Effective Approach to LTL$_f$ Best-Effort Synthesis in Multi-Tier Environments

Benjamin Aminof$^1$, Giuseppe De Giacomo$^{1,2}$, Gianmarco Parretti$^1$ and Sasha Rubin$^3$

$^1$Università degli Studi di Roma “La Sapienza”
$^2$University of Oxford
$^3$University of Sydney
aminof@forsyte.at, parretti@diag.uniroma1.it, giuseppe.degiacomo@cs.ox.ac.uk, sasha.rubin@sydney.edu.au

Abstract

We consider an agent acting in a complex environment modeled through a multi-tiered specification, in which each tier adds nondeterminism in the environment response to the agent actions. In this setting, we devise an effective approach to best-effort synthesis, i.e., synthesizing agent strategies that win against a maximal set of possible environment responses in each tier. We do this in a setting where both the multi-tier environment and agent goal are specified in the linear temporal logic on finite traces (LTL$_f$). While theoretical solution techniques based on automata on infinite trees have been developed previously, we completely sidestep them here and focus on a DFA-based game-theoretic technique, which can be effectively implemented symbolically. Specifically, we present a provably correct algorithm that is based on solving separately DFA-based games for each tier and then combining the obtained solutions on-the-fly. This algorithm is linear, as opposed to being exponential, in the number of tiers, and thus, it can gracefully handle multi-tier environments formed of several tiers.

1 Introduction

There is a growing interest in Reasoning about Actions, Planning, and Sequential Decision Making on developing autonomous AI systems that can operate effectively in complex and dynamic environments where the level of nondeterminism is high. We typically assume that the AI system, i.e., the agent, has a single or flat model of the environment (specified, e.g., in the Situation Calculus [Reiter, 2001], or in PDDL [Haslum et al., 2019], or in Temporal Logics [Aminof et al., 2018; Camacho et al., 2019; Aminof et al., 2019]), which the agent uses to deliberate how to achieve its goals. However, accurately modeling such environments can be challenging, particularly when there is high degree of uncertainty. Hence, the scientific community has been exploring the concept of multi-tier models of environment behavior, i.e., having simultaneously several models, or tiers, of the environment such that, in each tier the environment is more nondeterministic than in the previous one [Aminof et al., 2020; Ciolek et al., 2020; Aminof et al., 2021a]. For example, an agent may have a tier that represents the expected environment behavior, but also other tiers that represent increasingly nondeterministic deviations from that behavior, due to deteriorated or exceptional responses.

Given a multi-tier environment model, the agent simultaneously reasons on the effects of its actions in all tiers when deliberating what to do. This increases the robustness and adaptability of its operations when deployed in complex and uncertain environments. However, while the agent may have winning strategies (plans) to achieve its goals in the most deterministic tier, it may be impossible to have winning strategies also for the most nondeterministic tiers. This calls for notions of strategies that are less stringent than the usual ones used in Formal Methods [Finkbeiner, 2016], or in Planning [Geffner and Bonet, 2013].

One option is to introduce stochastic/quantitative aspects in the models and base reasoning on optimization with probabilistic guarantees [Geffner and Bonet, 2013]. But also in the non-quantitative setting there are interesting solutions, in particular that of best-effort strategies [Aminof et al., 2020; Aminof et al., 2021a; Aminof et al., 2021b]: if a strategy to win the goal against all possible environment responses does not exist, instead of giving up, we return a strategy that wins against a maximal set (though not all) of possible environment responses. Best-effort strategies are based on the game-theoretic rationality principle that a player (the agent) should not use a strategy that is “dominated” by another one (i.e., if another strategy fulfills the goal against more environment responses, then the player should adopt that strategy). Best-effort strategies have some notable properties: (i) they always exist, (ii) if a winning strategy exists, then best-effort strategies are exactly the winning strategies, (iii) for Linear Temporal Logic specifications both on infinite traces (LTL) [Pnueli, 1977] and on finite traces (LTL$_f$) [De Giacomo and Vardi, 2013], they can be computed in worst case 2EXPTIME, just as for winning strategies (best-effort synthesis is indeed 2EXPTIME-complete, just as is standard synthesis) [Aminof et al., 2021b].

These results extend to multi-tier environments. In particular, a strategy that is best-effort in all tiers of the multi-tier environment model always exists, i.e., there exists a strategy that in every tier wins against a maximal set of environ-
ment strategies. Moreover, such a strategy can be computed in 2EXPTIME for LTL/LTL$_f$ specifications. The latter result has been proved constructively in [Aminof et al., 2021a] by providing a solution technique based on automata on infinite trees. However that automata-based technique is not particularly promising for implementation.

Interestingly, in the case of flat environment models, one can resort to an alternative synthesis technique [Aminof et al., 2021b], which is a game-theoretic construction that can be implemented effectively, especially for LTL$_f$. This technique is based on solving an adversarial and a cooperative game over an arena provided by the environment specification (and the agent goal) and then combining the two solutions.

In this paper, we focus on LTL$_f$. Using LTL$_f$ allows to specify every LTL guarantee specification for the goal and every LTL safety specification for the environment [Manna and Pnueli, 1990]. Notably, safety environment specifications are a generalization of nondeterministic planning domain specifications (for example, they allow for non-Markovian properties [Gabaldon, 2011]), e.g., written in PDDL making use of onef (dropping preconditions in favor of conditional effects) [Haslum et al., 2019; Aminof et al., 2018; De Giacomo et al., 2023a].

Our first contribution is to show that the game-theoretic approach in [Aminof et al., 2021b] can be extended to handle multi-tier environments. Specifically, we show that we can combine adversarial and cooperative games for each tier of the environment and generate a strategy that is best-effort for all of them (Algorithm 1). However, by adopting such a basic technique, we obtain a game arena that is exponential in the number of tiers, limiting the applicability to only few tiers.

Our second contribution is to show that this exponential blowup in the number of tiers can be avoided. We present a refinement (Algorithm 2) of the basic technique, which is based on solving the games corresponding to an environment specification separately and combining the solutions on-the-fly (in linear time) to obtain a strategy that step-wise returns the next action to be performed. The result is an algorithm that is linear in the number of environment specifications, and worst-case doubly exponential only in the size of the formulas specifying the goal and the environments (Theorem 6).

Our third contribution is to analyze two notable cases of multi-tier environments specified in LTL$_f$: (i) the case in which all tiers share a (large) common base component, i.e., have the form $E_i = \mathcal{E}_c \land \mathcal{E}_f^m$, and (ii) the case in which, each tier is obtained by adjoining some further conditions to the previous one, i.e., $E_{i-1} = \mathcal{E}_f \land \mathcal{E}^m_{i-1}$. We exploit this additional structure, getting a construction that is even more scalable.

To show the practicality of the proposed approach, we provide symbolic implementations by leveraging the framework of [Zhu et al., 2017] and, using such implementations, perform an empirical evaluation on some scalable benchmarks.

2 LTL$_f$ Synthesis

A trace over an alphabet of symbols $\Sigma$ is a finite or infinite sequence of elements from $\Sigma$. The empty trace is denoted $\lambda$. Traces are indexed starting at zero, and we write $\pi = \pi_0 \pi_1 \ldots$. For a finite trace $\pi$, let $\text{lst}(\pi)$ denote the index of the last element of $\pi$, i.e., $\text{lst}(\pi) = |\pi| - 1$.

Linear Temporal Logic on finite traces (LTL$_f$) is a specification language for expressing temporal properties on finite traces [De Giacomo and Vardi, 2013]. LTL$_f$ has the same syntax as LTL (which is instead interpreted over infinite traces [Pnueli, 1977]). Given a set $AP$ of atomic propositions (aka atoms), the LTL$_f$ formulas over $AP$ are generated by the following grammar: $\varphi ::= a \mid \varphi \land \varphi \mid \neg \varphi \mid \varphi \lor \varphi \mid \varphi \Rightarrow \varphi \mid \varphi \Leftrightarrow \varphi$ where $a \in AP$. Here $\circ$ (Next) and $\mathcal{U}$ (Until) are temporal operators. We use standard Boolean abbreviations such as $\lor$ (or), $\land$ (and), $\neg$ (not), $\Rightarrow$ (implies), $\equiv$ (equals), true and false. Moreover, we define the following abbreviations: $\diamond \varphi \equiv \neg \square \neg \varphi$ (Weak Next), $\square \varphi \equiv \varphi$ (Eventually), and $\lozenge \varphi \equiv \neg \square \neg \varphi$ (Always).

The size of $\varphi$, written $|\varphi|$, is the number of its subformulas. Formulas are interpreted over finite traces $\pi$ over the alphabet $\Sigma = 2^{AP}$, i.e., the alphabet consisting of the propositional interpretations of the atoms. Thus, for $0 \leq i \leq |\text{lst}(\pi)|$, $\pi_i \in 2^{AP}$ is the $i$-th interpretation of $\pi$. That an LTL$_f$ formula $\varphi$ holds at instant $i \leq \text{lst}(\pi)$, written $\pi_i \models \varphi$, is defined inductively: 1. $\pi_i \models a$ if $a \in \pi_i$ (for $a \in AP$); 2. $\pi_i \models \neg \varphi$ if $\pi_i \not\models \varphi$; 3. $\pi_i \models \varphi_1 \land \varphi_2$ if $\pi_i \models \varphi_1$ and $\pi_i \models \varphi_2$; 4. $\pi_i \models \varphi_1 \lor \varphi_2$ if $\pi_i \not\models \varphi_1$ and $\pi_i \models \varphi_2$; 5. $\pi_i \models \varphi_1 \Rightarrow \varphi_2$ if $\pi_i \not\models \varphi_2$ or $\forall k < |\text{lst}(\pi)|$ such that $\pi_i \models \varphi_1$ and $\pi_j \models \varphi_2$ for all $k \leq j \leq |\text{lst}(\pi)|$, and $\forall k \leq i < j$ we have that $\pi_j \not\models \varphi_1$. We say that $\pi$ satisfies $\varphi$, written $\pi_i \models \varphi$, if $\pi, 0 \models \varphi$.

LTL$_f$ (reactive) synthesis [De Giacomo and Vardi, 2015] concerns finding a strategy to satisfy an LTL$_f$ goal specification. Goals are expressed as LTL$_f$ formulas over $AP = \mathcal{X} \cup \mathcal{Y}$, where $\mathcal{X}$ and $\mathcal{Y}$ are disjoint sets of variables. Intuitively, $\mathcal{X}$ (resp. $\mathcal{Y}$) is under the agent’s (resp. environment’s) control. Traces over $\Sigma = 2^{\mathcal{X} \cup \mathcal{Y}}$ will be denoted $\pi = (Y_0 \cup X_0)(Y_1 \cup X_1)\ldots$ where $X_i \subseteq \mathcal{X}$ and $Y_i \subseteq \mathcal{Y}$ for every $i$. Such infinite traces are called plays, and finite traces are called histories and represent a sequence of moves of the players ending in an environment move since we assume that the agent moves first.

An agent strategy is a function $\sigma_{ag} : (2^X)^* \rightarrow (2^Y)^*$ mapping sequences of environment moves to an agent move. Similarly, an environment strategy is a function $\sigma_{env} : (2^Y)^* \rightarrow (2^X)^*$ mapping non-empty sequences of agent moves to an environment move. The domain of $\sigma_{ag}$ includes the empty sequence $\lambda$ as we assumed that the agent moves first. A trace $\pi$ is $\sigma_{ag}$-consistent if $Y_0 = \sigma_{ag}(\lambda)$ and $Y_{j+1} = \sigma_{ag}(X_0 \cdots X_j)$ for every $j \geq 0$. Analogously, $\pi$ is $\sigma_{env}$-consistent if $X_j = \sigma_{env}(Y_0 \cdots Y_j)$ for every $j \geq 0$. We define $\text{PLAY}(\sigma_{ag}, \sigma_{env})$ to be the unique (infinite) trace that is consistent with both $\sigma_{ag}$ and $\sigma_{env}$.

Let $\varphi$ be an LTL$_f$ formula over $\mathcal{X} \cup \mathcal{Y}$. An agent strategy $\sigma_{ag}$ is winning for (aka enforces) $\varphi$ if, for every environment strategy $\sigma_{env}$, some finite prefix of $\text{PLAY}(\sigma_{ag}, \sigma_{env})$ satisfies $\varphi$. An agent strategy is cooperatively winning for $\varphi$ if there exists an environment strategy $\sigma_{env}$ such that some finite prefix of $\text{PLAY}(\sigma_{ag}, \sigma_{env})$ satisfies $\varphi$. LTL$_f$ synthesis is the problem of finding an agent strategy $\sigma_{ag}$ that enforces $\varphi$, if one exists [De Giacomo and Vardi, 2015].

In this paper, we are interested in LTL$_f$ synthesis under environment specifications [Aminof et al., 2018]. Environment
specifications describe some knowledge about how the environment works and are expressed as $LTL_f$ formulas $E$ over $\mathcal{Y} \cup \mathcal{X}$. An environment strategy $\sigma_{env}$ is winning for (aka enforces) $E$ if, for every agent strategy $\sigma_{ag}$, every finite prefix of $PLAY(\sigma_{ag}, \sigma_{env})$ satisfies $E$. An environment specification is an $LTL_f$ formula $E$ that is enforceable by some environment strategy. We denote by $\Sigma_E$ the set of environment strategies that enforce $E$.

**Definition 1.** [Aminof et al., 2018] Let $\varphi$ (resp. $E$) be an $LTL_f$ formula over $\mathcal{Y} \cup \mathcal{X}$ denoting an agent goal (resp. env spec). $LTL_f$ Synthesis under environment specifications is the problem of finding an agent strategy $\sigma_{ag}$ such that, for every environment strategy $\sigma_{env} \in \Sigma_E$, some finite prefix of $PLAY(\sigma_{ag}, \sigma_{env})$ satisfies $\varphi$, if one exists. Such a strategy is winning for $\varphi$ in $E$ (aka enforces $\varphi$ in $E$).

An agent strategy $\sigma_{ag}$ is cooperatively winning for $\varphi$ in $E$ if there exists an environment strategy $\sigma_{env} \in \Sigma_E$ such that $PLAY(\sigma_{ag}, \sigma_{env})$ has a finite prefix that satisfies $\varphi$.

Synthesis, both with $LTL_f$ environment specifications or not, is $2EXPTIME$-complete [De Giacomo and Vardi, 2015; Aminof et al., 2018]. Synthesis under environment specifications is a generalization of synthesis which is obtained by taking $E = true$.

## 3 Best-Effort Strategies

We start by recalling basic notions on best-effort strategies [Aminof et al., 2020; Aminof et al., 2021b; Aminof et al., 2021a].

**Definition 2.** Let $\varphi$ and $E$ be $LTL_f$ formulas over $\mathcal{Y} \cup \mathcal{X}$ denoting an agent goal and an environment specification, respectively, and let $\sigma_1$ and $\sigma_2$ be agent strategies. We say that $\sigma_1$ dominates $\sigma_2$, written $\sigma_1 \geq_{\varphi|E} \sigma_2$, if, for every $\sigma_{env} \in \Sigma_E$, if some finite prefix of $PLAY(\sigma_2, \sigma_{env})$ satisfies $\varphi$ then some finite prefix of $PLAY(\sigma_1, \sigma_{env})$ satisfies $\varphi$. Furthermore, $\sigma_1$ strictly dominates $\sigma_2$, written $\sigma_1 >_{\varphi|E} \sigma_2$, if $\sigma_1 \geq_{\varphi|E} \sigma_2$ and $\sigma_2 \not>_{\varphi|E} \sigma_1$.

Intuitively, $\sigma_1 >_{\varphi|E} \sigma_2$ means that $\sigma_1$ does at least as well as $\sigma_2$ against every environment strategy enforcing $\varphi$ and strictly better against at least one such strategy. An agent using $\sigma_2$ is not doing its best, since it could achieve its goal against a strictly larger set of environment strategies using $\sigma_1$. In this framework, a best-effort strategy is one which is not strictly dominated by any other strategy.

**Definition 3.** An agent strategy $\sigma$ is best-effort, or maximal, for $\varphi$ in $E$, written $\sigma \in \text{Max}_{\varphi|E}$, if there does not exist another agent strategy $\sigma'$ such that $\sigma' >_{\varphi|E} \sigma$.

Best-effort strategies also admit a local characterization that uses the notion of value of a history [Aminof et al., 2021b]. Intuitively, the value of a history $h$ is: “winning”, if the agent can enforce $\varphi$ in $E$ from $h$; otherwise, “pending”, if the agent has a cooperatively winning strategy for $\varphi$ in $E$ from $h$; otherwise, “losing”. With this notion, best-effort strategies are those that witness the maximum value of each history $h$ consistent with them.

For a history $h$ and an agent strategy $\sigma_{ag}$, we denote by $\Sigma_E(h, \sigma_{ag})$ the set of environment strategies $\sigma_{env}$ enforcing $E$ such that $h$ is consistent with $\sigma_{ag}$ and $\sigma_{env}$. For an agent strategy $\sigma_{ag}$, we denote by $H_E(\sigma_{ag})$ the set of all histories $h$ such that $\Sigma_E(h, \sigma_{ag})$ is non-empty, i.e., $H_E(\sigma_{ag})$ is the set of all histories that are consistent with $\sigma_{ag}$ and some environment strategy enforcing $E$. For $h \in H_E(\sigma_{ag})$ define:

1. $\text{val}_{\varphi|E}(\sigma_{ag}, h) = +1$ (“winning”), if for every $\sigma_{env} \in \Sigma_E(h, \sigma_{ag})$, $PLAY(\sigma_{ag}, \sigma_{env})$ has a finite prefix that satisfies $\varphi$; otherwise,
2. $\text{val}_{\varphi|E}(\sigma_{ag}, h) = 0$ (“pending”), if for some $\sigma_{env} \in \Sigma_E(h, \sigma_{ag})$, $PLAY(\sigma_{ag}, \sigma_{env})$ has a finite prefix that satisfies $\varphi$; otherwise,
3. $\text{val}_{\varphi|E}(\sigma_{ag}, h) = -1$ (“losing”).

Finally, we denote by $\text{val}_{\varphi|E}(h)$ the maximum of $\text{val}_{\varphi|E}(\sigma_{ag}, h)$ over all $\sigma_{ag}$ such that $h \in H_E(\sigma_{ag})$ (we define $\text{val}_{\varphi|E}(h)$ only in case $h \in H_E(\sigma)$ for some $\sigma$). Here is the local characterization of best-effort strategies:

**Theorem 1.** [Aminof et al., 2021b] A strategy $\sigma_{ag}$ is best-effort for $\varphi$ in $E$ (i.e., $\sigma_{ag} \in \text{Max}_{\varphi|E}$) iff $\text{val}_{\varphi|E}(\sigma_{ag}, h) = \text{val}_{\varphi|E}(h)$ for every $h \in H_E(\sigma_{ag})$.

## 4 Best-Effort Synthesis in Multi-Tier Environments

We now introduce best-effort synthesis in multi-tier environments, i.e., environment models consisting of several tiers, each allowing more nondeterminism than the previous one. Formally, an $LTL_f$ multi-tier environment specification (aka multi-tier environment model) is a tuple $E = (\mathcal{E}_1, \ldots, \mathcal{E}_n)$ of $LTL_f$ environment tiers such that $\Sigma_{\mathcal{E}_i} \subseteq \Sigma_{\mathcal{E}_{i+1}}$ for every $i$ ($i \leq i < n$). In this framework, a best-effort strategy is one that is simultaneously best-effort for every tier.

**Definition 4.** Given an $LTL_f$ goal $\varphi$ and an $LTL_f$ multi-tier environment specification $E = (\mathcal{E}_1, \ldots, \mathcal{E}_n)$, best-effort synthesis is the problem of finding an agent strategy that is best-effort for $\varphi$ in $E$, i.e. such that $\sigma_{ag} \in \bigcap_i \text{Max}_{\varphi|\mathcal{E}_i}$.

Unlike classic $LTL_f$ synthesis [De Giacomo and Vardi, 2015; Pnueli and Rosner, 1989], a best-effort strategy for an $LTL_f$ goal $\varphi$ in a $LTL_f$ multi-tier environment specification $E$ always exists, though computing it requires $2EXPTIME$ as in classic synthesis [Aminof et al., 2021a].

**Theorem 2.** [Aminof et al., 2021a] Let $\varphi$ be an $LTL_f$ goal and $E = (\mathcal{E}_1, \ldots, \mathcal{E}_n)$ an $LTL_f$ multi-tier environment specification. There exists $\sigma_{ag} \in \bigcap_i \text{Max}_{\varphi|\mathcal{E}_i}$, and it can be computed in $2EXPTIME$ in the size of $\varphi, \mathcal{E}_1, \ldots, \mathcal{E}_n$.

We now illustrate such notions in a simple Robot Navigation (in FOND domains) scenario [Cimatti et al., 2003; Alford et al., 2014] where an agent has to plan in spite of increasing nondeterminism.

**Example 1.** An autonomous agent is assigned the goal of delivering packages in a building by moving across rooms. Assume that there is a kid in the building who has keys to close some doors. It is easy to see that the agent goal may not be realizable as, e.g., the kid might lock the robot in a room. Hence, the agent could use a best-effort strategy. In this scenario, an $LTL_f$ environment describes the initial state, the transitions of the planning domain, and that the kid might...
5 Solving Best-Effort Synthesis in Multi-Tier Environments

While [Aminof et al., 2021a] provide a solution technique for best-effort synthesis in multi-tier domains, their technique is based on automata on infinite trees and is not well suited for efficient implementation. Here, we provide a different technique based on DFA games, as that for best-effort synthesis in a flat environment from [Aminof et al., 2021b], but extended to handle multi-tier environments.

Deterministic Finite Automata. For convenience, we separate the acceptance condition of automata from their structure. We define a deterministic transition system (aka transition system) as a tuple $D = (\Sigma, S, s_0, \delta)$, where: $\Sigma$ is a finite input alphabet (usually $\Sigma = 2^{\delta \cup X}$); $S$ is a finite set of states; $s_0 \in S$ is the initial state; and $\delta : S \times S \rightarrow \delta$ is the transition function. The size of $D$ is the cardinality of $S$. Let $\alpha = \alpha_0\alpha_1 \ldots \alpha_n$ be a finite trace over the alphabet $\Sigma$. The run of $\alpha$ in $D$ is the finite sequence of states $\rho = s_0\alpha_1 \ldots s_{n+1}$ such that $s_0$ is the initial state of $D$ and $s_{i+1} = \delta(s_i, \alpha_i)$ for every $i \leq \text{fst}(\alpha)$. We extend $\delta$ to be a function $\delta : S \times \Sigma^* \rightarrow S$ as follows: $\delta(s, \lambda) = s$, and if $s_n = \delta(s, \alpha_0 \ldots \alpha_{n-1})$ then $\delta(s, \alpha_0 \ldots \alpha_n) = \delta(s_n, \alpha_n)$.

Definition 5. The synchronous product of two transition systems $D_i = (\Sigma_i, S_i, s_{i0}, \delta_i)$ (for $i = 1, 2$) over the same alphabet is the transition system $\text{PRODUCT}(D_1, D_2) = D_1 \times D_2 = (\Sigma, S, s_0, \delta)$ with: $S = S_1 \times S_2$; $s_0 = (s_{01}, s_{02})$; and $\delta((s_1, s_2), x) = \delta(s_1, x)\delta(s_2, x)$. The product $\text{PRODUCT}(D_1, \ldots, D_n)$ is defined analogously for any finite sequence $D_1, \ldots, D_n$ of transition systems over the same alphabet.

A deterministic finite automaton (DFA) is a pair $A = (D, F)$, where $D = (\Sigma, S, s_0, \delta)$ is a deterministic transition system and $F \subseteq S$ is the set of final states of the system. A trace $\alpha$ is accepted if $\delta(s_0, \alpha) \in F$. The language of $A$ is the set of traces that the automaton accepts.

Theorem 3. [De Giacomo and Vardi, 2013] Given an LTL$_f$ formula $\varphi$ over $AP$, we can build a DFA, denoted $\text{TDFA}(\varphi)$, whose size is at most $2\text{EXP}$ in $|\varphi|$ and whose language is the set of finite traces that satisfy $\varphi$.
Algorithm 0 SYNTHPOS($\varphi, \mathcal{E}$)

Input: LTL$_f$ goal $\varphi$ and an env. spec. $\mathcal{E}$
Output: goal DFA $A_{\varphi}$; env. DFA $A_{\mathcal{E}}$; winning region $W$; cooperatively winning region $W'$; positional winning strategy $\kappa$; positional cooperatively winning strategy $\gamma$

1: $A_{\varphi} = \text{toDFA}(\varphi)$; $A_{\mathcal{E}} = \text{toDFA}(\mathcal{E})$
Say $A_{\varphi} = (D_{\varphi}, F_{\varphi})$ and $A_{\mathcal{E}} = (D_{\mathcal{E}}, F_{\mathcal{E}})$
2: $D = \text{PRODUCT}(D_{\varphi}, D_{\mathcal{E}})$
3: Let:
   - $F_{D\supset\varphi} = \{(s, s') | s, s' \in F_{D} \supset s \neq s' \in F_{D}\}$
   - $F_{D\wedge\varphi} = \{(s, s') | s, s' \in F_{D} \wedge s \neq s' \in F_{D}\}$
4: $(W, \kappa) = \text{SOLVEADV}(D, F_{D\supset\varphi})$
5: $V = \text{ENYWIN}(D, F_{\mathcal{E}})$
6: $D' = \text{RESTRICT}(D, V)$
7: $(W', \gamma) = \text{SOLVECOOP}(D', F_{D\wedge\varphi})$
8: Return $(A_{\varphi}, A_{\mathcal{E}}, W, W', \kappa, \gamma)$

Algorithm 1 MULTIENVBESYNTH($\varphi, \mathcal{E}_1, \ldots, \mathcal{E}_n$)

Input: LTL$_f$ goal $\varphi$ and a multi-tier env. $\mathcal{E} = (\mathcal{E}_1, \ldots, \mathcal{E}_n)$
Output: Agent strategy $\sigma$ that is best-effort for $\varphi$ in $\mathcal{E}$

1: For $i = 1 \ldots n$:
   - $(A_{\varphi}, A_{\mathcal{E}_i}, W_i, W'_i, \kappa_i, \gamma_i) = \text{SYNTHPOS}(\varphi, \mathcal{E}_i)$
   - Say $A_{\varphi} = (D_{\varphi}, F_{\varphi})$, $D_{\mathcal{E}_i} = (2^n \cup \mathcal{E}_i, S_{\mathcal{E}_i}, s_{\mathcal{E}_i}, \delta_{\mathcal{E}_i})$
   - Say $A_{\mathcal{E}_i} = (D_{\mathcal{E}_i}, F_{\mathcal{E}_i})$, $D_{\mathcal{E}_i} = (2^n \cup \mathcal{E}_i, S_{\mathcal{E}_i}, s_{\mathcal{E}_i}, \delta_{\mathcal{E}_i})$
2: $D = \text{PRODUCT}(D_{\varphi}, \ldots, D_{\mathcal{E}_i}, \ldots, D_{\mathcal{E}_n})$
3: Define a positional strategy $\nu$ on $S$ as follows.
   - For $s = (s_1, \ldots, s_n, t) \in S$:
     - $j = \max\{i : (s_i, t) \in W_i\}$
     - $\ell = \min\{i : (s_i, t) \in W_i\}$
     - If $\ell$ exists then define $\nu(s) = \kappa_j(s_j, t)^4$
     - Else if $\ell$ exists then define $\nu(s) = \gamma_\ell(s_\ell, t)$
     - Else define $\nu(s) = Y$ (i.e., arbitrarily) endif
4: Return $\text{STRATEGY}(D, \nu)$

single environment LTL$_f$ best-effort synthesis problem, encapsulated in Algorithm 0. That allows one to compute a positional strategy as follows: it maps a state $s$ in $D = D_{\varphi} \times D_{\mathcal{E}}$ to $\kappa(s)$ if $s \in W$, to $\gamma(s)$ if $s \in W' \setminus W$, and is arbitrary otherwise. Intuitively, the histories whose induced runs in $D$ that pass or end in a state in $W$ have value $+1$ (as witnessed by $\kappa$), those ending in a state in $W' \setminus W$ have value $0$ (as witnessed by $\gamma$), and the rest have value $-1$. The correctness is a consequence of the local characterization (Theorem 1).

With the auxiliary procedure Algorithm 0 in place, we are ready to present our solution technique. For we make no effort to gain maximal efficiency, which we will do in the next section, this solution should be thought of as a conceptual solution and not yet a blueprint for implementation. The technique is detailed in Algorithm 1 and returns a strategy obtained by combining the solutions of simple (adversarial and cooperative) DFA games, two for each environment specification $\mathcal{E}_i$, computed in Step 1 by calling Algorithm 0. These strategies are combined into a positional strategy over the Cartesian product of all games computed in Step 3. Algorithm 1 exploits the fact the environment is given in the form of a multi-tier environment, i.e., $\Sigma_{\mathcal{E}_1} \subseteq \cdots \subseteq \Sigma_{\mathcal{E}_n}$, as follows. Suppose $k < i$. Then, an agent strategy that wins for $\varphi$ in $\mathcal{E}_i$ also wins for $\varphi$ in $\mathcal{E}_k$ since winning against all the strategies in $\Sigma_{\mathcal{E}_k}$ also wins against all the strategies in the subset $\Sigma_{\mathcal{E}_k}$. Similarly, an agent strategy that cooperatively wins for $\varphi$ in $\mathcal{E}_k$ also cooperatively wins for $\varphi$ in $\mathcal{E}_i$ since a cooperating environment strategy in $\Sigma_{\mathcal{E}_k}$ is also in $\Sigma_{\mathcal{E}_i}$. Intuitively, histories whose induced runs in the product $D$ that pass or end in a state whose $j$-th coordinate is in $W_j$ (where $j$ is computed in Step 3) have value $+1$ for each of the environment specifications $\mathcal{E}_i$ up to $\mathcal{E}_j$, and have value $0$ for each of the environment specifications $\mathcal{E}_{i+1}$ up to $\mathcal{E}_n$; of the remaining histories, those ending in a state whose $\ell$-th coordinate is
Exploiting Structure in Notable Cases. By taking advantage of the syntactic structure of these notable cases, we can devise optimized variants of Algorithm 2 that construct more efficiently the DFAs of the tiers. To do this, we exploit the following composition technique, whose correctness follows immediately by the notion of product of transition systems:

Theorem 7. Given $n$ LTL$_f$ formulas, $\psi$, let $\psi = \bigwedge_{1 \leq i \leq n} \psi_i$. If $A_{\psi_i} = (D_{\psi_i}, F_{\psi_i})$ is a DFA recognizing $\psi_i$ (for $1 \leq i \leq n$), then the DFA $A = (D, F)$ recognizes $\psi = \prod(D_{\psi_1}, \ldots, D_{\psi_n})$ and $F = F_{\psi_1} \times \cdots \times F_{\psi_n}$.

With Theorem 7, we can construct the DFAs of the tiers as follows: (i) we construct the DFA of the base conjunct, i.e., $E_c$ and $E_n$, respectively; (ii) we construct the DFAs of the conjuncts $E'_i$; (iii) we compose the obtained DFAs to construct the DFAs of the environment tiers. Constructing and composing the DFAs of the various conjuncts takes less time than transforming every tier into a DFA as a whole, especially if the size of the least refined tier is large and dominates that of the other conjuncts. This is confirmed empirically in Section 8.

8 Implementation and Evaluation

We implemented Algorithm 2 in a tool called MtSyft$^3$, leveraging the symbolic LTL$_f$ synthesis framework [Zhu et al., 2017], at the base of state-of-the-art LTL$_f$ synthesis tools [Bansal et al., 2020; Favorito and Zhu, 2023]. We also developed variants of MtSyft customized for the two notable cases above, called $c$-MtSyft and $conj$-MtSyft. In MtSyft, we build the minimized explicit-state DFAs of LTL$_f$ formulas with LyDia [De Giacomo and Favorito, 2021], which is among the best performing tools publicly available for LTL$_f$-to-DFA conversion. We code Boolean functions representing transitions and final states of symbolic DFAs by BDDs [Bryant, 1992] with the BDD library CUDD 3.0.0 [Somenzi, 2016]. We compute the positional strategies for the DFA-games through Boolean synthesis [Fried et al., 2016].

Setup. Experiments were run on a laptop with an operating system 64-bit Ubuntu 20.04, 3.6 GHz CPU, and 12 GB of memory. Timeout was set to 300 seconds.

Benchmark. To evaluate the performance of our implementations, we devised an extension of the counter game benchmarks presented in [De Giacomo et al., 2020b; Zhu et al., 2020] to construct multi-tier environment specifications. The counter game involves a $k$-bit counter as follows: (i) at each round, the environment chooses whether to request an increment of the counter ($add$), and the agent chooses whether to grant such a request or not; (ii), the counter is initialized with all bits set to 0, and the agent goal is for the counter to have all bits set to 1; (iii) multi-tier environment specifications define possible policies according to which the environment issues increment requests. Specifically, environment specifications are LTL$_f$ formulas $E_1 = add$, and for $m \geq 2$, $E_m = E_{m-1} \land \bullet \cdots \bullet add$, where there are $m - 1$ occurrences of $\bullet$ in $E_m$. In our experiments, the environment issues between 1 and 100 increment requests ($1 \leq m \leq 100$). Given a $k$-bits counter and an environment specification $E_m$,

$\text{https://github.com/GianmarcoDIA/MtSyft}$

---

**Algorithm 2** ONTHETFLYBE\text{SYNTH}(\varphi, E_1, \ldots, E_n)

**Input:** LTL$_f$ goal $\varphi$ and a multi-tier env. $E = (E_1, \ldots, E_n)$

**Output:** Agent strategy $\sigma$ that is best-effort for $\varphi$ in $E$

1: For $i = 1 \ldots n$
   1. $(A_{\varphi}, A_{E_i}, W_i, W'_i, \kappa_i, \gamma_i) = \text{SYNTHPOS}(\varphi, E_i)$
   2. Say $A_{\varphi} = (D_{\varphi}, F_{\varphi})$, $D_{\varphi} = (2^{|w|}_\text{set}, S_{\varphi}, s_{\varphi}, \delta_{\varphi})$
   3. Say $A_{E_i} = (D_{E_i}, F_{E_i})$, $D_{E_i} = (2^{|w_i|}_\text{set}, S_{E_i}, s_{E_i}, \delta_{E_i})$

2: Return the following best-effort strategy:

**While true:**

1. $j = \max \{ i : (s_{E_i}, s_{\varphi}) \in W'_i \}$
2. $\ell = \min \{ i : (s_{E_i}, s_{\varphi}) \in W'_i \}$
3. if $j$ exists, output $Y = s_j(s_{E_j}, s_{\varphi})$
4. else if $\ell$ exists, output $Y = s_{\ell}(s_{E_\ell}, s_{\varphi})$
5. else output $Y = Y'$ end if
6. On environment’s choice $X \subseteq X'$:
   - Update $s_{\varphi} = \delta_{\varphi}(s_{\varphi}, Y \cup X)$
   - For $i = 1 \ldots n$: update $s_{E_i} = \delta_{E_i}(s_{E_i}, Y \cup X)$

---

**Theorem 6.** Let $\varphi$ be an LTL$_f$ goal and $E = (E_1, \ldots, E_n)$ a multi-tier environment specification. Then Algorithm 2 computes a strategy $\sigma \in \bigcap \text{Max}_{\varphi,|E|}$ in 2EPXTIME in the size of $\varphi, E_1, \ldots, E_n$ and in linear time in $n$, the number of tiers in the multi-tier environment.

Specifically, Algorithm 2 finds, at each instant (history), the next agent move (assignment of the $Y$) in linear time in the number of tiers, i.e., Algorithm 2 (differently form Algorithm 1) scales graciously as the number of tiers grows.

Interestingly, the computations in Step 1 of Algorithms 1 and 2 can be done in parallel. The $n + 1$ steps for computing the DFAs of the goal and the environment specifications can be done in parallel; the $n$ steps for computing the regions $W_i$ and the strategies $\kappa_i$ can be done in parallel; the $n$ steps for computing the regions $W'_i$ and the strategies $\gamma_i$ can be done in parallel. As a result, if $n + 1$ processors are available, handling multi-tier environments is virtually for free, i.e., costs the same as handling the most computationally expensive tier.

These features suggest that Algorithm 2 is suited for efficient implementation, as confirmed empirically in Section 8.

7 Notable Cases

Before turning to implementation and experimental evaluation, we consider two notable cases of multi-tier environment models, for which we can offer further optimizations.

Multi-Tier Environments with a Common Base. In this case we have a (large) common base $E_c$, that is common to all tiers. That is, each tier $E_i$ is specified as conjunction of $E_c$ with some additional LTL$_f$ specification $E'_i$. Formally, multi-tier environments with a common base have the form: for every $i$ s.t. $1 \leq i \leq n$, $E_i = E_c \wedge E'_i$, where $\Sigma E'_i \subseteq \cdots \subseteq \Sigma E'_n$.

Multi-Tier Environments with Conjunctive Refinements. Next, we consider multi-tier environments consisting of tiers that conjoin further constraints to the previous tier, becoming more determined. That is, the base environment is $E_n$, and each tier $E_i$ refines $E_{i+1}$ with some conjunct $E'_i$. Formally, multi-tier environments with conjunctive refinements have the form: for every $i$ s.t. $1 \leq i < n$, $E_i = E_{i+1} \wedge E'_i$.

---

3\text{https://github.com/GianmarcoDIA/MtSyft}
Table 1: Coverage (solved instances out of 100) and average runtime (Avg. RT) achieved by MtSyft and cb-MtSyft in counter games with base conjunct \( \mathcal{E}_1 \) and number of tiers \( 1 \leq n \leq 100 \).

<table>
<thead>
<tr>
<th>Bits</th>
<th>Coverage MtSyft</th>
<th>Coverage cb-MtSyft</th>
<th>Avg. RT (secs) MtSyft</th>
<th>Avg. RT (secs) cb-MtSyft</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>82</td>
<td>82</td>
<td>30.98</td>
<td>54.92</td>
</tr>
<tr>
<td>2</td>
<td>82</td>
<td>82</td>
<td>55.85</td>
<td>51.46</td>
</tr>
<tr>
<td>3</td>
<td>81</td>
<td>83</td>
<td>52.04</td>
<td>54.95</td>
</tr>
<tr>
<td>4</td>
<td>80</td>
<td>82</td>
<td>49.55</td>
<td>53.40</td>
</tr>
<tr>
<td>5</td>
<td>82</td>
<td>84</td>
<td>55.75</td>
<td>57.99</td>
</tr>
<tr>
<td>6</td>
<td>82</td>
<td>82</td>
<td>58.84</td>
<td>55.33</td>
</tr>
<tr>
<td>7</td>
<td>81</td>
<td>80</td>
<td>64.05</td>
<td>57.64</td>
</tr>
<tr>
<td>8</td>
<td>76</td>
<td>77</td>
<td>73.42</td>
<td>74.94</td>
</tr>
<tr>
<td>9</td>
<td>0</td>
<td>0</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>10</td>
<td>0</td>
<td>0</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Total</td>
<td>646</td>
<td>652</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 1: Coverage (solved instances out of 100) and average runtime (Avg. RT) achieved by MtSyft and cb-MtSyft in counter games with base conjunct \( \mathcal{E}_1 \) and number of tiers \( 1 \leq n \leq 100 \).

the realizability of the agent goal (existence of a winning strategy for the goal) depends on \( k \) and \( m \): if \( m \geq 2^k - 1 \), the goal is realizable. Regardless of the realizability of the agent goal, a best-effort (possibly winning) strategy for the agent is to accept all environment increment requests.

Our benchmark consists of counter games with at most 10-bits. For each game, we constructed multi-tier environments with \( n \) tiers as follows: (i) we fixed a base conjunct \( \mathcal{E}_1 \); (ii) stacked the tiers \( \mathcal{E}_1, \ldots, \mathcal{E}_{k+n-1} \) in increments of 1. As base conjuncts, we considered \( \mathcal{E}_1 \), and \( \mathcal{E}_{10} \leq \mathcal{E}_{90} \) in increments of 10. In total, our benchmark consists of about 5600 instances.

**Empirical Results.** We performed experiments to assess: (i) the practical feasibility of best-effort synthesis in multi-tier environments as the number of tiers grows; and (ii) further scalability improvement obtainable exploiting the special structure of the two notable cases.

Table 1 shows the performance of MtSyft in counter game instances with number of tiers between 1 and 100 (1 \( \leq n \leq 100 \)) where \( \mathcal{E}_1 \) is the base conjunct. We can see that MtSyft, run on a laptop, solves at most 8-bits counter games up to 76 tiers within the 300 secs timeout. For 8-bits counter games (or lower) the computational bottleneck is converting LTL\(_{\mathcal{F}}\) tier specifications into DFAs. Instead, solving the single games and composing the synthesized positional strategies into a DFA, but also secondary goals that can be achieved even if the goal is weakened. For instance, the agent may have a primary goal for all tiers. However, it is also of interest to consider the case in which, as tiers become more undetermined, also the goal is weakened. For instance, the agent may have a primary goal that requires a certain type of environment behavior, but also secondary goals that can be achieved even if the environment does not behave as expected. This was studied for PDDL planning in [Ciolek et al., 2020]. We believe that the general approach presented here can be extended to handle this case as well. We leave the details for future work.

![Figure 1: MtSyft and cb-MtSyft comparison on 8-bits counter games with base conjunct \( \mathcal{E}_{80} \) and number of tiers \( 1 \leq n \leq 20 \).](image1)

![Figure 2: MtSyft and conj-MtSyft comparison on 1-bit counter games with base conjunct \( \mathcal{E}_{80} \) and number of tiers \( 1 \leq n \leq 20 \).](image2)

9 Conclusion

We developed an effective technique to solve LTL\(_{\mathcal{F}}\) best-effort synthesis in multi-tier environments which allow for increasing nondeterminism. In our framework, we have considered a single goal for all tiers. However, it is also of interest to consider the case in which, as tiers become more undetermined, also the goal is weakened. For instance, the agent may have a primary goal that requires a certain type of environment behavior, but also secondary goals that can be achieved even if the environment does not behave as expected. This was studied for PDDL planning in [Ciolek et al., 2020]. We believe that the general approach presented here can be extended to handle this case as well. We leave the details for future work.
Acknowledgments

This work is supported in part by the ERC Advanced Grant WhiteMech (No. 834228), the EU ICT-48 2020 project TAILOR (No. 952215), the PRIN project RIPER (No. 20203FYLYK), and the PNRR MUR project FAIR (No. PE0000013). Gianmarco Parretti is supported by the Italian National Ph.D. on Artificial Intelligence at Sapienza.

References


[Haslum et al., 2019] Patrik Haslum, Nir Lipovetzky, Daniele Magazzeni, and Christian Muise. An Introduction


