Do Your Best, but Don't Take Too Many Chances: LTL$_f$ Synthesis of Minimal Best-Effort Strategies in FOND Domains

Giuseppe De Giacomo[a,b], **Gianmarco Parretti** [b] **and Elisa Santini** [b]

[a]University of Oxford
[b]University of Rome "La Sapienza"

Abstract. Inspired by Joker strategies in games on graphs, we introduce *minimal best-effort* strategies for Linear Temporal Logic on Finite Traces (LTL$_f$) goals in *Fully Observable Nondeterministic* (FOND) domains and study their corresponding synthesis problem. Minimal best-effort strategies always exist and guarantee that, when a winning strategy does not exist: (*i*) the agent does its best to achieve its goal; (*ii*) it relies the least on the environment's cooperation. We present a game-theoretic algorithm to synthesize minimal best-effort strategies and prove its correctness as well as its optimality (wrt computational complexity). We implemented the algorithm and performed an experimental analysis on scalable benchmarks. The empirical results show that the computation of minimal best-effort strategies is quite efficient: it only requires a small overhead compared to standard best-effort strategies.

1 Introduction

In this paper, we consider a variant of (strong) planning in Fully Observable Nondeterministic (FOND) domains [10, 21], where goals are temporally extended and expressed in Linear Temporal Logic on finite traces (LTL$_f$) [13, 14, 12, 8]. Specifically, we focus on the case where a strong plan – here referred to as a *winning strategy*, as in the related field of reactive synthesis [14] – does not exist.

When a winning strategy – one that achieves the goal regardless of the environment's response – does not exist, the agent should at least avoid adopting strategies that preclude the possibility of achieving the goal. Best-effort strategies [1, 15] capture this intuition. More precisely, they embody the game-theoretic principle of rationality: a player should not adopt a strategy that is *dominated* by another, i.e., one that achieves the goal against fewer environmental responses than an available alternative.

Best-effort strategies have several notable properties: (*i*) they always exist; (*ii*) if a winning strategy exists, the best-effort strategies are exactly the winning strategies; (*iii*) they can be computed in 2EXPTIME, just like winning strategies (in fact, best-effort synthesis is 2EXPTIME-complete). However, unless a *dominant strategy* – a best-effort strategy that dominates all others – exists [2], there are generally many incomparable best-effort strategies, offering no principled basis for preferring one over another.

The key idea behind best-effort strategies is that, given a history the agent selects: (1) an action that enables it to force a win, if possible; otherwise, (2) an action that preserves the possibility of winning later, should the environment not behave adversarially now, i.e., if the environment *temporarily cooperates* with the agent. If there are multiple actions satisfying property (2), then any of them is equally good, leading to incomparable best-effort strategies.

Here, inspired by [30], we study the case in which, when the agent must rely on the environment's cooperation, c.f. (2), it selects a strategy with an additional requirement: *the goal can be achieved with the shortest possible sequence of cooperative environment responses*. In other words, an agent using one such a strategy – which we call *minimal best-effort* – relies the least on the environment's cooperation.

To define minimal best-effort strategies, we rely on Joker strategies introduced in the setting of two-player reachability games [30]. Joker strategies allow the protagonist, i.e., the agent, to play a Joker move, which also controls the response of the adversarial, i.e., the environment. This notion is relevant for this work since it provides a method to quantify reliance on environmental cooperation. In this paper, we show that Joker strategies do not correspond to best-effort strategies per se, but they can serve as the basis for defining minimal best-effort strategies, as discussed above. Subsequently, we present a game-theoretic technique for synthesizing minimal best-effort strategies that is correct and optimal in terms of worst-case complexity. We show that the complexity of synthesizing minimal best-effort strategies remains the same as that of synthesizing standard best-effort strategies, which, in turn, is the same as the complexity of synthesizing winning strategies. The synthesis technique involves alternating the computation of two least fixed points over the game arena that is generated by the Cartesian product of the FOND domain and the deterministic automaton corresponding to the LTL$_f$ goal. Notably, this technique can be implemented symbolically using the same kind of technology employed in Model Checking [3]. We implemented the technique in a symbolic solver, and using this solver, we show that, in practice, computing a minimal best-effort strategy brings a minimal overhead compared to computing a standard best-effort strategy.

2 Preliminaries

We consider specifications in Linear Temporal Logic on Finite Traces (LTL$_f$) [13]. LTL$_f$ is a formalism widely used in AI to specify, e.g., temporally extended goals [9, 12, 8] and state-trajectory constraints [5, 6] in planning. Given a set of atomic propositions (aka atoms) $\mathcal{AP}$, the LTL$_f$ formulas over $\mathcal{AP}$ are: $\varphi ::= p \mid \neg\varphi \mid \varphi_1 \wedge \varphi_2 \mid \bigcirc\varphi \mid \varphi_1 \,\mathcal{U}\, \varphi_2$, where $p \in \mathcal{AP}$. Symbols $\bigcirc$ and $\mathcal{U}$ denote the *strong next* and *until* temporal operator, respectively. Additional operators are defined as abbreviations and include: stan-

dard operators of propositional logic; $\Diamond\varphi = true\,\mathcal{U}\,\varphi$ (*eventually*); $\Box\varphi = \neg\Diamond\neg\varphi$ (*always*); and $\bullet\varphi = \neg\bigcirc\neg\varphi$ (*weak next*). The size of φ, denoted $|\varphi|$, is the number of subformulas in its abstract syntax tree.

LTL$_f$ formulas are interpreted over finite non-empty traces $\pi \in (2^{\mathcal{AP}})^+$ of propositional interpretations over $\mathcal{AP}$. We denote by $\pi_i \in 2^{\mathcal{AP}}$ the propositional interpretation in the i-th time step of π, with $i = 0, 1, \ldots, n$. Here $n = |\pi|$ is the length of the trace π. The empty trace is ε. Given an LTL$_f$ formula φ, a trace π, and an index i, the following inductive definition formalizes when π satisfies φ at i.

1. $\pi, i \models p$ iff $p \in \pi_i$, for $p \in \mathcal{AP}$;
2. $\pi, i \models \neg\varphi$ iff $\pi, i \not\models \varphi$;
3. $\pi, i \models \varphi_1 \wedge \varphi_2$ iff $\pi, i \models \varphi_1$ and $\pi, i \models \varphi_2$;
4. $\pi, i \models \bigcirc\varphi$ iff $i < |\pi| - 1$ and $\pi, i+1 \models \varphi$;
5. $\pi, i \models \varphi_1\,\mathcal{U}\,\varphi_2$ iff there exists j such that $i \leq j < |\pi|$ and $\pi, j \models \varphi_2$, and for every k such that $i \leq k < j$ we have $\pi, k \models \varphi_1$.

A finite trace π satisfies φ, written $\pi \models \varphi$, iff $\pi, 0 \models \varphi$.

Following [16], a *Fully Observable Nondeterministic (*FOND*) Domain* is a tuple $\mathcal{D} = (2^{\mathcal{F}}, s_0, Act, React, \alpha, \beta, \delta)$, where: $\mathcal{F}$ is a finite set of *fluents* and $2^{\mathcal{F}}$ is the *state space*; $s_0 \in 2^{\mathcal{F}}$ is the *initial domain state*; Act and $React$ are finite sets of *agent actions* and *environment reactions*, respectively; $\alpha : 2^{\mathcal{F}} \to 2^{Act}$ represents *action preconditions*; $\beta : 2^{\mathcal{F}} \times Act \to 2^{React}$ represents *reaction preconditions*; and $\delta : 2^{\mathcal{F}} \times Act \times React \mapsto 2^{\mathcal{F}}$ is the *partial transition function* such that $\delta(s, a, r)$ is defined iff $a \in \alpha(s)$ and $r \in \beta(s, a)$. We assume that FOND domains satisfy the following properties:

P1: *Existence of environment reaction*. For every domain state $s \in 2^{\mathcal{F}}$ and agent action $a \in \alpha(s)$, there exists one environment reaction $r \in \beta(s, a)$, written: $\forall s \in 2^{\mathcal{F}}.\forall a \in \alpha(s).\exists r \in \beta(s, a)$;

P2: *Uniqueness of environment reaction.* For every domain state $s \in 2^{\mathcal{F}}$, agent action $a \in \alpha(s)$, and successor state $s' = \delta(s, a, r)$ for some reaction r, the reaction r is unique, written:

$$\forall s \in 2^{\mathcal{F}}, \forall a \in \alpha(s), \forall r_1, r_2 \in \beta(s, a).$$
$$\delta(s, a, r_1) = \delta(s, a, r_2) \implies r_1 = r_2.$$

Nondeterministic domains usually adopted in FOND planning [10, 20], say expressed in PDDL [23], can be captured by the notion above by introducing reactions corresponding to the nondeterministic effects of agent actions [16]. Considering the domain compactly represented in PDDL, we identify its *size* with $|\mathcal{F}|$.

At each time step, a FOND domain evolves as follows: the agent performs an action a that satisfies action preconditions in the current domain state s; the environment chooses a reaction r that satisfies reaction preconditions; and the successor domain state is $s' = \delta(s, a, r)$. State transitions are defined iff both agent and environment follow their respective preconditions; nondeterminism for the agent comes from not knowing how the environment reacts.

Formally: a *domain trace* is a (finite or infinite) sequence $\tau = (s_0)(a_1, r_1, s_1)\cdots$ such that: (*i*) s_0 is the initial domain state; and (*ii*) for every $i \geq 0$, we have that $s_{i+1} = \delta(s_i, a_{i+1}, r_{i+1})$ for some $a_{i+1} \in \alpha(s_i)$ and $r_{i+1} \in \beta(s_i, a_{i+1})$.

An *agent strategy* is a *partial* function $\sigma : (2^{\mathcal{F}})^+ \mapsto Act$ such that, for every domain trace $\tau = (s_0)(a_1, r_1, s_1)\cdots(a_n, r_n, s_n)$, if $\sigma(s_0 \cdots s_n)$ is defined, then $\sigma(s_0 \cdots s_n) \in \alpha(s_n)$, i.e., σ follows action preconditions. We write $\sigma(\tau) = \perp$ to denote that σ is undefined in τ. A domain trace $\tau = (s_0)(a_1, r_1, s_1)\cdots$ is consistent with σ if: (*i*) $a_{i+1} = \sigma(s_0 \cdots s_i)$ for every $i \geq 0$; and (*ii*) if τ is finite, say $\tau = (s_0)(a_1, r_1, s_1)\cdots(a_n, r_n, s_n)$, then $\sigma(s_0 \cdots s_n) = \perp$. Note that a finite domain trace τ is consistent with σ if σ, being a partial function, is undefined in τ – which we think of as the strategy *terminating* its execution.

A *goal* is an LTL$_f$ formula φ over $\mathcal{F}$. Every domain trace $\tau = (s_0)(a_1, r_1, s_1)\cdots$ corresponds to a trace $\tau|_{2^{\mathcal{F}}} = s_0 s_1 \cdots$, so that we can evaluate LTL$_f$ goals over finite domain traces.

LTL$_f$ *synthesis in* FOND *domains* (aka LTL$_f$ *strong* FOND *planning*) is the problem of computing an agent strategy that achieves the goal against every nondeterministic environment response – called *winning strategy* (aka *strong plan*) – if one such strategy exists. Formally: an agent strategy σ is *winning* for φ in $\mathcal{D}$ if, for every domain trace τ consistent with σ, we have that $\tau|_{2^{\mathcal{F}}}$ is finite and $\tau|_{2^{\mathcal{F}}} \models \varphi$. When φ and/or $\mathcal{D}$ are clear from the context, we may simply say, e.g., σ is winning for φ. LTL$_f$ synthesis in FOND domains is 2EXPTIME-complete in the size of φ and EXPTIME-complete in the size $\mathcal{D}$ (i.e., $|\mathcal{F}|$), respectively [12].

3 Joker Strategies

In this paper, we also consider the case in which the agent does not have a winning strategy: in such a case, the agent can either "give up" or resort to a strategy that possibly achieves the goal. Inspired by Joker strategies in games on graphs [30], we introduce Joker strategies for LTL$_f$ goals in FOND domains. Such strategies assume that the environment is *not strictly adversarial*: it may sometimes be cooperative and help the agent to achieve its goal. This assumption holds in FOND: the domain – the world the agent operates in – is rather agnostic towards the agent and may sometimes unfold its nondeterminism to help the agent achieve its goal. When the environment's cooperation is assumed, Joker strategies prescribe a *Joker move*, denoted $\bigstar(a, r)$, which: (*i*) specifies the next agent action a; (*ii*) fixes the next environment reaction to r. Subsequently, we introduce minimal Joker strategies, which prescribe the minimum number of Joker moves, thus relying the least on the environment's cooperation.

Formally: a *Joker strategy* is a partial function $\sigma^{\bigstar} : (2^{\mathcal{F}})^+ \mapsto Act \cup \bigstar$, where $\bigstar = Act \times React$, such that, for every domain trace $\tau = (s_0)(a_1, r_1, s_1)\cdots(a_n, r_n, s_n)$, if $\sigma^{\bigstar}(s_0 \cdots s_n)$ is defined, then one of the following holds: (*i*) if $\sigma^{\bigstar}(s_0 \cdots s_n) = a_{n+1}$, then $a_{n+1} \in \alpha(s_n)$; (*ii*) if $\sigma^{\bigstar}(s_0 \cdots s_n) = \bigstar(a_{n+1}, r_{n+1})$, then $a_{n+1} \in \alpha(s_n)$ and $r_{n+1} \in \beta(s_n, a_{n+1})$. That is: $\sigma^{\bigstar}$ always follows action preconditions; if $\sigma^{\bigstar}$ prescribes a Joker move – thus fixing the environment reaction – then $\sigma^{\bigstar}$ also follows reaction preconditions.

A domain trace $\tau = (s_0)(a_1, r_1, s_1)\cdots$ is consistent with a Joker strategy $\sigma^{\bigstar}$ if: (*i*) for every $i \geq 0$, either $\sigma^{\bigstar}(s_0 \cdots s_i) = \bigstar(a_{i+1}, r_{i+1})$ *or* $\sigma^{\bigstar}(s_0 \cdots s_i) = a_{i+1}$; and (*ii*) if τ is finite, say $\tau = (s_0)(a_1, r_1, s_1)\cdots(a_n, r_n, s_n)$, then $\sigma^{\bigstar}(s_0 \cdots s_n) = \perp$. Observe that if $\sigma^{\bigstar}$ prescribes a Joker move – thus also fixing the environment reaction – every domain trace that does not follow that reaction is *not* consistent with $\sigma^{\bigstar}$. A Joker strategy $\sigma^{\bigstar}$ is *Joker-winning* for φ in $\mathcal{D}$ if, for every play τ consistent with it, we have that $\tau|_{2^{\mathcal{F}}}$ is finite and $\tau|_{2^{\mathcal{F}}} \models \varphi$. Intuitively, that a Joker strategy is Joker winning means that, if the environment cooperates as assumed in Joker moves, the agent will achieve the goal.

Now, suppose that there exists a domain trace $\tau = (s_0)(a_1, r_1, s_1)\cdots(a_n, r_n, s_n)$ such that $\tau|_{2^{\mathcal{F}}} \models \varphi$. The Joker strategy $\sigma^{\bigstar}$ such that $\sigma^{\bigstar}(s_0 \cdots s_i) = \bigstar(a_{i+1}, r_{i+1})$ for every $0 \leq i < n$ is Joker winning. An agent using one such strategy assumes a *maximally cooperative* environment, which is not suitable in FOND planning: during execution, when the environment is not guaranteed to cooperate, the agent may unnecessarily take a chance by relying on the environment's cooperation and expose to goal failure. Instead, we require the agent to rely the least on the environment's cooperation, reason for which we introduce *minimal* Joker strategies.

Let φ be an LTL$_f$ goal, $\mathcal{D}$ a FOND domain, and $\sigma^{\bigstar}$ a Joker strat-

egy. We assign a *cost* $c_{\varphi|\mathcal{D}}(\sigma^\star)$ to $\sigma^\star$, which is formally defined as follows: if $\sigma^\star$ is Joker winning, then $c_{\varphi|\mathcal{D}}(\sigma^\star)$ is the maximum number of Joker moves that $\sigma^\star$ makes in the domain traces consistent with it; else, if $\sigma^\star$ is not Joker winning, then $c_{\varphi|\mathcal{D}}(\sigma^\star) = +\infty$. A minimal Joker strategy is one that is assigned minimal cost.

Definition 1. *A Joker strategy $\sigma_1^\star$ is minimal (aka* mj*-strategy) for φ in $\mathcal{D}$ if $c_{\varphi|\mathcal{D}}(\sigma_1^\star) \leq c_{\varphi|\mathcal{D}}(\sigma_2^\star)$ for every Joker strategy $\sigma_2^\star$.*

It follows by the definition that *mj*-strategies always exist.

Proposition 1. *Let $\mathcal{D}$ be a* FOND *domain and φ an* LTL_f *goal. There exists a* mj*-strategy for φ in $\mathcal{D}$.*

4 Minimal Best-Effort Strategies

We now investigate the relation between *mj*- and winning strategies. Subsequently, we investigate the relation between *mj*- and *best-effort strategies* [1, 15] – which can be seen as a generalization of winning strategies. Best-effort strategies base on the game-theoretic notion of *dominance* and capture the intuition that, when a winning strategy does not exist, the agent should do its best to achieve the goal. Based on the relation between *mj*- and best-effort strategies, we introduce *minimal best-effort strategies*. Such strategies always exist and guarantee that the agent: (*i*) does its best to achieve the goal; (*ii*) it relies the least on the environment's cooperation. Finally, we formalize the problem which we will investigate in the rest of the paper: LTL_f synthesis of minimal best-effort strategies in FOND domains.

First, observe that a Joker strategy $\sigma^\star$ is *not* formally an agent strategy of the form $\sigma : (2^\mathcal{F})^+ \to Act$, since it can also prescribe Joker moves. However, each $\sigma^\star$ is equivalent to one such strategy σ, which can be obtained by projecting Joker moves on their first component: the agent action. Formally: a Joker strategy $\sigma^\star$ *induces* the agent strategy σ such that, for every $\tau \in (2^\mathcal{F})^+$: (*i*) if $\sigma^\star(\tau) = \bigstar(a_{n+1}, r_{n+1})$, then $\sigma(\tau) = a_{n+1}$; (*ii*) otherwise, $\sigma(\tau) = \sigma^\star(\tau)$.

Suppose that a winning strategy exists. We observe the following: (*i*) every *mj*-strategy has cost 0 and induces a winning strategy, since it achieves the goal without environment's cooperation; (*ii*) every winning strategy behaves exactly as an *mj*-strategy that does not prescribe Joker moves and has cost 0, thus being minimal. As a result:

Proposition 2. *If there exists a winning strategy for φ in $\mathcal{D}$, the strategies induced by the* mj*-strategies are exactly the winning strategies.*

In what follows, we review best-effort strategies. That an agent does its best to achieve a goal is formalized through the game-theoretic notion of *dominance*, which requires introducing *environment strategies* in FOND domains. Formally: an *environment strategy* is a *total* function $\gamma : Act^+ \to React$ such that, for every domain trace $\tau = (s_0)(a_1, r_1, s_1)\cdots$, we have that $r_{i+1} = \gamma(a_1 \cdots a_{i+1}) \in \beta(s_i, a_{i+1})$ for every $i \geq 0$, i.e., γ follows reaction preconditions. An environment strategy always exists due to existence of environment reaction, see Section 2, P1. A domain trace $\tau = (s_0)(a_1, r_1, s_1)\cdots$ is consistent with γ if $r_{i+1} = \gamma(a_1 \cdots a_{i+1})$ for every $i \geq 0$. Given an agent strategy σ and an environment strategy γ, there exists a unique domain trace consistent with both σ and γ, denoted $\tau(\sigma, \gamma) = (s_0)(a_1, r_1, s_1)\cdots$, defined as follows: (*i*) $a_{i+1} = \sigma(s_0 \cdots s_i)$ for every $i \geq 0$; (*ii*) $r_{i+1} = \gamma(a_1 \cdots a_{i+1})$ for every $i \geq 0$; and (*iii*) if $\tau(\sigma, \gamma)$ is finite, say $\tau(\sigma, \gamma) = (s_0)(a_1, r_1, s_1)\cdots(a_n, r_n, s_n)$, then $\sigma(s_0 \cdots s_n) = \bot$.

Let: $\mathcal{D}$ be a FOND domain; φ an LTL_f goal; and σ_1 and σ_2 agent strategies. We say that σ_1 *dominates* σ_2 for φ in $\mathcal{D}$, written $\sigma_1 \geq_{\varphi|\mathcal{D}} \sigma_2$, if, for every environment strategy γ, if $\tau(\sigma_2, \gamma)|_{2^\mathcal{F}}$ is finite and $\tau(\sigma_2, \gamma)|_{2^\mathcal{F}} \models \varphi$, then $\tau(\sigma_1, \gamma)|_{2^\mathcal{F}}$ is finite and $\tau(\sigma_1, \gamma)|_{2^\mathcal{F}} \models \varphi$. Furthermore, we say that σ_1 *strictly dominates* σ_2, 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, $\sigma_1 >_{\varphi|\mathcal{D}} \sigma_2$ means that σ_1 does at least as well as σ_2 against every environment strategy and strictly better against at least one such a strategy. An agent that uses σ_2 is not doing its best: if it used σ_1 instead, it could achieve the goal against a strictly larger set of environment strategies. Within this framework, a best-effort strategy is one that is not strictly dominated by any other strategy.

Definition 2. *An agent strategy σ is best-effort (aka* be*-strategy) for φ in $\mathcal{D}$ if there is no other agent strategy σ' such that $\sigma' >_{\varphi|\mathcal{D}} \sigma$.*

We note that *be*-strategies share some properties with *mj*-strategies: (*i*) they always exist; and (*ii*) if a winning strategy exists, the best-effort strategies are exactly the winning strategies [15].

LTL_f best-effort synthesis in FOND domains is the problem of computing a best-effort strategy; its complexity is EXPTIME-complete in the domain and 2EXPTIME-complete in the goal, respectively [15], i.e., as that of standard LTL_f synthesis in FOND domains.

Best-effort strategies also admit an alternative characterization – which we will use to investigate the relation between *be*- and *mj*-strategies. This characterization is based on the notion of value of a *history*, i.e., a finite domain trace. Intuitively, the value of a history h is as follows: "winning" $(+1)$ if the agent can achieve the goal from h regardless of the environment response; "pending" (0) if the agent can achieve the goal from h for some environment response, but not all; and "losing" (-1), otherwise. By exploiting this notion, we can characterize best-effort strategies as those that witness the maximum value of every history consistent with them.

Formally: for an agent strategy σ and a history h consistent with it, we denote by $\Gamma(h, \sigma)$ the set of environment strategies γ such that h is a prefix of $\tau(\sigma, \gamma)$. Furthermore, we denote by $\mathcal{H}_\mathcal{D}(\sigma)$ the set of histories h such that $\Gamma(h, \sigma)$ is non-empty, i.e., the set of histories of $\mathcal{D}$ consistent with σ and some environment strategy γ. We define:

- $val_{\varphi|\mathcal{D}}(\sigma, h) = +1$ ("winning"), if $\tau(\sigma, \gamma)|_{2^\mathcal{F}}$ is finite and $\tau(\sigma, \gamma)|_{2^\mathcal{F}} \models \varphi$ for every $\gamma \in \Gamma(h, \sigma)$;
- $val_{\varphi|\mathcal{D}}(\sigma, h) = 0$ ("pending"), if $\tau(\sigma, \gamma)|_{2^\mathcal{F}}$ is finite and $\tau(\sigma, \gamma)|_{2^\mathcal{F}} \models \varphi$ for some $\gamma \in \Gamma(h, \sigma)$, but not all;
- $val_{\varphi|\mathcal{D}}(\sigma, h) = -1$ ("losing"), otherwise.

Let $val_{\varphi|\mathcal{D}}(h)$ be the maximum value of $val_{\varphi|\mathcal{D}}(\sigma, h)$ among all the strategies σ such that $h \in \mathcal{H}_\mathcal{D}(\sigma)$[1]. The following is the history-based characterization of *be*-strategies:

Theorem 3 ([15])**.** *An agent strategy σ is best-effort for φ in $\mathcal{D}$ iff $val_{\varphi|\mathcal{D}}(\sigma, h) = val_{\varphi|\mathcal{D}}(h)$ for every $h \in \mathcal{H}_\mathcal{D}(\sigma)$.*

Following Theorem 3, a *be*-strategy behaves as follows from every history consistent with it: (*i*) if the history is winning, the best-effort strategy will achieve the goal regardless of the environment response; else, (*ii*) if the history is pending, the best-effort strategy will achieve the goal should the environment cooperate; else, (*iii*) if the history is losing, the best-effort strategy prescribes some action.

We now establish the relation between *mj*- and *be*-strategies. In doing so, we represent FOND domains using transition graphs as in Figures 1 and 2: nodes represent domain states; edges denote state transitions resulting from agent actions and environment reactions; labels on edges denote agent actions and environment reactions, colored in blue and red, respectively.

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

For convenience, in the following we may omit $\mathcal{D}$ and φ and consider, e.g., *be*-strategy as an abbreviation for *be*-strategy for φ in $\mathcal{D}$.

First, we observe that *be*-strategies are in general distinct from strategies induced by *mj*-strategies, since they may rely too much on the environment's cooperation. This is stated in the following:

Theorem 4. *There exists a* be*-strategy that is not induced by any* mj*-strategy.*

Proof. Consider the domain in Figure 1 and the LTL$_f$ goal $\varphi = \Diamond(\ell_3)$. There exist two best-effort strategies σ_1 and σ_2 such that:
1. σ_1 prescribes: $go(\ell_0, \ell_1)$; $go(\ell_1, \ell_2)$; and $go(\ell_2, \ell_3)$;
2. σ_2 prescribes: $go(\ell_0, \ell_5)$; $go(\ell_5, \ell_4)$; and $go(\ell_4, \ell_3)$.

We have that σ_1 and σ_2 are both best-effort as they satisfy Theorem 3. However: σ_1 requires environment's cooperation twice – in ℓ_1 and ℓ_2; σ_2 requires environment's cooperation only once – in ℓ_5.

There exists a unique *mj*-strategy $\sigma^\star$ that prescribes: $go(\ell_0, \ell_5)$; $\bigstar(go(\ell_5, \ell_4), move)$; and $go(\ell_4, \ell_3)$. We have that $\sigma^\star$ is minimal as it plays the minimum number of Joker moves. The strategy induced by $\sigma^\star$ is σ_2. As a result, σ_1 is a *be*-strategy that is not induced by any *mj*-strategy, which proves the claim. □

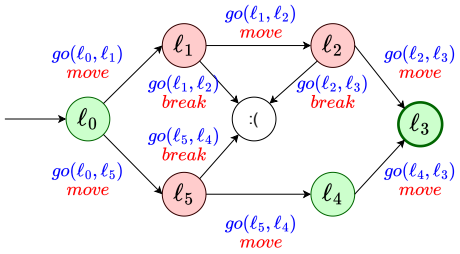

Figure 1. FOND domain where agent can navigate between locations through $go(\ell_i, \ell_j)$. We distinguish between *safe* and *dangerous* locations, colored in green and red, respectively: from safe locations, the agent always reaches the neighboring location – captured by the environment reaction $move$; from dangerous locations, the agent may also break – captured by the environment reaction $break$ – in which case it cannot move anymore.

We have an analogous result in the other direction: strategies induced by *mj*-strategies may not be *be*-strategies. This is because strategies induced by *mj*-strategies may be losing in histories where the environment does not cooperate as assumed in Joker moves, as shown in the following:

Theorem 5. *There exists a strategy induced by a* mj*-strategy that is not a* be*-strategy.*

Proof. Consider the domain in Figure 2 and the LTL$_f$ goal $\varphi = \Diamond(\ell_2)$. Let $\sigma^\star$ be the *mj*-strategy that prescribes $\bigstar(go(\ell_0, \ell_2), move)$ in the initial state and is undefined otherwise. The strategy induced by $\sigma^\star$ is σ and prescribes $go(\ell_0, \ell_2)$. As the agent performs $go(\ell_0, \ell_2)$ it may slip and reach ℓ_1. In ℓ_1, the agent has a strategy that reaches ℓ_2 with the environment's cooperation: the generated history is pending (has value 0). However, σ is undefined in that history: it is losing (has value -1). By Theorem 3, σ is not a *be*-strategy, which proves the claim. □

Theorems 4 and 5 leads to the following observations:
1. *Be*-strategies: (*i*) achieve the goal in every pending history consistent with them, should the environment cooperate; but (*ii*) may rely too much on the environment's cooperation, see Theorem 4.

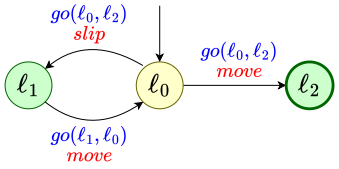

Figure 2. FOND domain where agent can navigate between locations through $go(\ell_i, \ell_j)$. We distinguish between *safe* and *slippery* locations, colored in green and yellow, respectively: from safe locations, the agent always reaches the neighboring location – captured by the environment reaction $move$; from slippery locations, the agent may also slip to another neighboring location – captured by the environment reaction $slip$.

2. Strategies induced by *mj*-strategies: (*iii*) rely the least on the environment's cooperation; but (*iv*) may be losing in pending histories where the environment does not respond as assumed in Joker moves, see Theorem 5.

In this paper, we are interested in strategies that satisfy both and only (*i*) and (*iii*): they achieve the goal in every pending history consistent with them, should the environment cooperate, but rely the least on the environment's cooperation. To capture these requirements, we formalize the notion of *minimal best-effort* strategy.

First, we use the notion of history to introduce some useful concepts for the formalization. Let $\mathcal{D}$ be a FOND domain, φ an LTL$_f$ goal, and h a history: (1) an agent strategy σ is *winning* for φ in $\mathcal{D}$ *from* h if, for every domain trace τ that is consistent with σ and such that h is a prefix of τ, we have that $\tau|_{2^\mathcal{F}}$ is finite and $\tau|_{2^\mathcal{F}} \models \varphi$; (2) a Joker strategy $\sigma^\star$ is *Joker winning* for φ in $\mathcal{D}$ *from* h if, for every domain trace τ consistent with $\sigma^\star$ and such that h is a prefix of τ, we have that $\tau|_{2^\mathcal{F}}$ is finite and $\tau|_{2^\mathcal{F}} \models \varphi$. For convenience, we may sometimes omit φ and/or $\mathcal{D}$ and say, e.g., σ is winning from h. (3) We assign a *cost* to $\sigma^\star$ *from* h, denoted $c_{\varphi|\mathcal{D}}(\sigma^\star, h)$, as follows: if $\sigma^\star$ is Joker winning from h, then $c_{\varphi|\mathcal{D}}(\sigma^\star, h)$ is the maximum number of Joker moves that $\sigma^\star$ makes in the suffixes τ' of the domain traces $\tau = h \cdot \tau'$ consistent with it; else, if $\sigma^\star$ is not Joker winning from h, then $c_{\varphi|\mathcal{D}}(\sigma^\star, h) = +\infty$. (4) A Joker strategy $\sigma_1^\star$ is *minimal* (aka *mj*-strategy) for φ in $\mathcal{D}$ *from* h if $c_{\varphi|\mathcal{D}}(\sigma_1^\star, h) \leq c_{\varphi|\mathcal{D}}(\sigma_2^\star, h)$ for every Joker strategy $\sigma_2^\star$. Intuitively, *mj*-strategies from h are those that rely the least on the environment's cooperation to achieve the goal from h. (5) An agent strategy σ_1 *follows* the agent strategy σ_2 from h if, for every domain trace $\tau \in (2^\mathcal{F})^+$ such that $h|_{2^\mathcal{F}}$ is a prefix of τ, we have $\sigma_1(\tau) = \sigma_2(\tau)$.

Theorem 6. *Let $\mathcal{D}$ be a* FOND *domain, φ an* LTL$_f$ *goal, and h a history. There exists a* mj*-strategy for φ in $\mathcal{D}$ from h.*

Theorem 7. *If there exists a winning strategy for φ in $\mathcal{D}$ from h, the strategies induced by* mj*-strategies from h are exactly the winning strategies from h.*

Recall we are interested in a strategy σ that achieves properties (*i*) and (*iii*) above. Such properties can be achieved by having σ following, from every history h consistent with it, the strategy induced by a *mj*-strategy from h. One such strategy σ is called *minimal best-effort*.

Definition 3. *An agent strategy σ is* minimal best-effort *(aka* mbe*-strategy) for φ in $\mathcal{D}$ if, for every $h \in \mathcal{H}_\mathcal{D}(\sigma)$, σ follows the strategy induced by a* mj*-strategy for φ in $\mathcal{D}$ from h.*

To see why a *mbe*-strategy σ satisfies requirement (*i*) above, observe that it is best-effort by the history-based characterization of *be*-strategies in Theorem 3. Let h be a history and observe that a *mj*-strategy from h always exists by Theorem 6:

1. If h is winning, every *mj*-strategy from h induces a winning strategy from h by Theorem 7; σ follows one such induced strategy – thus being winning from h;
2. If h is pending, every *mj*-strategy from h induces a strategy that achieves the goal should the environment cooperate; σ follows one such an induced strategy – thus being pending from h;
3. If h is losing, every agent strategy is losing from h, including σ.

Theorem 8. *Let σ be a* mbe*-strategy for φ in $\mathcal{D}$. We have that σ is a* be*-strategy for φ in $\mathcal{D}$.*

To see why a *mbe*-strategy σ satisfies requirement (*iii*) above, observe that there exists a *mj*-strategy $\sigma^\star$ that induces σ. Indeed: σ follows the strategy induced by an *mj*-strategy $\sigma^\star_h$ from every history $h \in \mathcal{H}_\mathcal{D}(\sigma)$; it follows that $\sigma^\star$ can be constructed by having $\sigma^\star$ following $\sigma^\star_h$ from every history h consistent with σ.

Theorem 9. *Let σ be a* mbe*-strategy for φ in $\mathcal{D}$. There exists a* mj*-strategy for φ in $\mathcal{D}$ that induces σ.*

Inherited from *mj*- and *be*-strategies, we have the following relation between *mbe*-strategies and winning strategies when a winning strategy exists.

Theorem 10. *If a winning strategy for φ in $\mathcal{D}$ exists, the* mbe*-strategies are exactly the winning strategies.*

As for standard *mj*- and *be*-strategies, *mbe*-strategies always exist.

Theorem 11. *Let $\mathcal{D}$ be a* FOND *domain and φ an* LTL$_f$ *goal. There exists a* mbe*-strategy for φ in $\mathcal{D}$.*

In this paper, we investigate LTL$_f$ synthesis of *mbe*-strategies in FOND domains, defined as follows:

Definition 4. *Let $\mathcal{D}$ be a* FOND *domain and φ an* LTL$_f$ *goal.* LTL$_f$ *synthesis in* FOND *domains of* mbe*-strategies is the problem of computing a* mbe*-strategy for φ in $\mathcal{D}$.*

5 Synthesizing *mbe*-Strategies

In this section, we present an algorithm for LTL$_f$ synthesis of *mbe*-strategies in FOND domains. This algorithm reduces to solving reachability games played over deterministic finite automata, which we briefly review. We establish the correctness of the algorithm and its optimality wrt the computational complexity of the problem.

Games over Deterministic Finite Automata A *transition system* is a tuple $\mathcal{T} = (\Sigma, S, \iota, \delta)$, where: Σ is a finite input alphabet; S is a finite set of states; $\iota \in S$ is the initial state; $\delta : S \times \Sigma \to S$ is a *total* transition function; and $F \subseteq S$ is the set of final states. The size of $\mathcal{T}$ is $|S|$. Given a finite trace $\alpha = \alpha_0\alpha_1 \dots \alpha_n$ over Σ, we extend δ to be a function $\delta : S \times \Sigma^* \to S$ as follows: $\delta(s, \epsilon) = s$, where ϵ is the empty trace, and, if $s_n = \delta(s, \alpha_0 \dots \alpha_{n-1})$, then $\delta(s, \alpha_0 \dots \alpha_n) = \delta(s_n, \alpha_n)$. A *deterministic finite automaton* (DFA) is a pair $\mathcal{A} = (\mathcal{T}, R)$, where $\mathcal{T} = (\Sigma, S, \iota, \delta)$ is a deterministic transition system and $R \subseteq S$ is the set of final states. A trace α is accepted by $\mathcal{A}$ if $\delta(\iota, \alpha) \in R$. The *language* of $\mathcal{A}$ is the set of traces that $\mathcal{A}$ accepts. In this paper, we focus on the following property: for every LTL$_f$ formula φ, there exists a DFA $\mathcal{A}_\varphi$ that accepts the traces that satisfy φ; the size of $\mathcal{A}_\varphi$ is at most doubly-exponential in that of φ [14]. However, the worst-case doubly-exponential blowup while constructing DFAs of LTL$_f$ formulas is rare in practice, see, e.g., [19].

A DFA game is a DFA $G = (\mathcal{T}, R)$, where: $\mathcal{T} = (Act \times React, S, \iota, \delta)$ is a transition system, also called the *game arena*; Act and $React$ are disjoint sets of actions and reactions under control of agent and environment, respectively. A play over $\mathcal{G}$ is a finite or infinite sequence $\tau = (a_1, r_1)(a_2, r_2) \cdots$. A game strategy is a partial function $\kappa : S \mapsto Act$ that maps states of the game to agent actions. We write $\kappa(s) = \bot$ to denote that κ is undefined in s – which we think of as the strategy terminating its execution. A play $\tau = (a_1, r_1)(a_2, r_2) \cdots$ is consistent with κ if: (*i*) $a_{k+1} = \kappa(\delta(\iota, \tau^k))$, where $\tau^k = (a_1, r_1) \cdots (a_k, r_k)$, for every $k \geq 0$; (*ii*) if τ is finite, say $\tau = (a_1, r_1) \cdots (a_n, r_n)$, then $\kappa(\delta(\iota, \tau)) = \bot$. A game strategy is *winning* in $\mathcal{G}$ if, for every play τ that is consistent with κ, τ is finite and accepted by $\mathcal{G}$. That is: DFA games requires the set of final states to be visited at least once.

Game Arena for LTL$_f$ Synthesis in FOND Domains Let $\mathcal{D}$ be a FOND domain and φ an LTL$_f$ goal. We review how to construct a game arena for LTL$_f$ synthesis in FOND domains [15].

The FOND domain is transformed into a transition system. It should be noted that: the transition system of the domain must have a *total* transition function; but the transition function of the FOND domain is *partial*. To obtain a total transition function we drop agent and environment preconditions and introduce two states, s^{err}_{ag} and s^{err}_{env}, respectively called *agent* and *environment error state*, denoting that agent and environment violate their preconditions. Formally: given a FOND $\mathcal{D} = (2^\mathcal{F}, s_0, Act, React, \alpha, \beta, \delta)$, its corresponding transition system is $\mathcal{T}_\mathcal{D} = (Act \times React, 2^\mathcal{F} \cup \{s^{ag}_{err}, s^{env}_{err}\}, s_0, \delta_+)$, where: δ_+ is the transition function such that: (*i*) if $s \in \{s^{ag}_{err}, s^{env}_{err}\}$, then $\delta_+(s, a, r) = s$; else (*ii*) $\delta_+(s, a, r) = \delta(s, a, r)$ if $a \in \alpha(s)$ and $r \in \beta(s, a)$; else (*iii*) $\delta(s, a, r) = s^{err}_{ag}$ if $a \notin \alpha(s)$; else (*iv*) $\delta(s, a, r) = s^{err}_{env}$ if $a \in \alpha(s)$ and $r \notin \beta(s, a)$.

We now review how to construct the game arena. This is obtained by composing the transition systems of the FOND domain $\mathcal{D}$ and (the transition system of) the DFA of the agent goal φ. Formally: let $\mathcal{T}_\mathcal{D} = (Act \times React, 2^\mathcal{F} \cup \{s^{ag}_{err}, s^{env}_{err}\}, s_0, \delta')$ be the transition system of the domain, and $\mathcal{A}_\varphi = (\mathcal{T}_\varphi, R_\varphi)$, where $\mathcal{T}_\varphi = (2^\mathcal{F}, Q_\varphi, \iota_\varphi, \delta_\varphi)$, the DFA of φ; we construct the game arena $\mathcal{T} = \mathcal{T}_\mathcal{D} \times \mathcal{T}_\varphi = (Act \times React, (2^\mathcal{F} \cup \{s^{err}_{ag}, s^{err}_{env}\}) \times Q_\varphi, (s_0, \delta_\varphi(\iota_\varphi, s_0)), \partial)$, where:

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

Where $s' = \delta_+(s, a, r)$.

We will consider the following regions over $\mathcal{T}_\mathcal{D} \times \mathcal{T}_\varphi$:

1. $R'_\varphi = \{(s, q) \mid q \in R_\varphi\}$: the set of states where φ holds;
2. $S^{err}_{ag} = \{(s, q) \mid s = s^{err}_{ag}\}$: the set of states where the agent violated an action precondition;
3. $S^{err}_{env} = \{(s, q) \mid s = s^{err}_{env}\}$: the set of states where the environment violated a reaction precondition.

We will also denote by $\overline{S^{err}_{ag}}$ and $\overline{S^{err}_{env}}$ the complement of S^{err}_{ag} and S^{err}_{env} wrt the state space of $\mathcal{T}$, respectively.

Consider the set of states $\overline{S^{err}_{ag}} \cap (S^{err}_{env} \cup R'_\varphi)$. This is reached if: (*i*) the agent never violates action preconditions; and (*ii*) either the environment violates reaction preconditions or φ holds. As a result, there exists a winning strategy for φ in $\mathcal{D}$ iff there exists a winning strategy in game $(\mathcal{T}_\mathcal{D} \times \mathcal{T}_\varphi, \overline{S^{err}_{ag}} \cap (S^{err}_{env} \cup R'_\varphi))$ [15].

Synthesis Algorithm for *mbe*-Strategies We present the synthesis algorithm for *mbe*-strategies. The algorithm consists of five steps: Steps 1 and 2 construct the game arena as above; Step 3 performs a nested fixpoint computation over the state space of a DFA game; Steps 4 and 5 compute the output *mbe*-strategy.

The fixpoint computations performed by the algorithm use: (*i*) the *universal preimage* $PreA(\mathcal{G}, \mathcal{E})$; and (*ii*) the *existential preimage* $PreE(\mathcal{G}, \mathcal{E})$. The former is a function that, given a game $\mathcal{G} = (\mathcal{T}, R)$ with $\mathcal{T} = (Act \times React, S, \iota, \delta)$ and a set of states $\mathcal{E} \subseteq S$, returns the set of states for which there exists an agent action that progresses $\mathcal{G}$ to a state in $\mathcal{E}$ regardless of the environment reaction; the latter is similar, except that it returns the set of states for which there exists an agent action and an environment reaction that progress $\mathcal{G}$ to a state in $\mathcal{E}$. Formally:

$$PreA(\mathcal{G}, \mathcal{E}) = \{s \in S \mid \exists a \in Act. \forall r \in React. \delta(s, (a, r)) \in \mathcal{E}\}$$
$$PreE(\mathcal{G}, \mathcal{E}) = \{s \in S \mid \exists a \in Act. \exists r \in React. \delta(s, (a, r)) \in \mathcal{E}\}$$

In what follows, we use the abbreviation $W(\mathcal{T}, R)$ to denote the following fixpoint computation:

$$W_0(\mathcal{T}, R) = R$$
$$W_{k+1}(\mathcal{T}, R) = W_k(\mathcal{T}, R) \cup PreA(\mathcal{T}, W_k(\mathcal{T}, R))$$
$$W(\mathcal{T}, R) = \bigcup_{k \in \mathbb{N}} W_k(\mathcal{T}, R)$$

That is: $W(\mathcal{T}, R)$ returns the set of states of $\mathcal{T}$ where the agent can reach R regardless of the environment reaction.

We use the following results to construct the output *mbe*-strategy:

(†) Given a game arena $\mathcal{T}$, where $\mathcal{T} = (Act \times React, S, \iota, \delta)$, and a game strategy κ, we can construct a strategy $\sigma'_{(\mathcal{T},\kappa)} : React^* \to Act$ as follows: (*i*) $\sigma'(\epsilon) = \kappa(\iota)$; (*ii*) for every $\tau = (a_1, r_1) \cdots (a_n, r_n)$, define $\sigma'_{(\mathcal{T},\kappa)}(r_1 \cdots r_n) = \kappa(\partial(\iota, \tau))$. The pair $(\mathcal{T}, \kappa)$ is called a *transducer*, where κ is the *output function*, and represents $\sigma'_{(\mathcal{T},\kappa)}$ [4].

(††) In the FOND domains we consider, see Section 2, agent strategies of the form $\sigma' : React^* \to Act$ and agent strategies of the form $\sigma : (2^{\mathcal{F}})^+ \to Act$ are *equivalent* [15]. The equivalence follows by: (*i*) having defined δ as a deterministic function; (*ii*) the property of uniqueness of environment reaction, see Section 2, P2.

We report below the algorithm for synthesizing *mbe*-strategies.

Algorithm 1. Let $\mathcal{D}$ be a FOND domain and φ an LTL$_f$ goal.

1. Transform: $\mathcal{D}$ into its corresponding transition system $\mathcal{T}_{\mathcal{D}}$; φ into its corresponding DFA $\mathcal{A}_\varphi = (\mathcal{T}_\varphi, R_\varphi)$;
2. Construct the game arena $\mathcal{T} = \mathcal{T}_{\mathcal{D}} \times \mathcal{T}_\varphi$. Let $Goal = \overline{S^{err}_{ag}} \cap (S^{err}_{env} \cup R'_\varphi)$. Denote by ∂ the transition function of $\mathcal{T}$ and by t its states (t_0 is the initial state).
3. Perform the following nested fixpoint computations:

$$JW_0(\mathcal{T}, Goal) = W(\mathcal{T}, Goal)$$
$$JW^{\bigstar}_{k+1}(\mathcal{T}, Goal) = JW_k(\mathcal{T}, Goal) \cup PreE(\mathcal{T}, JW_k(\mathcal{T}, Goal) \cap \overline{S^{err}_{env}}))$$
$$JW_{k+1}(\mathcal{T}, Goal) = W(\mathcal{T}, JW^{\bigstar}_{k+1}(\mathcal{T}, Goal))$$
$$JW(\mathcal{T}, Goal) = \bigcup_{k \in \mathbb{N}} JW_k(\mathcal{T}, Goal)$$

4. Define the game strategy κ such that:
   - 4.1. If $t \in W_{k+1}(\mathcal{T}, Goal) \setminus W_k(\mathcal{T}, Goal)$, then $\kappa(t) = a$ s.t. $\forall r \in React. \partial(t, (a, r)) \in W_k(\mathcal{T}, Goal)$; else
   - 4.2. If $t \in JW^{\bigstar}_{k+1}(\mathcal{T}, Goal) \setminus JW_k(\mathcal{T}, Goal)$, then $\kappa(t) = a$ s.t. $\exists r \in React. \partial(t, (a, r)) \in JW_k(\mathcal{T}, Goal)$; else
   - 4.3. If $t \in W_{\ell+1}(\mathcal{T}, JW^{\bigstar}_{k+1}(\mathcal{T}, Goal)) \setminus W_\ell(\mathcal{T}, JW^{\bigstar}_{k+1}(\mathcal{T}, Goal))$ then $\kappa(t) = a$ s.t. $\forall r \in React. \partial(t, (a, r)) \in W_\ell(\mathcal{T}, JW^{\bigstar}_{k+1}(\mathcal{T}, Goal))$; else
   - 4.4. $\kappa(t) = \bot$.
5. **Return** the strategy $\sigma'_{(\mathcal{T},\kappa)}$ as in (†). We have that $\sigma'_{(\mathcal{T},\kappa)}$ is equivalent to a strategy of the form $\sigma : (2^{\mathcal{F}})^+ \to Act$ by (††). One such a strategy σ matches the form of agent strategies required for LTL$_f$ synthesis in FOND domains, see Section 2.

We elaborate on Steps 3 and 4. The output strategy is returned as a transducer whose output function is a game strategy. The output function is constructed using the states collected during the fixpoint computations in Step 3. Recall that a *mbe*-strategy follows, from every history consistent with it, the strategy induced by a *mj*-strategy, c.f., Definition 3. The construction exploits the following:

1. Paths leading to a state added to $W(\mathcal{T}, Goal)$ correspond to histories h that admit a *mj*-strategy that induces a winning strategy from h. From every such a state, the synthesized strategy follows the fixpoint computation until reaching a state where φ holds or the environment did not follow its preconditions (Step 4, Line 4.1) – thus following a winning strategy from h.
2. Paths leading to a state added to $JW_{k+1}(\mathcal{T}, Goal)$ correspond to histories h that admit a *mj*-strategy that has cost $k + 1$. Among such states, those added to $JW^{\bigstar}_{k+1}(\mathcal{T}, Goal)$ require the environment's cooperation – captured by the existential preimage $PreE$ and the environment not violating its preconditions, written $\overline{S^{err}_{env}}$. From every state in $JW_{k+1}(\mathcal{T}, Goal)$, the synthesized strategy follows the fixpoint computation until a state where φ holds should the environment cooperate (Step 4, Lines 4.2 and 4.3) – thus following the strategy induced by a *mj*-strategy with cost $k + 1$ from h.
3. Paths leading to the remaining states correspond to histories h that only admit *mj*-strategies with cost $+\infty$, i.e., no Joker winning strategy exists and every Joker strategy is minimal. In every such a state, the synthesized strategy is undefined (Step 4, Line 4.4) – thus following the strategy induced by a *mj*-strategy from h.

The correctness of Algorithm 1 follows by Items 1-3 above and is established in the following:

Theorem 12. *Let σ be the strategy synthesized by Algorithm 1. We have that σ is a* mbe*-strategy for φ in $\mathcal{D}$.*

We investigate the complexity of Algorithm 1. That the fixpoint computations in Step 3 terminate follows from: (*i*) the monotonicity of $PreA$ and $PreE$; and (*ii*) the finiteness of the state space of the game constructed in Step 2. Such fixpoint computations terminate in at most quadratic time in the size of the game; the size of the game is at most singly- and doubly-exponential in that of $\mathcal{D}$ and φ, respectively. It follows that Algorithm 1 establishes membership of synthesis of *mbe*-strategies in EXPTIME and 2EXPTIME wrt $\mathcal{D}$ and φ, respectively. In fact, Algorithm 1 is optimal wrt the computational complexity of synthesis of *mbe*-strategies, established below:

Theorem 13. LTL$_f$ *synthesis of* mbe*-strategies in* FOND *domains is: 1.* EXPTIME*-complete in the size of the* FOND *domain; 2.* 2EXPTIME*-complete in the size of the* LTL$_f$ *goal.*

(Hardness follows by that of LTL$_f$ synthesis in FOND domains [12].)

Theorem 13 states that synthesis of *mbe*-strategies has the same computational complexity as that of *be*-strategies. This leads to the following observation. Suppose that the environment is not strictly adversarial and may sometimes help the agent to achieve its goal: *the agent should always use a* mbe*-strategy rather than a standard* be*-strategy*. This is because *mbe*-strategies guarantee that the agent does its best to achieve the goal, but also rely the least on the environment's cooperation – unlike some standard *be*-strategies, c.f.,

Theorem 4. In fact, *be*- and *mbe*-strategies should be computable approximatively with the same computational overhead. This is also confirmed empirically, as we discuss in the next section.

6 Experimental Analysis

We implemented Algorithm 1 in Section 5 by extending the symbolic synthesis framework proposed in [16, 15, 31], integrated in state-of-the-art synthesis tools [18]. Specifically, we rely on Lydia, which is among the best performing LTL$_f$-to-DFA conversion tool publicly available [11], to generate DFAs from LTL$_f$ formulas. Following the methodology outlined in [31], we adopt a fully symbolic representation, where both the structure of the DFA game graph and its transition relations are encoded using Binary Decision Diagrams (BDDs), with CUDD-3.0.0 [29] employed as the underlying BDD library.

Experimental Comparison. The goal of the experimental analysis is to evaluate the effectiveness of our *mbe*-synthesis algorithm. To conduct this evaluation, we compared the performance of our *mbe*-strategy synthesizer with that of the *be*-strategy synthesizer in [15]. We expect the *be*-strategy synthesizer to perform better than the *mbe*-strategy synthesizer since the *be*-synthesis algorithm requires linear time in the size of the game arena – instead of quadratic, as required by the *mbe*-synthesis algorithm. Recall that the size of the game arena is exponential (resp. doubly-exponential) in the size of the domain $\mathcal{D}$ (resp. LTL$_f$ goal φ), see Section 5. We aim to show that synthesizing a *mbe*-strategy brings a minimal overhead compared to synthesizing *be*-strategies – thus showing the effectiveness of our approach.

Benchmarks. For our experimental analysis, we selected a benchmark suite featuring the FOND domains TRIANGLETIRE and COFFEE, adapted from [26, 9, 7]. In the TRIANGLETIRE benchmarks, the agent must navigate a triangle-shaped grid environment, dealing with nondeterministic success or failure when moving; failure results in the agent having a flat tire, which the agent can change if the current agent's location contains a spare tire. In COFFEE benchmarks, the agent has to prepare and deliver coffee to different offices, dealing with similar nondeterministic success or failure when delivering a coffee; failure results in the agent not delivering the coffee. We considered COFFEE and TRIANGLETIRE benchmarks where the agent does not have a winning strategy, i.e., the agent has to use either a *be*-strategy or a *mbe*-strategy.

In TRIANGLETIRE benchmarks, we considered the following types of LTL$_f$ goals:

$$\Diamond(\Diamond(at\text{-}\ell_1) \vee \cdots \vee \Diamond(at\text{-}\ell_i)) \quad (1)$$
$$\Diamond(at\text{-}\ell_1) \wedge \cdots \wedge \Diamond(at\text{-}\ell_i) \quad (2)$$
$$\bigcirc(\bigcirc(\cdots(\bigcirc(at\text{-}\ell_i)\cdots)) \quad (3)$$
$$\Diamond(at\text{-}\ell_1 \wedge \Diamond(at\text{-}\ell_2 \wedge \Diamond(\cdots \wedge \Diamond(at\text{-}\ell_n)\cdots))) \quad (4)$$

Where atom $at\text{-}\ell$ denotes that the agent is in location ℓ. To reach a location ℓ, the agent must select a path and navigate it until reaching ℓ should the environment cooperate, i.e., not cause the agent to have a flat tire in a location that does not contain a spare tire. However, there is a significant difference between *mbe*- and *be*-strategies: mbe-*strategies select paths where the agent relies the least on the environment's cooperation;* be-*strategies may select paths where the agent relies too much on the environment's cooperation, see, e.g., Theorem 4*. The size of the domain scales with the length of its side, which ranges between $2 \leq n \leq 11$. Formulas (1), (2), and (4) scale in n. Formulas (3) scale with k, the number of nested $\bigcirc$ operators, which we set to half of n rounded down. We have 10 instances of each formula type, for a total of 40 instances.

Domain	I	Coverage MBE	Coverage BE	Avg. RT (secs) MBE	Avg. RT (secs) BE
TRIANGLETIRE-1	10	**10**	**10**	2.82	**0.71**
TRIANGLETIRE-2	10	7	**9**	**82.8**	89.32
TRIANGLETIRE-3	10	**10**	**10**	0.94	**0.88**
TRIANGLETIRE-4	10	8	**10**	**58.17**	70.53
COFFEE-1	15	**13**	12	109.59	**42.15**
COFFEE-2	15	**15**	**15**	0.51	**0.34**
COFFEE-3	15	**12**	**12**	**42.07**	57.36
COFFEE-4	15	**15**	**15**	0.28	**0.26**
Total	100	90	93	-	-

Table 1. Number of solved instances and average runtime (in seconds) of the *mbe*- and *be*-strategy synthesizers in COFFEE and TRIANGLETIRE benchmarks. Column I is the number of instances for each formula type.

In COFFEE benchmarks, we considered the following types of LTL$_f$ goals:

$$\Diamond(o_1) \wedge \cdots \wedge \Diamond(o_n) \quad (1)$$
$$\Diamond(\Diamond(o_1) \vee \cdots \vee \Diamond(o_n)) \quad (2)$$
$$\bigcirc(\bigcirc(\cdots(\bigcirc(o_i)\cdots)) \quad (3)$$
$$\Diamond(o_1 \wedge \Diamond(o_2 \wedge \Diamond(\cdots \wedge \Diamond(o_n)\cdots))) \quad (4)$$

Where o_i denotes that the agent delivered a coffee in office i. To deliver a coffee, the agent must prepare the coffee in the kitchen and navigate to the office where it will deliver the coffee should the environment cooperate, i.e., not cause the agent to fail to deliver the coffee. In the context of COFFEE benchmarks, the *mbe*-strategies are exactly the *be*-strategies: we only use these benchmarks for the purposes of the experimental analysis. The size of the domain in COFFEE benchmarks grows with the number of offices, which ranges between $7 \leq n \leq 21$. Formulas (1), (2), and (4) scale in n; formulas (3) scale in k, the number of nested $\bigcirc$ operators, which we set to n. We have 15 instances for each formula type, for a total of 60 instances.

Experiment Setup. All experiments were run on a laptop with an operating system 64-bit Ubuntu 20.04, 1.80 GHz CPU, and 15.5 GiB of memory. Time-out was set to 1000 seconds.

Experimental Results. Table 1 shows the performance comparison of the *mbe*- and *be*-strategy synthesizers on the considered benchmarks. In TRIANGLETIRE benchmarks, the *mbe*-strategy synthesizer managed to solve almost all instances solved by the *be*-strategy synthesizer; in the solved TRIANGLETIRE instances, the average runtime of the *mbe*- and *be*-strategy are comparable. In COFFEE-benchmarks, the *mbe*-strategy synthesizer solved all instances solved by the *be*-strategy synthesizer and even one more COFFEE-1 instance; in the solved COFFEE instances, the average runtime of the *mbe*-strategy synthesizer is comparable to that of the *be*-strategy synthesizer, except for COFFEE-1 benchmarks: the difference stems since the *mbe*-strategy synthesizer almost times out in the hardest COFFEE-1 instances it solves, which results in an increase in its average runtime. These results confirm that the *mbe*-strategy synthesis brings a minimal overhead compared to *be*-strategy synthesis – which shows the effectiveness of our *mbe*-synthesis algorithm.

7 Conclusion

We introduced *mbe*-strategies for LTL$_f$ goals in FOND domains. An *mbe*-strategy follows, from every history consistent with it, the strategy induced by a *mj*-strategy, and ensures that the agent: (*i*) does its best to achieve the goal; (*ii*) relies the least on the environment's cooperation. We showed that a *mbe*-strategy always exists; provided a correct and optimal game-theoretic algorithm to compute it; and showed that computing a *mbe*-strategy brings a minimal overhead compared to computing a standard *be*-strategy.

There are many open questions regarding *mbe*-strategies, which we plan to investigate in future work and include: the exact relation between *mbe*-strategies and dominant strategies [1]; how to adapt *replanning* and *forward-search* techniques used by state-of-the-art FOND planners and LTL_f synthesizers [26, 24, 22, 17, 25] to compute *mbe*-strategies; and how to compute *mbe*-strategies in the setting of synthesis with LTL/LTL_f goals and environment assumptions [27, 28, 1].

Acknowledgments. This work is supported in part by the ERC Advanced Grant WhiteMech (No. 834228), the PRIN project RIPER (No. 20203FFYLK), the PNRR MUR project FAIR (No. PE0000013), and the UKRI Erlangen AI Hub on Mathematical and Computational Foundations of AI. Gianmarco Parretti is supported by the Italian National Phd on AI at the University of Rome "La Sapienza".

References

[1] B. Aminof, G. De Giacomo, and S. Rubin. Best-effort synthesis: Doing your best is not harder than giving up. In *IJCAI*, pages 1766–1772. ijcai.org, 2021.

[2] B. Aminof, G. De Giacomo, and S. Rubin. Reactive synthesis of dominant strategies. In *AAAI*, pages 6228–6235. AAAI Press, 2023.

[3] C. Baier and J.-P. Katoen. *Principles of Model Checking*. 2008.

[4] S. Bansal, Y. Li, L. M. Tabajara, M. Y. Vardi, and A. M. Wells. Model checking strategies from synthesis over finite traces. In *ATVA (1)*, volume 14215 of *Lecture Notes in Computer Science*, pages 227–247. Springer, 2023.

[5] L. Bonassi, A. E. Gerevini, F. Percassi, and E. Scala. On planning with qualitative state-trajectory constraints in PDDL3 by compiling them away. In *ICAPS*, pages 46–50, 2021.

[6] L. Bonassi, A. E. Gerevini, and E. Scala. Planning with qualitative action-trajectory constraints in PDDL. In *IJCAI*, pages 4606–4613, 2022.

[7] L. Bonassi, G. D. Giacomo, M. Favorito, F. Fuggitti, A. E. Gerevini, and E. Scala. FOND planning for pure-past linear temporal logic goals. In K. Gal, A. Nowé, G. J. Nalepa, R. Fairstein, and R. Radulescu, editors, *ECAI*, pages 279–286, 2023.

[8] A. Camacho and S. A. McIlraith. Strong fully observable non-deterministic planning with LTL and LTL_f goals. In *IJCAI*, pages 5523–5531, 2019.

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

[10] A. Cimatti, M. Pistore, M. Roveri, and P. Traverso. Weak, strong, and strong cyclic planning via symbolic model checking. *Artif. Intell.*, 147 (1-2):35–84, 2003.

[11] G. De Giacomo and M. Favorito. Compositional approach to translate ltlf/ldlf into deterministic finite automata. In *AAAI*, volume 31, pages 122–130, 05 2021. doi: 10.1609/icaps.v31i1.15954.

[12] G. De Giacomo and S. Rubin. Automata-theoretic foundations of FOND planning for LTL_f and LDL_f goals. In *IJCAI*, pages 4729–4735. ijcai.org, 2018.

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

[14] G. De Giacomo and M. Y. Vardi. Synthesis for LTL and LDL on finite traces. In *IJCAI*, pages 1558–1564. AAAI Press, 2015.

[15] G. De Giacomo, G. Parretti, and S. Zhu. LTL_f best-effort synthesis in nondeterministic planning domains. In *ECAI*, pages 533–540, 2023.

[16] G. De Giacomo, G. Parretti, and S. Zhu. Symbolic ltl_f best-effort synthesis. In *EUMAS*, volume 14282 of *Lecture Notes in Computer Science*, pages 228–243. Springer, 2023.

[17] M. Favorito. Forward LTL_f synthesis: DPLL at work. In *IPS-RCRA-SPIRIT@AI*IA*, volume 3585 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2023.

[18] M. Favorito and S. Zhu. LydiaSyft: A compositional symbolic synthesizer for ltl_f specifications. In *SYNTCOMP23*, 2023.

[19] L. Geatti, M. Montali, and A. Rivkin. Foundations of reactive synthesis for declarative process specifications. In *AAAI*, pages 17416–17425, 2024.

[20] H. Geffner and B. Bonet. *A Concise Introduction to Models and Methods for Automated Planning*. Synthesis Lectures on Artificial Intelligence and Machine Learning. Morgan & Claypool Publishers, 2013.

[21] M. Ghallab, D. S. Nau, and P. Traverso. *Acting, Planning and Learning*. Cambridge University Press, 2025.

[22] G. D. Giacomo, M. Favorito, J. Li, M. Y. Vardi, S. Xiao, and S. Zhu. LTL_f synthesis as AND-OR graph search: Knowledge compilation at work. In *IJCAI*, pages 2591–2598, 2022.

[23] P. Haslum, N. Lipovetzky, D. Magazzeni, and C. Muise. *An Introduction to the Planning Domain Definition Language*. 2019.

[24] R. Mattmüller, M. Ortlieb, M. Helmert, and P. Bercher. Pattern database heuristics for fully observable nondeterministic planning. In *ICAPS*, pages 105–112, 2010.

[25] C. Muise, S. A. McIlraith, and J. C. Beck. PRP rebooted: Advancing the state of the art in FOND planning. In *AAAI*, pages 20212–20221. AAAI Press, 2024.

[26] C. J. Muise, S. A. McIlraith, and J. C. Beck. Improved non-deterministic planning by exploiting state relevance. In *ICAPS*, 2012.

[27] A. Pnueli. The temporal logic of programs. In *18th Annual Symposium on Foundations of Computer Science (sfcs 1977)*, pages 46–57. ieee, 1977.

[28] A. Pnueli and R. Rosner. On the synthesis of a reactive module. In *POPL*, pages 179–190. ACM Press, 1989.

[29] F. Somenzi. Cudd: Cu decision diagram package release 2.3. 0. In *University of Colorado at Boulder*, volume 621, 1998.

[30] P. van den Bos and M. Stoelinga. With a little help from your friends: Semi-cooperative games via joker moves. In *FORTE*, volume 13910 of *Lecture Notes in Computer Science*, pages 155–172. Springer, 2023.

[31] S. Zhu, L. M. Tabajara, J. Li, G. Pu, and M. Y. Vardi. Symbolic LTL_f synthesis. In *IJCAI*, pages 1362–1369. ijcai.org, 2017.