ECAI 2023
K. Gal et al. (Eds.)

doi:10.3233/FAIA230313

LTL_f Best-Effort Synthesis in Nondeterministic Planning Domains

Giuseppe De Giacomo[a,b], **Gianmarco Parretti**[b;*] **and Shufang Zhu**[a;**]

[a]University of Oxford, UK
[b]University of Rome "La Sapienza", Italy

Abstract. We study best-effort strategies (aka plans) in fully observable nondeterministic domains (FOND) for goals expressed in Linear Temporal Logic on Finite Traces (LTL_f). The notion of best-effort strategy has been introduced to also deal with the scenario when no agent strategy exists that fulfills the goal against every possible nondeterministic environment reaction. Such strategies fulfill the goal if possible, and do their best to do so otherwise. We present a game-theoretic technique for synthesizing best-effort strategies that exploit the specificity of nondeterministic planning domains. We formally show its correctness and demonstrate its effectiveness experimentally, exhibiting a much greater scalability with respect to a direct best-effort synthesis approach based on re-expressing the planning domain as generic environment specifications.

1 Introduction

Recently there has been quite some interest in synthesis [27, 21] for realizing goals (or tasks) φ against environment specifications $\mathcal{E}$ [3, 4], especially when both φ and $\mathcal{E}$ are expressed in Linear Temporal Logic on finite traces (LTL_f) [19, 20], the finite trace variant of LTL [28], a logic specification language that is commonly adopted in Formal Methods [6]. In this setting, synthesis amounts to finding an agent strategy that wins, i.e., generates a trace satisfying φ, whatever is the (counter-)strategy chosen by the environment, which in turn has to satisfy its specification $\mathcal{E}$. This form of synthesis can be seen as an extension of FOND planning [23, 24], as shown in, e.g., [18, 10].

Obviously, a winning strategy for the agent may not exist. To handle this possibility, the notion of strong cyclic plans was introduced [12, 11]: if a (strong) plan does not exist, there may still exist a plan that could win assuming the environment is not strictly adversarial. Building on this intuition, Aminof et al. [1] proposed the notion of best-effort strategies (or plans), which formally capture the idea that the agent could do its best by adopting a strategy that wins against a maximal set (though not all) of possible environment strategies. It was then shown in [5] that best-effort strategies capture the game-theoretic rationality principle that a player (the agent) would not use a strategy that is "dominated" by another one (i.e., if another strategy fulfills the goal against more environment behaviors, then the player should adopt that strategy). Best-effort strategies have some notable properties: (*i*) they always exist, (*ii*) if a winning strategy exists, then best-effort strategies are exactly the winning strategies, (*iii*) best-effort strategies can be computed in 2EXPTIME as winning strategies (best-effort synthesis is indeed 2EXPTIME-complete).

In [5] an algorithm for LTL_f best-effort synthesis has been presented. This algorithm is based on creating, solving, and combining the solutions of three distinct games (with three different objectives) played in the same game arena. The arena is obtained from the deterministic finite-state automata (DFAs) corresponding to the LTL_f specification of the agent goal φ and the LTL_f specification of the environment $\mathcal{E}$. As a result, the size of the arena is, in the worst-case, double-exponential both in φ and in $\mathcal{E}$. Using this framework, we can also capture best-effort synthesis in nondeterministic planning domains. In particular, one can simply re-express nondeterministic planing domains (FOND) in LTL_f [19, 3, 26] and then use the LTL_f best-effort synthesis approach directly. However, observe that in planning, while the (temporally extended) goal φ is typically small, the environment specification $\mathcal{E}$ is large, being the entire planning domain (i.e., a representation of how the world the agent is immersed in works). This observation motivates our paper.

We study LTL_f best-effort synthesis directly in the context of nondeterministic (adversarial) planning domains. Specifically, our contributions are:

- A framework for best-effort synthesis in nondeterministic (adversarial) planning domains;
- A synthesis technique for best-effort synthesis, inspired by the one in [5], that takes full advantage of the specificity of planning domains as environment specifications;
- A symbolic best-effort synthesis algorithm that performs and scales well;
- An empirical evaluation of the practical effectiveness of the approach.

Our results show that computing best-effort strategies for LTL_f goals in nondeterministic domains is much more effective than using LTL_f best-effort synthesis directly. In fact, our technique can be implemented quite efficiently, with only a small overhead wrt to computing winning strategies (i.e., strong plans) in FOND. Hence, it is completely feasible in practice to return a best-effort strategy instead of giving up when a winning strategy does not exist.

Running Example. We now present a relatively simple example to illustrate the key structure of LTL_f best-effort synthesis in nondeterministic planning domains. This example, adapted from [26], represents a human-robot co-assembly task in a shared workplace. Specifically, the human and robot involved in the task are considered as the environment and the agent, respectively. In the shared workplace, the

* Corresponding Author. Email: parretti@diag.uniroma1.it
** Corresponding Author. Email: shufang.zhu@cs.ox.ac.uk

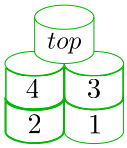

Figure 1: An arch. We denote location names in the blocks.

robot can perform grasp/place actions to pick/drop blocks and transfer/transit actions to move its robotic arm with/without a block in its gripper. After every robot action, the human can react by also moving blocks among locations to interfere with the robot, hence introducing nondeterminism to robot actions. Consider a robot *goal* (aka *task*) of assembling an arch of blocks, as depicted in Figure 1. It is easy to see that the agent has *no winning strategy* (aka *strong solution*) to assemble the arch, since the human can always disassemble it. Therefore, standard synthesis [20, 18] would conclude the task as *unrealizable*, hence "giving up". However, the robot still has the chance to fulfill the goal, should the human cooperate or even perform flawed reactions due to e.g. lack of adequate training. Therefore, instead of simply giving up, the agent should try its best to pursue the goal by exploiting human reactions. *Best-effort strategies* (aka *best-effort solutions*) precisely capture this intuition.

2 Preliminaries

Traces. Let Σ be a set of propositions. A *trace* $\pi = \pi_0\pi_1\ldots$ is a sequence of propositional interpretations (sets), where for every $i \geq 0$, $\pi_i \in 2^\Sigma$ is the i-th interpretation of π. Intuitively, π_i is interpreted as the set of propositions that are $true$ at instant i. The length of a trace is $|\pi|$. A trace π is an *infinite* trace if $|\pi| = \infty$, which is formally denoted as $\pi \in (2^\Sigma)^\omega$; otherwise π is a *finite* trace, denoted as $\pi \in (2^\Sigma)^*$. If a trace π is finite, we denote by $\mathtt{lst}(\pi)$ its last instant (i.e., index). Moreover, by $\pi^k = \pi_0 \cdots \pi_k$ we denote the *prefix* of π up to the k-th iteration.

LTL$_f$ Basics. *Linear Temporal Logic on finite traces* (LTL$_f$) is a specification language to express temporal properties on finite and non-empty traces [19]. In particular, LTL$_f$ has the same syntax as LTL, which is instead interpreted over infinite traces [28]. Given a set of propositions Σ, LTL$_f$ formulas are generated as follows:

$$\varphi ::= p \mid \varphi \wedge \varphi \mid \neg\varphi \mid \mathsf{O}\varphi \mid \varphi\,\mathcal{U}\,\varphi.$$

$p \in \Sigma$ is an *atom*, O (*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* $\bullet\varphi \equiv \neg\mathsf{O}\neg\varphi$, *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 subformulas of φ.

Given an LTL$_f$ formula φ over Σ and a finite, non-empty trace $\pi \in (2^\Sigma)^+$, we define when φ *holds* at instant i ($0 \leq i \leq \mathtt{lst}(\pi)$), written as $\pi, i \models \varphi$, 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 $\pi, i \models \varphi_2$;
- $\pi, i \models \mathsf{O}\varphi$ iff $i < \mathtt{lst}(\pi)$ and $\pi, i+1 \models \varphi$;
- $\pi, i \models \varphi_1\,\mathcal{U}\,\varphi_2$ iff $\exists j$ such that $i \leq j \leq \mathtt{lst}(\pi)$ and $\pi, j \models \varphi_2$, and $\forall k, i \leq k < j$ we have that $\pi, k \models \varphi_1$.

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

FOND Planning for LTL$_f$ Goals. Planning in *Fully Observable Nondeterministic* (FOND) domains for LTL$_f$ goals concerns computing a strategy to fulfill a temporally extended goal expressed as an LTL$_f$ formula in a planning domain, where the agent has full observability, regardless of how the environment non-deterministically reacts to agent actions. In this paper, we extend existing works on synthesis-based approaches to planning in FOND domains [18, 9, 10] to compute best-effort strategies [5].

3 Framework

We begin by presenting our framework. First, we introduce the notion of nondeterministic planning domain and then formalize the problem of LTL$_f$ best-effort synthesis in nondeterministic planning domains.

3.1 Nondeterministic Planning Domains

In our framework, a nondeterministic planning domain is intuitively considered as the arena of a *two-player game*, in which the agent and the environment can perform actions and reactions, respectively. Starting from the initial state of the domain, the agent and the environment move in turns, such that at each turn, the agent makes an action and the environment responds with some reaction. State transitions are determined only if both players complete their moves following the preconditions of their respective moves. In such a domain, the *nondeterminism* for the agent comes from not knowing how the environment will react.

Formally, we define a nondeterministic planning domain as a tuple $\mathcal{D} = (2^\mathcal{F}, s_0, Act, React, \alpha, \beta, \delta)$, where: $\mathcal{F}$ is a finite set of fluents such that $2^\mathcal{F}$ is the state space and $|\mathcal{F}|$ is the size of the domain; $s_0 \in 2^\mathcal{F}$ is the initial state; Act and $React$ are finite sets of agent actions and environment reactions, respectively; $\alpha : 2^\mathcal{F} \to 2^{Act}$ and $\beta : 2^\mathcal{F} \times Act \to 2^{React}$ denote preconditions of agent actions and environment reactions, respectively; and $\delta : 2^\mathcal{F} \times Act \times React \mapsto 2^\mathcal{F}$ is the transition function such that $\delta(s, a, r) = s' \in 2^\mathcal{F}$ if $a \in \alpha(s)$ and $r \in \beta(s, a)$, and $\delta(s, a, r)$ is undefined otherwise.

In particular, we require planning domains to satisfy the following three rules:

- *Existence of agent action*. For every state $s \in 2^\mathcal{F}$, there exists at least one agent action a such that $a \in \alpha(s)$. Formally:
$$\forall s \in 2^\mathcal{F}.\alpha(s) \neq \emptyset.$$
This rule guarantees that the agent can perform at least one action in every state of the domain[1].
- *Existence of environment reaction*. For every state $s \in 2^\mathcal{F}$ and agent action $a \in \alpha(s)$, there exists at least one environment reaction $r \in \beta(s, a)$ such that $\delta(s, a, r)$ is defined. Formally:
$$\forall s \in 2^\mathcal{F}, \forall a \in \alpha(s).\beta(s, a) \neq \emptyset.$$
This rule guarantees that the environment is able to respond to any agent action that follows the action precondition.
- *Uniqueness of environment reaction*. For every state $s \in 2^\mathcal{F}$, agent action $a \in \alpha(s)$, and successor state $s' = \delta(s, a, r)$ for some $r \in \beta(s, a)$, the reaction r is unique. Formally:
$$\forall s \in 2^\mathcal{F}, \forall a \in \alpha(s), \forall r_1, r_2 \in \beta(s, a).$$
$$\delta(s, a, r_1) = \delta(s, a, r_2) \supset r_1 = r_2.$$
This rule guarantees that, given a state s and an agent action $a \in \alpha(s)$, all the environment responses in $\beta(s, a)$ are distinguishable by just looking at the resulting successor state[2].

Observe that the nondeterministic planning domains adopted in FOND [12, 23], say expressed in PDDL [25], can be immediately captured by our notion along the line discussed in [17].

[1] If this is not the case, it is sufficient to add a new agent action nop with a corresponding environment reaction $nopr$ such that $\delta(s, nop, nopr) = s$.

[2] Observe that if this rule does not hold in our domain, we can easily modify the domain by introducing fluents to represent possible environment reactions and record the reaction in the resulting state, hence complying to the rule.

A *state trace* of $\mathcal{D}$ is a (finite or infinite) sequence $\tau = s_0 s_1 \ldots$ of states in $2^{\mathcal{F}}$ such that s_0 is the initial state of $\mathcal{D}$. A trace τ is *legal*, if for every i there exists an agent action a_i and an environment reaction r_i such that $s_{i+1} = \delta(s_i, a_i, r_i)$ with $a_i \in \alpha(s_i)$ and $r_i \in \beta(s_i, a_i)$.

We denote by $\vec{a} = a_0 a_1 \ldots$ (resp. $\vec{r} = r_0 r_1 \ldots$) a sequence of agent actions (resp. environment reactions) and by $\vec{a}^{\,k}$ (resp. $\vec{r}^{\,k}$) the prefix of $\vec{a}$ (resp. $\vec{r}$) up to action a_k (resp. reaction r_k). We use $\vec{a}_\epsilon$ and $\vec{r}_\epsilon$ to denote the empty sequence of agent actions and environment reactions (when $k < 0$), respectively. Consider $\vec{a}$ and $\vec{r}$ with the same length, we denote the trace induced by $\vec{a}$ and $\vec{r}$ on $\mathcal{D}$ as $\texttt{Trace}(\vec{a}, \vec{r})$ defined as follows:

- $\texttt{Trace}(\vec{a}_\epsilon, \vec{r}_\epsilon) = s_0$
- $\texttt{Trace}(\vec{a}^{\,k}, \vec{r}^{\,k}) =$
  $\texttt{Trace}(\vec{a}^{\,k-1}, \vec{r}^{\,k-1}) \cdot \delta(\texttt{lst}(\texttt{Trace}(\vec{a}^{\,k-1}, \vec{r}^{\,k-1}), a_k, r_k))$

If $\delta(\texttt{lst}(\texttt{Trace}(\vec{a}^{\,k-1}, \vec{r}^{\,k-1}), a_k, r_k))$ is undefined or $\vec{a}$ and $\vec{r}$ are in different length, $\texttt{Trace}(\vec{a}, \vec{r})$ returns undefined.

An agent strategy is a function $\sigma : (2^{\mathcal{F}})^+ \rightarrow Act$ mapping traces of states on $\mathcal{D}$ to agent actions. Note that in the synthesis literature, agent strategies are typically defined as functions $\sigma' : React^* \rightarrow Act$ mapping histories of environment reactions to agent actions. However, we define agent strategies equivalently as functions $\sigma : (2^{\mathcal{F}})^+ \rightarrow Act$ to be more in line with existing works on planning and reasoning about actions [18]. The equivalence follows by noting that: (*i*) every strategy $\sigma : (2^{\mathcal{F}})^+ \rightarrow Act$ directly corresponds to a strategy $\sigma' : React^* \rightarrow Act$ by the requirement of uniqueness of environment reaction; (*ii*) every strategy $\sigma' : React^* \rightarrow Act$ directly corresponds to a strategy $\sigma : (2^{\mathcal{F}})^+ \rightarrow Act$ since the transition function δ is deterministic. An agent strategy σ is *legal* if, for every legal trace τ, it holds that for all its prefixes τ^k, if $a_k = \sigma(\tau^k)$, then $a_k \in \alpha(\texttt{lst}(\tau^k))$.

An *environment strategy* is a function $\gamma : Act^+ \rightarrow React$. Given a sequence of agent actions $\vec{a}$ and an environment strategy γ, we denote by $\vec{\gamma}(\vec{a})$ the sequence of environment reactions obtained by applying γ recursively on every finite prefix $\vec{a}^{\,k}$ ($k \geq 0$) of $\vec{a}$, i.e., $\vec{\gamma}(\vec{a})_k = \gamma(\vec{a}^{\,k})$. An environment strategy is *legal*, if for every sequence of agent actions $\vec{a}$, it holds that for all its prefixes $\vec{a}^{\,k}$, if $\tau = \texttt{Trace}(\vec{a}^{k-1}, \vec{\gamma}(\vec{a}^{k-1}))$ is defined, hence legal, and $a_k \in \alpha(\texttt{lst}(\tau))$, then $r_k = \gamma(\vec{a}^{\,k-1} \cdot a_k) \in \beta(\texttt{lst}(\tau), a_k))$.

Given a legal agent strategy σ and a legal environment strategy γ, there exists a unique trace of $\mathcal{D}$ that is *induced* by both σ and γ, which we denote as $\texttt{Play}(\sigma, \gamma)$. Formally, $\texttt{Play}(\sigma, \gamma) = s_0 s_1 \ldots$ is such that s_0 is the initial state of $\mathcal{D}$ and for every $i \geq 0$, $s_{i+1} = \delta(s_i, a_i, r_i)$ with $a_i = \sigma(s_0 \ldots s_i)$ and $r_i = \gamma(a_0 \ldots a_i)$. We can prove by induction that $\texttt{Play}(\sigma, \gamma)$ is indeed defined and *legal*.

Theorem 1 *Let $\mathcal{D}$ be a planning domain, σ a legal agent strategy, and γ a legal environment strategy. Then, $\texttt{Play}(\sigma, \gamma)$ is defined and legal.*

Strong and Cooperative Solutions. Given a planning domain $\mathcal{D}$, an agent *goal* is an LTL$_f$ formula φ defined over the fluents $\mathcal{F}$ of the domain $\mathcal{D}$. A legal trace τ of $\mathcal{D}$ satisfies φ in $\mathcal{D}$, written $\tau \models_{\mathcal{D}} \varphi$, if there exists a prefix of τ that satisfies φ. An agent strategy σ is a *strong solution* for φ in $\mathcal{D}$ if, σ is legal and for every legal environment strategy γ, it holds that $\texttt{Play}(\sigma, \gamma) \models_{\mathcal{D}} \varphi$. If such a strategy σ exists, we say that there exists a strong solution for φ in $\mathcal{D}$, which is σ. Furthermore, an agent strategy σ is a *cooperative solution* for φ in $\mathcal{D}$, if σ is legal and there exists a legal environment strategy γ such that $\texttt{Play}(\sigma, \gamma) \models_{\mathcal{D}} \varphi$. If such a strategy σ exists, there exists a cooperative solution for φ in $\mathcal{D}$, which is σ.

3.2 Best-Effort Synthesis in Planning Domains

We now introduce the problem of LTL$_f$ best-effort synthesis in nondeterministic planning domains. In doing so, we need to define what it means for an agent to make its best-effort to achieve its goal in a planning domain. To formalize such strategies, we adapt the definitions in [5] and consider first the notion of dominance.

Definition 1 *Let $\mathcal{D}$ be a planning domain, φ an LTL$_f$ formula, and σ_1, σ_2 two legal agent strategies. σ_1 dominates σ_2 for φ in $\mathcal{D}$, written $\sigma_1 \geq_{\varphi|\mathcal{D}} \sigma_2$, if for every legal environment strategy γ, $\texttt{Play}(\sigma_2, \gamma) \models_{\mathcal{D}} \varphi$ implies $\texttt{Play}(\sigma_1, \gamma) \models_{\mathcal{D}} \varphi$.*

Furthermore, a legal agent strategy σ_1 *strictly dominates* legal agent strategy σ_2 for goal φ in $\mathcal{D}$, written $\sigma_1 >_{\varphi|\mathcal{D}} \sigma_2$, if $\sigma_1 \geq_{\varphi|\mathcal{D}} \sigma_2$ and $\sigma_2 \not\geq_{\varphi|\mathcal{D}} \sigma_1$. Intuitively, $\sigma_1 >_{\varphi|\mathcal{D}} \sigma_2$ shows that σ_1 does at least as well as σ_2 against every legal environment strategy in $\mathcal{D}$ and strictly better against at least one such strategy. If σ_1 strictly dominates σ_2, then an agent using σ_2 is not doing its "best" to achieve the goal. Within this framework, a best-effort solution is a legal agent strategy that is not strictly dominated by any other legal strategies.

Definition 2 *A legal agent strategy σ is a best-effort solution for φ in $\mathcal{D}$ iff there is no legal agent strategy σ' such that $\sigma' >_{\varphi|\mathcal{D}} \sigma$.*

It is worth noting that, although we adapt the definitions of best-effort solutions from [5], the key difference is that, in our framework, best-effort solutions are required to be legal agent strategies, i.e., to always satisfy agent action preconditions. However, in the best-effort synthesis setting defined in [5], since the environment is specified as an LTL/LTL$_f$ formula, there are no such preconditions for the agent to follow.

Definition 3 *The problem of LTL$_f$ best-effort synthesis in nondeterministic planning domains is defined as a pair $\mathcal{P} = (\mathcal{D}, \varphi)$, where $\mathcal{D}$ is a nondeterministic planning domain and φ is an LTL$_f$ formula. Best-effort synthesis of $\mathcal{P}$ computes a best-effort solution for φ in $\mathcal{D}$.*

In particular, if the synthesis problem $\mathcal{P} = (\mathcal{D}, \varphi)$ admits a strong solution for φ in $\mathcal{D}$, then computing a best-effort solution coincides with computing a strong solution. This observation is analogous to the one for LTL$_f$ best-effort synthesis in [5], where the environment is specified in LTL/LTL$_f$.

Proposition 1 *Let $\mathcal{P} = (\mathcal{D}, \varphi)$ be the defined synthesis problem that admits a strong solution for φ in $\mathcal{D}$ and σ an agent strategy. We have that σ is a best-effort solution for φ in $\mathcal{D}$ iff σ is a strong solution for φ in $\mathcal{D}$.*

In order to compute best-effort solutions, we first observe that a best-effort solution can be considered as a strategy σ such that for every history $h \in (2^{\mathcal{F}})^+$, i.e., a finite and legal sequence of states of a planning domain $\mathcal{D}$, that is consistent with σ, σ extends h to fulfill the goal whenever possible. Hence, one of the following possibilities holds: (*i*) if the agent can extend h to fulfill the goal regardless of how the environment reacts, then any extension of h following σ should ensure goal completion; (*ii*) if no extension of h allows the agent to fulfill the goal, then σ just behaves legally; (*iii*) if there are only some extensions of h, but not all, that allow the agent to fulfill the goal, then there should exist an extension of h following σ that fulfills the goal. Observe that (*iii*) guarantees that the agent does its best, assuming the environment might cooperate to help the agent fulfill its goal.

Consider an agent strategy σ. We now formalize these three possibilities by assigning a value to each history h consistent with σ, as in [5]. For a legal agent strategy σ and a history h that is consistent with σ, we use $\Gamma(h, \sigma)$ to denote the set of legal environment strategies γ in $\mathcal{D}$ such that h is a prefix of $\texttt{Play}(\sigma, \gamma)$. Furthermore, we denote by $\mathcal{H}_{\mathcal{D}}(\sigma)$ the set of legal histories h such that $\Gamma(h, \sigma)$ is nonempty, i.e., the set of legal histories of $\mathcal{D}$ that are consistent with σ with some legal environment strategy γ. We define the value of h that is consistent with a legal agent strategy σ as follows:

- $val(\sigma, h) = +1$ (winning), if $\texttt{Play}(\sigma, \gamma) \models_{\mathcal{D}} \varphi$ for every $\gamma \in \Gamma(h, \sigma)$;
- $val(\sigma, h) = -1$ (losing), if $\texttt{Play}(\sigma, \gamma) \not\models_{\mathcal{D}} \varphi$ for every $\gamma \in \Gamma(h, \sigma)$;
- $val(\sigma, h) = 0$ (pending), otherwise.

Let $val(h)$ be the maximal value of $val(\sigma, h)$, considering all the strategies σ such that $h \in \mathcal{H}_{\mathcal{D}}(\sigma)$[3]. The following theorem states that a best-effort solution σ guarantees to maximize the value of every history h that is consistent with σ.

Theorem 2 (Maximality Condition) *An agent strategy σ is a best-effort solution for φ in $\mathcal{D}$ iff for every $h \in \mathcal{H}_{\mathcal{D}}(\sigma)$ it holds that $val(\sigma, h) = val(h)$.*

While classical synthesis and planning settings (see e.g. [27, 20, 18]) first require checking the realizability of the problem, i.e., the existence of a solution, in the case of best-effort synthesis realizability is trivial, as there always exists a best-effort solution.

Theorem 3 *Let $\mathcal{P} = (\mathcal{D}, \varphi)$ be a problem of* LTL$_f$ *best-effort synthesis in nondeterministic planning domains. There always exists a best-effort solution σ for φ in $\mathcal{D}$.*

In order to prove Theorem 3, we show that we can always construct a strategy $\hat{\sigma}$ that satisfies the Maximality Condition. To do so, we construct a chain of legal agent strategies $\sigma_0, \sigma_1, \ldots$ by selecting for each $i \geq 0$, a history $h \in \mathcal{H}_{\mathcal{D}}(\sigma_i)$ that has not yet been marked as stabilized, i.e., such that $val(\sigma_i, h) < val(h)$, and maximizing the value $val(\sigma_{i+1}, h)$. σ_{i+1} is constructed by selecting a strategy σ such that $val(\sigma, h) = val(h)$ and setting $\sigma_{i+1} = \sigma_i[h \leftarrow \sigma]$, where $\sigma_i[h \leftarrow \sigma]$ denotes the strategy that agrees with σ_i everywhere, except in h and all its extensions where it agrees with σ. $\hat{\sigma}$ is the point-wise limit of the chain of strategies, i.e., $\hat{\sigma} = lim_i\ \sigma_i$.

4 Synthesizing Best-Effort Solutions

We now provide an algorithm for LTL$_f$ best-effort synthesis in nondeterministic planning domains based on game-theoretic techniques. In particular, we use DFA games for adversarial/cooperative reachability and safety [20, 5, 31] to capture the environment being adversarial/cooperative and agent always following its action preconditions, respectively.

4.1 DFA Games

A *DFA game* is a pair $\mathcal{A} = (\mathcal{T}, Acc)$, where $\mathcal{T}$ is a deterministic transition system acting as the *game arena* and $Acc \subseteq Q^\omega$ is the acceptance condition. Specifically, $\mathcal{T} = (Act \times React, Q, q_0, \varrho)$ is a *deterministic transition system*, where $Act \times React$ is the alphabet, in which Act and $React$ are two disjoint sets of variables under the control of the agent and the environment, respectively; Q is a finite set of states; $q_0 \in Q$ is the initial state; ϱ: $Q \times Act \times React \to Q$ is the transition function. Given an infinite word $\pi = \pi_0\pi_1 \ldots \in (Act \times React)^\omega$, the *run* $\rho = \mathsf{Run}(\pi, \mathcal{T})$ of π on $\mathcal{T}$ is an infinite sequence $\rho = q_0q_1 \ldots \in Q^\omega$ such that q_0 is the initial state of $\mathcal{T}$ and $q_{i+1} = \varrho(q_i, \pi_i)$ for every $i \geq 0$. Analogously, we denote by $\rho^{k+1} = \mathsf{Run}(\pi^k, \mathcal{T})$ the finite sequence $\rho^{k+1} = q_0q_1 \ldots q_{k+1}$ obtained from running π^k on $\mathcal{T}$. A run ρ is *accepting* if $\rho \in Acc$. We denote by $\mathcal{L}(\mathcal{A})$ the set of words π accepted by $\mathcal{A}$, i.e., such that $\rho = \mathsf{Run}(\pi, \mathcal{T})$ is accepting. In this work, we specifically consider the following acceptance conditions:

- *Reachability*. Given a set $R \subseteq Q$, $\texttt{Reach}(R) = \{q_0q_1 \ldots \in Q^\omega \mid \exists k \geq 0.q_k \in R\}$, i.e., a state in R is visited at least once.
- *Safety*. Given a set $S \subseteq Q$, $\texttt{Safe}(S) = \{q_0q_1 \ldots \in Q^\omega \mid \forall k \geq 0.q_k \in S\}$, i.e., only states in S are visited.
- *Safety-Reachability*. Given two sets $S, R \subseteq Q$, $\texttt{SafeReach}(S, R) = \{q_0q_1 \ldots \in Q^\omega \mid \exists \ell \geq 0.q_\ell \in R \text{ and } \forall 0 \leq k \leq \ell.q_k \in S\}$, i.e., a state in R is visited at least once and until then only states in S are visited.

Notably, a deterministic transition system with reachability acceptance condition defines a deterministic finite state automaton (DFA).

An *agent strategy* on a DFA game is a function $\kappa : Q \to Act$.[4] Given an agent strategy κ, a sequence of environment reactions $\vec{r} = r_0r_1 \ldots \in React^\omega$, and a transition system $\mathcal{T}$, we denote by $\mathsf{Run}(\kappa, \vec{r}, \mathcal{T})$, the unique sequence $q_0q_1 \ldots \in Q^\omega$ on $\mathcal{T}$ *induced* by (or *consistent with*) κ and $\vec{r}$ as follows: q_0 is the initial state of $\mathcal{T}$, and for every $i \geq 0$, $q_{i+1} = \varrho(q_i, a_i, r_i)$, where $a_i = \kappa(q_i)$.

An agent strategy κ is *winning* in the game $\mathcal{A} = (\mathcal{T}, Acc)$, if for every sequence of environment reactions $\vec{r} \in React^\omega$, it holds that $\mathsf{Run}(\kappa, \vec{r}, \mathcal{T}) \in Acc$. In *DFA games*, $q \in Q$ is a *winning* state, if the agent has a winning strategy in the game $\mathcal{A}' = (\mathcal{T}', Acc)$, where $\mathcal{T}' = (Act \times React, Q, q, \delta)$, i.e., on the same structure but with a new initial state q. We denote by $W_{adv}(\mathcal{T}, Acc)$ (sometimes W_{adv} when $\mathcal{T}$ and Acc are clear from the context) the set of all agent winning states. Intuitively, W_{adv} represents the "agent winning region", from which the agent can win the game, no matter how the environment behaves. A positional strategy that is winning from every state in the winning region is called *uniform winning*.

Similarly, an agent strategy κ is *cooperatively winning* in a game $\mathcal{A} = (\mathcal{T}, Acc)$ if there exists a sequence of environment reactions $\vec{r} \in React^\omega$ such that $\mathsf{Run}(\kappa, \vec{r}, \mathcal{T}) \in Acc$. Hence, $q \in Q$ is a *cooperatively winning state*, if the agent has a cooperatively winning strategy in the game $\mathcal{A}' = (\mathcal{T}', Acc)$, where $\mathcal{T}' = (Act \times React, Q, q, \delta)$. By $W_{coop}(\mathcal{T}, Acc)$ (sometimes W_{coop}), we denote the set of all the agent cooperatively winning states. A positional strategy that is winning from every state in the cooperatively winning region is called *uniform cooperatively winning*.

4.2 Synthesis Technique

The key idea of our game-theoretic synthesis approach is reducing the synthesis problem to DFA games, where the game arena is obtained by suitably composing the nondeterministic planning domain and the DFA of the LTL$_f$ formula. We first show how to transform the nondeterministic planning domain $\mathcal{D}$ into a deterministic transition system, which is then composed with the DFA of the LTL$_f$ formula to obtain the game arena.

[3] We consider $val(h)$ only if there exists at least one agent strategy σ such that $h \in \mathcal{H}_{\mathcal{D}}(\sigma)$.

[4] For the DFA games considered in this paper, it is sufficient to define strategies in this form, which are called *positional strategies*. More general forms of strategies, mapping histories of the game to agent actions/environment reactions, are only needed for more sophisticated games [33].

In order to transform the domain $\mathcal{D}$ to a deterministic transition system, we need to drop the preconditions of both agent actions and environment reactions and transform the partial transition function into a total transition function. To that end, we introduce two new states, s^{ag}_{err} and s^{env}_{err}, indicating that the agent and the environment violate their respective preconditions. Formally, given a nondeterministic planning domain $\mathcal{D} = (2^{\mathcal{F}}, s_0, Act, React, \alpha, \beta, \delta)$, the corresponding deterministic transition system $\mathcal{D}_+ = (Act \times React, 2^{\mathcal{F}} \cup \{s^{ag}_{err}, s^{env}_{err}\}, s_0, \delta')$ is constructed as follows:

- $Act \times React$ is the alphabet;
- $2^{\mathcal{F}} \cup \{s^{ag}_{err} \cup s^{env}_{err}\}$ is the state space, where s^{ag}_{err} is the agent error state, and s^{env}_{err} is the environment error state;
- $s_0 \in 2^{\mathcal{F}}$ is the initial state;
- $\delta' : (2^{\mathcal{F}} \cup \{s^{ag}_{err}, s^{env}_{err}\}) \times Act \times React \to 2^{\mathcal{F}} \cup \{s^{ag}_{err}, s^{env}_{err}\}$ is such that if $s \in \{s^{ag}_{err}, s^{env}_{err}\}$ then $\delta'(s, a, r) = s$, otherwise

$$\delta'(s,a,r) = \begin{cases} \delta(s,a,r) & \text{if } a \in \alpha(s) \text{ and } r \in \beta(s,a) \\ s^{ag}_{err} & \text{if } a \notin \alpha(s) \\ s^{env}_{err} & \text{if } a \in \alpha(s) \text{ and } r \notin \beta(s,a). \end{cases}$$

Intuitively, the transitions of $\mathcal{D}_+$ are the same as in $\mathcal{D}$, except that the transitions move to s^{ag}_{err} and s^{env}_{err} when agent action or environment reaction preconditions are violated, respectively. Once an error state is reached, $\mathcal{D}_+$ just keeps looping there.

We now construct the game arena, on which the agent and the environment control actions and reactions, respectively, through an ad-hoc composition of $\mathcal{D}_+$ and (the transition system of) the DFA of the agent goal φ. Given an LTL$_f$ formula φ over $\mathcal{F}$, we first obtain its corresponding DFA $\mathcal{A}_\varphi = (\mathcal{T}_\varphi, \texttt{Reach}(R_\varphi))$, where $\mathcal{T}_\varphi = (2^{\mathcal{F}}, Q, q_0, \varrho)$ is the transition system and $R_\varphi \subseteq Q$ is the set of final states. Note that the transitions of $\mathcal{A}_\varphi$ are defined wrt fluents $\mathcal{F}$, i.e., the transition function of $\mathcal{T}_\varphi$ is in the form of $\varrho : Q \times 2^{\mathcal{F}} \to Q$. But the transitions of $\mathcal{D}_+$ are defined wrt agent actions and environment reactions. Hence, the composition of $\mathcal{D}_+$ and $\mathcal{T}_\varphi$ needs to suitably synchronize the transitions of both. To that end, we map the fluent evaluations in the states of $\mathcal{D}_+$ to the transition conditions that follow the transition function ϱ.

Formally, the composition $\mathcal{T} = \mathcal{D}_+ \circ \mathcal{T}_\varphi$ is such that $\mathcal{T} = (Act \times React, (2^{\mathcal{F}} \cup \{s^{ag}_{err}, s^{env}_{err}\}) \times Q, (s_0, \varrho(q_0, s_0)), \partial)$, where:

- $Act \times React$ is the alphabet;
- $(2^{\mathcal{F}} \cup \{s^{ag}_{err}, s^{env}_{err}\}) \times Q$ is the set of states;
- $(s_0, \varrho(q_0, s_0))$ is the initial state;
- $\partial : ((2^{\mathcal{F}} \cup \{s^{ag}_{err}, s^{env}_{err}\}) \times Q) \times Act \times React \to ((2^{\mathcal{F}} \cup \{s^{ag}_{err}, s^{env}_{err}\}) \times Q)$ is the transition function such that:

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

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

Intuitively, $\mathcal{T}$ is a deterministic transition system that simultaneously retains the state of the domain $\mathcal{D}$ and the progress in the DFA $\mathcal{A}_\varphi$ in satisfying the LTL$_f$ goal φ. In particular, note that the function ∂ is defined such that if the $\mathcal{T}$ reaches an environment/agent error state, $\mathcal{T}$ cannot reach any other state afterwards. In fact, a positional strategy κ on $\mathcal{T} = \mathcal{D}_+ \circ \mathcal{T}_\varphi$ also induces a strategy $\sigma' : React^* \to Act$ in the planning domain $\mathcal{D}$.

Definition 4 *Let $\kappa : Q \to Act$ be a positional strategy on $\mathcal{T} = \mathcal{D}_+ \circ \mathcal{T}_\varphi$. κ induces an agent strategy $\sigma' : React^* \to Act$: $\sigma'(\vec{r}_\epsilon) = \kappa(t_0)$, where t_0 is the initial state of $\mathcal{T}$; for every $i \geq 0$, $\sigma'(\vec{r}^{\,i}) = \kappa(t)$, where $\vec{r}^{\,i} = r_0 \dots r_i$ and t is the last state in the finite sequence $\rho^{i+1} = \mathsf{Run}(\pi^i, \mathcal{T})$ with $\pi^i = (\sigma'(\vec{r}_\epsilon) \cup r_0)(\sigma'(\vec{r}^{\,0}) \cup r_1) \dots (\sigma'(\vec{r}^{\,i-1}) \cup r_i)$.*

Indeed, with σ' we can also obtain the equivalent strategy $\sigma : (2^{\mathcal{F}})^+ \to Act$ (c.f. Section 3.1) in $\mathcal{D}$. Sometimes, for simplicity, we directly say that positional strategy κ induces strategy $\sigma : (2^{\mathcal{F}})^+ \to Act$ (rather than saying that κ induces strategy $\sigma' : React^* \to Act$ which is equivalent to a strategy $\sigma : (2^{\mathcal{F}})^+ \to Act$).

Given a synthesis problem $\mathcal{P} = (\mathcal{D}, \varphi)$, we now detail how to compute a best-effort solution for φ in $\mathcal{D}$ by reducing to suitable DFA games on the game arena $\mathcal{T} = (Act \times React, (2^{\mathcal{F}} \cup \{s^{ag}_{err}, s^{env}_{err}\}) \times Q, (s_0, \varrho(q_0, s_0)), \partial)$ constructed as above. In the following, we denote by R'_φ the set of states in $\mathcal{T}$ by lifting the final states R_φ in $\mathcal{T}_\varphi$ to $\mathcal{T}$. Hence, $R'_\varphi = \{(s, q) \mid q \in R_\varphi\}$. Moreover, we denote by S^{ag}_{err} and S^{env}_{err} the sets of states by lifting the agent error state and the environment error state in $\mathcal{D}_+$ to $\mathcal{T}$, respectively. Therefore, S^{ag}_{err} and S^{env}_{err} indicate the agent and the environment violate their respective preconditions. Hence $S^{ag}_{err} = \{(s, q) \mid s = s^{ag}_{err}\}$ and $S^{env}_{err} = \{(s, q) \mid s = s^{env}_{err}\}$). Sometimes, we write $\neg S^{ag}_{err}$ as an abbreviation for $((2^{\mathcal{F}} \cup \{s^{ag}_{err}, s^{env}_{err}\}) \times Q) \setminus S^{ag}_{err}$ to indicate that the agent has not yet violated its precondition. Similarly for $\neg S^{env}_{err}$.

Algorithm 1. Given problem $\mathcal{P} = (\mathcal{D}, \varphi)$ of LTL$_f$ best-effort synthesis in nondeterministic planning domains, proceed as follows:

1. Construct the DFA $\mathcal{A}_\varphi = (\mathcal{T}_\varphi, \texttt{Reach}(R_\varphi))$ of agent goal φ and the transition system $\mathcal{D}_+$ of domain $\mathcal{D}$.
2. Construct arena $\mathcal{T} = (\mathcal{D}_+ \circ \mathcal{T}_\varphi)$ by composition.
3. In the DFA game $(\mathcal{T}, \texttt{Reach}(\neg S^{ag}_{err} \cap (S^{env}_{err} \cup R'_\varphi)))$, compute a positional uniform winning strategy κ_{adv}. Let W_{adv} be the winning region.
4. In the DFA game $(\mathcal{T}, \texttt{Reach}(\neg S^{ag}_{err} \cap \neg S^{env}_{err} \cap R'_\varphi))$, compute a positional uniform cooperatively winning strategy κ_{coop}. Let W_{coop} be the cooperatively winning region.
5. Construct positional strategy κ from κ_{adv} and κ_{coop} as follows:

$$\kappa(s,q) = \begin{cases} \kappa_{adv}(s,q) & \text{if } (s,q) \in W_{adv} \\ \kappa_{coop}(s,q) & \text{if } (s,q) \in W_{coop} \setminus W_{adv} \\ \text{any } a \in \alpha(s) & \text{otherwise} \end{cases}$$

   where $\alpha(s) = Act$, if $s \in \{s^{ag}_{err}, s^{env}_{err}\}$, such that the agent can perform any action at an error state.
6. **Return** the (best-effort) solution σ for φ in $\mathcal{D}$ *induced* by κ.

Due to the requirement of *existence of agent action*, at every state $s \in 2^{\mathcal{F}}$, there always exists an agent action $a \in \alpha(s)$ (possibly nop).

It is worth noting that Algorithm 1 also allows us to check whether the computed best-effort solution is a strong solution as well. This is indicated by the value of $val(\sigma, s_0)$. More specifically, we have that the computed solution σ is a strong solution if $t_0 \in W_{adv}$, where t_0 is the initial state of $\mathcal{T} = \mathcal{D}_+ \circ \mathcal{T}_\varphi$.

Theorem 4 (Correctness) *Let $\mathcal{D}$ be a nondeterministic planning domain, φ an LTL$_f$ goal, and σ the agent strategy returned by Algorithm 1. Then, σ is a best-effort solution for φ in $\mathcal{D}$.*

To prove this theorem, we make use of several intermediate results, which are given in the next section.

4.3 Correctness

In order to prove Theorem 4, we first show the connection between strong and cooperative solutions for the agent goal in a planning domain to winning and cooperatively winning strategies in the corresponding DFA games, respectively.

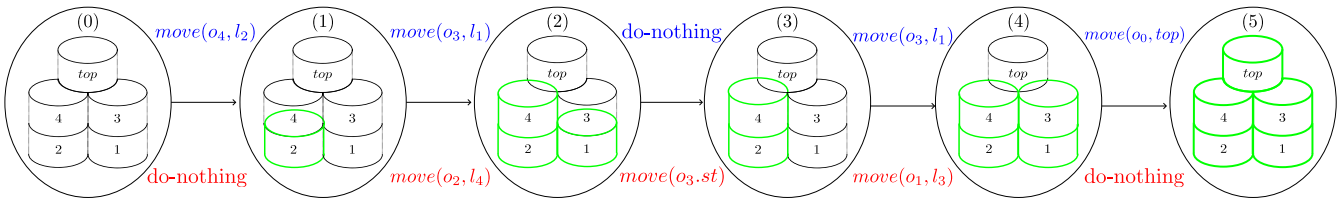

Figure 2: An execution example of a best-effort solution for the arch-building task described in Section 1. Robot actions and human reactions are shown in blue and red, respectively.

Theorem 5 *Let $\mathcal{D}$ be a planning domain, φ an* LTL$_f$ *goal, and $\mathcal{T} = \mathcal{D}_+ \circ \mathcal{T}_\varphi$ the constructed game arena. The following two hold:*

(a) *there is a strong solution for φ in $\mathcal{D}$ iff the agent has a winning strategy in $(\mathcal{T}, \texttt{SafeReach}(\neg S^{ag}_{err}, S^{env}_{err} \cup R'_\varphi))$.*

(b) *there is a cooperative solution for φ in $\mathcal{D}$ iff the agent has a cooperatively winning strategy in $(\mathcal{T}, \texttt{SafeReach}(\neg S^{ag}_{err} \cap \neg S^{env}_{err}, R'_\varphi))$.*

To prove Theorem 5, we first observe that a strong solution for φ in $\mathcal{D}$ guarantees that the agent always follows its action precondition (i.e., never visits S^{ag}_{err}) *and* either forces the environment to violate its reaction precondition or eventually satisfies φ (i.e., eventually reaches $S^{env}_{err} \cup R'_\varphi$). Hence, the agent's goal is to satisfy

$$\Box(\neg S^{ag}_{err}) \wedge \Diamond(S^{env}_{err} \vee R'_\varphi)$$

(For simplicity, we abuse notations and consider sets of states as atomic propositions). Therefore, the reduced game for computing a strong solution is $(\mathcal{T}, \texttt{SafeReach}(\neg S^{ag}_{err}, S^{env}_{err} \cup R'_\varphi))$. Similarly for proving (*b*).

Next, we show that the safety-reachability games can be solved by reducing to pure reachability games, the correctness of which can be shown by construction.

Lemma 1 *Let $\mathcal{T}$, S^{ag}_{err}, S^{env}_{err} and R'_φ be as defined above. Then, for every run ρ on $\mathcal{T}$:*

- *ρ is an accepting run of $\texttt{SafeReach}(\neg S^{ag}_{err}, S^{env}_{err} \cup R'_\varphi)$ iff ρ is an accepting run of $\texttt{Reach}(\neg S^{ag}_{err} \cap (S^{env}_{err} \cup R'_\varphi))$.*
- *ρ is an accepting run of $\texttt{SafeReach}(\neg S^{ag}_{err} \cap \neg S^{env}_{err}, R'_\varphi)$ iff ρ is an accepting run of $\texttt{Reach}(\neg S^{ag}_{err} \cap S^{env}_{err} \cap R'_\varphi)$.*

By Theorem 5 and Lemma 1, the computed winning strategy κ_{adv} for $\texttt{Reach}(\neg S^{ag}_{err} \cap (S^{env}_{err} \cup R'_\varphi))$ and the cooperative winning strategy κ_{coop} for $\texttt{Reach}(\neg S^{ag}_{err} \cap S^{env}_{err} \cap R'_\varphi)$ indeed correspond to a strong solution and a cooperative solution for φ in $\mathcal{D}$, respectively.

We now show that the computed final strategy σ by combing κ_{adv} and κ_{coop} is a best-effort solution for φ in $\mathcal{D}$. We prove this through Theorem 2 by showing that σ satisfies the Maximality Condition.

Theorem 6 *Let σ be the strategy returned by Algorithm 1. For every $h \in \mathcal{H}_\mathcal{D}(\sigma)$ it holds $val(\sigma, h) = val(h)$.*

To prove Theorem 6, first observe that every sequence of states ρ on $\mathcal{T}$ that does not end at an environment/agent error state corresponds to a legal history h on $\mathcal{D}$. The strategy σ returned by Algorithm 1 satisfies that, for every $h \in \mathcal{H}_\mathcal{D}(\sigma)$, one of the following holds: (*i*) the corresponding run ρ on $\mathcal{T}$ ends at a state $s \in W_{adv}$ hence h can surely be extended to satisfy φ in $\mathcal{D}$ using σ (i.e., from h the agent has a strong solution); (*ii*) the corresponding run ρ on $\mathcal{T}$ ends at a state $s \in W_{coop} \backslash W_{adv}$ hence h can possibly be extended to satisfy φ in $\mathcal{D}$ using σ, should the environment cooperate (i.e., from h the agent has a cooperative solution); (*iii*) the corresponding run ρ on $\mathcal{T}$ ends at a state $s \notin W_{coop} \cup W_{adv}$ hence h cannot be extended to satisfy φ in $\mathcal{D}$ (i.e., from h the agent has neither a strong nor a cooperative solution). Moreover, σ is guaranteed to be a legal strategy since the agent can only perform legal actions.

Finally, we can prove Theorem 4 (the correctness of Algorithm 1 in Section 4.2) as an immediate result of Theorems 6 and 2.

4.4 Computational Complexity

The computational complexity of LTL$_f$ best-effort synthesis in nondeterministic planning domains is the following:

Theorem 7 LTL$_f$ *best-effort synthesis in nondeterministic planning domains is:*

- *2EXPTIME-complete in the size of the* LTL$_f$ *goal;*
- *EXPTIME-complete in the size of the domain.*

Regarding the complexity in the size of the LTL$_f$ goal, the hardness comes from LTL$_f$ synthesis, and the membership comes from the LTL$_f$-to-DFA construction (double-exponential in the size of the LTL$_f$ formula). Hence, if we consider simple reachability goals in the form of $\varphi = \Diamond(R)$, where R is a propositional formula over the fluents $\mathcal{F}$ of the planning domain, the problem is just polynomial in the size of $\varphi = \Diamond(R)$ since the DFA can be constructed in polynomial time. Regarding the complexity in the size of the domain, the hardness comes from planning itself [18], and the membership comes from the construction of the deterministic transition system from the planning domain (single-exponential in the number of fluents).

An interesting observation from Theorem 7 is that best-effort synthesis provides an efficient alternative approach to planning in nondeterministic domains for both LTL$_f$ and reachability goals. In fact, instead of looking for a strong solution, which may not exist, one can look for a best-effort solution, which, on the one hand, always exists and on the other hand is directly a strong solution if the problem admits one.

Running Example (cont.). We present the computed best-effort solution produced by our synthesis algorithm for the running example in Section 1. We represent the arch-building task as a suitable LTL$_f$ formula and the dynamics of the interactions between the robot and the human (seen as the agent and the environment, respectively) as a nondeterministic planning domain. Figure 2 presents a (simplified) execution example of a best-effort solution to building the arch. Note that at the initial state, all blocks are placed in storage (state 0). Then the robot successfully placed the block o_4 at the location l_2 without any interference from the human (state 1). Next, the robot placed the block o_3 at l_1, and the human helped build the arch by placing o_2 at l_4 (state 2). Seeing the human being cooperative, the robot decides to be lazy, hence not doing anything. However, the human now punishes the robot for being lazy by undoing what the robot has done, thus removing o_3 back to storage (state 3). The robot now realizes that the human is not always cooperative, so it proceeds by placing o_3 back at l_1, and the human cooperates by placing o_1 at l_3 (state 4). Finally, the robot can build the arch by placing o_0 at the top, with

no interference from the human (state 5). This example well illustrates the basic characteristic of best-effort solutions: handling both adversarial and cooperative environment behaviors, even if the environment switches back and forth between behaving adversarially and cooperatively instead of always being adversarial or cooperative.

5 Implementation and Empirical Evaluation

We implemented our algorithm for LTL$_f$ best-effort synthesis in planning domains in a tool called *BeSyftP*, leveraging the symbolic LTL$_f$ synthesis framework [32] that is integrated in all state-of-the-art LTL$_f$ synthesis tools [7, 15]. The explicit-state DFAs of LTL$_f$ formulas are constructed by LYDIA [16], the overall best-performing tool for LTL$_f$-to-DFA construction. *BeSyftP* uses the *symbolic* DFA encoding proposed in [32] to represent the DFAs symbolically and integrates the symbolic encoding of a planning domain (specified in a variant of PDDL) proposed in [26] for symbolic domain representation. Both symbolic DFAs and planning domains are represented in Binary Decision Diagrams (BDDs) [8], with the BDD library CUDD-3.0.0 [29]. Following [32], *BeSyftP* constructs and solves symbolic DFA games using Boolean operations provided by CUDD, such as quantification, negation and conjunction. In particular, positional strategies are obtained with Boolean synthesis [22]. Finally, best-effort strategies are computed by applying suitable Boolean operations to the obtained positional winning strategy and cooperatively winning strategy.

Experimental Comparison. To show the efficiency of our technique for LTL$_f$ best-effort synthesis in nondeterministic planning domains, we want to compare it with the approach of reducing to LTL$_f$ best-effort synthesis, by re-expressing the domain as an LTL$_f$ formula. More specifically, we adapted the domain-to-LTL$_f$ translation in [26] to obtain the LTL$_f$ formula $\mathcal{E}$ of the nondeterministic planning domain. Next, we can utilize the LTL$_f$ best-effort synthesis technique from [5] to solve the problem of φ (agent goal) under $\mathcal{E}$.

We also evaluated the performance of *BeSyftP* considering the overhead of computing best-effort solutions wrt computing strong and cooperative solutions. Therefore, we also implemented the adversarial and cooperative synthesis approaches for computing strong and cooperative solutions in *AdvSyftP* and *CoopSyftP*, respectively.

Benchmarks. For benchmarks, we consider nondeterministic planning domains as the same one in the running example of building-an-arch described in Section 1. In order to have scalable benchmarks, we consider agent goals as putting O objects at L locations in line, hence the difficulty of the benchmarks increases as either O or L increases. Note that for all these benchmarks, the agent does not have a strong solution to fulfill the goal, i.e., it needs a best-effort solution. Our benchmarks consider at most 8 objects ($1 \leq |O| \leq 8$) and 1000 locations ($1 \leq |L| \leq 1000$).

Experiment Setup. All experiments were run on a laptop with an operating system 64-bit Ubuntu 20.04, 3.6 GHz CPU, and 12 GB of memory. Time out was set to 1200 seconds.

Experimental Results. As shown in Figure 3 (for better vision, we only plot the results on instances with up to 10 locations), utilizing the LTL$_f$ best-effort synthesis approach from [5] can solve the instances with one object ($|O| = 1$) up to 6 locations ($|L| \leq 6$).[5] Instead, *BeSyftP* can solve all the instances with $|O| = 1$ up to 10 locations and more (up to 1000), showing much greater scalability.[6]

[5] Notice that if we increase the number of objects to 2, the synthesis approach from [5] can still solve the corresponding instances. But if we increase the number of objects to 3, it times out immediately.

[6] If we increase $|O|$ to 6, our approach can only handle instances with $|L|$ up to 10. This is because $|L|$ only affects the size of the planning domain, on which the synthesis complexity is EXPTIME. Instead, $|O|$ also affects the LTL$_f$ goal, on which the complexity is 2EXPTIME.

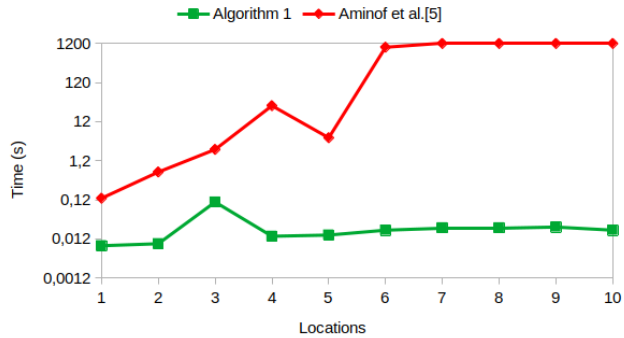

Figure 3: Comparison of *BeSyftP* and the algorithm from Aminof et al. [5] on benchmarks with $|O| = 1$ and $1 \leq |L| \leq 10$ (in log scale).

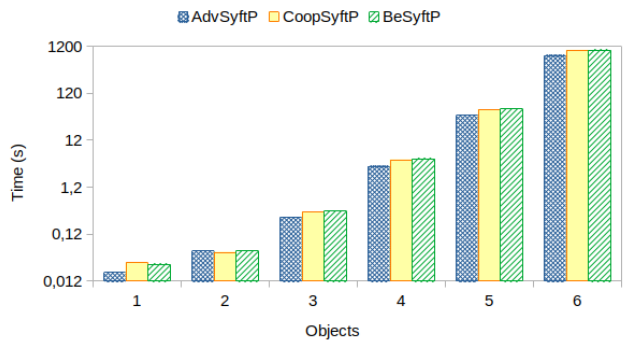

Figure 4: Comparison of *BeSyftP*, *AdvSyftP* and *CoopSyftP* on benchmarks with $|L| = 10$ and $1 \leq |O| \leq 6$ (in log scale).

Figure 4 shows the results of comparing the performance of computing best-effort solutions (*BeSyftP*) with that of computing strong (*AdvSyftP*) and cooperative (*CoopSyftP*) ones. Note that, since all benchmarks are *unrealizable*, *AdvSyftP* terminates earlier than both *CoopSyftP* and *BeSyftP* (with $20\% \sim 25\%$ less time cost). Interestingly, *BeSyftP* takes roughly the same time as *CoopSyftP* (with $10\% \sim 15\%$ more time cost). This shows that, although computing a best-effort solution requires solving two games and combining the computed adversarial and cooperative solutions, *BeSyftP* requires only a small overhead with respect to both *AdvSyftP* and *CoopSyftP*.

6 Conclusion

In this paper, we have developed a framework for LTL$_f$ best-effort synthesis in nondeterministic (adversarial) planning domains, and a solution technique that admits a scalable symbolic implementation. The approach presented here can be extended to different forms of goal specification. In particular, we can expect a further complexity reduction if we express goals in pure-past temporal logics [14]. Also, the game-based techniques adopted here could possibly be extended to handle best-effort synthesis under multiple environment specifications [13, 1, 2]. Another promising extension is considering maximally permissive strategies [30], which allows the agent to choose a best-effort solution during execution instead of committing to a single solution beforehand We leave this for future work.

Acknowledgments

This work has been partially supported by the ERC-ADG WhiteMech (No. 834228), the EU ICT-48 2020 project TAILOR (No. 952215), the PRIN project RIPER (No. 20203FFYLK), and the PNRR MUR project FAIR (No. PE0000013). This work has been carried out while Gianmarco Parretti was enrolled in the Italian National Doctorate on Artificial Intelligence run by Sapienza University of Rome.

References

[1] Benjamin Aminof, Giuseppe De Giacomo, Alessio Lomuscio, Aniello Murano, and Sasha Rubin, 'Synthesizing strategies under expected and exceptional environment behaviors', in *IJCAI*, pp. 1674–1680, (2020).

[2] Benjamin Aminof, Giuseppe De Giacomo, Alessio Lomuscio, Aniello Murano, and Sasha Rubin, 'Synthesizing best-effort strategies under multiple environment specifications', in *KR*, pp. 42–51, (2021).

[3] Benjamin Aminof, Giuseppe De Giacomo, Aniello Murano, and Sasha Rubin, 'Planning and synthesis under assumptions', *arXiv*, (2018).

[4] Benjamin Aminof, Giuseppe De Giacomo, Aniello Murano, and Sasha Rubin, 'Planning under LTL environment specifications', in *ICAPS*, pp. 31–39, (2019).

[5] Benjamin Aminof, Giuseppe De Giacomo, and Sasha Rubin, 'Best-Effort Synthesis: Doing Your Best Is Not Harder Than Giving Up', in *IJCAI*, pp. 1766–1772, (2021).

[6] Christel Baier, Joost-Pieter Katoen, and Kim Guldstrand Larsen, *Principles of Model Checking*, MIT Press, 2008.

[7] Suguman Bansal, Yong Li, Lucas M. Tabajara, and Moshe Y. Vardi, 'Hybrid compositional reasoning for reactive synthesis from finite-horizon specifications', in *AAAI*, pp. 9766–9774, (2020).

[8] Randal E. Bryant, 'Symbolic Boolean Manipulation with Ordered Binary-Decision Diagrams', *ACM Comput. Surv.*, **24**(3), 293–318, (1992).

[9] Alberto Camacho, Meghyn Bienvenu, and Sheila A McIlraith, 'Towards a unified view of ai planning and reactive synthesis', in *ICAPS*, pp. 58–67, (2019).

[10] Alberto Camacho and Sheila A. McIlraith, 'Strong fully observable non-deterministic planning with LTL and LTL$_f$ goals', in *IJCAI*, pp. 5523–5531, (2019).

[11] Alessandro Cimatti, Marco Pistore, Marco Roveri, and Paolo Traverso, 'Weak, strong, and strong cyclic planning via symbolic model checking.', *AIJ*, **1–2**(147), 35–84, (2003).

[12] Alessandro Cimatti, Marco Roveri, and Paolo Traverso, 'Strong planning in non-deterministic domains via model checking', in *AIPS*, pp. 36–43, (1998).

[13] Daniel Alfredo Ciolek, Nicolás D'Ippolito, Alberto Pozanco, and Sebastian Sardiña, 'Multi-tier automated planning for adaptive behavior', in *ICAPS*, pp. 66–74, (2020).

[14] Giuseppe De Giacomo, Antonio Di Stasio, Francesco Fuggitti, and Sasha Rubin, 'Pure-past linear temporal and dynamic logic on finite traces', in *IJCAI*, pp. 4959–4965, (2020).

[15] Giuseppe De Giacomo and Marco Favorito, 'Compositional approach to translate LTL$_f$/LDL$_f$ into deterministic finite automata', in *ICAPS*, pp. 122–130, (2021).

[16] Giuseppe De Giacomo and Marco Favorito, 'Lydia: A tool for compositional LTL$_f$ /LDL$_f$ synthesis', in *ICAPS*, pp. 122–130, (2021).

[17] Giuseppe De Giacomo and Yves Lespérance, 'The nondeterministic situation calculus', in *KR*, pp. 216–226, (2021).

[18] Giuseppe De Giacomo and Sasha Rubin, 'Automata-Theoretic Foundations of FOND Planning for LTLf and LDLf Goals.', in *IJCAI*, pp. 4729–4735, (2018).

[19] Giuseppe De Giacomo and Moshe Y. Vardi, 'Linear Temporal Logic and Linear Dynamic Logic on Finite Traces', in *IJCAI*, pp. 854–860, (2013).

[20] Giuseppe De Giacomo and Moshe Y. Vardi, 'Synthesis for LTL and LDL on Finite Traces', in *IJCAI*, pp. 1558–1564, (2015).

[21] Bernd Finkbeiner, 'Synthesis of Reactive Systems.', *Dependable Software Systems Eng.*, **45**, 72–98, (2016).

[22] Dror Fried, Lucas M. Tabajara, and Moshe Y. Vardi, 'BDD-based Boolean functional synthesis', in *CAV*, pp. 402–421, (2016).

[23] H. Geffner and B. Bonet, *A Concise Introduction to Models and Methods for Automated Planning*, 2013.

[24] Malik Ghallab, Dana S. Nau, and Paolo Traverso, *Automated planning - theory and practice*, 2004.

[25] Patrik Haslum, Nir Lipovetzky, Daniele Magazzeni, and Christian Muise, *An Introduction to the Planning Domain Definition Language*, 2019.

[26] Keliang He, Andrew M Wells, Lydia E Kavraki, and Moshe Y Vardi, 'Efficient Symbolic Reactive Synthesis for Finite-Horizon Tasks', in *ICRA*, pp. 8993–8999, (2019).

[27] A. Pnueli and R. Rosner, 'On the synthesis of a reactive module', in *POPL*, p. 179–190, (1989).

[28] Amir Pnueli, 'The temporal logic of programs', in *FOCS*, pp. 46–57, (1977).

[29] Fabio Somenzi, 'CUDD: CU Decision Diagram Package 3.0.0. Universiy of Colorado at Boulder', (2016).

[30] Shufang Zhu and Giuseppe De Giacomo, 'Synthesis of maximally permissive strategies for ltlf specifications', in *IJCAI*, pp. 2783–2789. ijcai.org, (2022).

[31] Shufang Zhu, Lucas M. Tabajara, Jianwen Li, Geguang Pu, and Moshe Y. Vardi, 'A symbolic approach to safety LTL synthesis', in *HVC*, pp. 147–162, (2017).

[32] Shufang Zhu, Lucas M. Tabajara, Jianwen Li, Geguang Pu, and Moshe Y. Vardi, 'Symbolic LTL$_f$ Synthesis', in *IJCAI*, pp. 1362–1369, (2017).

[33] Martin Zimmermann, Felix Klein, and Alexander Weinert, 'Infinite games', in *Lecture Notes (Chapters 1 and 2)*, (2016).