Proceedings of the 22nd International Conference on Principles of Knowledge Representation and Reasoning
KR in Planning and Scheduling

LTL Synthesis under Multi-Agent Environment Assumptions

Benjamin Aminof[1] , **Giuseppe De Giacomo**[1,2] , **Giuseppe Perelli**[1] , **Sasha Rubin**[3]
[1]University of Rome "La Sapienza", Italy
[2]University of Oxford, United Kingdom
[3]University of Sydney, Australia

Abstract

We investigate LTL synthesis under structured assumptions about the environment. In our setting, the environment is viewed by the protagonist as a collection of peer agents acting together in a shared world. In contrast to the symmetrical frameworks typically studied in multi-agent systems, we take a strikingly asymmetric first-person perspective in which the *protagonist* ascribes a specification to each of its peer agents and the world, capturing its understanding of their possible strategies. We show that in this setting, LTL synthesis has the same computational complexity as standard LTL synthesis, i.e., 2EXPTIME-complete. We establish this via a sophisticated, yet fully implementable, argument that builds on the notion of traces compatible with strategies: we use the fact that if the basic specification of the world and of each agent is given in LTL then the sets of traces compatible with the strategies describing the behaviors of the agents are omega-regular. This enables the use of word-automata rather than the more complicated tree-automata.

1 Introduction

We study LTL synthesis under environment assumptions, where the agent assumes that the environment reacts to its actions according to some specification, which can be exploited in realizing the agent's specification (or task, or goal) (Camacho, Bienvenu, and McIlraith 2018; Aminof et al. 2018; Aminof et al. 2019). Such assumptions can take many forms, ranging from fully observable nondeterministic (FOND) domains (Cimatti et al. 2003; De Giacomo and Rubin 2018; Camacho, Bienvenu, and McIlraith 2019; Ghallab et al. 2025) — where synthesis is related to strong planning — to non-Markovian domains where, for example, the preconditions and effects of actions may depend on the entire history (Gabaldon 2011; Brafman and De Giacomo 2019; Bonassi et al. 2024), to more general safety specifications (FONDs and their non-Markovian variants are specific forms of safety specifications), and to LTL arbitrary specifications that include both safety and liveness conditions (Aminof et al. 2019).

In this paper we consider environment assumptions that have a certain structure. We assume that the protagonist agent, the one for which we are introducing decision-making abilities (i.e., synthesizing strategies for), sees the environment as composed of a finite set of peer agents acting alongside itself in a shared world. The protagonist ascribes a specification to each of its peer agents and the world, capturing its understanding of the possible strategies they may adopt. The peer agents are assumed to be unaware of the specifications of the other agents (including of the protagonist), but share with the protagonist the model of the world.

The specifications of all agents and the world itself are expressed in Linear Temporal Logic (LTL) (Pnueli 1977), which is arguably the most widely used specification language for dynamic properties in verification (Clarke et al. 2018). These formulas are defined over a common set of variables, including the possible actions of each agent and the world's reactions to such actions. Note that each agent controls only its own actions (and the world its reactions).

In this structured environment, the protagonist aims to devise a strategy to fulfill its own specification *assuming* that all peers will employ strategies to realize their own specifications in the shared world. This is clearly related to rational decision-making in multi-agent systems (MAS) (Wooldridge 2009), see related work below. However, our setting induces a striking asymmetry, reflecting a first-person perspective (that of the protagonist who is ascribing specifications to its peers), in contrast to the more symmetrical situations typically studied in MAS.

We show that synthesis in this setting remains 2EXPTIME-complete, matching the complexity of standard LTL synthesis. Moreover, we present an automata-theoretic solution technique that is amenable to implementation in a way analogous to standard synthesis.

To get these results, we take a detour through a sophisticated, yet fully implementable, argument (cf. Technical Intermezzo Section). We note that each peer agent has an infinite collection of possible strategies that realize its specification despite the behavior of others, and each of them will employ one such strategy. We define the notion of traces consistent with some strategy in a given set. This notion of consistency gives rise to a set of traces (a language), analogous to the set of traces satisfying an LTL property. We show that, if the set of strategies is defined through realizability in LTL, then the consistent traces, while not themselves expressed in LTL, are still omega-regular properties. This allows us to leverage automata-theoretic methods. Exploiting this result, we develop a multi-layered approach: at the first layer, we capture the set of traces compatible with the peers'

strategies that realize their specification in the world; then, we synthesize a protagonist's strategy that realizes its specification in an environment defined by the first layer specification. Based on this construction, we develop a suitable synthesis technique for our setting that remains within 2EXPTIME. In fact, as we discuss at the end of the paper, this approach can be extended to arbitrary numbers of layers as long as the assumptions remain acyclic.

Related work. Our work is related to the literature on synthesis in the context of MAS (Fagin et al. 1995; Wooldridge 2009; Albrecht and Stone 2018). Essentially, we can single out two fundamental differences in our setting: the asymmetry of agents' assumptions and the first-person point of view on the synthesis problem. The first characteristic makes iterated admissible refinement inapplicable (Berwanger 2007; Brenguier, Raskin, and Sassolas 2014; Brenguier, Raskin, and Sankur 2017). The second one sets our work apart from the conventional third-person one of Multi-Agent Systems, where the aim is synthesizing a *strategy profile* that satisfies a desired property of the system, e.g., being in an Equilibrium (Gutierrez, Harrenstein, and Wooldridge 2015).

Among the MAS Synthesis problems, Rational Synthesis (RS) deserves a special mention, as both third- and first-person perspectives are combined. Indeed, *weak-RS*, concerns with finding a strategy for a protagonist agent (first-person) *and* a strategy profile in Equilibrium (third-person) for the remaining ones such that the resulting outcome satisfies the protagonist's trace property (Fisman, Kupferman, and Lustig 2010). While, *strong-RS* concerns with finding a strategy for the protagonist such that *all possible outcomes* resulting from an Equilibrium satisfy the protagonist's trace property (Kupferman, Perelli, and Vardi 2016). Note that both forms of RS require to reason iteratively on the other agents' assumption to get the Equilibrium.

In Formal Methods, Distributed Synthesis (Pnueli and Rosner 1990; Finkbeiner and Schewe 2005) is concerned with a coalition of agents that cooperatively coordinates to ensure a shared goal regardless of all possible behavior of a single (adversarial) environment. However, the partial visibility on the state of the local agents makes this problem very difficult ranging from nonelementary, if state visibility is hierarchical, to undecidable, otherwise. Solutions for special cases based on assume-guarantee frameworks have been recently proposed (Majumdar et al. 2020; Schuppe and Tumova 2020; Fijalkow et al. 2022).

Finally, our synthesis problem can be expressed in *Strategy Logic* (SL) (Mogavero et al. 2014), which quantifies directly over strategies, though this does not help in analyzing its computational properties.

2 Preliminaries

The set of integers $\{1, \dots, k\}$ is denoted $[k]$. For a set X, its powerset is denoted $\mathbb{P}(X)$, and the set of finite (resp. infinite) sequences of elements from X is denoted X^* (resp. X^ω). The empty sequence is written ϵ. For a sequence β, we write β_i for its ith element; the first element is β_0; the length of a finite sequence is $|\beta|$; the prefix of β of length i is denoted by $\beta_{<i}$ or $\beta_{\leq i-1}$; we write $\mathit{Inf}(\beta)$ for the set of elements appearing infinitely often in β, i.e., $x \in \mathit{Inf}(\beta)$ iff $x = \beta_i$ for infinitely many i. If β is a prefix of α we also say that α *extends* β.

For a finite set *AP* of *atomic propositions* (aka *atoms*), sequences over $\mathbb{P}(AP)$ are called *traces*. A *trace property (aka property)* is a set $\mathcal{P} \subseteq \mathbb{P}(AP)^\omega$ of infinite traces. If $\tau \in \mathcal{P}$ we say that τ *satisfies* $\mathcal{P}$. If $\mathcal{P}$ is non-empty we say that it is *satisfiable*. We will sometimes use logical operators for operations on trace properties. In particular, for $A, B \subseteq \mathbb{P}(AP)^\omega$, we may write $\neg A$ for A's complement ($\mathbb{P}(AP)^\omega \setminus A$), write $A \vee B$ for the union $A \cup B$, write $A \wedge B$ for the intersection $A \cap B$, and write $A \Rightarrow B$ for $\neg A \vee B$, i.e. for $(\mathbb{P}(AP)^\omega \setminus A) \cup B$.

2.1 Linear-time Temporal Logic (LTL)

Let *AP* be a set of atoms. The *formulas of LTL over AP* are defined by the following BNF (where $p \in AP$):

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

We use the usual abbreviations, $\varphi \supset \varphi' \doteq \neg\varphi \vee \varphi'$, $\mathsf{true} \doteq p \vee \neg p$, $\mathsf{F}\,\varphi \doteq \mathsf{true}\,\mathsf{U}\,\varphi$, $\mathsf{G}\,\varphi \doteq \neg\,\mathsf{F}\,\neg\varphi$, etc. The *size* $|\varphi|$ of a formula φ is the number of symbols in it. A *trace* τ is an infinite sequence of valuations of the atoms (we say that *the trace is over AP*, to emphasize the atoms). For $n \geq 0$, write τ_n for the valuation at position n. Given a trace τ, an integer n, and an LTL formula φ, the satisfaction relation $(\tau, n) \models \varphi$, stating that φ holds at step n of the sequence τ, is defined as follows: $(\tau, n) \models p$ iff $p \in \tau_n$; $(\tau, n) \models \varphi_1 \vee \varphi_2$ iff $(\tau, n) \models \varphi_1$ or $(\tau, n) \models \varphi_2$; $(\tau, n) \models \neg\varphi$ iff it is not the case that $(\tau, n) \models \varphi$; $(\tau, n) \models \mathsf{X}\,\varphi$ iff $(\tau, n+1) \models \varphi$; and $(\tau, n) \models \varphi_1 \,\mathsf{U}\, \varphi_2$ iff there exists $m \geq n$ such that: $(\tau, m) \models \varphi_2$ and $(\tau, j) \models \varphi_1$ for all $n \leq j < m$. We write $\tau \models \varphi$ if $(\tau, 0) \models \varphi$, read τ *satisfies* φ. Note that the set of traces that satisfy φ is a trace property, and we will sometimes blur the distinction between the formula φ and this trace property.

2.2 Automata and Games

Transition systems. A *deterministic transition system* $D = (\Sigma, Q, \iota, \delta)$ consists of a finite *input alphabet* Σ (typically $\Sigma = \mathbb{P}(AP)$), 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 $q_0 q_1 \cdots$ 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: $\delta(q, \epsilon) = q$, and for $n > 0$, let $\delta(q, \alpha_0\alpha_1 \cdots \alpha_n) = \delta(\delta(q, \alpha_0 \cdots \alpha_{n-1}), \alpha_n)$. Given k deterministic transition systems $T_1, \dots, T_k$ over the same alphabet, i.e., $T_i = (\Sigma, Q_i, \iota_i, \delta_i)$, for $i \in [k]$, their *product* $T_1 \times T_2 \ldots \times T_k$ is the deterministic transition system $(\Sigma, Q', \iota', \delta')$ where $Q' = Q_1 \times \ldots \times Q_k$, $\iota' = (\iota_1, \dots, \iota_k)$, and $\delta((q_1, \dots, q_k), a) = (\delta_1(q_1, a), \dots, \delta_k(q_k, a))$.

Automata. A *deterministic automaton* (or simply *automaton*) $\mathcal{A} = (D, Acc)$ is a deterministic transition system $D = (\Sigma, Q, \iota, \delta)$ augmented with an *acceptance condition* *Acc*. The acceptance condition can be thought of as a subset of Q^ω, though it is usually not given as a set, but indirectly using some other mechanism as we describe below.

An infinite string α over Σ is *accepted* by $\mathcal{A}$ if its run ρ is in *Acc*, in which case we also say that ρ is *accepting* or that it *satisfies* the acceptance condition. The set of infinite strings accepted by $\mathcal{A}$ is the *language* of $\mathcal{A}$, which we denote by $L(\mathcal{A})$. We focus on Emerson-Lei acceptance conditions, which generalize all traditional acceptance conditions used for ω-regular languages (Emerson and Lei 1987; Hausmann, Lehaut, and Piterman 2024).

Emerson-Lei (EL) acceptance conditions are described by a triple (Γ, λ, B), where Γ is a finite set of *labels*, $\lambda\colon Q \to \mathbb{P}(\Gamma)$ is a *labeling function* that assigns to a state a (possibly empty) subset of labels, and $B\colon \mathbb{P}(\Gamma) \to \{\mathsf{true}, \mathsf{false}\}$ is a Boolean function over the set Γ (treated as variables). For a state $q \in Q$ and a label $l \in \lambda(q)$, we say that q *visits* l. We will sometimes (but not always) write B as a Boolean formula over the set Γ of variables, with the usual syntax and semantics; e.g., the formula $l_1 \wedge l_2 \wedge \neg l_3$ denotes the Boolean function that assigns true to a set $Z \subseteq \Gamma$ iff Z contains l_1 and l_2 but not l_3. Given a sequence $\rho \in Q^\omega$, the set of labels that are visited infinitely many times by states on ρ is $\mathit{Inf}_\lambda(\rho) = \bigcup\{\lambda(q) : q \in \mathit{Inf}(\rho)\}$. A sequence ρ satisfies the *EL acceptance condition* iff $B(\mathit{Inf}_\lambda(\rho)) = \mathsf{true}$, i.e., iff the set of labels that are visited infinitely often by states of ρ satisfies B. An automaton $\mathcal{A} = (D, (\Gamma, \lambda, B))$ with an EL acceptance condition, is called a *DELA (deterministic Emerson-Lei automaton)*.

A very useful property of DELAs is that they are closed under Boolean operations, with very simple constructions (and trivial proofs).

Lemma 1 (DELA closed under Boolean operations)**.**

1. *Given a DELA $\mathcal{A} = (T, (\Gamma, \lambda, B))$, the DELA $(T, (\Gamma, \lambda, \neg B))$ accepts the complement of $L(\mathcal{A})$.*
2. *Given DELA $\mathcal{A}_i = (T_i, (\Gamma_i, \lambda_i, B_i))$ for $i \in [k]$, suppose the Γ_i are pairwise disjoint, and the T_i have the same alphabet. Let $\mathcal{C}$ be the DELA $(T, (\Gamma, \lambda, B))$ where: $T = T_1 \times \cdots \times T_k$, $\Gamma = \cup_{i\in[k]}\Gamma_i$, the labeling function λ is defined by letting $\lambda(q_1, \cdots, q_k) = \bigcup_{i\in[k]} \lambda_i(q_i)$, and $B = \bigvee_{i\in[k]} B_i$ (resp. $\bigwedge_{i\in[k]} B_i$); then $\mathcal{C}$ accepts the language $\bigcup_{i\in[k]} L(\mathcal{A}_i)$ (resp. $\bigcap_{i\in[k]} L(\mathcal{A}_i)$).*

The following Theorem says that one can compile formulas into automata.

Theorem 2 (Formulas to Automata)**.** *(Vardi and Wolper 1994; Safra 1988) Fix AP. Given an LTL formula φ over AP, one can build a DELA $\mathcal{A}_\varphi = (T, (\Gamma, \lambda, B))$ that accepts exactly the traces over AP that satisfy φ, whose size $|T|$ is at most 2EXP in $|\varphi|$, whose number of labels $|\Gamma|$ is at most EXP in $|\varphi|$, and whose function B can be written as a Boolean formula of size linear in the number of labels.*[1]

Two-player games on automata. An *arena* is a tuple $(\mathcal{A}, \mathbf{A}, \mathbf{B})$ where $\mathbf{A}, \mathbf{B}$ are disjoint sets of variables and a $\mathcal{A}$ is an automaton with alphabet $\Sigma = \mathbb{P}(\mathbf{A} \cup \mathbf{B})$. We informally describe a two-player game played on $\mathcal{A}$. The players (aka *opponents*) are called Adam and Eve. Adam controls variables from $\mathbf{A}$ and Eve from $\mathbf{B}$. The game proceeds by the players pushing a pebble along the transition system of $\mathcal{A}$ as follows: if the pebble is currently in q, first Adam moves by selecting $A' \subseteq \mathbf{A}$, then Eve moves by selecting $B' \subseteq \mathbf{B}$, and the position in the game is updated to the state $\delta(q, (A' \cup B'))$. Starting at the initial state, this interaction generates an infinite run, and Adam is declared the winner if the run is accepting.

We now describe this game formally. A *play* $\pi = A_0 \cdot B_0 \cdot A_1 \cdot B_1 \cdots$ is an element of $(\mathbb{P}(\mathbf{A}) \cdot \mathbb{P}(\mathbf{B}))^\omega$; this play is an alternative representation of the trace $(A_0 \cup B_0)(A_1 \cup B_1) \cdots$, *induced* by π. A *history* h is a finite prefix of a play. A *strategy for Adam* is a function $\sigma : (\mathbb{P}(\mathbf{A}) \cdot \mathbb{P}(\mathbf{B}))^* \to \mathbb{P}(\mathbf{A})$ that maps histories ending in Eve's moves (including the empty history since Adam moves first) to an Adam move. A *strategy for Eve* is a function $\sigma : (\mathbb{P}(\mathbf{A}) \cdot \mathbb{P}(\mathbf{B}))^* \cdot \mathbb{P}(\mathbf{A}) \to \mathbb{P}(\mathbf{B})$ that maps histories ending in Adam's moves to an Eve move. A play π is *consistent* with a strategy if for every history $\pi_{<i}$ in the domain of σ we have that $\pi_i = \sigma(\pi_{<i})$. If σ is a strategy for Adam and δ is a strategy for Eve then we write $\textsc{play}(\sigma, \delta)$ for the unique play consistent with both these strategies. We say that a strategy σ for Adam (resp. Eve) is *winning* if for all strategies δ for Eve (resp. Adam) we have that $\textsc{play}(\sigma, \delta)$ is accepted (resp. not accepted) by $\mathcal{A}$. Deciding if a given player has a winning strategy (and returning such a strategy if one exists) is called *solving* the game for that player. To emphasize the arena $(\mathcal{A}, \mathbf{A}, \mathbf{B})$ we may write "solving the game on the arena $(\mathcal{A}, \mathbf{A}, \mathbf{B})$", and if the sets $\mathbf{A}, \mathbf{B}$ are understood, then we may write "solving the game on the automaton $\mathcal{A}$". It will be technically convenient to sometimes blur the distinction between traces and plays. E.g., we say that a play π *satisfies* a trace property $\mathcal{P}$ if the trace induced by π is in $\mathcal{P}$; we say that a trace τ is consistent with a strategy σ if the play that induces τ is consistent with σ.

Theorem 3. *(Hausmann, Lehaut, and Piterman 2024; Aminof et al. 2025) Games on DELAs can be solved in time polynomial in the size of the arena, exponential in the number of labels, and polynomial in the size of the formula.*[2]

3 Synthesis under Assumptions

In nondeterministic planning, one purpose of the planning domain is to specify the rules of conduct that the agent thinks will be obeyed by the environment. Instead of allowing the environment to choose any effect in response to an agent

[1] Usually this result is cited for Rabin Automata and follows two steps: from LTL to Nondeterministic Büchi Automata (Vardi and Wolper 1994), and from such automata to Deterministic Rabin Automata (Safra 1988). To get DELA is very easy: simply convert the Rabin acceptance condition into an Emerson-Lei condition on the same transition systems. This is done by labeling each state by each of the sets in each of the Rabin pairs that it belongs to, and the Boolean formula simply expresses the Rabin condition.

[2] Games on automata, such as ours, are notational variants of traditional games played on bipartite directed-graphs without edge-labels. Indeed, one can convert a game on an automaton into a traditional model by introducing for $q \in Q$ and $A \in \mathbf{A}$, a new vertex q_A, and replacing each edge of the form $\delta(q, A \cup B) = q'$ by an edge from q to q_A (controlled by Adam), and an edge from q_A to q' (controlled by Eve). The label of q_A is usually taken to be the same as that of q.

action, the domain limits the environment's choices (action effects). A generalization of this is to limit the environment to a non-empty set χ of strategies that the agent thinks could be used — indeed, under this view, a planning domain induces the set of environment strategies that resolve the nondeterminism at every step by picking one of the available transitions in the domain. Such a non-empty set χ of environment strategies is the *assumption* that the agent has about its environment, i.e., it is the agent's view of the possible environments it might face. This *strategy view* of the environment unites nondeterministic planning and reactive synthesis (Aminof et al. 2019), and we will follow it here. Observe that viewing the environment as employing strategies does not necessarily imply that the agent considers the environment to be sentient or rational, it simply means that, at every point in time, the next action effect is a function of the history of the interaction between the agent and the environment thus far.

Usually, instead of providing an environment specification χ explicitly as a set, one provides instead an environment-enforceable trace-property $\mathcal{P}$ (often as the set of traces that satisfy an LTL formula) which induces the assumption χ as the set of environment-strategies that enforce $\mathcal{P}$. For example, a planning domain can be specified this way by an LTL formula that expresses that all transitions along a trace are only those allowed by the transition relation of the domain. The *LTL synthesis under assumptions* problem is the following (Aminof et al. 2019): given LTL formulae φ_0 and φ_{env}, find a strategy of the agent such that $\text{PLAY}(\sigma_0, \sigma_{\mathsf{env}})$ satisfies φ_0 for every environment strategy σ_{env} that enforces φ_{env}. While these notions have been given for the case of an agent in a monolithic environment, we now make use of them for environments with agent peers.

3.1 Synthesis under Multi-Agent Environment Assumptions

We consider an agent, called agent 0 that is trying to enforce the LTL goal φ_0. As discussed in the introduction, we will focus on the following natural case regarding the information the protagonist has about its environment. First, that its environment consists of N other agents, called agent 1, agent 2, …, agent N, and the rest of the environment, called env, that all the agents are operating in, e.g., env may capture the relevant physical and communication-infrastructure of the world. We now describe agent 0's specification of its environment, i.e., of the set of strategies that it thinks agents i (for $i \geq 1$) and env may use.

1. Agent 0 assumes that env will use a strategy that enforces the LTL formula φ_{env} (in particular, agent 0 assumes that env makes no assumptions about any of the agents). For this to make sense, φ_{env} should be enforceable by env.
2. Agent 0 assumes that agent i (for $i \geq 1$) will use a strategy that enforces the LTL formula φ_i under the assumption that the environment will enforce φ_{env} (in particular, agent 0 assumes that the other agents make no assumptions about any of the other agents). For this to make sense, φ_i should be enforceable under the assumption that the environment will enforce φ_{env}.

The computational problem is to decide, given $\varphi_0, \cdots, \varphi_N, \varphi_{\mathsf{env}}$, if there is an agent 0 strategy that enforces φ_0 under these stated assumptions, and return such a strategy if there is one. Note that if $N = 0$ we are in the case of a single agent and a monolithic environment, and this computational problem is simply synthesis under assumptions (Aminof et al. 2019).

Remark 1. *Our approach reflects a first-person view from the perspective of the agent 0 (the protagonist). This results in an asymmetry in our formalisation, i.e., agent* 0 *is treated differently to agents* $i \geq 1$*. In contrast, if every agent has an assumption about every other agent, and if we were synthesizing strategies for all the agents, Game-Theoretic solutions concepts based on equilibria would be more appropriate (Fisman, Kupferman, and Lustig 2010; Gutierrez, Harrenstein, and Wooldridge 2015; Kupferman, Perelli, and Vardi 2016; Wooldridge et al. 2016; Gutierrez et al. 2020).*

Fix $N \geq 0$. An *actor* is an element of the set $V = \{0, 1, \cdots, N, \mathsf{env}\}$. The actor env is called the *environment* (referred to as "the world" in the introduction), and the other actors are called *agents*. Let $\mathbf{X}$ and $\mathbf{Y}$ be a partition of *AP* into two disjoint finite sets of Boolean variables, called the *agent variables* and *environment variables*, respectively. We assume that $\mathbf{X}$ is partitioned into $N + 1$ disjoint parts, i.e., $\mathbf{X}^0, \cdots, \mathbf{X}^N$, where $\mathbf{X}^a$ is controlled by agent a. For convenience, we will also write $\mathbf{X}^{\mathsf{env}} = \mathbf{Y}$. Then, $\mathbb{P}(\mathbf{X}^v)$ is the set of actor v's *moves*.

Remark 2. *In our formalisation we will assume that, in each step, all the agents move simultaneously and then* env *responds. We make this choice since it reflects the turn-taking assumption typical in single-agent nondeterministic planning where an agent move is followed by the environment's response (Geffner and Bonet 2013), and because simultaneous moves are quite general (and can be relatively easily weakened to turn-based moves if one wishes).*

A *play* $\pi = X_0 \cdot Y_0 \cdot X_1 \cdot Y_1 \cdots$ is an element of $(\mathbb{P}(\mathbf{X}) \cdot \mathbb{P}(\mathbf{Y}))^\omega$; this play is an alternative representation of the trace $(X_0 \cup Y_0)(X_1 \cup Y_1) \cdots$, *induced* by π. A *history* h is a finite prefix of a play. As in the two-player case, it will be technically convenient to sometimes blur the distinction between traces and plays.

Since the agents move first, and simultaneously, we define their strategies as follows. A strategy for agent a is a function $\sigma_a : (\mathbb{P}(\mathbf{X}) \cdot \mathbb{P}(\mathbf{Y}))^* \to \mathbb{P}(\mathbf{X}^a)$, i.e., it assigns to every history that ends in an environment move (as well as the empty history since the agents move first) an agent a move. A strategy for the environment is a function $\sigma_{\mathsf{env}} : (\mathbb{P}(\mathbf{X}) \cdot \mathbb{P}(\mathbf{Y}))^* \cdot \mathbb{P}(\mathbf{X}) \to \mathbb{P}(\mathbf{Y})$, i.e., it assigns to every history that ends in the agents' moves an environment move. A strategy for actor v is called a v*-strategy*. Let $Z \subseteq V$ be a non-empty set of actors. A *strategy profile for* Z is a function $\bar{\sigma}$, with domain Z, that assigns to every $v \in Z$ a v-strategy $\bar{\sigma}(v)$. If Z is the set of all actors, then we call $\bar{\sigma}$ a *full strategy profile*, otherwise it is a *partial strategy profile*. A play or history ρ is *consistent* with a v-strategy σ if for every proper prefix $\rho_{<i}$ in the domain of σ we have that $\rho_i \cap \mathbf{X}^v = \sigma(\rho_{<i})$. Say that ρ is *consistent* with a strategy

profile $\bar{\sigma}$ if it is consistent with every strategy in the range of $\bar{\sigma}$. If $\bar{\sigma}$ is a full strategy profile, write $\text{PLAY}(\bar{\sigma})$ for the unique play consistent with $\bar{\sigma}$.

An *assumption for actor* v is a non-empty set Θ_v of partial strategy profiles for $V \setminus \{v\}$, i.e., for all the actors except v. A common special case of assumptions are *rectangular* ones. An assumption Θ_v is *rectangular* if it is determined by its projections $\mathsf{proj}_w(\Theta_v)$ for $w \neq v$, where $\mathsf{proj}_w(\Theta_v)$ is the set of all w-strategies appearing in profiles in Θ_v. That is, Θ_v is rectangular iff every profile $\bar{\sigma}$, satisfying $\bar{\sigma}(w) \in \mathsf{proj}_w(\Theta_v)$ for every w, is in Θ_v.

We say that σ_v *enforces* $\mathcal{P}$ *assuming* Θ_v if for every $\bar{\sigma} \in \Theta_v$, extending it to a full strategy profile $\bar{\sigma}'$ by letting $\bar{\sigma}'(v) = \sigma_v$ (and $\bar{\sigma}'(w) = \bar{\sigma}(w)$ for $w \neq v$) we have that $\text{PLAY}(\bar{\sigma}')$ satisfies $\mathcal{P}$. Given $\mathcal{P}$ and Θ_v, deciding whether actor v has a strategy that enforces $\mathcal{P}$ assuming Θ_v (and optionally returning such a strategy), is called the *synthesis under assumption* problem for actor v. We observe that in the special case of a single agent (i.e. $N = 0$), such an agent's assumption Θ_0 is simply a set of strategies for the environment. When this (nonempty) set of environment strategies is specified as those that enforce an LTL formula φ_{env}, and the trace property $\mathcal{P}$ is specified as the set of traces satisfying an LTL formula φ, then the definition above coincides with the definition from (Aminof et al. 2019) of synthesis under assumption of φ_o under the assumption φ_{env}. Thus, our definition above is a generalization of the definition of synthesis under assumption from (Aminof et al. 2019).

Basic synthesis problem. We now describe the problem of LTL synthesis under multi-agent environment assumptions. Fix $N \geq 0$, and for every actor $v \in V$ let ϕ_v be an LTL formula.

1. Let Ω_{env} be the set of env-strategies that enforce ϕ_{env}.
2. For every agent $a \neq 0$, let Θ_a be the rectangular assumption where $\mathsf{proj}_w(\Theta_a) = \Omega_{\mathsf{env}}$ if $w = \mathsf{env}$, and otherwise $\mathsf{proj}_w(\Theta_a)$ is the set of all w-strategies. Define Ω_a to be the set of a-strategies that enforce ϕ_a assuming Θ_a.
3. Let Θ_0 be the rectangular assumption for agent 0 defined by letting $\mathsf{proj}_w(\Theta_0) = \Omega_w$ for all w.

We say that *an agent* 0 *strategy enforces* ϕ_0 *under* $\phi_1, \cdots, \phi_N, \phi_{\mathsf{env}}$ if it enforces ϕ_0 assuming Θ_0.

Problem 1 (LTL synthesis under multi-agent environment assumptions)**.** *Given LTL formulas* $\phi_0, \phi_1, \cdots, \phi_N, \phi_{\mathsf{env}}$*, decide if there is an agent* 0 *strategy that enforces* ϕ_0 *under* $\phi_1, \cdots, \phi_N, \phi_{\mathsf{env}}$*, and return such a strategy if there is one.*

4 Technical Intermezzo

When we do not have peer agents, i.e., $N = 0$, our problem reduces to the one studied in (Aminof et al. 2019) where the agent's goal is given by an LTL formula φ_0, and the environment assumption is given by an environment-enforceable LTL formula φ_{env}. In this case, it is known (Aminof et al. 2019) that the following are equivalent: there is an agent strategy enforcing φ_0 under the assumption φ_{env}, iff there is an agent strategy enforcing the LTL formula $\varphi_{\mathsf{env}} \Rightarrow \varphi_0$. In other words, synthesis under assumptions and synthesis of the implication formula are equirealizable. In fact, looking for an agent strategy enforcing the formula $\varphi_{\mathsf{env}} \Rightarrow \varphi_0$ has been the standard approach for many years for handling assumptions. Only later it was observed and argued (Aminof et al. 2019; Aminof, De Giacomo, and Rubin 2021) that this logical implication, at the level of the LTL specification, may not be the correct general approach for synthesis under assumptions, and the strategy-based view was promoted.

It is important to note that even in this simple case, of a single agent, synthesis under assumption and synthesizing for the implication $\varphi_{\mathsf{env}} \Rightarrow \varphi_0$ are equirealizable, but *not* equivalent, problems. That is, while every strategy that enforces $\varphi_{\mathsf{env}} \Rightarrow \varphi_0$ also enforces φ_0 under the assumption φ_{env}, the converse is not true (Aminof et al. 2019). However, one can obtain equivalence, and not merely equirealizabilty, if one considers, as the agent's goal, instead of the LTL implication formula $\varphi_{\mathsf{env}} \Rightarrow \varphi_0$, the following more general implication of trace properties (Aminof et al. 2024): "if a trace is consistent with an environment strategy that enforces ψ then it also satisfies φ". We now generalize this result to the multi-agent setting.

We start by formalizing some key notions. Let σ be a v-strategy, for some actor $v \in V$. A play π extending a history h is *consistent with* σ *from* h if for every $i \geq |h|$, if $\pi_{<i}$ is in the domain of σ then $\pi_i = \sigma(\pi_{<i}) \cap \mathbf{X}^v$. A play π is *consistent with a strategy profile* $\bar{\sigma}$ *from* h if it is consistent from h with every strategy in the range of $\bar{\sigma}$. For a full strategy profile $\bar{\sigma}$, the $\bar{\sigma}$*-extension of* h is the unique play $\text{PLAY}(\bar{\sigma})_h$ such that (i) the play $\text{PLAY}(\bar{\sigma})_h$ extends h, and (ii) for every actor v, the play $\text{PLAY}(\bar{\sigma})_h$ is consistent with the v-strategy $\bar{\sigma}(v)$ from h (note that this definition does not require h or any of its prefixes to be consistent with $\bar{\sigma}$). Intuitively, $\text{PLAY}(\bar{\sigma})_h$ is the result of playing the strategies in $\bar{\sigma}$ starting from h. Given a trace property $\mathcal{P}$, an actor v, and a v-strategy σ_v, we say that σ_v *enforces* $\mathcal{P}$ *from* h if for every full strategy profile $\bar{\sigma}$ with $\bar{\sigma}(v) = \sigma_v$ (i.e., in which actor v uses σ_v), we have that the $\bar{\sigma}$-extension of h satisfies $\mathcal{P}$. If h is the empty sequence, then the definition of consistent with σ from h (resp. $\text{PLAY}(\bar{\sigma})_h$, resp. enforces $\mathcal{P}$ from h) coincides with the definition of consistent (resp. $\text{PLAY}(\bar{\sigma})$, resp. enforces $\mathcal{P}$) from Section 3.1.

Remark 3. *For every actor* v*, every trace* τ *is consistent with* some *strategy profile for* $V \setminus \{v\}$*. For example, the strategy profile* $\bar{\sigma}$ *that at time* n*, regardless of what the actual history is, outputs the relevant portions of the* n*'th letter of* τ*. More formally, for* $w \neq v$*, the strategy* $\bar{\sigma}(w)$ *is defined, for a history* $h = X_0 \cdot Y_0 \cdots X_{n-1} \cdot Y_{n-1} \cdot z$ *(where* $z \subseteq \mathbf{X}$ *are the agents' moves if* $w = \mathsf{env}$*, and otherwise* z *is the empty word) to be* $\bar{\sigma}(w)(h) = \tau_n \cap \mathbf{X}^w$*. This means, in particular, that a* v*-strategy* σ_v *enforces* $\mathcal{P}$ *(resp. from* h*) iff every play consistent with* σ_v *(resp. from* h*) is in* $\mathcal{P}$*.*

Observe that the setting given above captures, in particular, the dynamics and the definitions of histories, strategies, and enforcement of the two-player games on automata from Section 2.2 by taking $N = 0$, agent 0 being identified with Adam, and env identified with Eve.

The set of traces consistent with a strategy σ (resp. strategy profile $\bar{\sigma}$) is denoted $\mathrm{Cns}(\sigma)$ (resp. $\mathrm{Cns}(\bar{\sigma})$). Extend this to sets of strategies, and sets of strategy profiles, in the natural way: e.g., for a set Ξ of strategy profiles we have $\mathrm{Cns}(\Xi) \doteq \bigcup_{\bar{\sigma}\in\Xi} \mathrm{Cns}(\bar{\sigma})$. Write $\mathrm{Enf}_v(\mathcal{P})$ (resp. $\mathrm{Enf}_v(\varphi)$) for the set of v-strategies that enforce $\mathcal{P}$ (resp. the set of traces that satisfy the LTL formula φ), and say that $\mathcal{P}$ (resp. φ) is *v-enforceable* if this set of strategies is not empty.

We are now ready to generalize the result in (Aminof et al. 2024).

Theorem 4. *Given an actor v, a trace property $\mathcal{P}$, and an assumption Θ_v for v, a v-strategy σ_v enforces $\mathcal{P}$ assuming Θ_v iff σ_v enforces $Cns(\Theta_v) \Rightarrow \mathcal{P}$ (i.e., the set of v-strategies that enforce $\mathcal{P}$ assuming Θ_v is $Enf_v(Cns(\Theta_v) \Rightarrow \mathcal{P})$).*

Proof. For the first direction, assume that σ_v enforces $\mathcal{P}$ assuming Θ_v, and let π be a play consistent with σ_v. If π is consistent with some $\bar{\sigma} \in \Theta_v$ then it follows that π is consistent with the full strategy profile $\bar{\sigma}'$ obtained by letting $\bar{\sigma}'(v) = \sigma_v$, and $\bar{\sigma}'(w) = \bar{\sigma}(w)$ for $w \neq v$. Hence, by our assumption on σ_v, it follows that $\pi \in \mathcal{P}$, and thus π satisfies $\mathrm{Cns}(\Theta_v) \Rightarrow \mathcal{P}$. If, on the other hand, π is not consistent with any $\bar{\sigma} \in \Theta_v$, then $\pi \notin \mathrm{Cns}(\Theta_v)$, so it again satisfies $\mathrm{Cns}(\Theta_v) \Rightarrow \mathcal{P}$.

For the other direction, assume that σ_v enforces $\mathrm{Cns}(\Theta_v) \Rightarrow \mathcal{P}$, take some $\bar{\sigma} \in \Theta_v$, and its extension $\bar{\sigma}'$ to a full strategy profile obtained by letting $\bar{\sigma}'(v) = \sigma_v$ and $\bar{\sigma}'(w) = \bar{\sigma}(w)$ for $w \neq v$. Observe that $\pi = \mathrm{PLAY}(\bar{\sigma}') \in \mathrm{Cns}(\Theta_v)$, and that π satisfies $\mathrm{Cns}(\Theta_v) \Rightarrow \mathcal{P}$ by our assumption on σ_v. It follows that π satisfies $\mathcal{P}$. □

Intuitively, the theorem above says that σ_v achieves $\mathcal{P}$ playing against any strategy profile in Θ_v iff: for every play π consistent with σ_v, if π is consistent with some strategy profile in Θ_v then π satisfies $\mathcal{P}$. This allows us to transform the synthesis under assumption problem for actor v to an equivalent synthesis problem with no assumption (by "folding" the assumption into the new trace property for v). Observe that this has the important advantage of turning a problem that talks about strategy profiles into one that deals with only trace properties. Nonetheless, one still needs to *reason* about strategy profiles in order to capture the set of traces $\mathrm{Cns}(\Theta_v)$.

It is interesting to note that there are usually infinitely many traces consistent with any strategy profile in Θ_v, and that $\mathrm{Cns}(\Theta_v)$ "throws" together all these traces for all the strategy profiles in Θ_v into one bag, without keeping track of which traces are consistent with which strategy profiles. Somewhat surprisingly, the theorem above shows that this lost information is not needed in order to solve the synthesis under assumption problem.

Corollary 5. *Fix an actor v, a trace-property $\mathcal{G}$, w-enforceable trace properties $\mathcal{P}_w$ for $w \neq v$, and a rectangular assumption Θ for v such that $\mathsf{proj}_w(\Theta) = Enf_w(\mathcal{P}_w)$ for $w \neq v$. A v-strategy enforces $\mathcal{G}$ assuming Θ iff it enforces $(\wedge_{w\neq v} Cns(Enf_w(\mathcal{P}_w))) \Rightarrow \mathcal{G}$.*

In case $N = 0$ and $w = \mathsf{env}$, the set $\mathrm{Cns}(\mathrm{Enf}_w(\mathcal{P}_w))$ of traces has been studied and called the *core* of $\mathcal{P}_w$ (Aminof et al. 2024) [3]. We define the multi-agent analogue:

Definition 1. *For an actor v and a trace property $\mathcal{P}$, we call $Cns(Enf_v(\mathcal{P}))$ the* v-core of $\mathcal{P}$, *written* $\mathrm{Core}_v(\mathcal{P})$.

Recall from Section 2 that an LTL formula ϕ is often used to refer to the set of traces that satisfy it. Consequently, we may sometimes write $\mathrm{Core}_v(\phi)$. Similarly, recall from Section 3.1 that we will blur the distinction between a play and the trace that it induces. The following characterization of the v-core will be used in the proof of correctness of our synthesis algorithm:

Theorem 6 (Characterization of the v-core)**.** *Fix a trace property $\mathcal{P}$ and an actor v. A play π is in the v-core of $\mathcal{P}$ iff: (i) π satisfies $\mathcal{P}$; and (ii) for every proper prefix h of π, there is a v-strategy that enforces $\mathcal{P}$ from h.*

Proof. For the first direction, assume that π is in the v-core of $\mathcal{P}$, and let σ_v be a strategy witnessing this, i.e., a v-strategy enforcing $\mathcal{P}$ with which π is consistent. Since π is consistent with a strategy enforcing $\mathcal{P}$ it satisfies $\mathcal{P}$, and thus item (i) holds. For item (ii), given a proper prefix h of π, observe that σ_v enforces $\mathcal{P}$ from h. Indeed, since π is consistent with σ_v, then so is h; hence, for every full strategy profile $\bar{\sigma}$ with $\bar{\sigma}(v) = \sigma_v$, the $\bar{\sigma}$-extension of h is consistent with σ_v, and thus satisfies $\mathcal{P}$.

For the other direction, assume that items (i) and (ii) hold, and for every proper prefix h of π pick a v-strategy σ^h that enforces $\mathcal{P}$ from h. Define a v-strategy σ_v as follows for an h' in its domain: if $h' = \pi_{<i}$ is a prefix of π then define $\sigma_v(h') = \pi_i \cap \mathbf{X}^v$; otherwise, let $\sigma_v(h') = \sigma^h(h')$, where h is the longest common prefix of h' and π. Observe that π is consistent with σ_v and thus, to show that π is in the v-core of $\mathcal{P}$, it suffices to show that every play π' consistent with σ_v satisfies $\mathcal{P}$. If $\pi' = \pi$ then it satisfies $\mathcal{P}$ by item (i). If $\pi' \neq \pi$ then let h be the longest common prefix of π and π', and observe that σ_v and σ^h agree on every history that extends h. It follows that π' is consistent with σ^h from h, and so π' is the $\bar{\sigma}$-extension of h for some full strategy profile $\bar{\sigma}$ with $\bar{\sigma}(v) = \sigma^h$. Hence, by item (ii), π' satisfies $\mathcal{P}$. □

We now show how to reduce the question of whether an actor v enforces an ω-regular property $\mathcal{P}$ to the question of whether a certain player (Eve if $v = \mathsf{env}$, and Adam otherwise) has a winning strategy in a two-player game on a DELA whose language is $\mathcal{P}$.

Definition 2 (Arena induced by a DELA and an actor)**.** *Given an actor v, and a DELA $\mathcal{A}$ with $L(\mathcal{A}) \subseteq \mathbb{P}(AP)^\omega$, define $\mathcal{A}_v = (\mathcal{A}, \mathbf{A}, \mathbf{B})$ to be the arena obtained by partitioning AP into the sets $\mathbf{A}, \mathbf{B}$ as follows: If v is an agent let $\mathbf{A} = \mathbf{X}^v$ and $\mathbf{B} = (\cup_{w\neq v}\mathbf{X}^w)$, and if $v = \mathsf{env}$ let $\mathbf{A} = (\cup_{w\neq v}\mathbf{X}^w)$ and $\mathbf{B} = \mathbf{X}^v$.*

In the definition above, intuitively, if the actor v is an agent then Adam represents v and gets to make its moves (seeing the moves of all the previous rounds), while Eve gets to make the moves for env and all the other agents

[3] Note that here, the notion of core is unrelated to the one of Core Equilibrium (Osborne and Rubinstein 1994), for which a Synthesis problem has been studied in (Gutierrez et al. 2023).

(seeing the moves of all the previous rounds, as well as the move of actor v in this round); in case v is the environment, then Eve represents v and gets to makes its moves, while Adam makes the moves of all the agents. We write $player(v)$ for the player that represents v. Observe that when $v = \mathsf{env}$, the strategies of v in the multi-tier setting are exactly the same (i.e, they are functions with the same domain and co-domain) as the strategies of $player(v)$ (i.e., Eve) in the game over the arena $\mathcal{A}_v$; but when v is an agent the strategies of v and those of $player(v)$ have the same co-domains but not the same domain (since the variables in AP are not partitioned the same way between $\mathbf{A}$ and $\mathbf{B}$ in $\mathcal{A}_v$ as they are partitioned between $\mathbf{X}$ and $\mathbf{Y}$ in the multi-tier setting). However, this is a *mere technicality* since the strings in these two different domains contain exactly the same information, only interleaved differently. Indeed, consider the bijection, between the domain of Adam strategies and the domain of v-strategies, that maps $h = A_0 \cdot B_0 \cdots A_n \cdot B_n$ to $h' = X_0 \cdot Y_0 \cdots X_n \cdot Y_n$, where $X_i = A_i \cup B_i \setminus \mathbf{Y}$ and $Y_i = B_i \cap \mathbf{Y}$, for every i. The bijection between histories also induces a bijection between v-strategies and Adam strategies, that maps a v-strategy $\sigma_v : (\mathbf{X} \cdot \mathbf{Y})^* \to \mathbf{X}^v$ to the Adam strategy $\sigma_A : (\mathbf{A} \cdot \mathbf{B})^* \to \mathbf{A}$ defined by letting $\sigma_A(h) = \sigma_v(h')$ for every history h in the domain of σ_A (note that every Adam strategy in $\mathcal{A}_v$ is of the form σ_A for some v-strategy σ_v). The bijection between histories extends to a bijection between plays $\pi = A_0 \cdot B_0 \cdots$ and $\pi' = X_0 \cdot Y_0 \cdots$, and π is consistent with σ_v iff π' is consistent with σ_A. Moreover, the trace corresponding to π is equal to the trace corresponding to π', because $A_i \cup B_i = X_i \cup Y_i$ for every i. Thus, we have the following fundamental property: a trace τ is consistent with σ_v iff it is consistent with σ_A.

Consequently, regardless if v is the environment or an agent, in the next lemma we blur the distinction between v-strategies and $player(v)$ strategies, and consider them to be the same objects.

The following lemma shows that one can reduce the problem, of deciding whether an actor v can enforce $L(\mathcal{A})$ in the multi-tier setting, to the simple problem of deciding whether $player(v)$ can win the 2-player game on $\mathcal{A}_v$. Intuitively, this is because the ability of v to enforce $L(\mathcal{A})$ is not affected by whether all the other actors can cooperate with each other or not — which is why we can lump them all together into a single opponent to $player(v)$, nor by how much the other actors know of v's past moves or strategy — which is why it does not matter if we let this opponent see moves of $player(v)$ a step earlier then when the corresponding v moves become visible to other actors in the multi-tier setting (due to concurrency). All that matters is that we make sure, at every point in time, that the same information is available to v as to $player(v)$, which is guaranteed by the definitions of $\mathcal{A}_v$, and $player(v)$.

Lemma 7. *Given a multi-tier synthesis setting, an actor v, and a DELA $\mathcal{A}$ with $L(\mathcal{A}) \subseteq \mathbb{P}(AP)^\omega$, there is a v-strategy enforcing $L(\mathcal{A})$ iff $player(v)$ has a winning strategy in the game on the arena $\mathcal{A}_v$.*

Proof. Follows directly from Dfn 2 and Rem 3. □

We now show that the v-core preserves ω-regularity.

Theorem 8. *Given an actor v, and an ω-regular trace property $\mathcal{P}$, the v-core of $\mathcal{P}$ is ω-regular. Moreover, a DELA $\mathcal{A}$ whose language is $\mathcal{P}$ can be converted into a DELA $\mathcal{A}'$ whose language is $\mathrm{Core}_v(\mathcal{P})$ with no blowup.*

Proof. Let $\mathcal{A} = (D, (\Gamma, \lambda, B))$ be a DELA for $\mathcal{P}$, with $D = (\mathbb{P}(AP), Q, \iota, \delta)$. The DELA for $\mathrm{Core}_v(\mathcal{P})$ is obtained, intuitively, by redirecting transitions from a designated subset of states $W \subseteq Q$ to a new rejecting sink state $sink$. The set W is the set of states that the automaton reaches after reading prefixes of traces that correspond to histories from which v *cannot* enforce $\mathcal{P}$. We begin by introducing the tools needed to formally define the set W.

Recall that $AP = \mathbf{X} \cup \mathbf{Y}$, and that a history $h = X_0 \cdot Y_0 \cdots X_n \cdot Y_n$ in the multi-tier setting induces the finite trace $trace(h) = (X_0 \cup Y_0) \cdots (X_n \cup Y_n) \in \mathbb{P}(AP)^*$. Let $state(h)$ denote the last state on the run of $\mathcal{A}$ on $trace(h)$. We claim that: (†) if h, h' are two histories such that $state(h) = state(h')$ then v can enforce $\mathcal{P}$ from h iff it can enforce $\mathcal{P}$ from h'. To prove this claim, let σ be a v-strategy enforcing $\mathcal{P}$ from h, and let σ' be the v-strategy defined by letting $\sigma'(h' \cdot w) = \sigma(h \cdot w)$ for every $h' \cdot w$ in the domain of σ' (σ' can be defined arbitrarily on histories that do not extend h'). Pick any play $h' \cdot \pi$ consistent with σ' from h and observe that $h \cdot \pi$ is consistent with σ from h, and thus the trace $trace(h) \cdot \beta$ induced by it satisfies $\mathcal{P}$. Conclude that, since $L(\mathcal{A}) = \mathcal{P}$, the run ρ of $\mathcal{A}$ on $trace(h) \cdot \beta$ is accepting. Consider now the trace $trace(h') \cdot \beta$ induced by $h' \cdot \pi$, and observe that the run ρ' of $\mathcal{A}$ on it is must also be accepting. Indeed, whether a run of a DELA is accepting or not does not depend on any finite prefix of the run, and by our assumption that $state(h) = state(h')$ we know that after reading h' the automaton is in the same state as after reading h, and thus ρ' proceeds in exactly the same way while reading β as does ρ. This concludes the proof of (†).

For $q \in Q$ say that *v can enforce $\mathcal{P}$ from q* if v can enforce $\mathcal{P}$ from h for some h with $state(h) = q$. Let $W \subseteq Q$ be the set of states from which v cannot enforce $\mathcal{P}$.

The automaton $\mathcal{A}' = (D', (\Gamma \cup \{\bot\}, \lambda', B'))$, with $D' = (\mathbb{P}(AP), Q \cap \{sink\}, \iota, \delta')$, is defined as follows. The transition function δ' is defined by setting $\delta'(q, \sigma) = sink$ for every $q \in W \cup \{sink\}$ and every σ, and setting $\delta'(q, \sigma) = \delta(q, \sigma)$ otherwise (i.e., it is identical to δ except that it redirects all edges from states in W into a sink state); the labelling function λ' is defined by setting $\lambda'(q) = \lambda(q)$ for every $q \in Q$, and $\lambda'(sink) = \bot$; finally, the Boolean function B' is $B \wedge \neg\bot$ (which makes every run that gets stuck in the sink state rejecting).

We now show that the language of $\mathcal{A}'$ is $\mathrm{Core}_v(\mathcal{P})$. We do this by showing that a play π induces a trace τ accepted by $\mathcal{A}'$ iff it satisfies the conditions (i) and (ii) in Theorem 6 (Characterization of the v-core). For the first direction, assume that τ is accepted by $\mathcal{A}'$. It follows that the run ρ of $\mathcal{A}'$ on τ never reaches a state in W. Since $\mathcal{A}'$ behaves identically to $\mathcal{A}$ as long as states in $W \cup \{sink\}$ are not visited, then ρ is also an accepting run of $\mathcal{A}$ on τ, and thus: (a) since $L(\mathcal{A}) = \mathcal{P}$, condition (i) holds; (b) since ρ only visits

states from which v can enforce $\mathcal{P}$, then by (†) also condition (ii) holds. For the other direction, assume that the two conditions in Theorem 6 hold. By condition (ii) and (†), it follows that the run ρ of $\mathcal{A}$ on τ never reaches a state in W, and thus ρ is also a run of $\mathcal{A}'$ on τ. Furthermore, by the first condition, ρ is an accepting run of $\mathcal{A}$, and thus also (by construction) of $\mathcal{A}'$. □

The following theorem supplies the time-complexity of transforming an automaton $\mathcal{A}$ into one for $\mathrm{Core}_v(L(\mathcal{A}))$.

Theorem 9. *Given a DELA $\mathcal{A} = (D, (\Gamma, \lambda, B))$ and an actor v, a DELA $\mathcal{A}'$ for $\mathrm{Core}_v(L(\mathcal{A}))$ can be built in time polynomial in the size of D, exponential in the number $|\Gamma|$ of labels, and polynomial in the size of B.*

Proof. The proof of Theorem 8 defines a DELA $\mathcal{A}'$ for $\mathrm{Core}_v(L(\mathcal{A}))$. The cost of doing this is dominated by the cost of computing the set $W \subseteq Q$ which we analyze below.

For every $q \in Q$, let $\mathcal{A}^q$ be the DELA formed from $\mathcal{A}$ by making q the initial state. We claim that the following are equivalent for a state $q \in Q$: (1) actor v can enforce $L(\mathcal{A})$ from some history h with $state(h) = q$ (i.e., $q \in Q \setminus W$); (2) $player(v)$ can win the game on the arena $(\mathcal{A}^q)_v$ obtained from $\mathcal{A}^q$ by Definition 2. We now prove this claim.

Suppose, by (1), that a v-strategy σ enforces $L(\mathcal{A})$ from h with $state(h) = q$. Define a v-strategy σ' as follows: $\sigma'(h') = \sigma(h \cdot h')$ for all h'. It is sufficient to show that σ' enforces $L(\mathcal{A}^q)$, from which it follows by Lemma 7 that σ' wins the game on the arena $(\mathcal{A}^q)_v$, which gives (2). Thus, let π be a play consistent with σ'. We must show that π is in $L(\mathcal{A}^q)$. By the definition of σ' we have that $h \cdot \pi$ is consistent with σ from h. Thus, by our supposition that σ enforces $L(\mathcal{A})$ from h, we have that $h \cdot \pi \in L(\mathcal{A})$, i.e., the run ρ in the DELA $\mathcal{A}$ on the input $h \cdot \pi$ is accepting. Since by (1) we know that $state(h) = q$ then the run ρ' of $\mathcal{A}^q$ on π is a suffix of ρ. Since the EL acceptance condition is prefix independent[4] then ρ' is also accepting, i.e., $\pi \in L(\mathcal{A}^q)$.

For the other direction, assume by (2) that σ is a $player(v)$-strategy that wins the game on the arena $(\mathcal{A}^q)_v$. By Lemma 7 (recall the note before it saying that we can identify $player(v)$-strategies with v-strategies), σ is a v-strategy that enforces $L(\mathcal{A}^q)$. Let h be any history such that $state(h) = q$. Define a v-strategy σ' as follows: for a history of the form $h \cdot h'$ define $\sigma'(h \cdot h') = \sigma(h')$, and otherwise define σ' arbitrarily. We will prove that σ' enforces $L(\mathcal{A})$ from h, which gives (1). To that end, let $\pi = h \cdot \pi'$ be a play that is consistent with σ' from h. Observe that π' is consistent with σ and thus, since we assumed that σ enforces $L(\mathcal{A}^q)$, then $\pi' \in L(\mathcal{A}^q)$, i.e., the run ρ' of $\mathcal{A}^q$ on π' is accepting. By the prefix-independence of the EL condition, and since $state(h) = q$, conclude that the run ρ of $\mathcal{A}$ on π is also accepting, i.e., $\pi \in L(\mathcal{A})$.

This proves the claim. So, $q \in W$ iff $player(v)$ cannot win the game on the arena $(\mathcal{A}^q)_v$. Thus, we can compute W by solving, for each state $q \in Q$, the game on the arena $(\mathcal{A}^q)_v$. Using that the size of $\mathcal{A}^q$ is the same as the size of $\mathcal{A}$, the time complexity follows by Theorem 3. □

[4] An acceptance condition is *prefix independent* if for every sequence of states, if some suffix of it (not necessarily proper) satisfies the condition then every suffix of it does. This is obviously the case for an EL condition since acceptance only depends on the set of states that occur infinitely often on a run.

5 Solving the Synthesis Problem

In this section we present our solution to the LTL synthesis under multi-agent environment assumptions problem (Problem 1).

Our solution is in three steps. In Step (A) we define a trace property $\mathcal{O}$ such that an agent 0 strategy enforces ϕ_0 under the assumption Θ_0 iff it enforces $\mathcal{O}$. This trace property will be given by an expression using Boolean operations and core operations Core_w (for $w \neq 0$) over the trace properties corresponding to the formulas $\phi_0, \cdots, \phi_N, \phi_{\mathsf{env}}$. In Step (B), by traversing the parse-tree for the expression defining $\mathcal{O}$, we build a DELA $\mathcal{A}$ whose language is $\mathcal{O}$. Finally, in Step (C), we define a 2-player game on $\mathcal{A}$ such that winning strategies for Adam in this game correspond to 0-strategies enforcing $\mathcal{O}$; by solving this game for Adam we solve the synthesis problem.

Step (A). Let $\mathcal{O}$ be the expression

$$(\wedge_{w \neq 0} \Psi_w) \Rightarrow \phi_0 \tag{1}$$

where Ψ_{env} is $\mathrm{Core}_{\mathsf{env}}(\phi_{\mathsf{env}})$ and Ψ_w is $\mathrm{Core}_w(\mathrm{Core}_{\mathsf{env}}(\phi_{\mathsf{env}}) \Rightarrow \phi_w)$ for $w > 0$.

Lemma 10. *An agent 0 strategy enforces ϕ_0 under assumption Θ_0 iff it enforces $\mathcal{O}$.*

Proof. By Corollary 5, an agent 0 strategy enforces ϕ_0 under assumption Θ_0 iff it enforces $(\wedge_{w \neq 0} \mathsf{Cns}(\mathsf{proj}_w(\Theta_0))) \Rightarrow \phi_0$. It remains to show that (*): $\mathsf{Cns}(\mathsf{proj}_{\mathsf{env}}(\Theta_0)) = \mathrm{Core}_{\mathsf{env}}(\phi_{\mathsf{env}})$, and for $w > 0$ that (**): $\mathsf{Cns}(\mathsf{proj}_w(\Theta_0)) = \mathrm{Core}_w(\mathrm{Core}_{\mathsf{env}}(\phi_{\mathsf{env}}) \Rightarrow \phi_w)$. Recall from the definition of Θ_0 from Section 3.1 that $\mathsf{proj}_{\mathsf{env}}(\Theta_0) = \mathsf{Enf}_{\mathsf{env}}(\phi_{\mathsf{env}})$ and thus, by the definition of core (Definition 1), equation (*) holds. For equation (**), recall that $\mathsf{proj}_w(\Theta_0)$ is the set of w-strategies that enforce ϕ_w assuming Θ_w. Hence, by Theorem 4, we get (†): $\mathsf{proj}_w(\Theta_0) = \mathsf{Enf}_w(\mathsf{Cns}(\Theta_w) \Rightarrow \phi_w)$. Recall that Θ_w is a rectangular assumption where $\mathsf{proj}_{\mathsf{env}}(\Theta_w) = \mathsf{proj}_{\mathsf{env}}(\Theta_0)$, and $\mathsf{proj}_a(\Theta_w)$ is the set of all a-strategies, for every agent a. Thus, $\mathsf{Cns}(\Theta_w) = \mathsf{Cns}(\mathsf{proj}_{\mathsf{env}}(\Theta_0))$ since for each agent a, every trace is consistent with some a-strategy. By equation (*), conclude that $\mathsf{Cns}(\Theta_w) = \mathrm{Core}_{\mathsf{env}}(\phi_{\mathsf{env}})$. Equation (**) now follows by applying Definition 1 to (†). □

Step (B). For every $v \in V$, let $\mathcal{A}_{\phi_v} = (D_v, (\Gamma_v, \lambda_v, B_v))$ be a DELA for ϕ_v (by Theorem 2). Note that the automata have the same alphabet, i.e., $\mathbb{P}(AP)$. We suppose, without loss of generality, that the labels sets Γ_v are pairwise disjoint (simply rename if needed). In order to build an automaton for the trace property $\mathcal{O}$, we traverse, bottom-up, the parse-tree of the expression defining $\mathcal{O}$.[5] When the traversal visits a node it constructs a DELA for the trace property of the

[5] One may optimize, e.g., the automaton for ϕ_{env} appears in many leaves, and one need only take it once. Since such optimizations do not change the complexity analysis, we will not explicitly do this here.

expression rooted at this node using the DELAs previously constructed for its child nodes, as follows: for a leaf of the parse-tree labeled ϕ_v, the corresponding DELA is simply $\mathcal{A}_{\phi_v}$; for an internal node labeled by a Boolean operation the corresponding DELA is obtained using Lemma 1; and for an internal node labeled by a core operation, the corresponding DELA is obtained using Theorem 8. This process eventually results in a DELA $\mathcal{A} = (D, (\Gamma, \lambda, B))$ for the root of the parse-tree. Hence, by construction, we get the following:

Lemma 11. *The language of the automaton $\mathcal{A}$ is the set of traces that satisfy $\mathcal{O}$.*

Step (C). Apply Definition 2 to $\mathcal{A}$ and $v = 0$, to get the arena $\mathcal{A}_0$. By Lemma 7, we can solve our synthesis problem by solving the game on $\mathcal{A}_0$ for Adam.

Theorem 12. *The LTL synthesis under multi-agent environment assumptions problem is 2EXPTIME-complete.*

Proof. For the cost of step (B), we first calculate the size of the DELA constructed for each node in the parse-tree of $\mathcal{O}$. Let z be the size of the largest input LTL formula. By Thm. 2, for each leaf, the associated DELA has at most 2EXP in z many states, and an acceptance condition whose size (both in terms of the number of labels and in terms of the size of the Boolean formula) which is 1EXP in z. By Thm. 8, an internal node representing a core operation has a DELA whose size is essentially the same as that of it's child node. By Lemma 1, an internal node representing a Boolean operation has a DELA whose number of states is a product of the number of states of its child DELAs, and whose number of labels (resp. Boolean formula) is linear in the sum of the number of labels (resp. in the sizes of the Boolean formulas) of its child DELAs. Hence, overall, we get that the number of states of any constructed DELA (and thus in particular that of the DELA $\mathcal{A}$ constructed for $\mathcal{O}$ at the root node) is at most the number of states of the biggest leaf DELA raised to the power K^H, where K is the maximal arity of any Boolean operation in $\mathcal{O}$, and H is the depth of the parse tree. Since K is linear in N, and H is constant in our setting[6], each constructed DELA has at most 2EXP many states, 1EXP many labels, and a Boolean formula of size at most 1EXP in the size of the input.

It is easy to see that for leaf nodes, as well as for Boolean operations nodes, the time spent in step (B) constructing the corresponding DELA is linearly related to the size of this DELA. By Theorem 9, and the upper-bound for the sizes of the DELAs given above, we can deduce that the time needed to construct a DELA for a core operation node is at most 2EXP in the size of the input. Hence, since the size of the parse tree of $\mathcal{O}$ is polynomial in the input size, the total time spent in step (B) is 2EXP in the input size.

The cost of producing $\mathcal{O}$ in step (A) of the algorithm is clearly polynomial in the size of the input formulas (and the definition of the synthesis problem). Combining this cost with that of step (B) above, and using Theorem 3 to derive the time for step (C), one gets the required 2EXPTIME upper-bound. For the lower bound, note that taking $N = 0$ is the LTL synthesis under assumption problem, known to be 2EXPTIME-hard (Aminof et al. 2019). □

[6]Note that the overall complexity does not change if H is not constant but considered to be part of the input (or even if it is some polynomial function of the input).

6 Discussion

We considered LTL synthesis where the environment is modeled as a collection of peer agents acting in a shared world. The setting assumes an asymmetry between the protagonist and its peers: the protagonist ascribes specifications to its peers, but believes that the peers do not make any assumptions about the behaviours of any of the agents, and only share his assumption about the world. To address this problem, we proposed a multi-layered approach. In the first layer, we identified the set of traces compatible with the peers' strategies that realize their specifications in the shared world. In the second layer, we synthesized a strategy for the protagonist that satisfies its own specification, given the environment defined by the first-layer specification.

This multi-layered approach can, in fact, be extended to handle more layers, as long as acyclicity is maintained among them. For example, in a three-layer setting, the first layer may capture the set of traces compatible with provisional protagonist strategies that realize its specification, assuming the peers are unconstrained. The second layer then considers the set of traces compatible with the peers' strategies that realize their specifications in an environment defined by the first layer, i.e., assuming the protagonist aims to fulfill its specification, but without assuming the protagonist knows the peers' specifications. Finally, the third layer synthesizes a refined strategy for the protagonist in the environment defined by the second layer. The techniques presented in Section 4 and Section 5 can be applied to this extended setting with minimal modification. Further extending our approach to even more complex (still acyclic) scenarios appears also possible. For example, one could consider the depth of specification nesting individually for each peer agent. This is an interesting direction for future work.

Although our focus has been on LTL specifications, our results seamlessly extend to the recently introduced LTLf+/PPLTL+, which are based on lifting finite-trace properties to infinite traces (De Giacomo and Vardi 2013; Aminof et al. 2025). The reason for this is that those papers show how to compile formulas into DELAs, which is the only fact we use in this paper about LTL. In fact, they also apply to LTLf/PPLTL specifications over finite traces, since these can be easily lifted to equivalent LTLf+/PPLTL+ formulas. The simplicity of these logics on finite traces may lead to simplified automata constructions, which we leave as another promising direction for future work.

Acknowledgements

ERC Advanced Grant WhiteMech (No. 834228), PRIN projects RIPER (No. 20203FFYLK) and PINPOINT (No. B87G22000450001), PNRR MUR project FAIR (No.PE0000013), UKRI Erlangen AI Hub on Mathematical and Computational Foundations of AI, Sapienza U. of Rome under the Progetti Grandi di Ateneo programme, RG123188B3F7414A (ASGARD).

References

Albrecht, S. V., and Stone, P. 2018. Autonomous Agents Modelling other Agents: A Comprehensive Survey and Open Problems. *Artificial Intelligence* 258:66–95.

Aminof, B.; De Giacomo, G.; Murano, A.; and Rubin, S. 2018. Planning and Synthesis Under Assumptions. *CoRR* abs/1807.06777.

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. 2024. Proper Linear-time Specifications of Environment Behaviors in Nondeterministic Planning and Reactive Synthesis. In *KR*.

Aminof, B.; De Giacomo, G.; Rubin, S.; and Vardi, M. 2025. LTLf+ and PPLTL+: Extending LTLf and PPLTL to Infinite Traces. In *IJCAI*.

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

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

Bonassi, L.; De Giacomo, G.; Gerevini, A. E.; and Scala, E. 2024. Shielded FOND: Planning with Safety Constraints in Pure-Past Linear Temporal Logic. In *ECAI*.

Brafman, R. I., and De Giacomo, G. 2019. Planning for LTLf /LDLf Goals in Non-Markovian Fully Observable Nondeterministic Domains. In *IJCAI*.

Brenguier, R.; Raskin, J.; and Sankur, O. 2017. Assume-Admissible Synthesis. *Acta Informatica* 54(1):41–83.

Brenguier, R.; Raskin, J.; and Sassolas, M. 2014. The Complexity of Admissibility in Omega-regular Games. In *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. 2019. Towards a Unified View of AI Planning and Reactive Synthesis. In *ICAPS*.

Cimatti, A.; Pistore, M.; Roveri, M.; and Traverso, P. 2003. Weak, Strong, and Strong Cyclic Planning via Symbolic Model Checking. *Artif. Intell.* 147(1-2):35–84.

Clarke, E. M.; Henzinger, T. A.; Veith, H.; and Bloem, R., eds. 2018. *Handbook of Model Checking*. Springer.

De Giacomo, G., and Rubin, S. 2018. Automata-Theoretic Foundations of FOND Planning for LTLf and LDLf Goals. In *IJCAI*.

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

Emerson, E. A., and Lei, C. 1987. Modalities for Model Checking: Branching Time Logic Strikes Back. *Sci. Comput. Program.* 8(3):275–306.

Fagin, R.; Halpern, J. Y.; Moses, Y.; and Vardi, M. Y. 1995. *Reasoning About Knowledge*. MIT Press.

Fijalkow, N.; Maubert, B.; Murano, A.; Rubin, S.; and Vardi, M. 2022. Public and Private Affairs in Strategic Reasoning. In *KR*.

Finkbeiner, B., and Schewe, S. 2005. Uniform Distributed Synthesis. In *LICS*.

Fisman, D.; Kupferman, O.; and Lustig, Y. 2010. Rational Synthesis. In *TACAS*.

Gabaldon, A. 2011. Non-Markovian Control in the Situation Calculus. *Artif. Intell.* 175(1):25–48.

Geffner, H., and Bonet, B. 2013. *A Concise Introduction to Models and Methods for Automated Planning*. Springer.

Ghallab, M.; Nau, D. S.; Traverso, P.; and Milano, M. 2025. *Acting, Planning, and Learning*. CUP.

Gutierrez, J.; Najib, M.; Perelli, G.; and Wooldridge, M. 2020. Automated Temporal Equilibrium Analysis: Verification and Synthesis of Multi-Player Games. *Artif. Intell.* 287:103353.

Gutierrez, J.; Kowara, S.; Kraus, S.; Steeples, T.; and Wooldridge, M. 2023. Cooperative Concurrent Games. *Artif. Intell.* 314:103806.

Gutierrez, J.; Harrenstein, P.; and Wooldridge, M. 2015. Iterated Boolean Games. *Information and Computation* 242:53–79.

Hausmann, D.; Lehaut, M.; and Piterman, N. 2024. Symbolic Solution of Emerson-Lei Games for Reactive Synthesis. In *FoSSaCS*.

Kupferman, O.; Perelli, G.; and Vardi, M. 2016. Synthesis with Rational Environments. *Ann. Math. Artif. Intell.* 78(1):3–20.

Majumdar, R.; Mallik, K.; Schmuck, A.; and Zufferey, D. 2020. Assume-Guarantee Distributed Synthesis. *IEEE Trans. Comput. Aided Des. Integr. Circuits Syst.* 39(11):3215–3226.

Mogavero, F.; Murano, A.; Perelli, G.; and Vardi, M. 2014. Reasoning About Strategies: On the Model-Checking Problem. *ACM TOCL* 15(4):34:1–34:47.

Osborne, M., and Rubinstein, A. 1994. *A Course in Game Theory.* MIT Press.

Pnueli, A., and Rosner, R. 1990. Distributed Reactive Systems Are Hard to Synthesize. In *FOCS*.

Pnueli, A. 1977. The Temporal Logic of Programs. In *FOCS*.

Safra, S. 1988. On the Complexity of omega-Automata. In *FOCS*.

Schuppe, G. F., and Tumova, J. 2020. Multi-Agent Strategy Synthesis for LTL Specifications through Assumption Composition. In *CASE*.

Vardi, M. Y., and Wolper, P. 1994. Reasoning About Infinite Computations. *Inf. Comput.* 115(1):1–37.

Wooldridge, M. J.; Gutierrez, J.; Harrenstein, P.; Marchioni, E.; Perelli, G.; and Toumi, A. 2016. Rational Verification: From Model Checking to Equilibrium Checking. In *AAAI*.

Wooldridge, M. J. 2009. *An Introduction to MultiAgent Systems, Second Edition*. Wiley.