LTL$_f$ Adaptive Synthesis for Multi-Tier Goals in Nondeterministic Domains

Giuseppe De Giacomo[1,2], **Gianmarco Parretti**[2], **Shufang Zhu**[3]

[1]University of Oxford
[2]University of Rome "La Sapienza"
[3]University of Liverpool
giuseppe.degiacomo@cs.ox.ac.uk, parretti@diag,uniroma1.it, shufang.zhu@liverpool.ac.uk

Abstract

We study a variant of LTL$_f$ synthesis that synthesizes adaptive strategies for achieving a multi-tier goal, consisting of multiple increasingly challenging LTL$_f$ objectives in nondeterministic planning domains. Adaptive strategies are strategies that at any point of their execution (*i*) enforce the satisfaction of as many objectives as possible in the multi-tier goal, and (*ii*) exploit possible cooperation from the environment to satisfy as many as possible of the remaining ones. This happens dynamically: if the environment cooperates (*ii*) and an objective becomes enforceable (*i*), then our strategies will enforce it. We provide a game-theoretic technique to compute adaptive strategies that is sound and complete. Notably, our technique is polynomial, in fact quadratic, in the number of objectives. In other words, it handles multi-goals with only a minor overhead compared to standard LTL$_f$ synthesis.

1 Introduction

There has been a growing interest in Reasoning about Actions, Planning, and Sequential Decision Making, to techniques from formal methods for strategic reasoning such as reactive synthesis (Pnueli and Rosner 1989; Fijalkow et al. 2024), which can play a crucial role in developing autonomous AI systems capable of self-programming their actions to complete desired goals. Such capabilities are particularly important when these systems operate within complex, dynamic environments, in particular, with high levels of nondeterminism.

Most of the literature on strategy synthesis (Fijalkow et al. 2024; De Giacomo and Vardi 2015; De Giacomo and Rubin 2018; Camacho, Bienvenu, and McIlraith 2019) and planning (Ghallab, Nau, and Traverso 2016; Haslum et al. 2019) assumes that the AI system or the *agent*, is working towards a singular goal (typically specified, e.g., in standard reachability (Haslum et al. 2019), or in Temporal Logics (Aminof et al. 2019; Camacho, Bienvenu, and McIlraith 2019)). The focus of planners/synthesizers is generally on finding a plan/strategy as quickly as possible (Haslum et al. 2019; Geffner and Bonet 2013; Fijalkow et al. 2024). However, in real-world applications, users often struggle to specify an agent goal accurately using only high-level characteristics. Preferences-based planning (PBP) (Son and Pontelli

2006; Bienvenu, Fritz, and McIlraith 2006; Jorge, McIlraith et al. 2008) addresses this challenge by allowing users to specify preferences over the plans. The aim in PBP is to produce plans that satisfy as many user-specified preferences as possible, such as goal preferences, action preferences, state preferences and temporal preferences (Baier, Bacchus, and McIlraith 2009). Planning with soft goals (Keyder and Geffner 2009) is a simple model of PBP, which aims to maximize utility in cases where the agent may not be able to achieve all its goals. PBP frameworks optimize the selection of objectives, deciding which to achieve and which to discard, with the discarded ones being those that do not contribute to optimal preference satisfaction.

In this paper we focus on Fully Observable Nondeterministic Domains (FOND) (Geffner and Bonet 2013) and consider temporal preferences that are ordinal (Son and Pontelli 2006; Bienvenu, Fritz, and McIlraith 2006). Specifically, the agent goals are multi-tier goals consisting of a multi-tier hierarchy of increasingly challenging objectives expressed in Linear Temporal Logic on finite traces (LTL$_f$) (De Giacomo and Vardi 2013). This enables users to specify when a strategy is better than another by simply considering additional tasks, e.g., a strategy that enforces "clean room A and deliver package B, $\varphi_2 = \Diamond(clean_A) \land \Diamond(deliver_B)$" is better than another strategy that only enforces "clean room A $\varphi_1 = \Diamond(clean_A)$". If we simply aim at satisfying as many objectives as possible, considering that they are in a hierarchy of increasingly challenging objectives, the problem reduces to finding the most challenging objective for which a winning strategy can be synthesized. This can be done with off-the-shelf techniques (De Giacomo and Rubin 2018).

However, as observed in (Shaparau, Pistore, and Traverso 2006) in the context of FOND for reachability goals, in nondeterministic domains it is interesting to make the optimality criteria depend on the state of the domain and the response of the environment. Importantly, all objectives must remain under consideration at all instants, as objectives that are currently unachievable could become feasible later due to nondeterministic responses. For example while currently only $\varphi_1 = \Diamond(clean_A)$ is achievable for any environment response, a specific environment response might make $\varphi_2 = \Diamond(clean_A) \land \Diamond(deliver_B)$ achievable as well.

Embracing this intuition, we seek *adaptive strate-*

gies[1] (plans) that at any point during execution (*i*) enforce the satisfaction of as many LTL$_f$ objectives as possible, and (*ii*) exploit possible environment cooperation to satisfy as many remaining ones as possible. This happens dynamically: if the environment cooperates (*ii*) and an objective becomes enforceable (*i*), then our strategies will enforce it.

Formally we base our approach on *best-effort strategies* (Aminof et al. 2020, 2021; Aminof, De Giacomo, and Rubin 2021). Best-effort strategies are strategies that enforce objective satisfaction whenever possible, and do nothing that needlessly prevent satisfying the objective, otherwise. Intuitively, if an objective that is not enforceable now but could still become enforceable later during execution due to possible cooperative environment response, best-effort strategies will exploit such response to enforce objective satisfaction. In this sense we can think best-effort strategies as adaptive strategies for a single objective. Note, if winning strategies exist, then best-effort strategies are exactly the winning strategies. However while winning strategies may not exist, best effort strategies always do. Best-effort strategies for LTL$_f$ objectives can be computed in worst-case 2EXPTIME, just as for winning strategies (best-effort synthesis is 2EXPTIME-complete, just as in standard synthesis) (Aminof, De Giacomo, and Rubin 2021). Note that such worst-case blowup depends on the need to build a DFA for the LTL$_f$ formula, and occurs only rarely in practice, see, e.g. (De Giacomo and Vardi 2015; Camacho et al. 2019; Zhu et al. 2020; Geatti, Montali, and Rivkin 2024).

Leveraging results on best-effort strategies, we study synthesis of adaptive strategies for multi-tier goals. Specifically we want to synthesize a strategy that in every history fulfills a maximal number of objectives. Hence, an adaptive strategy for a multi-tier goal keeps all objectives active by winning as many objectives as possible, regardless of environment responses, and not ruling out completion for as many as possible of the remaining ones. In particular, we prove that such strategies (as best-effort strategies for single objectives) always exist (see Theorem 1).

Our main contribution is a game-theoretic technique to compute an adaptive strategy for multi-tier goals that is sound and complete, as well as practically efficient. Our technique first identifies the most challenging objective that the agent can win in spite of the environment, which is referred to as *maximally winning objective*. This objective represents the highest tier, i.e., most challenging, that the agent can enforce against all possible environment behaviors, hence also enforcing lower-tier, i.e., easier, objectives in the multi-tier goal hierarchy. Then, we identify the *maximally winning-pending objective*, which is higher than (or equal to) the maximally winning one. In this way we can win for the maximally winning objective and leave open the possibility to further win as many as possible of the higher-tier objectives if the environment cooperates. To compute the maximally winning-pending objective, our approach re-

quires consideration of only the maximally winning objective and each of those more challenging objectives. As a result, our technique remains worst-case 2EXPTIME-complete in the size of the objectives in the LTL$_f$ multi-tier goal as standard LTL$_f$ synthesis, but notably it is *polynomial*, in fact quadratic, in the number of objectives (Theorem 5). In this way, we are able to handle multi-tier goals with only a minor overhead compared to standard LTL$_f$ synthesis.

2 Preliminaries

A *trace* over an alphabet of symbols Σ is a finite or infinite sequence of elements from Σ. The empty trace is λ. Traces are indexed starting at zero, and we write $\pi = \pi_0\pi_1\cdots$. The length of a trace is $|\pi|$. For a finite trace π, we denote by $\texttt{lst}(\pi)$ the index of the last element of π, i.e., $|\pi| - 1$.

LTL$_f$. Linear Temporal Logic on finite traces (LTL$_f$) is a specification language to express temporal properties for finite and non-empty traces (De Giacomo and Vardi 2013). LTL$_f$ shares the syntax with LTL (Pnueli 1977), but is interpreted over finite non-empty traces. Given a set of atoms AP, the LTL$_f$ formulas over AP are generated as follows:
$$\varphi ::= p \mid \varphi_1 \wedge \varphi_2 \mid \neg\varphi \mid \bigcirc\varphi \mid \varphi_1 \, \mathcal{U} \, \varphi_2.$$
$p \in AP$ is an *atom*, $\bigcirc$ (*Next*), and $\mathcal{U}$ (*Until*) are temporal operators. We use standard Boolean abbreviations such as $\vee$ (or) and $\supset$ (implies), *true* and *false*. Moreover, we define the following abbreviations *Weak Next* $\bullet\varphi \equiv \neg\bigcirc\neg\varphi$, *Eventually* $\Diamond\varphi \equiv true \, \mathcal{U} \, \varphi$ and *Always* $\Box\varphi \equiv \neg\Diamond\neg\varphi$. The size of φ, written $|\varphi|$, is the number of all its subformulas.

LTL$_f$ formulas are interpreted over finite non-empty traces π over the alphabet $\Sigma = 2^{AP}$, i.e., the alphabet consisting of the propositional interpretations of the atoms. Thus, for $i \leq \texttt{lst}(\pi)$, we have that $\pi_i \in 2^{AP}$ is the i-th interpretation of π. An LTL$_f$ formula φ *holds* at instant i of a trace π is defined inductively on the structure of φ as:

- $\pi, i \models p$ iff $p \in \pi_i$;
- $\pi, i \models \neg\varphi$ iff $\pi, i \not\models \varphi$;
- $\pi, i \models \varphi_1 \wedge \varphi_2$ iff $\pi, i \models \varphi_1$ and $\pi, i \models \varphi_2$;
- $\pi, i \models \bigcirc\varphi$ iff $i < \texttt{lst}(\pi)$ and $\pi, i+1 \models \varphi$;
- $\pi, i \models \varphi_1 \, \mathcal{U} \, \varphi_2$ iff $\exists j$ such that $i \leq j \leq \texttt{lst}(\pi)$ and $\pi, j \models \varphi_2$, and $\forall k, i \leq k < j$ we have that $\pi, k \models \varphi_1$.

We say that π *satisfies* φ, written as $\pi \models \varphi$, if $\pi, 0 \models \varphi$.

Nondeterministic Planning Domains. Following (De Giacomo, Parretti, and Zhu 2023b), a *nondeterministic planning domain* is a tuple $\mathcal{P} = (2^{\mathcal{F}}, s_0, Act, React, \alpha, \beta, \delta)$, where: $\mathcal{F}$ is a finite set of fluents, $|\mathcal{F}|$ is the size of $\mathcal{P}$, and $2^{\mathcal{F}}$ is the state space; $s_0 \in 2^{\mathcal{F}}$ is the initial state; Act is a finite set of agent actions; $React$ is a finite set of environment reactions; $\alpha : 2^{\mathcal{F}} \to 2^{Act}$ denotes agent action preconditions; $\beta : 2^{\mathcal{F}} \times Act \to 2^{React}$ denotes environment reaction preconditions; and $\delta : 2^{\mathcal{F}} \times Act \times React \mapsto 2^{\mathcal{F}}$ is the transition function such that $\delta(s, a, r)$ is defined iff $a \in \alpha(s)$ and $r \in \beta(s, a)$. As in (De Giacomo, Parretti, and Zhu 2023b), we require nondeterministic planning domains to satisfy: • *Existence of agent action*: $\forall s \in 2^{\mathcal{F}}.\exists a \in \alpha(s)$; • *Existence of environment reaction*: $\forall s \in 2^{\mathcal{F}}, a \in \alpha(s).\exists r \in \beta(s, a)$; • *Uniqueness of environment reaction*: $\forall s \in 2^{\mathcal{F}}, a \in \alpha(s).\delta(s, a, r_1) = \delta(s, a, r_2) \supset r_1 = r_2$.

---

[1]We use the term *adaptive* to qualify strategies with the three properties above, but the term "adaptive" can be used to denote very different concepts, see e.g., (D'Ippolito et al. 2014; Ciolek et al. 2020; Rodríguez and Sánchez 2024).

These properties allow the defined nondeterministic planning domain to capture classical FOND domains (Cimatti, Roveri, and Traverso 1998; Geffner and Bonet 2013) expressed in PDDL (Haslum et al. 2019) by introducing reactions corresponding to oneof clauses of agent actions (De Giacomo, Parretti, and Zhu 2023a).

A *trace* of $\mathcal{P}$ is a finite or infinite sequence $\tau = s_0(a_1, r_1, s_1) \cdots$, where s_0 is the initial state of $\mathcal{P}$, and s_i, a_i, and r_i, are the state, agent action, and environment reaction, respectively, at the i-th time step. A trace is *legal* if for every $i > 0$: (*i*) $a_i \in \alpha(s_{i-1})$; (*ii*) $r_i \in \beta(s_{i-1}, a_i)$; and (*iii*) $s_i = \delta(s_{i-1}, a_i, r_i)$. We denote by $\mathcal{H}_\mathcal{P}$ the set of legal traces of $\mathcal{P}$. Given a trace $\tau = s_0(a_1, r_1, s_1) \cdots (a_n, r_n, s_n)$, we denote τ projected on the states by $\tau|_{2^\mathcal{F}} = s_0 \cdots s_n$, and τ projected on agent actions and environment reactions by $\tau|_{Act \times React} = (a_1, r_1) \cdots (a_n, r_n)$.

An *agent strategy* is a *partial* function $\sigma : (2^\mathcal{F})^+ \to Act$ that maps sequences of states of the domain to agent actions. We write $\sigma(\tau) = \bot$ to denote that σ is undefined in τ, which we think of as the strategy terminating its execution. Note that such agent strategies definitions are more in line with existing works on planning and reasoning about actions. However, in synthesis agent strategies are typically defined as $\sigma' : React^* \to Act$ that map sequences of environment reactions to agent actions. However, for the planning domains we consider, we can equivalently define agent strategies as $\sigma : (2^\mathcal{F})^+ \to Act$ or $\sigma' : React^* \to Act$. The equivalence follows by noting that: (*i*) every strategy $\sigma : (2^\mathcal{F})^+ \to Act$ directly corresponds to a strategy $\sigma' : React^* \to Act$ by the requirement of uniqueness of environment reaction; (*ii*) every strategy $\sigma' : React^* \to Act$ directly corresponds to a strategy $\sigma : (2^\mathcal{F})^+ \to Act$ since the transition function δ is deterministic. An agent strategy σ is *legal* if, for every legal trace $\tau = s_0(a_1, r_1, s_1) \cdots (a_n, r_n, s_n)$, if $\sigma(\tau|_{2^\mathcal{F}})$ is defined, then $\sigma(\tau|_{2^\mathcal{F}}) \in \alpha(s_n)$, so that a legal agent strategy always satisfies action preconditions.

An *environment strategy* is a *total* function $\gamma : (Act)^+ \to React$ mapping sequences of agent actions to environment reactions. An environment strategy γ is *legal* if, for every legal trace $\tau = s_0(a_1, r_1, s_1) \cdots (a_n, r_n, s_n)$ and agent action $a_{n+1} \in \alpha(s_n)$, we have that $\gamma(a_1 \cdots a_{n+1}) \in \beta(s_n, a_{n+1})$, so that a legal environment strategy always satisfies reaction preconditions. In the rest of the paper, we consider only legal agent strategies and legal environment strategies.

An agent strategy σ and an environment strategy γ induce a unique legal trace on $\mathcal{P}$ that is *consistent* with both, defined as $\texttt{Play}(\sigma, \gamma) = s_0(a_1, r_1, s_1) \cdots$, where: (*i*) s_0 is the initial state; (*ii*) for every $i > 0$, $a_i = \sigma(s_0 \cdots s_{i-1})$, $r_i = \gamma(a_1 \cdots a_i)$, and $s_i = \delta(s_{i-1}, a_i, r_i)$; and (*iii*) if $\texttt{Play}(\sigma, \gamma)$ is finite, say $\texttt{Play}(\sigma, \gamma) = s_0(a_1, r_1, s_1) \cdots (a_n, r_n, s_n)$, then $\sigma(s_0 \cdots s_n) = \bot$. Note that $\texttt{Play}(\sigma, \gamma)$ always exists by the properties of Existence of agent action and Existence of environment reaction.

Winning and Cooperative Strategies. Given a domain $\mathcal{P}$, an objective φ as an LTL$_f$ formula over the fluents in $\mathcal{P}$. A *finite* legal trace τ satisfies φ in $\mathcal{P}$, written $\tau \models_\mathcal{P} \varphi$, if $\tau|_{2^\mathcal{F}} \models \varphi$. We denote by $\mathcal{L}_\mathcal{P}(\varphi)$ the set of legal traces of $\mathcal{P}$

satisfying φ. An agent strategy σ is *cooperative* for φ in $\mathcal{P}$ if $\texttt{Play}(\sigma, \gamma)$ is finite and $\texttt{Play}(\sigma, \gamma) \models_\mathcal{P} \varphi$ for some environment strategy γ. Furthermore, σ is *winning* for (or *enforces*) φ in $\mathcal{P}$ if $\texttt{Play}(\sigma, \gamma)$ is finite and $\texttt{Play}(\sigma, \gamma) \models_\mathcal{P} \varphi$ for every environment strategy γ.

LTL$_f$ Best-Effort Synthesis. LTL$_f$ best-effort synthesis in nondeterministic domains computes a strategy that enables the agent to do its best to satisfy an LTL$_f$ objective in a nondeterministic domain (Aminof, De Giacomo, and Rubin 2021; De Giacomo, Parretti, and Zhu 2023a).

The notion of best-effort strategy bases on the game-theoretic relation of *dominance*. An agent strategy σ_1 dominates an agent strategy σ_2 (for φ in $\mathcal{P}$), written $\sigma_1 \geq_{\varphi|\mathcal{P}} \sigma_2$ if, for every environment strategy γ, $\texttt{Play}(\sigma_2, \gamma)$ is finite and $\texttt{Play}(\sigma_2, \gamma) \models_\mathcal{P} \varphi$ implies $\texttt{Play}(\sigma_1, \gamma)$ is finite and $\texttt{Play}(\sigma_1, \gamma) \models_\mathcal{P} \varphi$. Furthermore, σ_1 *strictly dominates* σ_2, written $\sigma_1 >_{\varphi|\mathcal{P}} \sigma_2$, if $\sigma_1 \geq_{\varphi|\mathcal{P}} \sigma_2$ and $\sigma_2 \not\geq_{\varphi|\mathcal{P}} \sigma_1$.

Intuitively, $\sigma_1 >_{\varphi|\mathcal{P}} \sigma_2$ means that σ_1 does at least as well as σ_2 against every environment strategy and strictly better against at least one such strategy. An agent using σ_2 is not doing its "best" to satisfy the goal. If the agent used σ_1 instead, it could achieve the goal against a strictly larger set of environment strategies. As a result, a best-effort strategy σ is one that is not strictly dominated by any other strategy, i.e., no other strategy σ' exists such that $\sigma' >_{\varphi|\mathcal{P}} \sigma$.

Best-effort strategies also admit an alternative characterization that describes their behavior when executed after a history, i.e., a finite legal trace. Given a domain $\mathcal{P}$, an agent strategy σ, and a history h, we denote by $\Gamma_\mathcal{P}(\sigma, h)$ the set of environment strategies γ such that h is a prefix of $\texttt{Play}(\sigma, \gamma)$. Also, we denote by $\mathcal{H}_\mathcal{P}(\sigma)$ the set of histories h such that $\Gamma_\mathcal{P}(\sigma, h)$ is non-empty, i.e., the set of histories that are consistent with σ and some environment strategy γ. Given an agent goal φ, we define:

- $\text{val}_{\varphi|\mathcal{P}}(\sigma, h) = win$ (σ is winning for φ from h) if $\texttt{Play}(\sigma, \gamma)$ is finite and $\texttt{Play}(\sigma, \gamma) \models_\mathcal{P} \varphi$ for every $\gamma \in \Gamma_\mathcal{P}(\sigma, h)$;
- $\text{val}_{\varphi|\mathcal{P}}(\sigma, h) = pend$ (σ is pending for φ from h) if, $\texttt{Play}(\sigma, \gamma)$ is finite and $\texttt{Play}(\sigma, \gamma) \models_\mathcal{P} \varphi$ for some $\gamma \in \Gamma_\mathcal{P}(\sigma, h)$, but not all;
- $\text{val}_{\varphi|\mathcal{P}}(\sigma, h) = lose$ (σ is losing for φ from h), otherwise.

The value of h, written $\text{val}_{\varphi|\mathcal{P}}(h)^2$, is the maximum of $\text{val}_\varphi(\sigma, h)$ for every agent strategy σ such that $h \in \mathcal{H}_\mathcal{P}(\sigma)$. The history-based characterization of best-effort strategies is as follows: an agent strategy σ is best-effort for φ in $\mathcal{P}$ iff $\text{val}_{\varphi|\mathcal{P}}(\sigma, h) = \text{val}_{\varphi|\mathcal{P}}(h)$ for every $h \in \mathcal{H}_\mathcal{P}(\sigma)$.

By the history-based characterization, a best-effort strategy σ behaves as follows. Starting from every history $h \in \mathcal{H}_\mathcal{P}(\sigma)$: (*i*) if φ is enforceable regardless of the nondeterminism in the domain, σ enforces φ; else, (*ii*) if φ is satisfiable only depending on how the nondeterminism in the domain unfolds, σ satisfies φ if the environment cooperates; else, (*iii*) if φ is unsatisfiable, σ prescribes some action which does not violate action preconditions. Note that a best-effort strategy adapts so that if a goal becomes enforceable due to environment cooperation, then the strategy will enforce it.

---

[2]We define $\text{val}_\varphi(h)$ only if there exists σ s.t. $h \in \mathcal{H}(\sigma)$.

By the definition of best-effort strategy, it follows that, if a winning strategy exists, the best-effort strategies are exactly the winning strategies. Furthermore, best-effort strategies have the notable property that they always exist. LTL$_f$ best-effort synthesis is 2EXPTIME-complete in the size of the goal φ and EXPTIME-complete in the size of the domain $\mathcal{P}$ (De Giacomo, Parretti, and Zhu 2023b).

3 Multi-Tier Goals

In this paper, we address synthesis with multi-tier goals consisting of a multi-tier hierarchy of increasingly challenging LTL$_f$ objectives: the initial objective is the simplest, with each subsequent objective adding more obligations, making the final objective the most challenging. Formally, we define a multi-tier goal specified in LTL$_f$ as follows.

Definition 1 (Multi-Tier Goal)**.** *Given a nondeterministic planning domain $\mathcal{P}$ with fluents $\mathcal{F}$, a multi-tier LTL$_f$ goal is a sequence of LTL$_f$ objectives $\Phi = \langle \varphi_1, \cdots, \varphi_n \rangle$ over $\mathcal{F}$ such that $\mathcal{L}_{\mathcal{P}}(\varphi_1) \supseteq \cdots \supseteq \mathcal{L}_{\mathcal{P}}(\varphi_n)$.*

Multi-tier LTL$_f$ goals can capture *ordinal temporal preferences* (Son and Pontelli 2006), which enable users to specify when a strategy is better than another. Given a multi-tier goal $\Phi = \langle \varphi_1, \cdots, \varphi_n \rangle$, we can view φ_1 as the "hard" objective that the agent should enforce, and the various φ_i (where $i > 1$) as "soft" objectives that refine φ_1, which the agent should satisfy if possible (Keyder and Geffner 2009). As a result, the strategies that satisfy φ_i are preferred to those that only satisfy φ_j for $j < i$.

It follows by Definition 1 that if an agent strategy enforces (resp. cooperates for) φ_i, then it also enforces (resp. cooperates for) every φ_j such that $j < i$.

One important aspect of synthesis in nondeterministic domains is that, at every time step, an objective φ can be:
- *Enforceable*, i.e., there exists an agent strategy that satisfies φ regardless of adversarial environment responses;
- *Pending*, i.e., there exists an agent strategy that satisfies φ should the environment cooperatively respond;
- *Unsatisfiable*, i.e., no agent strategy exists that satisfies φ regardless of environment responses.

It is worth noting that an objective that is currently pending might become enforceable during execution should the environment cooperate (or unsatisfiable should the environment be adversarial). Hence in order to fully exploit the nondeterminism in a nondeterministic domain considering a multi-tier goal, the agent should employ an *adaptive strategy*, i.e., a strategy with the following properties:
- *Property 1.* The strategy enforces the satisfaction of all objectives that are currently enforceable;
- *Property 2.* The strategy exploits possible cooperation from the environment to satisfy as many as possible of the currently pending objectives;
- *Property 3.* Dynamically, at each time step, if the environment cooperates and a pending objective becomes enforceable, the strategy will enforce its satisfaction.

Synthesizing a strategy that achieves Properties 1, 2, and 3, requires more sophisticated techniques than those developed for PBP in deterministic domains (Son and Pontelli 2006; Baier, Bacchus, and McIlraith 2009; Keyder and Geffner 2009), where the environment response is fixed and objectives are only either enforceable or unsatisfiable. Moreover handling ordinal temporal objectives in FOND requires significant advancements compared to PBP in FOND planning addressed in (Shaparau, Pistore, and Traverso 2006) for static preferences (which can only express additional "soft" reachability objectives). In particular, the machinery developed for best-effort strategies for singular objectives (Aminof, De Giacomo, and Rubin 2021) plays a prominent role in synthesizing adaptive strategies for multi-tiers goals.

4 LTL$_f$ Adaptive Synthesis for Multi-Tier Goals

We now formalize the notion of adaptive strategies for multi-tier goals that comply with Properties 1, 2, and 3 discussed in the previous section.

We define adaptive strategies for a multi-tier goal by leveraging the notion of best-effort strategy for a single objective. Specifically, we leverage the history-based characterization of best-effort strategies to satisfy the three properties above when considering multi-tier goals.

Note that an adaptive strategy requires identifying all enforceable objectives at every point in time (cf. Property 1 and Property 3). Therefore, we define the *maximally winning objective* as the highest tier that the agent can enforce, regardless of an adversarial environment in a nondeterministic planning domain, for any given history.

Definition 2 (Maximally Winning Objective)**.** *Let $\mathcal{P}$ be a nondeterministic planning domain, $\Phi = \langle \varphi_1, \cdots, \varphi_n \rangle$ a multi-tier goal, and $h \in \mathcal{H}_{\mathcal{P}}$ a history. We say that φ_ℓ is the maximally winning objective wrt h if $\mathrm{val}_{\varphi_\ell | \mathcal{P}}(h) = win$ and $\mathrm{val}_{\varphi_j | \mathcal{P}}(h) \neq win$ for every $j > \ell$.*

One can define the *maximally pending objective* analogously, i.e., the highest tier that the agent can satisfy should the environment cooperate.

An adaptive strategy must always leverage possible environment cooperation to fulfill all objectives that are currently not enforceable but satisfiable, while continuously enforcing those objectives that are enforceable. To this end, we define the *maximally winning-pending objective*, referring to the highest tier objective the agent can satisfy with environment cooperation, while simultaneously enforcing those currently enforceable (cf. Property 1 and Property 2).

Definition 3 (Maximally Winning-Pending Objective)**.** *Let $\mathcal{P}$ be a nondeterministic planning domain, $\Phi = \langle \varphi_1, \cdots, \varphi_n \rangle$ a multi-tier goal, $h \in \mathcal{H}_{\mathcal{P}}$ a history, and φ_ℓ the maximally winning objective of Φ wrt h. We say that φ_k ($k > \ell$) is the maximally winning-pending objective wrt h if (i) there exists a strategy σ consistent with h that is winning for φ_ℓ such that $\mathrm{val}_{\varphi_k | \mathcal{P}}(\sigma, h) = pend$, and (ii) it does not exist a strategy η consistent with h and winning for φ_ℓ such that $\mathrm{val}_{\varphi_j | \mathcal{P}}(\eta, h) = pend$ for some $j > k$.*

We now define our notion of adaptive strategy for multi-tier goals. Specifically:
- To achieve Property 1, the adaptive strategy will always enforce the maximally winning objective, if one exists,

regardless of the adversarial environment, thus enforcing every objective that is currently enforceable;

- To achieve Property 2, the adaptive strategy will always satisfy the maximally winning-pending objective or maximally pending objective (if the maximally winning objective does not exist), if one exists, should the environment cooperate, thus satisfying every pending objective;
- To achieve Property 3, Properties 1 and 2 should hold for every history consistent with the adaptive strategy.

Formally, adaptive strategies are defined as follows:

Definition 4 (Adaptive Strategy for Multi-Tier Goals)**.** *Let $\mathcal{P}$ be a nondeterministic planning domain and $\Phi = \langle \varphi_1, \cdots, \varphi_n \rangle$ a multi-tier goal. An agent strategy σ is an adaptive strategy for Φ in $\mathcal{P}$ if, for every $h \in \mathcal{H}_{\mathcal{P}}(\sigma)$, one of the following holds:*

1. *Suppose both the maximally winning objective φ_ℓ and the maximally winning-pending objective φ_k exist, then: (i) σ is winning for φ_ℓ in $\mathcal{P}$ from h, and (ii) there exists an agent strategy σ' such that $\mathrm{val}_{\varphi_k | \mathcal{P}}(\sigma', h) = pend$ and $\sigma(h) = \sigma'(h)$*
2. *Suppose the maximally winning objective φ_ℓ exists, yet no maximally winning-pending exists, then σ is winning for φ_ℓ in $\mathcal{P}$ from h;*
3. *Suppose neither the maximally winning objective nor the maximally winning-pending objective exists, yet there exists a maximally pending objective φ_p, then there exists an agent strategy σ' such that $\mathrm{val}_{\varphi_p | \mathcal{P}}(\sigma', h) = pend$ and $\sigma(h) = \sigma'(h)$.*

In this paper, we study synthesis of adaptive strategies for LTL_f multi-tier goals in nondeterministic planning domains.

Definition 5 (LTL_f Adaptive Synthesis for Multi-Tier Goals)**.** *Given a nondeterministic planning domain $\mathcal{P}$ and an LTL_f multi-tier goal Φ, adaptive synthesis for multi-tier goals is to compute an adaptive strategy for Φ in $\mathcal{P}$.*

The major contribution of this paper is a game-theoretic synthesis technique that computes adaptive strategies for LTL_f multi-tier goals, which we will show in Section 7. The correctness of the synthesis technique also shows that LTL_f adaptive strategies for multi-tier goals always exist (analogously to LTL_f best-effort strategies for single objectives).

Theorem 1. *An adaptive strategy always exists for an LTL_f multi-tier goal Φ in a planning domain $\mathcal{P}$.*

5 Illustration Example.

We present an example drawn from robot navigation to illustrate the notions presented in previous sections and show how multi-tier goals can capture temporal preferences.

Consider a cleaning robot operating in a circular building, as shown in Figure 1. The robot can freely access Offices A, B, C, and D. However, entry to Labs I and II requires passing through secure gates in the hallway, shown in brown in Figure 1. These gates restrict access when hazardous materials are present, managed by the building manager. Solid arrows indicate areas with unrestricted access, while dashed arrows represent limited-access zones. Assume the robot starts in Office A, with the secure gates initially open. Consider the multi-tier goal $\Phi = \langle \varphi_1, \varphi_2, \varphi_3 \rangle$, where:
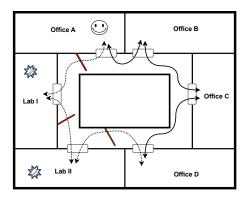

Figure 1: Robot working in an office building.

- $\varphi_1 = \Diamond(\mathit{Office\_D\_clean})$;
- $\varphi_2 = \Diamond(\mathit{Office\_D\_clean}) \wedge \Diamond(\mathit{Lab\_II\_clean})$;
- $\varphi_3 = \Diamond(\mathit{Lab\_II\_clean} \wedge \bigcirc(\Diamond(\mathit{Office\_D\_clean})))$;

The multi-tier goal Φ specifies the robot should clean Office D (φ_1), but cleaning Lab II also is preferable (φ_2), and cleaning Lab II before Office D (φ_3) is even more preferable.

Initially, φ_1 is enforceable, while φ_2 and φ_3 are pending. Thus, φ_1 is the maximally winning objective at this step. A winning strategy for φ_1 is to clean Office D without selecting a path that traverses the secure gates. This is because if the robot moves through the secure gates, the building manager might close the gates and lock the robot in one of the Labs, thus preventing φ_1 from being satisfied.

With φ_1 as the maximally winning objective, we have that φ_2 is the maximally winning-pending objective, as any winning strategy for φ_1 avoids clean Lab II before Office D such that not satisfying φ_3. A strategy σ that navigates through Offices A, B, C, and D, cleans Office D, and proceeds to cleans Lab II if the secure gate is open, is winning for φ_1 and cooperative for φ_2. Once the robot finishes cleaning Office D, φ_2 becomes enforceable if the building manager left secure gates open, in which case φ_2 becomes the maximally winning objective and σ enforces φ_2 as well. Thus σ is a strategy that satisfies Definition 4 and is adaptive for Φ.

This example also shows that a best-effort strategy for any single objective in Φ is not guaranteed to be an adaptive strategy for the multi-tier goal Φ. A best-effort (winning) strategy for φ_1 can navigate to Offices A, B, C, and D, then clean Office D, and stay there, so that φ_2 would never be satisfied. Instead, a best-effort strategy for φ_2 can navigate to and clean Lab II and thereafter navigate to and clean Office D. This might get the robot stuck between secure gates so that not satisfying φ_1 (analogously for φ_3).

6 Building Blocks

We develop a game-theoretic technique to address adaptive synthesis for multi-tier goals, involving two-player games over deterministic finite automata, reviewed briefly below.

A *deterministic transition system* is a tuple $\mathcal{T} = (\Sigma, Q, \iota, \varrho)$, where: Σ is a finite input alphabet; Q is a finite set of states; $\iota \in Q$ is the initial state; and $\varrho : Q \times \Sigma \to Q$ is the transition function. The *size* of $\mathcal{T}$ is $|Q|$. Given a finite trace $\pi = \pi_0 \cdots \pi_n$ over Σ, we extend ϱ to a func-

tion $\varrho : Q \times \Sigma^* \to Q$ as follows: $\varrho(q, \lambda) = q$ and, if $q_n = \varrho(q, \pi_0 \cdots \pi_{n-1})$, then $\varrho(q, \pi_0 \cdots \pi_n) = \varrho(q_n, \pi_n)$.

Definition 6. *The product of two transition systems $\mathcal{T}_i = (\Sigma, Q_i, \iota_i, \varrho_i)$ (for $i = 1, 2$) is the transition system* PRODUCT$(\mathcal{T}_1, \mathcal{T}_2) = (\Sigma, Q_1 \times Q_2, (\iota_1, \iota_2), \varrho)$, *where* $\varrho((q_1, q_2), a, r) = (\varrho_1(q_1, a, r), \varrho_2(q_2, a, r))$.

Given a transition system $\mathcal{T} = (\Sigma, Q, \iota, \varrho)$ and a set of states $V \subseteq Q$, the complement of V wrt Q is $\overline{V} = Q \setminus V$.

A *deterministic finite automaton* (DFA) is a tuple $\mathcal{A} = (\mathcal{T}, R)$, where $\mathcal{T} = (\Sigma, Q, \iota, \varrho)$ is a transition system, and $R \subseteq Q$ is a set of *final states*. A word $\pi \in \Sigma^*$ is *accepted* by $\mathcal{A}$ if $\varrho(\iota, \pi) \in R$. The language recognized by $\mathcal{A}$, written $\mathcal{L}(\mathcal{A})$, is the set of words that the automaton accepts.

Theorem 2. *(De Giacomo and Vardi 2013) Given an* LTL$_f$ *formula φ we can build a* DFA*, denoted* TODFA(φ), *with size at most double-exponential in $|\varphi|$ and whose language is the set of finite traces that satisfy φ.*

A DFA game is a DFA with alphabet $Act \times React$, where Act and $React$ are two disjoint sets under control of agent and environment, respectively. Formally, a DFA game is a pair $\mathcal{G} = (\mathcal{T}, R)$, where: $\mathcal{T} = (Act \times React, Q, \iota, \varrho)$ is a transition system and $R \subseteq Q$ is a set of final states.

A game strategy is a partial function $\kappa : Q \to Act$ that maps states of the game to agent actions. Given a game strategy κ, a sequence of environment reactions $\vec{r} = r_0 r_1 \ldots \in React^\omega$, and a DFA game $\mathcal{G}$, we denote by $\text{Run}(\kappa, \vec{r}, \mathcal{G})$, the *run* $q_0 q_1 \cdots$ of states of $\mathcal{G}$ *induced* by (or *consistent with*) κ and $\vec{r}$ as follows: (*i*) q_0 is the initial state of $\mathcal{G}$; (*ii*) for every $i \geq 0$, we have $q_{i+1} = \varrho(q_i, a_i, r_i)$, where $a_i = \kappa(q_i)$; and (*iii*) if $\text{Run}(\kappa, \vec{r}, \mathcal{G}) = q_0 \cdots q_n$ is finite, then $\kappa(q_n) = \bot$. A game strategy is winning (resp. cooperative) in $\mathcal{G}$ if $\rho = \text{Run}(\kappa, \vec{r}, \mathcal{G})$ is finite and $\text{lst}(\rho) \in R$ for every $\vec{r} \in React^\omega$ (resp. for some $\vec{r} \in React^\omega$). A state $q \in Q$ is a *winning* (resp. *cooperative*) *state* if the agent has a winning (resp. cooperative) game strategy in the game $\mathcal{G}' = (\mathcal{T}', R)$, where $\mathcal{T}' = (Act \times React, Q, q, \varrho)$, i.e., the same game as $\mathcal{G}$, but with initial state q. The *winning* (resp. *cooperative*) *region* W (resp. C) is the set of winning (resp. cooperative) states. A game strategy that is winning from every state in the winning (resp. cooperative) region is called *uniform winning* (resp. *uniform cooperative*). We observe that while a game strategy is not formally an agent strategy (i.e., a function from environment reactions to agent actions), it *induces* one as follows.

Definition 7. *Given a transition system $\mathcal{T} = (Act \times React, Q, \iota, \delta)$, a game strategy $\kappa : Q \to Act$ induces an agent strategy $\sigma' : React^* \to Act$ as follows: for every $h \in (Act \times React)^*$, $\sigma'(h|_{React}) = \kappa(\delta(\iota, h))$. The pair $(\mathcal{T}, \kappa)$ denotes a transducer, i.e., a transition system with output function κ.*

(We also recall: in our setting every strategy $\sigma' : React^* \to Act$ is equivalent to a strategy $\sigma : (2^\mathcal{F})^+ \to Act$.)

Solving a DFA game $\mathcal{G}$ aims at computing the winning (resp. cooperative) region and a uniform winning (resp. cooperative) strategy, written $(W, \kappa) = \text{SOLVEADV}(\mathcal{G})$ (resp. $(C, \nu) = \text{SOLVECOOP}(\mathcal{G})$). DFA games can be solved in linear time in their size by a fixpoint computation over the state

space of the game (Apt and Grädel 2011). For states not in W (resp. C), we assume that κ (resp. ν) is undefined.

Given two DFAs $\mathcal{A}_i = (\mathcal{T}_i, R_i)$ for $(i = 1, 2)$ and the product $\mathcal{T}$ of their transition systems, we can *lift* the final states R_i to the product $\mathcal{T}$ as follows.

Definition 8. *For* DFA*s $\mathcal{A}_i = (\mathcal{T}_i, R_i)$ and $\mathcal{T} = $* PRODUCT$(\mathcal{T}_1, \mathcal{T}_2)$, *the lifting of R_i to $\mathcal{T}$ is the set of states* LIFT$(\mathcal{T}, R_i) = \{(q_1, q_2) \in Q_1 \times Q_2 \text{ s.t. } q_i \in R_i\}$.

We now show how to transform planning domains into DFAs by introducing two error states s_{err}^{ag} and s_{err}^{env}, denoting that agent and environment violate their respective preconditions, as in (De Giacomo, Parretti, and Zhu 2023b).

Definition 9. *Given a nondeterministic planning domain $\mathcal{P} = (2^\mathcal{F}, s_0, Act, React, \alpha, \beta, \delta)$, we define* $(\mathcal{P}_+, \{s_{err}^{ag}\}, \{s_{err}^{env}\}) = $ TODFA$(\mathcal{P})$[3], *where $\mathcal{P}_+ = (Act \times React, 2^\mathcal{F} \cup \{s_{err}^{ag}, s_{err}^{env}\}, s_0, \delta')$ is a transition system with transition function δ' such that $\delta'(s, a, r) = \delta(s, a, r)$ if $a \in \alpha(s)$ and $r \in \beta(s, a)$; or $\delta'(s, a, r) = s_{err}^{ag}$ if $a \notin \alpha(s)$; or $\delta'(s, a, r) = s_{err}^{env}$ if $a \in \alpha(s)$ and $r \notin \beta(s, a)$.*

Our synthesis technique utilizes the synthesis approach developed for LTL$_f$ synthesis in nondeterministic domains (De Giacomo, Parretti, and Zhu 2023a). This approach involves constructing the product of the transition systems of the domain, defined over $Act \times React$ and the DFA of the LTL$_f$ objective, defined over $2^\mathcal{F}$.

Definition 10. *Let $\mathcal{P}_+ = (Act \times React, 2^\mathcal{F} \cup \{s_{err}^{ag}, s_{err}^{env}\}, s_0, \delta')$ be the transition system of a domain $\mathcal{P}$ and $\mathcal{T}_\varphi = (2^\mathcal{F}, Q, \iota, \varrho)$ the transition system of the* DFA *of an* LTL$_f$ *objective. The product of $\mathcal{P}_+$ and $\mathcal{T}_\varphi$ is* PRODUCT$(\mathcal{P}_+, \mathcal{T}_\varphi) = (Act \times React, (2^\mathcal{F} \cup \{s_{err}^{ag}, s_{err}^{env}\}) \times Q, (s_0, \varrho(\iota, s_0)), \partial)$, *where:*

$$\partial((s, q), a, r) = \begin{cases} (s', \varrho(q, s')) & \text{if } s' \notin \{s_{err}^{ag}, s_{err}^{env}\} \\ (s_{err}^{ag}, q) & \text{if } s' = s_{err}^{ag} \\ (s_{err}^{env}, q) & \text{if } s' = s_{err}^{env} \end{cases}$$

with $s' = \delta'(s, a, r)$.

Intuitively, $\mathcal{T}$ is a transition system that simultaneously retains the progression in $\mathcal{P}$ and $\mathcal{T}_\varphi$.

7 Synthesis Technique

In this section, we present a game-theoretic technique to address adaptive synthesis for LTL$_f$ multi-tier goals in nondeterministic planning domains. This technique relies on solving and combining on-the-fly the solutions of several DFA games, constructed from both the planning domain and the objectives within the multi-tier goal.

We first review the key steps for best-effort synthesis of a single LTL$_f$ objective (De Giacomo, Parretti, and Zhu 2023b), as outlined in Algorithm 1. That solves two distinct games over the same transition system $\mathcal{T}$ (with different final states) constructed from the domain and the LTL$_f$ objective. Solving these games returns the following: the winning region W with a uniform winning game strategy κ, and the

---

[3]For simplicity, we extend the notation TODFA to return two DFAs $(\mathcal{P}_+, \{s_{err}^{ag}\}, \{s_{err}^{env}\})$ and $(\mathcal{P}_+, \{s_{err}^{ag}\}, \{s_{err}^{env}\})$, which share the same transition system but differ in their final states.

Algorithm 1: SYNTHDSSINGLEOBJ($\mathcal{P}, \varphi$)

Input: Domain $\mathcal{P}$ and LTL$_f$ goal φ
Output: Trans. sys. $\mathcal{T}$, win. region W, win. strategy κ,
   coop. region C, and coop. strategy ν
1: $(\mathcal{P}_+, \{s_{err}^{ag}\}, \{s_{err}^{env}\}) = \text{ToDFA}(\mathcal{P})$
2: $(\mathcal{T}_\varphi, R_\varphi) = \text{ToDFA}(\varphi)$
3: $\mathcal{T} = \text{PRODUCT}(\mathcal{P}_+, \mathcal{T}_\varphi)$
4: $AgErr = \text{LIFT}(\mathcal{T}, \{s_{err}^{ag}\})$
5: $EnvErr = \text{LIFT}(\mathcal{T}, \{s_{err}^{env}\})$
6: $R'_\varphi = \text{LIFT}(\mathcal{T}, R_\varphi)$
7: $Adv = \overline{AgErr} \cap (EnvErr \cup R'_\varphi)$
8: $Coop = \overline{AgErr} \cap \overline{EnvErr} \cap R'_\varphi$
9: $(\text{W}, \kappa) = \text{SOLVEADV}(\mathcal{T}, Adv)$
10: $(\text{C}, \nu) = \text{SOLVECOOP}(\mathcal{T}, Coop)$
11: **Return** $(\mathcal{T}, \text{W}, \text{C}, \kappa, \nu)$

Algorithm 2: SYNTHDSWINPEND($\mathcal{P}, \varphi_1, \varphi_2$)

Input: Domain $\mathcal{P}$; LTL$_f$ objs φ_1, φ_2 st $\mathcal{L}_\mathcal{P}(\varphi_1) \supseteq \mathcal{L}_\mathcal{P}(\varphi_2)$
Output: Win-coop region WP; agent strategy σ
1: $(\mathcal{P}_+, s_{err}^{ag}, s_{err}^{env}) = \text{ToDFA}(\mathcal{P})$
2: **For** $i = 1, 2$:
   2.1: $(\mathcal{T}_{\varphi_i}, R_{\varphi_i}) = \text{ToDFA}(\varphi)$
   2.2: $\mathcal{T}_i = \text{PRODUCT}(\mathcal{P}_+, \mathcal{T}_{\varphi_i})$
      /* Q'_i is the state space of $\mathcal{T}_i$ */
   2.3: $AgErr_i = \text{LIFT}(\mathcal{T}_i, \{s_{err}^{ag}\})$
   2.4: $EnvErr_i = \text{LIFT}(\mathcal{T}_i, \{s_{err}^{env}\})$
   2.5: $R'_\varphi = \text{LIFT}(\mathcal{T}, R_\varphi)$
   2.6: $Adv_i = \overline{AgErr_i} \cap (EnvErr_i \cup R'_{\varphi_i})$
   2.7: $Coop_i = \overline{AgErr_i} \cap \overline{EnvErr_i} \cap R'_{\varphi_i}$
3: $(\text{W}_1, \kappa_1) = \text{SOLVEADV}(\mathcal{T}_1, Adv_1)$
4: $(\text{C}_2, \text{-}) = \text{SOLVECOOP}(\mathcal{T}_2, Coop_2)$
5: $\mathcal{T} = \text{PRODUCT}(\mathcal{T}_1, \mathcal{T}_2)$
   /* $Q' = Q'_1 \times Q'_2$ is the state space of $\mathcal{T}$ */
   /* ∂ is the transition function of $\mathcal{T}$ */
6: Let $\text{WP}_{-1} = \{\emptyset\}$ and $\text{WP}_0 = Adv_1 \times Coop_2$
7: While $\text{WP}_{i+1} \neq \text{WP}_i$:
   Let $\text{WP}_{i+1} = \{q' \in Q' \mid \exists a. \exists r. \partial(q', a, r) \in \text{WP}_i \wedge$
      $\forall r. \partial(q', a, r) \in \text{WP}_i \cup (\text{W}_1 \times \overline{\text{C}_2})\}$
8: Define strategy σ on $\mathcal{T}$ as follows. For every (q'_1, q'_2):

$$\sigma((q'_1, q'_2)) = \begin{cases} a & \text{if } (q'_1, q'_2) \in \text{WP}_{i+1} \setminus \text{WP}_i \\ & \text{and } \exists r. \partial((q'_1, q'_2), a, r) \in \text{WP}_i \\ \kappa_1(q'_1) & \text{otherwise} \end{cases}$$

9: **Return** (WP, σ)

---

cooperative region C with a uniform cooperative game strategy ν. We have that W, κ, C, and ν satisfy the following: paths ending in W correspond to histories with value *win*, as witnessed by κ; paths ending in C \ W correspond to histories with value *pend*, as witnessed by ν; the remaining paths correspond to histories with value *lose*. This property will later serve to detect maximally winning and maximally pending objectives when synthesizing adaptive strategies.

A crucial step in adaptive synthesis for multi-tier goals is to determine, for every history, the current maximally winning-pending objective. To this end, we provide in Algorithm 2 an auxiliary procedure that, for a pair of LTL$_f$ objectives $\langle \varphi_1, \varphi_2 \rangle$ such that $\mathcal{L}_\mathcal{P}(\varphi_1) \supseteq \mathcal{L}_\mathcal{P}(\varphi_2)$, computes a strategy σ that satisfies the following: for every history h, if there exists a strategy to enforce φ_1 from h and to satisfy φ_2 from h with environment cooperation, i.e., currently, φ_1 is a winning objective and φ_2 is a winning-pending objective, then σ should fulfill both φ_1 and φ_2 if the environment cooperates; if fulfilling both is not feasible, i.e., currently, φ_2 is not a winning-pending objective, then σ enforces only φ_1. Intuitively, σ prioritizes satisfying both φ_1 and φ_2 when possible, but defaults to enforcing φ_1 if necessary.

Concretely, Algorithm 2 first computes the following: the winning region W$_1$ and a winning strategy κ_1 for the game obtained from φ_1; and the cooperative region C$_2$ for the game obtained from φ_2 (Lines 1-4). Next, Algorithm 2 identifies game states, where paths ending at these states indicate that currently φ_1 is the maximally winning objective and φ_2 is the maximally winning-pending objective within the multi-tier goal $\langle \varphi_1, \varphi_2 \rangle$. Such states are collected in WP through the fixpoint computation in Lines 5-7, in a game that aims to satisfy φ_1 adversarially, meanwhile satisfying φ_2 cooperatively. A state q' is added to WP$_{i+1}$ when there exists an agent action a such that: (*i*) some environment reaction moves the agent towards satisfying both φ_1 and φ_2, written $\exists r. \partial(q', a, r) \in \text{WP}_i$; *and* (*ii*) every environment reaction either moves the agent towards satisfying both φ_1 and φ_2 or prevents φ_2 from being satisfied, but still ensures φ_1, written $\forall r. \partial(q', a, r) \in \text{WP}_i \cup (\text{W}_1 \times \overline{\text{C}_2})$. Paths ending in WP directly corresponds to histories that admit a strategy

that simultaneously wins φ_1 and cooperates for φ_2. Algorithm 2 constructs the output strategy σ by combining κ_1 for enforcing φ_1 by default and the fixpoint computation information while collecting WP (Line 8). The construction is the following: for every state in WP, for which there exists an action that definitely advances φ_1 and can advance φ_2 if the environment cooperates, then σ follows this action (first case, Line 8); otherwise σ follows κ_1, advancing φ_1 only (second case, Line 8).

Let $\text{val}_{\langle \varphi_1, \varphi_2 \rangle | \mathcal{P}}(h)$ denote $\langle \text{val}_{\varphi_1 | \mathcal{P}}(h), \text{val}_{\varphi_2 | \mathcal{P}}(h) \rangle$ (and similarly for $\text{val}_{\langle \varphi_1, \varphi_2 \rangle | \mathcal{P}}(\sigma, h)$). The following theorem shows the correctness of Algorithm 2.

Theorem 3. *Let $\mathcal{P}$ be a domain, φ_1 and φ_2 two LTL$_f$ objectives such that $\mathcal{L}_\mathcal{P}(\varphi_1) \supseteq \mathcal{L}_\mathcal{P}(\varphi_2)$, and σ the strategy returned by Algorithm 2. For every $h \in \mathcal{H}_\mathcal{P}(\sigma)$, if $\text{val}_{\langle \varphi_1, \varphi_2 \rangle | \mathcal{P}}(h) = \langle win, pend \rangle$, either:*

1. *$\text{val}_{\langle \varphi_1, \varphi_2 \rangle | \mathcal{P}}(\sigma, h) = \langle win, pend \rangle$; or*
2. *If no agent strategy η exists s.t. $\text{val}_{\langle \varphi_1, \varphi_2 \rangle | \mathcal{P}}(\eta, h) = \langle win, pend \rangle$, then $\text{val}_{\varphi_1 | \mathcal{P}}(\sigma, h) = win$.*

With Algorithms 1 and 2 in place, we present our complete adaptive synthesis technique for LTL$_f$ multi-tier goals, outlined in Algorithm 3. Algorithm 3 begins by computing all the winning and cooperative game strategies for every objective in the multi-tier goal Φ using Algorithm 1. It then proceeds to compute strategies for winning-pending objectives with Algorithm 2, covering all combinations of objective pairs (φ_i, φ_j), where $j > i$. The output adap-

Algorithm 3: SYNTHMULTITIER$(\mathcal{P}, \Phi)$

Input: Domain $\mathcal{P}$; Multi-tier goal $\Phi = \langle \varphi_1, \cdots, \varphi_n \rangle$.
Output: Agent strategy that is adaptive for Φ in $\mathcal{P}$
1: **For** $i = 1, \cdots, n$:
    1.1: $(\mathcal{T}_i, W_i, C_i, \kappa_i, \nu_i) = $ SYNTHDSSINGLEOBJ$(\mathcal{P}, \varphi_i)$;
    1.2: **For** $j = i + 1, \cdots, n$:
      $(WP_{ij}, \omega_{ij}) = $ SYNTHDSWINPEND$(\mathcal{P}, \varphi_i, \varphi_j)$
2: **Return** $\sigma : (2^{\mathcal{F}})^+ \to Act$ defined as follows.
    Let h be the input history:
    • For $i = 1, \cdots, n$: let $q_i' = \partial_i(\iota_i, h|_{Act \times React})$, where ι_i and ∂_i are the initial state and transition function of $\mathcal{T}_i$, respectively.
    • $j = max\{i \text{ s.t. } q_i' \in W_i\}$
    • $\ell = max\{i \text{ s.t. } j > i \text{ and } (q_i', q_j') \in WP_{ij}\}$
    • $m = max\{i \text{ s.t. } q_i' \in C_i\}$

$$\sigma(h) = \begin{cases} \omega_{j\ell}(q_j', q_\ell') & \text{if } \ell > j > 0 \text{ exists} \\ \kappa_j(q_j') & \text{else if } j > 0 \text{ exists} \\ \nu_m(q_m') & \text{else if } m > 0 \text{ exists} \\ \bot & \text{otherwise} \end{cases}$$

tive strategy is generated in the form of an executable program that selects, at each time step, the next action on-the-fly from among the actions suggested by all the previously computed strategies. The selection procedure involves checking the state corresponding to the input history h to identify which objectives are currently winning, pending, and winning-pending. Recall that paths ending in W_i (resp. C_i) are with value *win* (resp. *pend*). Hence we can determine the maximally winning objective φ_j, where $j = max\{i \text{ s.t. } q_i' \in W_i\}$. Subsequently, we can determine the maximally winning-pending objective φ_ℓ, where $\ell = max\{i \text{ s.t. } j > i \text{ and } (q_i', q_j') \in WP_{ij}\}$. Analogously for the maximally pending objective φ_m. Based on this, the strategy selects an action following Properties 1,2&3. In details, at each time step, the strategy selects the output action as follows: (*i*) it selects the action returned by the strategy $\omega_{j\ell}$ that enforces the maximally winning objective φ_j and cooperates for the maximally winning-pending objective φ_ℓ, if any; else (*ii*) it selects the action returned by the strategy κ_j that enforces the maximally winning objective, if any; else (*iii*) it selects the action returned by the strategy ν_m that cooperates for the maximally pending objective φ_m, if any; else (*iv*) is undefined.

Concretely, the output strategy is implemented by executing simultaneously all transducers (representing the strategies) computed in Line 1. For the current history h, the output strategy σ evaluates the response of each such transducer, and selects the output action $\sigma(h)$ based on the tests in Line 2 and discussed above. The action $\sigma(h)$, together with the environment reaction, extends h, thus generating a new history, and the process starts again. The strategy terminates when it selects $\bot$, i.e., it is undefined, because either no maximally pending objective exists or the selected transducer outputs $\bot$. Figure 2 sketches the implementation of the strategy computed with Algorithm 3.

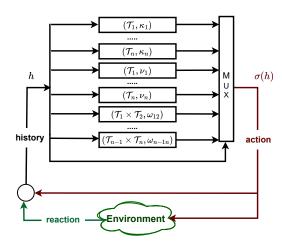**Theorem 4.** *Alg. 3 returns an adaptive strategy for Φ in $\mathcal{P}$.*

Figure 2: Implementation of the strategy returned by Alg. 3.

Regarding complexity, Algorithm 3 solves games with size 2EXPTIME and EXPTIME in that of objectives and domain, respectively. In fact, the number of solved games is *quadratic* in the number of objectives in the multi-tier goal. The following theorem shows the complexity characterization of adaptive synthesis for LTL$_f$ multi-tier goals in nondeterministic planning domains.

Theorem 5. LTL$_f$ *adaptive synthesis for multi-tier goals is:*
• 2EXPTIME-*complete in the size of the objectives;*
• EXPTIME-*complete in the size of the domain;*
• polynomial *in n, the number of objectives.*

We also note that Line 1 of Algorithm 3 is fully parallelizable. Synthesizing strategies for each objective can be done in parallel with n processors; synthesizing strategies for pairs of objectives can be done in parallel with n^2 processors. As a result, if $n + n^2$ processors are available, adaptive synthesis for multi-tier goals is *virtually* for free, i.e., it costs as handling the most expensive objectives. These nice computational results confirm that adaptive synthesis for multi-tier goals brings a minimal overhead to standard synthesis.

8 Conclusion and Future Work

In this paper, we introduced the problem of LTL$_f$ adaptive for multi-tier goals in nondeterministic planning domains. We developed a synthesis technique that is both sound and complete for computing an adaptive strategy in this context. This technique allows for straightforward utilization of symbolic synthesis techniques (Zhu et al. 2017; De Giacomo, Parretti, and Zhu 2023a), aiming for promising performance and scalability. Currently, our framework considers a single environment model. However, it is also of interest to consider the setting that involves multiple environment models (Aminof et al. 2020; Ciolek et al. 2020; Aminof et al. 2021), accounting for varying nondeterminism of the environment. For instance, as the environment becomes more nondeterministic, we might anticipate achieving a less challenging objective. We believe that the general approach presented here can be extended to handle this setting as well.

Acknowledgments

References

Aminof, B.; De Giacomo, G.; Lomuscio, A.; Murano, A.; and Rubin, S. 2020. Synthesizing Strategies under Expected and Exceptional Environment Behaviors. In *IJCAI*.

Aminof, B.; De Giacomo, G.; Lomuscio, A.; Murano, A.; and Rubin, S. 2021. Synthesizing Best-effort Strategies under Multiple Environment Specifications. In *KR*, 42–51.

Aminof, B.; De Giacomo, G.; Murano, A.; and Rubin, S. 2019. Planning under LTL Environment Specifications. In *ICAPS*, 31–39.

Aminof, B.; De Giacomo, G.; and Rubin, S. 2021. Best-Effort Synthesis: Doing Your Best Is Not Harder Than Giving Up. In *IJCAI*, 1766–1772.

Apt, K. R.; and Grädel, E., eds. 2011. *Lectures in Game Theory for Computer Scientists*. Cambridge University Press.

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. 2006. Planning with Qualitative Temporal Preferences. In *KR*.

Camacho, A.; Bienvenu, M.; and McIlraith, S. A. 2019. Towards a unified view of AI planning and reactive synthesis. In *ICAPS*, 58–67.

Camacho, A.; Icarte, R. T.; Klassen, T. Q.; Valenzano, R. A.; and McIlraith, S. A. 2019. LTL and Beyond: Formal Languages for Reward Function Specification in Reinforcement Learning. In *IJCAI*, 6065–6073.

Cimatti, A.; Roveri, M.; and Traverso, P. 1998. Strong Planning in Non-Deterministic Domains Via Model Checking. In *AIPS*, 36–43. AAAI.

Ciolek, D. A.; D'Ippolito, N.; Pozanco, A.; and Sardiña, S. 2020. Multi-Tier Automated Planning for Adaptive Behavior. In *ICAPS*, 66–74. AAAI Press.

De Giacomo, G.; Parretti, G.; and Zhu, S. 2023a. LTL$_f$ Best-Effort Synthesis in Nondeterministic Planning Domains. In *ECAI*, 533–540.

De Giacomo, G.; Parretti, G.; and Zhu, S. 2023b. Symbolic LTL$_f$ Best-Effort Synthesis. In *EUMAS*, 228–243.

De Giacomo, G.; and Rubin, S. 2018. Automata-Theoretic Foundations of FOND Planning for LTL$_f$ and LDL$_f$ 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.

D'Ippolito, N.; Braberman, V. A.; Kramer, J.; Magee, J.; Sykes, D.; and Uchitel, S. 2014. Hope for the Best, Prepare for the Worst: Multi-Tier Control for Adaptive Systems. In *ICSE*, 688–699. ACM.

Fijalkow, N.; Finkbeiner, B.; Pérez, G. A.; Polgreen, E.; and Morvan, R. 2024. The Futures of Reactive Synthesis (Dagstuhl Seminar 23391). *Dagstuhl Reports*, 13(9): 166–184.

Geatti, L.; Montali, M.; and Rivkin, A. 2024. Foundations of Reactive Synthesis for Declarative Process Specifications. In *AAAI*, 17416–17425. AAAI Press.

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

Ghallab, M.; Nau, D. S.; and Traverso, P. 2016. *Automated planning and acting*. Cambridge.

Haslum, P.; Lipovetzky, N.; Magazzeni, D.; and Muise, C. 2019. *An Introduction to the Planning Domain Definition Language*. Morgan & Claypool.

Jorge, A.; McIlraith, S. A.; et al. 2008. Planning with preferences. *AI Magazine*, 29(4): 25–25.

Keyder, E.; and Geffner, H. 2009. Soft goals can be compiled away. *Journal of Artificial Intelligence Research*, 36: 547–556.

Pnueli, A. 1977. The Temporal Logic of Programs. In *FOCS*, 46–57.

Pnueli, A.; and Rosner, R. 1989. On the Synthesis of a Reactive Module. In *POPL*, 179–190.

Rodríguez, A.; and Sánchez, C. 2024. Adaptive Reactive Synthesis for LTL and LTLf Modulo Theories. In *AAAI*, 10679–10686. AAAI Press.

Shaparau, D.; Pistore, M.; and Traverso, P. 2006. Contingent planning with goal preferences. In *Proceedings of the national conference on artificial intelligence*, volume 21, 927.

Son, T. C.; and Pontelli, E. 2006. Planning with preferences using logic programming. *Theory and Practice of Logic Programming*, 6(5): 559–607.

Zhu, S.; De Giacomo, G.; Pu, G.; and Vardi, M. Y. 2020. LTL$_f$ synthesis with fairness and stability assumptions. In *AAAI*, 3088–3095.

Zhu, S.; Tabajara, L. M.; Li, J.; Pu, G.; and Vardi, M. Y. 2017. Symbolic LTL$_f$ Synthesis. In *IJCAI*, 1362–1369.