Proper linear-time specifications of environment behaviors in nondeterministic planning and reactive synthesis

Benjamin Aminof[1] , **Giuseppe De Giacomo**[1,2] , **Sasha Rubin**[3] , **Florian Zuleger**[4]
[1]University of Rome "La Sapienza", Italy
[2]University of Oxford, United Kingdom
[3]University of Sydney, Australia
[4]TU Vienna, Austria

Abstract

To help it achieve its goal, an agent exploits assumptions it has about the behavior of its environment. The common view in planning and reactive synthesis is that such assumptions are sets of traces. This trace-centric view has the advantage of having well-understood specification formalisms, such as linear-time temporal logic. An alternative view, that we have promoted as being conceptually superior, is strategy-centric: assumptions are non-empty sets of environment strategies. In this work we relate these views and show that the strategy-centric view is a refinement of the trace-centric view. We thus address the following fundamental question: when should a set of traces be considered an assumption that the agent has about the environment's behavior? Our answer is in terms of coverability: every trace in the set should be consistent with some environment strategy that enforces it. We call such sets "proper environment specifications". Typical examples are given by (the traces consistent with a given) planning domain, and fairness constraints, but not arbitrary trace constraints. We provide an algorithm that, given a specification in linear-time temporal logic (LTL) decides whether or not it is a proper environment specification. Furthermore, we show that every set of traces has a "proper environment core", which excludes traces that the agent can ignore when devising its plan. We provide an algorithm for computing a representation of the core of an LTL formula, and prove that the core of an LTL-definable property is itself LTL-definable.

1 Introduction

In reasoning about actions, the agent has some knowledge about the possible behaviors of its environment. Such knowledge is often represented in planning in an action description language (Haslum et al. 2019), and in reactive synthesis in a linear-time temporal logic (Pnueli and Rosner 1989). In devising its plans, the agent takes advantage of this knowledge: the agent *assumes* that the world behaves in a certain way, and *exploits such an assumption in devising its plans*. Thus, a specific plan of the agent is encapsulated by a function that describes, for every history of interaction between the agent and its environment, which action to do. Such functions are called *strategies* in reactive synthesis and *(history-dependent) policies* in planning.

While interacting with the agent, the environment (whether it is composed of other agents, nature, or a combination) is also constrained by the uni-directional flow of time in the sense that what it does at every moment in time can only depend on the history of the interaction so far, but not on the future. In other words, the environment can also be viewed as employing a function from histories to effects (Aminof et al. 2018).[1] We call such a function an *environment strategy* (note that by doing so we do not necessarily ascribe rationality/sentience to the environment).

In some cases, the agent's assumptions about its environment are precise enough to pin down a single strategy that the environment must use. This is the case with classical planning where the domain is deterministic: such a domain can be considered to be a single environment strategy that, for every history of interaction that ends with an agent action, updates the fluents. In other cases, the agent's knowledge is not enough to pinpoint a single environment strategy, and the agent must consider a set S of possible environment strategies. This is the case with planning in nondeterministic domains: the agent takes S to be the set of environment strategies that respect the domain's allowed effects. In general, the agent's knowledge of its environment is captured (naturally and generally) by a *set of environment strategies it considers possible* (Aminof et al. 2018). This *strategy-centric* view, in which assumptions are non-empty sets of environment strategies (Berwanger 2007; Faella 2009; Brenguier, Raskin, and Sassolas 2014), has been shown to be rich enough to describe general forms of nondeterministic planning, e.g., planning for temporally-extended goals under fairness (Aminof et al. 2018; Aminof et al. 2019), and is argued as being formally better suited, especially for advanced solution concepts such as best-effort synthesis or dominant-synthesis (Aminof, De Giacomo, and Rubin 2021; Aminof et al. 2022; Aminof et al. 2023; Aminof, De Giacomo, and Rubin 2023).

On the other hand, the common view of the agent's assumptions about its environment involves much simpler objects than sets of environment strategies. It is *trace-centric*: an assumption that the agent has about its environment is a set of traces, often expressed in a specification logic such as linear-time temporal logic (LTL). This view is found in planning (Gerevini and Long 2005; Bacchus and Kabanza 2000;

[1]Functions with randomness (for the agent, environment, or both) do not change the basic picture in this paper, and we focus on the case without randomness.

Bonet et al. 2017; Camacho, Bienvenu, and McIlraith 2019; Bonet and Geffner 2020; Rodriguez et al. 2022) (the planning domain itself can be recast as a set of traces definable in LTL), and in reactive synthesis (Pnueli and Rosner 1989; Bloem et al. 2014; Klein and Pnueli 2010; Li, Dworkin, and Seshia 2011; Bloem et al. 2012; D'Ippolito et al. 2013; Alur, Moarref, and Topcu 2013; Camacho, Bienvenu, and McIlraith 2018; Kress-Gazit, Lahijanian, and Raman 2018).

Thus, there is a fundamental tension between the trace-centric view and the strategy-centric one: when is a set E of traces representing the agent's knowledge about the environment a faithful proxy for a set S of environment strategies? The motivation for our work is to resolve this tension. We thus ask, for a set E of traces:

Q1: Does E exactly capture the possible interactions of the agent with its environment?

Q2: If no, which traces in E should the agent consider possible?

Contributions To answer Q1, we call a set E of traces *environment coverable* if it is the set of traces consistent with some set of environment strategies. If E is also non-empty, we call it a *proper environment specification*. Not surprisingly, planning typically considers proper environment specifications, e.g., the set E of traces consistent with a given nondeterministic planning domain is a proper environment specification, as is the subset that admits fairness of effects. However, refining E by arbitrary trajectory constraints need not result in a proper environment specification.

To answer Q2, we show that there is a unique largest subset of E that itself is environment coverable, that we call the *core of* E. Thus, if the agent thinks that E is an over-approximation of the true environment, then it may consider the core of E to be the true specification of the environment. Thus, to answer Q2, the agent should only consider as possible the traces that are in the core of E. Said differently, even though a trace may be specified by E, if it is not in the core of E, the agent can deduce that it cannot possibly arise out of an interaction it has with the environment, and thus it can ignore such traces when devising its plans.

For algorithmic purposes, we consider the case that E is definable in LTL. We provide algorithms for three problems: (A) deciding if a given LTL formula φ is a proper environment specification; (B) computing a representation of the core of a given LTL formula; (C) computing a trace that witnesses that φ is not a proper environment specification. As a by product, we show the interesting result that the core of an LTL formula φ is also LTL definable. Our algorithms take an automata-theoretic approach, first compiling φ into certain deterministic automata that operate on infinite traces.

2 Preliminaries

For a sequence $x = x_0x_1x_2\cdots$, the length of x is denoted $|x| \in \mathbb{N} \cup \{\infty\}$. For $0 \leq i \leq |x|$, we denote the prefix $x_0x_1\cdots x_{i-1}$ of x by $x_{<i}$ or $x_{\leq i-1}$, and we denote the suffix $x_ix_{i+1}\cdots x_{|x|-1}$ of x by $x_{\geq i}$. By convention, $\infty - 1 = \infty$. Thus $|x| - 1$ is the last index of x if x is finite, and ∞ otherwise. The empty string is denoted λ. For a finite set AP of *atomic propositions*, sequences over $\Sigma = 2^{AP}$ are called *traces*. A *trace property (aka property)* is a set E of infinite traces. If $\tau \in E$ we say that τ *satisfies* E. If E is non-empty we say that it is *satisfiable*.

Unless stated otherwise, we fix a set AP of atomic propositions for the rest of this paper. We will use lower-case letters to denote atoms, e.g., $x \in AP$, and upper-case letters to denote evaluations, e.g., $X \subseteq AP$.

Linear-time temporal logic The formulas of LTL over AP are defined by the following BNF (where $p \in AP$): $\varphi ::= p \mid \varphi \vee \varphi \mid \neg\varphi \mid \mathsf{X}\varphi \mid \varphi\,\mathsf{U}\,\varphi$. Here X is called the *next* operator, and U the *until* operator. We use the usual abbreviations, $\varphi \supset \varphi' \doteq \neg\varphi \vee \varphi'$, $\mathsf{true} \doteq p \vee \neg p$, $\mathsf{F}\,\varphi \doteq \mathsf{true}\,\mathsf{U}\,\varphi$ (read *eventually*), $\mathsf{G}\,\varphi \doteq \neg\,\mathsf{F}\,\neg\varphi$ (read *always*), etc. The *size* $|\varphi|$ of a formula φ is the number of symbols in it. Given an infinite trace τ, an integer $n \geq 0$, and an LTL formula φ, the satisfaction relation $(\tau, n) \models \varphi$, stating that φ *holds at step* n *of* 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 $(\tau, n+1) \models \varphi$; $(\tau, n) \models \varphi_1\,\mathsf{U}\,\varphi_2$ iff $(\tau, m) \models \varphi_2$ for some $m \geq n$, and $(\tau, j) \models \varphi_1$ for all j with $n \leq j < m$. Write $\tau \models \varphi$ if $(\tau, 0) \models \varphi$, read τ *satisfies* φ (Alternatively, τ is a *model* of φ). The set of traces that satisfy φ is denoted $L(\varphi)$; we will usually overload notation and write φ to denote the trace property $L(\varphi)$.

Reactive synthesis Let $\mathbf{X}$ and $\mathbf{Y}$ be disjoint finite sets of Boolean variables, called the *agent variables* and *environment variables*, respectively. Then $2^{\mathbf{X}}$ (resp. $2^{\mathbf{Y}}$) is the set of agent (resp. environment) *moves*. Let $AP = \mathbf{X} \cup \mathbf{Y}$ and $\Sigma = 2^{\mathbf{X}\cup\mathbf{Y}}$. Then a trace $(X_0 \cup Y_0)(X_1 \cup Y_1)\cdots$ over Σ is also called a *joint trace* to emphasize that it includes the moves of both the agent and the environment. A *play* $\pi = X_0 \cdot Y_0 \cdot X_1 \cdot Y_1 \cdots$ is an element of $(2^{\mathbf{X}} \cdot 2^{\mathbf{Y}})^\omega$. Plays are an alternative representation of joint-traces (Assuming, as we do, that the agent moves first) and it will be technically convenient to sometimes blur the distinction between them, e.g., if E is a set of traces and π is a play, we may say that π *is in* E to mean that the trace induced by π is in E.

A *history* h is a finite prefix of a play. Let H be the set of histories ending in environment moves including the empty-history, i.e., $H \doteq (2^{\mathbf{X}} \cdot 2^{\mathbf{Y}})^*$. An *agent strategy* is a function $\sigma_{\mathsf{ag}} : H \to 2^{\mathbf{X}}$ that maps histories ending in environment moves (including, since the agent moves first, the empty history) to agent moves. An *environment strategy* is a function $\sigma_{\mathsf{env}} : H \cdot 2^{\mathbf{X}} \to 2^{\mathbf{Y}}$ that maps histories ending in agent moves to environment moves. For an agent (resp. environment) strategy σ and a play/history ρ, say that ρ *is consistent with* σ if, for every prefix $\rho_{<i}$ in the domain of σ, 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}.

A play π *satisfies* a trace property E if the trace induced by π is in E. An agent strategy σ_{ag} *enforces* (aka, *realizes*) E if $\textsc{play}(\sigma_{\mathsf{ag}}, \sigma_{\mathsf{env}})$ satisfies E for every environment strategy σ_{env}, and in this case we say that φ is *agent enforceable*. Similarly, an environment strategy σ_{env} *enforces* (aka, *realizes*) E if $\textsc{play}(\sigma_{\mathsf{ag}}, \sigma_{\mathsf{env}})$ satisfies E for every agent strategy σ_{ag}, and in this case we say that φ is *environ-*

ment enforceable. We write $\mathrm{enf}(E)$ for the set of environment strategies enforcing E. Observe that $E' \subseteq E$ implies $\mathrm{enf}(E') \subseteq \mathrm{enf}(E)$. For a set S of environment strategies, let $\mathrm{con}(S)$ denote the set of plays consistent with some strategy in S. Note that $S \subseteq S'$ implies that $\mathrm{con}(S) \subseteq \mathrm{con}(S')$.

The *LTL synthesis problem* is: given an LTL formula φ and a player (agent or environment), return a strategy for that player that enforces φ, or say there is none. This problem is 2EXPTIME-complete (Pnueli and Rosner 1990).

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 $\alpha = \alpha_0\alpha_1 \cdots$ be a finite or infinite sequence of letters in Σ. The *run/path* induced by α (AKA the run of D on 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 range over $Q \times \Sigma^*$ as follows: $\delta(q, \lambda) = q$, and for $n > 0$, if $q_n = \delta(q, \alpha_0 \cdots \alpha_{n-1})$ then $\delta(q, \alpha_0\alpha_1 \cdots \alpha_n) = \delta(q_n, \alpha_n)$.

Automata A *Deterministic Parity Word automaton (DPA)* $\mathcal{A} = (D, c)$ consists of a deterministic transition system $D = (\Sigma, Q, \iota, \delta)$ and a *coloring function* $c : Q \to \mathbb{Z}$. 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 $\mathcal{A}$, which we denote by $L(\mathcal{A})$. A state $q \in Q$ is *reachable* iff there is a run $q_0q_1 \cdots q_n$ with $q_0 = \iota$ and $q = q_n$.

Theorem 1 (Formulas to Automata)**.** *cf. (Vardi 1995) 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 $|\varphi|$.*

Games on deterministic automata Solving synthesis for an LTL goal can be reduced to solving a two-player game played on the DPA for the formula. We recall the relevant definitions and results on automata games, adapted to our setting (Apt and Grädel 2011). Informally, the current position in the game is a state q of the automaton, first the agent moves by setting $X' \subseteq \mathbf{X}$, then the environment 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 *environment* is declared the winner if the run is accepting, i.e., the environment is the protagonist. Formally, a *DPA-game* is played on a DPA $\mathcal{A} = (D, c)$ 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 c is called the *objective*. An environment strategy σ_{env} is *winning* if for every agent strategy σ_{ag}, we have that $\textsc{play}(\sigma_{\mathsf{ag}}, \sigma_{\mathsf{env}})$ satisfies c (i.e., the objective is from the environment's point of view). The environment's *winning region* in the game on $\mathcal{A}$ is the set of states q for which the environment 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. The relevant computational problem, called *solving* the DPA-game, is to compute the environment's winning region W. DPA-games can be solved in time polynomial in the size, and exponential in the index, of the given DPA (Apt and Grädel 2011). We also consider the case where the agent and environment co-operate. The *co-operative winning region* W' of a DPA-game $\mathcal{A}$ is the set of states q for which there is a pair of strategies $\sigma_{\mathsf{ag}}, \sigma_{\mathsf{env}}$ for which $\textsc{play}(\sigma_{\mathsf{ag}}, \sigma_{\mathsf{env}})$ satisfies c in the arena $(2^{\mathbf{X}\cup\mathbf{Y}}, Q, q, \delta)$, i.e., starting from q (we call such a pair of strategies *co-operatively winning* from q). Solving a co-operative DPA-game on $\mathcal{A}$ means to find W'. Co-operative DPA-games can be solved in time polynomial in the size and index of the DPA (Apt and Grädel 2011).

Counter-free automata A *Deterministic Rabin Automaton (DRA)* $\mathcal{A} = (D, \Omega)$ consists of a deterministic transition system $D = (\Sigma, Q, \iota, \delta)$ and a Rabin acceptance condition $\Omega = \{(B_1, G_1), \ldots, (B_k, G_k)\}$, with $B_i, G_i \subseteq Q$, where k is called the index of the Rabin automaton. An infinite run $\rho = q_0q_1 \cdots$ in D *satisfies* Ω (aka is *accepted*) iff there is an $1 \leq i \leq k$ such that $q_j \in B_i$ for finitely many q_j, and $q_j \in G_i$ for infinitely many q_j. A DRA $\mathcal{A}$ is *counter-free* if for every state q, every word w, and every $k \geq 1$, we have that D returns from q to q reading w iff D returns from q to q reading w^k; for references to counter-free automata and their relationship to logic, see (Thomas 1981; Diekert and Gastin 2008; Boker, Lehtinen, and Sickert 2022). We use the following result:

Theorem 2. *(Thomas 1981) A set L of infinite words is representable by an LTL formula iff L is accepted by a* counter-free *DRA. Moreover, there are effective procedures for both conversions.*

3 Proper specifications

Definition 1. *A trace property E is an* environment coverable specification *(aka,* environment coverable*) if there is some set S of environment strategies such that a trace is in E iff it is consistent with a strategy in S. If, in addition E is satisfiable, then call E a* proper environment specification.

Proposition 3. *The following are equivalent for a trace property E:*

1. *E is environment coverable,*
2. *$E \subseteq \mathrm{con}(\mathrm{enf}(E))$, i.e., every trace in E is consistent with some environment strategy that enforces E.*
3. *$E = \mathrm{con}(\mathrm{enf}(E))$.*

Proof. To see that 1 implies 2 note that E is environment coverable means that $E = \mathrm{con}(S)$ for some S and thus $S \subseteq \mathrm{enf}(E)$; conclude that $E = \mathrm{con}(S) \subseteq \mathrm{con}(\mathrm{enf}(E))$. To see that 2 implies 3 use the fact that $\mathrm{con}(\mathrm{enf}(E)) \subseteq E$ for all E. Item 3 implies 1 by taking $S = \mathrm{enf}(E)$. □

If a play π is consistent with an environment strategy σ enforcing E then π satisfies E, and we call σ a *witness strategy* to π satisfying E. Thus, Proposition 3 (item 2) states that every trace in E has a witness strategy.

Remark 1. *If E is not satisfiable then it is trivially environment coverable (take $S = \emptyset$). If E is satisfiable, but not environment enforceable (i.e., $\emptyset \neq E$ and $\emptyset = \mathrm{enf}(E)$), then it*

is not a proper environment specification (by Proposition 3). However, not every satisfiable and environment-enforceable set E is a proper environment specification (see Example 1).

An important example of environment coverable specifications are those induced by planning domains, as we now explain. A state-based representation of a planning domain consists of a set $S = 2^F$ of *states* that are the evaluations of a finite set of Boolean variables called *fluents*; an initial state $\iota \in S$; a finite set A of actions; and a transition relation $\delta : S \times A \to 2^S$. We assume that for every s there is some a such that $\delta(s, a) \neq \emptyset$. An *execution* in the planning domain is an infinite sequence $s_0, a_0, s_1, a_1, \cdots$ such that $s_0 = \iota$ and $\delta(s_i, a_i) = s_{i+1}$ for all i. We can encode the domain into the setting of reactive synthesis as follows: let X be large enough so that $|A| \leq |2^X|$, i.e., so that it can encode each agent action as an evaluation of the variables in X, and let $Y = F$. Then, every execution is encoded by a play, i.e., $a_0 \cdot s_1 \cdot a_1 \cdot s_2 \cdot a_2 \cdot s_3 \cdot \cdots$ (for technical convenience we omit the initial state). If E is the property consisting of all such plays (technically, of the traces induced by such plays) then E is a proper environment specification. To see this we apply Proposition 3 (Item 2): the environment strategies that enforce E are those that respect δ, i.e., on a history $a_0 \cdot s_1 \cdot a_1 \cdot s_2 \cdot \cdots \cdot s_n \cdot a_n$ respond with some $s_{n+1} \in \delta(s_n, a_n)$, and every play in E is consistent with some such strategy. Similarly, the subset of E consisting of the fair traces is a proper environment specification.

We now provide some technical examples of trace properties E, given by an LTL formula φ, that are (resp. are not) proper environment specification. The justifications follow from Proposition 3.

Example 1. Say $\mathbf{X} = \{x\}$ and $\mathbf{Y} = \{y\}$; so, the agent moves are $\{x\}$ and $\emptyset$ (called "doing x" and "not doing x", respectively), and the environment moves are $\{y\}$ and $\emptyset$ (called "doing y" and "not doing y", respectively).

1. If φ only contains environment variables (e.g., $\varphi = y$, or $\varphi = \mathsf{G}\, y$), then E is environment coverable. Indeed, every play $\pi = X_0 \cdot Y_0 \cdot X_1 \cdot Y_1 \cdots$ satisfying φ is consistent with the environment strategy σ_{env} enforcing φ defined by: $\sigma_{\mathsf{env}}(h) = Y_{(|h|-1)/2}$, for every h.
2. If φ is satisfiable but not enforceable (e.g., $\varphi = x$, or $\varphi = x \wedge y$) then, by Remark 1, it is not a proper environment specification.
3. $\varphi = x \leftrightarrow y$ is a proper environment specification. Indeed, every play $\pi = X_0 \cdot Y_0 \cdot X_1 \cdot Y_1 \cdots$ that satisfies E is consistent with the following environment strategy that enforces E: for $h = \{x\}$ (resp. $h = \emptyset$) respond with $\{y\}$ (resp. $\emptyset$), and for every other h respond with $Y_{(|h|-1)/2}$. Similar reasoning shows that $E = x \vee y$ is a proper environment specification.
4. $\varphi = (\mathsf{F}\, x) \supset (\mathsf{G}\, y)$ is not a proper environment specification. Indeed, the play in which the agent never does x and the environment never does y satisfies E but is not consistent with any environment strategy enforcing φ since these must do y in case the agent ever does x in the future.
5. $\varphi = (\mathsf{X}\, \mathsf{F}\, x) \vee y$ is not a proper environment specification: the play $(\{x\} \cdot \{\})^\omega$ satisfies φ but is not in $\mathrm{con}(\mathrm{enf}(\varphi))$ since every environment strategy that enforces φ does y on its first move.
6. $\varphi = (y\, \mathsf{W}\, \neg x) \wedge \mathsf{G}(\neg x \supset \mathsf{G}\, \neg y)$ is a proper environment specification. Indeed, every play satisfying E is consistent with the following environment strategy that enforces E: the environment does y as long as the agent has only done x, and if ever the agent doesn't do x then from that point on the environment never does y.

The next result shows us how we may or may not build new environment coverable specifications, and proper environment specification, from old ones. We overload notation and apply temporal operators to sets of traces too, e.g., $\mathsf{X}\, E$ is the set of traces π such that $\pi_{\geq 1}$ satisfies E, and $\mathsf{G}\, E$ is the set of traces π such that $\pi_{\geq n}$ satisfies E for every $n \geq 0$.

Proposition 4 (Closure properties)**.** *The set of environment coverable specifications (resp. proper environment specifications) is closed under (arbitrary) $\cup$, X, and F, but not closed under $\neg$, $\cap$, U nor G.*

Proof. Note that, since $\cup$, X, and F preserve satisfiability, it is enough to prove the closure properties for environment coverable specifications. For the non-closure properties, observe that it is enough to show counter-examples where we start with proper environment specifications, but the operation yields a set that is not environment coverable.

For union, let $\mathfrak{F}$ be a family of environment coverable sets of traces. To see that the union $\cup\mathfrak{F}$ is environment coverable, observe that a play π satisfies $\cup\mathfrak{F}$ iff it satisfies some $E \in \mathfrak{F}$ and thus (since E is environment coverable and by Proposition 3 item 2) the trace induced by π is in $\mathrm{con}(\mathrm{enf}(E))$. Finally, recall that $E \subseteq \cup\mathfrak{F}$ implies $\mathrm{con}(\mathrm{enf}(E)) \subseteq \mathrm{con}(\mathrm{enf}(\cup\mathfrak{F}))$, and use Proposition 3 item 2 applied to $\cup\mathfrak{F}$.

The closures under X and F are similar, so we treat them at the same time. Suppose E is environment coverable and consider a play π satisfying $\mathsf{F}\, E$ (resp. $\mathsf{X}\, E$). Hence, for some $n \geq 0$ (resp. for $n = 1$) we have that $\pi_{\geq 2n}$ satisfies E. By Proposition 3 item 2 applied to E, there exists $\sigma \in \mathrm{enf}(E)$ such that $\pi_{\geq 2n} \in \mathrm{con}(\sigma)$. Define σ' to mimic the first n environment moves on π and then follow σ. Formally, given $h = X'_0 \cdot Y'_0 \cdots X'_m$: if $m < n$ then define $\sigma'(h) = \pi_{2m+1}$, and if $m \geq n$ then define $\sigma'(h) = \sigma(X'_n \cdot Y'_n \cdots X'_m)$. This ensures that π is consistent with σ', and that σ' enforces $\mathsf{F}\, E$ (resp. $\mathsf{X}\, E$). Now use Proposition 3 item 2 applied to $\mathsf{F}\, E$ (resp. $\mathsf{X}\, E$).

To handle the non-closure properties, we use the same sets of variables (and associated language to describe moves) as in Example 1, and we express properties in LTL.

For the non-closure under negation, let $\varphi = x \supset \mathsf{G}\, y$. It is a proper environment specification since every play $\pi = X_0 \cdot Y_0 \cdot X_1 \cdot Y_1 \cdots$ that satisfies φ is consistent with the following environment strategy σ_{env} that enforces φ: If $X_0 = \{x\}$, then $\sigma_{\mathsf{env}}(h) = \{y\}$ for every h; if $X_0 = \emptyset$, then $\sigma_{\mathsf{env}}(h) = Y_{(|h|-1)/2}$ for every h. Now consider $\varphi' = \neg\varphi$. Then the play $\pi = (\{x\} \cdot \{\})^\omega$ satisfies φ' but no environment strategy enforces φ' since the agent can simply not do x on its first move.

For the non-closure under intersection, recall from Example 1 that $\varphi \doteq (y \,\mathsf{W}\, \neg x) \wedge \mathsf{G}(\neg x \supset \mathsf{G}\, \neg y)$ and $\varphi' \doteq \mathsf{G}\, \neg y$ are proper environment specifications. But their conjunction $\varphi \wedge \varphi'$ is not environment coverable. Indeed, the only play that satisfies the conjunction is $(\{\} \cdot \{\})^\omega$. But no environment strategy can enforce this play since the agent can simply do x on its first move.

For the non-closure under U, recall from Example 1 that $\varphi' \doteq x \leftrightarrow y$ is a proper environment specification. Also, $\varphi'' \doteq (\neg x) \supset \mathsf{G}\, y$ is a proper environment specification (as argued in the case for negation, simply replace x with $\neg x$). Consider $\varphi \doteq \varphi'' \,\mathsf{U}\, \varphi'$. Then the play $\pi = \{\} \cdot \{y\} \cdot (\{x\} \cdot \{y\})^\omega$ satisfies φ. But π is not consistent with any environment strategy enforcing φ: Indeed, if the agent doesn't do x on its first move, then the environment can either respond with not doing y on its first move (in order to satisfy φ') or by doing y for all its moves (in order to satisfy φ''). However, if in the later case the agent never does x, the resulting play does not satisfy φ; thus, the environment is forced to respond with not doing y on its first move.

For the non-closure under G, note that $\varphi = y \wedge (x \supset \neg \mathsf{X}\, y)$ is a proper environment specification since every play $\pi = X_0 \cdot Y_0 \cdot X_1 \cdot Y_1 \cdots$ that satisfies φ is consistent with the following environment strategy that enforces φ: The environment does y on its first move; if the agent does x on its first move, the environment does not do y on it second move, and otherwise it does Y_1 on its second move; it does Y_i on its ith move for all $i \geq 2$. Now consider $\varphi' = \mathsf{G}\, \varphi$. Then the play $\pi = (\{\} \cdot \{y\})^\omega$ satisfies φ'. However, φ' is not environment enforceable (since if the agent does x on its first move, then the environment is required to respond on its second move with y and with $\emptyset$). Now use Proposition 3 item 2 applied to φ'. □

3.1 Proper environment core

If E is not environment coverable then it is natural to ask for the largest set of traces contained in E that is environment coverable.

Definition 2. *The* environment coverable core *(aka,* core*) of E, denoted* $\mathrm{core}(E)$*, is the maximum (wrt set containment) subset of E that is environment coverable. If E is also satisfiable, then* $\mathrm{core}(E)$ *is called a* proper environment core.

To see that the core is well-defined (i.e., that a maximum set exists), recall that the environment coverable specifications are closed under arbitrary unions (Proposition 4), and thus the core of E is the union of all its environment coverable subsets. Obviously, E is environment coverable iff $\mathrm{core}(E) = E$.

Remark 2. *One might also ask if there is a minimum (wrt set containment) superset of E that is environment coverable. Unfortunately, such a set does not always exist. Being a minimum, such a set would contain E and be contained in every superset of E that is environment coverable. However, Proposition 4 gives an example where E is not environment coverable, but it is equal to the intersection of two environment coverable specifications E', E''. Thus, there is no minimum superset of E that is environment coverable.*

The following characterization is useful.

Proposition 5. *The core of E is equal to the set of traces consistent with some environment strategy that enforces E, i.e.,* $\mathrm{core}(E) = \mathrm{con}(\mathrm{enf}(E))$.

Proof. To see that $\mathrm{con}(\mathrm{enf}(E)) \subseteq \mathrm{core}(E)$ note that $\mathrm{con}(\mathrm{enf}(E))$ is environment coverable (by Definition 1) and a subset of E. For the reverse containment, note that by Proposition 3 $\mathrm{core}(E) = \mathrm{con}(\mathrm{enf}(\mathrm{core}(E)))$, and that $\mathrm{core}(E) \subseteq E$ implies that $\mathrm{enf}(\mathrm{core}(E)) \subseteq \mathrm{enf}(E)$. Hence, $\mathrm{core}(E) = \mathrm{con}(\mathrm{enf}(\mathrm{core}(E))) \subseteq \mathrm{con}(\mathrm{enf}(E))$. □

Example 2. Say $\mathbf{X} = \{x\}$ and $\mathbf{Y} = \{y\}$. We give the core for some properties shown in Example 1 not to be environment coverable.

1. Consider $E = x$. Since $\mathrm{enf}(E) = \emptyset$, the core of E is empty, expressible by the LTL formula false.
2. Consider $E = (\mathsf{X}\,\mathsf{F}\, x) \vee y$. Since the only environment strategies that enforce E do y on their first move, we have that $\mathrm{core}(E)$ is expressible by the LTL formula y.
3. Consider $E = (\mathsf{F}\, x) \supset (\mathsf{G}\, y)$. Since $\mathrm{enf}(E)$ only consists of the environment strategy that does y for every move, the core E is expressible by the LTL formula $\mathsf{G}\, y$.

We now provide two history-based characterizations: one for $\mathrm{core}(E)$, and one for when E is environment coverable. The following terminology from (Berwanger 2007; Aminof, De Giacomo, and Rubin 2021) is useful.

Definition 3. *Fix a trace property E. Call a history h:*

- winning *if there is an environment strategy σ* winning from *h, i.e., such that h is consistent with σ and every play extending h that is consistent with σ satisfies E;*
- pending *if it is not winning, but there is a play extending h satisfying E;*
- losing *if it is neither winning nor pending, i.e., no play extending h satisfies E.*

We may write, e.g., winning wrt *E to emphasize E.*

The following is a useful characterization of $\mathrm{core}(E)$.

Theorem 6. *Let E be a trace property, and let π be a play. Then $\pi \in \mathrm{core}(E)$ iff: (i) π satisfies E and (ii) every prefix of π is winning wrt E.*

Proof. Say $\pi = X_0 \cdot Y_0 \cdot X_1 \cdot Y_1 \cdots$. If $\pi \in \mathrm{core}(E) = \mathrm{con}(\mathrm{enf}(E))$ (the equality is by Proposition 3), then obviously conditions (i) and (ii) hold. For the converse, suppose (i) and (ii) hold. We construct an environment strategy σ enforcing E with which π is consistent. For $i \geq 0$, let $h_i = X_0 \cdot Y_0 \cdots X_i \cdot Y_i$, and let $h_{-1} = \lambda$ (the empty history). By (ii), for every $i \geq -1$ there is an environment strategy σ_i winning from h_i. Define σ as follows. For every history h ending in an agent move: if h is a prefix of π, say $h = X_0 \cdot Y_0 \cdots X_{i-1} \cdot Y_{i-1} \cdot X_i$ for some $i \geq 0$, then define $\sigma(h) \doteq Y_i$; otherwise, define $\sigma(h) \doteq \sigma_i(h)$, where $i \geq -1$ is such that h_i is the longest common prefix of h and π. Clearly, π is consistent with σ. To see that σ enforces E, note that every trace π' consistent with σ is either equal to π and thus satisfies E by (i); or it satisfies E since it extends h_i and is consistent with the strategy σ_i that is winning from h_i, where h_i is the longest common prefix of π' and π. □

The following characterization says that E is environment coverable iff, no matter what the history is, the environment never needs the help of the agent to achieve E — this is either impossible, or it can be enforced by the environment *regardless of what the agent does*.

Theorem 7. *A trace property E is environment coverable iff no history is pending.*

Proof. Assume that some history h is pending. Hence, there is play π extending h that satisfies E, but since h is not winning then $\pi \notin \mathrm{con}(\mathrm{enf}(E))$. Conclude, by Proposition 3, that E is not environment coverable. Conversely, suppose that no history is pending. If π is a play that satisfies E then, by definition, no prefix of π is losing and thus, every prefix of π is winning. Hence, by Theorem 6, $\pi \in \mathrm{core}(E)$, i.e., $E \subseteq \mathrm{core}(E)$, and so E is environment coverable. □

Corollary 8. *A trace property E is environment coverable iff no history ending in an environment move is pending.*

Proof. The "only if" direction follows immediately. For the "if" direction, assume no history ending in an environment move is pending. We show that every history $h \cdot X$ ending in an agent move is also not pending. Indeed, if h is losing then all its extensions are losing, and if h is winning then a strategy that wins from h also wins from $h \cdot X$. □

We illustrate Theorem 7 with some examples.

Example 3. 1. $E = (\mathsf{X}\,\mathsf{F}\,x) \vee \mathsf{F}\,y$ is environment coverable because every history h is winning: the strategy that responds to h by doing y wins from h.

2. $E = (\mathsf{F}\,x) \supset (\mathsf{G}\,y)$ is not environment coverable because, e.g., the history $h = \{\} \cdot \{\}$ is pending since it is not winning (no environment strategy is winning from h since the agent can eventually do x), but some play extending h satisfies E, e.g., $(\{\} \cdot \{\})^\omega$.

3.2 Discussion of trace-view and strategy-view

Recall from the introduction that there are two views of the agent's assumptions about its environment. In this section we relate the two views: we show that they coincide if one restricts the trace view to proper environment specifications E. We then discuss some anomalies that may arise if one takes the trace view but does not require E to be a proper environment specification. We do this in the context of reactive synthesis. Here G stands for the agent's goal, i.e., the set of traces that the agent is trying satisfy.

In the strategy-view, the agent's assumption about its environment is a non-empty set S of environment strategies. In this case, the reactive synthesis problem asks, given a goal G and such a set S, to find an agent strategy σ_{ag} such that for every environment strategy σ_{env} from S, the resulting interaction $\textsc{play}(\sigma_{\mathsf{ag}}, \sigma_{\mathsf{env}})$ satisfies G. We say that σ_{ag} *enforces G under the assumption S*.

On the other hand, in the trace-view, the agent's assumption about its environment is encapsulated as a set E of traces, and the reactive synthesis problem asks, given a goal G and such a set E, to find an agent strategy σ_{ag} that enforces $E \supset G$, i.e., such that every play consistent with σ_{ag} and satisfies E also satisfies G.

What is the relationship between these two notions?

1. On the one hand, the strategy view with a (non-empty) set of strategies S can be seen as a special case of the trace view — restricted to proper environment specifications— by taking $E = \mathrm{con}(S)$. Indeed, it is not hard to see that the set $\mathrm{con}(S)$ is a proper environment specification, and that a strategy σ_{ag} enforces G under the assumption S iff it enforces $\mathrm{con}(S) \supset G$.
2. On the other hand, if E is a proper environment specification, then the trace view with E can be seen as a special case of the strategy view by taking $S = \mathrm{enf}(E)$. To see this, first note that $E = \mathrm{con}(\mathrm{enf}(E))$ by Proposition 3. Then, observe that, by definition, an agent strategy σ_{ag} realizes $\mathrm{con}(\mathrm{enf}(E)) \supset G$ iff for every play π consistent with σ_{ag} we have that: if π satisfies $\mathrm{con}(\mathrm{enf}(E))$ then it also satisfies G. Finally, observe that this is equivalent to σ_{ag} enforces G under the assumption $\mathrm{enf}(E)$.

In summary, the strategy view coincides with the trace view restricted to proper environment specifications.

We remark that the standard definition of the strategy-view is usually given with respect to any environment enforceable trace property E as follows (Aminof et al. 2019): given E and G, say that σ_{ag} enforces G under the assumption E iff for every $\sigma_{\mathsf{env}} \in \mathrm{enf}(E)$ we have that the play $\textsc{play}(\sigma_{\mathsf{ag}}, \sigma_{\mathsf{env}})$ satisfies G. In other words, the standard definition implicitly substitutes the set of traces E with the set of strategies that enforce E. As noted in item 2 above, if E is a proper environment specification then this substitution simply transforms the trace view with respect to E to an equivalent strategy view. However, if E is not a proper environment specification, then this substitution amounts to *switching to a strategy view that is equivalent to the trace view for* $\mathrm{core}(E)$, since for such E we do not have that $E = \mathrm{con}(\mathrm{enf}(E))$, but rather (by Proposition 5) that $\mathrm{core}(E) = \mathrm{con}(\mathrm{enf}(E))$.

We now discuss some anomalies that arise when taking the trace-view without requiring E to be a proper environment specification. Intuitively, the anomalies arise since the trace view considers every trace in E as possible, while the strategy-view only considers traces in $\mathrm{core}(E)$ as possible.

One well-known anomaly is that if the environment cannot enforce E then a synthesis algorithm may return an agent strategy that enforces $E \supset G$ by sometimes, or even always, falsifying E instead of satisfying G. This may obviously be undesirable. For instance, if the goal G consists of traces that have the agent drive a car from one city to another, and the assumption E that the agent has about the environment consists of traces where the car does not crash, then the agent can enforce $E \supset G$ by simply crashing the car into a wall. This anomaly has been ameliorated in a variety of different ways, including requiring that the set E be environment enforceable, e.g., (Bloem et al. 2014; Aminof et al. 2018; Camacho, Bienvenu, and McIlraith 2018).

However, even if, in order to avoid this anomaly, we require that E be environment enforceable — and the agent assumes that the environment uses a strategy enforcing E (otherwise, what is the meaning of requiring that E be environment enforceable?) — synthesizing for $E \supset G$ presents

a new anomaly. Namely, it implicitly considers every trace that satisfies E to be possible, even though that trace may not be consistent with any environment strategy enforcing E (i.e., may not be in $\mathrm{core}(E)$), and thus, by the agent's assumption, is in fact impossible!

How serious is this anomaly? It turns out that for classic reactive synthesis it does not pose a practical problem since it does not prevent the synthesis algorithm from producing an agent strategy just because it considers as possible more traces than it should. This follows from the following "equi-realizability" result: if E is environment enforceable, then there is an agent strategy that enforces $E \supset G$ iff there is an agent strategy that enforces G under the assumption $\mathrm{enf}(E)$ (Aminof et al. 2019). However, if one steps out of the comfort zone of this setting then — unless similar extenuating circumstances, like the equi-realizability result above, can be shown to exist — the trace-view may prove inadequate: supplying no solutions, or only supplying anomalous ones, in cases where the strategy-view supplies meaningful solutions. We illustrate this with two examples.

First, the scenario in (Aminof, De Giacomo, and Rubin 2021, Section 6) shows that when taking the trace-view to synthesize a best-effort strategy (because, e.g., there are no enforcing strategies), one can get anomalous solution strategies that are guaranteed to never achieve the goal G on any trace that satisfies E! (the strategy-view rules these out as solutions). Looking at that example one sees that the reason for that is that the environment specification is not environment coverable (even though it is environment enforceable). Second, consider the case of synthesizing enforcing strategies under partial observability. The next example provides a set E that is not a proper environment specification (but is environment enforceable), and a goal G, where the trace-view does not supply a solution, but the strategy-view does.

Example 4. Let $\mathbf{X} = \{x\}, \mathbf{Y} = \{y\}$, and consider $E = y \vee \neg(y \leftrightarrow \mathsf{X}\, x)$ and $G = y \leftrightarrow \mathsf{X}\, x$. Informally, the agent's goal G says to do x on its second move iff the environment did y on its first move ("matching pennies"). Suppose that the environment's moves are hidden from the agent, and thus the agent's strategy is simply a function that maps the current time step to a move. Observe that no agent strategy can realize $E \supset G$. Indeed, if the agent does x (resp. $\neg x$) as its second move, then $E \wedge \neg G$ holds if the environment does $\neg y$ (resp. y) on its first move. On the other hand, the only environment strategies that enforce E do y on their first move (since, at this time, they do not have access to the agent's second move). And thus, any agent strategy that does x in the second move enforces G under the assumption E.

4 Algorithmic problems

In this section we discuss the following natural algorithmic problems. Given a trace property E, represented by an LTL formula φ or a DPA $\mathcal{A}$:

Problem 1: Decide if E is environment coverable.

Problem 2: Compute a representation of $\mathrm{core}(E)$.

Problem 3: Compute a representation of a trace that satisfies E but not $\mathrm{core}(E)$, if there is one (i.e., a trace witnessing that E is not a proper environment specification).

We now give an overview of our algorithms for solving these problems. When starting with φ, we take an automata-theoretic approach and start by translating φ into an equivalent deterministic automaton $\mathcal{A}$.

For Problem 1, we use the characterization in Corollary 8. Since the states of $\mathcal{A}$ represent equivalent histories, it is sufficient to determine if there are any reachable pending states, which we do by computing the winning regions of 2-player and 1-player games on $\mathcal{A}$, as in (Aminof, De Giacomo, and Rubin 2021).

For Problem 2, we compute the winning region of 2-player game on $\mathcal{A}$, then make all states not in the winning region into rejecting sinks. The language of the resulting automaton $\mathcal{A}'$ is the core of $L(\mathcal{A})$. In case $\mathcal{A}$ is equivalent to an LTL formula φ, by Theorem 2 it can be taken to be counter-free, and the translation from $\mathcal{A}$ to $\mathcal{A}'$ preserves this. Hence, $\mathcal{A}'$ can be transformed into an equivalent LTL formula. This gives a representation of the core of φ as an LTL formula. We note that it is not a priori obvious that the core of an LTL formula is LTL definable.

For Problem 3, we use a solution of Problem 2 to produce a witnessing trace (if there is one). The trace is ultimately periodic and finitely represented.

4.1 Algorithms for DPA specifications

In this section we assume that the specification is given by a DPA $\mathcal{A} = (D, c)$ over the alphabet $\Sigma = 2^{\mathbf{X} \cup \mathbf{Y}}$. The following Algorithm decides if $L(\mathcal{A})$ is environment coverable.

Algorithm 1: Let $\mathcal{A} = (D, c)$ be a DPA with $D = (2^{\mathbf{X} \cup \mathbf{Y}}, Q, \iota, \delta)$. Decide if $L(\mathcal{A})$ is environment coverable.

1. W.l.o.g., we assume that every state in $\mathcal{A}$ is reachable (otherwise restrict $\mathcal{A}$ to its reachable states).
2. Solve the DPA-game (D, c), and compute the winning region $W \subseteq Q$ for the environment.
3. Solve the cooperative DPA-game (D, c) and compute the co-operative winning region $W' \subseteq Q$.
4. Output "$L(\mathcal{A})$ is environment coverable" iff $W = W'$.

Theorem 9. *Given a DPA $\mathcal{A}$, we can decide whether $L(\mathcal{A})$ is environment coverable in time polynomial in the size, and exponential in the index, of $\mathcal{A}$.*

Proof. For a history h ending in an environment move, let *state*(h) be the state reached by $\mathcal{A}$ starting in the initial state and reading h. Correctness of Algorithm 1 easily follows from Corollary 8, and the following two claims:

1. h is not losing wrt $L(\mathcal{A})$ iff *state*$(h) \in W'$ — this follows from the definitions;
2. h is winning wrt $L(\mathcal{A})$ iff *state*$(h) \in W$.

We prove 2. Assume h is winning, and let σ be a strategy winning from h. Observe that the strategy σ' defined by $\sigma'(h') = \sigma(h \cdot h')$ wins the DPA game from *state*(h). Conversely, let σ' be a strategy that wins the DPA game from *state*(h). Define the strategy σ to follow h and then to mimic

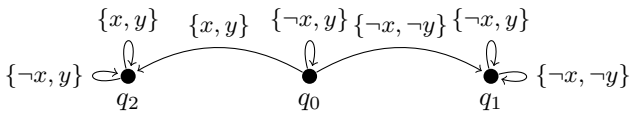

Figure 1: Automaton from Example 5

σ'. Formally, for a history h': if h' is a proper prefix of h, then $\sigma(h') = a$ where a is the next symbol on h following h'; if h' is a proper extension of h, say $h' = h \cdot h''$, then $\sigma(h') = \sigma'(h'')$; otherwise, $\sigma'(h')$ is defined arbitrarily. Obviously, h is consistent with σ'. Let $\pi = h\zeta$ be any play consistent with σ that extends h, and observe that ζ is consistent with σ', and thus the run ρ' of $\mathcal{A}$ on ζ (starting in *state*(h)) satisfies the acceptance condition c. The run ρ of $\mathcal{A}$ on π is the concatenation of the run of $\mathcal{A}$ on h (starting at the initial state and ending in *state*(h)) with the suffix of ρ' (without its first state *state*(h)). Since the acceptance condition is prefix independent, also ρ satisfies it, i.e., $\pi \in L(\mathcal{A})$. □

Example 5. Figure 1 shows the DPA for the LTL formula $\varphi = (\mathsf{F}\, x) \supset (\mathsf{G}\, y)$ — all shown states have color 0, missing transitions go to a rejecting sink state (not shown) with color 1. Note that the winning region W is $\{q_0, q_2\}$, and the co-operative winning region W' is $\{q_0, q_1, q_2\}$. Thus, Algorithm 1 returns that φ is not environment coverable.

For Problem 2, the following algorithm computes a DPA representation of $\mathrm{core}(L(\mathcal{A}))$ for a given DPA $\mathcal{A}$.

Algorithm 2: Let $\mathcal{A} = (D, c)$ be a DPA with $D = (2^{\mathbf{X}\cup\mathbf{Y}}, Q, \iota, \delta)$. Compute a representation of $\mathrm{core}(L(\mathcal{A}))$.

1. Solve the DPA-game (D, c) and compute the winning region $W \subseteq Q$ of the environment.
2. Obtain a DPA $\mathcal{A}' = (D', c')$ by removing all states that do not belong to W and redirecting all transitions that leave W to a fresh rejecting *sink* state. Formally, $D' = (2^{\mathbf{X}\cup\mathbf{Y}}, W \cup \{sink\}, \iota', \delta')$ where: $\iota' \doteq \iota$ if $\iota \in W$, and $\iota' \doteq sink$ otherwise; $\delta'(q, a) \doteq \delta(q, a)$ if $q \in W$ and $\delta(q, a) \in W$, and $\delta(q, a) \doteq sink$ otherwise; Finally, let $c'(q) = c(q)$ for $q \in W$, and $c(sink) = 1$.

Theorem 10. *Given a DPA $\mathcal{A}$, one can compute a DPA $\mathcal{A}'$ with $L(\mathcal{A}') = \mathrm{core}(L(\mathcal{A}))$ in time polynomial in the size, and exponential in the index, of $\mathcal{A}$. Moreover, the size (resp. index) of $\mathcal{A}'$ is bounded by 1 plus the size (resp. index) of $\mathcal{A}$.*

Proof. Let $\mathcal{A}' = (D', c')$ be the DPA returned by Algorithm 2. We prove that $\mathrm{core}(L(\mathcal{A})) = L(\mathcal{A}')$: it is easy to verify that, by the definition of $\mathcal{A}'$, there are no pending histories for $L(\mathcal{A}')$. Hence, by Theorem 7, $L(\mathcal{A}')$ is environment coverable. $L(\mathcal{A}') \subseteq L(\mathcal{A})$ by the definition of $\mathcal{A}'$ and thus, by the definition of core, $L(\mathcal{A}') \subseteq \mathrm{core}(L(\mathcal{A}))$. For the reverse inclusion, given a trace $\tau \in \mathrm{core}(L(\mathcal{A}))$, apply Proposition 3 item 2 to $\mathrm{core}(L(\mathcal{A}))$ to obtain a strategy σ enforcing $L(\mathcal{A})$ with which τ is consistent. It follows that the run of $\mathcal{A}$ on any trace consistent with σ (and in particular its run r on τ) satisfies c and only visits states in W. It follows that r is also the run of $\mathcal{A}'$ on τ and, since c' agrees with c on states in W, conclude that $\tau \in L(\mathcal{A}')$. □

Remark 3. *The algorithm for computing $\mathrm{core}(L(\mathcal{A}))$ provides an alternative algorithm for deciding whether $L(\mathcal{A})$ is environment coverable: first compute $\mathcal{A}' = \mathrm{core}(L(\mathcal{A}))$, and then check the language equivalence $L(\mathcal{A}') = L(\mathcal{A})$. However, computing the winning region of the cooperative game is easier than deciding language equivalence, and thus our earlier algorithm is preferable.*

For Problem 3, since we can obtain from Algorithm 2 a DPA for $\mathrm{core}(E)$, standard closure properties of DPAs yield:

Proposition 11. *Given a DPA $\mathcal{A}$, let $E = L(\mathcal{A})$. We can compute a DPA whose language is $E \setminus \mathrm{core}(E)$, and, in case of $E \neq \mathrm{core}(E)$, some trace $\tau = \tau_1(\tau_2)^\omega \in E \setminus \mathrm{core}(E)$; both computations can be done in time polynomial in the size, and exponential in the index, of $\mathcal{A}$.*

4.2 Algorithms for LTL specifications

In this section we assume that the specification is given by an LTL formula φ. For Problem 1 (deciding if φ is environment coverable) the algorithm extends the earlier algorithm for DPAs, by a pre-processing step: first build the DPA $\mathcal{A}_\varphi$ which accepts exactly the models of φ, and then apply Algorithm 1. This gives the following result:

Theorem 12. *Given an LTL formula φ, we can decide in double-exponential time in $|\varphi|$ whether φ is environment coverable.*

Proof. By Theorem 1, the size (resp. index) of $\mathcal{A}_\varphi$ is at most 2EXP (resp. EXP) in $|\varphi|$. Conclude using Theorem 9. □

For Problem 2, computing a representation of the core of a given LTL formula φ, we provide two results.

1. We give an algorithm that computes a DPA for $\mathrm{core}(\varphi)$, a DPA for $L(\varphi) \setminus \mathrm{core}(\varphi)$, and a trace that witnesses $L(\varphi) \neq \mathrm{core}(\varphi)$ in case there is such a trace.
2. We prove that $\mathrm{core}(\varphi)$ is in fact representable by an LTL formula, demonstrating the LTL formulas are closed under the operation of taking the core! The procedure used in the proof is effective, however, maybe not optimal. We leave it as an open problem to find a better procedure.

The algorithms for the first result extend the earlier algorithms for DPAs, by a pre-processing step: first build the DPA $\mathcal{A}_\varphi$, which accepts exactly the models of φ, then apply the algorithms from Section 4.1. We obtain the following:

Theorem 13. *For an LTL formula φ, we can compute a DPA $\mathcal{A}$ such that $L(\mathcal{A}) = \mathrm{core}(\varphi)$. In addition, we can compute a DPA $\mathcal{A}'$ with $L(\mathcal{A}') = L(\varphi) \setminus \mathrm{core}(\varphi)$, and, in case $L(\varphi) \neq \mathrm{core}(\varphi)$, some trace $\tau = \tau_1(\tau_2)^\omega \in L(\varphi) \setminus \mathrm{core}(\varphi)$. This can all be done in double-exponential time in $|\varphi|$.*

Proof. By Theorem 1, the size (resp. index) of $\mathcal{A}_\varphi$ is at most double-exponential (resp. exponential) in $|\varphi|$. We then conclude by applying Theorem 10 and Proposition 11. □

We now come to the problem of computing an LTL formula for $\mathrm{core}(\varphi)$. We note that it is *a priori* unclear whether $\mathrm{core}(\varphi)$ can be represented as an LTL formula. We prove so in the following. The proof makes use of Theorem 2, and thus begins by constructing a deterministic *counter-free* Rabin automata.

Theorem 14. *Given an LTL formula φ, the core of φ is itself LTL definable, and an LTL formula for $\mathrm{core}(\varphi)$ can be computed from φ.*

Proof. By Theorem 2, construct a counter-free DRA $\mathcal{A}_\varphi$ with $L(\mathcal{A}_\varphi) = L(\varphi)$. The definitions and algorithms of Section 4.1 also work, with trivial changes, when DPA are replaced by DRA. In particular, we apply the algorithm from Section 4.1 for extracting the core of $\mathcal{A}_\varphi$. That is, we consider the DRA-game played by the agent and the environment on $\mathcal{A}_\varphi = (D, \Omega)$, where $D = (\Sigma, Q, \iota, \delta)$ is a deterministic transition system and $\Omega = \{(B_1, G_1), \ldots, (B_k, G_k)\}$ is a Rabin acceptance condition. We solve the DRA-game and compute the winning region $W \subseteq Q$ of the environment. We now obtain the DRA $\mathcal{A}' = (D', \Omega')$, by removing all states not in W and redirecting all transitions that leave W to a fresh rejecting sink state, called $sink$. Formally, the deterministic transition system D' is defined as in Algorithm 2, and the acceptance condition is defined by setting $\Omega' = \{(B'_1, G'_1), \ldots, (B'_k, G'_k)\}$, where $B'_i = B_i \cap W$ and $G'_i = G_i \cap W$. By the same proof as for Theorem 10, we obtain that $L(\mathcal{A}') = \mathrm{core}(L(\mathcal{A}_\varphi))$.

We now show that D' is counter-free. For every word w, and every $k \geq 1$, we have the following. Let $q \in W$, and assume that the run r of D' from q reading w^k ends back in q. Hence, (*) the run r (and thus also its prefix run from q on w) does not leave W (since otherwise it would not return to q). Consequently, r is also the run of D from q reading w^k. Since we assumed that D is counter-free we conclude that D returns from q to q reading w, which together with (*) implies that D' also returns from q to q reading w. For the case $q = sink$, obviously D' returns from q to q reading any word. Combining the last two observations we get that $\mathcal{A}'$ is counter-free. Hence, by Theorem 2, there in an LTL formula ψ with $L(\psi) = L(\mathcal{A}') = \mathrm{core}(L(\mathcal{A}_\varphi)) = \mathrm{core}(L(\varphi))$. □

5 Lower bounds

We provide lower bounds for the decision problem of whether a given trace property, given by an LTL formula φ, is environment coverable. We first give an auxiliary result:

Proposition 15. *Deciding if a given LTL formula of the form $\mathsf{F}\,\psi$ is environment enforceable is 2EXPTIME-hard.*[2]

Proof. We will reduce from the fact that deciding if a given LTL formula φ is environment enforceable is 2EXPTIME-hard (Pnueli and Rosner 1990). Say $AP = \mathbf{X} \cup \mathbf{Y}$ and let p be a new variable. So, given an LTL formula φ over AP, define $\psi \doteq \varphi \wedge p \wedge \mathsf{X}\,\mathsf{G}\,\neg p$, $\mathbf{X}' = \mathbf{X}$, and $\mathbf{Y}' = \mathbf{Y} \cup \{p\}$ (thus, the environment controls p). We now show that φ is environment enforceable iff $\mathsf{F}\,\psi$ is environment enforceable. Clearly, if φ is environment enforceable, then so is $\mathsf{F}\,\psi$ (since every strategy that enforces φ also enforces $\mathsf{F}\,\psi$ if the it additionally does p in the first round, and then $\neg p$ for all later rounds).

Conversely, take a strategy σ that enforces $\mathsf{F}\,\psi$. We first prove that: (*) there is a history h consistent with σ such that for any trace $\tau = h \cdot \xi$ that is consistent with σ, we have that $p \in \xi_0$ and $p \notin \xi_i$ for all $i > 0$. Take a history h consistent with σ such that $p \in \sigma(h)$. Such a history must exist because σ enforces $\mathsf{F}\,\psi$. Now, we check if there is a history h' consistent with σ that strictly extends h such that $p \in \sigma(h')$. If so, we set $h = h'$ and iterate this procedure. We claim that after a finite number of steps we must have found a history h such that no history h' consistent with σ that strictly extends h satisfies $p \in \sigma(h')$. Indeed, if this is not the case then the sequence of histories above induces an infinite trace $\tau = z_0 z_1 \cdots$ consistent with σ such that $p \in z_i$ for infinitely many $i \geq 0$. However, this is a contradiction to the assumption that σ enforces $\mathsf{F}\,\psi$. Hence, (*) holds. Now, we define a strategy σ' that enforces φ as follows: for every history h' ending in an agent move, let $\sigma'(h') \doteq \sigma(h \cdot h') \setminus \{p\}$. It is not hard to see that, by (*) and the assumption that σ enforces $\mathsf{F}(\varphi \wedge p \wedge \mathsf{X}\,\mathsf{G}\,\neg p)$, σ' indeed enforces φ. □

We can now give the promised lower bound.

Theorem 16. *Checking whether a given LTL formula φ is environment coverable is 2EXPTIME-hard.*

Proof. We will reduce from the problem in Proposition 15. Say $AP = \mathbf{X} \cup \mathbf{Y}$ and let q be a new variable. Given an LTL formula of the form $\mathsf{F}\,\psi$, define $\varphi \doteq \mathsf{F}\,\psi \vee q$, $\mathbf{X}' = \mathbf{X} \cup \{q\}$ (so q is controlled by the agent), and $\mathbf{Y}' = \mathbf{Y}$. We show that $\mathsf{F}\,\psi$ is environment enforceable iff φ is environment coverable. First, assume that $\mathsf{F}\,\psi$ is environment enforceable. Thus, every history h is winning wrt $\mathsf{F}\,\psi$ (indeed, have the environment strategy follow h as long as possible, and once this is no longer possible, either because the agent left h or all of h has been seen, mimic the given strategy that enforces $\mathsf{F}\,\psi$). Hence, also every history is winning wrt φ, so φ is environment coverable by Theorem 7.

Conversely, suppose $\mathsf{F}\,\psi$ is not environment enforceable. Then, the empty history must be losing or pending wrt $\mathsf{F}\,\psi$. If it is losing wrt $\mathsf{F}\,\psi$ then it is pending wrt φ (since the agent can do q on its first move). If it is pending wrt $\mathsf{F}\,\psi$ then it is pending wrt φ (it is not losing since the agent can do q on its first move, and it is not winning since then the environment could use that strategy to win $\mathsf{F}\,\varphi$ because q is not mentioned in φ). Either way, the empty string is pending wrt φ. So, φ is not environment coverable by Theorem 7. □

We remark that similar proofs can be given for specifications given by DPA instead of LTL. Here, we get that checking whether a trace property given by a DPA is environment coverable is at least as hard as deciding the winner of DPA-games.

6 Outlook

The standard approach to synthesis is, given sets E, G of traces (specified, e.g., in LTL), to find an agent strategy that

[2] An almost identical proof shows that deciding if such a formula is agent enforceable is also 2EXPTIME-hard.

enforces the implication $E \supset G$. Although this trace-centric view of synthesis is common in assume-guarantee reasoning in the formal methods literature, it is not satisfactory *for all applications* for the following reasons. First, E here is not a real specification of the environment, but rather it is an assumption that the agent can use to achieve its goal G. This conflicts with the view, common in planning and reasoning about actions, that the agent has a model of the world, i.e., a model of its environment. Second, if E is not environment enforceable, well-known anomalies arise (Section 3.2).

The strategy-centric view was identified to address these issues (Aminof et al. 2018; Aminof et al. 2019). In this view: E is considered as a specification of the strategies that the environment can choose, i.e., those environment strategies that enforce E; the caveat that E should be enforceable becomes a natural consistency condition, i.e., there must be at least one enforcing environment strategy for E; the equi-realizability theorem for classical reactive synthesis still allows one to solve synthesis, i.e., to find an agent strategy that satisfies G against every environment strategy enforcing E, by instead finding an agent strategy that enforces $E \supset G$.

In this paper we show how to relate both these views. That is, given a trace property E, we show that not all traces that satisfy E are relevant when using E to specify the environment strategies that enforce E. In fact, only the set $\mathrm{core}(E)$ of traces is relevant. This means that E contains some spurious traces that are not relevant for the specification of E, and if we get rid of these spurious traces we get the essence of the specification, i.e., $\mathrm{core}(E)$. Thus, an agent strategy satisfies G against every environment strategy that enforces E iff it enforces $\mathrm{core}(E) \supset G$.

Finally, this paper provides an explanation of why, despite the problems mentioned above, the trace-centric view has proved successful in many settings: it is either because E is a proper environment specification (e.g., in nondeterministic planning), or since $E \supset G$ and $\mathrm{core}(E) \supset G$ are equi-realizable (as in strategic reasoning in the standard setting). In such cases we do not need to compute $\mathrm{core}(E)$ and can work with E instead. However, as one advances to more sophisticated settings such as best-effort synthesis, partial observability, etc., the situation changes.

Acknowledgments

This work is partially supported by the ERC Advanced Grant WhiteMech (No.834228) and by PRIN project RIPER (No.20203FFYLK). We thank the reviewers for their careful reading.

References

Alur, R.; Moarref, S.; and Topcu, U. 2013. Counter-strategy guided refinement of GR(1) temporal logic specifications. In *FMCAD*.

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.; Rubin, S.; and Zuleger, F. 2022. Beyond strong-cyclic: Doing your best in stochastic environments. In *IJCAI*.

Aminof, B.; De Giacomo, G.; Rubin, S.; and Zuleger, F. 2023. Stochastic best-effort strategies for Borel goals. In *LICS*.

Aminof, B.; De Giacomo, G.; and Rubin, S. 2021. Best-effort synthesis: Doing your best is not harder than giving up. In *IJCAI*.

Aminof, B.; De Giacomo, G.; and Rubin, S. 2023. Reactive synthesis of dominant strategies. In *AAAI*.

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).

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

Bloem, R.; Jobstmann, B.; Piterman, N.; Pnueli, A.; and Sa'ar, Y. 2012. Synthesis of reactive(1) designs. *J. Comput. Syst. Sci.* 78(3):911–938.

Bloem, R.; Ehlers, R.; Jacobs, S.; and Könighofer, R. 2014. How to handle assumptions in synthesis. In *SYNT*.

Boker, U.; Lehtinen, K.; and Sickert, S. 2022. On the translation of automata to linear temporal logic. In *FOSSACS*.

Bonet, B., and Geffner, H. 2020. Qualitative numeric planning: Reductions and complexity. *J. Artif. Intell. Res.* 69:923–961.

Bonet, B.; De Giacomo, G.; Geffner, H.; and Rubin, S. 2017. Generalized planning: Non-deterministic abstractions and trajectory constraints. In *IJCAI*.

Brenguier, R.; Raskin, J.; and Sassolas, M. 2014. The complexity of admissibility in omega-regular games. In *CSL-LICS*.

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

Camacho, A.; Bienvenu, M.; and McIlraith, S. A. 2019. Towards a unified view of AI planning and reactive synthesis. In *ICAPS*.

Diekert, V., and Gastin, P. 2008. First-order definable languages. In *Logic and Automata: History and Perspectives [in Honor of Wolfgang Thomas]*, volume 2 of *Texts in Logic and Games*, 261–306.

D'Ippolito, N.; Braberman, V. A.; Piterman, N.; and Uchitel, S. 2013. Synthesizing nonanomalous event-based controllers for liveness goals. *ACM Trans. Softw. Eng. Methodol.* 22(1):9:1–9:36.

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

Gerevini, A., and Long, D. 2005. Plan constraints and preferences in PDDL3. Technical report, Technical Report 2005-08-07, Department of Electronics for Automation.

Haslum, P.; Lipovetzky, N.; Magazzeni, D.; and Muise, C. 2019. *An Introduction to the Planning Domain Definition Language*. Springer.

Klein, U., and Pnueli, A. 2010. Revisiting synthesis of GR(1) specifications. In *HVC*.

Kress-Gazit, H.; Lahijanian, M.; and Raman, V. 2018. Synthesis for robots: Guarantees and feedback for robot behavior. *Annu. Rev. Control. Robotics Auton. Syst.* 1:211–236.

Li, W.; Dworkin, L.; and Seshia, S. A. 2011. Mining assumptions for synthesis. In *MEMOCODE*.

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

Pnueli, A., and Rosner, R. 1990. Distributed reactive systems are hard to synthesize. In *FOCS*.

Rodriguez, I. D.; Bonet, B.; Sardiña, S.; and Geffner, H. 2022. Flexible FOND planning with explicit fairness assumptions. *J. Artif. Intell. Res.* 74.

Thomas, W. 1981. A combinatorial approach to the theory of omega-automata. *Inf. Control.* 48(3):261–283.

Vardi, M. Y. 1995. An automata-theoretic approach to linear temporal logic. In *Logics for Concurrency – Structure versus Automata (Banff Higher Order Workshop)*.