Optimal LTL$_f$ Synthesis

Yujian Cao , **Sven Schewe** , **Qiyi Tang** , **Shufang Zhu**
University of Liverpool
{yujian.cao, sven.schewe, qiyi.tang, shufang.zhu}@liverpool.ac.uk

Abstract

Strategy synthesis typically follows an all-or-nothing paradigm, returning unrealisable whenever a specification cannot be guaranteed in an uncertain environment. In this paper, we introduce optimal LTL$_f$ synthesis, where the goal is to realise as many objectives as possible from a given specification consisting of multiple objectives, especially for the case that they are not all jointly realisable. We first consider max-guarantee synthesis, which commits to a maximal set of objectives that we can a priori guarantee to realise. We then introduce max-observation synthesis, which maximises a posteriori realised objectives that may be incomparable on different executions. Finally, we present incremental max-observation synthesis, which further improves strategies by exploiting opportunities for stronger guarantees when they arise during an execution. Experimental results show that different variations of optimal synthesis scale broadly equally well, solving a large fraction of the benchmark instances within the given timeout, demonstrating the practical feasibility of the approach.

1 Introduction

Strategy synthesis is a key problem in Artificial Intelligence (AI), where an autonomous agent must compute a strategy/policy/controller that guarantees task achievement despite uncertainty and nondeterminism in the environment [Reiter, 2001; Ghallab *et al.*, 2016]. Automated synthesis from high-level temporal specifications addresses this challenge by allowing designers to describe *what* the agent should achieve, while the synthesis process determines *how* to do so. This is formalised by reactive synthesis [Pnueli and Rosner, 1989], where the agent interacts with an adversarial environment and must ensure satisfaction of the specification under all possible environment behaviours.

For finite and terminating tasks, Linear Temporal Logic over finite traces (LTL$_f$) [De Giacomo and Vardi, 2013] provides a natural specification language, and LTL$_f$ synthesis [De Giacomo and Vardi, 2015] constructs strategies that guarantee satisfaction of an LTL$_f$ formula along every possible execution. LTL$_f$ synthesis shares deep similarities with planning in fully observable nondeterministic domains (FOND, strong plans) [Cimatti *et al.*, 2003; Geffner and Bonet, 2013].

Standard approaches to LTL$_f$ synthesis [Pnueli and Rosner, 1989; De Giacomo and Vardi, 2015; Zhu *et al.*, 2017; De Giacomo and Rubin, 2018] assume that an agent must satisfy a single LTL$_f$ formula while interacting with a fully adversarial environment [Aminof *et al.*, 2019; Camacho *et al.*, 2019]. Under this assumption, synthesis is typically an all-or-nothing approach: either a strategy that guarantees satisfaction of the specification exists, or the specification is declared unrealisable and no strategy is returned. While this model offers strong correctness guarantees, it is often too restrictive in practice, since real environments are rarely fully adversarial [Aminof *et al.*, 2021; Ghallab *et al.*, 2016] and users may struggle to capture all desired objectives in a single specification [Dimitrova *et al.*, 2020; De Giacomo *et al.*, 2025].

To overcome these limitations, prior work has mainly explored two directions. One direction extends the specification formalism to allow multiple objectives, as in preferences-based planning, such as planning with soft goals [Son and Pontelli, 2006; Bienvenu *et al.*, 2006; Jorge and McIlraith, 2008; Keyder and Geffner, 2009] and maximum realisability for temporal logic specifications [Dimitrova *et al.*, 2020], which focuses on safety properties treated as soft constraints and allows their temporary violation. Another direction relaxes assumptions about the environment: best-effort synthesis [Aminof *et al.*, 2021; De Giacomo *et al.*, 2025] or strong cyclic planning [Cimatti *et al.*, 2003; Geffner and Bonet, 2013], for example, allow agents to exploit cooperative environment behaviours instead of giving up when strict guarantees cannot be enforced.

In this paper, we focus on LTL$_f$ synthesis with multiple objectives and study the problem of *optimal synthesis*, especially when the conjunction of all objectives is not realisable. Here, the synthesis problem is no longer whether a strategy that guarantees all specifications exists, but to find a strategy that guarantees as many of the specifications as possible.

But what makes a strategy optimal? That depends on the circumstances. Consider an agent where we have five goals of equal value, e.g. that a robot eventually visits Rooms 1 through 5. The robot first chooses either a path that allows it to *surely* visit Room 1, or to go down a path where, depending

on a choice of the environment, it can guarantee *either* to visit Rooms 2 and 3 *or* to visit Rooms 4 and 5.

If we only value guarantees that we can give *a priori*, then we would favour being sure to visit Room 1. We refer to this as *max-guarantee synthesis*: max-guarantee synthesis provides the maximal (or maximally valued) subset of objectives that can be jointly realised in the classic sense.

If we want to maximise (the value of) the specifications that we will achieve, on the other hand, we would prefer to satisfy two goals, even if we cannot commit to any of them in particular. We introduce *max-observation* synthesis to address this problem. Instead of committing to a fixed subset of objectives in advance, max-observation synthesis evaluates a strategy based on the objectives it actually satisfies along each execution, and focuses on maximising the number (or value) of achieved objectives.

In the robot example, we would steer the robot towards the point where it can ensure visiting two rooms, albeit which ones is down to the environment. This strategy exploits observed environment behaviour, making max-observation synthesis always at least as good as max-guarantee synthesis. Moreover, our framework naturally supports quantitative weights on objectives, enabling users to express the relative importance of individual objectives and allowing the synthesis of strategies that optimise the total achieved value.

We further improve max-observation synthesis to *incremental* max-observation synthesis, where we want to maximise the ensured number (value) of objectives for every history. In the robot example, the robot might find itself in a situation where, after visiting Room 2, it can choose to continue to Room 3, or go down a route where it can visit Rooms 4 and 5. If we only consider what the strategy can ensure from the beginning of its execution, these routes are of equal value, because *other* runs of the robot curtail this value to 2. In incremental max-observation synthesis, we want more: after each observed history, we seek a strategy that maximises the value that can be ensured from that point onward. In this setting, we would prefer visiting Rooms 4 and 5 over visiting Room 3 only, since—for the given history—this allows us to increase our promise to visit at least three rooms overall.

Both optimisation goals, max-guarantee and max-observation, are natural and have their place: some goals are only worth satisfying if one can guarantee them a priori, while in other cases it is just as good to see them satisfied a posteriori. We thus study both and show that they can be combined.

We have implemented optimal LTL_f synthesis using symbolic techniques [Zhu *et al.*, 2017; Duret-Lutz *et al.*, 2025], including max-guarantee synthesis, max-observation synthesis, and incremental max-observation synthesis. We also evaluated these approaches on benchmarks taken from the LTL_f synthesis track of the annual reactive synthesis competition SyntCOMP (available at https://www.syntcomp.org/), for which we have constructed our own multi-objective specifications. Our empirical results show that the different approaches exhibit broadly comparable performance and solve a large fraction of the benchmarks, producing more informative strategies than simply concluding "unrealisable". Incremental max-observation synthesis shows similar, sometimes better, performance than non-incremental max-observation, demonstrating that strictly better guarantees can be obtained in practice at little additional cost.[1]

2 Preliminaries

A *trace* over an alphabet of symbols Σ is a finite or infinite sequence of elements from Σ. The empty trace is ϵ. Traces are indexed starting at zero, and we write $\pi = \pi_0\pi_1\cdots$. The length of a trace is $|\pi|$. For a finite trace π, we denote by $\mathtt{lst}(\pi)$ the index of the last element of π, i.e. $|\pi| - 1$. For $i \leq \mathtt{lst}(\pi)$, we have that $\pi_i \in 2^{AP}$ is the i-th interpretation of π. By $\pi^k = \pi_0 \cdots \pi_k$ we denote the *prefix* of π up to the k-th iteration.

2.1 LTL_f Basics

Linear Temporal Logic on finite traces (LTL_f) is a specification language to express temporal properties for finite non-empty traces [De Giacomo and Vardi, 2013]. LTL_f shares the syntax with LTL [Pnueli, 1977]. Given a set of atoms AP, the LTL_f formulas over AP are generated as follows:

$$\varphi ::= p \mid \varphi_1 \wedge \varphi_2 \mid \neg\varphi \mid \mathsf{X}^!\varphi \mid \varphi_1 \,\mathsf{U}\, \varphi_2.$$

$p \in AP$ is an *atom*, $\mathsf{X}^!$ (*Next*), and U (*Until*) are temporal operators. We use standard Boolean abbreviations such as $\vee$ (or) and $\supset$ (implies), *tt* (*true*) and *ff* (*false*). Moreover, we define the following abbreviations: *Weak Next* $\mathsf{X}\varphi \equiv \neg\mathsf{X}^!\neg\varphi$, *Eventually* $\mathsf{F}\varphi \equiv tt\,\mathsf{U}\,\varphi$ and *Always* $\mathsf{G}\varphi \equiv \neg\mathsf{F}\neg\varphi$. The size of φ, written $|\varphi|$, is the number of its subformulas. We assume standard semantics from [De Giacomo and Vardi, 2013].

An LTL_f formula φ can be represented by a Deterministic Finite Automaton (DFA) $\mathcal{A}_\varphi = (2^{AP}, \mathcal{Q}, q_0, \delta, F)$, where 2^{AP} is a finite alphabet, $\mathcal{Q}$ is a finite set of states, $q_0 \in \mathcal{Q}$ is the initial state, $\delta : \mathcal{Q} \times 2^{AP} \to \mathcal{Q}$ is the transition function, and $F \subseteq \mathcal{Q}$ is the set of accepting states, such that, for every trace π, we have $\pi \models \varphi$ iff π is accepted by $\mathcal{A}_\varphi$ [De Giacomo and Vardi, 2013].

2.2 LTL_f Synthesis

An LTL_f synthesis specification can be described by a tuple $(\varphi, \mathcal{X}, \mathcal{Y})$, where φ is an LTL_f formula over the propositions in $\mathcal{X} \cup \mathcal{Y}$. $\mathcal{X}$ and $\mathcal{Y}$ are two disjoint sets: $\mathcal{X}$ is the set of input variables controlled by the environment and $\mathcal{Y}$ is the set of output variables controlled by the agent. A strategy is a function $\sigma : (2^{\mathcal{X}})^* \to 2^{\mathcal{Y}}$ that maps each finite sequence of input variables to an output assignment. $\mathbf{X} \in (2^{\mathcal{X}})^\omega$ denotes an infinite sequence of valuations of input variables, and the trace $\pi(\mathbf{X}, \sigma)$ induced by the prefix of $\mathbf{X}$ and σ is:

$$\pi(\mathbf{X}, \sigma) = (X_0 \cup \sigma(\epsilon))(X_1 \cup \sigma(X_0)) \cdots .$$

Definition 1 (Winning strategy)**.** *Given an LTL_f synthesis specification $(\varphi, \mathcal{X}, \mathcal{Y})$, a strategy $\sigma : (2^{\mathcal{X}})^* \to 2^{\mathcal{Y}}$ is a* winning strategy *for φ, denoted $\sigma \triangleright \varphi$, if for every infinite sequence of the input variables $\mathbf{X} \in (2^{\mathcal{X}})^\omega$, the induced trace $\pi(\mathbf{X}, \sigma)$ satisfies φ, i.e. there exists a finite prefix of $\pi(\mathbf{X}, \sigma)$ that satisfies φ.*

[1] The full version of this paper: https://arxiv.org/abs/2605.11544

Definition 2 (Realisability)**.** *An LTL$_f$ synthesis specification $(\varphi, \mathcal{X}, \mathcal{Y})$ is* realisable *if there exists a winning strategy $\sigma : (2^{\mathcal{X}})^* \to 2^{\mathcal{Y}}$ for φ, i.e. $\sigma \triangleright \varphi$. The LTL$_f$ realisability problem is to determine whether $(\varphi, \mathcal{X}, \mathcal{Y})$ is* realisable*.*

Definition 3 (Synthesis)**.** *The synthesis problem consists in computing a winning strategy for a realisable specification.*

Sometimes, for simplicity, we omit $\mathcal{X}$ and $\mathcal{Y}$ when they are clear from the context.

2.3 DFA Games

Classic solutions to LTL$_f$ synthesis rely on a reduction to two-player DFA games (reachability games). A *DFA game* is described as a tuple $\mathcal{G} = (\mathcal{A}, F)$, where $\mathcal{A} = (2^{AP}, \mathcal{Q}, q_0, \delta, -)$ is a DFA representing the game arena of which the set of accepting states does not matter and $F \subseteq \mathcal{Q}$ is the *reachability winning objective*. For LTL$_f$ synthesis, we often take the corresponding DFA of the formula as the game arena and the set of accepting states as the winning objective. Given a play $\pi = (X_0 \cup Y_0)(X_1 \cup Y_1) \ldots \in (2^{\mathcal{X} \cup \mathcal{Y}})^\omega$, running $\mathcal{A}$ on π gives us the infinite sequence $\rho = q_0 q_1 \ldots \in \mathcal{Q}^\omega$ such that q_0 is the initial state and $q_{i+1} = \delta(q_i, X_i \cup Y_i)$ for all $i \geq 0$. Since the transitions in $\mathcal{A}$ are all deterministic, we thus denote by $\rho = \mathrm{run}(\pi, \mathcal{A})$ the unique sequence of running $\mathcal{A}$ on π. Analogously, we denote by $\rho^k = \mathrm{run}(\pi^k, \mathcal{A})$ the unique finite sequence of running π^k on $\mathcal{A}$, and $\rho^k = q_0 q_1 \ldots q_k$. The *reachability winning objective* F indicates a set of desired states to reach for the agent. A play π is a *winning play* with respect to winning objective F if some state $s \in F$ occurs in $\mathrm{run}(\pi, \mathcal{A})$. An agent strategy σ is *winning* in $\mathcal{G} = (\mathcal{A}, F)$ if for every infinite sequence of the input variables $\mathbf{X} \in (2^{\mathcal{X}})^\omega$, the induced play $\pi(\mathbf{X}, \sigma)$ is a winning play with respect to F.

A DFA game with reachability objective $\mathcal{G} = (\mathcal{A}, F)$ can be solved by the following least fixed point computation.

$$\mathrm{Win}_0(\mathcal{G}) = F$$
$$\mathrm{Win}_{i+1}(\mathcal{G}) = \mathrm{Win}_i(\mathcal{G}) \cup \mathrm{PreC}(\mathrm{Win}_i(\mathcal{G}))$$

where $\mathrm{PreC}(S) = \{q \in \mathcal{Q} \mid \exists Y \in 2^{\mathcal{Y}}. \forall X \in 2^{\mathcal{X}}. \delta(q, (X, Y)) \in S\}$ is the *controllable preimage*. The computation stabilises at the least index i such that $\mathrm{Win}_{i+1}(\mathcal{G}) = \mathrm{Win}_i(\mathcal{G})$ and we denote this fixed point by $\mathrm{Win}(\mathcal{G}) = \mathrm{Win}_i(\mathcal{G})$. Every state $q \in \mathrm{Win}(\mathcal{G})$ is a winning state for the agent, from where she has a winning strategy in the game $\mathcal{G}_q = (\mathcal{A}_q, F)$, where $\mathcal{A}_q = (2^{\mathcal{X} \cup \mathcal{Y}}, \mathcal{Q}, q, \delta, -)$, i.e. the same arena $\mathcal{A}$ but with the new initial state q. Intuitively, $\mathrm{Win}(\mathcal{G})$ represents the "agent winning region", from which the agent is able to win the game, no matter how the environment behaves.

3 Optimal Synthesis

We consider a set Φ of LTL$_f$ formulas, of which we intuitively want to satisfy as many as possible. Φ is realisable (resp. unrealisable) if $\bigwedge_{\varphi \in \Phi} \varphi$ is realisable (resp. unrealisable), i.e. there exists (resp. does not exist) a winning strategy for Φ. The case we are interested in is where Φ is unrealisable, but we can realise as large, or important, a subset of Φ as possible. When posing this question, we have to define whether this means *first* selecting the subset $\Psi \subseteq \Phi$ we promise to realise and *then* executing the agent with the goal of realising Ψ (guarantee synthesis), or if we find a strategy whose traces satisfy potentially different subsets $\Psi_1, \ldots, \Psi_n \subseteq \Phi$ of Φ, and we maximise the minimal value of the trace (observation synthesis). We consider both versions as well as a combination, where we maximise a weighted sum of the two values.

Let Φ be a set of LTL$_f$ formulas over the alphabet $2^{\mathcal{X} \cup \mathcal{Y}}$, where $\mathcal{X}$ and $\mathcal{Y}$ are a fixed partition of the atoms into environment input and agent output, respectively. Let $\mathcal{P} = (\Phi, \mathcal{X}, \mathcal{Y})$ be a tuple and $G, V : \Phi \to (0, 1]$ be functions specifying the preference value of every LTL$_f$ formula $\varphi \in \Phi$ as guarantee (G) and observed value (V), respectively.

We first define the *max-guarantee* LTL$_f$ synthesis problem. Abusing the notation slightly, for a strategy $\sigma : (2^{\mathcal{X}})^* \to 2^{\mathcal{Y}}$, we define the *guarantee value* of σ with respect to $\mathcal{P}$ as

$$G(\sigma) = \sum_{\varphi \in \Phi : \sigma \triangleright \varphi} G(\varphi).$$

Definition 4 (Max-guarantee winning strategy)**.** *Given $\mathcal{P} = (\Phi, \mathcal{X}, \mathcal{Y})$ and a guarantee value function G, a strategy $\sigma^* : (2^{\mathcal{X}})^* \to 2^{\mathcal{Y}}$ is a* max-guarantee winning strategy *for $\mathcal{P}$ if*

$$G(\sigma^*) = \max_{\sigma : (2^{\mathcal{X}})^* \to 2^{\mathcal{Y}}} G(\sigma).$$

Definition 5 (Max-guarantee LTL$_f$ synthesis)**.** *Given $\mathcal{P} = (\Phi, \mathcal{X}, \mathcal{Y})$ and a guarantee value function G, the* max-guarantee synthesis problem *is to compute a pair $(\sigma^\star, v^\star)$, where $\sigma^\star : (2^{\mathcal{X}})^* \to 2^{\mathcal{Y}}$ is a strategy and $v^\star \in \mathbb{R}$ is its guarantee value, such that*

$$v^\star = G(\sigma^\star) \qquad \textit{and} \qquad v^\star = \max_{\sigma : (2^{\mathcal{X}})^* \to 2^{\mathcal{Y}}} G(\sigma).$$

Next, we define the *max-observation* LTL$_f$ synthesis problem. For a trace $\pi : (2^{\mathcal{X} \cup \mathcal{Y}})^\omega$, we define the *observed value* of π with respect to $\mathcal{P}$ as

$$V(\pi) = \sum_{\varphi \in \Phi : \pi \models \varphi} V(\varphi),$$

and the observed value of a strategy σ as

$$V(\sigma) = \min_{\mathbf{X} \in (2^{\mathcal{X}})^\omega} V(\pi(\mathbf{X}, \sigma)).$$

Definition 6 (Max-observation winning strategy)**.** *Given $\mathcal{P} = (\Phi, \mathcal{X}, \mathcal{Y})$ and an observed value function V, a strategy $\sigma^* : (2^{\mathcal{X}})^* \to 2^{\mathcal{Y}}$ is a* max-observation winning strategy *for $\mathcal{P}$ if*

$$V(\sigma^*) = \max_{\sigma : (2^{\mathcal{X}})^* \to 2^{\mathcal{Y}}} V(\sigma).$$

Definition 7 (Max-observation LTL$_f$ synthesis)**.** *Given $\mathcal{P} = (\Phi, \mathcal{X}, \mathcal{Y})$ and an observed value function V, the* max-observation synthesis problem *is to compute a pair $(\sigma^\star, v^\star)$, where $\sigma^\star : (2^{\mathcal{X}})^* \to 2^{\mathcal{Y}}$ is a strategy and $v^\star \in \mathbb{R}$ is its observed value, such that*

$$v^\star = V(\sigma^\star) \qquad \textit{and} \qquad v^\star = \max_{\sigma : (2^{\mathcal{X}})^* \to 2^{\mathcal{Y}}} V(\sigma).$$

All of these problems are in the same complexity class as ordinary LTL$_f$ synthesis.

Theorem 1. *The max-guarantee and max-observation LTL$_f$ synthesis problems are 2EXPTIME-complete.*

4 Max-Guarantee Synthesis

We now present a solution to the max-guarantee synthesis problem for an instance $\mathcal{P} = (\Phi, \mathcal{X}, \mathcal{Y})$ with a guarantee value function G. We show that the problem can be reduced to computing a strategy that realises a maximal subset of formulas, termed a *max realisable core*, such that every play induced by the strategy satisfies all formulas in this subset.

Definition 8 (Realisable core)**.** *Let $\Phi = \{\varphi_1, \varphi_2, \cdots, \varphi_n\}$ be a set of LTL$_f$ formulas and $\Psi \subseteq \Phi$ a realisable subset of Φ. Ψ is a realisable core if for every $\varphi_i \in \Phi \setminus \Psi$, $\varphi_i \wedge \bigwedge_{\varphi_j \in \Psi} \varphi_j$ is unrealisable.*

Definition 9 (Max realisable core)**.** *A realisable core $\Psi^* \subseteq \Phi$ is* maximal *if for any other realisable core $\Psi' \subseteq \Phi$, it holds that $\sum_{\varphi \in \Psi^*} G(\varphi) \geq \sum_{\varphi \in \Psi'} G(\varphi)$.*

It is straightforward to see that the max-guarantee synthesis problem, computing the maximal guarantee v^*, can be reduced to computing a maximal realisable core Ψ^*, where $v^* = \sum_{\varphi \in \Psi^*} G(\varphi)$. We solve the max-guarantee synthesis problem in the following steps:

Step 1: Automata Construction. For every LTL$_f$ formula $\varphi_i \in \Phi$, we build its corresponding DFA $\mathcal{A}^i = (2^{\mathcal{X} \cup \mathcal{Y}}, \mathcal{Q}^i, q_0^i, \delta^i, F^i)$, where $F^i \subseteq \mathcal{Q}^i$ is the set of accepting states for φ_i.

Step 2: Game Arena Construction. We build the product automaton $\mathcal{A} = \bigotimes_{\varphi_i \in \Phi} \mathcal{A}^i$, such that $\mathcal{A} = (2^{\mathcal{X} \cup \mathcal{Y}}, \mathcal{Q}, q_0, \delta, \emptyset)$, where $\mathcal{Q} = \times_{\varphi_i \in \Phi} \mathcal{Q}^i$ is the product state space, $q_0 = (q_0^i)_{\varphi_i \in \Phi}$ is the initial state, δ is the product transition function induced by the $\{\delta^i\}_{\varphi_i \in \Phi}$.

Step 3: Guarantee Value-Based Partition. We partition the search space into a sequence of formula subsets ordered by non-increasing guarantee values. For each $\varphi_i \in \Phi$, we define $F^i_\otimes = \{(q_1, \cdots, q_n) \in \mathcal{Q} \mid q_i \in F^i\}$ as the set of states in the product automaton where φ_i is among the satisfied specifications. We then consider the finite set of values $v_1 > v_2 > \cdots > v_\ell$ such that each $v_j = \sum_{\varphi \in \Psi} G(\varphi)$ can take for subsets Ψ of Φ, and define $F_\Psi = \bigcap_{\varphi_i \in \Psi} F^i_\otimes$.

Step 4: Game Solving. We iterate over the value v from Step 3 in non-increasing order. For each Ψ, we construct the DFA game $\mathcal{G}_\Psi = (\mathcal{A}, F_\Psi)$, using the product automaton $\mathcal{A}$ from Step 2 as the game arena and F_Ψ as the set of accepting states. Then we solve the game $\mathcal{G}_\Psi$ by computing its winning set $\mathsf{Win}(\mathcal{G}_\Psi)$. The max-value v^* is the first v that satisfies $q_0 \in \mathsf{Win}(\mathcal{G}_\Psi)$; we return a winning strategy of $\mathcal{G}_\Psi$ where Ψ is a max realisable core. □

The overall synthesis procedure is presented in Algorithm 1. The algorithm iterates through F_Ψ in non-increasing order of guaranteed values (Lines 2-4) to ensure that the first realisable core found is maximal. Once a realisable core Ψ is found via the reachability check (Lines 7-8), the corresponding guaranteed value with a winning strategy is returned.

The correctness of Algorithm 1 is ensured by the guaranteed value-based non-increasing search strategy.

Theorem 2. *Algorithm 1 returns a max-guarantee $v = \sum_{\varphi \in \Psi} G(\varphi)$ where Ψ is a max realisable core. That is, the subset Ψ is realisable, and for any other realisable subset $\Psi' \subseteq \Phi$, it holds that $\sum_{\varphi \in \Psi} G(\varphi) \geq \sum_{\varphi \in \Psi'} G(\varphi)$.*

Algorithm 1 Max-Guarantee Synthesis

Require: A set of LTL$_f$ formulas $\Phi = \{\varphi_1, \cdots, \varphi_n\}$
Ensure: A winning strategy σ^* for a max realisable core $\Psi \subseteq \Phi$, and $\sum_{\varphi \in \Psi} G(\varphi)$
1: Construct the product automaton $\mathcal{A}$ from Φ
2: Partition the search space into a sequence of subsets $\mathcal{S} = (\Psi_1, \Psi_2, \ldots, \Psi_{2^{|\Phi|}})$ where $\sum_{\varphi \in \Psi_i} G(\varphi) \geq \sum_{\varphi \in \Psi_{i+1}} G(\varphi)$ holds for all $i < 2^{|\Phi|}$.
3: **for** j in 1 to $2^{|\Phi|}$ **do**
4: $\quad F_{\Psi_j} \leftarrow \bigcap_{\varphi_i \in \Psi_j} F^i_\otimes$
5: $\quad$ Construct DFA game $\mathcal{G}_{\Psi_j} = (\mathcal{A}, F_{\Psi_j})$
6: $\quad$ Compute winning set $\mathsf{Win}(\mathcal{G}_{\Psi_j})$ and extract a winning strategy σ^*
7: $\quad$ **if** $q_0 \in \mathsf{Win}(\mathcal{G}_{\Psi_j})$ **then**
8: $\quad\quad$ **return** $(\sigma^*, \sum_{\varphi \in \Psi_j} G(\varphi))$
9: $\quad$ **end if**
10: **end for**

Theorem 3. *The running time of Algorithm 1 is $O(2^{|\Phi|} \cdot |\mathcal{A}|)$, where $|\mathcal{A}|$ is the size of the product automaton $\mathcal{A}$.*

We note that we build the arenas $\bigotimes_{\varphi_i \in \Psi} \mathcal{A}^i$ individually for each Ψ. Normally, the structures of the $\mathcal{A}^i$ are such that these automata are substantially smaller than $\mathcal{A}$ for $\Psi \neq \Phi$. In this case the running time for $\Psi = \Phi$ normally dominates.

5 Max-Observation Synthesis

In this section, we first present a basic algorithm for computing a strategy for max-observation synthesis. We then introduce incremental max-observation synthesis, where the synthesis space is updated dynamically with the execution history trace, and show how to adapt the basic algorithm to construct an incremental max-observation strategy, followed by an improved algorithm.

5.1 Basic Algorithm

The basic algorithm for the max-observation synthesis problem begins with automata construction and game arena construction, which coincide with the first two steps of the max-guarantee synthesis procedure described in Section 4. We describe the remaining steps as follows:

Step 3: Target Construction. For each $\varphi_i \in \Phi$, we define $F^i_\otimes = \{(q_1, \cdots, q_n) \in \mathcal{Q} \mid q_i \in F^i\}$ as the set of states in the product automaton where φ_i is among the satisfied specifications. We then consider the finite set of values $v_1 > v_2 > \cdots > v_\ell$ that $\sum_{\varphi \in \Psi} V(\varphi)$ can take for subsets Ψ of Φ, and define $F_v = \bigcup\{F_\Psi \mid \sum_{\varphi \in \Psi} V(\varphi) \geq v\}$ for each of these values, where $F_\Psi = \bigcap_{\varphi_i \in \Psi} F^i_\otimes$.

Step 4: Game Solving. We iterate over the value v from Step 3 in descending order. We construct the DFA game $\mathcal{G}_v = (\mathcal{A}, F_v)$, using the product automaton $\mathcal{A}$ from Step 2 as the game arena and F_v as the reachability winning objective. Then we solve the game $\mathcal{G}_v$ by computing its winning

set $\mathsf{Win}(\mathcal{G}_v)$. The max-value v^* is the first v that satisfies $q_0 \in \mathsf{Win}(\mathcal{G}_v)$; we return a winning strategy of $\mathcal{G}_{v^*}$. □

The following lemma is key to the correctness of the basic algorithm of the max-observation synthesis problem.

Lemma 4. *Let $\mathcal{P} = (\Phi, \mathcal{X}, \mathcal{Y})$ with the observed value function V be a max-observation synthesis problem and $\mathsf{Win}(\mathcal{G}_v)$ the agent winning region of the reachability game $\mathcal{G}_v = (\mathcal{A}, F_v)$, computed as above, where v is an arbitrary value. Then there is a strategy σ with ensured value v, i.e. $V(\sigma) \geq v$ iff $q_0 \in \mathsf{Win}(\mathcal{G}_v)$.*

Theorem 5. *Let $\mathcal{P} = (\Phi, \mathcal{X}, \mathcal{Y})$ with the observed value function V be a max-observation LTL$_f$ synthesis problem and v^* the first value (in descending order) such that $q_0 \in \mathsf{Win}(\mathcal{G}_{v^*})$, where $\mathcal{G}_{v^*} = (\mathcal{A}, F_{v^*})$. Let σ^* be any winning strategy for $\mathcal{G}_{v^*}$ returned by Step 4. Then σ^* is a max-observation winning strategy for $\mathcal{P}$, and the maximal realisable value is v^* (i.e. $V(\sigma^*) = v^*$).*

We conclude by showing that the complexity of the basic algorithm matches that of the max-observation LTL$_f$ synthesis problem.

Theorem 6. *The running time of the basic algorithm is $O(2^{|\Phi|} \cdot |\mathcal{A}|)$, where $|\mathcal{A}|$ is the size of the product automaton $\mathcal{A}$.*

We note that this can be improved to $O(|\Phi| \cdot |\mathcal{A}|)$ (or: $O(\ln(\ell) \cdot |\mathcal{A}|)$) by considering the values along an ordered tree: if there are $f > 1$ feasible values left, we can next test for the $f/2$ largest among them (rounded up or down), reducing the feasible values to $\lceil f/2 \rceil$ and providing $\ln(\ell)$ iterations (e.g. $O(\ln(|\Phi|))$ if all formulas have the same value).

The above basic algorithm solves the max-observation synthesis problem. We extend this algorithm in the next subsection by introducing the concept of *incremental max-observation synthesis*, where the synthesis space is updated dynamically with the execution history trace.

5.2 Incremental Max-Observation Synthesis

The goal of the principled algorithm from the previous section is to provide maximal ensured value. However, after a prefix $h \in (2^{AP})^*$, we might be in a position to provide *better* ensured values. We first define the set of *prefixes* of a strategy σ, denoted $\mathsf{pre}(\sigma) \subseteq (2^{AP})^*$, the set of prefixes of $\{\pi(\mathbf{X}, \sigma) \mid \mathbf{X} \in (2^{\mathcal{X}})^\omega\}$ of runs induced by σ. Conversely, we define strategies compatible with a history $h \in (2^{AP})^*$ as $\mathsf{comp}(h) = \{\sigma \in (2^{\mathcal{X}})^* \to 2^{\mathcal{Y}} \mid h \in \mathsf{pre}(\sigma)\}$. Then for all $h \in \mathsf{pre}(\sigma)$, we define

$$V_h(\sigma) = \min\{V(h \cdot \rho) \mid h \cdot \rho \in \pi(\mathbf{X}, \sigma),\ \mathbf{X} \in (2^{\mathcal{X}})^\omega\}.$$

Definition 10 (Incremental max-observation optimal strategy)**.** *Given $\mathcal{P} = (\Phi, \mathcal{X}, \mathcal{Y})$ with an observed value function V, a strategy $\sigma^* : (2^{\mathcal{X}})^* \to 2^{\mathcal{Y}}$ is* observation optimal *for a history $h \in \mathsf{pre}(\sigma^*)$ for $\mathcal{P}$ if*

$$V_h(\sigma^*) = \max_{\sigma \in \mathsf{comp}(h)} V_h(\sigma)$$

holds, and incremental observation optimal *if it is observation optimal for all histories $h \in \mathsf{pre}(\sigma)$ for $\mathcal{P}$.*

Extending the Basic Algorithm. To synthesise an incremental observationally optimal strategy, we can adapt the basic algorithm from Section 5.1. In particular, in Step 4, after solving each DFA game $\mathcal{G}_v$, we record for every game state q the maximal value v_q such that $q \in \mathsf{Win}(\mathcal{G}_{v_q})$, together with the corresponding local strategy $\sigma_q \in 2^{\mathcal{Y}}$. This local strategy can be viewed as the first move of an optimal strategy for the game $\mathcal{G}_{v_q}$ when the initial game state is replaced by q. Finally, we combine the local strategies σ_q for all game states q into a global strategy $\sigma\colon (2^{\mathcal{X}})^* \to 2^{\mathcal{Y}}$ and show that σ is an incremental observationally optimal strategy.

The correctness of this procedure follows from the fact that whenever a game state q of the product automaton $\mathcal{A}$ is reached under the strategy σ, it follows the corresponding local strategy σ_q, thus achieving the maximal observed value v_q when q is treated as the initial state of the game.

Improved Algorithm. While the basic algorithm can be adapted to compute an incremental max-observation optimal strategy, it is theoretically *inefficient*, as each game state and state-action pair may be considered multiple times: a state $q \in \mathsf{Win}(\mathcal{G}_{v_i})$ also belongs to $\mathsf{Win}(\mathcal{G}_{v_j})$ for all $j > i$. To synthesise an incremental max-observation optimal strategy efficiently, we retain the first two steps of the algorithm and then perform incremental game solving and target construction directly.

Step 3: Incremental Max-Obs Optimal Strategy Computation. We set $W_0 = \emptyset$ and then iteratively calculate targets of increasing size as follows for $k = 1, \ldots, \ell$:

- $F^{=}_{v_k} = \bigcup\{F_\Psi \mid \sum_{\varphi \in \Psi} V(\varphi) = v_k\}$
- $T_k = W_{k-1} \cup F^{=}_{v_k}$, $\mathcal{G}_k = (\mathcal{A}, T_k)$, and $W_k = \mathsf{Win}(\mathcal{G}_k)$

until $q_0 \in W_k$.

We use the attractor strategies to each target T_k to define σ^* (and fix it arbitrarily elsewhere). □

The correctness of the improved algorithm is based on two facts: (1) $W_k = \mathsf{Win}(\mathcal{G}_{v_k})$ where $\mathcal{G}_{v_k} = (\mathcal{A}, F_{v_k})$ is the DFA game constructed in Step 4 of Section 5.1; and (2) for any history h, if the maximal achievable value has not already been observed along h, then the state q_h reached by h must belong to $\mathsf{Win}(\mathcal{G}_{v^*})$ to enforce v^*.

Theorem 7. *Let $\mathcal{P} = (\Phi, \mathcal{X}, \mathcal{Y})$ with an observed value function V be an incremental max-observation LTL$_f$ synthesis problem and v_{k^*} the first value (in descending order) such that $q_0 \in \mathsf{Win}(\mathcal{G}_{k^*})$, and σ^* be a strategy returned by our algorithm. Then σ^* is incremental observation optimal.*

While the improved algorithm works on the same arena as the algorithm described in the previous section, it never considers a state-action pair twice. When explicitly constructing the arena $\mathcal{A}$, the sets $F^{=}_{v_k}$ can be built with it. In this case, each state is considered at most twice across all fixed point computations: once when added to one of the $F^{=}_{v_k}$ and once when added to a winning region during fixed point construction.

Theorem 8. *The running time of the improved algorithm is $O(|\mathcal{A}|)$, where $|\mathcal{A}|$ is the size of the product automaton $\mathcal{A}$.*

This is in contrast to the principled algorithm from the previous section, where every state and state-action pair can be considered once per fixed point construction.

As an additional computational advantage, computing $\mathsf{Win}(\mathcal{G}_k)$ could take fewer steps than computing $\mathsf{Win}(\mathcal{G}_{v_k})$ since we start with a larger target ($T_k \supseteq F_{v_k}$).

5.3 Combination with Max-Guarantee

It is straightforward to combine (incremental) max-observation with max-guarantee: to compute the (incremental) max-observation value subject to guaranteeing a given subset $\Gamma \subseteq \Phi$ of Φ, we can simply define $F_{v,\Gamma} = \bigcup\{F_\Psi \mid \sum_{\varphi\in\Psi} V(\varphi) \geq v,\ \Psi \supseteq \Gamma\}$ and $F^{=}_{v,\Gamma} = \bigcup\{F_\Psi \mid \sum_{\varphi\in\Psi} V(\varphi) = v,\ \Psi \supseteq \Gamma\}$.

We can then find optimal solutions and values v^*_Ψ for (incremental) max-observation for each $\Psi \supseteq \Gamma$ individually for realisable Ψ (note that, for $\Psi \neq \emptyset$, Ψ is realisable iff $v^*_\Psi > 0$ holds, so that the realisability of Ψ is obtained as a side product and does not have to be tested individually), and find the best value $v^*_\Psi + \sum_{\varphi\in\Psi} G(\varphi)$ for realisable Ψ.

6 Symbolic Optimal Synthesis

In this section, we introduce symbolic techniques of our proposed approaches to optimal LTL_f synthesis. The symbolic techniques below are faithful encodings of the explicit constructions, following the symbolic framework of [Zhu *et al.*, 2017]. Their correctness follows from Theorems 2, 5, and 7.

Symbolic DFA Games. We utilise the *symbolic* techniques in [Zhu *et al.*, 2017], where the symbolic representation of a DFA $\mathcal{A} = (2^{AP}, \mathcal{Q}, q_0, \delta, -)$ is a tuple $\mathcal{A}_s = (\mathcal{X}, \mathcal{Y}, \mathcal{Z}, I, \eta)$, where $\mathcal{Z}$ is a set of variables such that $|\mathcal{Z}| = \lceil \log |\mathcal{Q}| \rceil$, and every state $q \in \mathcal{Q}$ corresponds to an interpretation $Z \in 2^{\mathcal{Z}}$ over $\mathcal{Z}$. I is a Boolean formula satisfied only by the interpretation of the initial state q_0. $\eta\colon 2^{\mathcal{X}} \times 2^{\mathcal{Y}} \times 2^{\mathcal{Z}} \to 2^{\mathcal{Z}}$ is a Boolean function representing the transition function such that $\eta = \langle \eta_{z_0}, \cdots, \eta_{z_m} \rangle$, where $\eta_z\colon 2^{\mathcal{X}} \times 2^{\mathcal{Y}} \times 2^{\mathcal{Z}} \to \{0,1\}$. Accordingly, we denote the symbolic reachability game as $\mathcal{G}_s = (\mathcal{A}_s, g)$, where g is a Boolean formula over $\mathcal{Z}$ representing the reachability objective.

The game can be solved by symbolic fixpoint computation over two Boolean formulas w over $\mathcal{Z}$ and t over $\mathcal{Z} \cup \mathcal{Y}$, which represent the agent winning region and winning state-action pairs, respectively. The initial step is $w_0(\mathcal{Z}) = g(\mathcal{Z})$ and $t_0(\mathcal{Z}, \mathcal{Y}) = g(\mathcal{Z})$. t_{i+1} and w_{i+1} are constructed as follows:

$$t_{i+1}(\mathcal{Z}, \mathcal{Y}) = t_i(\mathcal{Z}, \mathcal{Y}) \vee (\neg w_i(\mathcal{Z}) \wedge \forall X. w_i(\eta(\mathcal{Z}, \mathcal{X}, \mathcal{Y})))$$
$$w_{i+1}(\mathcal{Z}) = \exists Y. t_i(\mathcal{Z}, \mathcal{Y})$$

The computation terminates when $I \models w_i$. Then, we can use t_{i+1} to compute a winning strategy through the mechanism of Boolean functional synthesis [Fried *et al.*, 2016]. Details of each step can be found in [Zhu *et al.*, 2017].

6.1 Symbolic Max-Guarantee Synthesis

The symbolic characterisation of max-guarantee synthesis from Section 4 is relatively straightforward. Given a set of LTL_f objectives $\Phi = \{\varphi_1, \cdots, \varphi_n\}$, for each φ_i we construct a symbolic DFA $\mathcal{A}^i_s = (\mathcal{X}, \mathcal{Y}, \mathcal{Z}^i, I^i, \eta^i, g^i)$. The symbolic product automaton $\mathcal{A}_s = (\mathcal{X}, \mathcal{Y}, \mathcal{Z}, I, \eta, -)$ can be constructed on the fly by taking $\mathcal{Z} = \bigcup_i \mathcal{Z}^i$, $I = \bigwedge_i I^i$, and defining the transition function as $\eta(\mathcal{Z}, \mathcal{X}, \mathcal{Y}) = \bigwedge_i \eta^i(\mathcal{Z}^i, \mathcal{X}, \mathcal{Y})$. Since each $\eta^i = \langle \eta^i_{z_0}, \cdots, \eta^i_{z_m} \rangle$ is represented as a vector of Boolean update formulas over the state variables in $\mathcal{Z}^i$, the transition function η of the product automaton can be obtained simply by collecting these update formulas. Then, max-guarantee synthesis reduces to solving a sequence of symbolic DFA games $\mathcal{G}_\Psi = (\mathcal{A}_s, g_\Psi)$, where $g_\Psi = \bigwedge_{\psi_i\in\Psi} g^i$ following the guarantee value-based partition as in Step 4 of Section 4.

6.2 Symbolic Max-Observation Synthesis

We follow the same DFA construction and symbolic product automaton construction as in max-guarantee synthesis. For each value v, the reachability objective of the corresponding symbolic game is $g_v = \bigvee_{\Psi\subseteq\Phi:\sum_{\varphi\in\Psi} V(\varphi)\geq v} \bigwedge_{\varphi_i\in\Psi} g^i$. We then solve a sequence of reachability games following the descending order of v as in Step 4 of Section 5.1.

Symbolic Incremental Max-Observation Synthesis. *Extending the basic max-observation synthesis algorithm* to obtain an incremental max-observation optimal strategy relies on recording and combining winning information across values. Note that solving each game $\mathcal{G}_v$ produces two Boolean formulas: $w_v(\mathcal{Z})$ encoding the winning region $\mathrm{Win}(\mathcal{G}_v)$ and $t_v(\mathcal{Z}, \mathcal{Y})$, encoding winning state-action pairs. Instead of explicitly associating each state q with its maximal achievable value, we collect the pairs (w_v, t_v) obtained in descending order of v and construct an incremental observationally optimal strategy by prioritising higher values. We start from the winning state-action pairs of the highest value v_1 and iteratively add new winning state-action pairs from lower-value games only for states not already winning at a higher value, i.e. $t = t_{v_1} \vee \bigvee_{1<k\leq v^*}(t_{v_k} \wedge \neg w_{v_{k-1}})$. The resulting formula t thus compactly represents incremental max-observation optimal strategies, and we then can utilise Boolean functional synthesis to obtain a single strategy from t.

Symbolic *improved incremental max-observation algorithm* performs incremental game solving directly on growing target sets. Each target set T_k is represented by a Boolean formula $g_{T_k} = w_{k-1} \vee g^{=}_{v_k}$, where $g^{=}_{v_k}$ encodes the states in $F^{=}_{v_k}$ and w_{k-1} is the winning region computed at the previous iteration. Since t_k now encode local strategies for all winning states w_k, incremental max-observation optimal strategies can be obtained by simply combining the collected winning state-action pairs t_k from all iterations as $t = t_1 \vee \bigvee_{k>1}(t_k \wedge \neg w_{k-1})$. Note that, although the construction of t is syntactically identical to that used in the extension of the basic max-observation algorithm, the underlying reason differs. In the extension, $t_{v_k} \subseteq t_{v_{k+1}}$ holds due to repeated solving on nested targets, whereas this inclusion does not hold in the improved algorithm, since each iteration already includes the previous winning region in its target. Therefore, the term $(t_k \wedge \neg w_{k-1})$ is necessary to discard vacuous tt agent actions introduced at the initial fixpoint step of game $\mathcal{G}_k$. The final strategy σ^* is then synthesised from t using Boolean functional synthesis.

7 Experimental Evaluation

Implementation. We implemented a prototype tool, called OPTLTLFSYNT, to evaluate our approaches. OPTLTLFSYNT

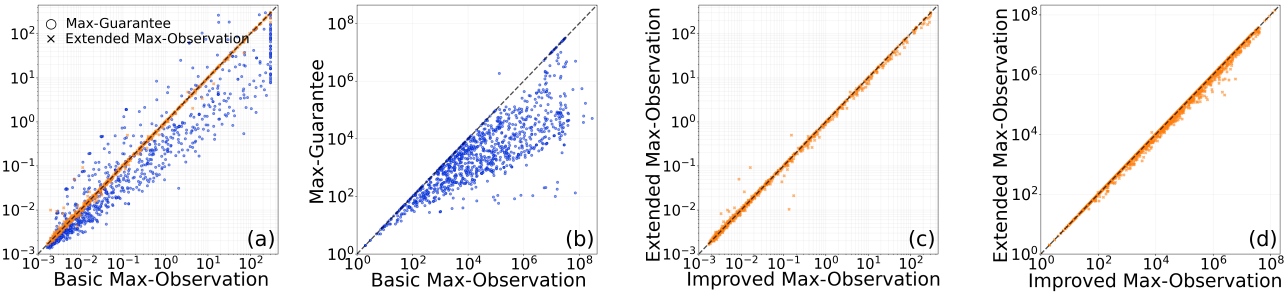

Figure 1: (a) Runtime: basic max-observation (x-axis) vs. max-guarantee and extended max-observation (y-axis). (b) Maximum BDD size during fixpoint (preimage) computations in symbolic game solving: basic max-observation vs. max-guarantee. (c) Runtime: improved max-observation (x-axis) vs. extended max-observation (y-axis). (d) Maximum BDD size during preimage computations in symbolic game solving: extended max-observation vs. improved max-observation. Each point corresponds to one benchmark instance in (a)-(c) and one fixpoint step (preimage computation) in (d); points below the diagonal indicate better performance of the method on the y-axis.

uses Spot [Duret-Lutz *et al.*, 2025] for state-of-the-art LTL_f-to-DFA translation, and relies on the CUDD 3.0.0 [Somenzi, 2016] BDD library for symbolic synthesis, including Boolean formula representation and manipulation.

Benchmarks. We evaluate our approach on benchmarks taken from the LTL_f synthesis track of SyntCOMP (https://www.syntcomp.org/). We restrict our evaluation to instances that are unrealisable and conjunction-decomposable, i.e. specifications given as a conjunction of multiple individual LTL_f formulas, since our goal is to synthesise strategies that satisfy as many individual specifications as possible. Therefore, we have 1145 benchmark instances, in total. The number of conjuncts per instance ranges from 2 to 50.

Experiment setup. All tests were run on a MacBook with an Apple M4 chip (10 physical cores, up to 4.41 GHz) and 16 GB of RAM. The timeout was set to 300 seconds.

Evaluation results are shown in Figure 1. Generally, different methods work broadly equally fast, with a relatively small factor between them and different approaches perform faster on different instances. We have four approaches: *max-guarantee*, *basic max-observation*, *extended max-observation*, and *improved max-observation* (the latter two are for incremental max-observation synthesis). Within the 300s timeout, these approaches can solve 1060, 1005, 1006, 1008 instances, respectively, out of 1145 benchmarks with 30 instances timing out during individual LTL_f-to-DFA construction, demonstrating the practical feasibility of our optimal LTL_f synthesis techniques.

Specifically, max-observation achieves a strictly larger ensured value than that returned by max-guarantee synthesis on 16 instances. This confirms that optimising achieved value rather than a maximal realisable core can sometimes lead to realising more potential. As shown in Figure 1(a), max-guarantee is palpably faster than max-observation on a noticeable subset of instances. We believe that this is because subsets of specifications induce much smaller symbolic arenas due to the on-the-fly symbolic automata product. Max-observation, on the other hand, has to always work on the full arena (but can generally realise more specifications and needs fewer calls), hence dealing with more complex game solving. Moreover, the target sets also become complex. In particular, timeouts occurred in cases where the target required satisfying, for example, 16 out of 20 objectives, which results in 4845 possible combinations. The corresponding target is a large disjunction of individual target sets, and the resulting BDDs (together with their successively constructed controlled attractors) can be large, making symbolic operations costly, as shown in Figure 1(b).

Figure 1(a) also shows that extending basic max-observation to incremental strategies incurs only minor overhead, and in some cases even improves performance. This is again explained by the symbolic nature of the implementation, where strategy extraction relies on BDD-based Boolean functional synthesis. The iterative construction of t (see Section 6.2) can produce smaller and more structured BDD representations, which in turn accelerates strategy extraction.

Figure 1(c) compares the extension of basic max-observation with the improved max-observation algorithm. We can see that, despite solving slightly fewer instances, the extension can sometimes outperform the improved algorithm in runtime. This is, again, because the improved algorithm works on larger target sets in each game iteration, which can reduce the number of fixpoint iterations but makes each fixpoint (preimage) step more expensive in practice (see Figure 1(d)), despite its stronger theoretical guarantees.

The full version provides detailed experimental results.

8 Conclusions

We have studied optimal synthesis for LTL_f specifications in cases where the conjunction of objectives is unrealisable. We introduced max-guarantee, max-observation, and incremental max-observation synthesis to capture different optimisation goals and to allow strategies to exploit the observed behaviour of the environment instead of always assuming a purely adversarial one. Our experimental evaluation on a prototype implementation shows that optimal LTL_f synthesis has practical potential, with good scalability and only minor overhead. This work is a first step towards addressing unrealisability in LTL_f synthesis. For future work, we plan to investigate on-the-fly techniques to further improve efficiency and extensions to synthesis under partial observability.

Acknowledgements

This work was supported by the EPSRC through the projects TRUSTED (EP/X03688X/1) and Games for Good (EP/X042596/1).

References

[Aminof *et al.*, 2019] Benjamin Aminof, Giuseppe De Giacomo, Aniello Murano, and Sasha Rubin. Planning under LTL environment specifications. In *ICAPS*, pages 31–39, 2019.

[Aminof *et al.*, 2021] Benjamin Aminof, Giuseppe De Giacomo, and Sasha Rubin. Best-Effort Synthesis: Doing your best is not harder than giving up. In *IJCAI*, pages 1766–1772, 2021.

[Bienvenu *et al.*, 2006] Meghyn Bienvenu, Christian Fritz, and Sheila A. McIlraith. Planning with qualitative temporal preferences. In *KR*, 2006.

[Camacho *et al.*, 2019] Alberto Camacho, Meghyn Bienvenu, and Sheila A. McIlraith. Towards a unified view of AI planning and reactive synthesis. In *ICAPS*, pages 58–67. AAAI Press, 2019.

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

[De Giacomo and Rubin, 2018] Giuseppe De Giacomo and Sasha Rubin. Automata-theoretic foundations of FOND planning for LTL_f and LDL_f goals. In *IJCAI*, pages 4729–4735, 2018.

[De Giacomo and Vardi, 2013] Giuseppe De Giacomo and Moshe Y. Vardi. Linear Temporal Logic and Linear Dynamic Logic on Finite Traces. In *IJCAI*, 2013.

[De Giacomo and Vardi, 2015] Giuseppe De Giacomo and Moshe Y. Vardi. Synthesis for LTL and LDL on finite traces. In *IJCAI*, 2015.

[De Giacomo *et al.*, 2025] Giuseppe De Giacomo, Gianmarco Parretti, and Shufang Zhu. LTL_f adaptive synthesis for multi-tier goals in nondeterministic domains. *ICAPS*, pages 334–342, 2025.

[Dimitrova *et al.*, 2020] Rayna Dimitrova, Mahsa Ghasemi, and Ufuk Topcu. Reactive synthesis with maximum realizability of linear temporal logic specifications. *Acta Informatica*, 57(1–2):107–135, 2020.

[Duret-Lutz *et al.*, 2025] Alexandre Duret-Lutz, Shufang Zhu, Nir Piterman, Giuseppe De Giacomo, and Moshe Y. Vardi. Engineering an LTL_f synthesis tool. In *CIAA*, pages 129–147, 2025.

[Fried *et al.*, 2016] Dror Fried, Lucas M. Tabajara, and Moshe Y. Vardi. BDD-Based boolean functional synthesis. In *CAV*, pages 402–421, 2016.

[Geffner and Bonet, 2013] Hector Geffner and Blai Bonet. *A Concise Introduction to Models and Methods for Automated Planning*. Morgan & Claypool Publishers, 2013.

[Ghallab *et al.*, 2016] Malik Ghallab, Dana S. Nau, and Paolo Traverso. *Automated planning and acting*. Cambridge, 2016.

[Jorge and McIlraith, 2008] A. Jorge and Sheila A. McIlraith. Planning with preferences. *AI Magazine*, 29(4):25, 2008.

[Keyder and Geffner, 2009] Emil Keyder and Hector Geffner. Soft goals can be compiled away. *Journal of Artificial Intelligence Research*, 36:547–556, 2009.

[Pnueli and Rosner, 1989] Amir Pnueli and Roni Rosner. On the synthesis of a reactive module. In *POPL*, 1989.

[Pnueli, 1977] Amir Pnueli. The temporal logic of programs. In *FOCS*, pages 46–57, 1977.

[Reiter, 2001] Raymond Reiter. *Knowledge in Action: Logical Foundations for Specifying and Implementing Dynamical Systems*. MIT, 2001.

[Somenzi, 2016] Fabio Somenzi. CUDD: CU decision diagram package 3.0.0. University of Colorado at Boulder. 2016.

[Son and Pontelli, 2006] Tran Cao Son and Enrico Pontelli. Planning with preferences using logic programming. *Theory and Practice of Logic Programming*, 6(5):559–607, 2006.

[Zhu *et al.*, 2017] Shufang Zhu, Lucas M. Tabajara, Jianwen Li, Geguang Pu, and Moshe Y. Vardi. Symbolic LTL_f synthesis. In *IJCAI*, pages 1362–1369, 2017.