Synthesizing Strategies under Expected and Exceptional Environment Behaviors

Benjamin Aminof[1] , **Giuseppe De Giacomo**[2] , **Alessio Lomuscio**[3] ,
Aniello Murano[4] and **Sasha Rubin**[5]
[1] TU Vienna, Austria,
[2] University of Rome "La Sapienza", Italy,
[3] Imperial College London, UK,
[4] University of Naples "Federico II", Italy,
[5] University of Sydney, Australia

Abstract

We consider an agent that operates with two models of the environment: one that captures expected behaviors and one that captures additional exceptional behaviors. We study the problem of synthesizing agent strategies that enforce a goal against environments operating as expected while also making a best effort against exceptional environment behaviors. We formalize these concepts in the context of linear-temporal logic, and give an algorithm for solving this problem. We also show that there is no trade-off between enforcing the goal under the expected environment specification and making a best-effort for it under the exceptional one.

1 Introduction

Synthesis, originally developed in the context of reactive systems, is concerned with the automatic generation of behaviors, or *strategies*, from requirements. In its usual form, the requirements are given as temporal specifications, and the resulting behaviors are returned in the form of automata [Pnueli and Rosner, 1989; Finkbeiner, 2016]. Applying and developing concepts related to synthesis in an AI context poses considerable challenges compared with the traditional setting, including the presence of multiple agents, noise, normative requirements, and fairness constraints [Daniele *et al.*, 1999; De Giacomo *et al.*, 2013; D'Ippolito *et al.*, 2018; Belardinelli *et al.*, 2017; Bonet *et al.*, 2017; Maubert and Murano, 2018; Belle, 2018; Aminof *et al.*, 2020].

A crucial aspect that all the work on synthesis in AI has in common is the need for a *model of the environment* in which the agent acts. In planning, for example, such a model is given in terms of a "domain" which specifies, in each state, how the environment enacts the (possibly nondeterministic) effects of the action performed by agent [Geffner and Bonet, 2013; Reiter, 2001]. This should be contrasted with traditional synthesis where one does not make an actual distinction between the agent specification, i.e., the goal, and the environment specification [Pnueli and Rosner, 1989]. Synthesizing with a model of the environment is related to synthesis under assumptions as studied in the Verification literature [Chatterjee and Henzinger, 2007; Bloem *et al.*, 2015; Brenguier *et al.*, 2017] and, building on that literature, it has been satisfactorily addressed in [Camacho *et al.*, 2018; Aminof *et al.*, 2018] and especially in [Aminof *et al.*, 2019a].

Following [Aminof *et al.*, 2019a], the environment is specified in terms of possible strategies that it could enact, and is formalized through a linear temporal logic specification $\mathcal{E}$. In this way we are able to extend planning domains in many ways, e.g., by allowing for the possibility of having deferred effects, or introducing fairness or stability conditions on effects. The agent goal is also expressed as a linear temporal logic specification φ, much as is done in planning for temporally extended goals [Bacchus and Kabanza, 1996; Bacchus and Kabanza, 2000; De Giacomo and Vardi, 1999; De Giacomo and Rubin, 2018; Camacho *et al.*, 2018]. Then, synthesis under environment specifications is the problem of finding a *winning strategy* for the agent, which enforces the goal φ no matter what strategy, among those specified by $\mathcal{E}$, the environment happens to follow.

While this strategy-based approach to modeling environments is well suited for capturing synthesis in AI, it suffers from one severe limitation: the environment model is assumed to be given in a "*monolithic way*", i.e., with a single specification, and without distinguishing *expected* behaviors (i.e., strategies) and *exceptional* or uncommon variations, such as rare faults or failures. As a consequence, the agent is meant to synthesise a winning strategy against all possible environment behaviors, independently of them being expected or exceptional. In real situations however, coming up with a *suitable monolithic* model of the environment may be unfeasible. If the model only captures the expected behaviors of the environment, then a winning strategy for the agent may be ineffective, since it may be losing against exceptional behaviors not captured by the model. On the other hand, if the model also includes possible exceptions, such as faults, failures, or non-compliance with the expected behaviors, there may simply not exist a winning strategy for the agent since handling all possible exceptions may be impossible.

In this paper we abandon a single monolithic model of the environment in favor of *two distinct models*, i.e., two environment specifications: one capturing the *expected behaviors*, and one that also includes the *exceptional behaviors*. Using

these two models we formally show how to appropriately and simultaneously handle expected and exceptional behaviors.

The central tenet of our contribution is that, in such situations, an agent should use a winning strategy against the expected environment behaviors, that, *in addition*, also provides a satisfactory response, *as far as possible*, to the exceptional behaviors of the environment. Such a "best-effort" strategy may not be sufficient to realize the agent specification against *all* possible exceptions, but it does constitute rational agent behavior: on the one hand, it is guaranteed to be winning against the expected behaviors; and on the other hand, it also does not miss any opportunity to be winning against exceptional environment behaviors. We make these intuitions precise by formulating "as far as possible" as requiring the agent strategy to be also winning against a *maximal set* of exceptional environment behaviors, where "maximal" is induced by the game-theoretic notion of (weak)-dominance order on agent strategies. Notably, we show that if there is a winning strategy against the expected-environment specification then there is one that, in addition, is making a best effort against the exceptional-environment specification. Moreover, we show that one can compute such a strategy.

2 Preliminaries

We fix the notation for concepts that will be used in the rest of the paper. We follow the notation in [Aminof *et al.*, 2019a].

Linear-time Temporal Logic (LTL). Given a finite set AP of atomic propositions, *formulas of* LTL *over* AP are defined by the following BNF (where $p \in AP$): $\varphi ::= p \mid \varphi \vee \varphi \mid \neg\varphi \mid \mathsf{X}\varphi \mid \varphi \,\mathsf{U}\, \varphi$. The *size* $|\varphi|$ of a formula φ is the number of symbols in it. A *trace* τ is an infinite sequence of valuations of the atoms, i.e., $\tau \in (2^{AP})^\omega$. For a non-negative integer n, write τ_n for the valuation at position n (we index starting with $n = 0$). A history h is a finite prefix of a trace; its length is denoted by $|h|$. 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 $(\tau, n) \models \varphi$; $(\tau, n) \models \mathsf{X}\,\psi$ iff $(\tau, n+1) \models \psi$; and $(\tau, n) \models \varphi_1 \,\mathsf{U}\, \varphi_2$ iff there exists $m \geq n$ such that: $(\tau, m) \models \varphi_2$ and $(\tau, j) \models \varphi_1$ for all $n \leq j < m$. We write $\tau \models \varphi$ if $(\tau, 0) \models \varphi$ to represent that τ *satisfies* φ and that τ is a *model* of φ. We use the usual abbreviations, $\varphi \supset \varphi' \doteq \neg\varphi \vee \varphi'$, $\mathsf{true} \doteq p \vee \neg p$, $\mathsf{false} \doteq \neg\mathsf{true}$, $\mathsf{F}\,\varphi \doteq \mathsf{true}\,\mathsf{U}\,\varphi$, $\mathsf{G}\,\varphi \doteq \neg\,\mathsf{F}\,\neg\varphi$. We write $\models \psi$ when ψ is *logically valid*, i.e., $\tau \models \psi$ for all traces τ.

Reactive synthesis. (Reactive) synthesis concerns constructing the behaviours of an agent that achieves a given goal while interacting with its environment. We specify goals as well as environment specifications by LTL formulas. Formally, let X and Y be disjoint finite sets of Boolean variables, called the *environment variables* and *agent variables* respectively, and let $AP \doteq X \cup Y$. A trace is then written $(X_0 \cup Y_0)(X_1 \cup Y_1)\ldots$ where $X_i \subseteq X, Y_i \subseteq Y$ for all i. An *agent strategy* is a function $\sigma_{\mathsf{ag}} : (2^X)^+ \to 2^Y$ that maps a non-empty sequence of valuations of the environment variables to the next valuation of the agent variables. Similarly, an *environment strategy* is a function $\sigma_{\mathsf{env}} : (2^Y)^* \to 2^X$. Note that an environment strategy can accept the empty sequence λ since the environment makes the first move. We denote by $\Sigma^{all}_{\mathsf{ag}}$ (resp. $\Sigma^{all}_{\mathsf{env}}$) the set of all agent (resp. environment) strategies. A trace $(X_i \cup Y_i)_i$ is σ_{ag}-*consistent* if $\sigma_{\mathsf{ag}}(X_0 X_1 \cdots X_j) = Y_j$ for every $j \geq 0$; and is σ_{env}-*consistent* if $\sigma_{\mathsf{env}}(\lambda) = X_0$ and $\sigma_{\mathsf{env}}(Y_0 Y_1 \cdots Y_j) = X_{j+1}$ for every $j \geq 0$. Let $\textsc{play}(\sigma_{\mathsf{ag}}, \sigma_{\mathsf{env}})$ denote the unique trace consistent with both strategies. Let φ be an LTL formula over $X \cup Y$. An agent strategy σ_{ag} *enforces* φ, written $\sigma_{\mathsf{ag}} \rhd \varphi$, if for every environment strategy σ_{env}, the sequence $\textsc{play}(\sigma_{\mathsf{ag}}, \sigma_{\mathsf{env}})$ satisfies φ, i.e., $\forall \sigma_{\mathsf{env}} \in \Sigma^{all}_{\mathsf{env}}.\, \textsc{play}(\sigma_{\mathsf{ag}}, \sigma_{\mathsf{env}}) \models \varphi$. Such strategies are called *winning strategies*. A specification φ is *agent enforceable* if some agent strategy enforces it. The set of all such agent strategies is denoted by $\Sigma^{\varphi}_{\mathsf{ag}}$. The notion of *environment enforceability*, $\sigma_{\mathsf{env}} \rhd \varphi$, and the set $\Sigma^{\varphi}_{\mathsf{env}}$ of all environment strategies enforcing φ, are defined similarly, e.g., $\sigma_{\mathsf{env}} \rhd \varphi$ if $\forall \sigma_{\mathsf{ag}} \in \Sigma^{all}_{\mathsf{ag}}.\, \textsc{play}(\sigma_{\mathsf{ag}}, \sigma_{\mathsf{env}}) \models \varphi$. Given an LTL formula φ (the *goal*), the synthesis problem for the agent is to devise a strategy σ_{ag} for the agent that enforces φ, i.e., such that $\sigma_{\mathsf{ag}} \rhd \varphi$. The synthesis problem for the environment is defined similarly. Note that if a strategy exists, also a finite-state one exists, i.e., one that is computed by a finite-state transducer [Pnueli and Rosner, 1989].

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

Environment specifications. In most AI scenarios the agent has some knowledge of how the environment works, which the agent can exploit in order to enforce its goals. To capture this, besides the specification of the goal, we introduce a specification of the environment. In particular, an environment specification is also given by an LTL formula $\mathcal{E}$ and, following [Aminof *et al.*, 2019a], it determines the set $\Sigma^{\mathcal{E}}_{\mathsf{env}}$ of strategies that the environment can adopt, i.e., those that enforce $\mathcal{E}$. Formally, let $\Sigma^{\mathcal{E}}_{\mathsf{env}} \doteq \{\sigma_{\mathsf{env}} \in \Sigma^{all}_{\mathsf{env}} \mid \sigma_{\mathsf{env}} \rhd \mathcal{E}\}$. For the environment specification to be useful, one requires that $\Sigma^{\mathcal{E}}_{\mathsf{env}}$ to be nonempty, i.e., that $\mathcal{E}$ is environment-enforceable. Given an LTL formula φ and LTL specification $\mathcal{E}$ (typically assumed to be environment-enforceable), the *synthesis under environment specifications* problem concerns determining an agent strategy σ_{ag} such that: $\forall \sigma_{\mathsf{env}} \in \Sigma^{\mathcal{E}}_{\mathsf{env}}.\, \textsc{play}(\sigma_{\mathsf{ag}}, \sigma_{\mathsf{env}}) \models \varphi$. It can be shown that most forms of planning with LTL goals, including on nondeterministic domains, are special cases of this synthesis problem, see [Aminof *et al.*, 2019a]. Moreover:

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

We observe that the verification literature on synthesis under assumptions, e.g., [Chatterjee and Henzinger, 2007], gives one an alternative (weaker) way to interpret LTL environment specifications $\mathcal{E}$, i.e., $\mathcal{E}$ directly restricts the *traces* of interest to the ones that *satisfy* $\mathcal{E}$ (in contrast, we restrict the strategies to the ones that enforce $\mathcal{E}$). In that case, synthesis amounts to devising a strategy of the agent that enforces the implication $\mathcal{E} \supset \varphi$. It turns out, that while these two ways of interpreting environment specifications are distinct, their synthesis problems are inter-reducible [Aminof *et al.*, 2019a]. We will return to this point at the end of Section 3.

3 Synthesis Under Expected and Exceptional Environment Specifications

Consider a setting with two environment specifications — $\mathcal{E}$ captures the "expected" behaviors of the environment, while $\mathcal{E}'$ captures, in addition, "exceptional" behaviors of the environment, i.e., satisfying the following constraints (†):

1. the goal φ, and environment specifications $\mathcal{E}$ and $\mathcal{E}'$ are LTL formulas;
2. $\Sigma^{\mathcal{E}}_{\mathsf{env}} \subseteq \Sigma^{\mathcal{E}'}_{\mathsf{env}}$;

Furthermore, we will be mostly interested in the case where φ is agent-enforceable under $\mathcal{E}$ but not under $\mathcal{E}'$, i.e., no agent strategy works (to satisfy the goal) against all environment strategies in $\Sigma^{\mathcal{E}'}_{\mathsf{env}}$, but there may be agent strategies that work against *some* of them. Although it may not be clear what a good agent-strategy should do, there is a basic principle about what it should not do: it should not do anything for which it can do better. Said positively, it should do a "best effort". Intuitively, the agent should use a strategy that enforces the goal under $\mathcal{E}$, and also achieve the goal, as much as possible, under $\mathcal{E}'$. Before we formalize this, consider the following:

Example 1. Take a robot whose goal is to go from the lobby to the roof using either the main elevator (ME) or the service elevator (SE). The robot starts by ME, and has 20 time-steps worth of battery life. There is a charging port inside each elevator (thus, once inside an elevator, it is certain to achieve the goal). The trip between the elevators takes 11 time-steps. Furthermore, the robot makes no partial trips (unless it runs out of battery), and once an elevator arrives the robot immediately takes it. The two environment specifications are:

- Expected ($\mathcal{E}$): ME arrives within 5 time-steps of being called; the timing of SE is unpredictable.
- Exceptional ($\mathcal{E}'$): as above, but one of the elevators may be broken. Thus, in particular, if ME does not come within 5 time-steps then it is broken.

Consider the following two types of strategies *(1)* wait for ME until it comes or the battery runs out; *(2)* wait for ME for $0 \leq w < 9$ time-steps and then go to SE and wait there until it comes, or the battery runs out. Below we give an intuitive analysis of these strategies (the analysis will be formalised after we define the concept of best-effort in the next section).

Under the expected specification, the robot can enforce the goal using any strategy that waits at least 5 time-steps for ME (and does whatever if it doesn't arrive). In particular, it can do so using the strategy of type (1). However, the strategy of type (1) performs very poorly under the exceptional specification: if ME is broken, the goal is guaranteed to never be achieved.

Nonetheless, strategies that *(i)* enforce the goal under the expected specification, and *(ii)* do a best-effort under the exceptional specification — instead of hopelessly waiting forever for an obviously broken ME — do exist (such strategies will be captured by Definition 2 in Section 3). E.g., the strategy of type (2) with $w = 5$. It is important to note that this is not the only such strategy (i.e., it is not the "best" strategy) and that any strategy of type (2) with $5 \leq w < 9$ is just as good. While it seems that taking $w = 5$ gives the most opportunities to catch SE ("why wait by an obviously broken elevator?"), this is not true, as it implicitly assumes that the timing of SE is independent of the robot's actions — an assumption that is not part of the specification. Indeed, when facing the environment strategy that breaks ME and then sends SE to the lobby at time $\max(w + 1, 27 - w)$, the robot would achieve the goal if it chooses $w = 8$, but not if it takes $w = 5$. □

Best-Effort Strategies. To formalize the notion of best-effort strategy, we first define what it means for an agent strategy to be better than another. The notion is sometimes called *weak-dominance* [Apt and Grädel, 2011]. Here we call it simply *dominance*. 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 $\sigma_{\mathsf{env}} \in \Sigma^{\mathcal{E}}_{\mathsf{env}}$, $\text{PLAY}(\sigma_2, \sigma_{\mathsf{env}}) \models \varphi$ implies $\text{PLAY}(\sigma_1, \sigma_{\mathsf{env}}) \models \varphi$. 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* dominates *σ_2* (for goal φ under environment specification $\mathcal{E}$).

Note that $\geq_{\varphi|\mathcal{E}}$ is a preorder, and $>_{\varphi|\mathcal{E}}$ is a strict partial order. For a set Σ_{ag} of agent strategies, we say that $\sigma_2 \in \Sigma_{\mathsf{ag}}$ is *maximal* (with respect to $>_{\varphi|\mathcal{E}}$) if there is no $\sigma_1 \in \Sigma_{\mathsf{ag}}$ such that $\sigma_1 >_{\varphi|\mathcal{E}} \sigma_2$. The set of such maximal strategies is denoted $\text{MAX}_{\varphi|\mathcal{E}}(\Sigma_{\mathsf{ag}})$.

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

Obviously, enforcing strategies dominate all others. Thus:

Proposition 3. *If φ is agent-enforceable under $\mathcal{E}$, then $\text{MAX}_{\varphi|\mathcal{E}}(\Sigma^{all}_{\mathsf{ag}})$ is the set of strategies enforcing φ under $\mathcal{E}$.*

Going back to Example 1, we remark that we have seen that if an agent only focuses on strategies that enforce the goal φ under the expected specification $\mathcal{E}$, it may result in strategies that fall short of the ideal. We now argue that only focusing on strategies that are best-effort under the exceptional specification $\mathcal{E}'$ may also fall short of the ideal. Indeed, since under this specification it is not clear which elevator works, and what the timing of the service elevator is, no strategy of type (2) is better than another, i.e., they are all making a best-effort for the goal under $\mathcal{E}'$. This includes the strategy that immediately goes to SE and waits there, which clearly does not enforce the goal under $\mathcal{E}$.

The Synthesis Problem. We now formally define the synthesis problem under expected and exceptional environments.

Definition 2. *Given $\varphi, \mathcal{E}, \mathcal{E}'$ satisfying (†), find an agent strategy in $\text{MAX}_{\varphi|\mathcal{E}'}(\Sigma^{\varphi}_{\mathsf{ag}})$.*

In words: amongst the agent strategies enforcing φ under the expected environment specification $\mathcal{E}$ (if any), find one that does the best-effort for φ under the exceptional environment specification $\mathcal{E}'$. Note that the case $\mathcal{E} = \mathcal{E}'$ is synthesis

under environment specifications (cf. Theorem 2), and the cast $\mathcal{E} = \mathcal{E}' = \mathsf{true}$ is classical synthesis (cf. Theorem 1).

Our synthesis problem has two important properties, captured by Theorems 4 and 7 below. The first is that if a goal φ is agent-enforceable under the expected specification $\mathcal{E}$, then there *always exists* a strategy that enforces φ under the expected specification $\mathcal{E}$ and that is best-effort under the exceptional specification $\mathcal{E}'$:

Theorem 4. *Given $\varphi, \mathcal{E}, \mathcal{E}'$ satisfying (†), if φ is agent-enforceable under $\mathcal{E}$, then* $\text{MAX}_{\varphi|\mathcal{E}'}(\text{MAX}_{\varphi|\mathcal{E}}(\Sigma^{all}_{\mathsf{ag}})) \neq \emptyset$.

The proof of this theorem requires a sophisticated argument, which we now sketch. We start by giving alternative characterizations of dominance and maximality, which are of interest in their own right. These characterizations are adaptations of ones in [Berwanger, 2007] to our setting. Intuitively, one assigns to a strategy σ_{ag}, and an environment-history $h \in (2^X)^+$, the value $+1$ (*winning*) if σ_{ag}, starting from h, enforces φ under $\mathcal{E}$; the value -1 (*losing*) if σ_{ag}, starting from h, enforces $\neg\varphi$ under $\mathcal{E}$; and 0 (*pending*) otherwise. The characterization says that $\sigma_1 \geq_{\varphi|\mathcal{E}} \sigma_2$ iff the value of σ_1 is as least as big as that of σ_2 on all environment histories at which σ_1, σ_2 agree except for the last point. Formally, an environment-history $h \in (2^X)^+$ is the *projection* of a (joint) history $h' \in (2^{X \cup Y})^+$ if $h_n = h'_n \cap X$ for all $n < |h|$. Say that h is *compatible* with $\sigma_{\mathsf{ag}}, \sigma_{\mathsf{env}}$ iff h is the projection of a prefix of $\text{PLAY}(\sigma_{\mathsf{ag}}, \sigma_{\mathsf{env}})$. Given a set Σ_{env} of environment strategies, let $H(\sigma_{\mathsf{ag}}, \Sigma_{\mathsf{env}})$ be the set of environment histories compatible with $\sigma_{\mathsf{ag}}, \sigma_{\mathsf{env}}$ for some $\sigma_{\mathsf{env}} \in \Sigma_{\mathsf{env}}$. Given a goal φ and an environment specification $\mathcal{E}$, for every $h \in H(\sigma_{\mathsf{ag}}, \Sigma^{\mathcal{E}}_{\mathsf{env}})$, let:

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

Agent-strategies σ_1, σ_2 *split* at an environment-history h if $\sigma_1(h') = \sigma_2(h')$ for every proper prefix h' of h, but $\sigma_1(h) \neq \sigma_2(h)$. Since σ_1, σ_2 agree on proper prefixes of h, then, for every environment strategy σ_{env}, the history h is compatible with $\sigma_1, \sigma_{\mathsf{env}}$ iff it is compatible with $\sigma_2, \sigma_{\mathsf{env}}$. If σ_1, σ_2 split at h, denote by $\sigma_1[h \leftarrow \sigma_2]$ the strategy that agrees with σ_1 everywhere, except on h and all its extensions where it agrees with σ_2. A set Σ_{ag} is *closed under shifting* if for all $\sigma_1, \sigma_2 \in \Sigma_{\mathsf{ag}}$ and every h at which they split, $\sigma_1[h \leftarrow \sigma_2] \in \Sigma_{\mathsf{ag}}$.

Lemma 5 (Characterization of Dominance)**.** *Given a goal φ, an environment specification $\mathcal{E}$, and agent strategies σ_1, σ_2, then A) $\sigma_1 \geq_{\varphi|\mathcal{E}} \sigma_2$ iff for every $h \in H(\sigma_1, \Sigma^{\mathcal{E}}_{\mathsf{env}})$, if σ_1, σ_2 split at h then: (i) $val_{\varphi|\mathcal{E}}(\sigma_1, h) \geq val_{\varphi|\mathcal{E}}(\sigma_2, h)$, and (ii) it is not the case that $val_{\varphi|\mathcal{E}}(\sigma_1, h) = val_{\varphi|\mathcal{E}}(\sigma_2, h) = 0$. Furthermore, B) $\sigma_1 >_{\varphi|\mathcal{E}} \sigma_2$ iff also (iii) σ_1, σ_2 split at some $h \in H(\sigma_1, \Sigma^{\mathcal{E}}_{\mathsf{env}})$ such that $val_{\varphi|\mathcal{E}}(\sigma_1, h) > val_{\varphi|\mathcal{E}}(\sigma_2, h)$.*

Proof sketch. We will show case A. For $i \in \{1, 2\}$, let $\ddagger_i$ denote that $\text{PLAY}(\sigma_i, \sigma_{\mathsf{env}}) \models \varphi$. Suppose (i) and (ii) hold for σ_1, σ_2. To see that $\sigma_1 \geq_{\varphi|\mathcal{E}} \sigma_2$ we show that for every $\sigma_{\mathsf{env}} \in \Sigma^{\mathcal{E}}_{\mathsf{env}}$ for which $\ddagger_2$ holds, also $\ddagger_1$ holds. Indeed, if $\text{PLAY}(\sigma_1, \sigma_{\mathsf{env}}) \neq \text{PLAY}(\sigma_2, \sigma_{\mathsf{env}})$, let h be the longest common environment-history compatible with $\sigma_1, \sigma_{\mathsf{env}}$ and $\sigma_2, \sigma_{\mathsf{env}}$, and note that σ_1, σ_2 split at h. If $\ddagger_2$ holds then $val_{\varphi|\mathcal{E}}(\sigma_2, h) \geq 0$ which implies, by (i) and (ii), that $val_{\varphi|\mathcal{E}}(\sigma_1, h) = 1$, and in particular $\ddagger_1$ holds.

For the converse, suppose that for some h at which σ_1, σ_2 split, either (i) or (ii) is violated. We will show that $\sigma_1 \not\geq_{\varphi|\mathcal{E}} \sigma_2$, i.e., that there is $\sigma_{\mathsf{env}} \in \Sigma^{\mathcal{E}}_{\mathsf{env}}$ such that $\ddagger_2$ holds but $\ddagger_1$ does not. If (i) fails then at h either σ_1 is losing and σ_2 is pending/winning, or σ_1 is pending and σ_2 is winning. In either case, for some σ_{env}, $\ddagger_2$ holds but not $\ddagger_1$. If (ii) fails, there are strategies $\delta_1, \delta_2 \in \Sigma^{\mathcal{E}}_{\mathsf{env}}$, such that h is compatible with both σ_1, δ_1 and σ_2, δ_2, and $\text{PLAY}(\sigma_1, \delta_1) \models \neg\varphi$ and $\text{PLAY}(\sigma_2, \delta_2) \models \varphi$. Let $w \in (2^Y)^+$ be the agent-history describing the outputs of σ_1 (as well as σ_2, since they agree on these prefixes) on the proper prefixes of h. Consider the environment-strategy σ_{env} that agrees with δ_1 everywhere, except on $w \cdot \sigma_2(h)$ and all its extensions where it agrees with δ_2, and observe that $\ddagger_2$ holds but not $\ddagger_1$. Note that $\sigma_{\mathsf{env}} \in \Sigma^{\mathcal{E}}_{\mathsf{env}}$ since both δ_1 and δ_2 enforce $\mathcal{E}$. □

Lemma 6 (Characterization of Maximality)**.** *For Σ_{ag} closed under shifting, $\sigma \in \text{MAX}_{\varphi|\mathcal{E}}(\Sigma_{\mathsf{ag}})$ iff for every $\sigma' \in \Sigma_{\mathsf{ag}}$, and every $h \in H(\sigma, \Sigma^{\mathcal{E}}_{\mathsf{env}})$ at which σ, σ' split, $val_{\varphi|\mathcal{E}}(\sigma, h) \geq val_{\varphi|\mathcal{E}}(\sigma', h)$.*

Proof sketch. If $val_{\varphi|\mathcal{E}}(\sigma, h) < val_{\varphi|\mathcal{E}}(\sigma', h)$ then $\sigma \notin \text{MAX}_{\varphi|\mathcal{E}}(\Sigma_{\mathsf{ag}})$ since, by Lemma 5, it is dominated by $\sigma[h \leftarrow \sigma']$. For the other direction, if σ' dominates σ then by Lemma 5, $val_{\varphi|\mathcal{E}}(\sigma', h) > val_{\varphi|\mathcal{E}}(\sigma, h)$ for some h. □

Proof Sketch of Theorem 4. Since φ is enforceable under $\mathcal{E}$ then, by Proposition 3, $\Sigma_{\mathsf{ag}} \doteq \text{MAX}_{\varphi|\mathcal{E}}(\Sigma^{all}_{\mathsf{ag}})$ is not empty. It remains to construct a strategy $\sigma_{\mathsf{ag}} \in \text{MAX}_{\varphi|\mathcal{E}'}(\Sigma_{\mathsf{ag}})$, which we do by constructing a chain of strategies, $\sigma_1, \sigma_2, \cdots$ that eventually stabilises on each history, and then let σ_{ag} be the point-wise limit of this sequence, i.e., $\sigma_{\mathsf{ag}}(h) := \lim_i \sigma_i(h)$. We start with an arbitrary strategy $\sigma_0 \in \Sigma_{\mathsf{ag}}$, and process the environment histories in increasing length-lexicographic order. At step $i \geq 1$, consider the smallest history h not yet marked as stabilized, and mark it. Let $\Sigma_h \subseteq \Sigma_{\mathsf{ag}}$ be the set of strategies that agree with σ_i on all proper prefixes of h, and let $\sigma \in \Sigma_h$ be a strategy for which $val_{\varphi|\mathcal{E}'}(\sigma, h)$ is the highest. If $val_{\varphi|\mathcal{E}'}(\sigma_i, h) < val_{\varphi|\mathcal{E}'}(\sigma, h)$, then let $\sigma_{i+1} \doteq \sigma_i[h \leftarrow \sigma]$, and otherwise let $\sigma_{i+1} \doteq \sigma_i$. Moreover, if $val_{\varphi|\mathcal{E}'}(\sigma_{i+1}, h) \neq 0$, then also mark all the extensions of h as stabilized. The full proof shows that $val_{\varphi|\mathcal{E}'}(\sigma_i, h)$ cannot deteriorate after h is marked and thus, by Lemma 5, σ_{ag} is not dominated. Finally, using a careful argument involving Lemma 6 and the fact that $\Sigma^{\mathcal{E}}_{\mathsf{env}} \subseteq \Sigma^{\mathcal{E}'}_{\mathsf{env}}$, one shows that $\sigma_{\mathsf{ag}} \in \Sigma_{\mathsf{ag}}$, and conclude that $\sigma_{\mathsf{ag}} \in \text{MAX}_{\varphi|\mathcal{E}'}(\Sigma_{\mathsf{ag}})$. □

The second important property of our synthesis problem is that instead of taking the best-effort strategies for φ assuming $\mathcal{E}$, and amongst those choosing one that is best-effort for φ assuming $\mathcal{E}'$, one can simply pick a strategy that is both best-effort for φ assuming $\mathcal{E}$ and best-effort for φ assuming $\mathcal{E}'$.

Theorem 7. *If $\varphi, \mathcal{E}, \mathcal{E}'$ satisfy the constraint (†) then* $\text{MAX}_{\varphi|\mathcal{E}'}(\text{MAX}_{\varphi|\mathcal{E}}(\Sigma^{all}_{\mathsf{ag}})) = \text{MAX}_{\varphi|\mathcal{E}}(\Sigma^{all}_{\mathsf{ag}}) \cap \text{MAX}_{\varphi|\mathcal{E}'}(\Sigma^{all}_{\mathsf{ag}})$.

Proof Sketch. Let $\Sigma_{\mathsf{ag}} \doteq \text{MAX}_{\varphi|\mathcal{E}}(\Sigma_{\mathsf{ag}}^{all})$ and observe that $\text{MAX}_{\varphi|\mathcal{E}'}(\Sigma_{\mathsf{ag}}^{all}) \cap \Sigma_{\mathsf{ag}} \subseteq \text{MAX}_{\varphi|\mathcal{E}'}(\Sigma_{\mathsf{ag}})$ since a strategy $\sigma_{\mathsf{ag}} \in \Sigma_{\mathsf{ag}}$ that is not dominated by any strategy is also not dominated by any strategy in Σ_{ag}. For the reverse inclusion, observe that we only need to show that if $\sigma_{\mathsf{ag}} \in \text{MAX}_{\varphi|\mathcal{E}'}(\Sigma_{\mathsf{ag}})$ then $\sigma_{\mathsf{ag}} \in \text{MAX}_{\varphi|\mathcal{E}'}(\Sigma_{\mathsf{ag}}^{all})$, i.e., no strategy whatsoever is $>_{\varphi|\mathcal{E}'} \sigma_{\mathsf{ag}}$. Suppose $\sigma >_{\varphi|\mathcal{E}'} \sigma_{\mathsf{ag}}$. Then $\sigma \geq_{\varphi|\mathcal{E}} \sigma_{\mathsf{ag}}$ by item (2) in the constraint (†). But σ_{ag} is $\geq_{\varphi|\mathcal{E}}$-maximal, hence so is σ, i.e., $\sigma \in \Sigma_{\mathsf{ag}}$. Since $\sigma_{\mathsf{ag}} \in \text{MAX}_{\varphi|\mathcal{E}'}(\Sigma_{\mathsf{ag}})$, no strategy in Σ_{ag} is $>_{\varphi|\mathcal{E}'} \sigma_{\mathsf{ag}}$, which is a contradiction. □

We now discuss this theorem for the case that φ is agent-enforceable under the expected specification $\mathcal{E}$. In this case, by Proposition 3, the agent-strategies that are best-effort for φ under $\mathcal{E}$ are simply those that enforce φ under $\mathcal{E}$. Then, Theorem 7 implies that solving the synthesis problem amounts to finding an agent strategy that enforces the goal under the expected environment on the one hand, and does a best-effort under the exceptional environment on the other hand. Thus, there is no trade-off between enforcing the goal under the expected environment specification, and making a best-effort for it under the exceptional one. This is quite surprising since it is analogous to saying that to get the tallest professors one can get the tallest people and intersect them with the professors. Indeed, there is no reason to think that there are any professors among the tallest people.

Interestingly, an analogue of Theorem 7 fails under the alternative view that environment specifications restrict traces of interest as in [Chatterjee and Henzinger, 2007], instead of environment strategies as in [Aminof *et al.*, 2019a] and this paper (cf. Section 2).

Proposition 8. *There exist* LTL *formulas φ, $\mathcal{E}$, and $\mathcal{E}'$ with $\models \mathcal{E} \supset \mathcal{E}'$, the formula $\mathcal{E} \supset \varphi$ agent-enforceable, the formula $\mathcal{E}' \supset \varphi$ not agent-enforceable, such that $\text{MAX}_{\mathcal{E}\supset\varphi|\mathsf{true}}(\Sigma_{\mathsf{ag}}^{all}) \cap \text{MAX}_{\mathcal{E}'\supset\varphi|\mathsf{true}}(\Sigma_{\mathsf{ag}}^{all}) = \emptyset$ and $\text{MAX}_{\mathcal{E}'\supset\varphi|\mathsf{true}}(\text{MAX}_{\mathcal{E}\supset\varphi|\mathsf{true}}(\Sigma_{\mathsf{ag}}^{all})) \neq \emptyset$.*

Proof. Let $X \doteq \{x\}, Y \doteq \{y\}, \mathcal{E} \doteq x \supset \mathsf{G}\, y, \mathcal{E}' := \mathsf{true}$ and $\varphi \doteq x \supset (\mathsf{GF}\, x \wedge \mathsf{G}\, y)$. Then, strategies in the set M of maximal agent-strategies for $\mathcal{E} \supset \varphi$ do the following: if the environment starts with $\neg x$ then they can do anything (since, in this case, both $\mathcal{E}$ and φ are true); if the environment starts with x then at some point they do $\neg y$ (since, doing this, the agent ensures that the assumption $\mathcal{E}$ fails, and thus $\mathcal{E} \supset \varphi$ is satisfied). On the other hand, strategies in the set M' of maximal agent-strategies for $\mathcal{E}' \supset \varphi$, i.e., for φ, do the following: if the environment does $\neg x$ they can do anything, and otherwise they always do y (which ensures, in this case, that φ is satisfied if the environment does x infinitely often). Note, however, that $M \cap M' = \emptyset$. On the other hand $\text{MAX}_{\mathcal{E}'\supset\varphi|\mathsf{true}}(\text{MAX}_{\mathcal{E}\supset\varphi|\mathsf{true}}(\Sigma_{\mathsf{ag}}^{all}))$ is non-empty (indeed, it contains all strategies in M.) □

Hence, even if the two views of environment specifications share most properties when synthesizing under single environment specifications [Aminof *et al.*, 2019a], they do not under expected and exceptional environment specifications.

4 Decidability of Synthesis

Next we show that the synthesis problem is decidable. We do so using an automata-theoretic approach. We represent strategies as trees, build tree-automata that recognize the sets $\text{MAX}_{\varphi|\mathcal{E}'}(\Sigma_{\mathsf{ag}}^{all})$ and $\Sigma_{\mathsf{ag}}^{\varphi}$, then pick a strategy (if one exists) in their intersection. This outline is correct by Theorem 7 and Proposition 3.

Theorem 9. *There is an algorithm that solves the problem of synthesis under expected and exceptional environments, i.e., that given $\varphi, \mathcal{E}, \mathcal{E}'$ satisfying (†) returns a finite-state strategy in $\text{MAX}_{\varphi|\mathcal{E}'}(\Sigma_{\mathsf{ag}}^{\varphi})$.*

We give a succinct review of the relevant notation and results about tree automata, see [Grädel *et al.*, 2002].

Encoding. Functions of the form $f : B^* \to A$ can be thought of as B-branching A-labeled trees, i.e., the nodes are the elements of B^* and the label of w is $f(w)$. Similarly, a pair (f_1, f_2) of functions $f_i : B^* \to A$ can be encoded as the tree in which node $w \in A^*$ is labeled by the pair $(f_1(w), f_2(w))$. We encode agent-strategies, and pairs of agent-strategies in this way, i.e., $B \doteq 2^X, A \doteq 2^Y$ (the label of the root is assumed to be fixed since it does not encode anything).

Tree Automata. An *alternating parity tree automata (APT)* T is a tuple $(B, A, Q, q_0, \delta, c)$ a finite branching alphabet B, a finite input alphabet A, a finite set of states Q, an initial state $q_0 \in Q$, a transition function $\delta : Q \times A \to \text{PosBool}(B \times Q)$, and a priority function $c : Q \to \mathbb{N}$. Here $\text{PosBool}(X)$ is the set of positive Boolean formulas over the atoms X. An APT in which every formula is of the form $\bigwedge_{b\in B}(b, q_b)$ is called *deterministic*. Tree automata take trees $t : B^* \to A$ as input, and process them, starting at the root in the initial state. Intuitively, if the APT is at node $w \in A^*$ of the input tree t, and the current state is $q \in Q$, then the APT picks a satisfying assignment $C \subseteq B \times Q$ of the formula $\delta(q, t(w))$ and, for each $(b, q) \in C$ *sends a copy of itself in state q in the direction b*. Intuitively, a tree t is *accepted* by an APT T if the evolution of every sequence of copies has the property that the smallest priority seen infinitely often is even. The set of all trees accepted by T is called the *language* of T. APTs are effectively closed under Boolean operations and projection. Moreover, one can decide, given an APT T, if its language is non-empty, and in this case also extract a finite-state representation of some t accepted by T. We also use automata on words. We can think of words as trees with $|B| = 1$, i.e., no branching. In this case we will ignore B and write $\delta : Q \times A \to \text{PosBool}(Q)$. Given an LTL formula φ, one can construct a deterministic parity word automaton (DPW) D_φ accepting exactly the models of φ. We write $D_\varphi = (2^{AP}, Q_\varphi, q_\varphi, \delta_\varphi, c_\varphi)$.

Proof Sketch of Theorem 9. The interesting step is to show how to build an APT D' that accepts exactly the pairs (σ_1, σ_2) of agent strategies such that $\sigma_1 \not\geq_{\varphi|\mathcal{E}'} \sigma_2$. Given this, one can build an APT $M_{\varphi|\mathcal{E}'}$ that accepts exactly the agent strategies σ in $\text{MAX}_{\varphi|\mathcal{E}'}(\Sigma_{\mathsf{ag}}^{all})$ by using the fact that maximality is definable in terms of D', Boolean operations, and projection, i.e., following Definition 1, σ is maximal iff $\neg\exists\sigma'.\, (\sigma' \geq_{\varphi|\mathcal{E}'} \sigma) \wedge (\sigma \not\geq_{\varphi|\mathcal{E}'} \sigma')$. Finally, one can extract a finite-state strategy from $M_{\varphi|\mathcal{E}'}$.

To illustrate some of the core ideas, we first sketch how to build an APT T' that accepts σ_{ag} iff σ_{ag} does not enforce φ under $\mathcal{E}'$, i.e., iff there exists σ_{env} such that $\sigma_{\mathsf{env}} \rhd \mathcal{E}'$ and $\textsc{play}(\sigma_{\mathsf{ag}}, \sigma_{\mathsf{env}}) \models \neg\varphi$. Let $D_{\mathcal{E}'}$ and $D_{\neg\varphi}$ be DPWs for $\mathcal{E}'$ and $\neg\varphi$ respectively. Intuitively, T' will use its nondeterminism to guess σ_{env} (step by step), and simulate $D_{\mathcal{E}'}$ to verify $\sigma_{\mathsf{env}} \rhd \mathcal{E}'$, and simulate $D_{\neg\varphi}$ to verify that $\textsc{play}(\sigma_{\mathsf{ag}}, \sigma_{\mathsf{env}}) \models \neg\varphi$.

The states of T' are of the form (s, q) and s where $s \in Q_{\mathcal{E}'}$ and $q \in Q_{\neg\varphi}$. For the transitions, we describe what the automaton does when it is at node $w \in (2^X)^*$ in a given state. When w is the empty string, the APT T' nondeterministically chooses $X' \subseteq X$ and sends a copy in state $(s_{\mathcal{E}'}, q_{\neg\varphi})$ in direction X' (this corresponds to guessing $\sigma_{\mathsf{env}}(\lambda)$, the first action of the environment). When w is not the empty-string, the APT T' reads the agent action $\sigma_{\mathsf{ag}}(w) \subseteq Y$, and works differently depending if it is in a state of the form (s, q) or s.

If the APT T' is in a state of the form (s, q) it does two things: 1) for every $Y'' \subseteq Y$ such that $Y'' \neq \sigma_{\mathsf{ag}}(w)$, it nondeterministically chooses an environment action $X'' \subseteq X$ and sends in direction X'' a copy of $D_{\mathcal{E}'}$ in state $s' := \delta_{\mathcal{E}'}(s, X'' \cup Y'')$; and 2) for $Y'' = \sigma_{\mathsf{ag}}(w)$, it also guesses an environment action $X'' \subseteq X$ and sends (s', q') in direction X'' where s' is as before and $q' := \delta_{\neg\varphi}(q, X'' \cup Y'')$. Intuitively, 1) means that T' is implicitly guessing an environment strategy σ_{env} and checks that it enforces $\mathcal{E}'$; and 2) means that it checks that $\textsc{play}(\sigma_{\mathsf{ag}}, \sigma_{\mathsf{env}})$ satisfies $\neg\varphi$.

If the APT T' is in a state of the form s it does the following: for every $Y'' \subseteq Y$ (including the current label) it nondeterministically chooses an environment action $X'' \subseteq X$ and sends in direction X'' a copy of $D_{\mathcal{E}'}$ in state $s' := \delta_{\mathcal{E}'}(s, X'' \cup Y'')$. Intuitively, this is like 1) in the previous case, except that since it is at a node that is not on $\textsc{play}(\sigma_{\mathsf{ag}}, \sigma_{\mathsf{env}})$ it no longer simulates $D_{\neg\varphi}$.

We now describe how to adapt this construction to build the APT D'. Recall that we require that D' accepts (σ_1, σ_2) iff there exists $\sigma_{\mathsf{env}} \rhd \mathcal{E}'$ such that $\textsc{play}(\sigma_1, \sigma_{\mathsf{env}}) \models \neg\varphi$ and $\textsc{play}(\sigma_2, \sigma_{\mathsf{env}}) \models \varphi$. The states of D' are of the form (s, q, r), (s, q), (s, r), and s, where $s \in Q_{\mathcal{E}'}$, $q \in Q_{\neg\varphi}$, and $r \in Q_{\varphi}$. Intuitively, transitions from states of the form s and (s, q) are as before, transitions from states (s, r) are like those of (s, q) but they use the automaton D_{φ} instead of $D_{\neg\varphi}$, and transitions of the form (s, q, r) from node $w \in (2^X)^+$ send copies of the form (s', q', r') if $\sigma_1(w) = \sigma_2(w)$, and copies of the form (s', q') and (s', r') otherwise. In any case, they also send copies of the form s', as before. Thus, the copies of s verify that $\sigma_{\mathsf{env}} \rhd \mathcal{E}'$, the copies of q that $\textsc{play}(\sigma_1, \sigma_{\mathsf{env}}) \models \neg\varphi$, and the copies of r that $\textsc{play}(\sigma_2, \sigma_{\mathsf{env}}) \models \varphi$. □

5 Related Work and Conclusions

Agents in our framework have two models of environment behaviour, expected and exceptional. The importance of multiple environment models is recognized in other work on agents operating in complex AI scenarios. E.g., in [Chakraborti *et al.*, 2019] agents use one model of the world to plan, and a different model is used for explanations for the human operator, and in [De Giacomo *et al.*, 2019; Camacho *et al.*, 2019] reinforcement learning agents complement their features extracted from the world with a second model which specifies rewards based on knowledge representation.

An alternative approach to capturing expected behaviour is to use a stochastic setting. In the usual approach to synthesis for probabilistic systems the environment is not given as a set of strategies, as we do, but rather as a single stochastic system (e.g., as an MDP). This setting has been extended to include a second environment in the so-called "beyond worst-case synthesis problem", which concerns finding a strategy that guarantees worst-case performance against an adversarial environment, while maximising the expected performance against a fixed stochastic environment [Bruyère *et al.*, 2017].

The synthesis problem can be expressed in a number of logics tailored for strategic reasoning. In particular, Strategy Logic and its variants [Mogavero *et al.*, 2014; Berthon *et al.*, 2017; Aminof *et al.*, 2019b] — which can quantify over variables for (possibly randomized) strategies — can naturally express complex strategic properties, including dominance and maximality.

"Best-effort" strategies, sometimes called "admissible", were studied in the verification literature, however, without distinguishing expected from exceptional environments. For instance, [Berwanger, 2007] studies the process of iteratively avoiding dominated strategies, [Faella, 2009] justifies admissibility amongst other solution-concepts in environments that may not be adversarial, and [Brenguier *et al.*, 2017] assumes the environment has its own objective and both the agent and the environment use non-dominated strategies.

Exceptional environments are related to safety analysis via fault-injection [Bozzano and Villafiorita, 2007; Ezekiel *et al.*, 2011; Ezekiel and Lomuscio, 2017]. Specifically, exceptional environments may be seen as the result of mutations from expected environments via fault-injection. While safety-analysis is concerned with generating these mutated models algorithmically using fault-profiles and then verifying them, we assume that they are given and use them to synthesize agent strategies against them.

The exact complexity of the synthesis problem under expected and exceptional environments is left open. Other future work includes finding better algorithms for special cases, including nondeterministic planning domains and LTL formulas on finite traces for specifying non-Markovian domains.

Acknowledgements

Benjamin Aminof is supported by the Austrian Science Fund (FWF): P 32021. Giuseppe De Giacomo is supported in part by the ERC under the European Union's Horizon 2020 Programme through the ERC Advanced Grant WhiteMech (No. 834228). Alessio Lomuscio is supported by a Royal Academy of Engineering Chair in Emerging Technologies.

References

[Aminof *et al.*, 2018] B. Aminof, G. De Giacomo, A. Murano, and S. Rubin. Synthesis under assumptions. In *KR*, 2018.

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

[Aminof *et al.*, 2019b] B. Aminof, M. Kwiatkowska, B. Maubert, A. Murano, and S. Rubin. Probabilistic strategy logic. In *IJCAI*, 2019.

[Aminof *et al.*, 2020] B. Aminof, G. De Giacomo, and S. Rubin. Stochastic fairness and language-theoretic fairness in planning on nondeterministic domains. In *ICAPS*, 2020.

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

[Bacchus and Kabanza, 1996] F. Bacchus and F. Kabanza. Planning for temporally extended goals. In *AAAI*, 1996.

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

[Belardinelli *et al.*, 2017] F. Belardinelli, A. Lomuscio, A. Murano, and S. Rubin. Verification of broadcasting multi-agent systems against an epistemic strategy logic. In *IJCAI*, 2017.

[Belle, 2018] V. Belle. On plans with loops and noise. In *AAMAS*, 2018.

[Berthon *et al.*, 2017] R. Berthon, B. Maubert, A. Murano, S. Rubin, and M.Y. Vardi. Strategy logic with imperfect information. In *LICS*, 2017.

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

[Bloem *et al.*, 2015] R. Bloem, R. Ehlers, and R. Könighofer. Cooperative reactive synthesis. In *ATVA*, 2015.

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

[Bozzano and Villafiorita, 2007] M. Bozzano and A. Villafiorita. The FSAP/NuSMV-SA safety analysis platform. *Software Tools for Technology Transfer*, 9(1), 2007.

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

[Bruyère *et al.*, 2017] V. Bruyère, E. Filiot, M. Randour, and J.F. Raskin. Meet your expectations with guarantees: Beyond worst-case synthesis in quantitative games. *Inf. Comput.*, 254:259–295, 2017.

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

[Camacho *et al.*, 2019] A. Camacho, R. Toro Icarte, T. Klassen, R. Valenzano, and S. McIlraith. LTL and beyond: Formal languages for reward function specification in reinforcement learning. In *IJCAI*, 2019.

[Chakraborti *et al.*, 2019] T. Chakraborti, A. Kulkarni, S. Sreedharan, D. Smith, and S. Kambhampati. Explicability? Legibility? Predictability? Transparency? Privacy? Security? The emerging landscape of interpretable agent behavior. In *ICAPS*, 2019.

[Chatterjee and Henzinger, 2007] K. Chatterjee and T. Henzinger. Assume-guarantee synthesis. In *TACAS*, 2007.

[Daniele *et al.*, 1999] M. Daniele, P. Traverso, and M. Vardi. Strong cyclic planning revisited. In *ECP*, 1999.

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

[De Giacomo and Vardi, 1999] G. De Giacomo and M. Vardi. Automata-theoretic approach to planning for temporally extended goals. In *ECP*, 1999.

[De Giacomo *et al.*, 2013] G. De Giacomo, F. Patrizi, and S. Sardiña. Automatic behavior composition synthesis. *Artif. Intell.*, 196, 2013.

[De Giacomo *et al.*, 2019] G. De Giacomo, L. Iocchi, M. Favorito, and F. Patrizi. Foundations for restraining bolts: Reinforcement learning with LTLf/LDLf restraining specifications. In *ICAPS*, 2019.

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

[Ezekiel and Lomuscio, 2017] J. Ezekiel and A. Lomuscio. Combining fault injection and model checking to verify fault tolerance, recoverability, and diagnosability in multi-agent systems. *Inf. Comput.*, 254(2):167–194, 2017.

[Ezekiel *et al.*, 2011] J. Ezekiel, A. Lomuscio, L. Molnar, and S. Veres. Verifying fault tolerance and self-diagnosability of an autonomous underwater vehicle. In *IJCAI*, 2011.

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

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

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

[Grädel *et al.*, 2002] E. Grädel, W. Thomas, and T. Wilke, editors. *Automata, Logics, and Infinite Games: A Guide to Current Research*, LNCS 2500, 2002.

[Maubert and Murano, 2018] B. Maubert and A. Murano. Reasoning about knowledge and strategies under hierarchical information. In *KR*, 2018.

[Mogavero *et al.*, 2014] F. Mogavero, A. Murano, G. Perelli, and M.Y. Vardi. Reasoning about strategies: On the model-checking problem. *ACM Trans. Comput. Log.*, 15(4):34:1–34:47, 2014.

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

[Reiter, 2001] R. Reiter. *Knowledge in Action: Logical Foundations for Specifying and Implementing Dynamical Systems*. MIT Press, 2001.