_n). Each timestep, the elevator receives one or more requests from some of the floors (this is modeled in PDDL with a cascade of sequential actions). The objective is to compute a controller for the elevator that serves each request. This is modeled with the formula:

$$\Box\Diamond(\neg(\texttt{at f\_1}) \to (\texttt{called})) \wedge \bigwedge_{i=1}^{n} \Box\Diamond((\texttt{req f\_i}) \to (\texttt{at f\_i}))$$

Waldo. The agent can inspect rooms, numbered in consecutive order, in a circular corridor. Problem p_{2n} has $2n$ rooms. The n-th and $2n$-th rooms have a special property: Waldo may non-deterministically appear when visiting one of those. The dynamics of the problem forces the agent to visit the $2n$-th room before revisting the n-th room (and vice versa). Each time both rooms are visited, a special fluent search_again is made true for only one timestep and a new cycle starts. The objective is to find a strategy that searches for Waldo, i.e.,

$$\Box\Diamond((\texttt{search\_again}) \vee (\texttt{seen}))$$

In the search for strong solutions to $\mathsf{LTL_f}$ FOND, strong planner myND detected unsolvability of the *Clark* problems relatively fast. The goals in the *Lift* problems resulted trivial. The *Waldo* problems show a more interesting scalability.

Strong solutions to LTL FOND in the *Lift* and *Waldo* domains are, conceptually, not very different than strong solutions to $\mathsf{LTL_f}$ FOND, except that the agent has to cycle. Noteworthy, we observed that the performance of the planning step in LTL FOND suffered significantly (at least with PRP) in comparison with the planning step in $\mathsf{LTL_f}$ FOND. Also in comparison, the search for *strong-cyclic* solutions to those LTL FOND problems (with Camacho *et al.* [2017]'s compilation) showed much better scalability. We conjecture three aspects that may challenge scalability in our approach to strong LTL FOND: (i) conditional effects affect performance of planners like PRP (ii) FOND planners based on plan aggregation, like PRP, may suffer from myopic behaviour because the compiled problem $\mathcal{P}'$ non-deterministically yields a goal state at each simulated timestep. (iii) it is challenging for the planner to generalize policies accross macrostates that vary on the co-Büchi idexes asociated to automaton state fluents.

	LTL FOND			$\mathsf{LTL_f}$ FOND	
problem	co-Büchi index	policy size	run time	policy size	run time
clerk-p1	6	164	0.98	N/A	0.46
clerk-p2	8	339	1.66	N/A	1.26
clerk-p3	10	1753	18.4	N/A	3.62
clerk-p4	12	1963	281	N/A	12.5
clerk-p5	–	–	timeout	N/A	67.4
lift-p1	1	38	0.06	1	0.03
lift-p2	7	290	1.02	1	0.02
waldo-p10	8	250	8.18	50	0.19
waldo-p20	13	477	84.1	85	0.38
waldo-p30	18	687	448	120	0.61
waldo-p40	23	–	timeout	155	0.65

Table 3: Summary of our experiments to compute strong solutions LTL FOND and $\mathsf{LTL_f}$ FOND planning via compilations to FOND planning. N/A indicates unsolvable problem. Timouts after 30 min.

The first two items identified above could be mitigated by new FOND planners. The third item is more delicate. If our conjecture (iii) is true, then bounded LTL FOND planning would suffer with final-state goals (e.g. $\Diamond\varphi$, for some propositional formula φ) and other temporally extended goals that require a large makespan, which translates into a large co-Büchi index. Our experiments suggest the need for more effective methods in LTL FOND that handle large co-Büchi indexes without severely affect performance.

8 Discussion and Future Work

Synthesizing programs for sequential decision-making in dynamical environments is a central problem in artificial intelligence. The problem can be cast as FOND planning with temporally extended goals expressed in LTL and its variant over finite traces, $\mathsf{LTL_f}$. While the problem is well understood theoretically, the number of existing tools is limited. We provided the first algorithmic approach to computing strong solutions to $\mathsf{LTL_f}$ FOND and LTL FOND planning. Our techniques complement previous techniques for computing strong-cyclic solutions, collectively providing a complete battery of tools for strong and strong-cyclic FOND planning. We conducted an analysis of the sources of computational complexity, and determined that our techniques for plan synthesis are computationally effective in the sense that they match the domain and goal complexity of the plan existence decision problems. We evaluated the performance of our approaches and identified potential sources of complexity that make it challenging to compute strong solutions to LTL FOND, relative to computing strong-cyclic ones.

Acknowledgements

We gratefully acknowledge funding from the Natural Sciences and Engineering Research Council of Canada.

References

[Bacchus and Kabanza, 1998] Fahiem Bacchus and Froduald Kabanza. Planning for temporally extended goals. *Annals of Mathematics and Artificial Intelligence*, 22(1-2):5–27, 1998.

[Baier and McIlraith, 2006a] Jorge A. Baier and Sheila A. McIlraith. Planning with first-order temporally extended goals using heuristic search. In *Proceedings of the 21st National Conference on Artificial Intelligence (AAAI)*, pages 788–795, 2006.

[Baier and McIlraith, 2006b] Jorge A. Baier and Sheila A. McIlraith. Planning with temporally extended goals using heuristic search. In *Proceedings of the 16th International Conference on Automated Planning and Scheduling (ICAPS)*, pages 342–345, 2006.

[Bienvenu *et al.*, 2011] Meghyn Bienvenu, Christian Fritz, and Sheila A. McIlraith. Specifying and computing preferred plans. *Artificial Intelligence*, 175(7–8):1308–1345, 2011.

[Camacho *et al.*, 2017] Alberto Camacho, Eleni Triantafillou, Christian Muise, Jorge A. Baier, and Sheila A. McIlraith. Non-deterministic planning with temporally extended goals: LTL over finite and infinite traces. In *Proceedings of the 31st AAAI Conference on Artificial Intelligence (AAAI)*, pages 3716–3724, 2017.

[Camacho *et al.*, 2018a] Alberto Camacho, Jorge A. Baier, Christian J. Muise, and Sheila A. McIlraith. Finite LTL synthesis as planning. In *Proceedings of the 28th International Conference on Automated Planning and Scheduling (ICAPS)*, pages 29–38, 2018.

[Camacho *et al.*, 2018b] Alberto Camacho, Christian J. Muise, Jorge A. Baier, and Sheila A. McIlraith. LTL realizability via safety and reachability games. In *Proceedings of the 27th International Joint Conference on Artificial Intelligence (IJCAI)*, pages 4683–4691, 2018.

[Camacho *et al.*, 2019] Alberto Camacho, Meghyn Bienvenu, and Sheila A. McIlraith. Towards a unified view of ai planning and reactive synthesis. In *Proceedings of the 29th International Conference on Automated Planning and Scheduling (ICAPS)*, 2019.

[Cimatti *et al.*, 2003] Alessandro Cimatti, Marco Pistore, Marco Roveri, and Paolo Traverso. Weak, strong, and strong cyclic planning via symbolic model checking. *Artificial Intelligence*, 147(1-2):35–84, 2003.

[De Giacomo and Rubin, 2018] Giuseppe De Giacomo and Sasha Rubin. Automata-theoretic foundations of FOND planning for LTLf and LDLf goals. In *Proceedings of the 27th International Joint Conference on Artificial Intelligence (IJCAI)*, pages 4729–4735, 2018.

[De Giacomo and Vardi, 2013] Giuseppe De Giacomo and Moshe Y. Vardi. Linear temporal logic and linear dynamic logic on finite traces. In *Proceedings of the 23rd International Joint Conference on Artificial Intelligence (IJCAI)*, pages 854–860, 2013.

[Duret-Lutz *et al.*, 2016] Alexandre Duret-Lutz, Alexandre Lewkowicz, Amaury Fauchille, Thibaud Michaud, Etienne Renault, and Laurent Xu. Spot 2.0 — a framework for LTL and ω-automata manipulation. In *Proceedings of the 14th International Symposium on Automated Technology for Verification and Analysis (ATVA)*, volume 9938 of *Lecture Notes in Computer Science*, pages 122–129. Springer, October 2016.

[Eriksson *et al.*, 2017] Salomé Eriksson, Gabriele Röger, and Malte Helmert. Unsolvability certificates for classical planning. In *Proceedings of the 27th International Conference on Automated Planning and Scheduling (ICAPS)*, pages 88–97, 2017.

[Geffner and Bonet, 2013] Hector Geffner and Blai Bonet. *A Concise Introduction to Models and Methods for Automated Planning*. Synthesis Lectures on Artificial Intelligence and Machine Learning. 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):619–668, 2009.

[Ghallab *et al.*, 2016] Malik Ghallab, Dana S. Nau, and Paolo Traverso. *Automated Planning and Acting*. Cambridge University Press, 2016.

[Kupferman and Vardi, 2005] Orna Kupferman and Moshe Y. Vardi. Safraless decision procedures. In *Proceedings of the 46th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 531–542, 2005.

[Mattmüller *et al.*, 2010] Robert Mattmüller, Manuela Ortlieb, Malte Helmert, and Pascal Bercher. Pattern database heuristics for fully observable nondeterministic planning. In *Proceedings of the 20th International Conference on Automated Planning and Scheduling (ICAPS)*, pages 105–112, 2010.

[Muise *et al.*, 2012] Christian Muise, Sheila A. McIlraith, and J. Christopher Beck. Improved Non-deterministic Planning by Exploiting State Relevance. In *Proceedings of the 22nd International Conference on Automated Planning and Scheduling (ICAPS)*, pages 172–180, 2012.

[Patrizi *et al.*, 2011] Fabio Patrizi, Nir Lipovetzky, Giuseppe De Giacomo, and Hector Geffner. Computing infinite plans for LTL goals using a classical planner. In *Proceedings of the 22nd International Joint Conference on Artificial Intelligence (IJCAI)*, pages 2003–2008, 2011.

[Patrizi *et al.*, 2013] Fabio Patrizi, Nir Lipovetzky, and Hector Geffner. Fair LTL synthesis for non-deterministic systems using strong cyclic planners. In *Proceedings of the 23rd International Joint Conference on Artificial Intelligence (IJCAI)*, pages 2343–2349, 2013.

[Pistore *et al.*, 2001] Marco Pistore, Renato Bettin, and Paolo Traverso. Symbolic techniques for planning with extended goals in non-deterministic domains. In *Proceedings of the 6th European Conference on Planning (ECP)*, pages 166–173, 2001.

[Pnueli and Rosner, 1989] Amir Pnueli and Roni Rosner. On the synthesis of a reactive module. In *Proceedings of the 16th Annual ACM Symposium on Principles of Programming Languages (POPL)*, pages 179–190, 1989.

[Pnueli, 1977] Amir Pnueli. The temporal logic of programs. In *Proceedings of the 18th IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 46–57, 1977.

[Rintanen, 2004] Jussi Rintanen. Complexity of planning with partial observability. In *Proceedings of the 14th International Conference on Automated Planning and Scheduling (ICAPS)*, pages 345–354, 2004.

[Torres and Baier, 2015] Jorge Torres and Jorge A. Baier. Polynomial-time reformulations of LTL temporally extended goals into final-state goals. In *Proceedings of the 24th International Joint Conference on Artificial Intelligence (IJCAI)*, pages 1696–1703, 2015.