Symbolic Approaches to LTL_f Best-Effort Synthesis

Giuseppe De Giacomo , **Gianmarco Parretti** and **Shufang Zhu**
Sapienza Univ. Roma, Rome, Italy
{degiacomo, zhu}@diag.uniroma1.it, parretti.1650564@studenti.uniroma1.it

Abstract

We consider an agent acting to fulfil tasks in a nondeterministic environment. When a strategy that fulfills the task regardless of how the environment acts does not exist, the agent should at least avoid adopting strategies that prevent from fulfilling its task. Best-effort synthesis captures this intuition. In this paper, we devise and compare various symbolic approaches for best-effort synthesis in Linear Temporal Logic on finite traces (LTL_f). These approaches are based on the same basic components, however they change in how these components are combined, and this has a significant impact on the performance of the approaches as confirmed by our empirical evaluations.

1 Introduction

We consider an agent acting to fulfil tasks in a nondeterministic environment, as considered in Planning in nondeterministic (adversarial) domains [Cimatti *et al.*, 2003; Ghallab *et al.*, 2004], except that we specify both the environment and the task in Linear Time Logic (LTL) [Aminof *et al.*, 2019], the formalism typically used for specifying complex dynamic properties in Formal Methods [Baier *et al.*, 2008].

In fact, in this paper we consider Linear Time Logic on finite traces (LTL_f) [De Giacomo and Vardi, 2013; De Giacomo and Vardi, 2015], which maintains the syntax of LTL [Pnueli, 1977], but is interpreted on finite traces. In this setting, we study synthesis [Pnueli and Rosner, 1989; Finkbeiner, 2016; De Giacomo and Vardi, 2015; Aminof *et al.*, 2019], a general form of planning. In particular, we look at how to synthesize a strategy that is guaranteed to satisfy the task against all environment behaviors that conform to the environment specification.

When a winning strategy that fulfills the agent task regardless of how the environment acts does not exist, the agent should at least avoid adopting strategies that prevent from fulfilling its task. Best-effort synthesis captures this intuition. More precisely, best-effort synthesis captures the game-theoretic rationality principle that a player would not use a strategy that is "dominated" by another of its strategies (i.e., if the other strategy would fulfil the task against more environment behaviors than the one chosen by the player). Aminof *et al.* [2021b] studied best effort strategies. Best-effort strategies have some notable properties: (*i*) they always exist, (*ii*) if a winning strategy exists then best-effort strategies are exactly the winning strategies, (*iii*) best-effort strategies can be computed in 2EXPTIME as computing winning strategies (best-effort synthesis is indeed 2EXPTIME-complete).

Aminof *et al.* [2021b] presented algorithms for best-effort synthesis in LTL and LTL_f. These algorithms are based on creating, solving and combining the solutions of three distinct games, but of the same game arena. The arena is obtained from the automata corresponding to the formulas $\mathcal{E}$ and φ constituting the environment and the task specifications.

In particular, the algorithm for LTL_f best-effort synthesis appears to be quite promising in practice, since well-performing techniques for each component of the algorithm are available in the literature. These components are: (*i*) transformation of the LTL_f formulas $\mathcal{E}$ and φ into deterministic finite automata (DFA), which can be double-exponential in the worst case, but for which various good techniques have been developed [Henriksen *et al.*, 1995; Zhu *et al.*, 2017; Bansal *et al.*, 2020; De Giacomo and Favorito, 2021b]; (*ii*) Cartesian product of DFAs, which is polynomial; (*iii*) minimization of DFAs, which is also polynomial; (*iv*) fixpoint computation over DFA to compute adversarial and cooperative winning strategies for reaching the final states, which is again polynomial.

In this paper, we refine the LTL_f best-effort synthesis presented by Aminof *et al.* [2021b] by using symbolic techniques [Bryant, 1992; Baier *et al.*, 2008; Zhu *et al.*, 2017]. In particular, we show three different symbolic approaches that combine the above operations in different ways (and in fact allow for different level of minimization). We then compare the three approaches through empirical evaluations. From this comparison, a clear winner emerges. Interestingly, the winner does not fully exploit DFA minimization to minimize the DFA whenever it is possible. Instead, this approach uses uniformly the same arena for all the three games (hence giving up on minimization at some level). Finally, it turns out that the winner performs better in computing best-effort solutions even than state-of-the-art tools that compute only adversarial and cooperative solutions. These findings confirm that LTL_f best-effort synthesis is indeed well suited for efficient and scalable implementations.

The rest of the paper is organized as follows. In Section 2, we remind the main notions of LTL$_f$ synthesis, and LTL$_f$ synthesis under environment assumption. In Section 3, we discuss LTL$_f$ best-effort synthesis, and the algorithm by Aminof *et al.* [2021b]. In Section 4, we introduce three distinct symbolic approaches for LTL$_f$ best-effort synthesis: the first (c.f., Subsection 4.2) is a direct symbolic implementation of the algorithm by [Aminof *et al.*, 2021b]; the second one (c.f., Subsection 4.3) favors maximally conducting DFA minimization, thus getting the smallest possible arenas for the three games; and the third one (c.f., Subsection 4.4) gives up DFA minimization at some level, and creates a single arena for the three games. In Section 5, we perform an empirical evaluation of the three algorithms. We conclude the paper in Section 6.

2 Preliminaries

LTL$_f$ Basics. *Linear Temporal Logic on finite traces* (LTL$_f$) is a specification language to express temporal properties on finite traces [De Giacomo and Vardi, 2013]. In particular, LTL$_f$ has the same syntax as LTL, which is instead interpreted over infinite traces [Pnueli, 1977]. Given a set of propositions $Prop$, LTL$_f$ formulas are generated as follows:

$$\varphi ::= a \mid (\varphi \wedge \varphi) \mid (\neg\varphi) \mid (\circ\varphi) \mid (\varphi\,\mathcal{U}\,\varphi).$$

$a \in Prop$ is an *atom*, $\circ$ (*Next*), and $\mathcal{U}$ (*Until*) are temporal operators. We make use of standard Boolean abbreviations such as $\vee$ (or) and $\rightarrow$ (implies), $true$ and $false$. In addition, we define the following abbreviations *Weak Next* $\bullet\varphi \equiv \neg\circ\neg\varphi$, *Eventually* $\Diamond\varphi \equiv true\,\mathcal{U}\,\varphi$ and *Always* $\Box\varphi \equiv false\,\mathcal{R}\,\varphi$, where $\mathcal{R}$ is for *Release*.

A *trace* $\pi = \pi_0\pi_1\ldots$ is a sequence of propositional interpretations (sets), where for every $i \geq 0$, $\pi_i \in 2^{Prop}$ is the i-th interpretation of π. Intuitively, π_i is interpreted as the set of propositions that are $true$ at instant i. We denote the last instant (i.e., index) in a trace π by $\mathsf{lst}(\pi)$. A trace π is an *infinite* trace if $\mathsf{lst}(\pi) = \infty$, which is formally denoted as $\pi \in (2^{Prop})^\omega$; otherwise π is a *finite* trace, denoted as $\pi \in (2^{Prop})^*$. Moreover, by $\pi^k = \pi_0 \cdots \pi_k$ we denote the *prefix* of π up to the k-th iteration, and $\pi^k = \epsilon$ denotes an empty trace if $k < 0$. LTL$_f$ formulas are interpreted over finite, nonempty traces. Given π, we define when an LTL$_f$ formula φ *holds* at instant i, $0 \leq i \leq \mathsf{lst}(\pi)$, written as $\pi, i \models \varphi$, inductively on the structure of φ, as:

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

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

Reactive Synthesis Under Environment Assumptions. Reactive synthesis is the problem of finding a strategy that allows an agent to achieve a goal while continuously interacting with the external environment. In reactive synthesis under environment assumptions, we can intuitively think of the environment assumption as the knowledge the agent knows a-priori about how the environment can enact. Therefore, the environment behaviors are restricted to those that are consistent with the environment assumption. In this work, we specify environment assumptions and agent goals as LTL$_f$ formulas $\mathcal{E}$ and φ, respectively. We denote the variables under the control of the environment and the agent by $\mathcal{X}$ and $\mathcal{Y}$, respectively ($\mathcal{X}$ and $\mathcal{Y}$ are disjoint).

An *agent strategy* is a function $\sigma_{ag} : (2^{\mathcal{X}})^+ \rightarrow 2^{\mathcal{Y}}$ that maps non-empty sequences of environment choices to an agent choice. Similarly, an *environment strategy* is a function $\sigma_{env} : (2^{\mathcal{Y}})^* \rightarrow 2^{\mathcal{X}}$ mapping (possibly empty) sequences of agent choices to an environment choice. A play $\pi = (X_0 \cup Y_0)(X_1 \cup Y_1)\ldots$ is σ_{ag}-*consistent* if $Y_k = \sigma_{ag}(X_0X_1\ldots X_k)$ for all $k \geq 0$. Similarly, π is σ_{env}-*consistent* if $X_0 = \sigma_{env}(\epsilon)$, where ϵ denotes the empty trace, and $X_k = \sigma_{env}(Y_0Y_1\ldots Y_{k-1})$ for all $k > 0$. We denote by $\pi(\sigma_{ag}, \sigma_{env})$ the unique trace that is consistent with strategies σ_{ag} and σ_{env}.

Consider LTL$_f$ formula φ over $\mathcal{X} \cup \mathcal{Y}$, we say that agent strategy σ_{ag} *realizes* φ, denoted as $\sigma_{ag} \rhd \varphi$, if $\pi(\sigma_{ag}, \sigma_{env}) \models \varphi$ holds for every environment strategy σ_{env}. φ is *agent realizable* if there exists an agent strategy σ_{ag} that realizes φ. *Environment realizability* is defined similarly. Furthermore, we say that agent strategy σ_{ag} *cooperatively realizes* φ, denoted as $\sigma_{ag} \blacktriangleright \varphi$, if there exists an environment strategy σ_{env} such that $\pi(\sigma_{ag}, \sigma_{env}) \models \varphi$. Hence, φ is *agent cooperatively realizable* if such an agent strategy exists. *Cooperative environment realizability* is defined similarly.

We define the problem of LTL$_f$ reactive synthesis under environment assumptions as follows.

Definition 1. *The* LTL$_f$ *reactive synthesis under environment assumptions problem is defined as a pair $\mathcal{P} = (\mathcal{E}, \varphi)$, where* LTL$_f$ *formula $\mathcal{E}$ is an environment assumption, and* LTL$_f$ *formula φ is the agent goal. Realizability of $\mathcal{P}$ checks whether there exists an agent strategy σ_{ag} that realizes φ under $\mathcal{E}$, i.e.,*

$$\forall \sigma_{env} \rhd \mathcal{E}, \pi(\sigma_{ag}, \sigma_{env}) \models \varphi$$

Synthesis of $\mathcal{P}$ computes such a strategy if exists.

A naive approach to this problem is a reduction to standard synthesis of LTL$_f$ formula $\mathcal{E} \rightarrow \varphi$ [Aminof *et al.*, 2018]. Moreover, it has been shown that the problem of LTL$_f$ reactive synthesis under environment assumptions is 2EXPTIME-complete [Aminof *et al.*, 2018].

3 Best-effort Synthesis Under Environment Assumptions

Reactive synthesis, in general, considers that the environment exhibits an adversarial behavior, thus aims at computing an agent strategy such that no matter how the environment behaves, the agent will achieve the goal. In this setting, the agent, instead of keep trying, just give up when the synthesis procedure returns *unrealizable*, although the environment can be possibly "over-approximated". In this section, we synthesize a strategy ensuring that the agent will do nothing that would needlessly prevent it from achieving its task – which we call a best-effort strategy. *Best-effort synthesis* is the problem of finding such a strategy [Aminof *et al.*, 2021b]. As for reactive synthesis under environment assumptions, in best-effort synthesis environment behaviors are restricted to those that are consistent with the environment assumption.

We start with showing how a strategy makes more effort with respect to another. Let $\mathcal{E}$ and φ be two LTL$_f$ formulas denoting an environment assumption and an agent goal, respectively, and σ_1 and σ_2 be two agent strategies. We say that σ_1 *dominates* σ_2 for goal φ under $\mathcal{E}$, written $\geq_{\varphi|\mathcal{E}}$, if for every $\sigma_{env} \rhd \mathcal{E}, \pi(\sigma_2, \sigma_{env}) \models \varphi$ implies $\pi(\sigma_1, \sigma_{env}) \models \varphi$. Furthermore, we say that σ_1 *strictly dominates* σ_2, written $\sigma_1 >_{\varphi|\mathcal{E}} \sigma_2$, if $\sigma_1 \geq_{\varphi|\mathcal{E}} \sigma_2$ and $\sigma_2 \not\geq_{\varphi|\mathcal{E}} \sigma_1$. An agent strategy σ^* is best-effort, or maximal, for the goal φ under the environment assumption $\mathcal{E}$, if there is no agent strategy σ such that $\sigma >_{\varphi|\mathcal{E}} \sigma^*$ Intuitively, $\sigma_1 >_{\varphi|\mathcal{E}} \sigma_2$ means σ_1 does at least as well as σ_2 against every environment strategy realizing $\mathcal{E}$ and strictly better against one such strategy. If σ_1 strictly dominates σ_2, then σ_1 makes more effort than σ_2 to satisfy the goal. In fact, if σ_2 is strictly dominated by σ_1, then an agent that uses σ_2 does not its best to achieve the goal: if it used σ_1 instead, it could achieve the goal against a strictly larger set of environment behaviors. Then, since best-effort strategies achieve the goal against the largest possible set of environment behaviors, an agent strategy is best-effort if it is not strictly dominated by another strategy.

We define the problem of LTL$_f$ best-effort synthesis under environment assumptions as follows:

Definition 2. *The* LTL$_f$ *best-effort synthesis problem under environment assumptions is defined as a pair $\mathcal{P} = (\mathcal{E}, \varphi)$, where* LTL$_f$ *formula $\mathcal{E}$ is an environment assumption, and* LTL$_f$ *formula φ is the agent goal. Best-effort synthesis of $\mathcal{P}$ computes an agent strategy that is best-effort.*

In standard synthesis under assumptions, it is common to first check whether the problem is realizable, indicating that there exists an agent strategy that realizes the problem. However, it has been shown that in the best-effort synthesis, there always exists an agent strategy that is best-effort for the goal φ under environment assumption $\mathcal{E}$.

Theorem 1 ([Aminof *et al.*, 2021b]). *Let $\mathcal{P} = (\mathcal{E}, \varphi)$ be an* LTL$_f$ *best-effort synthesis problem, there always exists an agent strategy that is best-effort.*

Aminof *et al.* [2021b] presented that LTL$_f$ best-effort synthesis can be solved by a reduction to suitable DFA games, and it has been shown to be 2EXPTIME-complete [Aminof *et al.*, 2021b].

DFA Games. A *DFA game* is a two-player game played on a DFA, described as a pair $(\mathcal{D}, F)$, where $\mathcal{D}$ is a deterministic transition system such that $\mathcal{D} = (2^{\mathcal{X}\cup\mathcal{Y}}, S, s_0, \delta)$, where $\delta\colon S \times 2^{\mathcal{X}\cup\mathcal{Y}} \to S$ is the *transition function* and $F \subseteq S$ is the set of the final states of the game. Given a play $\pi = (X_0 \cup Y_0)(X_1 \cup Y_1) \ldots \in (2^{\mathcal{X}\cup\mathcal{Y}})^\omega$, running π on $\mathcal{D}$ gives us an infinite sequence $\rho = s_0 s_1 \ldots \in S^\omega$ such that s_0 is the initial state and $s_{i+1} = \delta(s_i, X_i \cup Y_i)$ for all $i \geq 0$. Since the transitions in $\mathcal{D}$ are all deterministic, we thus denote by $\rho = \mathsf{Run}(\pi, \mathcal{D})$ the unique sequence of running π on $\mathcal{D}$. Analogously, we denote by $\rho^k = \mathsf{Run}(\pi^k, \mathcal{D})$ the unique finite sequence of running π^k on $\mathcal{D}$, and $\rho^k = s_0 s_1 \ldots s_{k+1}$. The set of final states F indicates the states in which the agent wins the game. Then, a play $\pi \in (2^{\mathcal{X}\cup\mathcal{Y}})^\omega$ is a *winning play* for the agent if $\rho = \mathsf{Run}(\pi, \mathcal{D})$ and $\exists \ell \geq 0\colon \rho_\ell \in F$. Intuitively, DFA games require F to be visited at least once.

Given σ_{ag} and σ_{env} denoting an agent strategy and an environment strategy, respectively, we denote by $PLAY(\sigma_{ag}, \sigma_{env})$ the unique play that is consistent with both σ_{ag} and σ_{env}. An agent strategy σ_{ag} is *winning* in the game $(\mathcal{D}, F)$ if $\forall \sigma_{env}$ it results that $PLAY(\sigma_{ag}, \sigma_{env})$ is winning. An environment strategy σ_{env} is *winning* in the game $(\mathcal{D}, F)$ if $\forall \sigma_{ag}$ it results that $PLAY(\sigma_{ag}, \sigma_{env})$ is not winning for the agent. In *DFA games*, $s \in S$ is a *winning* state for the agent (resp. environment) if the agent (resp. the environment) has a winning strategy in the game $(\mathcal{D}', F)$, where $\mathcal{D}' = (2^{\mathcal{X}\cup\mathcal{Y}}, S, s, \delta)$, i.e., the same arena $\mathcal{D}$ but with the new initial state s. By $\mathsf{W}_{ag}(\mathcal{D}, F)$ (resp. $\mathsf{W}_{env}(\mathcal{D}, F)$) we denote the set of all agent (resp. environment) winning states. Intuitively, W_{ag} represents the "agent winning region", from which the agent is able to win the game, no matter how the environment behaves.

We also define cooperatively winning strategies for DFA games. An agent strategy σ_{ag} is *cooperatively winning* in game $(\mathcal{D}, F)$ if $\exists \sigma_{env}$ such that $PLAY(\sigma_{ag}, \sigma_{env})$ is a winning play. An analogous definition holds for cooperatively winning environment strategies. Hence, $s \in S$ is a *cooperatively winning state* for the agent (resp. environment) if the agent (resp. the environment) has a cooperatively winning strategy in the game $(\mathcal{D}', F)$, where $\mathcal{D}' = (2^{\mathcal{X}\cup\mathcal{Y}}, S, s_0, \delta)$. By $\mathsf{W}^{coop}_{ag}(\mathcal{D}, F)$ (resp. $\mathsf{W}^{coop}_{env}(\mathcal{D}, F)$ we denote the set of all agent (resp. environment) cooperative winning states.

If the agent makes its choices based only on the current state of the game and on the choice of the environment, then we say that the agent uses a *positional strategy*. An *agent positional strategy* (a.k.a. *agent memory-less strategy*) is a function $\tau_{ag}\colon S \times 2^{\mathcal{X}} \to 2^{\mathcal{Y}}$. Similarly, we can define an environment positional strategy as a function $\tau_{env}\colon S \to 2^{\mathcal{X}}$. An agent positional strategy τ_{ag} *induces* an agent strategy σ_{ag} as follows: $\sigma_{ag}(X_0) = \tau_{ag}(s_0, X_0)$ and for $k > 0$ $\sigma_{ag}(X_0 X_1 \ldots X_k) = \tau_{ag}(s_k, X_k)$, where s_k is the last state in the finite sequence $\rho^{k-1} = \mathsf{Run}(\pi^{k-1}, \mathcal{D})$. Similarly, an environment positional strategy induces an environment strategy. An agent (resp. environment) positional strategy is winning if the agent (resp. environment) strategy it induces is winning. A positional strategy for a player that is winning from every state in its winning region is called *uniform winning*. Similarly, a positional strategy for a player is cooperatively winning if the strategy it induces is coopereatively winning. Finally, a positional strategy for a player that is winning from every state in the cooperatively winning region is called *uniform cooperatively winning*.

The solution to LTL$_f$ best-effort synthesis presented by Aminof *et al.* [2021b] can be summarized as follows.

Algorithm 0 [Aminof *et al.*, 2021b]**.** Given LTL$_f$ best-effort synthesis problem $\mathcal{P} = (\mathcal{E}, \varphi)$, proceed as follows:

1. For every $\xi \in \{\neg\mathcal{E}, \mathcal{E} \to \varphi, \mathcal{E} \wedge \varphi\}$ compute the DFAs $\mathcal{A}_\xi = (\mathcal{D}_\xi, F_\xi)$.
2. Form the product $\mathcal{D} = \mathcal{D}_{\neg\mathcal{E}} \times \mathcal{D}_{\mathcal{E}\to\varphi} \times \mathcal{D}_{\mathcal{E}\wedge\varphi}$. Lift the final states of each component to the product i.e. if $\mathcal{A}_\xi = (D_\xi, F_\xi)$ is the DFA for ξ, then the lifted condition G_ξ consists of all states$(s_{\neg\mathcal{E}}, s_{\mathcal{E}\to\varphi}, s_{\mathcal{E}\wedge\varphi})$ s.t. $s_\xi \in F_\xi$.
3. In the DFA game $(\mathcal{D}, G_{\mathcal{E}\to\varphi})$ compute a uniform positional winning strategy f_{ag}. Let $W_{ag} \subseteq S$ be the agent's winning region.

4. In the DFA game $(\mathcal{D}, G_{\neg\mathcal{E}})$ compute the environment's winning region $V \subseteq Q$.
5. Compute the environment restriction $\mathcal{D}'$ of $\mathcal{D}$ to V.
6. In the DFA game $(\mathcal{D}', G_{\mathcal{E}\wedge\varphi})$ find a uniform cooperatively winning uniform positional strategy g_{ag}.
7. **Return** the agent strategy induced by the positional strategy $k_{ag}(s, X') = f_{ag}(s, X')$ if $s \in W$, and $k_{ag}(s, X') = g_{ag}(s, X')$ otherwise.

4 Symbolic Best-effort Synthesis

In this section, we present our symbolic approaches to LTL$_f$ best-effort synthesis, consisting of 3 different versions, namely monolithic, explicit-compositional, and symbolic-compositional. In particular, we exploit the symbolic techniques presented by Zhu *et al.* [2017] to deal with DFA games

4.1 Symbolic DFA Games

We consider the DFA game representation described in 3 is an explicit-state representation. Instead, we are able to represent a DFA game compactly in a symbolic way. More specifically, the game arena $\mathcal{D} = (2^{\mathcal{X}\cup\mathcal{Y}}, S, s_0, \delta)$ can be represented *symbolically*, by encoding the state space using a logarithmic number of propositions. The *symbolic* representation of $\mathcal{D}$ is a tuple $\mathcal{D}^s = (\mathcal{X}, \mathcal{Y}, \mathcal{Z}, Z_0, \eta)$, where $\mathcal{Z}$ is a set of variables such that $|\mathcal{Z}| = \lceil \log |S| \rceil$, and every state $s \in S$ corresponds to an interpretation $Z \in 2^{\mathcal{Z}}$ over $\mathcal{Z}$. $Z_0 \in 2^{\mathcal{Z}}$ is the interpretation corresponding to the initial state s_0. $\eta\colon 2^{\mathcal{X}} \times 2^{\mathcal{Y}} \times 2^{\mathcal{Z}} \to 2^{\mathcal{Z}}$ is a Boolean function mapping interpretations X, Y and Z of the propositions of $\mathcal{X}$, $\mathcal{Y}$ and $\mathcal{Z}$ to a new interpretation Z' of the propositions of $\mathcal{Z}$, such that if Z corresponds to a state $s \in S$ then Z' corresponds to the state $\delta(s, X \cup Y)$. The set of goal states is represented by a Boolean formula f over $\mathcal{Z}$ that is only satisfied by the interpretations of states in F. Accordingly, we denote the symbolic reachability game as $(\mathcal{D}^s, f)$. Note that the transition function η can be represented by an indexed family consisting of a Boolean formula η_z for each state variable $z \in \mathcal{Z}$, which when evaluated over an assignment to $\mathcal{X} \cup \mathcal{Y} \cup \mathcal{Z}$ returns the next assignment to z.

We also define the product of two symbolic transition systems $\mathcal{D}_1^s = (\mathcal{X}, \mathcal{Y}, \mathcal{Z}_1, Z_{01}, \eta_1)$, $\mathcal{D}_2^s = (\mathcal{X}, \mathcal{Y}, \mathcal{Z}_2, Z_{02}, \eta_2)$ as $\mathcal{D}_1^s \times \mathcal{D}_2^s = (\mathcal{X}, \mathcal{Y}, \mathcal{Z}, Z_0, \eta)$, where $\mathcal{X}, \mathcal{Y}$ are as defined for $\mathcal{D}_1^s$ and $\mathcal{D}_2^s$, $\mathcal{Z} = \mathcal{Z}_1 \cup \mathcal{Z}_2$, $Z_0 = (Z_{01}, Z_{02})$ and $\eta : 2^{\mathcal{X}} \times 2^{\mathcal{Y}} \times 2^{\mathcal{Z}_1} \times 2^{\mathcal{Z}_2} \to 2^{\mathcal{Z}_1} \times 2^{\mathcal{Z}_2}$. Since the transition functions can be represented by an indexed family, the product transition function η is, in fact, a union of the two indexed families of η_1 and η_2.

Given a symbolic DFA game $(\mathcal{D}^s, f)$, we can compute a uniform winning positional agent strategy by a least fixpoint computation over two Boolean formulas w over $\mathcal{Z}$ and t over $\mathcal{Z} \cup \mathcal{X} \cup \mathcal{Y}$, which represent the agent winning region $\mathsf{W}_{ag}(\mathcal{D}^s, f)$ and tuples of winning states, environment actions with suitable agent actions, respectively. w and t are initialized as $w_0(\mathcal{Z}) = f(\mathcal{Z})$ and $t_0(\mathcal{Z}, \mathcal{X}, \mathcal{Y}) = f(\mathcal{Z})$, since every goal state is an agent winning state. Note that t_0 is independent of the propositions from $\mathcal{X} \cup \mathcal{Y}$, since once the play reaches goal states, the agent can do whatever it wants. t_{i+1} and w_{i+1} are constructed as follows:

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

The computation reaches a fixpoint when $w_{i+1} \equiv w_i$. At this point, no more states will be added, and so all agent winning states have been collected. By evaluating Z_0 on w_{i+1} we can know if there exists a winning strategy. If that is the case, t_{i+1} can be used to compute a uniform positional winning strategy through the mechanism of Boolean synthesis [Fried *et al.*, 2016]. More specifically, passing t_i to a Boolean synthesis procedure, setting $\mathcal{Z} \cup \mathcal{X}$ as input variables and $\mathcal{Y}$ as output variables, we obtain a uniform winning positional strategy $\tau : 2^{\mathcal{Z}} \times 2^{\mathcal{X}} \to 2^{\mathcal{Y}}$ that can be used to induce an agent winning strategy.

Also computating a uniform cooperatively winning positional strategy can be performed by a least fixpoint computation. To do this, we define again Boolean functions $\hat{w}$ over $\mathcal{Z}$ and $\hat{t}$ over $\mathcal{Z} \cup \mathcal{X} \cup \mathcal{Y}$, now representing the agent cooperatively winning region and tuples of cooperatively winning states and cooperative agent and environment actions, respectively. As before, we initialize $\hat{w}_0(\mathcal{Z}) = f(\mathcal{Z})$ and $\hat{t}_0(\mathcal{Z}, \mathcal{X}, \mathcal{Y}) = f(\mathcal{Z})$, since every goal state is a cooperatively winning state. Then, we construct $\hat{t}_{i+1}$ and $\hat{w}_{i+1}$ as follows:

$$\hat{t}_{i+1}(Z, Y, Y) = \hat{t}_i(Z, X, Y) \vee (\neg \hat{w}_i(Z) \wedge \hat{w}_i(\eta(X, Y, Z)))$$
$$\hat{w}_{i+1}(Z) = \exists X. \exists Y. \hat{t}_{i+1}(Z, X, Y);$$

Once the computation reaches a fixpoint, checking the existence and computing a uniform cooperatively winning positional strategy can be done as shown above.

Intuitively, when computing a uniform winning positional strategy we look for agent actions for which, no matter how the environment behaves, the agent is able to reach the winning region $\mathsf{W}_{ag}(\mathcal{D}^s, f)$. Differently, when computing a uniform cooperatively winning positional strategy we're looking for agent actions for which, if the environment acts cooperatively, the agent is able to reach the cooperatively winning region $\mathsf{W}_{ag}^{coop}(\mathcal{D}, f)$.

4.2 Monolithic Approach

The monolithic approach is a direct implementation of the best-effort synthesis by Aminof *et al.* [2021b] (i.e., of Algorithm 0), utilizing the symbolic synthesis framework introduced by Zhu *et al.* [2017]. Given a best-effort synthesis problem $\mathcal{P} = (\mathcal{E}, \varphi)$, we first construct the DFAs following the synthesis algorithm described in Section 3, and convert them into a symbolic representation. Finally, we solve suitable games on the symbolic DFAs and obtain a best-effort strategy. More specifically, this approach solves the best-effort synthesis problem $\mathcal{P} = (\mathcal{E}, \varphi)$ as follows.

Algorithm 1.

1. For LTL$_f$ formulas $\mathcal{E} \to \varphi$, $\neg\mathcal{E}$ and $\mathcal{E} \wedge \varphi$ compute the corresponding minimal explicit-state DFAs $\mathcal{A}_{\mathcal{E}\to\varphi} = (\mathcal{D}_{\mathcal{E}\to\varphi}, F_{\mathcal{E}\to\varphi})$, $\mathcal{A}_{\neg\mathcal{E}} = (\mathcal{D}_{\neg\mathcal{E}}, F_{\neg\mathcal{E}})$ and $\mathcal{A}_{\mathcal{E}\wedge\varphi} = (\mathcal{D}_{\mathcal{E}\wedge\varphi}, F_{\mathcal{E}\wedge\varphi})$, and convert the DFAs to a symbolic representation and obtain $\mathcal{A}^s_{\mathcal{E}\to\varphi} = (\mathcal{D}^s_{\mathcal{E}\to\varphi}, f_{\mathcal{E}\to\varphi})$, $\mathcal{A}^s_{\neg\mathcal{E}} = (\mathcal{D}^s_{\neg\mathcal{E}}, f_{\neg\mathcal{E}})$ and $\mathcal{A}^s_{\mathcal{E}\wedge\varphi} = (\mathcal{D}^s_{\mathcal{E}\wedge\varphi}, f_{\mathcal{E}\wedge\varphi})$.

2. Construct symbolic DFA games $(\mathcal{D}^s, f_{\neg\mathcal{E}})$ and $(\mathcal{D}^s, f_{\mathcal{E}\to\varphi})$, where $\mathcal{D}^s = \mathcal{D}^s_{\neg\mathcal{E}} \times \mathcal{D}^s_{\mathcal{E}\to\varphi} \times \mathcal{D}^s_{\mathcal{E}\wedge\varphi}$ is the product of the symbolic transition systems $\mathcal{D}^s_{\neg\mathcal{E}}$, $\mathcal{D}^s_{\mathcal{E}\to\varphi}$, and $\mathcal{D}^s_{\mathcal{E}\wedge\varphi}$.
3. In the DFA game $(\mathcal{D}^s, f_{\mathcal{E}\to\varphi})$ compute a uniform positional strategy τ_{ag}, and the agent winning region, which is represented as Boolean formula $\mathsf{W}_{ag}(\mathcal{D}^s, f_{\mathcal{E}\to\varphi})$ over state variables $\mathcal{Z}$.
4. In the DFA game $(\mathcal{D}^s, f_{\neg\mathcal{E}})$ compute the environment's winning region, represented as Boolean formula $\mathsf{W}_{env}(\mathcal{D}^s, f_{\neg\mathcal{E}})$ over $\mathcal{Z}$.
5. Construct symbolic DFA game $(\mathcal{D}'^s, f_{\mathcal{E}\wedge\varphi})$, where $\mathcal{D}'^s = \mathcal{D}^s \wedge \mathsf{W}_{env}(\mathcal{D}^s, f_{\neg\mathcal{E}})$ such that restricting the state space of transition system $\mathcal{D}^s$ to considering $\mathsf{W}_{env}(\mathcal{D}^s, f_{\neg\mathcal{E}})$ only.
6. In the DFA game $(\mathcal{D}'^s, f_{\mathcal{E}\wedge\varphi})$ find a positional cooperatively winning strategy γ_{ag}.
7. **Return** the best-effort strategy σ_{ag} induced by the positional strategy constructed as follows: $\kappa_{ag}(Z, X) = \tau_{ag}(Z, X)$ if $Z \models \mathsf{W}_{ag}(\mathcal{D}^s, f_{\mathcal{E}\to\varphi})$ and $\kappa_{ag}(Z, X) = \gamma_{ag}(Z, X)$ otherwise.

The operations workflow of the monolithic approach is shown in Figure 1. First, we to transform the formulas $\mathcal{E} \to \varphi$, $\neg\mathcal{E}$ and $\mathcal{E} \wedge \varphi$ into minimal explicit-state DFAs $\mathcal{A}_{\mathcal{E}\to\varphi}$, $\mathcal{A}_{\neg\mathcal{E}}$ and $\mathcal{A}_{\mathcal{E}\wedge\varphi}$. Then, we transform these explicit-state DFA into symbolic representations $\mathcal{A}^s_{\mathcal{E}\to\varphi}$, $\mathcal{A}^s_{\neg\mathcal{E}}$ and $\mathcal{A}^s_{\mathcal{E}\wedge\varphi}$. At this point, the arena of the games played by the algorithm is constructed through product as $\mathcal{D}^s_{\mathcal{E}\to\varphi} \times \mathcal{D}^s_{\neg\mathcal{E}} \times \mathcal{D}^s_{\mathcal{E}\wedge\varphi}$. Then, the games of the algorithms are constructed as: $(\mathcal{D}^s, f_{\mathcal{E}\to\varphi})$, $(\mathcal{D}^s, f_{\neg\mathcal{E}})$ and $(\mathcal{D}^s, f_{\mathcal{E}\wedge\varphi})$. These DFA games are left unminimized. Here, games $(\mathcal{D}^s, f_{\mathcal{E}\to\varphi})$, $(\mathcal{D}^s, f_{\mathcal{E}\wedge\varphi})$ and $(\mathcal{D}^s, f_{\neg\mathcal{E}})$ are solved as shown in Section 4.1.

After computing the winning positional strategy τ_{ag} and the cooperatively winning positional strategy γ_{ag}, we construct the positional best-effort strategy $\kappa_{ag}(Z, X)$ as follows: $\kappa_{ag}(Z, X) = \tau_{ag}(Z, X)$ if $Z \models \mathsf{W}_{ag}(\mathcal{D}^s, f_{\mathcal{E}\to\varphi})$ and $\kappa_{ag}(Z, X) = \gamma_{ag}(Z, X)$ otherwise, i.e., the best-effort strategy is constructed so as to prefer the output of the winning positional strategy to that of the cooperatively winning positional strategy. Only in case the winning positional strategy is not defined in a state Z, then the output of the cooperatively winning positional strategy is used. Note that the positional best-effort strategy induces an LTL$_f$ best-effort strategy.

The main challenge in the monolithic approach comes from the LTL$_f$-to-DFA conversion, which can take, in the worst case, doubly exponential time [De Giacomo and Vardi, 2013], and thus also considered as the bottleneck of LTL$_f$ synthesis [Zhu *et al.*, 2017]. To that end, we propose an explicit compositional approach to diminish this difficulty by decreasing the number of performing LTL$_f$-to-DFA conversions.

4.3 Explicit-Compositional Approach

As described in Section 4.2, the monolithic approach to a best-effort synthesis problem $\mathcal{P} = (\mathcal{E}, \varphi)$ involves three rounds of LTL$_f$-to-DFA conversions with respect to LTL$_f$ formulas $\mathcal{E} \to \varphi$, $\neg\mathcal{E}$ and $\mathcal{E} \wedge \varphi$. We observe that the constructed DFAs $\mathcal{A}_{\mathcal{E}\to\varphi}$, $\mathcal{A}_{\neg\mathcal{E}}$ and $\mathcal{A}_{\mathcal{E}\wedge\varphi}$ can, in fact, be constructed by

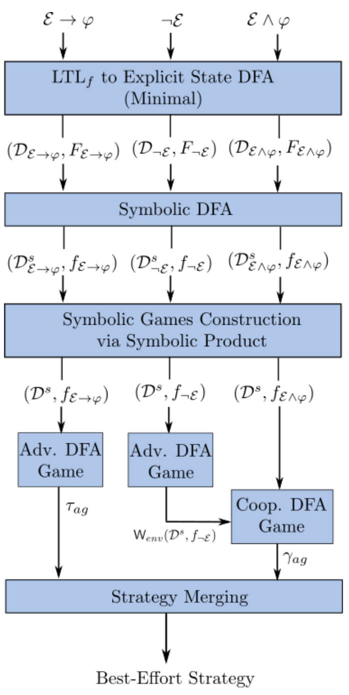
$\mathcal{E} \to \varphi$
$\neg\mathcal{E}$
$\mathcal{E} \wedge \varphi$
LTL$_f$ to Explicit State DFA (Minimal)
$(\mathcal{D}_{\mathcal{E}\to\varphi}, F_{\mathcal{E}\to\varphi})$
$(\mathcal{D}_{\neg\mathcal{E}}, F_{\neg\mathcal{E}})$
$(\mathcal{D}_{\mathcal{E}\wedge\varphi}, F_{\mathcal{E}\wedge\varphi})$
Symbolic DFA
$(\mathcal{D}^s_{\mathcal{E}\to\varphi}, f_{\mathcal{E}\to\varphi})$
$(\mathcal{D}^s_{\neg\mathcal{E}}, f_{\neg\mathcal{E}})$
$(\mathcal{D}^s_{\mathcal{E}\wedge\varphi}, f_{\mathcal{E}\wedge\varphi})$
Symbolic Games Construction via Symbolic Product
$(\mathcal{D}^s, f_{\mathcal{E}\to\varphi})$
$(\mathcal{D}^s, f_{\neg\mathcal{E}})$
$(\mathcal{D}^s, f_{\mathcal{E}\wedge\varphi})$
Adv. DFA Game
Adv. DFA Game
Coop. DFA Game
τ_{ag}
$\mathsf{W}_{env}(\mathcal{D}^s, f_{\neg\mathcal{E}})$
γ_{ag}
Strategy Merging
Best-Effort Strategy

Figure 1: Algorithm 1. Here $\mathcal{D}^s = \mathcal{D}^s_{\mathcal{E}\to\varphi} \times \mathcal{D}^s_{\neg\mathcal{E}} \times \mathcal{D}^s_{\mathcal{E}\wedge\varphi}$.

manipulating the two DFAs $\mathcal{A}_\varphi$ and $\mathcal{A}_\mathcal{E}$ of LTL$_f$ formulas $\mathcal{E}$ and φ, respectively. Specifically, given explicit-state DFAs $\mathcal{A}_\varphi$ and $\mathcal{A}_\mathcal{E}$, we obtain $\mathcal{A}_{\mathcal{E}\to\varphi}$, $\mathcal{A}_{\neg\mathcal{E}}$ and $\mathcal{A}_{\mathcal{E}\wedge\varphi}$ as follows:

- $\mathcal{A}_{\mathcal{E}\to\varphi} = \mathsf{Comp}(\mathsf{Inter}(\mathcal{A}_\mathcal{E}, \mathsf{Comp}(\mathcal{A}_\varphi))$;
- $\mathcal{A}_{\neg\mathcal{E}} = \mathsf{Comp}(\mathcal{A}_\mathcal{E})$;
- $\mathcal{A}_{\mathcal{E}\wedge\varphi} = \mathsf{Inter}(\mathcal{A}_\mathcal{E}, \mathcal{A}_\varphi)$;

where Comp and Inter denote complement and intersection on explicit-state DFA, respectively. Note that, the LTL$_f$-to-DFA is 2EXPTIME-complete, while DFA complement and intersection only takes linear time in the size of the DFA.

Thus, **Algorithm 2**, which follows the explicit-compositional approach to a best-effort synthesis problem $\mathcal{P} = (\mathcal{E}, \varphi)$, proceeds as shown in Figure 2. Here, we first translate directly the formulas $\mathcal{E}$ and φ into minimal explicit-state DFAs $\mathcal{A}_\mathcal{E}$ and $\mathcal{A}_\varphi$. Then, the DFA complement and intersection are performed to build DFAs $\mathcal{A}_{\mathcal{E}\to\varphi}$, $\mathcal{A}_{\neg\mathcal{E}}$ and $\mathcal{A}_{\mathcal{E}\wedge\varphi}$. Indeed, the constructed explicit-state DFAs can also be minimized. The remaining steps are the same as in the monolithic approach.

4.4 Symbolic-Compositional Approach

A few aspects, however, make the explicit-compositional approach unsatisfactory to us. Let us start with focusing on the two DFAs $\mathcal{A}_{\mathcal{E}\to\varphi}$ and $\mathcal{A}_{\mathcal{E}\wedge\varphi}$. Suppose we put automata minimization aside, the initial states and transition functions of both DFAs are essentially the same, and they only differ in accepting conditions. This is because the initial states and transition functions of $\mathcal{A}_{\mathcal{E}\to\varphi}$ and $\mathcal{A}_{\mathcal{E}\wedge\varphi}$ are obtained by performing cross product of $\mathcal{A}_\mathcal{E}$ and $\mathcal{A}_\varphi$. Consequently, if no minimization takes place, symbolic DFAs $\mathcal{A}^s_{\mathcal{E}\to\varphi} = (\mathcal{D}^s_{\mathcal{E}\to\varphi}, f_{\mathcal{E}\to\varphi})$ and $\mathcal{A}^s_{\mathcal{E}\wedge\varphi} = (\mathcal{D}^s_{\mathcal{E}\wedge\varphi}, f_{\mathcal{E}\wedge\varphi})$ share the same transition system such that $\mathcal{D}^s_{\mathcal{E}\to\varphi} = \mathcal{D}^s_{\mathcal{E}\wedge\varphi}$. Moreover, since

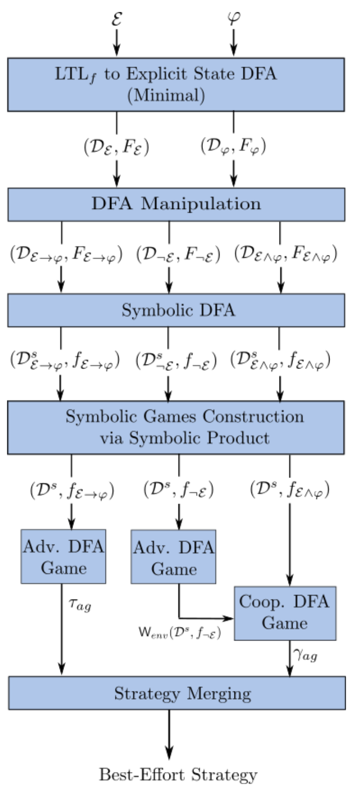
$\mathcal{E}$
φ
LTL$_f$ to Explicit State DFA (Minimal)
$(\mathcal{D}_\mathcal{E}, F_\mathcal{E})$
$(\mathcal{D}_\varphi, F_\varphi)$
DFA Manipulation
$(\mathcal{D}_{\mathcal{E}\to\varphi}, F_{\mathcal{E}\to\varphi})$
$(\mathcal{D}_{\neg\mathcal{E}}, F_{\neg\mathcal{E}})$
$(\mathcal{D}_{\mathcal{E}\wedge\varphi}, F_{\mathcal{E}\wedge\varphi})$
Symbolic DFA
$(\mathcal{D}^s_{\mathcal{E}\to\varphi}, f_{\mathcal{E}\to\varphi})$
$(\mathcal{D}^s_{\neg\mathcal{E}}, f_{\neg\mathcal{E}})$
$(\mathcal{D}^s_{\mathcal{E}\wedge\varphi}, f_{\mathcal{E}\wedge\varphi})$
Symbolic Games Construction via Symbolic Product
$(\mathcal{D}^s, f_{\mathcal{E}\to\varphi})$
$(\mathcal{D}^s, f_{\neg\mathcal{E}})$
$(\mathcal{D}^s, f_{\mathcal{E}\wedge\varphi})$
Adv. DFA Game
Adv. DFA Game
τ_{ag}
Coop. DFA Game
$\mathsf{W}_{env}(\mathcal{D}^s, f_{\neg\mathcal{E}})$
γ_{ag}
Strategy Merging
Best-Effort Strategy

Figure 2: Algorithm 2. Here $\mathcal{D}^s = \mathcal{D}^s_{\mathcal{E}\to\varphi} \times \mathcal{D}^s_{\neg\mathcal{E}} \times \mathcal{D}^s_{\mathcal{E}\wedge\varphi}$.

we represent the symbolic transition function as an indexed family, the transition function of $\mathcal{D}^s_{\neg\mathcal{E}}$ in $\mathcal{A}^s_{\neg\mathcal{E}} = (\mathcal{D}^s_{\neg\mathcal{E}}, f_{\neg\mathcal{E}})$ is in fact a subset of the transition function of $\mathcal{D}^s_{\mathcal{E}\to\varphi}$ (indeed also $\mathcal{D}^s_{\mathcal{E}\wedge\varphi}$). As a result, we have the following lemma.

Lemma 1. *Let $\mathcal{E}$ and φ be two* LTL$_f$ *formulas over $\mathcal{X} \cup \mathcal{Y}$. We have that $\mathcal{D}^s_{\mathcal{E}\to\varphi} \times \mathcal{D}^s_{\neg\mathcal{E}} \times \mathcal{D}^s_{\mathcal{E}\wedge\varphi} = \mathcal{D}^s_\mathcal{E} \times \mathcal{D}^s_\varphi$.*

Proof. First, we show that $\mathcal{D}^s_{\mathcal{E}\to\varphi} = \mathcal{D}^s_{\mathcal{E}\wedge\varphi}$, where $\mathcal{D}^s_{\mathcal{E}\to\varphi} = (\mathcal{X}, \mathcal{Y}, \mathcal{Z}_{\mathcal{E}\to\varphi}, Z_{(0,\mathcal{E}\to\varphi)}, \eta_{\mathcal{E}\to\varphi})$ and $\mathcal{D}^s_{\mathcal{E}\wedge\varphi} = (\mathcal{X}, \mathcal{Y}, \mathcal{Z}_{\mathcal{E}\wedge\varphi}, Z_{(0,\mathcal{E}\wedge\varphi)}, \eta_{\mathcal{E}\wedge\varphi})$. To see this, note that $\mathcal{D}^s_{\mathcal{E}\to\varphi} = \mathcal{D}^s_{\neg(\mathcal{E}\wedge\neg\varphi)}$. Now, since for every LTL$_f$ formula ψ, it holds that $\mathcal{D}^s_\psi = \mathcal{D}^s_{\neg\psi}$, we have that $\mathcal{D}^s_{\neg(\mathcal{E}\wedge\neg\varphi)} = \mathcal{D}^s_{\mathcal{E}\wedge\neg\varphi}$ holds. The transition system $\mathcal{D}^s_{\mathcal{E}\wedge\neg\varphi}$ can indeed be obtained by performing symbolic product of two transition systems $\mathcal{D}^s_\mathcal{E}$ and $\mathcal{D}^s_{\neg\varphi}$, such that $\mathcal{D}^s_{\mathcal{E}\wedge\neg\varphi} = \mathcal{D}^s_\mathcal{E} \times \mathcal{D}^s_{\neg\varphi}$. As a result, it holds that $\mathcal{D}^s_{\mathcal{E}\to\varphi} = \mathcal{D}^s_{\neg(\mathcal{E}\wedge\neg\varphi)} = \mathcal{D}^s_{\mathcal{E}\wedge\neg\varphi} = \mathcal{D}^s_\mathcal{E} \times \mathcal{D}^s_{\neg\varphi} = \mathcal{D}^s_\mathcal{E} \times \mathcal{D}^s_\varphi = \mathcal{D}^s_{\mathcal{E}\wedge\varphi}$. Thus, we can simplify $\mathcal{D}^s_{\mathcal{E}\to\varphi} \times \mathcal{D}^s_{\neg\mathcal{E}} \times \mathcal{D}^s_{\mathcal{E}\wedge\varphi}$ as $\mathcal{D}^s_{\neg\mathcal{E}} \times \mathcal{D}^s_{\mathcal{E}\wedge\varphi}$.

Furthermore, since $\mathcal{D}^s_{\neg\mathcal{E}} = \mathcal{D}^s_\mathcal{E} = (\mathcal{X}, \mathcal{Y}, \mathcal{Z}_\mathcal{E}, Z_{0,\mathcal{E}}, \eta_\mathcal{E})$ and $\mathcal{D}^s_{\mathcal{E}\wedge\varphi} = \mathcal{D}^s_\mathcal{E} \times \mathcal{D}^s_\varphi = (\mathcal{X}, \mathcal{Y}, (\mathcal{Z}_\mathcal{E} \cup \mathcal{Z}_\varphi), (Z_{0,\mathcal{E}} \wedge Z_{0,\varphi}), (\eta_\mathcal{E} \cup \eta_\varphi))$, the state variables $\mathcal{Z}_\mathcal{E}$ and the transition function $\eta_\mathcal{E}$, represented as an indexed family, of $\mathcal{D}^s_{\neg\mathcal{E}}$ are actually, the subsets of the state variables $(\mathcal{Z}_\mathcal{E} \cup \mathcal{Z}_\varphi)$ and the transition function $(\eta_\mathcal{E} \cup \eta_\varphi)$ of $\mathcal{D}^s_\mathcal{E} \times \mathcal{D}^s_\varphi$, respectively. Hence, $\mathcal{D}^s_{\neg\mathcal{E}} \times \mathcal{D}^s_{\mathcal{E}\wedge\varphi}$ can be further simplified as $\mathcal{D}^s_{\mathcal{E}\wedge\varphi}$, which is equivalent to $\mathcal{D}^s_\mathcal{E} \times \mathcal{D}^s_\varphi$. We thus conclude that $\mathcal{D}^s_{\mathcal{E}\to\varphi} \times \mathcal{D}^s_{\neg\mathcal{E}} \times \mathcal{D}^s_{\mathcal{E}\wedge\varphi} = \mathcal{D}^s_\mathcal{E} \times \mathcal{D}^s_\varphi$. $\square$

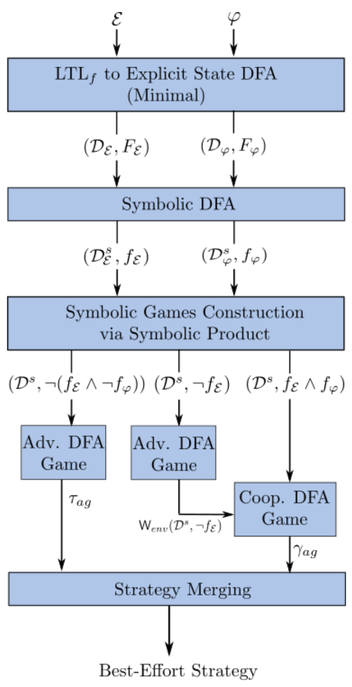
$\mathcal{E}$
φ
LTL$_f$ to Explicit State DFA (Minimal)
$(\mathcal{D}_\mathcal{E}, F_\mathcal{E})$
$(\mathcal{D}_\varphi, F_\varphi)$
Symbolic DFA
$(\mathcal{D}^s_\mathcal{E}, f_\mathcal{E})$
$(\mathcal{D}^s_\varphi, f_\varphi)$
Symbolic Games Construction via Symbolic Product
$(\mathcal{D}^s, \neg(f_\mathcal{E} \wedge \neg f_\varphi))$
$(\mathcal{D}^s, \neg f_\mathcal{E})$
$(\mathcal{D}^s, f_\mathcal{E} \wedge f_\varphi)$
Adv. DFA Game
Adv. DFA Game
τ_{ag}
Coop. DFA Game
$\mathsf{W}_{env}(\mathcal{D}^s, \neg f_\mathcal{E})$
γ_{ag}
Strategy Merging
Best-Effort Strategy

Figure 3: Algorithm 3. Here $\mathcal{D}^s = \mathcal{D}^s_\mathcal{E} \times \mathcal{D}^s_\varphi$.

Furthermore, since no DFA minimization takes place then the DFAs have the following form: $\mathcal{A}^s_{\mathcal{E}\to\varphi} = (\mathcal{D}^s, f_{\mathcal{E}\to\varphi})$, $\mathcal{A}^s_{\neg\mathcal{E}} = (\mathcal{D}^s, f_{\neg\mathcal{E}})$ and $\mathcal{A}^s_{\mathcal{E}\wedge\varphi} = (\mathcal{D}^s, f_{\mathcal{E}\wedge\varphi})$. This means that we can construct $\mathcal{D}^s = \mathcal{D}^s_\mathcal{E} \times \mathcal{D}^s_\varphi$ once and for all and then get the three automata by defining the final states functions (which are Boolean functions) from $f_\mathcal{E}$ and f_φ as follows:

- $f_{\mathcal{E}\to\varphi} = \neg(f_\mathcal{E} \wedge \neg(f_\varphi))$;
- $f_{\neg\mathcal{E}} = \neg f_\mathcal{E}$;
- $f_{\mathcal{E}\wedge\varphi} = f_\mathcal{E} \wedge f_\varphi$;

Thus, given a best-effort synthesis problem $\mathcal{P} = (\mathcal{E}, \varphi)$, the symbolic-compositional approach solves it as follows.

Algorithm 3.

1. Compute the minimal explicit-state DFAs $\mathcal{A}_\mathcal{E} = (\mathcal{D}_\mathcal{E}, F_\mathcal{E})$ and $\mathcal{A}_\varphi = (\mathcal{D}_\varphi, F_\varphi)$ and convert the DFAs to a symbolic representation and obtain $\mathcal{A}^s_\mathcal{E} = (\mathcal{D}^s_\mathcal{E}, f_\mathcal{E})$ and $\mathcal{A}^s_\varphi = (\mathcal{D}^s_\varphi, f_\varphi)$.
2. Construct the symbolic DFA games $\mathcal{G}^s_{\neg\mathcal{E}} = (\mathcal{D}^s, \neg f_\mathcal{E})$ and $\mathcal{G}^s_{\mathcal{E}\to\varphi} = (\mathcal{D}^s, \neg(f_\mathcal{E} \wedge \neg(f_\varphi)))$.
3. In the DFA game $\mathcal{G}^s_{\mathcal{E}\to\varphi} = (\mathcal{D}^s, \neg(f_\mathcal{E} \wedge \neg(f_\varphi)))$ compute a uniform positional winning strategy τ_{ag}, and the agent winning region, represented as Boolean formula $\mathsf{W}_{ag}(\mathcal{D}^s, \neg(f_\mathcal{E} \wedge \neg(f_\varphi)))$ over state variables $\mathcal{Z}$.
4. In the DFA game $(\mathcal{D}^s, \neg f_\mathcal{E})$ compute the environment's winning region, represented as Boolean formula $\mathsf{W}_{env}(\mathcal{D}^s, \neg f_\mathcal{E})$ over $\mathcal{Z}$.
5. Construct symbolic DFA game $(\mathcal{D}'^s, f_\mathcal{E} \wedge f_\varphi)$, where $\mathcal{D}'^s = \mathcal{D}^s \wedge \mathsf{W}_{env}(\mathcal{D}^s, \neg f_\mathcal{E})$ such that restricting the state space of transition system $\mathcal{D}^s$ to considering $\mathsf{W}_{env}(\mathcal{D}^s, \neg f_\mathcal{E})$ only.

6. In the DFA game $(\mathcal{D}'^s, f_{\mathcal{E}} \wedge f_{\varphi})$ find a positional cooperatively winning strategy γ_{ag}.
7. **Return** the best-effort strategy σ_{ag} induced by the positional strategy constructed as follows: $\kappa_{ag}(Z, X) = \tau_{ag}(Z, X)$ if $Z \models \mathsf{W}_{ag}(\mathcal{D}^s, \neg(f_{\mathcal{E}} \wedge \neg(f_{\varphi})))$ and $\kappa_{ag}(Z, X) = \gamma_{ag}(Z, X)$ otherwise.

As shown in Figure 3, the symbolic-compositional approach first transforms formulas $\mathcal{E}$ and φ into minimal explicit-state DFAs $\mathcal{A}_{\mathcal{E}}$ and $\mathcal{A}_{\varphi}$, and then directly into a symbolic representation $\mathcal{A}^s_{\mathcal{E}}$ and $\mathcal{A}^s_{\varphi}$. Hence, we construct the arena of the games as $\mathcal{D}^s_{\mathcal{E}} \times \mathcal{D}^s_{\varphi}$. Note that the result of this product is unminimized. At this point the games played by the algorithm are constructed as $(\mathcal{D}^s, \neg(f_{\mathcal{E}} \wedge \neg(f_{\varphi})))$, $(\mathcal{D}^s, \neg f_{\mathcal{E}})$ and $(\mathcal{D}^s, f_{\mathcal{E}} \wedge f_{\varphi})$. From now on, the remaining steps are the same as in the monolithic approach, indeed also the explicit-compositional approach.

5 Empirical Evaluations

In this section, we first describe how we implemented our symbolic LTL_f best-effort synthesis approaches described in Section 4. Then, by empirical evaluation, we show that Algorithm 3, i.e., the symbolic-compositional approach, shows an overall best-performance. In particular, we show that performing best-effort synthesis only brings a minimal overhead with respect to standard synthesis.

5.1 Implementation

We implemented the three different symbolic best-effort synthesis approaches described in Section 4 in a tool called *BeSyft*, by extending the symbolic synthesis framework [Zhu *et al.*, 2017; Tabajara and Vardi, 2019] integrated in state-of-the-art synthesis tools [Bansal *et al.*, 2020; De Giacomo and Favorito, 2021a]. In particular, we based on LYDIA [1], the overall best performing LTL_f-to-DFA conversion tool, to construct the explicit-state DFA from LTL_f formulas. It should be noted that all the explicit-state DFAs constructed by LYDIA are minimized. Moreover, *BeSyft* borrows the rich APIs from LYDIA to perform the relevant explicit-state DFA manipulations required by both of Algorithm 1, i.e., the monolithic approach (c.f., Subsection 4.2) and Algorithm 2, i.e., the explicit-compositional approach (c.f., Subsection 4.3), such as complement, intersection, minimization. As in [Zhu *et al.*, 2017; Tabajara and Vardi, 2019], the symbolic DFA games are represented in Binary Decision Diagrams (BDDs) [Bryant, 1992], utilizing CUDD-3.0.0 [Somenzi, 2016] as the BDD library. Thereby, *BeSyft* constructs and solves symbolic DFA games by making use of the Boolean operations provided by CUDD-3.0.0, such as negation, conjunction, qualification. The positional winning strategy τ_{ag} and the positional cooperatively winning strategy γ_{ag} are abstracted utilizing Boolean synthesis [Fried *et al.*, 2016]. The positional best-effort strategy is obtained by applying suitable Boolean operations on τ_{ag} and γ_{ag}. As a result, we have three derivations of *BeSyft*, namely *BeSyft*-Alg-1, *BeSyft*-Alg-2 and *BeSyft*-Alg-3, corresponding to the monolithic approach, the explicit-compositional approach, and the symbolic-compositional approach, respectively.

[1]https://github.com/whitemech/lydia

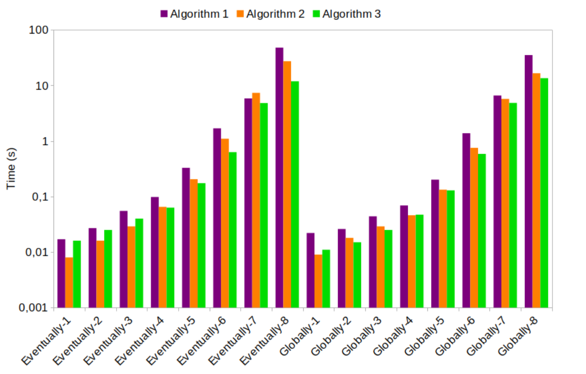

Figure 4: Comparison of the 3 best-effort synthesis algs in log scale.

5.2 Experiment Methodology

Experiment Setup. All experiments were run on a laptop with operating system 64-bit Ubuntu 20.04, 3.6 GHz CPU, and 12 GB of memory. Time out was set to 1000 seconds.

Benchmark. We devised a *counter game* benchmark, based on the one proposed by Zhu *et al.* [2020]. More specifically, there is an n-bit binary counter. At each round, the environment chooses whether to issue an increment request for the counter or not. The agent can choose to grant the request or ignore it. The goal is to get the counter having all bits set to 1. The increment requests only come from the environment, and occur in accordance with the environment assumption. Note that the size of the corresponding minimal DFA grows exponentially as the size of the counter game n grows.

In the experiments, we considered two types of environment assumptions: $\mathcal{E}_{\Box} = \Box(add)$ and $\mathcal{E}_{\Diamond} = \Diamond(add)$. We name the counter game instances with the assumption $\mathcal{E}_{\Box}$ as $\Box$-instances, and the ones with the assumption $\mathcal{E}_{\Diamond}$ as $\Diamond$-instances. It is easy to see that $\Box$-instances are always agent realizable, i.e., the agent has a strategy to realize the goal under environment assumption $\mathcal{E}_{\Box}$. Indeed, such a strategy is to grant all increment request coming from the environment. Instead, $\Diamond$-instances are agent realizable for 1-bit counter games, and cooperatively realizable otherwise. Indeed, in the first case, a winning strategy for the agent is to grant the increment request that will eventually occur. Differently, in the second case, the agent can achieve the goal only if the environment behaves cooperatively such that issuing more increment requests than that specified by the environment assumption. That is to say, the agent needs a best-effort strategy.

5.3 Experimental Results and Analysis.

Figure 4 (note that the y-axis is in log scale) shows the running time of each derivation of *BeSyft* on every instance of the counter games. Figure 5 shows the same result as Figure 4, but having the y-axis in linear scale, thus showing a clearer view of the performance of three derivations of *BeSyft*. Across these instances, we see that all the derivations were able to manage instances with up to 8 bits, for both $\Box$- and $\Diamond$- counter instances. In particular, *BeSyft*-Alg-1 is beaten by the other two derivations, since it requires three rounds of LTL_f-to-DFA conversions, which in the worst case, can lead

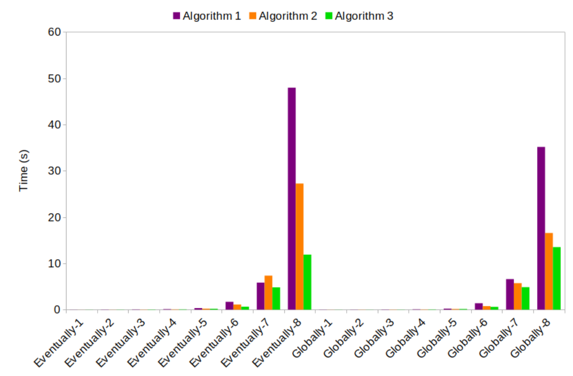

Figure 5: Same comparison of Figure 4 in linear scale.

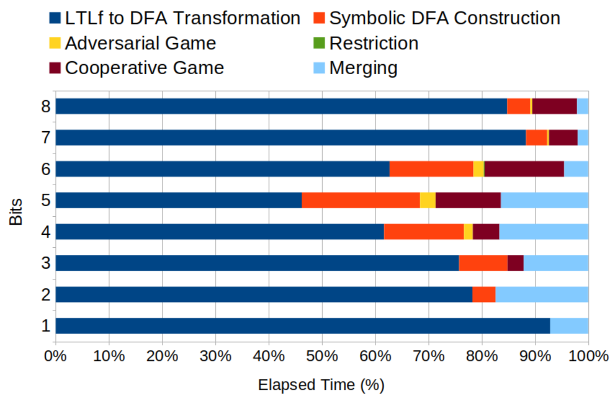

Figure 6: Relative time costs of each step in Algorithm 3.

to a double-exponential blowup. Note that the time cost of all derivations of *BeSyft* grows exponentially, instead of double-exponential. This is because LYDIA always returns the minimal explicit-state DFA of a given LTL$_f$ formula, and the powerful automata minimization can possibly disappear the exponential gap, as observed in [Tabajara and Vardi, 2019; Zhu *et al.*, 2020]. This also enables the state-of-the-art LTL$_f$ synthesis tools to take the maximal advantage of automata minimization [Bansal *et al.*, 2020; De Giacomo and Favorito, 2021a]. Nevertheless, it is not the case that automata minimization always leads to an improvement. Instead, there is a tread-off of performing automata minimization. As shown in Figures 4 and 5, *BeSyft*-Alg-3 shows better performance comparing to *BeSyft*-Alg-2, though the former cannot minimize the game arena after symbolic product, and the latter minimizes the game arena as much as possible.

On a closer inspection, we evaluated the time cost of each major operation of *BeSyft*-Alg-3, and present the results on $\Diamond$-counter instances in Figure 6. First, the results show that LTL$_f$-to-DFA conversion is the bottleneck of LTL$_f$ best-effort synthesis, the cost of which dominates the total running time. Moreover, we can see that the total time cost of cooperative DFA game and strategy merging of best-effort synthesis, only counts for $20\% \sim 30\%$ of the total time cost. As a result, we conclude that, in general, performing best-effort synthesis only brings a minimal overhead.

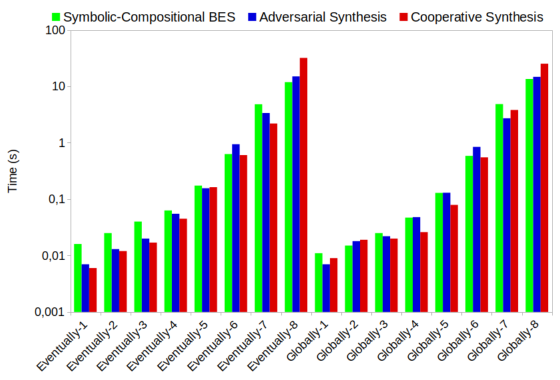

Figure 7: Comparison of *BeSyft*-Alg-3 and state-of-the-art algorithms for LTL$_f$ adversarial and cooperative synthesis.

Furthermore, we also compared the time cost of *BeSyft*-Alg-3 with that of standard synthesis on counter games. More specifically, we consider two kinds of reactive synthesizer: adversarial synthesizer that computes an agent winning strategy for LTL$_f$ formula $\mathcal{E} \rightarrow \varphi$; cooperative synthesizer that computes an agent cooperatively winning strategy for LTL$_f$ formula $\mathcal{E} \wedge \varphi$. Figure 7 shows that for small instances (up to 7 bits), *BeSyft*-Alg-3 only takes minor extra time than those of adversarial synthesizer and cooperative synthesizer. Interestingly, for the 8-bit counter instances, *BeSyft*-Alg-3 even takes less time. Although *BeSyft*-Alg-3 performs LTL$_f$-to-DFA conversion of LTL$_f$ formulas φ and $\mathcal{E}$ separately and combines them to obtain the final game arena without having automata minimization, it can happen that the total time cost of such is even less than the LTL$_f$-to-DFA conversion of a larger formula $\mathcal{E} \rightarrow \varphi$. This, again, confirms that performing best-effort synthesis only brings a minimal overhead with respect to standard synthesis.

6 Conclusion

We presented three different symbolic LTL$_f$ best-effort synthesis approaches: monolithic, explicit-compositional and symbolic-compositional. Empirical evaluations proved the outperformance of the symbolic-compositional approach. An interesting observation is that, although previous studies suggest taking the maximal advantage of automata minimization [Tabajara and Vardi, 2019; Zhu *et al.*, 2020], in the case of LTL$_f$ best-effort synthesis, there can be a trade-off of doing so. Another significant finding is that the best-performing LTL$_f$ best-effort synthesis approach only brings a minimal overhead comparing to standard synthesis. Given this nice computational result, a natural future direction would be looking into LTL best-effort synthesis [Aminof *et al.*, 2021b] and LTL$_f$ best-effort synthesis with multiple environment assumptions [Aminof *et al.*, 2021a].

Acknowledgments

This work is partially supported by the ERC Advanced Grant WhiteMech (No. 834228), the EU ICT-48 2020 project TAILOR (No. 952215), the PRIN project RIPER (No.

20203FFYLK), the JPMorgan AI Faculty Research Award "Resilience-based Generalized Planning and Strategic Reasoning".

References

[Aminof *et al.*, 2018] Benjamin Aminof, Giuseppe De Giacomo, Aniello Murano, and Sasha Rubin. Planning and synthesis under assumptions. *arXiv*, 2018.

[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.*, 2021a] Benjamin Aminof, Giuseppe De Giacomo, Alessio Lomuscio, Aniello Murano, and Sasha Rubin. Synthesizing best-effort strategies under multiple environment specifications. In *KR*, pages 42–51, 2021.

[Aminof *et al.*, 2021b] 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.

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

[Bansal *et al.*, 2020] Suguman Bansal, Yong Li, Lucas M. Tabajara, and Moshe Y. Vardi. Hybrid compositional reasoning for reactive synthesis from finite-horizon specifications. In *AAAI*, pages 9766–9774, 2020.

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

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

[De Giacomo and Favorito, 2021a] Giuseppe De Giacomo and Marco Favorito. Compositional approach to translate LTL$_f$/LDL$_f$ into deterministic finite automata. In *ICAPS*, pages 122–130, 2021.

[De Giacomo and Favorito, 2021b] Giuseppe De Giacomo and Marco Favorito. Lydia: A tool for compositional LTL$_f$ /LDL$_f$ synthesis. In *ICAPS*, pages 122–130, 2021.

[De Giacomo and Vardi, 2013] Giuseppe De Giacomo and Moshe Y. Vardi. Linear Temporal Logic and Linear Dynamic Logic on Finite Traces. In *IJCAI*, pages 854–860, 2013.

[De Giacomo and Vardi, 2015] Giuseppe De Giacomo and Moshe Y. Vardi. Synthesis for LTL and LDL on Finite Traces. In *IJCAI*, pages 1558–1564, 2015.

[Finkbeiner, 2016] Bernd Finkbeiner. Synthesis of Reactive Systems. *Dependable Software Systems Eng.*, 45:72–98, 2016.

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

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

[Henriksen *et al.*, 1995] Jesper G. Henriksen, Jakob L. Jensen, Michael E. Jørgensen, Nils Klarlund, Robert Paige, Theis Rauhe, and Anders Sandholm. Mona: Monadic Second-order Logic in Practice. In *TACAS*, pages 89–110, 1995.

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

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

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

[Tabajara and Vardi, 2019] Lucas Martinelli Tabajara and Moshe Y. Vardi. Partitioning techniques in LTL$_f$ synthesis. In *IJCAI*, pages 5599–5606, 2019.

[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.

[Zhu *et al.*, 2020] Shufang Zhu, Giuseppe De Giacomo, Geguang Pu, and Moshe Y. Vardi. LTL$_f$ synthesis with fairness and stability assumptions. In *AAAI*, pages 3088–3095, 2020.