Best-Effort Synthesis: Doing Your Best Is Not Harder Than Giving Up

Benjamin Aminof[1] , **Giuseppe De Giacomo**[2] and **Sasha Rubin**[3]
[1]TU Vienna, Austria
[2]University of Rome "La Sapienza", Italy
[3]University of Sydney, Australia
aminof@forsyte.at, degiacomo@diag.uniroma1.it, sasha.rubin@sydney.edu.au

Abstract

We study best-effort synthesis under environment assumptions specified in LTL, and show that this problem has exactly the same computational complexity of standard LTL synthesis: 2EXPTIME-complete. We provide optimal algorithms for computing best-effort strategies, both in the case of LTL over infinite traces and LTL over finite traces (i.e., LTL$_f$). The latter are particularly well suited for implementation.

1 Introduction

We study *best-effort synthesis under environment assumptions* in Linear-time Temporal Logic (LTL), the most used logic for dynamic system specifications in Verification [Baier *et al.*, 2008]. Given an LTL specification of the possible behaviors of the environment (i.e., an assumption on the strategies that the environment can enact), and an LTL specification of the task (or goal) for the agent acting in such environments, our aim is to synthesize a strategy for the agent [Church, 1963; Pnueli and Rosner, 1989; Finkbeiner, 2016] that accomplishes the task against all the relevant strategies of the environment [D'Ippolito *et al.*, 2018; De Giacomo and Rubin, 2018; Camacho *et al.*, 2018; Aminof *et al.*, 2019].

What do synthesis procedures return when the agent cannot always achieve its task even by exploiting the assumption on the environment? Typically, the algorithms "give up" declaring the task un-realizable. Here, instead, 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*. The importance of such strategies is clear if the environment is not purely adversarial. This happens, e.g., when the assumptions over-approximate the environment (i.e., include environment strategies that are not possible in reality) or when environments are agnostic to the agent. See, e.g., [Faella, 2009; Bloem *et al.*, 2014].

Observe that quantitative approaches based on optimization, such as MDPs, also always return strategies (or memoryless policies, in case of MDPs), that are in a sense "optimal", e.g., wrt expected rewards [Bertsekas, 2005]. Here, we focus on synthesis purely from logical specifications. We do not have external information about the probabilities of certain environment reactions, nor of utilities of taking certain actions. We do however have a logical specification which qualitatively defines the possible environment behaviors.

The main result of this paper is that best-effort synthesis under environment assumptions has exactly the same computational complexity as standard synthesis: 2EXPTIME-complete. Hence, "doing your best is not harder than giving up"! While it was already known that best-effort synthesis under environment assumptions was decidable [Aminof *et al.*, 2020], its complexity appeared to be much higher than standard synthesis: e.g., if one was to express this problem naturally in Strategy Logic [Mogavero *et al.*, 2014], one would get a 4EXPTIME upper-bound. Our algorithm employs a novel technique based on simultaneously playing multiple games with different objectives on the same arena. Then, by combining the different winning strategies, we get the desired best-effort strategy. We apply this technique to LTL specifications on infinite traces, as well as to LTL$_f$ specifications on finite traces [De Giacomo and Vardi, 2013; De Giacomo and Vardi, 2015]. The latter is particularly well suited for implementation, since it is based on manipulation of finite automata, for which a key step, determinization, while exponential in the worst case is often polynomial in practice [Tabakov and Vardi, 2005; Zhu *et al.*, 2017; Zhu *et al.*, 2020; Tabajara and Vardi, 2020].

Interestingly, best-effort synthesis under environment assumptions separates the traditional trace-based approach to synthesis [Pnueli and Rosner, 1989; Bloem *et al.*, 2014], and the strategy-based approach [Mogavero *et al.*, 2014; Kupferman *et al.*, 2001; Aminof *et al.*, 2019; Aminof *et al.*, 2020]. We discuss this point further in Section 6.

2 Synthesis under Environment Assumptions

Linear-time Temporal Logic (LTL)

Given a finite set AP of atomic propositions, *formulas of* LTL *over* AP are defined by the following BNF (where $p \in AP$):

$$\varphi ::= p \,|\, \varphi \lor \varphi \,|\, \neg\varphi \,|\, \mathsf{X}\varphi \,|\, \varphi \,\mathsf{U}\, \varphi$$

We use the usual abbreviations, $\varphi \supset \varphi' \doteq \neg\varphi \lor \varphi'$, $\mathsf{true} \doteq p \lor \neg p$, $\mathsf{F}\,\varphi \doteq \mathsf{true}\,\mathsf{U}\,\varphi$, $\mathsf{G}\,\varphi \doteq \neg\,\mathsf{F}\,\neg\varphi$. The *size* $|\varphi|$ of a formula φ is the number of symbols in it. A *trace* τ is an infinite sequence of valuations of the atoms, i.e., $\tau \in (2^{AP})^\omega$. For $n \geq 0$, write τ_n for the valuation at position n. A *history* h is a finite prefix of a trace. Given a trace τ, an integer n, and an LTL formula φ, the satisfaction relation $(\tau, n) \models \varphi$, stating that φ holds at step n of the sequence τ, is defined as follows: $(\tau, n) \models p$ iff $p \in \tau_n$; $(\tau, n) \models \varphi_1 \vee \varphi_2$ iff $(\tau, n) \models \varphi_1$ or $(\tau, n) \models \varphi_2$; $(\tau, n) \models \neg\varphi$ iff $(\tau, n) \models \varphi$ does not hold; $(\tau, n) \models \mathsf{X}\varphi$ iff $(\tau, n+1) \models \varphi$; and $(\tau, n) \models \varphi_1 \mathsf{U} \varphi_2$ iff there exists $m \geq n$ such that $(\tau, m) \models \varphi_2$ and $(\tau, j) \models \varphi_1$ for all $n \leq j < m$. We write $\tau \models \varphi$ if $(\tau, 0) \models \varphi$ and say that τ *satisfies* φ and that τ is a *model* of φ.

Reactive Synthesis

(Reactive) synthesis concerns constructing the behaviors of an agent that achieve a given goal (aka task) while interacting with its environment. We specify goals as well as environment assumptions by LTL formulas. Formally, let X and Y be disjoint finite sets of Boolean variables, called the *environment variables* and *agent variables* respectively, and let $AP \doteq X \cup Y$. A trace is then written as $(X_0 \cup Y_0)(X_1 \cup Y_1)\ldots$ where $X_i \subseteq X, Y_i \subseteq Y$ for all i.

An *agent strategy* is a function $\sigma_{\mathsf{ag}} : (2^X)^+ \to 2^Y$ that maps a non-empty *environment history* to the next valuation of the agent variables. Similarly, an *environment strategy* is a function $\sigma_{\mathsf{env}} : (2^Y)^* \to 2^X$ mapping *agent histories* to the next valuation of the environment variables. Note that the domain of σ_{env} includes the empty sequence λ since the environment makes the first move. A trace $(X_i \cup Y_i)_i$ is σ_{ag}-*consistent* if $\sigma_{\mathsf{ag}}(X_0 X_1 \cdots X_j) = Y_j$ for every $j \geq 0$; and is σ_{env}-*consistent* if $\sigma_{\mathsf{env}}(\lambda) = X_0$ and $\sigma_{\mathsf{env}}(Y_0 Y_1 \cdots Y_j) = X_{j+1}$ for every $j \geq 0$. We write $\text{PLAY}(\sigma_{\mathsf{ag}}, \sigma_{\mathsf{env}})$ for the unique trace consistent with both strategies.

Let φ be an LTL formula over $X \cup Y$. An agent strategy σ_{ag} *enforces* φ, written $\sigma_{\mathsf{ag}} \rhd \varphi$, if $\text{PLAY}(\sigma_{\mathsf{ag}}, \sigma_{\mathsf{env}}) \models \varphi$ for every environment strategy σ_{env}. In this case we say that φ is *agent enforceable*. *Environment enforceability* is defined similarly: $\sigma_{\mathsf{env}} \rhd \varphi$ if $\forall \sigma_{\mathsf{ag}}.\, \text{PLAY}(\sigma_{\mathsf{ag}}, \sigma_{\mathsf{env}}) \models \varphi$. The *synthesis problem* is, given an LTL formula φ, find (if there is one) an agent strategy σ that enforces φ.

Theorem 1. [Pnueli and Rosner, 1989] The LTL synthesis problem is 2EXPTIME-complete.

Reactive Synthesis under Environment Assumptions

In most AI scenarios the agent has some knowledge of how the environment works, which it can exploit in order to enforce the goal. We specify this knowledge by an LTL formula $\mathcal{E}$. Following [Aminof *et al.*, 2018; Aminof *et al.*, 2019], we view $\mathcal{E}$ as specifying the set of environment strategies that enforce it. Moreover, for it to be useful, we require this set to be nonempty, i.e., that $\mathcal{E}$ is environment-enforceable. In this case, we call $\mathcal{E}$ an *environment assumption*.

There is a weaker way to interpret LTL environment specifications (cf. [Bloem *et al.*, 2014]), i.e., $\mathcal{E}$ directly restricts the *traces* (instead of the strategies) of interest to the ones that *satisfy* $\mathcal{E}$. In that case, synthesis amounts to devising a strategy of the agent that enforces the implication $\mathcal{E} \supset \varphi$. While these two ways of interpreting environment specifications are distinct, their synthesis problems are inter-reducible [Aminof *et al.*, 2019]. Although we will exploit this fact in our algorithms, the trace-based approach falls short when dealing with the more general problem of best-effort synthesis under environment assumptions. We discuss this in Section 6.

Given an LTL formula φ and an LTL environment assumption $\mathcal{E}$, the *synthesis under environment assumptions* problem [Aminof *et al.*, 2019] is to find (if there is one) a strategy σ_{ag} such that:

$$\forall \sigma_{\mathsf{env}} \rhd \mathcal{E}.\, \text{PLAY}(\sigma_{\mathsf{ag}}, \sigma_{\mathsf{env}}) \models \varphi$$

Such a strategy is said to *enforce* φ *under* $\mathcal{E}$. Reactive synthesis is the special case taking $\mathcal{E} = \mathtt{true}$. Furthermore, it can be shown that most forms of planning with LTL goals, including on nondeterministic domains, are special cases of this synthesis problem [Aminof *et al.*, 2019].

Theorem 2. [Aminof *et al.*, 2019] Solving LTL synthesis under environment assumptions is 2EXPTIME-complete.

3 Best-effort Strategies

To formalize the notion of best-effort strategies, we first recall what it means for an agent strategy to be better than another. Let $\mathcal{E}$ be an environment assumption, and let φ be an agent goal. Define a binary relation $\geq_{\varphi|\mathcal{E}}$ on agent strategies:

Definition 1 (Dominance)**.** *$\sigma_1 \geq_{\varphi|\mathcal{E}} \sigma_2$ iff for every $\sigma_{\mathsf{env}} \rhd \mathcal{E}$, $\text{PLAY}(\sigma_2, \sigma_{\mathsf{env}}) \models \varphi$ implies $\text{PLAY}(\sigma_1, \sigma_{\mathsf{env}}) \models \varphi$.*

This notion can be defined also when there is no assumption involved, by simply taking $\mathcal{E} = \mathtt{true}$. As usual, write $\sigma_1 >_{\varphi|\mathcal{E}} \sigma_2$ iff $\sigma_1 \geq_{\varphi|\mathcal{E}} \sigma_2$ and $\sigma_2 \not\geq_{\varphi|\mathcal{E}} \sigma_1$. If $\sigma_1 >_{\varphi|\mathcal{E}} \sigma_2$ we say that σ_1 *strictly dominates* σ_2 *(for goal* φ *under environment assumption* $\mathcal{E}$*)*. The relation $\geq_{\varphi|\mathcal{E}}$ is a preorder, and $>_{\varphi|\mathcal{E}}$ is a strict partial order.

Intuitively, $\sigma_1 >_{\varphi|\mathcal{E}} \sigma_2$ means that σ_1 does at least as well as σ_2 against every environment strategy enforcing $\mathcal{E}$, and strictly better against at least one such strategy. In particular, if σ_2 is not maximal, say σ_1 strictly dominates it, then an agent that uses σ_2 is not doing its "best" to achieve the goal: if it used σ_1 instead, it could achieve the goal against a strictly larger set of environment strategies. Within this framework, maximal strategies do a "best-effort" to achieve the goal.

Remark 1. *Enforcing strategies dominate all others. In particular, φ is agent-enforceable under $\mathcal{E}$ iff every maximal strategy wrt $\geq_{\varphi|\mathcal{E}}$ enforces φ under $\mathcal{E}$.*

Definition 2 (Best-effort)**.** *An agent strategy σ_2 is* maximal, or best-effort, for the goal φ under the environment assumption $\mathcal{E}$*, iff there is no agent strategy σ_1 such that $\sigma_1 >_{\varphi|\mathcal{E}} \sigma_2$.*

When there is no environment assumption, i.e., $\mathcal{E} = \mathtt{true}$, we simply say that σ_2 *is best-effort for the goal* φ. The problem of best-effort synthesis under assumption is:

Definition 3. *Given an* LTL *goal φ and an* LTL *environment assumption $\mathcal{E}$, the* best-effort synthesis problem under environment assumptions *is to find an agent strategy that is best-effort for φ under $\mathcal{E}$.*

Example 1. Suppose $X = \{r\}, Y = \{g\}$, and $\varphi = \mathsf{G}(r \supset \mathsf{F}(g \wedge \neg \mathsf{X} r))$ and $\mathcal{E} = (\neg r \wedge \mathsf{X} r \wedge \mathsf{X}\mathsf{X} r) \supset \mathsf{X}\mathsf{X}\mathsf{X} r$. We think of the environment as issuing "requests" (r) and the agent as issuing "grants" (g). A best-effort strategy for φ under $\mathcal{E}$ has no obligation to do anything against a strategy that has produced the sequence $\neg r, r, r, \neg r$ (and thus violated $\mathcal{E}$). A strategy σ is best-effort iff, if such a sequence was not produced so far, then the strategy ensures that (i) every request is followed at some point in time by a grant, and (ii) there is no grant immediately following the sequence $\neg r, r, r$.

The realizability problem, i.e., deciding if such agent strategies exist, is trivial since they always exist:

Theorem 3. [Berwanger, 2007; Aminof *et al.*, 2020] Given an LTL goal φ and an LTL environment assumption $\mathcal{E}$, there is an agent strategy that is best-effort for the goal φ under $\mathcal{E}$.

The main problem addressed in this work is to establish the exact complexity and algorithms for *finding* such strategies.

We end with a remark regarding environment assumptions in the context of best-effort synthesis. Consider two agent strategies σ^1_{ag} and σ^2_{ag}, and two environment strategies σ^1_{env} and σ^2_{env}, such that $\text{PLAY}(\sigma^i_{\mathsf{ag}}, \sigma^j_{\mathsf{env}}) \models \varphi$ iff $i = j$. Observe that σ^1_{ag} and σ^2_{ag} are incomparable in the dominance order. Intuitively, lacking information regarding the relative importance of σ^1_{env} and σ^2_{env}, we have no way to rank σ^1_{ag} and σ^2_{ag}. Environment assumptions are one way of providing such information: in case only one of σ^1_{env} and σ^2_{env} enforces the assumption, the choice between σ^1_{ag} and σ^2_{ag} becomes clear.

4 Solving Games on Automata

In section 5 we give a 2EXPTIME algorithm for finding best-effort strategies under environment assumptions. The building blocks for this algorithm are games played on automata.

Deterministic Transition Systems

A *deterministic transition system* $D = (\Sigma, Q, \iota, \delta)$ consists of a finite *input alphabet* Σ (typically, $\Sigma = 2^{AP}$), a finite set Q of *states*, an *initial state* $\iota \in Q$, and a *transition function* $\delta : Q \times \Sigma \to Q$. The *size* of D is the number of its states.

Let $\alpha = \alpha_0\alpha_1\cdots$ denote a finite or infinite sequence over the alphabet Σ. The *run* (aka *path*) induced by α is the sequence $q_0q_1\cdots$ of states where $q_0 = \iota$ and $q_{i+1} = \delta(q_i, \alpha_i)$ for every $i < |\alpha|$. We extend δ to the function $Q \times \Sigma^* \to Q$ as follows: $\delta(q, \lambda) = q$, and for $n > 0$, if $q_n = \delta(q, \alpha_0\cdots\alpha_{n-1})$ then $\delta(q, \alpha_0\alpha_1\cdots\alpha_n) = \delta(q_n, \alpha_n)$.

Definition 4. *The* product *of two transition systems $D_i = (\Sigma, Q_i, \iota_i, \delta_i)$ (for $i = 1, 2$) over the same input alphabet is the transition system $D_1 \times D_2 = (\Sigma, Q, \iota, \delta)$ where $Q = Q_1 \times Q_2$; $\iota = (\iota_1, \iota_2)$; and $\delta((q_1, q_2), x) = (\delta(q_1, x), \delta(q_2, x))$. The product $D_1 \times D_2 \times \cdots \times D_n$ can be similarly defined for any finite sequence $D_1, D_2, \cdots, D_n$ of transition systems over the same input alphabet Σ.*

Parity Automata

Automata can be seen as transition-systems with an acceptance condition. To handle LTL, we use the parity acceptance condition. A *Deterministic Parity Word automaton (DPA)* $A = (D, c)$ consists of a deterministic transition system D and a *coloring function* $c : Q \to \mathbb{Z}$. The *index* of A is the number of integers in the image of c, i.e., $|\{n \in \mathbb{Z} \mid c^{-1}(n) \neq \emptyset\}|$. An infinite run $\rho = q_0q_1\cdots$ *satisfies* c (alternatively, *accepted*) iff the smallest n such that $c(q_i) = n$ for infinitely many i is even. An infinite string $\alpha \in \Sigma^\omega$ is *accepted by* $A = (D, c)$ iff its run satisfies c. The set of infinite strings accepted by A is the *language* of A.

Theorem 4 (LTL to DPA)**.** (cf. [Vardi, 1995]) Given an LTL formula φ, one can build a DPA A_φ accepting exactly the models of φ, whose size is at most 2EXP in $|\varphi|$ and whose index is at most EXP in $|\varphi|$.

If $A_i = (D_i, c_i)$ are DPAs, and D is the product transition-system, we can lift the winning conditions c_i to the product by ignoring the other components:

Definition 5. *For DPAs $A_i = (D_i, c_i)$ and writing D for the product of their transition systems $D_1, \cdots, D_k$, define the* lifting *of c_i to D to be the coloring function $d_i : Q \to \mathbb{Z}$ defined by $d_i(q_1, \cdots, q_k) \doteq c_i(q_i)$.*

Note that (D, d_i) is a DPA, and its language is the same as the language of the DPA (D_i, c_i).

Games on Deterministic Automata

It is known that solving synthesis (without environment assumptions) for an LTL goal φ can be reduced to solving a two-player game of perfect information on the DPA corresponding to φ. We describe this process below since we will later generalize it. This section is an adaptation of material on parity games [Apt and Grädel, 2011] to our setting.

Informally, the current position in the game is a state q of the DPA, first the environment moves by setting $X' \subseteq X$, then the agent follows by setting $Y' \subseteq Y$, and the position in the game is updated to the state $\delta(q, (X' \cup Y'))$. This interaction generates an infinite run, and the agent is declared the winner if the run satisfies the parity condition. Formally, a *DPA-game* is played on a DPA $A = (D, c)$ in which $\Sigma = 2^{X \cup Y}$, by the two players, *agent* and *environment*. The transition system $D = (\Sigma, Q, \iota, \delta)$ is called the *arena* of the game, and c is called the *objective* of the game. An agent strategy $\sigma_{\mathsf{ag}} : (2^X)^+ \to 2^Y$ is *winning* if for every strategy σ_{env} of the environment, the trace $\text{PLAY}(\sigma_{\mathsf{ag}}, \sigma_{\mathsf{env}})$ is accepted by the DPA A (i.e., the run on the trace satisfies c). Similarly, a strategy $\sigma_{\mathsf{env}} : (2^Y)^* \to 2^X$ for the environment is *winning* if for every strategy σ_{ag} of the agent, the trace $\text{PLAY}(\sigma_{\mathsf{ag}}, \sigma_{\mathsf{env}})$ is *not* accepted by the DPA A (thus, the objective c of the game is from the agent's point of view). A player's *winning region* in the DPA-game on $\mathcal{A}$ is the set of states q for which that player is winning in the DPA-game on $(\Sigma, Q, q, \delta, c)$, i.e., starting from q. A strategy for a player that is winning from every state in her winning region is called *uniform winning*.

If the agent makes its moves just by looking at the current environment move and the current state of the DPA (instead of the exact full history of environment moves), then we say that the agent is using a *positional strategy*. Formally, a function $f_{\mathsf{ag}} : Q \times 2^X \to 2^Y$ induces the *positional* agent strategy σ_{ag}: $\sigma_{\mathsf{ag}}(X_0) = f_{\mathsf{ag}}(\iota, X_0)$, and for $n > 0$, $\sigma_{\mathsf{ag}}(X_0X_1\cdots X_n) = f_{\mathsf{ag}}(q_n, X_n)$ where $q_n \doteq \delta(\iota, \alpha)$ and α is the finite sequence played up until now, i.e., $\alpha = (X_0 \cup \sigma_{\mathsf{ag}}(X_0))(X_1 \cup \sigma_{\mathsf{ag}}(X_0X_1))\cdots(X_{n-1} \cup \sigma_{\mathsf{ag}}(X_0X_1\cdots X_{n-1}))$. We can similarly define environment positional strategies as functions $f_{\mathsf{env}} : Q \to 2^X$.

The relevant computational problem associated with a DPA-game is to compute, for a given player, that player's winning region W, as well as a uniform winning positional strategy for that player. We call this *solving* the game.

Theorem 5. DPA-games can be solved in time polynomial in the size of A and exponential in the index of A.

We also consider the case where the agent and environment co-operate. Formally, a pair of strategies $\sigma_{\mathsf{ag}}, \sigma_{\mathsf{env}}$ is *co-*

operatively winning if the trace $\text{PLAY}(\sigma_{\mathsf{ag}}, \sigma_{\mathsf{env}})$ is accepted by the DPA A. The *co-operative winning region* W' of a DPA-game A is the set of states q for which there is a pair of strategies that are co-operatively winning in the game played on $(\Sigma, Q, q, \delta, c)$, i.e., starting from q. Solving a co-operative DPA-game on $\mathcal{A}$ means to find the set W', as well as a pair of uniform positional strategies (although we will only use the agent's) that co-operatively win from every state in W'. Note that this amounts to solving the emptiness problem for $\mathcal{A}$.

Theorem 6. Co-operative DPA-games can be solved in time polynomial in both the size and index of the given DPA A.

We will sometimes need a way to limit the environment to moves that do not leave a given set of states Q'. We do this by introducing a $sink$-state, and redirecting to it moves of the environment that start in Q' and for which some response of the agent leaves Q':

Definition 6. *Let $D = (\Sigma, Q, \iota, \delta)$ be an arena and let $Q' \subseteq Q$ be a set of states. The* environment-restriction of D to Q' *is the arena $(\Sigma, Q \cup \{sink\}, \iota, \delta')$, where δ' agrees with δ except that $\delta'(q, (X' \cup Y')) = sink$ in case $q = sink$, or $q \in Q'$ and $\delta(q, (X' \cup Z)) \notin Q'$ for some $Z \subseteq Y$.*

5 Computing Best-Effort Strategies

In this section we establish the main result of the paper:

Theorem 7. Best-effort synthesis under environment assumptions (for LTL goals and assumptions) is 2EXPTIME-complete.

The main implication of this theorem is that it provides the following alternative approach to synthesis, with or without assumptions: instead of trying to solve reactive synthesis — which in case the formula φ is not agent-enforceable returns no strategy (i.e., gives up) — one can for the same cost search for a best-effort strategy which always exists and, in case φ is agent-enforceable, enforces φ.

The lower-bound is achieved by a reduction from reactive synthesis (without environment assumptions), which is 2EXPTIME-complete [Pnueli and Rosner, 1990]. In the remainder of this section we provide an algorithm for the upper-bound. The idea is to reduce the problem to computing suitable winning and co-operatively winning strategies of certain DPA-games. Note that the case without environment assumption was solved in [Faella, 2009]. One can recover this case from our algorithm simply by setting $\mathcal{E} = \mathsf{true}$.

Algorithm A. Given LTL formulas φ and $\mathcal{E}$:

1. For every $\xi \in \{\neg\mathcal{E}, \mathcal{E} \supset \varphi, \mathcal{E} \wedge \varphi\}$ compute the DPAs $A_\xi = (D_\xi, c_\xi)$ (Theorem 4).
2. Form the product $D = D_{\neg\mathcal{E}} \times D_{\mathcal{E}\supset\varphi} \times D_{\mathcal{E}\wedge\varphi}$, and denote the lifted priority functions on D by $d_{\neg\mathcal{E}}, d_{\mathcal{E}\supset\varphi}$ and $d_{\mathcal{E}\wedge\varphi}$ (Definitions 4 and 5).
3. In the DPA-game on $(D, d_{\mathcal{E}\supset\varphi})$, compute a uniform positional winning agent strategy f_{ag} for $d_{\mathcal{E}\supset\varphi}$ (Theorem 5). Let $W \subseteq Q$ be the agent's winning region.
4. In the DPA-game on $(D, d_{\neg\mathcal{E}})$, compute the environment's winning region $V \subseteq Q$ (Theorem 5). Thus, the environment can ensure $\mathcal{E}$ holds from every state in V.
5. Compute the environment-restriction of D to the set V, and call the resulting transition system D' (Definition 6). Form the DPA-game $(D', d_{\mathcal{E}\wedge\varphi})$ where the parity of the new sink state is given an odd color smaller than all even colors in D.
6. In the DPA-game $(D', d_{\mathcal{E}\wedge\varphi})$, find a uniform positional strategy g_{ag} that is co-operatively winning (Theorem 6).
7. **Return** the agent strategy induced by the positional strategy $k_{\mathsf{ag}} : Q \times 2^X \to 2^Y$: $k_{\mathsf{ag}}(q, X') = f_{\mathsf{ag}}(q, X')$ if $q \in W$, and $k_{\mathsf{ag}}(q, X') = g_{\mathsf{ag}}(q, X')$ otherwise.

Correctness of Algorithm A

The correctness of Algorithm A uses the notion of a value $val(h)$ of a history h: the value is "winning" if the player can enforce the goal under the environment assumption starting from h; "losing" if all plays extending h, that are consistent with an environment strategy that enforces the assumption, violate the goal; and "pending" otherwise. This allows one to characterize best-effort strategies as those that witness the value $val(h)$ of every history h consistent with it [Berwanger, 2007; Aminof *et al.*, 2020]. Here is a formalization of the main considerations. For a joint history $h \in (2^{X\cup Y})^*$, let $E(h, \sigma_{\mathsf{ag}})$ be the set of all environment strategies σ_{env} that enforce $\mathcal{E}$ and for which h is consistent with $\sigma_{\mathsf{ag}}, \sigma_{\mathsf{env}}$; and let $H(\sigma_{\mathsf{ag}})$ be the set of all h for which $E(h, \sigma_{\mathsf{ag}}) \neq \emptyset$. Define:

- $val(\sigma_{\mathsf{ag}}, h) := +1$ (winning) if $\text{PLAY}(\sigma_{\mathsf{ag}}, \sigma_{\mathsf{env}}) \models \varphi$ for every $\sigma_{\mathsf{env}} \in E(h, \sigma_{\mathsf{ag}})$;
- $val(\sigma_{\mathsf{ag}}, h) := -1$ (losing) if $\text{PLAY}(\sigma_{\mathsf{ag}}, \sigma_{\mathsf{env}}) \models \neg\varphi$ for every $\sigma_{\mathsf{env}} \in E(h, \sigma_{\mathsf{ag}})$;
- $val(\sigma_{\mathsf{ag}}, h) := 0$ (pending) otherwise.

Let $val(h)$ be the maximum of $val(\sigma, h)$ where σ varies over all agent-strategies for which $h \in H(\sigma)$.[1] The next theorem follows from the cited characterization of dominance:

Theorem 8. An agent strategy σ is best-effort iff $val(\sigma, h) = val(h)$ for every history $h \in H(\sigma)$.

Example 2. To see that σ in Example 1 is best-effort use Theorem 8. To that end, let $h \in H(\sigma)$. First note that $val(h) < 1$ since the environment strategy that always requests, except where not specified in h, enforces $\mathcal{E}$ and $\neg\varphi$. Second, note that φ can be violated by a history that contains r and then (possibly at the same time) a g immediately followed by r. If h is such a history, then $val(h) = -1$, and thus $val(\sigma, h) = val(h)$. So, suppose h is not such a history. Let σ_{env} be the environment strategy that never requests, except where specified in h and immediately after h. Note that σ_{env} enforces $\mathcal{E}$, and thus $val(h) > -1$ since an agent can safely respond to any outstanding requests. Since, in particular, σ safely responds to any outstanding request, also $val(\sigma, h) > -1$, and thus also in this case $val(\sigma, h) = val(h) = 0$.

The correctness of Algorithm A follows immediately from:

Theorem 9. Let σ_A be the agent-strategy produced by Algorithm A, and let $h \in H(\sigma)$. Then $val(\sigma_A, h) = val(h)$.

Intuitively, the strategy σ_A returned by Algorithm A has the property that histories that lead to states in W are winning

[1] We use $val(h)$ only in cases where $h \in H(\sigma)$ for some σ.

(from these states the agent can enforce $\mathcal{E} \supset \varphi$, and thus in particular can enforce φ assuming $\mathcal{E}$), histories that lead to states in $W' \setminus W$, where W' is the co-operatively winning region from step 6 in the game $(D', d_{\mathcal{E}\wedge\varphi})$, are pending (from these states there is an environment strategy σ_{env} enforcing $\mathcal{E}$ such that $\text{PLAY}(\sigma_A, \sigma_{\mathsf{env}}) \models \varphi$), and the rest are losing.

6 Strategy-based vs. Trace-based Approaches

While this paper adopts a strategy-based approach to synthesis [Mogavero *et al.*, 2014; Kupferman *et al.*, 2001; Aminof *et al.*, 2019; Aminof *et al.*, 2020], traditional synthesis follows a trace-based approach [Pnueli and Rosner, 1989; Bloem *et al.*, 2014; Camacho *et al.*, 2018]: synthesis for φ using the environment assumption $\mathcal{E}$ is solved by finding an agent strategy that enforces $\mathcal{E} \supset \varphi$ against *all* environment strategies, i.e., the assumption $\mathcal{E}$ is pushed into the goal formula instead of being used to limit the set of environment strategies. Unfortunately, while this works when synthesizing enforcing strategies, the example below clearly shows that it is inadequate when looking for best-effort strategies.

Consider a Star-Wars setting where the droid C-3PO is captured and made to work at a space port, loading boxes onto a cargo vessel. At every time-step, a few new boxes, of varying weights, may be added to the pending pile. Being a protocol droid, C-3PO is not very strong, and can load at most 30 kilograms in one time step. With the help of R2-D2, C-3PO learns that the space-port works under the following strict protocol: if at the time the cargo vessel has to leave, the loading is not finished, he will be declared inefficient and dismantled for parts in the next time step. He is otherwise safe except that, if he ever stops loading when there are pending boxes, he will be declared lazy and dismantled in the next time step. C-3PO thus forms the assumption $\mathcal{E}$ on the environment by taking the rules above conjuncted with the fact that if he sees Luke Skywalker then he will be rescued after two time steps.

For obvious reasons, C-3PO has the following goal φ: $\neg\text{dismantled} \mathsf{U} \text{rescued}$. Unfortunately, the vessel's departure time, the new boxes that may be added at each time step, and Luke Skywalker's timing, are not under his control, making it impossible for him to have a winning strategy.

It is not hard to see that a best-effort strategy for φ under the assumption $\mathcal{E}$ will result in C-3PO continuously loading boxes unless there are no boxes to load, or the cargo vessel has to leave, or he previously saw Luke Skywalker. Indeed, any other strategy cannot be maximal since it is guaranteed, by the assumption $\mathcal{E}$, to get C-3PO dismantled.

On the other hand, when considering the implication $\mathcal{E} \supset \varphi$, any strategy σ_{ag} that loads nothing on the first step is *not dominated* by any strategy σ'_{ag} that does! To see that, consider an environment strategy σ_{env} that operates as follows: it starts with 40 kilograms of boxes in the pile; if C-3PO loads nothing in the first time step then do nothing in the next step; otherwise, declare that it is time for the vessel to leave, and dismantle C-3PO in the next step. Hence, against σ_{env}, the strategy σ_{ag} achieves the implication, whereas σ'_{ag} does not. Obviously, σ_{env} is irrelevant since it blatantly violates $\mathcal{E}$. However, when synthesizing for $\mathcal{E} \supset \varphi$ (unlike synthesizing for φ under $\mathcal{E}$) it is still considered. Hence, by taking a maximal element from the set of strategies σ_{ag} that load nothing on the first step, we get a best-effort strategy for $\mathcal{E} \supset \varphi$ that is a guaranteed suicide: it is an unconditionally losing strategy for the goal under $\mathcal{E}$, and it never achieves the goal on any trace that satisfies $\mathcal{E}$!

7 Best-Effort Synthesis in LTL$_f$

Although many environment assumptions are naturally expressed in LTL (e.g., fairness assumptions), other classic ones such as planning domains and safety can be expressed in LTL$_f$. In this section we show that the framework for LTL also applies to LTL$_f$. The main difference is that in the this case we can give an algorithm for finding best-effort strategies that only uses classic simple graph-algorithm building blocks, such as breadth-first search, fixed-point algorithms, and subset constructions for determinization. In other words, the algorithm in this section lends itself to implementation.

Linear-time Temporal Logic on Finite Traces (LTL$_f$)

LTL$_f$ has the same syntax as LTL but is interpreted on finite traces $\tau \in (2^{AP})^+$ [De Giacomo and Vardi, 2013] [2]. For the semantics, only the temporal operators are reinterpreted. This is done as follows. For $n < |\tau|$:

- $(\tau, n) \models \mathsf{X}\,\varphi$ iff $n+1 < len(\tau)$ and $(\tau, n+1) \models \varphi$;
- $(\tau, n) \models \varphi_1 \mathsf{U} \varphi_2$ iff there exists i with $n \leq i < len(\tau)$ such that $(\tau, i) \models \varphi_2$ and for all $i \leq j < n$, $(\tau, j) \models \varphi_1$.

Let $\tilde{\mathsf{X}}$ be the dual of X, i.e., $\tilde{\mathsf{X}} \doteq \neg\mathsf{X}\neg\varphi$. Semantically, $(\tau, n) \models \tilde{\mathsf{X}}\varphi$ iff $n+1 < len(\tau)$ implies $(\tau, n+1) \models \varphi$.

Thus, the main difference is that while in LTL the X operator is its own dual (i.e., $\neg\mathsf{X}\neg\varphi \equiv \mathsf{X}\,\varphi$), this is not true in LTL$_f$, where the dual of X has a different meaning to X.

An important computational advantage of LTL$_f$ is that it can be translated into ordinary finite-state automata. A *Deterministic Finite Automaton (DFA)* $A = (D, F)$ consists of a deterministic transition system D and a set $F \subseteq Q$ of *final states*. A finite run $\rho = q_0 q_1 \cdots q_n$ is *accepting* iff $q_n \in F$. A finite string $\alpha \in \Sigma^*$ is *accepted* if its run is accepting.

LTL$_f$ Synthesis under Assumptions

Over finite traces there is the issue of how a play stops. We redefine agent strategies to be *partial* functions $\sigma_{\mathsf{ag}} : (2^X)^+ \hookrightarrow (2^Y)$ and require environment strategies be total as before $\sigma_{\mathsf{env}} : (2^Y)^* \to (2^X)$. Then, we redefine $\text{PLAY}(\sigma_{\mathsf{ag}}, \sigma_{\mathsf{env}})$ to be the longest (finite or infinite) trace that is consistent with both strategies. Note that a play is finite if it ends in an environment move for which the agent strategy, being a partial function, specifies no next move (which we think of as the agent stopping the play).

Let $\varphi, \mathcal{E}$ be LTL$_f$ formulas over $X \cup Y$. We say that an agent strategy σ_{ag} *enforces* φ, written $\sigma_{\mathsf{ag}} \rhd \varphi$, iff

$$\forall \sigma_{\mathsf{env}}.\, (\text{PLAY}(\sigma_{\mathsf{ag}}, \sigma_{\mathsf{env}}) \text{ is finite and } \text{PLAY}(\sigma_{\mathsf{ag}}, \sigma_{\mathsf{env}}) \models \varphi)\,.$$

Asymmetrically, we say that an environment strategy σ_{env} *enforces* $\mathcal{E}$, written $\sigma_{\mathsf{env}} \rhd \mathcal{E}$, iff

$$\forall \sigma_{\mathsf{ag}}.\, (\text{PLAY}(\sigma_{\mathsf{ag}}, \sigma_{\mathsf{env}}) \text{ is finite implies } \text{PLAY}(\sigma_{\mathsf{ag}}, \sigma_{\mathsf{env}}) \models \mathcal{E})\,.$$

[2] In fact all results presented here apply also to the extension LDL$_f$ of LTL$_f$, which captures monadic-second-order logic on finite traces [De Giacomo and Vardi, 2013].

The asymmetry stems from the fact that, intuitively, an agent cannot enforce φ unless she stops the play.

Remaining definitions. The rest of the LTL_f treatment mimics that of LTL simply by replacing LTL by LTL_f in Definitions 1, 2, and 3. We also replace DPAs by ordinary finite-state automata. Then, Theorems 1 and 4 hold for LTL_f and DFA. In particular, one can obtain a DFA, of 2EXP size, accepting exactly the models of a given LTL_f formula [De Giacomo and Vardi, 2013]. Games can be played on DFAs as for DPAs except that the agent is trying to end the play in a final state, i.e., an agent strategy σ_{ag} is winning if for every environment strategy σ_{env}, the trace $\text{PLAY}(\sigma_{\mathsf{ag}}, \sigma_{\mathsf{env}})$ is finite and is accepted by the DFA; dually, an environment strategy σ_{env} is winning if for every agent strategy σ_{ag}, if $\text{PLAY}(\sigma_{\mathsf{ag}}, \sigma_{\mathsf{env}})$ is finite then it is not accepted by the DFA.

Best-Effort LTL_f Synthesis

The algorithm below finds best-effort strategies under environment assumptions in the LTL_f case (we contextually discuss each step). The main difference to Algorithm A is that we replace DPAs by DFAs.

Algorithm B. Given LTL_f formulas φ and $\mathcal{E}$:

1. For every $\xi \in \{\neg\mathcal{E}, \mathcal{E} \supset \varphi, \mathcal{E} \wedge \varphi\}$ compute the DFAs $A_\xi = (D_\xi, F_\xi)$ [De Giacomo and Vardi, 2013].
2. Form (in linear time) the product $D = D_{\neg\mathcal{E}} \times D_{\mathcal{E}\supset\varphi} \times D_{\mathcal{E}\wedge\varphi}$. Lift the final states of each component to the product, e.g., if $A_\mathcal{E} = (D_\mathcal{E}, F_\mathcal{E})$ is the DFA for $\mathcal{E}$, then the lifted condition $G_\mathcal{E}$ consists of all states $(q_{\neg\mathcal{E}}, q_{\mathcal{E}\supset\varphi}, q_{\mathcal{E}\wedge\varphi}) \in Q$ such that $q_\mathcal{E} \in F_\mathcal{E}$.
3. In the DFA-game $(D, F_{\mathcal{E}\supset\varphi})$, compute (in linear time) a uniform positional winning agent strategy f_{ag}. Let $W \subseteq Q$ be the agent's winning region. Finding winning regions and computing uniform positional winning strategies in DFA-games can be done in linear-time by a simple fixed-point algorithm [Apt and Grädel, 2011].
4. In the DFA-game $(D, F_{\neg\mathcal{E}})$, compute (in linear time) the environment's winning region $V \subseteq Q$.
5. Compute (in linear-time) the environment-restriction D' of D to the set V.
6. In the DFA-game $(D', F_{\mathcal{E}\wedge\varphi})$, find (in linear time, e.g. using breadth-first search) a co-operatively winning uniform positional strategy g_{ag}.
7. **Return** the agent strategy induced by the positional strategy $k_{\mathsf{ag}} : Q \times 2^X \to 2^Y$: $k_{\mathsf{ag}}(q, X') = f_{\mathsf{ag}}(q, X')$ if $q \in W$, and $k_{\mathsf{ag}}(q, X') = g_{\mathsf{ag}}(q, X')$ otherwise.

Algorithm B provides a 2EXPTIME upper-bound; the lower-bound is derived in a similar way to the LTL case by a reduction from LTL_f-synthesis which is 2EXPTIME-complete [De Giacomo and Vardi, 2015].

Theorem 10. Best-effort synthesis under environment assumptions (for LTL_f goals and assumptions) is 2EXPTIME-complete.

Thus, Algorithm B is optimal, and although it has the same worst-case complexity as Algorithm A, it makes use of much simpler graph-algorithm building-blocks.

8 Related Work

Best-effort strategies are maximal in the dominance order. These notions originate in game-theory, where dominance is also called weak-dominance [Apt and Grädel, 2011], and best-effort is sometimes called admissible.

In formal methods, best-effort synthesis, and synthesis under environment assumptions, were mainly considered in isolation, and almost always in the trace-based setting, e.g., [Pnueli and Rosner, 1989; Faella, 2009; Brenguier *et al.*, 2017]; see [Bloem *et al.*, 2014] for a survey. [Faella, 2009] argues for best-effort strategies as reasonable responses in environments that may not be adversarial, characterizes goals admitting positional best-effort strategies, and shows how to compute such strategies (for LTL goals, this results in a special case of Algorithm A). Few papers, either directly or indirectly, consider best-effort and assumptions at the same time (e.g., [Damm and Finkbeiner, 2014] does so in a trace-based setting). The problems studied there are different: they are mainly concerned with looking for a strategy that dominates *all* other strategies (i.e., maximum rather than maximal), and in case it exists, they synthesize the weakest-environment assumption under which it achieves the goal.

More closely related are works that consider the multi-agent setting where agent i has as an assumption the goal of agent j, and thus naturally take the strategy-based view. Notably, [Berwanger, 2007] pioneers the study of dominance in games played on graphs. [Brenguier *et al.*, 2014] studies the problem of deciding the existence of strategy profiles and corresponding model-checking problems. The problems studied here can be expressed in their setting, however, in our case the existence problem is trivial, and we deal with synthesis not with model-checking. [Brenguier *et al.*, 2017] studies assume-admissible (AA) synthesis which is similar but different to ours; cast in our setting, AA looks for strategies that enforce φ against environment strategies that are best-effort (without assumptions) for $\mathcal{E}$, whereas we are looking for best-effort agent strategies for φ against environments that enforce $\mathcal{E}$. In particular, if $\mathcal{E}$ is environment-enforceable, AA is equivalent to synthesis (without assumptions).

The closest paper to ours is [Aminof *et al.*, 2020] where the problem of best-effort synthesis under environment assumptions is introduced in the context of two environments (expected and exceptional), and shown to be decidable. Strictly speaking, we do not provide an alternative solution to their problem, since we only deal with a single environment specification; and their paper does not include our problem as a special case due to extra requirements that they impose. However, one can extract from their decidability proof a 4EXPTIME upper bound to our problem. Finally, their technique is different from ours: theirs characterizes the set of all best-effort strategies using tree-automata, while ours computes a single best-effort strategy by solving certain games.

Acknowledgments

This work is partially supported by the Austrian Science Fund (FWF): P 32021, the ERC Advanced Grant WhiteMech (No. 834228), and the EU ICT-48 2020 project TAILOR (No. 952215)

References

[Aminof *et al.*, 2018] Benjamin Aminof, Giuseppe De Giacomo, Aniello Murano, and Sasha Rubin. Synthesis under assumptions. In *KR*, pages 615–616, 2018.

[Aminof *et al.*, 2019] B. Aminof, G. De Giacomo, A. Murano, and S. Rubin. Planning under LTL environment specifications. In *ICAPS*, 2019.

[Aminof *et al.*, 2020] B. Aminof, G. De Giacomo, A. Lomuscio, A. Murano, and S. Rubin. Synthesizing strategies under expected and exceptional environment behaviors. In *IJCAI*, 2020.

[Apt and Grädel, 2011] K.R Apt and E. Grädel. *Lectures in game theory for computer scientists*. Cambridge, 2011.

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

[Bertsekas, 2005] Dimitri P. Bertsekas. *Dynamic programming and optimal control*. Athena Scientific, 2005.

[Berwanger, 2007] D. Berwanger. Admissibility in infinite games. In *STACS*, 2007.

[Bloem *et al.*, 2014] Roderick Bloem, Rüdiger Ehlers, Swen Jacobs, and Robert Könighofer. How to handle assumptions in synthesis. In *SYNT*, 2014.

[Brenguier *et al.*, 2014] Romain Brenguier, Jean-François Raskin, and Mathieu Sassolas. The complexity of admissibility in omega-regular games. In Thomas A. Henzinger and Dale Miller, editors, *CSL-LICS*, 2014.

[Brenguier *et al.*, 2017] R. Brenguier, J. Raskin, and O. Sankur. Assume-admissible synthesis. *Acta Inf.*, 54(1), 2017.

[Camacho *et al.*, 2018] A. Camacho, M. Bienvenu, and S. McIlraith. Finite LTL synthesis with environment assumptions and quality measures. In *KR*, 2018.

[Church, 1963] A. Church. Logic, arithmetics, and automata. In *Proc. Int. Cong. Mathematicians, 1962*, 1963.

[Damm and Finkbeiner, 2014] Werner Damm and Bernd Finkbeiner. Automatic compositional synthesis of distributed systems. In *FM*, 2014.

[De Giacomo and Rubin, 2018] G. De Giacomo and S. Rubin. Automata-theoretic foundations of FOND planning for LTLf and LDLf goals. In *IJCAI*, 2018.

[De Giacomo and Vardi, 2013] Giuseppe De Giacomo and Moshe Y. Vardi. Linear temporal logic and linear dynamic logic on finite traces. In *IJCAI*, 2013.

[De Giacomo and Vardi, 2015] G. De Giacomo and M. Vardi. Synthesis for LTL and LDL on finite traces. In *IJCAI*, 2015.

[D'Ippolito *et al.*, 2018] N. D'Ippolito, N. Rodríguez, and S. Sardiña. Fully observable non-deterministic planning as assumption-based reactive synthesis. *J. Artif. Intell. Res.*, 61, 2018.

[Faella, 2009] M. Faella. Admissible strategies in infinite games over graphs. In *MFCS*, 2009.

[Finkbeiner, 2016] B. Finkbeiner. Synthesis of reactive systems. *Dependable Software Systems Eng.*, 45, 2016.

[Kupferman *et al.*, 2001] Orna Kupferman, Moshe Y Vardi, and Pierre Wolper. Module checking. *Inf. Comput.*, 164(2):322–344, 2001.

[Mogavero *et al.*, 2014] F. Mogavero, A. Murano, G. Perelli, and M.Y. Vardi. Reasoning about strategies: On the model-checking problem. *ACM Trans. Comput. Log.*, 15(4), 2014.

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

[Pnueli and Rosner, 1990] Amir Pnueli and Roni Rosner. Distributed reactive systems are hard to synthesize. In *FOCS*, 1990.

[Tabajara and Vardi, 2020] Lucas M. Tabajara and Moshe Y. Vardi. LTLf Synthesis under Partial Observability: From Theory to Practice. In *GandALF*, 2020.

[Tabakov and Vardi, 2005] Deian Tabakov and Moshe Y. Vardi. Experimental evaluation of classical automata constructions. In *LPAR*, 2005.

[Vardi, 1995] Moshe Y. Vardi. An automata-theoretic approach to linear temporal logic. In Faron Moller and Graham M. Birtwistle, editors, *Logics for Concurrency - Structure versus Automata*, volume 1043 of *LNCS*, 1995.

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

[Zhu *et al.*, 2020] Shufang Zhu, Giuseppe De Giacomo, Geguang Pu, and Moshe Y. Vardi. LTLf synthesis with fairness and stability assumptions. In *AAAI*, 2020.