Reactive Synthesis of Dominant Strategies

Benjamin Aminof, [1,3] **Giuseppe De Giacomo,** [2,3] **Sasha Rubin** [4]

[1] TU Wien
[2] University of Oxford
[3] Università degli Studi di Roma "La Sapienza"
[4] University of Sydney
aminof@forsyte.at, giuseppe.degiacomo@cs.ox.ac.uk, degiacomo@diag.uniroma1.it, sasha.rubin@sydney.edu.au

Abstract

We study the synthesis under environment specifications problem for LTL/LTL$_f$ which, in particular, generalizes FOND (strong) planning with these temporal goals. We consider the case where the agent cannot enforce its goal — for which the argument for using best-effort strategies has been made — and study the intermediate ground, between enforcing and best-effort strategies, of *dominant* strategies. Intuitively, such strategies achieve the goal against any environment for which it is achievable. We show that dominant strategies may exist when enforcing ones do not, while still sharing with the latter many desirable properties such as being interchangeable with each other, and being monotone with respect to tightening of environment specifications. We give necessary and sufficient conditions for the existence of dominant strategies, and show that deciding if they exist is 2EXPTIME-complete — the same as for enforcing strategies. Finally, we give a uniform, optimal, game-theoretic algorithm for simultaneously solving the three synthesis problems of enforcing, dominant, and best-effort strategies.

Introduction

In Reasoning about Actions, the agent has an explicit representation of the environment, corresponding to what it knows about the behavior of the environment it is operating in (McCarthy and Hayes 1969; Green 1969). Nondeterministic environments are often represented as nondeterministic domains, e.g., in Planning on Fully Observable Nondeterministic Domains (FOND) (Geffner and Bonet 2013).

Planning on nondeterministic domains can be seen as a prominent case of Reactive Synthesis (Church 1963) under environment specifications (Aminof et al. 2018, 2019), which is the problem of automatically constructing a reactive system from a logical specification. That is, environments can be specified as sets of environment strategies that enforce a given temporal logic formula and the computed reactive system is the sought after plan. In this paper, we consider specifications expressed in linear-time temporal logic over infinite traces (LTL) and finite traces (LTL$_f$), c.f. (Pnueli and Rosner 1989; De Giacomo and Vardi 2015).

The standard solution concept in Reactive Synthesis is that of an *enforcing* strategy (Pnueli and Rosner 1989) (cf. *strong* policy in FOND planning), i.e., an agent strategy that achieves the goal against all the environment strategies that conform to the environment specification. However, often an enforcing strategy does not exist, leading to the introduction of other solution concepts, including *strong-cyclic* policies, which exploit the fact that often the environment resolves nondeterminism stochastically (Daniele, Traverso, and Vardi 1999; Cimatti et al. 2003), and more recently *best-effort* strategies, i.e., those that achieve the goal against a *maximal* set of environment strategies that conform to the specification (Berwanger 2007; Faella 2009; Aminof et al. 2020, 2021; Aminof, De Giacomo, and Rubin 2021).

In this work we study *dominant* agent strategies, i.e., those that achieve the goal against the *maximum* set of environment strategies that conform to the specification. Clearly, enforcing strategies are dominant, and dominant strategies are best-effort. However, while best-effort strategies always exist, neither dominant nor enforcing strategies always exist. Intuitively, an enforcing strategy ensures getting the best *imaginable outcome*, a best-effort strategy ensures that one has made the best *possible decision* (given that the decision has to be made before the exact environment is known), and a dominant strategy ensures that one will always get the best *possible outcome*. Our contributions are as follows:

1. We show that dominant strategies, like enforcing ones, are (i) interchangeable with each other and (ii) monotone with respect to tightening of environment specifications.

2. We provide a local characterization of the dominant strategies in terms of what the strategy should achieve from any given history. The characterization is inspired by a similar characterization of the best-effort strategies (Berwanger 2007; Aminof et al. 2020).

3. We study the problem of finding a dominant strategy given goals and environments specified in LTL/LTL$_f$. We prove that the corresponding realizability problem is 2EXPTIME-complete, the same as that for enforcing strategies. Based on the local characterization, we give a uniform, optimal, game-theoretic algorithm for simultaneously solving the three synthesis problems of enforcing, dominant, and best-effort strategies. The algorithm generalizes the one for best-effort strategies (Aminof, De Giacomo, and Rubin 2021).

Preliminaries

For a sequence x, we write x_i for its ith element; the first element is x_0; the length of x is $|x| \in \mathbb{N} \cup \{\infty\}$ (with the convention that $\infty - 1 = \infty$); the prefix of x of length $0 \leq i \leq |x|$ is denoted by $x_{<i}$ or $x_{\leq i-1}$.

Linear-time Temporal Logic (LTL) For a set AP of atomic propositions, *formulas of* LTL *over* AP are defined by the following BNF (where $p \in AP$):

$$\varphi ::= p \mid \varphi \vee \varphi \mid \neg\varphi \mid \mathsf{X}\varphi \mid \varphi \,\mathsf{U}\, \varphi$$

We use the usual abbreviations, $\varphi \supset \varphi' \doteq \neg\varphi \vee \varphi'$, $\mathsf{true} \doteq p \vee \neg p$, $\mathsf{F}\,\varphi \doteq \mathsf{true}\,\mathsf{U}\,\varphi$, $\mathsf{G}\,\varphi \doteq \neg\,\mathsf{F}\,\neg\varphi$, etc. The *size* $|\varphi|$ of a formula φ is the number of symbols in it. A *trace* $\tau \in (2^{AP})^\omega$ is an infinite sequence of valuations of the atoms. For $n \geq 0$, write τ_n for the valuation at position n. 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 it is not the case that $(\tau, n) \models \varphi$;
- $(\tau, n) \models \mathsf{X}\,\varphi$ iff $n < |\tau| - 1$ and $(\tau, n+1) \models \varphi$;
- $(\tau, n) \models \varphi_1 \,\mathsf{U}\, \varphi_2$ iff $(\tau, m) \models \varphi_2$ for some $n \leq m < |\tau|$, and $(\tau, j) \models \varphi_1$ for all $n \leq j < m$.

Here X is read "next" and U is read "until". Write $\tau \models \varphi$ if $(\tau, 0) \models \varphi$, read τ *satisfies* φ. We also consider the variant LTL$_f$ (Bacchus and Kabanza 2000; Baier and McIlraith 2006; De Giacomo and Vardi 2013). It has the same syntax and semantics as LTL above except that τ is a finite sequence. Observe that the satisfaction of X and U on finite traces is defined "pessimistically", i.e., a trace cannot end before the promised eventuality holds. We use the following convention for interpreting LTL$_f$ formulas ψ over infinite traces τ: write $\tau \models \psi$ to mean that some finite prefix of τ satisfies ψ (analogous to the agent using an explicit "stop" action).

Reactive Synthesis The problem of (reactive) synthesis is to construct an agent that achieves a goal while continually interacting with its environment. We specify goals and environments with LTL formulas. Let $\mathbf{X}$ and $\mathbf{Y}$ be disjoint finite sets of Boolean variables, called the *environment variables* and *agent variables*, respectively. Then $2^{\mathbf{X}}$ (resp. $2^{\mathbf{Y}}$) is the set of environment (resp. agent) *moves*. A *play* $\pi \doteq X_0 \cdot Y_0 \cdot X_1 \cdot Y_1 \cdots$ is an element of $(2^{\mathbf{X}} \cdot 2^{\mathbf{Y}})^\omega$. The *trace induced by* π is the infinite sequence $(X_i \cup Y_i)_{i \geq 0}$, i.e., position i of the trace consists of both X_i and Y_i. A play π *satisfies* an LTL/LTL$_f$ formula φ over $AP = \mathbf{X} \cup \mathbf{Y}$, written $\pi \models \varphi$, if the trace induced by π satisfies φ. A *history* h is a finite prefix of a play. An *agent strategy* is a function $\sigma_{\mathsf{ag}} : (2^{\mathbf{X}} \cdot 2^{\mathbf{Y}})^* \cdot 2^{\mathbf{X}} \to 2^{\mathbf{Y}}$ that maps histories ending in environment moves to agent moves. Similarly, an *environment strategy* is a function $\sigma_{\mathsf{env}} : (2^{\mathbf{X}} \cdot 2^{\mathbf{Y}})^* \to 2^{\mathbf{X}}$ that maps histories ending in agent moves (including, since the environment moves first, the empty history λ) to environment moves. We let $\Sigma^{all}_{\mathsf{ag}}$ denote the set of all agent strategies.

A strategy σ is called *oblivious* if its output only depends on the length of the input history, i.e., if $|h| = |h'| \Rightarrow \sigma(h) = \sigma(h')$. Intuitively, an oblivious strategy is blind to the actions of the other side, and only sees the passage of time; it is thus equivalent to an infinite string of moves.

For an agent (resp. env) strategy σ and a play/history ρ, say that σ *and* ρ *are consistent*, if for every proper prefix $\rho_{<i}$ of odd (resp. even) length we have that $\rho_i = \sigma(\rho_{<i})$. Let $\textsc{play}(\sigma_{\mathsf{ag}}, \sigma_{\mathsf{env}})$ be the unique play consistent with both σ_{ag} and σ_{env}. An agent strategy σ_{ag} *enforces* ψ, written $\sigma_{\mathsf{ag}} \rhd \psi$, iff $\textsc{play}(\sigma_{\mathsf{ag}}, \sigma_{\mathsf{env}}) \models \psi$ for every environment strategy σ_{env}; if such a strategy exists we say the *agent can enforce* ψ. A symmetric definition holds for environment strategies.

Let $\mathcal{E}$ be an LTL/LTL$_f$ formula. We write $\Sigma^{\mathcal{E}}_{\mathsf{env}}$ for the set of environment strategies that enforce $\mathcal{E}$. If $\Sigma^{\mathcal{E}}_{\mathsf{env}} \neq \emptyset$ we say that $\mathcal{E}$ is an *environment specification* (aka *assumption*). The idea of requiring environment specifications to be environment-enforceable formulas is justified in (Aminof et al. 2018). Write $\Sigma^{\mathcal{E},h}_{\mathsf{env}}$ for those environment strategies that enforce $\mathcal{E}$ and are consistent with a history h. For the rest of this paper, we use $\mathcal{E}$ to denote an environment specification. The *synthesis under environment specifications problem*[1] asks, for an LTL *goal* φ and an LTL environment specification $\mathcal{E}$, to find (if there is one) an agent strategy σ_{ag} such that $\textsc{play}(\sigma_{\mathsf{ag}}, \sigma_{\mathsf{env}}) \models \varphi$ for every σ_{env} that enforces $\mathcal{E}$; and in this case we say that σ_{ag} *enforces* φ *under* $\mathcal{E}$. This problem is 2EXPTIME-complete (Aminof et al. 2019, 2018). The case $\mathcal{E} = \mathsf{true}$, called the *synthesis problem*, was pioneered in (Pnueli and Rosner 1989) and has the same complexity.

Defining Dominant Strategies

To formalize the notion of dominant strategies, we first define what it means for an agent strategy to *dominate* another. Let $\mathcal{E}$ be an environment specification, 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 $\sigma_{\mathsf{env}} \rhd \mathcal{E}$, $\textsc{play}(\sigma_2, \sigma_{\mathsf{env}}) \models \varphi$ implies $\textsc{play}(\sigma_1, \sigma_{\mathsf{env}}) \models \varphi$. In this case we say that* σ_1 dominates *σ_2 (for goal φ under environment specification $\mathcal{E}$).*

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 specification $\mathcal{E}$)*. 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. The relation $\geq_{\varphi|\mathcal{E}}$ is a preorder, and $>_{\varphi|\mathcal{E}}$ is a strict partial order. A maximum element (if it exists) in the preorder $\geq_{\varphi|\mathcal{E}}$ is called *dominant*:

Definition 2 (Dominant)**.** *An agent strategy σ_1 is* dominant*, aka* maximum*, for the goal φ under the environment specification $\mathcal{E}$, iff $\sigma_1 \geq_{\varphi|\mathcal{E}} \sigma_2$ for every agent strategy σ_2.*

If there is no environment specification, i.e., $\mathcal{E} = \mathsf{true}$, we say that σ_2 *is dominant for the goal* φ. A slightly weaker notion than being dominant is being best-effort (Berwanger 2007; Aminof, De Giacomo, and Rubin 2021):

Definition 3 (Best-effort)**.** *An agent strategy σ_1 is* best-effort*, aka* maximal*, for the goal φ under the env. specification $\mathcal{E}$, iff there is no agent strategy σ_2 s.t. $\sigma_2 >_{\varphi|\mathcal{E}} \sigma_1$.*

[1] In the literature this is sometimes called the *synthesis under assumptions* problem.

Given $\varphi, \mathcal{E}$, and an agent strategy σ, let $\Sigma^{\mathcal{E}}_{\mathsf{env}}(\varphi, \sigma) \subseteq \Sigma^{\mathcal{E}}_{\mathsf{env}}$ be the set of environment strategies $\sigma_{\mathsf{env}} \in \Sigma^{\mathcal{E}}_{\mathsf{env}}$ such that $\text{PLAY}(\sigma, \sigma_{\mathsf{env}}) \models \varphi$, i.e., the set of environment strategies σ_{env} that enforce $\mathcal{E}$ and against which σ achieves the goal. Using this notation we can characterize enforcing, dominant, and maximal strategies (for a goal φ under the environment specification $\mathcal{E}$) as follows: σ is enforcing iff $\Sigma^{\mathcal{E}}_{\mathsf{env}}(\varphi, \sigma) = \Sigma^{\mathcal{E}}_{\mathsf{env}}$; it is dominant iff $\Sigma^{\mathcal{E}}_{\mathsf{env}}(\varphi, \sigma') \subseteq \Sigma^{\mathcal{E}}_{\mathsf{env}}(\varphi, \sigma)$ for every σ'; and it is maximal iff $\Sigma^{\mathcal{E}}_{\mathsf{env}}(\varphi, \sigma) \not\subset \Sigma^{\mathcal{E}}_{\mathsf{env}}(\varphi, \sigma')$ for every σ'. 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": if it used σ_1 instead it could achieve the goal against a strictly larger set of environment strategies. This explains why, within this framework, maximal strategies are called "best-effort". When the goal φ and the env. specification $\mathcal{E}$ are clear from the context we may say "enforcing", "dominant", and "best-effort" without explicitly saying "for the goal φ under the environment specification $\mathcal{E}$".

An easy consequence of the definitions is the following:

Theorem 1 (Hierarchy)**.** *For a goal φ and environment specification $\mathcal{E}$, every enforcing strategy is dominant, and every dominant strategy is best-effort. Moreover:*

1. *If there exists an enforcing agent strategy then the dominant agent strategies are exactly the enforcing ones.*
2. *If there exists a dominant agent strategy then the best-effort agent strategies are exactly the dominant ones.*

Best-effort strategies always exist, and finding them is not harder than solving the classic synthesis problem.

Theorem 2. *(Aminof, De Giacomo, and Rubin 2021) Given LTL formulas $\varphi, \mathcal{E}$, there exists a best-effort strategy for φ under $\mathcal{E}$. Furthermore, finding such a strategy can be done in 2EXPTIME.*

The following example shows that dominant strategies (and thus also enforcing strategies) do not always exist.

Example 1. Let the environment and the agent each have a single variable (x, y, respectively), and consider a 'matching pennies' game where the agent moves first. This can be encoded by the goal $\varphi \doteq y \Leftrightarrow \mathsf{X}x$. The environment is $\mathcal{E} \doteq \mathsf{true}$. Clearly, the agent has no strategy that enforces the goal. Moreover, there is no dominant strategy since the two agent strategies, i.e., σ_i in which the agent's first move is $y = i$ for $i \in \{\mathsf{false}, \mathsf{true}\}$, are incomparable (technically, there are infinitely many agent strategies; however, effectively there are only two since only the first agent move is meaningful for the given goal). Indeed, let δ_i be the environment strategy that always does $x = i$, for $i \in \{\mathsf{false}, \mathsf{true}\}$, and note that $\text{PLAY}(\sigma_i, \delta_j) \models \varphi$ iff $i = j$. Finally, note that all agent strategies are best-effort.

The following example shows a case where enforcing strategies do not exist, but dominant ones do.

Example 2. Consider a robotic vacuum cleaner which at each time step cleans one of two possible rooms. This is encoded using the agent variable $\mathbf{Y} = \{C_A\}$, where $C_A = \mathsf{true}$ (resp. $C_A = \mathsf{false}$) means that the robot cleans room A (resp. B). Also, at each time step, the environment can optionally add dirt to each room. This is encoded using the environment variables $\mathbf{X} = \{D_A, D_B\}$, where a true variable means that dirt is added to the corresponding room. Consider the goal $\varphi \doteq \mathsf{G}((D_A \supset C_A) \wedge (D_B \supset \neg C_A))$ stating that when dirt is added to a room it should be cleaned in the same time step. The environment is $\mathcal{E} \doteq \mathsf{true}$. The agent has no enforcing strategy since the goal is violated if at any time both environment variables are true; however, it has dominant strategies: e.g., the strategy σ that at every time step cleans room A if dirt was added to it, and otherwise cleans room B. Finally, note that if we take $\mathcal{E} \doteq \mathsf{F}(D_A \wedge \mathsf{X}D_A) \supset \mathsf{G}\neg D_B$, then more agent strategies become dominant: e.g., the strategy σ' that behaves like σ except that once it sees that D_A holds for two consecutive time steps always cleans room A.

In Theorem 10 we show that finding a dominant strategy is 2EXPTIME hard. This fact, combined with Theorem 1 and Theorem 2, imply that from a practical standpoint — at least when viewing things through the lens of complexity classes — one should always look for a best-effort strategy, instead of a dominant or an enforcing one. Indeed, the cost of finding a best-effort strategy is not higher than looking for a dominant or enforcing one, and in case a dominant (resp. enforcing) strategy exists, the returned best-effort strategy will be dominant (resp. enforcing). Hence, the main interest when it comes to dominant strategies (or enforcing ones for that matter) is deciding or characterizing when they exist.

Characterizing Dominant Strategies

While the appeal of enforcing strategies is obvious, and the appeal of best-effort strategies was argued before – see (Berwanger 2007; Aminof, De Giacomo, and Rubin 2021) for more on this – one may justifiably ask what is the appeal of dominant strategies. One answer is the following: dominant strategies (like enforcing strategies) have the property that they are interchangeable. In other words, two dominant agent strategies σ_1 and σ_2 perform exactly the same in all environments: they either both achieve the goal or both fail to do so. On the other hand, choosing between two best-effort strategies σ_1 and σ_2 (that are not dominant) is always a gamble in the sense that against some non-empty set of environment strategies, σ_1 achieves the goal and σ_2 does not, while against another non-empty set of environment strategies the situation will be reversed. Thus: selecting an enforcing strategy ensures always getting the best *imaginable outcome*; selecting a dominant strategy ensures that one will always get the best *possible outcome*, whereas selecting a best-effort strategy ensures that one has made the best *possible decision* (given that the decision has to be made before the exact environment is known).

Hopeful characterization Another answer to the question of why dominant strategies are appealing is given by the following characterization which states, intuitively, that a dominant strategy achieves the goal in all cases except when it is absolutely impossible.

Call an environment strategy σ_{env} *hopeless* (wrt φ) if it enforces $\neg\varphi$; otherwise call it *hopeful*. Thus, the agent has no way of achieving φ against hopeless environments.

Proposition 3. *Given $\varphi, \mathcal{E}$, an agent strategy σ is dominant (wrt $\geq_{\varphi|\mathcal{E}}$) iff* $\text{PLAY}(\sigma, \sigma_{\mathsf{env}}) \models \varphi$ *for every hopeful environment strategy $\sigma_{\mathsf{env}} \in \Sigma^{\mathcal{E}}_{\mathsf{env}}$.*

Proof. For the "if" direction: clearly, for every agent strategy σ' we have that $\Sigma^{\mathcal{E}}_{\mathsf{env}}(\varphi, \sigma')$ does not contain any hopeless strategy, and thus $\Sigma^{\mathcal{E}}_{\mathsf{env}}(\varphi, \sigma') \subseteq \Sigma^{\mathcal{E}}_{\mathsf{env}}(\varphi, \sigma)$, so σ is dominant. For the other direction, assume that σ is dominant and consider a hopeful environment strategy $\sigma_{\mathsf{env}} \in \Sigma^{\mathcal{E}}_{\mathsf{env}}$. Being hopeful, there is an agent strategy σ' such that $\sigma_{\mathsf{env}} \in \Sigma^{\mathcal{E}}_{\mathsf{env}}(\varphi, \sigma')$, and since σ is dominant then $\Sigma^{\mathcal{E}}_{\mathsf{env}}(\varphi, \sigma') \subseteq \Sigma^{\mathcal{E}}_{\mathsf{env}}(\varphi, \sigma)$. Hence, $\sigma_{\mathsf{env}} \in \Sigma^{\mathcal{E}}_{\mathsf{env}}(\varphi, \sigma)$ as required. □

When there is no environment specification, i.e., when we take $\mathcal{E} = \mathsf{true}$, and there are no enforcing agent strategies, then a somewhat surprising connection between the existence of dominant agent strategies and the existence of oblivious hopeless environment strategies exists:

Proposition 4. *Given a goal φ, if the agent has a dominant strategy for φ, that does not enforce φ, then there is an* oblivious *environment strategy enforcing $\neg\varphi$.*

Proof. Let σ_{ag} be a dominant but not enforcing strategy. By Theorem 1, the agent cannot enforce φ and thus, by the Borel-determinacy (Martin 1975), of the corresponding game, the environment can enforce $\neg\varphi$, say using the strategy σ_{env}. Let $\pi \doteq \text{PLAY}(\sigma_{\mathsf{ag}}, \sigma_{\mathsf{env}})$, and let σ'_{env} be the oblivious strategy defined by letting $\sigma'_{\mathsf{env}}(h) = \pi_{|h|}$. Observe that $\text{PLAY}(\sigma_{\mathsf{ag}}, \sigma'_{\mathsf{env}}) = \pi$, and that $\pi \not\models \varphi$. Since σ_{ag} is dominant it follows that $\text{PLAY}(\sigma, \sigma'_{\mathsf{env}}) \not\models \varphi$ for every agent strategy σ, so σ'_{env} enforces $\neg\varphi$, i.e., is hopeless. □

Proposition 4 gives a necessary condition — namely the existence of an oblivious hopeless environment strategy — for the existence of dominant non-enforcing strategies for a goal φ. This condition is, however, not sufficient. To see that, take Example 1 but modify the goal to be $\varphi \doteq x \wedge (y \Leftrightarrow \mathsf{X}\, x)$. Note that an environment strategy that starts with $\neg x$ is hopeless, but there is no dominant strategy (use the same arguments as for the original goal in Example 1).

At first glance, Proposition 4 suggests that dominant strategies may be somewhat rare: why should one expect that $\neg\varphi$ can be enforced by an oblivious environment that is blind to the agent's actions? Note, however, that this proposition only covers the case with no environment specification, where it is quite likely that the modeling over-approximates the possible environments so much that it admits spurious oblivious hopeless strategies. Once an environment specification is introduced, such strategies would most probably fail to enforce it, and will be ignored by Definition 1.

Monotonicity wrt tightening Another desirable property shared by enforcing and dominant strategies is monotonicity with respect to tightening of environment specifications. Given $\mathcal{E}_1, \mathcal{E}_2$, we say that $\mathcal{E}_1$ is a *tightening* of $\mathcal{E}_2$ if $\Sigma^{\mathcal{E}_1}_{\mathsf{env}} \subseteq \Sigma^{\mathcal{E}_2}_{\mathsf{env}}$. Proposition 3 implies that, given a goal φ, every dominant strategy for φ under $\mathcal{E}_2$ is also dominant for φ under a tightening $\mathcal{E}_1$ of $\mathcal{E}_2$. I.e., tightening the environment specification cannot remove dominant strategies, but may add ones. (The same holds for enforcing strategies, as can be easily deduced from the definition.)

Proposition 5. *For a goal φ and env. specifications $\mathcal{E}_1, \mathcal{E}_2$ s.t. $\Sigma^{\mathcal{E}_1}_{\mathsf{env}} \subseteq \Sigma^{\mathcal{E}_2}_{\mathsf{env}}$, if σ is dominant (resp. enforcing) for φ under $\mathcal{E}_2$ then it is dominant (resp. enforcing) for φ under $\mathcal{E}_1$.*

It is interesting to note that this monotonicity is *not* in general true for best-effort strategies. That is, a strategy that is best-effort for a goal under one environment specification may or may not be best effort for the same goal under a tightening of that specification. In fact, the situation is quite intricate: (Aminof et al. 2021) show an example where tightening the specification *reduces the number* of best-effort strategies, as well as an example with three environments $\Sigma^{\mathcal{E}_1}_{\mathsf{env}} \subseteq \Sigma^{\mathcal{E}_2}_{\mathsf{env}} \subseteq \Sigma^{\mathcal{E}_3}_{\mathsf{env}}$ and an agent strategy σ that is best-effort for a goal φ under $\mathcal{E}_1$ as well as $\mathcal{E}_3$, but not under $\mathcal{E}_2$.

We remark that the possible lack of monotonicity for best-effort strategies is mitigated by the following facts. First, by Theorem 1, when dominant or enforcing strategies exist then they are exactly the best-effort strategies. Second, it is shown in (Aminof et al. 2021) that given a goal φ and a chain of environment specifications $\Sigma^{\mathcal{E}_1}_{\mathsf{env}} \subseteq \Sigma^{\mathcal{E}_2}_{\mathsf{env}} \subseteq \cdots \subseteq \Sigma^{\mathcal{E}_n}_{\mathsf{env}}$, one can always find an agent strategy σ that is simultaneously best-effort for φ under $\mathcal{E}_i$ for all $1 \leq i \leq n$.

Local Characterization of Dominant Strategies

We now provide a local characterization of the dominant strategies. Let $H_{\mathcal{E}}(\sigma_{\mathsf{ag}})$ be the set of histories h that are consistent with the agent strategy σ_{ag} and for which $\Sigma^{\mathcal{E},h}_{\mathsf{env}} \neq \emptyset$.

Definition 4. *Given $\varphi, \mathcal{E}$, an agent strategy σ_{ag}, and a history $h \in H_{\mathcal{E}}(\sigma_{\mathsf{ag}})$, define $val_{\varphi|\mathcal{E}}(\sigma_{\mathsf{ag}}, h)$ as follows:*

- $val_{\varphi|\mathcal{E}}(\sigma_{\mathsf{ag}}, h) := 1$ *(winning) if* $\text{PLAY}(\sigma_{\mathsf{ag}}, \sigma_{\mathsf{env}}) \models \varphi$ *for every $\sigma_{\mathsf{env}} \in \Sigma^{\mathcal{E},h}_{\mathsf{env}}$;*
- $val_{\varphi|\mathcal{E}}(\sigma_{\mathsf{ag}}, h) := -1$ *(losing) if* $\text{PLAY}(\sigma_{\mathsf{ag}}, \sigma_{\mathsf{env}}) \models \neg\varphi$ *for every $\sigma_{\mathsf{env}} \in \Sigma^{\mathcal{E},h}_{\mathsf{env}}$;*
- $val_{\varphi|\mathcal{E}}(\sigma_{\mathsf{ag}}, h) := 0$ *(pending) otherwise.*

We can compare agent strategies by looking at histories at which they make a different decision, i.e., "split". Formally, σ_1, σ_2 *split* at a history h if h ends in an environment move, is consistent with σ_1 and with σ_2, and $\sigma_1(h) \neq \sigma_2(h)$. The following proposition characterizes dominance by comparing the values of agent strategies at their split points.

Proposition 6 (Characterization of Dominance)**.** *(Aminof et al. 2020; Berwanger 2007) Given agent strategies σ_1, σ_2, we have that $\sigma_1 \geq_{\varphi|\mathcal{E}} \sigma_2$ iff for every history h with $\Sigma^{\mathcal{E},h}_{\mathsf{env}} \neq \emptyset$, at which σ_1, σ_2 split:*

(1) $val_{\varphi|\mathcal{E}}(\sigma_1, h) \geq val_{\varphi|\mathcal{E}}(\sigma_2, h)$, and
(2) it does not hold that $val_{\varphi|\mathcal{E}}(\sigma_1, h) = val_{\varphi|\mathcal{E}}(\sigma_2, h) = 0$.

The next definition assigns values to histories alone:

Definition 5. *(Aminof et al. 2020; Berwanger 2007) For a history h, define $val_{\varphi|\mathcal{E}}(h)$ as the maximum of $val_{\varphi|\mathcal{E}}(\sigma_{\mathsf{ag}}, h)$, where σ_{ag} varies over all agent-strategies for which $h \in H_{\mathcal{E}}(\sigma_{\mathsf{ag}})$; if there are no such strategies then write $val_{\varphi|\mathcal{E}}(h) = und$ (which stands for "undefined").*

We also call a history h "winning", "pending", or "losing" if $val_{\varphi|\mathcal{E}}(h)$ is 1, 0, or -1, respectively.

Proposition 7 (Local characterisation of best-effort)**.** *(Aminof et al. 2020; Berwanger 2007) An agent strategy σ is best-effort (wrt $\geq_{\varphi|\mathcal{E}}$) iff $val_{\varphi|\mathcal{E}}(\sigma, h) = val_{\varphi|\mathcal{E}}(h)$ for every history $h \in H_{\mathcal{E}}(\sigma)$ that ends in an environment move.*

The following characterizes dominant strategies.

Theorem 8 (Local characterisation of dominant)**.** *An agent strategy σ_1 is dominant (wrt $\geq_{\varphi|\mathcal{E}}$) iff for every $h \in H_{\mathcal{E}}(\sigma_1)$ that ends in an environment move:*

(a) if $val_{\varphi|\mathcal{E}}(h) = 1$ then $val_{\varphi|\mathcal{E}}(\sigma_1, h) = 1$, and
(b) if $val_{\varphi|\mathcal{E}}(h) = 0$ then (i) $val_{\varphi|\mathcal{E}}(\sigma_1, h) = 0$, and (ii) $val_{\varphi|\mathcal{E}}(h \cdot Y') = -1$ for every agent move $Y' \neq \sigma_1(h)$.

Proof. We start with the "if" direction. Let σ_2 be any agent strategy, and let h be any history with $\Sigma^{\mathcal{E},h}_{\mathsf{env}} \neq \emptyset$ at which σ_1, σ_2 split. Note that this means, in particular, that $h \in H_{\mathcal{E}}(\sigma_1)$, and thus conditions (a) and (b) above apply to h. To show that σ_1 is dominant it is enough to show that the conditions (1) and (2) of Proposition 6 hold. There are three cases to consider. First, if $val_{\varphi|\mathcal{E}}(h) = 1$, then by (a) also $val_{\varphi|\mathcal{E}}(\sigma_1, h) = 1$, and so (1) and (2) hold. Second, if $val_{\varphi|\mathcal{E}}(h) = -1$, then $val_{\varphi|\mathcal{E}}(\sigma_1, h) = val_{\varphi|\mathcal{E}}(\sigma_2, h) = -1$, and so (1) and (2) hold. Third, if $val_{\varphi|\mathcal{E}}(h) = 0$ then by the first part of (b) we have that $val_{\varphi|\mathcal{E}}(\sigma_1, h) = 0$ so (1) holds; since σ_1, σ_2 split at h we have that $\sigma_1(h) \neq \sigma_2(h)$ and thus by the second part of (b) $val_{\varphi|\mathcal{E}}(\sigma_2, h) = val_{\varphi|\mathcal{E}}(\sigma_2, h \cdot \sigma_2(h)) = -1$ and (2) holds.

For the other direction, assume that σ_1 is dominant and let h be a history as defined in the condition of this theorem. Since a dominant strategy is in particular maximal then by Proposition 7 $val_{\varphi|\mathcal{E}}(h) = val_{\varphi|\mathcal{E}}(\sigma_1, h)$, and thus (a) and the first part of (b) hold. Assume by contradiction that the second part of (b) does not hold, i.e., that there is some $Y' \neq \sigma_1(h)$ with $val_{\varphi|\mathcal{E}}(h \cdot Y') \neq -1$. Since h ends in an environment move then $val_{\varphi|\mathcal{E}}(h \cdot Y') \leq val_{\varphi|\mathcal{E}}(h)$ and thus, since $val_{\varphi|\mathcal{E}}(h) = 0$, it must be that $val_{\varphi|\mathcal{E}}(h \cdot Y') = 0$. Let σ_2 be an agent strategy that witnesses the fact that $val_{\varphi|\mathcal{E}}(h \cdot Y') = 0$ (i.e., $h \cdot Y' \in H_{\mathcal{E}}(\sigma_2)$ and $val_{\varphi|\mathcal{E}}(\sigma_2, h \cdot Y') = 0$), and note that $h \cdot Y' \in H_{\mathcal{E}}(\sigma_2)$ implies that $\sigma_2(h) = Y'$. It follow that $val_{\varphi|\mathcal{E}}(\sigma_2, h) = 0$ and that σ_1, σ_2 split at h. Recall that the assumption that σ_1 is dominant implies that $\sigma_1 \geq_{\varphi|\mathcal{E}} \sigma_2$, and obtain a contradiction to item (2) in Proposition 6. □

Theorem 8 highlights an interesting fact about dominant strategies: all dominant strategies behave exactly the same on pending histories (that do not extend winning ones). In other words, the difference between two dominant strategies is only in how they win from winning histories, or lose from losing histories. This is the basis of the following characterization of when dominant strategies exist:

Theorem 9. *The agent has a dominant strategy for φ under $\mathcal{E}$ iff for every history h that ends in an environment move:*

(a) either $val_{\varphi|\mathcal{E}}(h') = 1$ for some prefix h' of h, or
(b) $val_{\varphi|\mathcal{E}}(h \cdot Y') = 0$ for at most one agent move Y'.

Proof. For the forward direction, let σ be a dominant strategy, and take some history h ending in an environment move. Consider the longest prefix $h' \in H_{\mathcal{E}}(\sigma)$ of h ending in an environment move (h' always exists since the empty history is in $H_{\mathcal{E}}(\sigma)$). If $val_{\varphi|\mathcal{E}}(h') = 1$ then (a) holds. If $val_{\varphi|\mathcal{E}}(h') \in \{-1, und\}$ then the same is true for all of the extensions of h' and (b) holds. If $val_{\varphi|\mathcal{E}}(h') = 0$, by Theorem 8 we have that $val_{\varphi|\mathcal{E}}(h' \cdot Y') = -1$ for every $Y' \neq \sigma(h')$, that $val_{\varphi|\mathcal{E}}(h' \cdot \sigma(h')) = 0$, and that if $h = h'$ then (b) holds and we are done. If, however, $h \neq h'$, then either h extends $h' \cdot Y'$ for $Y' \neq \sigma(h')$, in which case h and all its extensions are also losing and (b) holds; or h extends $h' \cdot \sigma(h')$ and thus (recall that h ends in an environment move) there is some environment move X' such that $h'' \doteq h' \cdot \sigma(h') \cdot X'$ is a prefix of h. Observe that h'' is consistent with σ, and by our choice of h' we have that $h' \notin H_{\mathcal{E}}(\sigma)$. This implies that $\Sigma^{\mathcal{E},h''}_{\mathsf{env}} = \emptyset$, i.e., that h'' is not consistent with any strategy in $\Sigma^{\mathcal{E}}_{\mathsf{env}}$. The same is obviously true for any extension of h'' and thus, for any agent move Y', we have that $val_{\varphi|\mathcal{E}}(h \cdot Y') = und$, and (b) holds.

For the other direction, assume (a) and (b) above hold. Given a history h, observe that †: if $val_{\varphi|\mathcal{E}}(h) = 0$ then it must be that (b) holds (since extensions of winning histories cannot be pending). Let σ be a best-effort strategy (by Theorem 2 there is such a strategy), and apply Proposition 7 to σ in order to obtain, together with †, that the conditions of Theorem 8 hold, and thus σ is dominant. □

We remark that in the statement of Theorem 9, the case where $val_{\varphi|\mathcal{E}}(h)$ is undefined falls under condition (b).

Computing Dominant Strategies

The main computational problem addressed in this work is to establish the complexity and algorithms for deciding if a dominant strategy exists and in this case finding one.

Definition 6 (Computational problem)**.** *Given an* LTL/LTL$_f$ *goal φ and environment specification $\mathcal{E}$, the* dominant synthesis problem under environment specifications *is to find an agent strategy that is dominant for φ under $\mathcal{E}$, or say there is none.*

Theorem 10. *The dominant synthesis problem for* LTL/LTL$_f$ *goals, even without env. specifications, is 2EXPTIME hard. This is true even for the problem of deciding the existence of dominant strategies.*

Proof. We will show the latter statement, from which the former follows. The proof is by reduction the LTL/LTL$_f$ synthesis problem, known to be 2EXPTIME hard (Pnueli and Rosner 1989; De Giacomo and Vardi 2015). Assume w.l.o.g. that $x \in \mathbf{X}$ and $y, y' \in \mathbf{Y}$ are some of the agent and environment variables. Given a goal φ, consider the formula $\psi \doteq (y \rightarrow \mathsf{X}\varphi) \wedge (\neg y \rightarrow (y' \Leftrightarrow \mathsf{X}x))$. Intuitively, ψ says that if the agent asserts y in his first move then he wants to play for φ, and otherwise he wants to play "rigged matching pennies". Consider the history $h \doteq \{x\}$ of the single environment move asserting x (this choice is arbitrary, and any other move will do). Observe that this history is winning for ψ iff the agent can enforce φ. Furthermore, if the agent cannot enforce φ then there are at least two actions Y' (namely $Y' \doteq \{\neg y, y'\}$, $Y' \doteq \{\neg y, \neg y'\}$) with

$val_{\varphi|\mathcal{E}}(h \cdot Y') = 0$. Hence, by Theorem 9, there is a dominant strategy for ψ iff there is an enforcing strategy for φ. Note that for LTL$_f$, the agent may terminate the matching pennies game too early, and thus we replace the second conjunct by $\neg y \rightarrow (\mathsf{X}\,\mathsf{true} \wedge (y' \Leftrightarrow \mathsf{X}\,x))$. □

In the remainder of this section we develop algorithms for solving the dominant synthesis problem under environment specifications for LTL/LTL$_f$. Our algorithm returns an enforcing strategy if there is one, a dominant strategy if there is one, and a best-effort strategy otherwise.

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* $\delta : Q \times \Sigma \rightarrow Q$. The *size* of D is the number of its states.

Let $\alpha = \alpha_0\alpha_1 \cdots$ be a finite or infinite sequence of letters in 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 $Q \times \Sigma^* \rightarrow Q$ 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 7. *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 $D_1 \times D_2 \times \cdots \times D_k$ can be similarly defined for any finite sequence $D_1, D_2, \cdots, D_k$ of transition systems over the same input alphabet Σ.*

Automata. A *Deterministic Finite Word automaton (DFA)* $\mathcal{A} = (D, F)$ consists of a deterministic transition system D and a set $F \subseteq Q$ of *accepting states*. A finite run $\rho = q_0q_1 \cdots q_n$ in D is called *accepting* if $q_n \in F$. A finite string $\alpha \in \Sigma^*$ is *accepted* by $\mathcal{A}$ iff its run is accepting. The set of finite strings accepted by $\mathcal{A}$ is the *language* of $\mathcal{A}$. A *Deterministic Parity Word automaton (DPA)* $\mathcal{A} = (D, c)$ consists of a deterministic transition system D and a *coloring function* $c : Q \rightarrow \mathbb{Z}$. The *index* of $\mathcal{A}$ 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 $\rho = q_0q_1 \cdots$ in D *satisfies* c (aka *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 $\mathcal{A}$ iff the run induced by it satisfies c. The set of infinite strings accepted by $\mathcal{A}$ is the *language* of the $\mathcal{A}$.

Theorem 11 (Formulas to Automata)**.** *(cf. (Vardi 1995) and (De Giacomo and Vardi 2013))*

1. *One can build a DFA $\mathcal{A}_\varphi$, accepting exactly the models of an* LTL$_f$ *formula φ, whose size is at most 2EXP in $|\varphi|$.*
2. *One can build a DPA $\mathcal{A}_\varphi$, accepting exactly the models of an* LTL *formula φ, whose size is at most 2EXP in $|\varphi|$ and whose index is at most EXP in $|\varphi|$.*

We can lift the acceptance conditions of components of D to D as follows:

Definition 8. *For k-many DPAs $\mathcal{A}_i = (D_i, c_i)$ (or DFAs $\mathcal{A}_i = (D_i, F_i)$), and writing D for the product of their transition systems $D_1, \cdots, D_k$, define the* lifting *of c_j to D (resp.* lifting *of F_j to D) to be the coloring function $d_j : Q \rightarrow \mathbb{Z}$ defined by $d_j(q_1, \cdots, q_k) \doteq c_j(q_j)$ (resp. the set $Q_1 \times \cdots \times Q_{i-1} \times F_i \times Q_{i+1} \times \cdots \times Q_k$).*

Note that (D, d_j) is a DPA (resp. (D, F_j) is a DFA), and its language is the same as the language of the DPA (D_j, c_j) (resp. DFA (D_j, F_j)).

We sometimes limit the environment to moves that do not leave a set Q' of states. We do this by adding 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 9. *Let $D = (\Sigma, Q, \iota, \delta)$ be a transition system and let $Q' \subseteq Q$ be a set of states. The* environment-restriction of D to Q' *is the transition system $(\Sigma, Q \cup \{sink\}, \iota, \delta')$, 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 $Z \subseteq \mathbf{Y}$.*

Games on Deterministic Automata. It is known that solving synthesis (without environment specifications) for an LTL/LTL$_f$ goal φ can be reduced to solving a two-player game of perfect information on the DPA/DFA corresponding to φ. We describe this process below since we will later generalize it. The contents of this section are an adaptation of standard material on games-graphs (Apt and Grädel 2011).

Informally, the current position in the game is a state q of the automaton, first the environment moves by setting $X' \subseteq \mathbf{X}$, then the agent follows by setting $Y' \subseteq \mathbf{Y}$, 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 is accepting.

Formally, a *DPA-game* (resp. a *DFA-game*) is played on a DPA $\mathcal{A} = (D, c)$ (resp. DFA $\mathcal{A} = (D, F)$), by two players: *agent* and *environment*. The transition system $D = (2^{\mathbf{X} \cup \mathbf{Y}}, Q, \iota, \delta)$ is called the *arena*, and the acceptance condition of $\mathcal{A}$ is called the *objective*. An agent strategy σ_{ag} is *winning* if for every environment strategy σ_{env}, the trace PLAY$(\sigma_{\mathsf{ag}}, \sigma_{\mathsf{env}})$ is accepted by $\mathcal{A}$. Similarly, an environment strategy σ_{env} is *winning* if for every agent strategy σ_{ag}, the trace PLAY$(\sigma_{\mathsf{ag}}, \sigma_{\mathsf{env}})$ is *not* accepted by $\mathcal{A}$ (the objective is from the agent's point of view). A player's *winning region* in the game on $\mathcal{A}$ is the set of states q for which that player has a winning strategy in the game with the same objective but on the arena $(2^{\mathbf{X} \cup \mathbf{Y}}, Q, q, \delta)$, i.e., starting from q. A strategy winning from every state in the 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 automaton (instead of the exact full history of environment moves), then we say that the agent is using a *positional strategy*. Formally, a function $f_{\mathsf{ag}} : Q \times 2^{\mathbf{X}} \rightarrow 2^{\mathbf{Y}}$ induces the *positional* agent strategy σ_{ag}: $\sigma_{\mathsf{ag}}(X_0) = f_{\mathsf{ag}}(\iota, X_0)$, and for $n > 0$ and $\alpha = X_0Y_0X_1Y_1 \cdots X_{n-1}Y_{n-1}$, define $\sigma_{\mathsf{ag}}(\alpha \cdot X_n) = f_{\mathsf{ag}}(q_n, X_n)$ where $q_n \doteq \delta(\iota, \alpha)$. We can similarly define environment positional strategies as functions $f_{\mathsf{env}} : Q \rightarrow 2^{\mathbf{X}}$.

The relevant computational problem associated with a DPA/DFA-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. This can be done for DPA/DFA games by a simple fixed-point algorithm.

Theorem 12 (cf. (Apt and Grädel 2011))**.** *DFA-games can be solved in time polynomial in the size of the given DFA $\mathcal{A}$; DPA-games can be solved in time polynomial in the size, and EXP in the index, of the given DPA $\mathcal{A}$.*

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 automaton $\mathcal{A}$. The *co-operative winning region* W' of a DPA/DFA-game $\mathcal{A}$ 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/DFA-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 13 (cf. (Apt and Grädel 2011))**.** *Co-operative DFA-games can be solved in time polynomial in the size of the given DFA $\mathcal{A}$; Co-operative DPA-games can be solved in time polynomial in the size and index of the given DPA $\mathcal{A}$.*

Unified Synthesis Algorithm

We now present our algorithms for solving the dominant synthesis problem under environment specifications for LTL/LTL$_f$. We present both algorithms together as a single algorithm since the steps are the same, and the only difference between LTL and LTL$_f$ is handled by the type of automata and games we use. The algorithm is an extension of the best-effort synthesis algorithm from (Aminof, De Giacomo, and Rubin 2021), and the fact that it correctly finds a best-effort strategy is shown there. We enhance this algorithm by adding checks to see if an enforcing or a dominant strategy exist; by Theorem 1, if an enforcing (resp. dominant) strategy exists then the best-effort strategy found is enforcing (resp. dominant).

Unifying Algorithm. Given LTL/LTL$_f$ formulas $\varphi, \mathcal{E}$:

1. Using Theorem 11, for every $\xi \in \{\neg\mathcal{E}, \mathcal{E} \supset \varphi, \mathcal{E} \wedge \varphi\}$ compute the DPAs (resp. DFAs) $\mathcal{A}_\xi = (D_\xi, F_\xi)$.
2. Using Definition 7 form (in linear time) the product $D = D_{\neg\mathcal{E}} \times D_{\mathcal{E}\supset\varphi} \times D_{\mathcal{E}\wedge\varphi}$. Using Definition 8, lift the final states of each component to the product, e.g., the lifted condition $G_{\neg\mathcal{E}}$ consists of all states $(q_{\neg\mathcal{E}}, q_{\mathcal{E}\supset\varphi}, q_{\mathcal{E}\wedge\varphi}) \in Q$ s.t. $q_{\neg\mathcal{E}} \in F_{\neg\mathcal{E}}$. Let δ denote D's transition function.
3. Use Theorem 12, compute a uniform positional winning agent strategy f_{ag} in the DPA-game (resp. DFA-game) $(D, F_{\mathcal{E}\supset\varphi})$. Let $W \subseteq Q$ be the agent's winning region.
4. If the initial state of D is in W then set $EnforceFlag =$ true, and goto step 8.
5. Use Theorem 12, find the environment's winning region $V \subseteq Q$ in the DPA-game (resp. DFA-game) $(D, F_{\neg\mathcal{E}})$.
6. Using Definition 9, compute (in linear-time) the environment-restriction D' of D to the set V.
7. Using Theorem 13, compute a co-operatively winning uniform positional strategy g_{ag} in the DPA-game (resp. DFA-game) $(D', F_{\mathcal{E}\wedge\varphi})$. Let $W' \subseteq Q$ be the co-operative winning region.
8. Compute the agent strategy σ_{ag} induced by the positional strategy $k_{\mathsf{ag}} : Q \times 2^{\mathbf{X}} \to 2^{\mathbf{Y}}$ defined as follows:

   If $EnforceFlag$ is true then define $k_{\mathsf{ag}}(q, X') = f_{\mathsf{ag}}(q, X')$ for $q \in W$, and $k_{\mathsf{ag}}(q, X')$ is arbitrary for $q \notin W$; otherwise, define $k_{\mathsf{ag}}(q, X') = f_{\mathsf{ag}}(q, X')$ if $q \in W$, and $k_{\mathsf{ag}}(q, X') = g_{\mathsf{ag}}(q, X')$ for $q \notin W$.
9. If $EnforceFlag$ is true, then **return** σ_{ag} and the statement "the strategy σ_{ag} is enforcing".
10. Otherwise, if there is some state $q \in Q$, reachable from the initial state, such that (a) there is no path from the initial state to q that visits a state in W; and (b) there is some env. move X' and two agent moves Y', Y'' such that both $\delta(q, X' \cup Y')$ and $\delta(q, X' \cup Y'')$ are in W', then **return** σ_{ag} and the statement "the strategy σ_{ag} is best-effort, and there is no dominant strategy". These conditions can be checked in polynomial time in the size of D.
11. Otherwise **return** σ_{ag} and the statement "the strategy σ_{ag} is dominant, and there is no enforcing strategy".

The last two steps of the algorithm are correct by Theorem 9 and Theorem 1.

The algorithm above, together with Theorem 10, give us:

Theorem 14. *Dominant synthesis under environment specifications (for LTL/LTL$_f$ goals and environment specifications) is 2EXPTIME-complete.*

Related Work

We have discussed the closely related work on (classic) synthesis (Pnueli and Rosner 1989), synthesis under environment specifications (Aminof et al. 2019), and best-effort synthesis (Aminof, De Giacomo, and Rubin 2021).

We now discuss trace-based variants of the notion of dominant synthesis. (Damm and Finkbeiner 2011, 2014; Finkbeiner and Passing 2020) study *remorse-free dominance*, which can be expressed using our terminology as follows: for an agent goal φ, define $\sigma_1 \geq_{tr} \sigma_2$ if $\text{PLAY}(\sigma_2, \sigma_{\mathsf{env}}) \models \varphi$ implies $\text{PLAY}(\sigma_2, \sigma_{\mathsf{env}}) \models \varphi$ for every *oblivious* strategy σ_{env}; and dominant strategies are defined to be maximums wrt $\geq_{tr}$. They provide an optimal automata-theoretic algorithm that solves synthesis of dominant strategies for LTL goals; prove that a dominant strategy exists iff there is a unique weakest environment assumption that would guarantee a winning strategy; and apply this to provide an incremental algorithm for solving distributed synthesis which finds a dominant strategy for one process at a time and propagating the assumption to the processes to be synthesized. An identical notion called *good-enough* synthesis is studied in (Almagor and Kupferman 2020) (although these papers define their notions differently, they can be easily seen to be equivalent by a slight variant of Proposition 3) mainly in a multi-valued setting, and in (Li et al. 2021) in the single-agent setting for LTL$_f$ goals.

We remark that the restriction to oblivious environment strategies inherent in these trace-based works is severe, and results in different properties then we have in our much broader setting.

Acknowledgments

This work is partially supported by the Austrian Science Fund (FWF) P 32021, by the ERC Advanced Grant WhiteMech (No. 834228), by the EU ICT-48 2020 project TAILOR (No. 952215), and by the PRIN project RIPER (No. 20203FFYLK).

References

Almagor, S.; and Kupferman, O. 2020. Good-Enough Synthesis. In Lahiri, S. K.; and Wang, C., eds., *CAV*, volume 12225 of *Lecture Notes in Computer Science*, 541–563. Springer.

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

Aminof, B.; De Giacomo, G.; Lomuscio, A.; Murano, A.; and Rubin, S. 2021. Synthesizing Best-effort Strategies under Hierarchical Environment Specifications. In *KR*.

Aminof, B.; De Giacomo, G.; Murano, A.; and Rubin, S. 2018. Synthesis under Assumptions. In *KR*.

Aminof, B.; De Giacomo, G.; Murano, A.; and Rubin, S. 2019. Planning under LTL Environment Specifications. In *ICAPS*.

Aminof, B.; De Giacomo, G.; and Rubin, S. 2021. Best-Effort Synthesis: Doing Your Best Is Not Harder Than Giving Up. In *IJCAI*.

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

Bacchus, F.; and Kabanza, F. 2000. Using temporal logics to express search control knowledge for planning. *Artif. Intell.*, 116(1-2).

Baier, J. A.; and McIlraith, S. A. 2006. Planning with First-Order Temporally Extended Goals using Heuristic Search. In *AAAI*.

Berwanger, D. 2007. Admissibility in Infinite Games. In *STACS*.

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

Cimatti, A.; Pistore, M.; Roveri, M.; and Traverso, P. 2003. Weak, Strong, and Strong Cyclic Planning via Symbolic Model Checking. *AIJ*, 1–2(147).

Damm, W.; and Finkbeiner, B. 2011. Does It Pay to Extend the Perimeter of a World Model? In *FM 2011*. Springer.

Damm, W.; and Finkbeiner, B. 2014. Automatic Compositional Synthesis of Distributed Systems. In *FM*.

Daniele, M.; Traverso, P.; and Vardi, M. 1999. Strong Cyclic Planning Revisited. In *ECP*.

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

De Giacomo, G.; and Vardi, M. Y. 2013. Linear Temporal Logic and Linear Dynamic Logic on Finite Traces. In *IJCAI*.

Faella, M. 2009. Admissible Strategies in Infinite Games over Graphs. In *MFCS*.

Finkbeiner, B.; and Passing, N. 2020. Dependency-Based Compositional Synthesis. In *ATVA*.

Geffner, H.; and Bonet, B. 2013. *A Coincise Introduction to Models and Methods for Automated Planning*. Morgan & Claypool.

Green, C. 1969. Theorem Proving by resolution as basis for question-answering systems. In *Machine Intelligence*, volume 4, 183–205. American Elsevier.

Li, Y.; Turrini, A.; Vardi, M. Y.; and Zhang, L. 2021. Synthesizing Good-Enough Strategies for LTLf Specifications. In *IJCAI*, 4144–4151. ijcai.org.

Martin, D. A. 1975. Borel determinacy. *Annals of Mathematics*, 363–371.

McCarthy, J.; and Hayes, P. J. 1969. Some Philosophical Problems From the StandPoint of Artificial Intelligence. *Machine Intelligence*, 4: 463–502.

Pnueli, A.; and Rosner, R. 1989. On the Synthesis of a Reactive Module. In *POPL*.

Vardi, M. Y. 1995. An Automata-Theoretic Approach to Linear Temporal Logic. In Moller, F.; and Birtwistle, G. M., eds., *Logics for Concurrency - Structure versus Automata*, volume 1043 of *LNCS*.