Act for Your Duties but Maintain Your Rights

Shufang Zhu* , **Giuseppe De Giacomo**
Sapienza University of Rome, Rome, Italy
{zhu,degiacomo}@diag.uniroma1.it

Abstract

Most of the synthesis literature has focused on studying how to synthesize a strategy to fulfill a task. This task is a duty for the agent. In this paper, we argue that intelligent agents should also be equipped with rights, that is, tasks that the agent itself can choose to fulfill (e.g., the right of recharging the battery). The agent should be able to maintain these rights while acting for its duties. We study this issue in the context of LTL_f synthesis: we give duties and rights in terms of LTL_f specifications, and synthesize a suitable strategy to achieve the duties that can be modified on-the-fly to achieve also the rights, if the agent chooses to do so. We show that handling rights does not make synthesis substantially more difficult, although it requires a more sophisticated solution concept than standard LTL_f synthesis. We also extend our results to the case in which further duties and rights are given to the agent while already executing.

1 Introduction

Consider the following example: we give to a robot the task of cleaning one by one series of rooms. The robot has a model of the world describing the effects of its actions, and, given the task specification, it synthesizes a strategy to accomplish its cleaning task. However, in going after its task, the robot would like to be sure to be able to recharge its battery, if it thinks the battery level is getting too low. Both cleaning and recharging batteries are (temporally extended) tasks. Once the cleaning task is accepted, the agent must fulfill it, i.e., the cleaning task is a *duty*. Instead, recharging the battery, is what we may call a *right* of the robot, i.e., a task that the agent must be given the ability to fulfil, such that the agent itself can decide to actually fulfill or not. Handling both *duties* and *rights* is the issue studied in this paper.

The literature on strategy synthesis (Pnueli and Rosner 1989; Finkbeiner 2016), as well as the literature on planning (Ghallab, Nau, and Traverso 2016; Haslum et al. 2019), focus only on fulfilling duties, without considering rights. Instead, our notion of rights is implicitly related to the notion of *ability* studied in autonomous agents and reasoning about actions, see e.g., (Lespérance et al. 2000). Indeed, the ability of performing some task requires the existence of strategies for fulfilling the task, but not necessarily the decision to follow such strategy to actually fulfill it. In our case the agent has the *ability* of satisfying also its *rights* while executing the strategy for satisfying the duties, but actually satisfies the rights only if it wants to do so. Also, talking about duties and rights calls for connections with *obligations* and *permissions* in Deontic Logic (Gabbay et al. 2013). However, here we focus mainly on synthesis and leave the exact connection with Deontic Logic for future studies.

Specifically, in this paper, we study how to handle duties and rights in the context of Linear Temporal Logic on finite traces (LTL_f), see (De Giacomo and Vardi 2013) for a survey. LTL_f, on the one hand, allows for specifying a rich set of temporally extended specifications (Bacchus and Kabanza 2000; de Silva, Meneguzzi, and Logan 2020), and on the other hand, focuses on finite traces, which makes it particularly suitable for specifying tasks of intelligent agents. Note that intelligent agents will not get stuck accomplishing a task for all their lifetime, but only for a finite (but unbounded) number of steps.

Technically, our starting point is LTL_f synthesis under environment specifications (De Giacomo and Vardi 2015; Aminof et al. 2019; De Giacomo et al. 2021a). We assume the agent is acting in an environment that is specified through safety specifications, which can be thought of as an extension, possibly with non-Markovian features (Gabaldon 2011), of nondeterministic fully observable planning domains (Cimatti et al. 2003; Ghallab, Nau, and Traverso 2016), as discussed, e.g., in (Camacho, Bienvenu, and McIlraith 2018; Aminof et al. 2018). Wlog, we are going to use LTL_f, also for these environment safety specifications as in (De Giacomo et al. 2021b). Over this environment, we give duties and rights to the agent, expressing both of them as arbitrary LTL_f specifications. The problem that we want to solve is to synthesize a suitable strategy to achieve the duties that can be modified while in execution to achieve also the rights, if the agent chooses to do so.

We show that handling duties and rights is 2EXPTIME-complete, as standard LTL_f synthesis (De Giacomo and Vardi 2015), though it requires a more sophisticated solution concept. Essentially, we do not only compute the winning strategy as a transducer, but we guarantee that during its execution such a strategy never leaves the *winning region* (which technically captures the ability to fulfil) of both the duties and the rights. Moreover, by storing such winning region, we can readily build a further transducer represent-

*Corresponding Author

ing a strategy to fulfil also the rights at the moment the agent decides to do so while executing the first strategy.

We then study the case in which further duties and rights are given to the agent while the agent is already executing the strategy for the original duties and rights. Handling further duties that are given while already executing a strategy is related to *live synthesis*, which has been recently introduced in Formal Methods (Finkbeiner, Klein, and Metzger 2021). So as a by-product of our work, we devise a technique for live synthesis in LTL_f. We however extend this form of synthesis to handle also rights. We show that, even in this case, synthesis remains 2EXPTIME-complete, and we present techniques to effectively compute such kind of strategies with only a small overhead.

2 Preliminaries

2.1 LTL_f Basics

Linear Temporal Logic on finite traces (LTL_f) is a specification language to express temporal properties on finite traces (De Giacomo and Vardi 2013). In particular, LTL_f has the same syntax as LTL, one of the popular specification languages in Formal Methods, which is interpreted over infinite traces (Pnueli 1977). Given a set of propositions $Prop$, the formulas of LTL_f are generated as follows:

$$\varphi ::= a \mid (\neg\varphi) \mid (\varphi \wedge \varphi) \mid (\bigcirc\varphi) \mid (\varphi\, \mathcal{U}\, \varphi),$$

where $a \in Prop$ is an atom, $\bigcirc$ for *Next*, and $\mathcal{U}$ for *Until* are temporal operators. We make use of standard Boolean abbreviations, such as $\vee$ (or) and $\rightarrow$ (implies), $true$ and $false$. Moreover, we have the following abbreviations for temporal operators, *Eventually* as $\Diamond\varphi \equiv true\, \mathcal{U}\, \varphi$ and *Always* as $\Box\varphi \equiv \neg\Diamond\neg\varphi$. In addition, we have the *Weak Next* operator $\bullet$ as abbreviation of $\bullet\varphi \equiv \neg\bigcirc\neg\varphi$.

A *trace* $\pi = \pi_0\pi_1 \ldots$ is a sequence of propositional interpretations (sets), where for every $i \geq 0$, $\pi_i \in 2^{Prop}$ is the i-th interpretation of π. Intuitively, π_i is interpreted as the set of propositions that are $true$ at instant i. A trace π is an *infinite* trace if $last(\pi) = \infty$, which is formally denoted as $\pi \in (2^{Prop})^\omega$; otherwise π is a *finite* trace, denoted as $\pi \in (2^{Prop})^*$. Moreover, by $\pi^k = \pi_0 \ldots \pi_k$ we denote the *prefix* of π up to the k-th instant. Sometimes we call a prefix of a trace *history*. We denote by ϵ the empty prefix, i.e., the history of length 0. LTL_f formulas are interpreted over finite and nonempty traces. Given $\pi \in (2^{Prop})^+$, we define when an LTL_f formula φ *holds* at instant i ($0 \leq i \leq last(\pi)$), written as $\pi, i \models \varphi$, inductively on the structure of φ, as:

- $\pi, i \models a$ iff $a \in \pi_i$ (for $a \in Prop$);
- $\pi, i \models \neg\varphi$ iff $\pi, i \not\models \varphi$;
- $\pi, i \models \varphi_1 \wedge \varphi_2$ iff $\pi, i \models \varphi_1$ and $\pi, i \models \varphi_2$;
- $\pi, i \models \bigcirc\varphi$ iff $i < last(\pi)$ and $\pi, i+1 \models \varphi$;
- $\pi, i \models \varphi_1\, \mathcal{U}\, \varphi_2$ iff $\exists j$ such that $i \leq j \leq last(\pi)$ and $\pi, j \models \varphi_2$, and $\forall k, i \leq k < j$, we have that $\pi, k \models \varphi_1$.

We say π *satisfies* φ, written as $\pi \models \varphi$, if $\pi, 0 \models \varphi$.

2.2 LTL_f for Safety Properties

Safety properties assert that *undesired things never happen*, i.e., a trace always behaves within some allowed boundaries. Thereby, safety properties exclude traces that can be violated by a "bad" finite prefix. Typically, safety properties are captured as LTL formulas (Kupferman and Vardi 2001), interpreted over infinite traces. Alternatively, it has been shown in (De Giacomo et al. 2021b) that, one can use LTL_f formulas to capture safety properties over both of finite and infinite traces, by applying an alternative notion of satisfaction that interprets an LTL_f formula over *all* prefixes of a trace.

Definition 1. *A (finite or infinite) trace π satisfies an LTL_f formula φ on* all prefixes*, denoted $\pi \models_\forall \varphi$, if every nonempty finite prefix of π satisfies φ. That is, $\pi^k = \pi_0\pi_1 \ldots \pi_k \models \varphi$, for every $0 \leq k \leq last(\pi)$.*

Moreover, all safety properties expressible in LTL, i.e., all first-order (logic) safety properties (Lichtenstein, Pnueli, and Zuck 1985), can be specified using LTL_f on all prefixes.

Theorem 1. *(De Giacomo et al. 2021b) Every first-order safety property can be expressed as an LTL_f formula on all prefixes.*

2.3 LTL_f Synthesis with Safety Env Specs

Reactive synthesis can be viewed as a game between the *environment* and the *agent*, contrasting each other by controlling two disjoint sets of variables $\mathcal{X}$ and $\mathcal{Y}$, respectively. The goal of reactive synthesis is to synthesize an agent strategy such that no matter how the environment behaves, the combined trace from two players satisfy desired properties (Pnueli and Rosner 1989). In standard synthesis, the agent assumes the environment to be free to choose an arbitrary move at each step, but in AI typically the agent has some knowledge of how the environment works. The environment knowledge that the agent knows apriori is called *environment specification* (De Giacomo and Vardi 2015; Aminof et al. 2019; De Giacomo et al. 2021a).

In particular, we focus on the environment specifications that are formed by safety properties. In this way our environment specifications can be thought as an extension of fully observable nondeterministic domains (Cimatti et al. 2003; Ghallab, Nau, and Traverso 2016), see also (Rintanen 2004). Formally, an environment specification is an LTL_f safety formula env, while the agent task is expressed as a standard LTL_f formula φ_{task}. We describe the synthesis problem as a tuple $\mathcal{P} = (env, \varphi_{task})$. Note that for simplicity, we do not explicitly list $\mathcal{X}$ and $\mathcal{Y}$ here, since they are given as inputs by default and thus are clear from the context.

An environment strategy is a function $\gamma : (2^{\mathcal{Y}})^* \rightarrow 2^{\mathcal{X}}$, and an agent strategy is a function $\sigma : (2^{\mathcal{X}})^+ \rightarrow 2^{\mathcal{Y}}$. A trace $\pi = (X_0 \cup Y_0)(X_1 \cup Y_1) \cdots \in (2^{\mathcal{X} \cup \mathcal{Y}})^\omega$, is *compatible* with an environment strategy γ if $\gamma(\epsilon) = X_0$ and $\gamma(Y_0Y_1 \ldots Y_i) = X_{i+1}$ for every i. A trace π being compatible with an agent strategy σ is defined analogously. Sometimes, we write $\sigma(\pi^k)$ instead of $\sigma(X_0X_1 \cdots X_k)$ for simplicity. We denote the unique infinite sequence that is compatible with γ and σ as $\mathsf{Trace}(\gamma, \sigma)$. We also generalize these definitions to finite traces in the obvious way.

Turning to agent strategies, wlog, we require them to be *stopping*, i.e., we require the agent to perform a mandatory stop action. More specifically, every action of the agent is considered as an assignment over $\mathcal{Y}$, and $\mathbf{stop}$ is one of them. For convenience, wlog, $\mathbf{stop}$ is encoded as an

assignment where all variables in $\mathcal{Y}$ are set to $false$, i.e., $\mathbf{stop} = \bigwedge_{y \in \mathcal{Y}} \neg y$.

Definition 2. *(De Giacomo et al. 2021a) A* stopping *agent strategy is a function $\sigma : (2^{\mathcal{X}})^+ \to 2^{\mathcal{Y}}$, such that for every trace $\pi \in (2^{\mathcal{X} \cup \mathcal{Y}})^\omega$ that is compatible with σ, there exists $i \in \mathbb{N}$ such that $\sigma(\pi^j) = \mathbf{stop}$ for every $j \geq i$ and $\sigma(\pi^h) \neq \mathbf{stop}$ for every $h < i$.*

Having *stopping* agent strategies, we define the *play* induced by given γ and σ as the finite prefix of $\mathsf{Trace}(\gamma, \sigma)$ that ends right before the first $\mathbf{stop}$, denoted by $\mathsf{Play}(\gamma, \sigma)$. Formally, $\mathsf{Play}(\gamma, \sigma) = (X_0 \cup Y_0)(X_1 \cup Y_1) \dots (X_i \cup Y_i)$ where $Y_{i+1} = \mathbf{stop}$, and $Y_j \neq \mathbf{stop}$ for every $0 \leq j \leq i$.

Given an environment safety specification env, which is an LTL_f formula, an environment strategy γ *enforces* env, written $\gamma \rhd env$, if for every agent strategy σ, it holds that $\mathsf{Trace}(\gamma, \sigma) \models_\forall env$. We denote the set of environment strategies enforcing env by $[\![env]\!]$.

Definition 3. *(De Giacomo et al. 2021a) The problem of synthesis is described as a tuple $\mathcal{P} = (env, \varphi_{task})$. Realizability of $\mathcal{P}$ checks whether there exists an agent strategy σ such that $\forall \gamma \in [\![env]\!]$, $\mathsf{Play}(\gamma, \sigma) \models \varphi_{task}$. Synthesis of $\mathcal{P}$ computes such a strategy if exists.*

As usual, we require that env must be *environment realizable*, i.e., $[\![env]\!]$ is nonempty. As shown in (De Giacomo et al. 2021a; De Giacomo et al. 2021b), this kind of synthesis can be solved through a reduction to a suitable two-player game constructed from LTL_f formulas env and φ_{task}, which takes 2EXPTIME. The problem itself is 2EXPTIME-complete.

2.4 Two-player Games

A *two-player game* is a game between the *environment* and the *agent*, controlling two disjoint sets of variables $\mathcal{X}$ and $\mathcal{Y}$, respectively. The game is described by a *deterministic automaton* (DA), which is a tuple $\mathcal{A} = (2^{\mathcal{X} \cup \mathcal{Y}}, Q, I, \delta, \alpha)$, where $2^{\mathcal{X} \cup \mathcal{Y}}$ is the alphabet, Q is a finite set of states, $I \in Q$ is the initial state, $\delta : Q \times 2^{\mathcal{X} \cup \mathcal{Y}} \to Q$ is the transition function, and $\alpha \subseteq Q^\omega$ is an acceptance condition. Given an infinite word $\pi = \pi_0 \pi_1 \pi_2 \dots \in (2^{\mathcal{X} \cup \mathcal{Y}})^\omega$, the *run* $\rho = \mathsf{Run}(\mathcal{A}, \pi)$ of $\mathcal{A}$ on π is an infinite sequence $\rho = q_0 q_1 q_2 \dots \in Q^\omega$, where $q_0 = I$ and $q_{i+1} \in \delta(q_i, \pi_i)$ for every $i \geq 0$. The run of $\mathcal{A}$ on a finite prefix π^k is defined analogously, and so $\mathsf{Run}(\mathcal{A}, \pi^k) = q_0 q_1 q_2 \dots q_{k+1}$. A run ρ is *accepting* if $\rho \in \alpha$. The *language* of $\mathcal{A}$, denoted by $\mathcal{L}(\mathcal{A})$, is the set of words accepted by $\mathcal{A}$. In this work, we specifically consider the following acceptance conditions:

- *Reachability.* Given a set $R \subseteq Q$, $\text{Reach}(R) = \{q_0 q_1 q_2 \dots \in Q^\omega \mid \exists k \geq 0 : q_k \in R\}$, i.e., a state in R is visited at least once.
- *Safety.* Given a set $S \subseteq Q$, $\text{Safe}(S) = \{q_0 q_1 q_2 \dots \in Q^\omega \mid \forall k \geq 0 : q_k \in S\}$, i.e., only states in S are visited.
- *Reachability-Safety.* Given two sets $R, S \subseteq Q$, $\text{Reach–Safe}(R, S) = \{q_0 q_1 q_2 \dots \in Q^\omega \mid \exists k \geq 0 : q_k \in R \text{ and } \forall j, 0 \leq j \leq k : q_j \in S\}$, i.e., a state in R is visited at least once, and until then only states in S are visited.

Notably, a DA with reachability acceptance condition defines a deterministic finite automaton (DFA). Depending on the actual acceptance condition α, we get reachability, safety, or reachability-safety games.

Theorem 2. *(De Giacomo et al. 2021b) Reachability-safety game can be solved by a linear-time reduction to a reachability game.*

Given a game $\mathcal{A} = (2^{\mathcal{X} \cup \mathcal{Y}}, Q, I, \delta, \alpha)$ defined above, an agent strategy σ is *winning* if $\forall \gamma. \mathsf{Trace}(\gamma, \sigma) \in \mathcal{L}(\mathcal{A})$. A state $q \in Q$ is an agent (resp. environment) *winning* state if the agent (resp. environment) has a winning strategy in $\mathcal{A}' = (2^{\mathcal{X} \cup \mathcal{Y}}, Q, q, \delta, \alpha)$, i.e., same structure but a new initial state q. By Agn (resp. Env) we denote the set of all agent (resp. environment) winning states, also called the agent (environment) *winning region*. All the games defined above are *determined*, i.e., $q \in Q$ is an agent winning state ($q \in \mathsf{Agn}$) iff q is not an environment winning state ($q \notin \mathsf{Env}$) (Martin 1975).

3 Synthesis with Duties and Rights

In a common synthesis setting, the agent typically follows a strategy blindly. In other words, any action that the agent performs is expected to serve the task. In this paper, we would like to assign more freedom to the agent, and thus look into the scenario where the agent has its own rights of doing some work in its own favor. For example, along the way in cleaning a series of rooms, the robot should remain able to recharge the battery, if it thinks the battery level is getting too low. Note that the robot must make sure that the rooms are cleaned when it stops, no matter whether it chooses or not to recharge the battery while cleaning.

In this synthesis setting, we divide agent tasks into two types: *duties*, expressed as an LTL_f formula φ_{d}, specifying the mandatory tasks that the agent has to accomplish; *rights*, expressed as an LTL_f formula φ_{r}, specifying the optional tasks that the agent has the right to decide whether to accomplish. To make sure that the agent can purse φ_{r} whenever it chooses to do so, the agent should be equipped with the ability of accomplishing also φ_{r} while achieving φ_{d}.

We start with defining a strategy that enforces a specification φ with respect to a history h, indicating the moment that the agent chooses to pursue φ.

Definition 4. *Let φ be an LTL_f formula and $h \in (2^{\mathcal{X} \cup \mathcal{Y}})^*$ be a history. An agent strategy σ* enforces *φ, with respect to history h, denoted by $\sigma \rhd_h \varphi$, if $\forall \gamma \in [\![env]\!]$ such that $\mathsf{Play}(\gamma, \sigma)$ has h as a prefix, we have that $\mathsf{Play}(\gamma, \sigma) \models \varphi$.*

It should be noted that, in Definition 4, we only consider the cases where both the environment strategy γ and the agent strategy σ are compatible with h. That is, $h = (X_0 \cup Y_0)(X_1 \cup Y_1) \dots (X_i \cup Y_i) \in (2^{\mathcal{X} \cup \mathcal{Y}})^*$ is such that for every $0 \leq j \leq i$: $\sigma(X_0 X_1 \cdots X_j) = Y_j$ and $Y_j \neq \mathbf{stop}$; $\gamma(\epsilon) = X_0$ and $\gamma(Y_0 Y_1 \cdots Y_j) = X_{j+1}$.

Computing such a strategy is analogous to computing a strategy that enforces φ. We start with computing $[\![env]\!]$ by taking the following steps:

1. Build $\mathcal{A}_{\text{e}} = (2^{\mathcal{X} \cup \mathcal{Y}}, Q_{\text{e}}, I_{\text{e}}, \delta_{\text{e}}, \text{Safe}(S))$ that accepts a trace π iff $\pi \models_\forall env$.
2. Solve the safety game on $\mathcal{A}_{\text{e}}$ for the environment, thus obtaining the *environment winning region* Env.

3. Restrict $\mathcal{A}_{\mathrm{e}}$ with Env into $\mathcal{A}'_{\mathrm{e}} = (2^{\mathcal{X}\cup\mathcal{Y}}, \mathsf{Env}, I_{\mathrm{e}}, \delta'_{\mathrm{e}}, \mathrm{Safe}(\mathsf{Env}))$, $\delta'_{\mathrm{e}}(q, X \cup Y) =$ undefined, if $\exists Y' \in 2^{\mathcal{Y}}.\delta_{\mathrm{e}}(q, X \cup Y') \notin \mathsf{Env}$; $\delta'_{\mathrm{e}}(q, X \cup Y) = \delta_{\mathrm{e}}(q, X \cup Y)$ otherwise.

It should be noted that for safety games, there exists a unique "nondeterministic" strategy that can capture the set of all winning strategies. This strategy can be intuitively interpreted as a "staying in the winning region" strategy (Bernet, Janin, and Walukiewicz 2002). Therefore, $\mathcal{A}'_{\mathrm{e}}$ precisely captures $[\![env]\!]$.

Now we translate LTL$_f$ formula φ into DA $\mathcal{A}_\varphi = (2^{\mathcal{X}\cup\mathcal{Y}}, Q_\varphi, I_\varphi, \delta_\varphi, \mathrm{Reach}(R_\varphi))$ that accepts a trace π iff $\pi^k \models \varphi$ for some $k \geq 0$, and take the product of $\mathcal{A}'_{\mathrm{e}}$ and $\mathcal{A}_\varphi$ into $\mathcal{A} = (2^{\mathcal{X}\cup\mathcal{Y}}, Q, I, \delta, \mathrm{Reach}(R))$, where $Q = \mathsf{Env} \times Q_\varphi$, $I = (I_{\mathrm{e}}, I_\varphi)$, $\delta((q_1, q_2), X \cup Y) = (\delta_{\mathrm{e}}(q_1, X \cup Y), \delta_\varphi(q_2, X \cup Y))$, and $R = R_\varphi$ (for simplicity, we omit the projection of states in R to R_φ here, we do the same later for similar usage). Indeed, $\delta((q_1, q_2), X \cup Y) =$ undefined, if $\delta_{\mathrm{e}}(q_1, X \cup Y) =$ undefined. At the end, solve a reachability game on $\mathcal{A}$ for the agent via a least fixpoint computation and obtain the *agent winning region* $\mathsf{Agn}_\varphi = \bigcup_{0\leq l\leq u} \mathsf{Agn}^l_\varphi$, where Agn^l_φ are the "approximates" of the fixpoint computation. Clearly, if $\mathcal{A}$ does not have an agent winning strategy, i.e., $I \notin \mathsf{Agn}_\varphi$, or $\mathsf{Run}(\mathcal{A}, h)$ does not always visit states in Agn_φ, then there does not exist an agent strategy enforcing φ with respect to h. Otherwise, we abstract σ enforcing φ with respect to h, by first restricting σ to be compatible with h, considering only the environment strategies that are compatible with h, then following the least fixpoint computation to get closer to R_φ at every step until reaching R_φ. At the end, σ keeps playing $\mathbf{stop}$ right after reaching R_φ. The correctness of the construction is justified by the following lemma, which is easy to prove by construction.

Lemma 1. *Let φ be an* LTL$_f$ *formula, $h \in (2^{\mathcal{X}\cup\mathcal{Y}})^*$ be a finite history, and σ be constructed as above. Then σ enforces φ with respect to h.*

The following theorem shows that computing a strategy that enforces φ with respect to a history h is not more difficult than computing a strategy that just enforces φ.

Theorem 3. *Let φ be an* LTL$_f$ *formula and $h \in (2^{\mathcal{X}\cup\mathcal{Y}})^*$ be a history. Computing an agent strategy that enforces φ with respect to history h is 2EXPTIME-complete in φ.*

For the synthesis setting that allows agent rights φ_{r} while pursuing φ_{d}, we expect an agent strategy being able to enforce φ_{d}, and along the execution until then, the agent is always able to enforce also φ_{r}, i.e., to enforce $\varphi_{\mathrm{d}} \wedge \varphi_{\mathrm{r}}$.

Definition 5. *Agent strategy σ enforcing φ_{d} is right-aware for φ_{r} if $\forall\gamma \in [\![env]\!]$:*

- $\mathsf{Play}(\gamma, \sigma) \models \varphi_{\mathrm{d}}$*;*
- *for every prefix h of $\mathsf{Play}(\gamma, \sigma)$, there exists an agent strategy σ_h that enforces $\varphi_{\mathrm{d}} \wedge \varphi_{\mathrm{r}}$ with respect to history h, i.e., $\sigma_h \rhd_h \varphi_{\mathrm{d}} \wedge \varphi_{\mathrm{r}}$.*

The problem of LTL$_f$ synthesis with duties and rights is defined as follows.

Definition 6 (LTL$_f$ synthesis with duties and rights)**.** *The problem is described as a tuple $\mathcal{P} = (env, \varphi_{\mathrm{d}}, \varphi_{\mathrm{r}})$, where env is an* LTL$_f$ *formula specifying the environment safety specification, φ_{d} and φ_{r} are* LTL$_f$ *formulas specifying the duties and rights, respectively. Realizability of $\mathcal{P}$ checks whether there exists an agent strategy σ enforcing φ_{d} that is right-aware for φ_{r}. Synthesis of $\mathcal{P}$ computes a strategy σ if exists.*

This class of synthesis problem is able to naturally reflect the problem structure of many autonomous agent applications. We illustrate this with a relatively simple example.

Example 1. *Consider a cleaning robot working in a circular hallway, where the charging station is located close to the entrance. Suppose the robot gets assigned a duty of "cleaning room A" $\varphi_{\mathrm{d}} = \Diamond(\neg \mathit{Dust\_A} \wedge \mathit{RobotOut\_A})$, together with the rights of "fully charging battery" $\varphi_{\mathrm{r}} = \Diamond(\mathit{BatteryFull})$. In this hallway, the robot has two strategies to enforce φ_{d}:*

1. *Take the direction that passes the charging station to room A and clean it. The remaining battery after enforcing φ_{d} still allows the robot to reach the charging station;*
2. *Take the other direction to reach room A and clean it. The remaining battery after enforcing φ_{d} is not enough for the robot to reach the charging station.*

Although both strategies allow the robot to enforce φ_{d}, only strategy (1) allows the robot to enforce φ_{d} and be right-aware of φ_{r} "fully charging battery".

3.1 Synthesis Technique

Following the construction explained above, we can compute $[\![env]\!]$ and represent it as $\mathcal{A}'_{\mathrm{e}} = (2^{\mathcal{X}\cup\mathcal{Y}}, \mathsf{Env}, I_{\mathrm{e}}, \delta'_{\mathrm{e}}, \mathrm{Safe}(\mathsf{Env}))$. Moreover, we know that both duties φ_{d} and rights φ_{r} can be represented by DAs $\mathcal{A}_{\mathrm{d}}$ and $\mathcal{A}_{\mathrm{r}}$ with reachability conditions, respectively. The crucial difference is that, apart from achieving φ_{d} through a reachability game on $\mathcal{A}_{\mathrm{d}}$, agent rights allow the agent to decide whether to achieve φ_{r}. To do so, the agent should have the ability to make such decision, which can be naturally captured by the agent winning region of the reachability game on $\mathcal{A}_{\mathrm{r}}$.

Given the synthesis problem $\mathcal{P} = (env, \varphi_{\mathrm{d}}, \varphi_{\mathrm{r}})$, we have the following: regardless of which strategy the environment chooses to enforce env, thus staying in $\mathcal{A}'_{\mathrm{e}}$, the desired agent strategy must make sure that the generated trace satisfies the reachability condition of $\mathcal{A}_{\mathrm{d}}$, and that if the agent decides to pursue also φ_{r}, there exists a strategy that the agent can take to satisfy the reachability conditions of $\mathcal{A}_{\mathrm{d}}$ and $\mathcal{A}_{\mathrm{r}}$.

To synthesize such a strategy, we do the following: (*i*) compute the agent winning region $\mathsf{Agn}_{\mathrm{r}}$, from where the agent is able to lead the trace to satisfy the reachability conditions of $\mathcal{A}_{\mathrm{d}}$, also $\mathcal{A}_{\mathrm{r}}$; (*ii*) compute an agent winning strategy σ s.t. for every $\gamma \in [\![env]\!]$, $\mathsf{Trace}(\gamma, \sigma)$ satisfies the reachability condition of $\mathcal{A}_{\mathrm{d}}$ by visiting *only* states in $\mathsf{Agn}_{\mathrm{r}}$. In this way, the agent maintains the ability of also satisfying the reachability conditions of $\mathcal{A}_{\mathrm{r}}$. We now elaborate on every step.

Step 1. Compute $\mathsf{Agn}_{\mathrm{r}}$. Build $\mathcal{A}_{\mathrm{d}} = (2^{\mathcal{X}\cup\mathcal{Y}}, Q_{\mathrm{d}}, I_{\mathrm{d}}, \delta_{\mathrm{d}}, \mathrm{Reach}(R_{\mathrm{d}}))$ that accepts a trace π iff $\pi^k \models \varphi_{\mathrm{d}}$ for some $k \geq 0$, and $\mathcal{A}_{\mathrm{r}} = (2^{\mathcal{X}\cup\mathcal{Y}}, Q_{\mathrm{r}}, I_{\mathrm{r}}, \delta_{\mathrm{r}}, \mathrm{Reach}(R_{\mathrm{r}}))$ that accepts a trace π iff $\pi^k \models \varphi_{\mathrm{r}}$ for some $k \geq 0$. Take the product of $\mathcal{A}'_{\mathrm{e}}$, $\mathcal{A}_{\mathrm{d}}$, and $\mathcal{A}_{\mathrm{r}}$ into $\mathcal{A} =$

$(2^{\mathcal{X}\cup\mathcal{Y}}, Q, I, \delta, \mathrm{Reach}(R))$, where $Q = \mathsf{Env} \times Q_\mathrm{d} \times Q_\mathrm{r}$, $I = (I_\mathrm{e}, I_\mathrm{r}, I_\mathrm{d})$, $\delta((q_1, q_2, q_3), X \cup Y) = (\delta_\mathrm{e}(q_1, X \cup Y), \delta_\mathrm{d}(q_2, X \cup Y), \delta_\mathrm{r}(q_3, X \cup Y))$, and $R = R_\mathrm{d} \cap R_\mathrm{r}$. Indeed, $\delta((q_1, q_2, q_3), X \cup Y) =$ undefined, if $\delta_\mathrm{e}(q_1, X \cup Y) =$ undefined. At the end, solve a reachability game on $\mathcal{A}$ for the agent via a least fixpoint computation, thus obtaining $\mathsf{Agn}_\mathrm{r} = \bigcup_{0 \le i \le m} \mathsf{Agn}_\mathrm{r}^i$. If $I \notin \mathsf{Agn}_\mathrm{r}$, return "unrealizable".

Lemma 2. *Let $\mathcal{P}$ be a problem of LTL_f synthesis with duties and rights, and Agn_r the agent winning region of the reachability game on $\mathcal{A} = (2^{\mathcal{X}\cup\mathcal{Y}}, Q, I, \delta, \mathrm{Reach}(R_\mathrm{d} \cap R_\mathrm{r}))$ computed as above. Then $\mathcal{P}$ is realizable iff $I \in \mathsf{Agn}_\mathrm{r}$.*

Proof. We prove the lemma in both directions.

($\Leftarrow$) We need to show that if $I \in \mathsf{Agn}_\mathrm{r}$, then $\mathcal{P}$ is realizable. By construction, $I \in \mathsf{Agn}_\mathrm{r}$ shows that there exists an agent strategy σ such that, for every $\gamma \in [\![env]\!]$, $\pi = \mathsf{Trace}(\gamma, \sigma)$ is such that $\pi \in \mathcal{L}(\mathcal{A})$. That is to say, $\pi^k \models \varphi_\mathrm{d} \wedge \varphi_\mathrm{r}$ for some $k \ge 0$, thus it also holds that $\pi^k \models \varphi_\mathrm{d}$. In this case, σ starts playing $\mathbf{stop}$ after π^k, and so we have $\mathsf{Play}(\gamma, \sigma) = \pi^k$. Moreover, for every prefix h of $\mathsf{Play}(\gamma, \sigma)$, we can construct an agent strategy σ_h that works exactly the same as σ, which indeed enforces $\varphi_\mathrm{d} \wedge \varphi_\mathrm{r}$ with respect to history h.

($\Rightarrow$) We prove by contradiction. If $I \notin \mathsf{Agn}_\mathrm{r}$, then there does not exist an agent winning strategy of the reachability game on $\mathcal{A}$. Suppose the agent decides to pursue also φ_r at the very beginning, then the agent does not have a strategy that enforces $\varphi_\mathrm{d} \wedge \varphi_\mathrm{r}$ with respect to history $h = \epsilon$, i.e., empty trace. Hence, $\mathcal{P}$ is unrealizable. □

Step 2. Compute strategy σ. Note that σ needs to lead the play to reach R_d by visiting states in Agn_r only. First, we define a new DA with reachability-safety condition $\mathcal{A}_1 = (2^{\mathcal{X}\cup\mathcal{Y}}, Q, I, \delta, \mathrm{Reach\text{–}Safe}(R_\mathrm{d}, \mathsf{Agn}_\mathrm{r}))$ from $\mathcal{A} = (2^{\mathcal{X}\cup\mathcal{Y}}, Q, I, \delta, \mathrm{Reach}(R_\mathrm{d} \cap R_\mathrm{r}))$. It has been shown in (De Giacomo et al. 2021b) that $\mathcal{A}_1$ can be reduced to a new DA $\mathcal{A}_1' = (2^{\mathcal{X}\cup\mathcal{Y}}, Q, I, \delta', \mathrm{Reach}(R'))$ with δ' and R' as follows:

- $\delta'(q, X \cup Y) = \begin{cases} \delta(q, X \cup Y) & \text{if } q \in \mathsf{Agn}_\mathrm{r} \\ q & \text{if } q \notin \mathsf{Agn}_\mathrm{r} \end{cases}$
- $R' = R_\mathrm{d} \cap \mathsf{Agn}_\mathrm{r}$

Intuitively, the only change in δ' is to turn all non-safe states (states not in Agn_r) into sink states, while R' requires reaching a goal state (a state in R_d) that is also safe (i.e., it is in Agn_r). Then we solve a reachability game on $\mathcal{A}_1'$ via a least fixpoint computation and obtain $\mathsf{Agn} = \bigcup_{0 \le j \le n} \mathsf{Agn}^j$. Note that $I \in \mathsf{Agn}$ indeed holds, which is guaranteed by the reachability game for computing Agn_r in the previous step. Finally, we define a strategy generator based on $\mathsf{Agn} = \bigcup_{0 \le j \le n} \mathsf{Agn}^j$, represented as a transducer $\mathcal{T} = (2^{\mathcal{X}\cup\mathcal{Y}}, Q, I, \varrho, \tau)$, where

- $2^{\mathcal{X}\cup\mathcal{Y}}$, Q and I are the same as in $\mathcal{A}$;
- $\varrho : Q \times 2^{\mathcal{X}} \to 2^Q$ is the transition function such that $\varrho(q, X) = \{q' \mid q' = \delta(q, X \cup Y) \text{ and } Y \in \tau(q)\}$;
- $\tau : Q \times 2^{\mathcal{X}} \to 2^{2^{\mathcal{Y}}}$ is the output function s.t. $\forall X \in 2^{\mathcal{X}}$, $\tau(q, X) = \{Y \mid \delta(q, X \cup Y) \in \mathsf{Agn}^j\}$ if $q \in (\mathsf{Agn}^{j+1} \backslash \mathsf{Agn}^j)$, otherwise $\tau(q, X) = 2^{\mathcal{Y}}$.

This transducer generates an agent strategy $\sigma : (2^{\mathcal{X}})^+ \to 2^{\mathcal{Y}}$ in the following way: for every $\xi^k \in (2^{\mathcal{X}})^+$ $(k \ge 0)$

$$\sigma(\xi^k) = \begin{cases} \mathbf{stop} & \text{if } \mathsf{Run}(\mathcal{A}, \pi^{k-1}) \text{ visited } R_\mathrm{d}, \\ Y \in \tau(q_k, X_k) & \text{otherwise.} \end{cases}$$

where $\mathsf{Run}(\mathcal{A}, \pi^{k-1}) = q_0 q_1 q_2 \dots q_k$ s.t. $q_0 = I$, and $\pi^{k-1} = (X_0 \cup Y_0)(X_1 \cup Y_1) \dots (X_{k-1} \cup Y_{k-1})$. Note that $\mathcal{T}$ generates a strategy in the way of restricting τ to return only one of its values (chosen arbitrarily).

Lemma 3. *Let $\mathcal{P}$ be a problem of LTL_f synthesis with duties and rights, and $\mathcal{T}$ constructed as above. Any strategy returned by $\mathcal{T}$ is a strategy that solves the synthesis of $\mathcal{P}$.*

Proof. Let σ be an arbitrary strategy generated by $\mathcal{T}$, i.e., $I \in \mathsf{Agn}$, and $\gamma \in [\![env]\!]$ be an arbitrary environment strategy that enforces env. First, $\mathcal{T}$ already restricts the environment to be able to only choose strategies from $[\![env]\!]$. Then, by construction, $\mathsf{Play}(\gamma, \sigma)$ satisfies the following:

- $\mathsf{Play}(\gamma, \sigma) \models \varphi_\mathrm{d}$, since σ forces $\mathsf{Play}(\gamma, \sigma)$ to get closer to R_d at every step until reaching R_d. Moreover, σ starts playing $\mathbf{stop}$ only after then.
- there exists $\sigma_h \rhd_h \varphi_\mathrm{d} \wedge \varphi_\mathrm{r}$ for every prefix h of $\mathsf{Play}(\gamma, \sigma)$. This holds since σ restricts $\mathsf{Play}(\gamma, \sigma)$ to visit states in Agn_r only. Therefore, for every prefix h of $\mathsf{Play}(\gamma, \sigma)$, there exists an agent strategy $\hat{\sigma}$ of the reachability game on $\mathcal{A}_h = (2^{\mathcal{X}\cup\mathcal{Y}}, Q, \delta(I, h), \delta, \mathrm{Reach}(R_\mathrm{d} \cap R_\mathrm{r}))$. Hence, we can construct an agent strategy σ_h that first copies h until reaching $\delta(I, h)$ and works as $\hat{\sigma}$ until reaching $R_\mathrm{d} \cap R_\mathrm{r}$, then plays $\mathbf{stop}$ forever. Therefore, σ_h holds that $\sigma_h \rhd_h \varphi_\mathrm{d} \wedge \varphi_\mathrm{r}$. □

Notice that by the construction described above, if the reachability game on $\mathcal{A}$ (in Step 1) does not have an agent winning strategy, then $\mathcal{T}$ trivially returns *no strategy* and indeed, by Lemma 2, $\mathcal{P}$ is unrealizable. As an immediate consequence of Lemmas 2&3, we have:

Theorem 4. *Let $\mathcal{P}$ be a problem of LTL_f synthesis with duties and rights. Realizability of $\mathcal{P}$ can be solved by reducing to a suitable reachability game. Synthesis of $\mathcal{P}$ can be solved by generating a strategy from $\mathcal{T}$ constructed as above.*

Theorem 5. *Let $\mathcal{P}$ be a problem of LTL_f synthesis with duties and rights. Realizability of $\mathcal{P}$ is 2EXPTIME-complete.*

Theorem 6. *Let $\mathcal{P}$ be a problem of LTL_f synthesis with duties and rights. Then computing a strategy solving $\mathcal{P}$ can take, in the worst case, double-exponential time in the size of $|\varphi_\mathrm{d}| + |\varphi_\mathrm{r}| + |env|$.*

We observe that if env is specified, for example, using, e.g., PDDL (Haslum et al. 2019) instead of LTL_f, then the complexity with respect to the environment specification env only becomes EXPTIME-complete (membership from a construction, hardness from planning of Fully Observable Nondeterministic Domains (FOND) (Rintanen 2004)).

Enforcing also rights while executing. Given problem $\mathcal{P} = (env, \varphi_\mathrm{d}, \varphi_\mathrm{r})$, suppose we have synthesized a strategy σ for $\mathcal{P}$, which enforces φ_d and is right-aware for φ_r, and while executing σ, the agent wants to satisfy also its rights φ_r. Then we can consider the history h generated

with the environment so far, and synthesize a strategy σ_h, that enforces $\varphi_{\mathrm{d}} \wedge \varphi_{\mathrm{r}}$ with respect to history h. This can take 2EXPTIME, as shown by Theorem 3.

Nevertheless, if we consider the construction above, we actually do not need to compute the new strategy σ_h from scratch. This is because we can, base on the immediate results obtained from computing the original strategy σ, to construct a transducer $\mathcal{T}_{\mathrm{r}}$ for generating σ_h of a given history h. In particular, this transducer is independent of h. Therefore, we can construct $\mathcal{T}_{\mathrm{r}}$ apriori, and use it to obtain σ_h when the agent chooses to satisfy φ_{r} after history h. The essential ingredients for constructing $\mathcal{T}_{\mathrm{r}}$ is the DA $\mathcal{A} = (2^{\mathcal{X}\cup\mathcal{Y}}, Q, I, \delta, \mathrm{Reach}(R_{\mathrm{d}} \cap R_{\mathrm{r}}))$ and the agent winning region $\mathsf{Agn}_{\mathrm{r}} = \bigcup_{0\leq i\leq m} \mathsf{Agn}_{\mathrm{r}}^i$. We construct $\mathcal{T}_{\mathrm{r}} = (2^{\mathcal{X}\cup\mathcal{Y}}, Q, I, \varrho_{\mathrm{r}}, \tau_{\mathrm{r}})$ as follows:

- $2^{\mathcal{X}\cup\mathcal{Y}}$, Q and I are the same as in $\mathcal{A}$;
- $\varrho_{\mathrm{r}} : Q \times 2^{\mathcal{X}} \to 2^Q$ is the transition function such that $\varrho_{\mathrm{r}}(q, X) = \{q' \mid q' = \delta(q, X \cup Y) \text{ and } Y \in \tau_{\mathrm{r}}(q)\}$;
- $\tau_{\mathrm{r}} : Q \times 2^{\mathcal{X}} \to 2^{2^{\mathcal{Y}}}$ is the output function such that $\forall X \in 2^{\mathcal{X}}$, $\tau_{\mathrm{r}}(q, X) = \{Y \mid \delta(q, X \cup Y) \in \mathsf{Agn}_{\mathrm{r}}^i\}$ if $q \in \mathsf{Agn}_{\mathrm{r}}^{i+1} \backslash \mathsf{Agn}_{\mathrm{r}}^i$, otherwise $\tau_{\mathrm{r}}(q, X) = 2^{\mathcal{Y}}$.

Suppose while executing σ, which enforces φ_{d} and is right-aware for φ_{r}, the agent chooses to satisfy φ_{r} after history h, the transducer $\mathcal{T}_{\mathrm{r}}$ generates an agent strategy $\sigma_h : (2^{\mathcal{X}})^+ \to 2^{\mathcal{Y}}$ in the following way: for every $\xi^k \in (2^{\mathcal{X}})^+$ that is compatible with history $h = (X_0 \cup Y_0)(X_1 \cup Y_1) \ldots (X_i \cup Y_i)$

$$\sigma_h(\xi^k) = \begin{cases} Y_k & \text{if } 0 \leq k \leq i \\ \mathbf{stop} & \text{if } \mathsf{Run}(\mathcal{A}, \pi^{k-1}) \text{ visited } R_{\mathrm{d}} \cap R_{\mathrm{r}} \\ Y \in \tau(q_k, X_k) & \text{otherwise.} \end{cases}$$

where $\mathsf{Run}(\mathcal{A}, \pi^{k-1}) = q_0 q_1 q_2 \ldots q_k$ such that $q_0 = I$, and $\pi^{k-1} = (X_0 \cup Y_0)(X_1 \cup Y_1) \ldots (X_{k-1} \cup Y_{k-1})$. Intuitively, given history h, $\mathcal{T}_{\mathrm{r}}$ generates a strategy σ_h by first following h, and after h, choosing suitable agent action to enforce the play to get closer to $R_{\mathrm{d}} \cap R_{\mathrm{r}}$. At the end, σ_h keeps playing $\mathbf{stop}$ right after visiting $R_{\mathrm{d}} \cap R_{\mathrm{r}}$.

Theorem 7. *Let $\mathcal{P}$ be a problem of LTL_f synthesis with duties and rights, σ be an agent strategy computed by $\mathcal{T}$ that solves the synthesis of $\mathcal{P}$, and σ_h be an agent strategy that is generated by $\mathcal{T}_{\mathrm{r}}$ for a history $h \in (2^{\mathcal{X}\cup\mathcal{Y}})^*$. Then σ_h* enforces *$\varphi_{\mathrm{d}} \wedge \varphi_{\mathrm{r}}$, with respect to history h, i.e., $\sigma_h \rhd_h \varphi_{\mathrm{d}} \wedge \varphi_{\mathrm{r}}$.*

Proof. Let σ_h be an arbitrary strategy generated by $\mathcal{T}_{\mathrm{r}}$ for history h, and $\gamma \in [\![env]\!]$ be an arbitrary environment strategy that enforces env. By Lemma 3, we have that $\mathsf{Run}(\mathcal{A}, h)$ only visits states in $\mathsf{Agn}_{\mathrm{r}}$. By construction of σ_h, $\mathsf{Play}(\gamma, \sigma)$ has h as a prefix, and $\mathsf{Play}(\gamma, \sigma) \models \varphi_{\mathrm{d}} \wedge \varphi_{\mathrm{r}}$. Therefore, $\sigma_h \rhd_h \varphi_{\mathrm{d}} \wedge \varphi_{\mathrm{r}}$ holds. □

The advantage of building transducer $\mathcal{T}_{\mathrm{r}}$ is that this transducer works for any history h generated by σ that enforces φ_{d} and is right-aware for φ_{r}. Moreover, when building the transducer $\mathcal{T}$ for σ, we already have all the ingredients to build also $\mathcal{T}_{\mathrm{r}}$, with only a constant overhead (i.e., since we are computing two transducers, sharing essentially the same cost, instead of one).

We now extend Example 1 to show how to utilize the transducer $\mathcal{T}_{\mathrm{r}}$ in the presence of robot also achieving rights.

Example 2. *Suppose the robot decides to also achieve its rights $\varphi_{\mathrm{r}} = \Diamond(\mathit{BatteryFull})$ while cleaning room A. Let us assume that the by now the running history is h. The robot will look into $\mathcal{T}_{\mathrm{r}}$ and choose a strategy σ_h out of $\mathcal{T}_{\mathrm{r}}$ that allows it to enforce $\varphi_{\mathrm{d}} \wedge \varphi_{\mathrm{r}}$.*

4 Handling Further Duties and Rights While Executing

Let us focus on duty only first. Commonly in synthesis the agent only gets one task (duty) to accomplish, after which, the agent can terminate. However, in practice, further tasks might arrive while executing the current task, e.g., a new room to clean while the robot is cleaning the rooms it got assigned at the beginning. Intuitively, the new task can be considered as an update of the previous task.

Synthesizing updated specifications has been recently studied in Formal Methods, under the name of *live synthesis* (Finkbeiner, Klein, and Metzger 2021), where the desired properties are specified in LTL and can get updated while executing a strategy of the original LTL specification. The goal of live synthesis is to synthesize a new strategy to replace an already running strategy. In particular, the correct handover from the already running strategy to the new strategy is specified by an extension of LTL, called LiveLTL. For specifications in LiveLTL, the synthesis problem shares the same complexity bound as standard LTL synthesis.

Despite that synthesis problems of LTL_f can be solved by a reduction to suitable problems of LTL, since LTL_f can be encoded in LTL, such reductions do not seem promising, as shown in (Zhu et al. 2017) for LTL_f synthesis, and (Zhu et al. 2020; De Giacomo et al. 2020) for LTL_f synthesis under environment specifications. So while we want to consider an LTL_f variant of live synthesis, we avoid a detour to LiveLTL synthesis and devise a direct synthesis technique for LTL_f.

We start by observing that the crucial difference between new duties and the ongoing duties is that, the agent should enforce ongoing duties from the very beginning, but enforce new duties after a history, i.e., starting from the moment that the new duties are assigned to the agent.

Definition 7. *Let φ be an LTL_f formula and $h \in (2^{\mathcal{X}\cup\mathcal{Y}})^*$ be a history. An agent strategy σ* enforces *φ after history h, denoted by $\sigma \rhd_{\mathrm{af(h)}} \varphi$, if $\forall \gamma \in [\![env]\!]$ such that $\mathsf{Play}(\gamma, \sigma)$ has h as a prefix, we have that $\mathsf{Play}(\gamma, \sigma), |h| \models \varphi$.*

Recall that Definition 4 describes how an *agent strategy enforces φ with respect to a history h*, and Definition 7 above describes how an *agent strategy enforces φ after a history h*. There is a significant difference between these two notions, since we use them to differentiate how agent strategies enforce ongoing duties and new duties. In particular, an agent strategy enforces ongoing duties with respect to a history h, but enforces new duties after a history h.

Note that the environment strategy enforcing env in any case starts from the very beginning. Moreover, we only consider the cases where both the environment strategy γ and the agent strategy σ are compatible with h. In other words,

we need to consider environment strategies that are in the set $[\![env]\!]^h = \{\gamma \mid \forall \sigma$ that is compatible with h we have $\mathsf{Trace}(\gamma,\sigma)$ has h as a prefix and $\mathsf{Trace}(\gamma,\sigma) \models_\forall env\}$.

In order to compute a strategy σ that enforces φ after h, we split the trace $\mathsf{Trace}(\gamma,\sigma)$ into two phases. In phase I, both strategies γ and σ are compatible with h. In phase II, the agent focuses on the environment strategies that enforce env with respect to h. We show how to compute a strategy σ enforcing φ after h by addressing two phases in reverse order. Specifically, we first compute the set of environment strategies that start executing after h, but enforce env when the compatible traces are concatenated to h. We denote this set of environment strategies by

$$[\![env, \mathrm{af(h)}]\!] = \{\gamma \mid \forall \sigma.\ h \cdot \mathsf{Trace}(\gamma,\sigma) \models_\forall env\}.$$

The fact that we can focus on this set of environment strategies is justified by the following lemma, which is easy to prove considering the two definitions of $[\![env]\!]^h$ and $[\![env, \mathrm{af(h)}]\!]$.

Lemma 4. *For every $\gamma \in [\![env]\!]^h$ there exists $\gamma' \in [\![env, \mathrm{af(h)}]\!]$ s.t. for every $\lambda = h|_{\mathcal{Y}} \cdot \lambda'$ we have $\gamma(\lambda) = \gamma'(\lambda')$.*

Viceversa, for every $\gamma' \in [\![env, \mathrm{af(h)}]\!]$ there exists $\gamma \in [\![env]\!]^h$ s.t. for every $\lambda = h|_{\mathcal{Y}} \cdot \lambda'$ we have $\gamma(\lambda) = \gamma'(\lambda')$.

To compute $[\![env, \mathrm{af(h)}]\!]$, we first compute $[\![env]\!]$, represented as $\mathcal{A}'_{\mathrm{e}} = (2^{\mathcal{X}\cup\mathcal{Y}}, \mathsf{Env}, I_{\mathrm{e}}, \delta'_{\mathrm{e}}, \mathrm{Safe}(\mathsf{Env}))$, as described in Section 3. In order to synchronize the starting point of the environment to be aligned with the instant after history h, we run $\mathcal{A}'_{\mathrm{e}}$ on h to obtain a new DA $\mathcal{A}'_{\mathrm{e,af(h)}} = (2^{\mathcal{X}\cup\mathcal{Y}}, \mathsf{Env}, I_{\mathrm{e,af(h)}}, \delta'_{\mathrm{e}}, \mathrm{Safe}(\mathsf{Env}))$ that differs from $\mathcal{A}'_{\mathrm{e}}$ only on the initial state, and $I_{\mathrm{e,af(h)}} = \delta'_{\mathrm{e}}(I_{\mathrm{e}}, h)$. The following lemma shows that $\mathcal{A}'_{\mathrm{e,af(h)}}$ precisely captures $[\![env, \mathrm{af(h)}]\!]$, which is easy to prove by construction.

Lemma 5. *Let env be an LTL_f formula specifying a safety property, $h \in (2^{\mathcal{X}\cup\mathcal{Y}})^*$ be a finite history, and $\mathcal{A}'_{\mathrm{e}}$ be constructed as above. Then $\mathcal{A}'_{\mathrm{e,af(h)}}$ represents the set $[\![env, \mathrm{af(h)}]\!]$.*

Having $[\![env, \mathrm{af(h)}]\!]$ represented as DA $\mathcal{A}'_{\mathrm{e,af(h)}}$, we can first construct the DA $\mathcal{A}_\varphi$ and then solve a reachability game on the product $\mathcal{A}'_{\mathrm{e,af(h)}} \times \mathcal{A}_\varphi$ to abstract an agent strategy $\hat{\sigma}$ that guides the play to satisfy the reachability condition of $\mathcal{A}_\varphi$, hence enforcing φ. The final agent strategy enforcing φ after h can be obtained by first copying h, and then switching to $\hat{\sigma}$ after h. Formally, for every $\xi^k \in (2^{\mathcal{X}})^+$ that is compatible with $h = (X_0 \cup Y_0)(X_1 \cup Y_1)\ldots(X_i \cup Y_i) \in (2^{\mathcal{X}\cup\mathcal{Y}})^*$

$$\sigma(\xi^k) = \begin{cases} Y_k & \text{if } 0 \le k \le i, \\ \hat{\sigma}(\iota) & \text{if } \xi^k = h \cdot \iota. \end{cases}$$

Lemma 6. *Let φ be an LTL_f formula, $h \in (2^{\mathcal{X}\cup\mathcal{Y}})^*$ be a history, and σ be constructed as above. Then σ enforces φ after h.*

Proof. Note that σ is constructed from h and a strategy $\hat{\sigma}$, which holds that for every $\hat{\gamma} \in [\![env, \mathrm{af(h)}]\!]$, $\mathsf{Play}(\hat{\gamma},\hat{\sigma}) \models \varphi$. Hence, together with Lemma 4, it holds that for every $\gamma' \in [\![env]\!]^h$, $\mathsf{Play}(\gamma',\sigma), |h| \models \varphi$. Clearly, σ holds that for every $\gamma \in [\![env]\!]$ such that $\mathsf{Play}(\gamma',\sigma)$ has h as a prefix, then $\mathsf{Play}(\gamma,\sigma), |h| \models \varphi$. Therefore, σ enforces φ after h. □

The following theorem shows that computing an agent strategy enforcing φ after a history h is not more difficult than computing a strategy that just enforces φ.

Theorem 8. *Let φ be an LTL_f formula and $h \in (2^{\mathcal{X}\cup\mathcal{Y}})^*$ be a history. Computing an agent strategy that enforces φ after history h is 2EXPTIME-complete in φ.*

Building on the above results and the results in Section 3, we now enrich our synthesis setting by allowing both further duties and further rights, specified as LTL_f formulas φ_{fd} and φ_{fr}, respectively.

Definition 8. *Let h be the formed history when further duties φ_{fd} and rights φ_{fr} arrive. Agent strategy σ enforcing φ_{fd} is right-aware for φ_{fr} after h, if $\forall\gamma \in [\![env]\!]$:*

- *$\mathsf{Play}(\gamma,\sigma)$ has h as a prefix and $\mathsf{Play}(\gamma,\sigma), |h| \models \varphi_{\mathrm{fd}}$;*
- *for every prefix l of $\mathsf{Play}(\gamma,\sigma)$ that has h as a prefix, there exists an agent strategy σ_l that enforces $\varphi_{\mathrm{fd}} \wedge \varphi_{\mathrm{fr}}$ after history h, i.e., $\sigma_l \rhd_{\mathrm{af(h)}} \varphi_{\mathrm{fd}} \wedge \varphi_{\mathrm{fr}}$.*

The enriched synthesis problem that allows further duties and rights is defined as follows.

Definition 9 (LTL_f synthesis for further duties and rights)**.** *The problem is described as a tuple $\hat{\mathcal{P}} = (env, \varphi_{\mathrm{d}}, \varphi_{\mathrm{r}}, \varphi_{\mathrm{fd}}, \varphi_{\mathrm{fr}}, h)$, where env is an LTL_f formula specifying the environment safety specification, φ_{d} and φ_{r} are LTL_f formulas specifying duties and rights, respectively, φ_{fd} and φ_{fr} are LTL_f formulas specifying further duties and rights that arrive after history h, respectively. Realizability of $\hat{\mathcal{P}}$ checks whether there exists an agent strategy σ s.t.:*

- *it enforces φ_{d} and is right-aware for φ_{r} wrt h;*
- *it enforces φ_{fd} and is right-aware for φ_{fr} after h.*

Synthesis of $\hat{\mathcal{P}}$ computes a strategy σ if exists.

Notice that, in Definition 9, we also only consider the cases where both the environment strategies and the agent strategies are compatible with h.

We extend Example 1 to address the intuition of this class of synthesis problem.

Example 3. *Suppose the robot is on its way to room A, while receiving a new duty of "cleaning room B" $\varphi_{\mathrm{fd}} = \Diamond(\neg$Dust_B $\wedge$ RobotOut_B$)$. The robot has generated a history h when receiving φ_{fd}. Now, the robot has one strategy to enforce φ_{d} with respect to h and enforce φ_{fd} after h:*

- *Take the direction that passes the charging station to reach room A and clean it. Then go to room B and clean it. The remaining battery after cleaning is not enough for the robot to reach the charging station.*

In this case, the robot would refuse the new duty φ_{fd}, since completing it would be conflicted with maintaining the robot rights φ_{r}.

On the other hand, let us consider the situation where the cleaning robot is able to handle original duties and rights, together with further duties and rights.

Example 4. *Suppose the cleaning robot is cleaning room A, while receiving a new duty of "cleaning room C" $\varphi_{\mathrm{fd}} = \Diamond(\neg$Dust_C $\wedge$ RobotOut_C$)$, and a new right of "emptying the garbage collector" $\varphi_{\mathrm{fr}} = \Diamond($Collector_Empty$)$. The*

robot has generated a history h when receiving $\varphi_{\rm fd}$ and $\varphi_{\rm fr}$. Since this is a circular hallway, the robot again has two directions to reach room C. Nevertheless, the robot only takes the strategy that allows it to reach the charging station, and the garbage station whenever it wants.

4.1 Synthesis Technique

Given problem $\hat{\mathcal{P}} = (env, \varphi_{\rm d}, \varphi_{\rm r}, \varphi_{\rm fd}, \varphi_{\rm fr}, h)$, the main complication comes from further duties and rights that arrive after history h. This is because apart from enforcing $\varphi_{\rm fd}$ while maintaining $\varphi_{\rm fr}$, the agent should also enforce unfinished $\varphi_{\rm d}$ and be right-aware for $\varphi_{\rm r}$. To synthesize an agent strategy that is able to do so, we do the following: (*i*) compute the agent winning region $\mathsf{Agn}_{\rm r}$ from where the agent is able to lead the trace to satisfy $\varphi_{\rm d}$ and $\varphi_{\rm r}$; (*ii*) compute the agent winning region $\mathsf{Agn}_{\rm r\wedge fr}$ from where the agent is able to lead the trace to also satisfy both $\varphi_{\rm fd}$ and $\varphi_{\rm fr}$, but after h; (*iii*) synthesize an agent strategy σ enforcing $\varphi_{\rm d}$ and is right-aware for $\varphi_{\rm r}$ wrt h, also enforcing $\varphi_{\rm fd}$ being right-aware for $\varphi_{\rm fr}$, but after h. We now elaborate on every step.

Step 1. Compute $\mathsf{Agn}_{\rm r}$. As described in Section 3.1, we can construct the corresponding DAs $\mathcal{A}_{\rm d} = (2^{\mathcal{X}\cup\mathcal{Y}}, Q_{\rm d}, I_{\rm d}, \delta_{\rm d}, \mathrm{Reach}(R_{\rm d}))$ and $\mathcal{A}_{\rm r} = (2^{\mathcal{X}\cup\mathcal{Y}}, Q_{\rm r}, I_{\rm r}, \delta_{\rm r}, \mathrm{Reach}(R_{\rm r}))$ of $\varphi_{\rm d}$ and $\varphi_{\rm r}$, respectively. The agent winning region $\mathsf{Agn}_{\rm r}$ can be computed via a least fixpoint computation on the product DA $\mathcal{A}_{\rm d\wedge r} = (2^{\mathcal{X}\cup\mathcal{Y}}, Q_{\rm d\wedge r}, I_{\rm d\wedge r}, \delta_{\rm d\wedge r}, \mathrm{Reach}(R)_{\rm d\wedge r})$ constructed out of $\mathcal{A}'_{\rm e}$, $\mathcal{A}_{\rm d}$, and $\mathcal{A}_{\rm r}$, where $\mathcal{A}'_{\rm e}$ captures $[\![env]\!]$.

Step 2. Compute $\mathsf{Agn}_{\rm r\wedge fr}$. Build DA $\mathcal{A}_{\rm fd} = (2^{\mathcal{X}\cup\mathcal{Y}}, Q_{\rm fd}, I_{\rm fd}, \delta_{\rm fd}, \mathrm{Reach}(R_{\rm fd}))$ of $\varphi_{\rm fd}$, and DA $\mathcal{A}_{\rm fr} = (2^{\mathcal{X}\cup\mathcal{Y}}, Q_{\rm fr}, I_{\rm fr}, \delta_{\rm fr}, \mathrm{Reach}(R_{\rm fr}))$ of $\varphi_{\rm fr}$. In order to synthesize an agent strategy that enforces $\varphi_{\rm fd}$ and is right-aware $\varphi_{\rm fr}$ after h, we need to run $\mathcal{A}_{\rm d\wedge r}$ on h to obtain a new DA $\mathcal{A}'_{\rm d\wedge r}$ that differs from $\mathcal{A}_{\rm d\wedge r}$ only on the initial state, and $\mathcal{A}_{\rm d\wedge r,af(h)} = (2^{\mathcal{X}\cup\mathcal{Y}}, Q_{\rm d\wedge r}, I_{\rm d\wedge r,af(h)}, \delta_{\rm d\wedge r}, \mathrm{Reach}(R))$, where $I_{\rm d\wedge r,af(h)} = \delta(I_{\rm d\wedge r}, h)$. Clearly, if $\mathsf{Run}(\mathcal{A}_{\rm d\wedge r}, h)$ does not visit states in $\mathsf{Agn}_{\rm r}$ only, return "unrealizable", since every agent strategy that is compatible with h cannot enforce $\varphi_{\rm d}$ and be right-aware for $\varphi_{\rm r}$, thus $\hat{\mathcal{P}}$ is simply unrealizable, and so $\mathsf{Agn}_{\rm r\wedge fr} = \emptyset$.

Otherwise, we continue as follows. Since the final agent strategy should be able to guide the play to reach $R_{\rm d}$ and $R_{\rm fd}$, and always able to reach also $R_{\rm r}$ and $R_{\rm fr}$, we take the product of $\mathcal{A}_{\rm d\wedge r,af(h)}$, $\mathcal{A}_{\rm fd}$ and $\mathcal{A}_{\rm fr}$ into $\mathcal{A} = (2^{\mathcal{X}\cup\mathcal{Y}}, Q, I, \delta, \mathrm{Reach}(R))$, where $Q = Q_{\rm d\wedge r} \times Q_{\rm fd} \times Q_{\rm fr}$, $I = (I_{\rm d\wedge r,af(h)}, I_{\rm fd}, I_{\rm fr})$, $\delta((q_1,q_2,q_3), X\cup Y) = (\delta_{\rm d\wedge r}(q_1, X\cup Y), \delta_{\rm fd}(q_2, X\cup Y), \delta_{\rm fr}(q_3, X\cup Y))$. Furthermore, $\delta((q_1,q_2,q_3), X\cup Y) =$ undefined, if $\delta_{\rm d\wedge r}(q_1, X\cup Y) =$ undefined. Finally, $R = R_{\rm d}\cap R_{\rm r}\cap R_{\rm fd}\cap R_{\rm fr}$. We now solve a reachability game on $\mathcal{A}$ for the agent via a least fixpoint computation, to obtain $\mathsf{Agn}_{\rm r\wedge fr} = \bigcup_{0\le i\le m} \mathsf{Agn}^i_{\rm r\wedge fr}$. If $I \notin \mathsf{Agn}_{\rm r\wedge fr}$, i.e., $\mathcal{A}$ does not have an agent winning strategy, return "unrealizable".

Lemma 7. *Let $\hat{\mathcal{P}}$ be a problem of* LTL_f *synthesis for further duties and rights, and $\mathsf{Agn}_{\rm r\wedge fr}$ the agent winning region of the reachability game on $\mathcal{A} = (2^{\mathcal{X}\cup\mathcal{Y}}, Q, I, \delta, \mathrm{Reach}(R))$ computed as above. $\hat{\mathcal{P}}$ is realizable iff $I \in \mathsf{Agn}_{\rm r\wedge fr}$.*

Proof. We prove the lemma in both directions.

($\Leftarrow$) We need to show that if $I \in \mathsf{Agn}_{\rm r\wedge fr}$, then $\hat{\mathcal{P}}$ is realizable. By construction, $I \in \mathsf{Agn}_{\rm r\wedge fr}$ shows that there exists an agent strategy σ such that, for every $\gamma \in [\![env, \mathrm{af(h)}]\!]$, $\pi = \mathsf{Trace}(\gamma, \sigma)$ is such that $\pi \in \mathcal{L}(\mathcal{A})$. That is to say, $\pi^k \models \varphi_{\rm fd} \wedge \varphi_{\rm fr}$ for some $k \ge 0$, thus it also holds that $\pi^k \models \varphi_{\rm fd}$. Moreover, since $I = (I_{\rm d\wedge r,af(h)}, I_{\rm fd}, I_{\rm fr})$, where $I_{\rm d\wedge r,af(h)} = \delta_{\rm d\wedge r}(I, h)$, it also holds that $h\cdot\pi^k \models \varphi_{\rm d}\wedge\varphi_{\rm r}$, and thus $h\cdot\pi^k \models \varphi_{\rm d}$. Moreover, for every prefix l of $\mathsf{Play}(\gamma,\sigma)$ that has h as a prefix, we can construct an agent strategy σ_l that works exactly the same as σ, which indeed enforces $\varphi_{\rm r}$ with respect to h, and enforces $\varphi_{\rm fr}$ after h.

($\Rightarrow$) We prove by contradiction. If $I \notin \mathsf{Agn}_{\rm r\wedge fr}$, then either $\mathsf{Agn}_{\rm r\wedge fr} = \emptyset$, i.e., $\mathsf{Run}(\mathcal{A}_{\rm d\wedge r}, h)$ does not visit states in $\mathsf{Agn}_{\rm r}$ only, then $\hat{\mathcal{P}}$ is simply unrealizable; or there does not exist an agent winning strategy of reachability game on $\mathcal{A}$. Therefore, suppose the agent decides to achieve both rights $\varphi_{\rm r}$ and $\varphi_{\rm fr}$ immediately after history h, then the agent does not have a strategy that enforces $\varphi_{\rm d}\wedge\varphi_{\rm r}$ with respect to history h, and $\varphi_{\rm fd}\wedge\varphi_{\rm fr}$ after h. Hence, $\hat{\mathcal{P}}$ is unrealizable. □

Step 3. Compute strategy σ. Note that strategy σ needs to lead the play to reach $R_{\rm d}$ and $R_{\rm fd}$ by visiting states in $\mathsf{Agn}_{\rm r\wedge fr}$ only. Moreover, if the agent already decides to achieve $\varphi_{\rm r}$ along h, here the strategy σ should lead the play to reach, instead, $R_{\rm d}\cap R_{\rm fd}\cap R_{\rm r}$, by visiting states in $\mathsf{Agn}_{\rm r\wedge fr}$ only. The following computation focuses on the former case (computing a strategy for the latter case is similar). Therefore, we can build a new DA with reachability-safety condition $\mathcal{A}_1 = (2^{\mathcal{X}\cup\mathcal{Y}}, Q, I, \delta, \mathrm{Reach\text{–}Safe}(R_{\rm d}\cap R_{\rm fd}, \mathsf{Agn}_{\rm r\wedge fr}))$ out of $\mathcal{A} = (2^{\mathcal{X}\cup\mathcal{Y}}, Q, I, \delta, \mathrm{Reach}(R))$, and solve it by reducing to a reachability game, which is analogous to the construction presented in Section 3.1.

Then we solve the reduced reachability game via a least fixpoint computation and obtain $\mathsf{Agn} = \bigcup_{0\le j\le n} \mathsf{Agn}^j$. Note that $I \in \mathsf{Agn}$ indeed holds, which is guaranteed by the reachability game for computing $\mathsf{Agn}_{\rm r\wedge fr}$ in the previous step. We now define a strategy generator that starts serving after h, based on $\mathsf{Agn} = \bigcup_{0\le j\le n} \mathsf{Agn}^j$, represented as a transducer $\hat{\mathcal{T}} = (2^{\mathcal{X}\cup\mathcal{Y}}, Q, I, \varrho, \tau)$, where

- $2^{\mathcal{X}\cup\mathcal{Y}}$, Q and I are the same as in $\mathcal{A}$;
- $\varrho : Q\times 2^{\mathcal{X}} \to 2^Q$ is the transition function s.t. $\varrho(q, X) = \{q' \mid q' = \delta(q, X\cup Y) \text{ and } Y \in \tau(q)\}$;
- $\tau : Q\times 2^{\mathcal{X}} \to 2^{2^{\mathcal{Y}}}$ is the output function s.t. $\forall X \in 2^{\mathcal{X}}$, $\tau(q,X) = \{Y \mid \delta(q, X\cup Y) \in \mathsf{Agn}^j\}$ if $q \in (\mathsf{Agn}^{j+1}\backslash\mathsf{Agn}^j)$, otherwise $\tau(q,X) = 2^{\mathcal{Y}}$.

The transducer $\hat{\mathcal{T}}$, together with history h, generates an agent strategy $\sigma : (2^{\mathcal{X}})^+ \to 2^{\mathcal{Y}}$ as follows: $\forall \xi^k \in (2^{\mathcal{X}})^+$ that is compatible with $h = (X_0\cup Y_0)(X_1\cup Y_1)\dots(X_i\cup Y_i) \in (2^{\mathcal{X}\cup\mathcal{Y}})^*$,

$$\sigma(\xi^k) = \begin{cases} Y_k & \text{if } 0\le k\le i,\\ \textbf{stop} & \text{if } \mathsf{Run}(\mathcal{A}, \pi^{k-1}) \text{ visited } R_{\rm d}\cap R_{\rm fd},\\ \tau(q_k, X_k) & \text{otherwise.}\end{cases}$$

where $\mathsf{Run}(\mathcal{A}, \pi^{k-1}) = q_0 q_1 q_2 \dots q_k$ such that $q_0 = I$, and $\pi^{k-1} = (X_0\cup Y_0)(X_1\cup Y_1)\dots(X_{k-1}\cup Y_{k-1})$.

Intuitively, given a history $h \in (2^{\mathcal{X}\cup\mathcal{Y}})^*$, the final strategy σ is generated by first following h, and starts taking $\hat{\mathcal{T}}$ by choosing suitable agent action to enforce the play to get closer to $R_{\rm d} \cap R_{\rm fd}$, then keeps playing $\mathbf{stop}$ right after visiting $R_{\rm d} \cap R_{\rm fd}$.

Lemma 8. *Let $\hat{\mathcal{P}}$ be a problem of LTL_f synthesis for further duties and rights, and $\hat{\mathcal{T}}$ be constructed as above. Any strategy returned by $\hat{\mathcal{T}}$ solves the synthesis of $\hat{\mathcal{P}}$.*

Proof. Let σ be an arbitrary strategy generated by $\hat{\mathcal{T}}$, i.e., $I \in \mathsf{Agn}_{\rm r\wedge fr}$, and $\gamma \in [\![env]\!]$ be an arbitrary environment strategy that enforces env such that $\mathsf{Play}(\gamma,\sigma)$ has h as a prefix. By construction, $\mathsf{Play}(\gamma,\sigma)$ satisfies the following:

- $\mathsf{Play}(\gamma,\sigma) \models \varphi_{\rm d}$ and $\mathsf{Play}(\gamma,\sigma), |h| \models \varphi_{\rm fd}$, since after h, σ forces $\mathsf{Play}(\gamma,\sigma)$ to get closer to $R_{\rm d} \cap R_{\rm fd}$ at every step until reaching $R_{\rm d} \cap R_{\rm fd}$. Moreover, only after reaching $R_{\rm d} \cap R_{\rm fd}$, σ starts playing $\mathbf{stop}$.
- for every prefix l of $\mathsf{Play}(\gamma,\sigma)$, if l is a prefix of h, then by definition, there exists an agent strategy σ_l that enforces $\varphi_{\rm d} \wedge \varphi_{\rm r}$ with respect to history l, i.e., $\sigma_l \ \rhd_l \ \varphi_{\rm d} \wedge \varphi_{\rm r}$. Otherwise, l has h as a prefix, then by construction, after h, $\mathsf{Play}(\gamma,\sigma)$ only visits states in $\mathsf{Agn}_{\rm r\wedge fr}$. Therefore, for any of the following cases, there exists an expected agent strategy that satisfies the conditions:
  - enforcing $\varphi_{\rm d} \wedge \varphi_{\rm r}$ with respect to h, and $\varphi_{\rm fd}$ after h, if the agent chooses to achieve $\varphi_{\rm r}$ only;
  - enforcing $\varphi_{\rm d}$ with respect to h, and $\varphi_{\rm fd} \wedge \varphi_{\rm fr}$ after h, if the agent chooses to achieve $\varphi_{\rm fr}$ only;
  - enforcing $\varphi_{\rm d} \wedge \varphi_{\rm r}$ with respect to h, and $\varphi_{\rm fd} \wedge \varphi_{\rm fr}$ after h, if the agent chooses to achieve both $\varphi_{\rm r}$ and $\varphi_{\rm fr}$. □

By the construction described above, if the reachability game on $\mathcal{A}$ does not have an agent winning strategy, then $\hat{\mathcal{T}}$ trivially returns *no strategy*, and indeed by Lemma 7, $\hat{\mathcal{P}}$ is unrealizable. As an immediate consequence of Lemmas 7&8, we have:

Theorem 9. *Let $\hat{\mathcal{P}}$ be a problem of LTL_f synthesis for further duties and rights. Realizability of $\hat{\mathcal{P}}$ can be solved by a reduction to a suitable reachability game. Synthesis of $\hat{\mathcal{P}}$ can be solved by generating a strategy from $\hat{\mathcal{T}}$, as constructed above.*

Theorem 10. *Let $\hat{\mathcal{P}}$ be a problem of LTL_f synthesis for further duties and rights. Realizability of $\hat{\mathcal{P}}$ is 2EXPTIME-complete.*

Theorem 11. *Let $\hat{\mathcal{P}}$ be a problem of LTL_f synthesis for further duties and rights. Then computing a strategy solving $\hat{\mathcal{P}}$ can take, in the worst case, double-exponential time in $|env| + |\varphi_{\rm d}| + |\varphi_{\rm r}| + |\varphi_{\rm fd}| + |\varphi_{\rm fr}|$.*

Analogously to Section 3.1, given problem $\hat{\mathcal{P}} = (env, \varphi_{\rm d}, \varphi_{\rm r}, \varphi_{\rm fd}, \varphi_{\rm fr}, h)$, suppose we have a strategy σ for $\hat{\mathcal{P}}$, and while executing σ, the agent wants to satisfy also its rights $\varphi_{\rm r}$ or $\varphi_{\rm fr}$ (maybe both). Then we can again, base on the immediate results obtained from the construction above, to construct three transducers $\hat{\mathcal{T}}_{\rm r}$, $\hat{\mathcal{T}}_{\rm fr}$ and $\hat{\mathcal{T}}_{\rm r\wedge fr}$, corresponding to agent options of achieving also $\varphi_{\rm r}$ only, $\varphi_{\rm fr}$ only, and $\varphi_{\rm r}$ together with $\varphi_{\rm fr}$, following the construction described in Section 3.1. Indeed, if the agent already decides to satisfy $\varphi_{\rm r}$ along h, i.e., before further duties and rights arrive, then the agent needs to stay with its choice of satisfying $\varphi_{\rm r}$. In this case, when further duties and rights arrive, the agent can only choose whether to satisfy also further rights $\varphi_{\rm fr}$.

We extend the cleaning robot example and demonstrate how the robot deal with further duties and rights.

Example 5. *Suppose the cleaning robot decides to achieve also its new rights "emptying the garbage collector" $\varphi_{\rm fr} = \Diamond(\mathit{Collector\_Empty})$ when being on its way to the charging station after cleaning rooms A and C. In other words, the robot already made the decision of achieving its rights "fully charging battery", so the current strategy indeed comes from $\hat{\mathcal{T}}_{\rm r}$. Let us assume that $\hat{\mathcal{T}}_{\rm r\wedge fr}$ is not prepared in advance and by now the running history is h'. Note that h' is different from the running history h generated when the robot receives further duties and rights. The robot will now compute $\hat{\mathcal{T}}_{\rm r\wedge fr}$, and choose a strategy $\sigma_{h'}$ out of $\hat{\mathcal{T}}_{\rm r\wedge fr}$ that allows it to enforce $\varphi_{\rm d} \wedge \varphi_{\rm r}$ with respect to h, and $\varphi_{\rm fd} \wedge \varphi_{\rm fr}$ after h.*

It should be noted that, one can generalize the problem of LTL_f synthesis with further duties and rights to allow multiple further duties and rights. In this case, computing such transducers in advance might lead to a tradeoff, since there can be an exponential number of agent options of choosing which rights to achieve. Therefore, it might be better to just keep the winning regions and compute the strategy for achieving the chosen rights only if and when the agent demands it.

5 Conclusion

We have studied synthesis for duties respecting rights. We have shown that we can actually compute such strategies with a small overhead wrt to the state-of-the-art LTL_f synthesis techniques (Zhu et al. 2017; Bansal et al. 2020; De Giacomo and Favorito 2021). We can do so by enriching the arena to contain also the information needed to handle the rights. For simplicity, we have considered a single duties specification and a single rights specification at the time. Considering multiple duties specifications simultaneously actually is like considering as duties the conjunction of the duties specifications. However, considering multiple rights specifications would require to consider satisfying arbitrary subsets of rights, as chosen by the agent. This can still be done with our techniques, though precomputing solutions as in Section 3 can lead to a combinatorial explosion. In fact, our solution to handle further duties and rights in Section 4 can be already applied to handle multiple duties and rights as well, by considering as history the empty history. Note that the technique presented there also handle contradicting rights, i.e., rights that cannot actually be satisfied simultaneously (but see the discussion at the end of Section 4). These extensions tighten up even more the connection with Deontic Logic, in particular in combination with actions (Gabbay et al. 2013). We leave though exploring this connection to future work.

Acknowledgements

This work is partially supported by the ERC Advanced Grant WhiteMech (No. 834228), the EU ICT-48 2020 project TAILOR (No. 952215), the PRIN project RIPER (No. 20203FFYLK), the JPMorgan AI Faculty Research Award "Resilience-based Generalized Planning and Strategic Reasoning"

References

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*.

Bacchus, F., and Kabanza, F. 2000. Using temporal logics to express search control knowledge for planning. *Artif. Intell.* 116(1-2):123–191.

Bansal, S.; Li, Y.; Tabajara, L. M.; and Vardi, M. Y. 2020. Hybrid Compositional Reasoning for Reactive Synthesis from Finite-Horizon Specifications. In *AAAI*, 9766–9774.

Bernet, J.; Janin, D.; and Walukiewicz, I. 2002. Permissive strategies: from parity games to safety games. *RAIRO Theor. Informatics Appl.* 36(3):261–275.

Camacho, A.; Bienvenu, M.; and McIlraith, S. A. 2018. Finite LTL synthesis with environment assumptions and quality measures. In *KR*, 454–463. AAAI Press.

Cimatti, A.; Pistore, M.; Roveri, M.; and Traverso, P. 2003. Weak, strong, and strong cyclic planning via symbolic model checking. 1–2(147).

De Giacomo, G., and Favorito, M. 2021. Compositional approach to translate LTL_f/LDL_f into deterministic finite automata. In *ICAPS*, 122–130.

De Giacomo, G., and Vardi, M. Y. 2013. Linear Temporal Logic and Linear Dynamic Logic on Finite Traces. In *IJCAI*, 854–860.

De Giacomo, G., and Vardi, M. Y. 2015. Synthesis for LTL and LDL on Finite Traces. In *IJCAI*, 1558–1564.

De Giacomo, G.; Di Stasio, A.; Vardi, M. Y.; and Zhu, S. 2020. Two-stage technique for ltlf synthesis under LTL assumptions. In *KR*, 304–314.

De Giacomo, G.; Di Stasio, A.; Perelli, G.; and Zhu, S. 2021a. Synthesis with Mandatory Stop Actions. In *KR*.

De Giacomo, G.; Di Stasio, A.; Tabajara, L. M.; Vardi, M. Y.; and Zhu, S. 2021b. Finite-trace and generalized-reactivity specifications in temporal synthesis. In *IJCAI*, 1852–1858.

de Silva, L.; Meneguzzi, F.; and Logan, B. 2020. BDI agent architectures: A survey. In *IJCAI*, 4914–4921. ijcai.org.

Finkbeiner, B.; Klein, F.; and Metzger, N. 2021. Live synthesis. *CoRR* abs/2107.01136.

Finkbeiner, B. 2016. Synthesis of reactive systems. In *Dependable Software Systems Engineering*, volume 45 of *NATO Science for Peace and Security Series - D: Information and Communication Security*. 72–98.

Gabaldon, A. 2011. Non-markovian control in the situation calculus. *Artif. Intell.* 175(1):25–48.

Gabbay, D.; Horty, J.; Parent, X.; van der Meyden, R.; and van der Torre, L. 2013. *Handbook of Deontic Logic and Normative System, Volume 2*. College Publications.

Ghallab, M.; Nau, D. S.; and Traverso, P. 2016. *Automated planning and acting*. Cambridge.

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

Kupferman, O., and Vardi, M. Y. 2001. Model Checking of Safety Properties. *Formal Methods in System Design* 19(3):291–314.

Lespérance, Y.; Levesque, H. J.; Lin, F.; and Scherl, R. B. 2000. Ability and knowing how in the situation calculus. *Stud Logica* 66(1):165–186.

Lichtenstein, O.; Pnueli, A.; and Zuck, L. D. 1985. The Glory of the Past. In *Logics of Programs*, 196–218.

Martin, D. 1975. Borel Determinacy. *Annals of Mathematics* 65:363–371.

Pnueli, A., and Rosner, R. 1989. On the Synthesis of a Reactive Module. In *POPL*, 179–190.

Pnueli, A. 1977. The temporal logic of programs. 46–57.

Rintanen, J. 2004. Complexity of planning with partial observability. In *ICAPS*, 345–354. AAAI.

Zhu, S.; Tabajara, L. M.; Li, J.; Pu, G.; and Vardi, M. Y. 2017. Symbolic LTL_f Synthesis. In *IJCAI*, 1362–1369.

Zhu, S.; De Giacomo, G.; Pu, G.; and Vardi, M. Y. 2020. LTL_f synthesis with fairness and stability assumptions. In *AAAI*, 3088–3095.