Responsibility Anticipation and Attribution in LTL_f

Giuseppe De Giacomo[1,2], **Emiliano Lorini**[3], **Timothy Parker**[3] and **Gianmarco Parretti**[2]
[1]University of Oxford
[2]University of Rome "La Sapienza"
[3]IRIT, CNRS, Toulouse University, France
giuseppe.degiacomo@cs.ox.ac.uk, Emiliano.Lorini@irit.fr, icetimp@gmail.com
parretti@diag.uniroma1.it

Abstract

Responsibility is one of the key notions in machine ethics and in the area of autonomous systems. It is a multi-faceted notion involving counterfactual reasoning about actions and strategies. In this paper, we study different variants of responsibility for LTL_f outcomes based on strategic reasoning. We show a connection with notions in reactive synthesis, including the synthesis of winning, dominant, and best-effort strategies. This connection provides a strong computational grounding of responsibility, allowing us to characterize the worst-case computational complexity and devise sound, complete, and optimal algorithms for anticipating and attributing responsibility.

1 Introduction

Responsibility is a key notion in the areas of machine ethics and multi-agent systems. In recent times, there have been several and diverse attempts to formalize it, using various approaches such as game-theoretic tools [Baier *et al.*, 2021; Braham and van Hees, 2012; Lorini and Mühlenbernd, 2018] and logical tools including STIT logic [Lorini *et al.*, 2014; Abarca and Broersen, 2022; Baltag *et al.*, 2021; Lorini and Schwarzentruber, 2011], LTL_f [Parker *et al.*, 2023], ATL [Yazdanpanah *et al.*, 2019; Bulling and Dastani, 2013], logics of strategic and extensive games [Shi, 2024; Naumov and Tao, 2021; Naumov and Tao, 2023], structural equation models [Chockler and Halpern, 2004]. However, the overall picture on the conceptual and computational aspects of responsibility remains rather fragmented. This is due to its polysemic nature and to its many dimensions (e.g., forward-looking vs backward-looking, active vs passive, direct vs indirect, causal vs moral, attributed vs anticipated).

In this paper we provide (*i*) a comprehensive analysis of the complexity of reasoning about strategic responsibility, namely, the responsibility of an agent due to its choice of a given strategy, and (*ii*) a number of algorithms that can be used to automate reasoning about strategic responsibility.

We limit ourselves to analyzing causal responsibility [Vincent, 2011], which considers whether or not an agent caused a certain state of affairs to occur by making a certain choice. It is a general notion of responsibility that is a necessary condition for both legal [Jansen, 2014] and moral [Talbert, 2023] responsibility. We consider the two main forms of responsibility that exist in the literature, active responsibility and passive responsibility. Active responsibility captures the notion of an agent making ω happen, while passive responsibility consists in the agent merely letting ω happen. The distinction between the two notions was proposed in [Lorini *et al.*, 2014] in an action-based setting. Active responsibility corresponds to the notion of *deliberative stit* studied in STIT logic [Belnap *et al.*, 2001] while passive responsibility corresponds to the counterfactual notion of *(something) could have been prevented* (CHP), a fundamental component of the notion of regret as highlighted in [Lorini and Schwarzentruber, 2011]. More recently, a plan-based analysis of active and passive responsibility was proposed in [Parker *et al.*, 2023]. In line with their work we distinguish the concepts of *responsibility anticipation* and *responsibility attribution*. Responsibility anticipation is an *ex ante* notion: it is the responsibility that an agent *could* incur by making a certain choice. Responsibility attribution is an *ex post* notion: it is ascribed to a given agent after the agent and the environment have made their choices and the result of their choices has been revealed.

Passive responsibility for ω, as defined in [Lorini *et al.*, 2014], is a notion of responsibility in a weak sense. As pointed out in [Braham and van Hees, 2012], to strengthen it one could add the requirement that the alternative option the agent could have chosen was preferred by some rationality requirement, implying that the agent has no excuse for letting ω happen. A simple and universal rationality requirement is dominance: there was an alternative recommended option that dominates its actual choice, which is equivalent to saying that the agent's actual choice was not best-effort.

Since the notions of dominance and best-effort are been well-studied in the field of strategy synthesis, it is natural to combine the theory of responsibility with existing work on best-effort synthesis, which is the focus of the present paper. By doing this we gain a better understanding of the computational complexity of reasoning about strategic responsibility and novel algorithms for it. It also leads to the novel notion of strong passive responsibility, which is particularly useful for agents who use responsibility to evaluate their options.

Our formal analysis of responsibility relies on LTL on finite traces (LTL_f), a popular temporal logic which we have

chosen due to its popularity in strategy synthesis, as well as normative specifications in multi-agent systems and AI more broadly. We formally specify various notions of responsibility and relate them to combinations of forms of LTL_f synthesis studied in the literature [De Giacomo and Vardi, 2015; Aminof *et al.*, 2019; Aminof *et al.*, 2021; Aminof *et al.*, 2023]. This allows us to give a computational grounding to such notions of responsibility and devise algorithms and computational complexity characterization to assess them. We also contribute to the LTL_f strategy reasoning by devising algorithms for and proving the computational complexity of *checking strategic properties*, like winning, dominant, and best-effort (vs. synthesizing strategies with these properties [De Giacomo and Vardi, 2015; Aminof *et al.*, 2019; Aminof *et al.*, 2021; Aminof *et al.*, 2023]).[1]

2 Preliminaries

A *trace* over an alphabet of symbols Σ is a finite or infinite sequence of elements from Σ. The empty trace is λ. The length of a trace is $|\pi|$. Traces are indexed starting at zero, and we write $\pi = \pi_0\pi_1\cdots$. For a finite trace π, we denote by $\mathsf{lst}(\pi)$ the index of its last element, i.e., $|\pi|-1$. We denote by $\pi^k = \pi_0\cdots\pi_k$ the prefix of π up to the k-th index.

Linear Temporal Logic on finite traces (LTL_f) is a specification language for expressing temporal properties over finite traces [De Giacomo and Vardi, 2013]. LTL_f has the same syntax as LTL [Pnueli, 1977], which is instead interpreted over infinite traces. Given a set AP of atomic propositions (aka atoms), the LTL_f formulas over AP are: $\omega ::= a \mid \neg\omega \mid \omega \wedge \omega \mid \bigcirc\omega \mid \omega\,\mathcal{U}\,\omega$, where $a \in AP$, and $\bigcirc$ (*Next*) and $\mathcal{U}$ (*Until*) are temporal operators. Additional operators are defined as abbreviations and include: standard Boolean operators of propositional logic; $\bullet\omega \equiv \neg\bigcirc\neg\omega$ (*Weak Next*); $\Diamond\omega \equiv \mathit{true}\,\mathcal{U}\,\omega$ (*Eventually*); and $\Box\omega \equiv \neg\Diamond\neg\omega$ (*Always*). The size of ω, written $|\omega|$, is the number of its subformulas.

LTL_f formulas are interpreted over finite traces π over the alphabet $\Sigma = 2^{AP}$, i.e., consisting of propositional interpretations of atoms. For $i \leq \mathsf{lst}(\pi)$, we have that $\pi_i \in 2^{AP}$ is the i-th interpretation of π. That an LTL_f formula ω *holds* at instant i of trace π, written $\pi, i \models \omega$, is defined inductively:

- $\pi, i \models a$ iff $a \in \pi_i$ (for $a \in AP$);
- $\pi, i \models \neg\omega$ iff $\pi, i \not\models \omega$;
- $\pi, i \models \omega_1 \wedge \omega_2$ iff $\pi, i \models \omega_1$ and $\pi, i \models \omega_2$;
- $\pi, i \models \bigcirc\omega$ iff $i < \mathsf{lst}(\pi)$ and $\pi, i+1 \models \omega$;
- $\pi, i \models \omega_1\,\mathcal{U}\,\omega_2$ iff $\exists j$ such that $i \leq j \leq \mathsf{lst}(\pi)$ and $\pi, j \models \omega_2$, and $\forall k, i \leq k < j$ we have that $\pi, k \models \omega_1$.

We say that π *satisfies* ω, written $\pi \models \omega$, if $\pi, 0 \models \omega$.

We consider LTL_f formulas over $AP = \mathcal{Y} \cup \mathcal{X}$, where $\mathcal{Y}$ and $\mathcal{X}$ are disjoint sets of atoms under control of agent and environment, respectively, as in LTL_f reactive synthesis [De Giacomo and Vardi, 2015]. Traces over $\Sigma = 2^{\mathcal{Y}\cup\mathcal{X}}$ will be denoted $\pi = (Y_0 \cup X_0)(Y_1 \cup X_1)\cdots$ where $Y_i \subseteq \mathcal{Y}$ and $X_i \subseteq \mathcal{X}$ for every $i \geq 0$. Such finite traces are sometimes also called *histories*.

An *agent strategy* is a function $\sigma_{ag} : (2^{\mathcal{X}})^* \to 2^{\mathcal{Y}}$ mapping sequences of environment choices to an agent choice. We require σ_{ag} to be *stopping* [De Giacomo *et al.*, 2021], i.e., the agent stops the execution of any action at some point of the trace, written `stop`. Formally, an agent strategy σ_{ag} is stopping if for every trace $\pi \in (2^{\mathcal{Y}\cup\mathcal{X}})^\omega$ there exists $k \in \mathbb{N}$ such that, for every $j \geq k$, we have that $\sigma_{ag}(\pi^j) = \texttt{stop}$ and, for every $i < k$, we have that $\sigma_{ag}(\pi^i) \neq \texttt{stop}$. An *environment strategy* is a function $\sigma_{env} : (2^{\mathcal{Y}})^+ \to 2^{\mathcal{X}}$ mapping non-empty sequences of agent choices to an environment choice. The domain of σ_{ag} includes the empty sequence λ as we assume that the agent moves first.

A trace $\pi = (Y_0 \cup X_0)\cdots(Y_n \cup X_n)$ is consistent with σ_{ag} if: *(i)* $Y_0 = \sigma(\lambda)$; *(ii)* $Y_i = \sigma_{ag}(X_0\cdots X_{i-1})$ for every $i > 0$; and *(iii)* $\sigma_{ag}(X_0\cdots X_n) = \texttt{stop}$. Similarly, π is consistent with σ_{env} if $X_j = \sigma_{env}(Y_0\cdots Y_j)$ for every $j \geq 0$. We denote by $\mathit{Play}(\sigma_{ag}, \sigma_{env})$ the shortest trace that is consistent with both σ_{ag} and σ_{env}.

Let ψ be an LTL_f formula over $\mathcal{Y} \cup \mathcal{X}$. An agent strategy σ_{ag} is *winning* for (aka *enforces*) ψ if $\mathit{Play}(\sigma_{ag}, \sigma_{env}) \models \psi$ for every environment strategy σ_{env}. Conversely, an environment strategy σ_{env} is *winning* (aka *enforces*) for ψ if *every* finite prefix of $\mathit{Play}(\sigma_{ag}, \sigma_{env})$ satisfies ψ for every agent strategy σ_{ag}. An environment specification $\mathcal{E}$ is an LTL_f formula that is environment enforceable [Aminof *et al.*, 2019] and $\Sigma_{\mathcal{E}}$ is the set of environment strategies that enforce $\mathcal{E}$. An agent strategy σ_{ag} is *winning* for (aka *enforces*) φ under $\mathcal{E}$ if $\mathit{Play}(\sigma_{ag}, \sigma_{env}) \models \varphi$ for every environment strategy $\sigma_{env} \in \Sigma_{\mathcal{E}}$. We may omit φ and/or $\mathcal{E}$ when they are clear from the context and say, e.g., σ_{ag} is winning for φ.

In this paper, we are interested in the result that every LTL_f formula ω can be transformed into a finite automaton that accepts exactly the traces that satisfy ω [De Giacomo and Vardi, 2015]. A *nondeterministic finite automaton* (NFA) is a tuple $\mathcal{N} = (\Sigma, S, s_0, \delta, F)$, where: Σ is a finite input alphabet; S is a finite set of states; $s_0 \in S$ is the initial state; $\delta : S \times \Sigma \to 2^S$ is the transition function; and $F \subseteq S$ is the set of final states. The size of $\mathcal{N}$ is $|S|$. Given a word $\pi = \pi_0\cdots\pi_n \in \Sigma^*$, a *run* of $\mathcal{N}$ in π is a sequence of states $s_0\cdots s_{n+1}$ starting in the initial state of $\mathcal{N}$ and such that $s_{i+1} \in \delta(s_i, \pi_i)$ for every $i \geq 0$. A word π is *accepted* by $\mathcal{N}$ if it has a run whose last reached state is final. The language of $\mathcal{N}$, written $\mathcal{L}(\mathcal{N})$, is the set of words accepted by $\mathcal{N}$. An automaton $\mathcal{N}$ is a *deterministic finite automaton* (DFA) if $|\delta(s,a)| \leq 1$ for every $(s,a) \in S \times \Sigma$. Checking non-emptiness of $\mathcal{L}(\mathcal{N})$, written $\textsc{NonEmpty}(\mathcal{N})$, can be done by checking the existence of a path from the initial state of $\mathcal{N}$ to some final state. Given the NFAs $\mathcal{N}_1$ and $\mathcal{N}_2$ with languages $\mathcal{L}(\mathcal{N}_1)$ and $\mathcal{L}(\mathcal{N}_2)$, respectively, we can build in polynomial time the product NFA $\mathcal{N} = \mathcal{N}_1 \times \mathcal{N}_2$ such that $\mathcal{L}(\mathcal{N}) = \mathcal{L}(\mathcal{N}_1) \cap \mathcal{L}(\mathcal{N}_2)$. We denote nondeterministic and deterministic automata by $\mathcal{N}$ and $\mathcal{A}$, respectively. Every LTL_f formula ω can be transformed into an NFA $\mathcal{N}_\omega = \textsc{ToNFA}(\omega)$ (resp. DFA $\mathcal{A}_\omega = \textsc{ToDFA}(\omega)$) with size at most exponential (resp. doubly-exponential) in $|\omega|$ and whose language is exactly the set of traces satisfying ω [De Giacomo and Vardi, 2015].

A DFA game is a DFA $\mathcal{G}$ with input alphabet $2^{\mathcal{Y}\cup\mathcal{X}}$, where $\mathcal{Y}$ and $\mathcal{X}$ are two disjoint sets under control of agent and environment, respectively. The notions of strategy and play also apply DFA games. An agent strategy is *winning* if

[1] We moved the proofs of some technical results to the supplementary material. Such technical results are marked with *

$Play(\sigma_{ag}, \sigma_{env})$ is accepted by $\mathcal{G}$ for every environment strategy σ_{env}. Conversely, an environment strategy is *winning* if $Play(\sigma_{ag}, \sigma_{env})$ is not accepted by $\mathcal{G}$ for every agent strategy σ_{ag}. The *agent winning region* is the set of states $s \in S$ for which the agent has a winning strategy in the game $\mathcal{G}' = (2^{\mathcal{Y} \cup \mathcal{X}}, S, s, \delta, F)$, i.e., the same game as $\mathcal{G}$, but with initial state s. The *environment winning region* is defined analogously. *Solving* a DFA game is the problem of computing the agent winning region, written $W = \textsc{WinRegion}(\mathcal{G})$. Games played over DFAs are determined, meaning that the agent winning region and the environment winning region partition the state space [Gale and Stewart, 1953]. The environment winning region is denoted $\textsc{EnvWin}(\mathcal{G})$. DFA games can be solved in polynomial time in the size of the game via a least-fixpoint computation [Apt and Grädel, 2011].

We will need to restrict transitions in a DFA to those that do not allow leaving a given set of states. Given a DFA $\mathcal{A}$ and a set of states $Q \subseteq S$, the restriction of $\mathcal{A}$ to Q can be built in polynomial time and is denoted $\mathcal{A}' = \textsc{Restr}(\mathcal{A}, Q)$.

We represent agent strategies as *terminating transducers* [Bansal *et al.*, 2023] $\sigma_{ag} = (2^{\mathcal{X}}, 2^{\mathcal{Y}}, S, s_0, \eta, \kappa, F)$, where: $2^{\mathcal{X}}$ is the input alphabet; $2^{\mathcal{Y}}$ is the output alphabet; S is a finite set of states; $s_0 \in S$ is the initial state; $\eta : S \times 2^{\mathcal{X}} \to S$ is the transition function; $\kappa : S \to 2^{\mathcal{Y}}$ is the output function; and $F \subseteq S$ is the set of terminating states. The size of σ_{ag}, denoted $|\sigma_{ag}|$, is $|S|$. Given an input sequence $X_0 \cdots X_n \in (2^{\mathcal{X}})^*$, the output sequence is $\kappa(s_0) \cdots \kappa(s_n)$, where s_0 is the initial state of σ_{ag} and $s_{i+1} = \eta(s_i, X_i)$ for every $i \geq 0$. A transducer σ_{ag} can be transformed in polynomial time into a DFA $\mathcal{A}_{\sigma_{ag}} = \textsc{ToDFA}(\sigma_{ag})$ whose language is the set of traces consistent with σ_{ag}.

3 Active Responsibility

The first notion of responsibility that we consider is *active responsibility*, which captures: (*i*) that the agent selected a strategy that forced a certain outcome, while (*ii*) having the possibility of selecting an alternative strategy that would have not forced that outcome for at least some environment response. In the context of LTL_f, we formalize this notion as follows:

Definition 1. *[Active Responsibility] The agent has active responsibility for ω under σ_{ag} and $\mathcal{E}$ if: (*i*) $Play(\sigma_{ag}, \sigma_{env}) \models \omega$ for every environment strategy $\sigma_{env} \in \Sigma_{\mathcal{E}}$; and (*ii*) there exists a pair of strategies $(\sigma'_{ag}, \sigma'_{env})$ such that $\sigma'_{env} \in \Sigma_{\mathcal{E}}$ and $Play(\sigma'_{ag}, \sigma'_{env}) \models \neg\omega$.*

By condition (*i*), active responsibility is related to winning strategies used in reactive synthesis [De Giacomo and Vardi, 2015; Aminof *et al.*, 2019] (see Preliminaries); by condition (*ii*), active responsibility is related to a form of strategies called *weak* [Cimatti *et al.*, 2003], which only ensure the existence of an environment strategy for which a certain outcome occurs. Formally, an agent strategy σ'_{ag} is weak for $\neg\omega$ under $\mathcal{E}$ if there exists an environment strategy $\sigma'_{env} \in \Sigma_{\mathcal{E}}$ such that $Play(\sigma'_{ag}, \sigma'_{env}) \models \neg\omega$. The following shows the relation between active responsibility, winning, and weak strategies.

Theorem 1. *The agent has active responsibility for ω under σ_{ag} and $\mathcal{E}$ iff σ_{ag} is winning for ω under $\mathcal{E}$ and there exists some agent strategy σ'_{ag} that is weak for $\neg\omega$ under $\mathcal{E}$.*

We now illustrate how these notions can be applied to assess active responsibility using a relatively simple example.

Example 1. *An agent is assigned the task to take care of a plant. In doing so, the agent must consider the possibility that the environment could also water the plan by raining.*
Consider the following outcomes:
- *ω = "The plant is watered at least once";*
- *$\neg\omega$ = "The plan is never watered".*

Furthermore, consider the following agent strategies:
- *σ_1 = "Water the plant in the morning";*
- *σ_2 = "Never water the plant".*

Finally, consider the following environment specifications:
- *$\mathcal{E}_1$ = "The weather can rain";*
- *$\mathcal{E}_2$ = "The weather surely rains".*

We have the following:
1. *The agent has active responsibility for ω under σ_1 and $\mathcal{E}_1$. In fact, σ_1 is winning for ω under $\mathcal{E}_1$ and a weak strategy for $\neg\omega$ under $\mathcal{E}_1$ exists. One such a weak strategy is σ_2, since it satisfies $\neg\omega$ together with the environment strategy that the weather never rains.*
2. *The agent does not have active responsibility for ω under σ_2 and $\mathcal{E}_1$, since σ_2 is not winning for ω under $\mathcal{E}_1$.*
3. *The agent does not have active responsibility for ω under σ_1 and $\mathcal{E}_2$. In fact, no weak strategy for $\neg\omega$ under $\mathcal{E}_2$ exists, i.e., regardless of which strategy the agent selects, the plant will be watered.*

In what follows, we study the decision problem of checking if the agent has active responsibility. That is: given an LTL_f formula ω, an LTL_f environment specification $\mathcal{E}$, and an agent strategy σ_{ag}, we want to decide whether the agent has active responsibility for ω under σ_{ag} and $\mathcal{E}$. The following establishes the computational complexity of the problem.

Theorem 2. *Checking if the agent has active responsibility for ω under σ_{ag} and $\mathcal{E}$ is:* PSPACE*-complete wrt ω;* 2EXPTIME*-complete wrt $\mathcal{E}$; and polynomial wrt σ_{ag}.*

We prove membership of checking active responsibility by exhibiting a sound and complete algorithm to solve it. Based on Theorem 1, this algorithm checks that σ_{ag} is winning for ω under $\mathcal{E}$ *and* that there exists a weak strategy for $\neg\omega$ under $\mathcal{E}$. First, we give an algorithm for each such check.

We begin by giving an algorithm to check if a strategy σ_{ag} is winning for ω under $\mathcal{E}$, denoted $\textsc{CheckWin}(\omega, \mathcal{E}, \sigma_{ag})$: 1. Construct the NFA $\mathcal{N}_{\neg\omega}$ of $\neg\omega$, the DFA $\mathcal{A}_{\mathcal{E}}$ of $\mathcal{E}$, and the DFA $\mathcal{A}_{\sigma_{ag}}$ of σ_{ag}; 2. Restrict $\mathcal{A}_{\mathcal{E}}$ to the environment winning region and obtain DFA $\mathcal{A}'_{\mathcal{E}}$; and 3. Check language non-emptiness of the product $\mathcal{N} = \mathcal{N}_{\neg\omega} \times \mathcal{A}'_{\mathcal{E}} \times \mathcal{A}_{\sigma_{ag}}$. Intuitively, $\textsc{CheckWin}(\omega, \mathcal{E}, \sigma_{ag})$ checks whether there exists a trace consistent with σ_{ag} that satisfies $\neg\omega$, i.e., that witnesses that σ_{ag} is not winning for ω. By the notion of product of automata, we have that the language of $\mathcal{N}$ is exactly the set of traces that satisfy $\neg\omega$ and are consistent with σ_{ag} and some environment strategy enforcing $\mathcal{E}$. As a result, we have that σ_{ag} is winning for ω under $\mathcal{E}$ iff $\mathcal{L}(\mathcal{N})$ is empty.

The complexity of $\textsc{CheckWin}(\omega, \mathcal{E}, \sigma_{ag})$ wrt ω is dominated by checking language non-emptiness of the product $\mathcal{N} = \mathcal{N}_{\neg\omega} \times \mathcal{A}'_{\mathcal{E}} \times \mathcal{A}_{\sigma_{ag}}$. That can be done on-the-fly while building the product [Vardi and Wolper, 1986]. Being $\mathcal{N}_{\neg\omega}$ exponential in $|\omega|$, we get PSPACE membership wrt

ω. The complexity wrt $\mathcal{E}$ is dominated by computing and restricting the automaton $\mathcal{A}_{\mathcal{E}}$, which is doubly-exponential in $|\mathcal{E}|$, and gives 2EXPTIME membership wrt $\mathcal{E}$. Finally, the product automaton $\mathcal{N}$ is polynomial in $|\sigma_{ag}|$, which gives polynomial complexity wrt σ_{ag}. The complexity of $\textsc{CheckWin}(\omega, \mathcal{E}, \sigma_{ag})$ establishes membership of checking if a strategy is winning for ω under $\mathcal{E}$.

Note that $\textsc{CheckWin}(\omega, \mathcal{E}, \sigma_{ag})$ constructs the DFA of $\mathcal{E}$, which involves a doubly-exponential blowup, while it constructs the NFA of $\neg\omega$, which involves a singly-exponential blowup instead. This is because in order to reason about traces consistent with environment strategies enforcing an environment specification $\mathcal{E}$, we must restrict its DFA $\mathcal{A}_{\mathcal{E}}$ to the environment winning region $W_{\mathcal{E}} = \textsc{EnvWin}(\mathcal{A}_{\mathcal{E}})$. That is, we have that $\mathcal{A}'_{\mathcal{E}} = \textsc{Restr}(\mathcal{A}_{\mathcal{E}}, W_{\mathcal{E}})$ accepts exactly the traces π that satisfy $\mathcal{E}$ and are consistent with some environment strategy enforcing $\mathcal{E}$. Such construction, which leads to a doubly-exponential blowup wrt $\mathcal{E}$, is unavoidable, as also confirmed by the computational complexity of checking if a strategy is winning, established in the following.

Lemma 1. *Checking if σ_{ag} is winning for ω under $\mathcal{E}$ is:* PSPACE*-complete wrt ω; 2*EXPTIME*-complete wrt $\mathcal{E}$; and polynomial wrt σ_{ag}.*

We now give an algorithm to check if a weak strategy for $\neg\omega$ under $\mathcal{E}$ exists, denoted $\textsc{ExistsWeak}(\neg\omega, \mathcal{E})$: 1. Construct the NFA $\mathcal{N}_{\neg\omega}$ of $\neg\omega$ and DFA $\mathcal{A}_{\mathcal{E}}$ of $\mathcal{E}$; 2. Restrict $\mathcal{A}_{\mathcal{E}}$ to the environment winning region and obtain DFA $\mathcal{A}'_{\mathcal{E}}$; and 3. Check language non-emptiness of the product $\mathcal{N} = \mathcal{N}_{\neg\omega} \times \mathcal{A}'_{\mathcal{E}}$. $\textsc{ExistsWeak}(\neg\omega, \mathcal{E})$ is similar to $\textsc{CheckWin}(\omega, \mathcal{E})$. It checks whether there exists a trace satisfying $\neg\omega$ and consistent with some environment strategy enforcing $\mathcal{E}$, i.e., that witnesses that a weak strategy for $\neg\omega$ under $\mathcal{E}$ exists. Its complexity analysis establishes PSPACE and 2EXPTIME membership wrt ω and $\mathcal{E}$, respectively, of checking the existence of a weak strategy. The following establishes the computational complexity of checking the existence of a weak strategy for an arbitrary LTL_f formula.

Lemma 2. *Checking the existence of a weak strategy for ω under $\mathcal{E}$ is:* PSPACE*-complete wrt ω; and 2*EXPTIME*-complete wrt $\mathcal{E}$.*

The algorithm for checking if the agent has active responsibility for ω under σ_{ag} and $\mathcal{E}$ checks both $\textsc{CheckWin}(\omega, \mathcal{E}, \sigma_{ag})$ and $\textsc{ExistsWeak}(\neg\omega, \mathcal{E})$. Such algorithm is sound and complete by Theorem 1. Its complexity establishes membership of checking active responsibility. Hardness of checking active responsibility follows by a polynomial-time reduction from checking if a strategy σ_{ag} is winning for ω under $\mathcal{E}$, whose hardness is shown in Lemma 1. The reduction shows that σ_{ag} is winning for ω under $\mathcal{E}$ iff the agent has active responsibility for $\omega \wedge \neg y$ under $\mathcal{E}$ and σ_{ag}, where y is a new atom under agent's control. Putting together membership and hardness of checking active responsibility, we obtain the complexity characterization in Theorem 2.

4 Passive Responsibility Anticipation

While active responsibility captures the notion of an agent forcing ω to occur, passive responsibility consists in the agent merely letting ω occur. In this context, we distinguish between responsibility *anticipation* and *attribution*. The first is an "ex ante" notion, we want to check whether the currently selected agent strategy allows for ω to occur; the latter is an "ex post" notion, in the actual history, generated by the currently selected agent strategy, ω occurred, and we want to check if the agent let it happen. Note that this distinction is not meaningful for active responsibility: to decide whether the agent has active responsibility for ω we need to check all histories consistent with its strategy, so looking at a specific history is not relevant. In other words, the definitions of anticipation and attribution for active responsibility are equivalent.

In the context of passive responsibility, we also distinguish between *weak* and *strong* responsibility. The former captures that, given the environment response, there exists another agent strategy which would have falsified ω; this is the "classical" notion of passive responsibility and is widely studied in the literature [Lorini *et al.*, 2014; Parker *et al.*, 2023]. The latter captures that there exists another agent strategy that would have been better at falsifying ω, considering all possible environment responses. As emphasized in Section 1, if an agent has strong passive responsibility for ω, then it cannot give an "excuse" for letting ω occur. To the best of our knowledge, this notion is novel to our work.

Definition 2 (Weak Passive Responsibility Anticipation)**.** *The agent anticipates weak passive responsibility for ω under σ_{ag} and $\mathcal{E}$ if: (*i*) there exists an environment strategy $\sigma_{env} \in \Sigma_{\mathcal{E}}$ such that $Play(\sigma_{ag}, \sigma_{env}) \models \omega$; and (*ii*) there exists an agent strategy σ'_{ag} such that $Play(\sigma'_{ag}, \sigma_{env}) \models \neg\omega$.*

The definition of strong passive responsibility is based on the game-theoretic notion of *dominance* [Apt and Grädel, 2011; Aminof *et al.*, 2021]. Let σ_1 and σ_2 be agent strategies. We say that σ_1 *dominates* σ_2 for ω under $\mathcal{E}$, written $\sigma_1 \geq_{\omega|\mathcal{E}} \sigma_2$, if $Play(\sigma_2, \sigma_{env}) \models \omega$ then $Play(\sigma_1, \sigma_{env}) \models \omega$ for every environment strategy $\sigma_{env} \in \Sigma_{\mathcal{E}}$. Furthermore, we say that σ_1 *strictly dominates* σ_2, written $\sigma_1 >_{\omega|\mathcal{E}} \sigma_2$, if $\sigma_1 \geq_{\omega|\mathcal{E}} \sigma_2$ and $\sigma_2 \not\geq_{\omega|\mathcal{E}} \sigma_1$. Intuitively: $\sigma_1 \geq_{\omega|\mathcal{E}} \sigma_2$ means that σ_1 is at least as good as σ_2 (wrt ω and $\mathcal{E}$); $\sigma_1 >_{\omega|\mathcal{E}} \sigma_2$ means that σ_1 is strictly better than σ_2 (wrt ω and $\mathcal{E}$).

Definition 3 (Strong Passive Responsibility Anticipation)**.** *The agent anticipates strong passive responsibility for ω under σ_{ag} and $\mathcal{E}$ if: (*i*) there exists an environment strategy $\sigma_{env} \in \Sigma_{\mathcal{E}}$ such that $Play(\sigma_{ag}, \sigma_{env}) \models \omega$; and (*ii*) there exists an agent strategy σ'_{ag} such that $\sigma'_{ag} \geq_{\neg\omega|\mathcal{E}} \sigma_{ag}$ and $Play(\sigma'_{ag}, \sigma_{env}) \models \neg\omega$.*

Weak and strong passive responsibility anticipation are related to a form of strategies called *dominant* and *best-effort*, respectively, which have been recently investigated in the literature on reactive synthesis [Aminof *et al.*, 2021; Aminof *et al.*, 2023]. In what follows, we briefly review dominant and best-effort strategies in the context of LTL_f.

Formally, an agent strategy σ_{ag} is dominant for ω under $\mathcal{E}$ if $\sigma_{ag} \geq_{\omega|\mathcal{E}} \sigma'_{ag}$ for every agent strategy σ'_{ag}. An agent strategy is best-effort for ω under $\mathcal{E}$ if there does not exist another agent strategy σ'_{ag} such that $\sigma'_{ag} >_{\omega|\mathcal{E}} \sigma_{ag}$. Best-effort strategies always exist and capture the game-theoretic rationality principle that the agent should not use a strategy that is strictly dominated [Aminof *et al.*, 2021]. If the agent uses

a strictly dominated strategy σ_1, say $\sigma_2 >_{\omega|\mathcal{E}} \sigma_1$, then it is not doing its best wrt ω: if it used σ_2 instead, it would have satisfied ω against a strictly larger set of environment strategies. Dominant strategies [Aminof *et al.*, 2023] are slightly stronger than best-effort strategies, but do not always exist. Intuitively, a dominant strategy satisfies ω against every environment strategy for which it is possible to do so. Winning, dominant, and best-effort strategies are related [Aminof *et al.*, 2023]. Specifically: if a winning strategy exists, the winning strategies are exactly the dominant strategies; and if a dominant strategy exists, the dominant strategies are exactly the best-effort strategies. The following establishes the relation between weak (resp. strong) passive responsibility anticipation and dominant (resp. best-effort) strategies.

Theorem 3. *The agent anticipates weak (resp. strong) passive responsibility for ω under σ_{ag} and $\mathcal{E}$ iff σ_{ag} is not dominant (resp. is not best-effort) for $\neg\omega$ under $\mathcal{E}$.*

Theorem 3 leads to the following observations.

1. If no dominant strategy for $\neg\omega$ exists, the agent anticipates weak passive responsibility for ω regardless of the strategy it selects. In the context of LTL_f, *dominant strategies rarely exists* [Aminof *et al.*, 2023], i.e., that the agent anticipates weak passive responsibility is often unavoidable;

2. A best-effort strategy for $\neg\omega$ always exists, meaning that the agent can always avoid to anticipate strong passive responsibility for ω by selecting one such a strategy.

Putting such observations together with the relation between dominant and best-effort strategies, we can observe the following. The agent should *always* select a strategy σ_{ag} that is best-effort for $\neg\omega$. On one hand, we have that σ_{ag} avoids strong passive responsibility anticipation for ω. On the other hand, if a dominant strategy for $\neg\omega$ exists, we have that σ_{ag} is also dominant for $\neg\omega$, and the agent does not anticipate neither weak nor strong passive responsibility for ω.

In other words, best-effort strategies unify under one strategy concept weak and strong passive responsibility anticipation. Given the relation between best-effort strategies and strong passive responsibility anticipation established in Theorem 3, we argue that strong passive responsibility is strictly more useful than *classical* weak passive responsibility to develop and evaluate responsibility-aware autonomous agents. This makes our notion of strong passive responsibility a significant contribution to the field of responsibility analysis.

We now illustrate such notions using the example of the autonomous agent that has to water a plant.

Example 2. *Consider the same outcomes, agent strategies, and environment specification in Example 1.*

1. *The agent does not anticipate neither weak nor strong passive responsibility for ω under σ_2 and $\mathcal{E}_1$. In fact, σ_2 is dominant (and hence, best-effort) for $\neg\omega$ under $\mathcal{E}_1$. To see this, observe that σ_2 is the unique agent strategy for which $\neg\omega$ is satisfied together with the environment strategy that the weather never rains.*
2. *The agent anticipates weak and strong passive responsibility for ω under σ_1 and $\mathcal{E}_1$. In fact, σ_1 is* not *dominant (and hence, not best-effort) for $\neg\omega$ under $\mathcal{E}_1$.*

Consider the following outcomes and agent strategy

- ω' = *"The plant is either never watered or watered more than once"*
- $\neg\omega'$ = *"The water is watered exactly once"*
- σ_3 = *"Water the plant day and night".*

We have the following.

3. *The agent does not anticipate strong passive responsibility for ω' under σ_2 and $\mathcal{E}_1$. In fact, σ_2 is best-effort for $\neg\omega'$ under $\mathcal{E}_1$. However, the agent anticipates weak passive responsibility for ω' under σ_2 and $\mathcal{E}_1$. This is because σ_2 not dominate σ_1 wrt $\neg\omega'$: σ_2 does not satisfy $\neg\omega'$ together with the environment strategy that the weather never rains, whereas σ_1 does.*
4. *The agent anticipates both weak and strong passive responsibility for ω' under σ_3 and $\mathcal{E}_1$. In fact, σ_3 is not best-effort for $\neg\omega'$ under $\mathcal{E}_1$. To see this, observe that σ_3 never satisfies $\neg\omega'$ for every environment strategy. On the other hand, σ_2 satisfies $\neg\omega'$ together with the environment strategy that the weather rains only in the morning, i.e., σ_2 strictly dominates σ_3 wrt $\neg\omega'$ and $\mathcal{E}_1$.*

We now study the decision problem of checking if the agent anticipates weak (resp. strong) passive responsibility. That is: given an LTL_f formula ω, an LTL_f environment specification $\mathcal{E}$, and an agent strategy σ_{ag}, we want to decide whether the agent anticipates weak (resp. strong) passive responsibility for ω under σ_{ag} and $\mathcal{E}$. The following establish the computational complexity of the problems.

Theorem 4. *Checking if the agent anticipates weak passive responsibility for ω under σ_{ag} and $\mathcal{E}$ is:* PSPACE*-complete wrt ω; 2*EXPTIME*-complete wrt $\mathcal{E}$; and polynomial wrt σ_{ag}.*

Theorem 5. *Checking if the agent anticipates strong passive responsibility for ω under σ_{ag} and $\mathcal{E}$ is: 2*EXPTIME*-complete wrt ω and $\mathcal{E}$; and polynomial wrt σ_{ag}.*

We prove membership in Theorems 4 and 5 constructively by exhibiting a sound and complete algorithm for checking their corresponding responsibility notion.

We consider weak passive responsibility first. Based on Theorem 3, checking weak passive responsibility anticipation can be reduced to checking if a strategy is (not) dominant. In what follows, we give an algorithm to check if a strategy σ_{ag} is dominant for ω under $\mathcal{E}$, denoted $\textsc{CheckDom}(\omega, \mathcal{E}, \sigma_{ag})$:

1. Let $\mathcal{Y}' = \{y' \text{ s.t. } y \in \mathcal{Y}\}$ and $\mathcal{X}' = \{x' \text{ s.t. } x \in \mathcal{X}\}$;
2. Define ω' and $\mathcal{E}'$ as copies of ω and $\mathcal{E}$ over $\mathcal{Y}' \cup \mathcal{X}'$;
3. $\mathcal{N}_{\neg\omega} = \textsc{ToNFA}(\neg\omega)$ and $\mathcal{N}_{\omega'} = \textsc{ToNFA}(\omega')$
4. $\mathcal{A}_{\mathcal{E}} = \textsc{ToDFA}(\mathcal{E})$ and $\mathcal{A}_{\mathcal{E}'} = \textsc{ToDFA}(\mathcal{E}')$
5. $W_{\mathcal{E}} = \textsc{EnvWin}(\mathcal{A}_{\mathcal{E}})$ and $W_{\mathcal{E}'} = \textsc{EnvWin}(\mathcal{A}_{\mathcal{E}'})$
6. $\mathcal{A}'_{\mathcal{E}} = \textsc{Restr}(\mathcal{A}_{\mathcal{E}}, W_{\mathcal{E}})$ and $\mathcal{A}'_{\mathcal{E}'} = \textsc{Restr}(\mathcal{A}_{\mathcal{E}'}, W'_{\mathcal{E}})$
7. $\mathcal{A}_{\sigma_{ag}} = \textsc{ToDFA}(\sigma_{ag})$
8. $\mathcal{N} = \mathcal{A}_{\mathcal{Y}\neq\mathcal{Y}'} \times (\mathcal{N}_{\neg\omega} \times \mathcal{A}'_{\mathcal{E}} \times \mathcal{A}_{\sigma_{ag}}) \times (\mathcal{N}_{\omega'} \times \mathcal{A}'_{\mathcal{E}'})$
9. **if** $\textsc{NonEmpty}(\mathcal{N})$ **return** *false*; **else return** *true*

Intuitively, $\textsc{CheckDom}(\omega, \mathcal{E}, \sigma_{ag})$ checks the existence of a pair of traces (π, π') such that, for some environment strategy σ_{env} enforcing $\mathcal{E}$, we have that $\pi = \mathit{Play}(\sigma_{ag}, \sigma_{env}) \models \neg\omega$, and, for some agent strategy σ'_{ag} distinct from the currently selected agent strategy σ_{ag}, we have that $\pi' = \mathit{Play}(\sigma'_{ag}, \sigma_{env}) \models \omega$, i.e., π and π' witness that σ_{ag} is not dominant for ω, as it does not dominate σ'_{ag}. To do so, $\textsc{CheckDom}(\omega, \mathcal{E}, \sigma_{ag})$ checks language non-

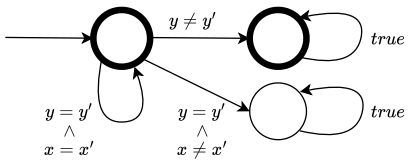

Figure 1: DFA $\mathcal{A}_{\mathcal{Y}\neq\mathcal{Y}'}$. Final states are in bold. $y = y'$ (resp. $y \neq y'$) denotes equal (resp. distinct) assignments of atoms in $\mathcal{Y}$ and $\mathcal{Y}'$. Similarly for $x = x'$ (resp. $x \neq x'$).

emptiness of an automaton $\mathcal{N}$ obtained by product of several automata, including:

- $\mathcal{A}_{\mathcal{Y}\neq\mathcal{Y}'}$, whose language is the set of pairs of traces (π, π') such that, if the agent makes the same choices in both π and π', then the environment does the same (see Figure 1);
- $(\mathcal{N}_{\neg\omega} \times \mathcal{A}'_{\mathcal{E}} \times \mathcal{A}_{\sigma_{ag}})$, whose language is the set of traces $\pi = Play(\sigma_{ag}, \sigma_{env}) \models \neg\omega$, where σ_{ag} is the input strategy and σ_{env} is some environment strategy enforcing $\mathcal{E}$;
- $(\mathcal{N}_{\omega'} \times \mathcal{A}'_{\mathcal{E}'})$, whose language is the set of traces $\pi' = Play(\sigma'_{ag}, \sigma_{env}) \models \omega$ for some agent strategy σ'_{ag} and some environment strategy σ_{env} enforcing $\mathcal{E}$.

By the notion of product of automata, the language of $\mathcal{N}$ is the set of pairs of traces (π, π') mentioned above. It follows that σ_{ag} is dominant for ω under $\mathcal{E}$ iff $\mathcal{L}(\mathcal{N})$ is empty.

The complexity of CHECKDOM$(\omega, \mathcal{E}, \sigma_{ag})$ gives PSPACE and 2EXPTIME membership wrt ω and $\mathcal{E}$, respectively, and polynomial complexity wrt σ_{ag}. Its analysis is similar to that of CHECKWIN$(\omega, \mathcal{E}, \sigma_{ag})$. The computational complexity of checking if a strategy is dominant is the following:

Lemma 3. *Checking if σ_{ag} is dominant for ω under $\mathcal{E}$ is:* PSPACE*-complete wrt ω; 2*EXPTIME*-complete wrt $\mathcal{E}$; and polynomial wrt σ_{ag}.*

The algorithm for checking if the agent anticipates weak passive responsibility for ω under σ_{ag} and $\mathcal{E}$ checks whether σ_{ag} is *not* dominant for $\neg\omega$ under $\mathcal{E}$, written $\neg$CHECKDOM$(\neg\omega, \mathcal{E}, \sigma_{ag})$. Such algorithm is sound and complete by Theorem 3 and its complexity establishes membership of checking weak passive responsibility anticipation. Hardness of weak passive responsibility anticipation follows by its relation with dominant strategies, established in Theorem 3, and hardness of checking if a strategy is dominant, established in Lemma 3. Putting together membership and hardness of weak passive responsibility anticipation, we obtain the complexity characterization in Theorem 4.

We now turn to the decision problem of checking strong passive responsibility anticipation. Based on Theorem 3, this problem is reducible to checking if a strategy is (not) best-effort. In what follows, we give an algorithm to check if a strategy σ_{ag} is best-effort for ω under $\mathcal{E}$. This algorithm bases on solving a DFA game, using the notions of winning and weak strategy. We reviewed winning strategies in DFA games in Section 2. In the context of a DFA game $\mathcal{G}$, an agent strategy σ_{ag} is *weak* if there exists an environment strategy σ_{env} such that $Play(\sigma_{ag}, \sigma_{env})$ is accepted by $\mathcal{G}$. The *weakly winning region* W' is defined analogously to the winning region, i.e., W' is the set of states of $\mathcal{G}$ where the agent has a weak strategy. The weakly winning region W' can be computed in polynomial time with a least fixpoint computation over the state space of $\mathcal{G}$, written $W' = \text{WEAKREGION}(\mathcal{G})$.

The following is the algorithm for checking if a strategy σ_{ag} is best-effort for ω under $\mathcal{E}$, denoted CHECKBE$(\omega, \mathcal{E}, \sigma_{ag})$.

1. $\mathcal{A}_\omega = \text{TODFA}(\omega)$ and $\mathcal{A}_\mathcal{E} = \text{TODFA}(\mathcal{E})$;
2. $W_\mathcal{E} = \text{ENVWIN}(\mathcal{A}_\mathcal{E})$; and $\mathcal{A}'_\mathcal{E} = \text{RESTR}(\mathcal{A}_\mathcal{E}, W_\mathcal{E})$;
3. $\mathcal{G} = \mathcal{A}_\omega \times \mathcal{A}'_\mathcal{E}$;
4. $W = \text{WINREGION}(\mathcal{G})$ and $W' = \text{WEAKREGION}(\mathcal{G})$;
5. $\mathcal{A}_{\sigma_{ag}} = \text{TODFA}(\sigma_{ag})$
6. $\mathcal{G}_{\sigma_{ag}} = \mathcal{G} \times \mathcal{A}_{\sigma_{ag}}$.Say: $S_\mathcal{G} \times S_{\sigma_{ag}}$ is the state set of $\mathcal{G}_{\sigma_{ag}}$; s_0 is the initial state of $\mathcal{G}_{\sigma_{ag}}$; and F is the set of final states of $\mathcal{G}_{\sigma_{ag}}$;
7. $W_{\sigma_{ag}} = \{(s_\mathcal{G}, s_{\sigma_{ag}}) \in S_\mathcal{G} \times S_{\sigma_{ag}}) \mid s_\mathcal{G} \in W\}$;
8. $W'_{\sigma_{ag}} = \{(s_\mathcal{G}, s_{\sigma_{ag}}) \in S_\mathcal{G} \times S_{\sigma_{ag}}) \mid s_\mathcal{G} \in W'\}$;
9. **for** every state $s \in S_\mathcal{G} \times S_{\sigma_{ag}}$ reachable from s_0:
   (A) **if** $s \in W_{\sigma_{ag}}$ and there exists a path $\rho = s \cdots s'$ such that $s' \notin F$ and ρ can't be extended to reach F **return** *false*;
   (B) **else if** $s \in W'_{\sigma_{ag}} \setminus W_{\sigma_{ag}}$ and no path from s to F exists **return** *false*;
10. **return** *true*

CHECKBE$(\omega, \mathcal{E}, \sigma_{ag})$ uses a characterization of best-effort strategies based on the notion of *value* of a history, which we sketch [Aminof *et al.*, 2021]. Intuitively, the value of a history h is: $+1$ ("winning") if the agent has a winning strategy for ω under $\mathcal{E}$ starting from h; 0 ("pending") if the agent has a weak strategy for ω under $\mathcal{E}$ starting from h; and -1 ("losing") otherwise. A best-effort strategy is one that witness the maximum value of each history consistent with it. The DFA game $\mathcal{G}$ solved by CHECKBE$(\omega, \mathcal{E}, \sigma_{ag})$ (Lines 1-4) satisfies the following: histories whose runs lead to a state in the winning region W are winning; among the remaining histories, those whose runs lead to a state in the weakly winning region W' are pending; and the remaining are losing [Aminof *et al.*, 2021]. By the notion of product of automata, the DFA $\mathcal{G}_{\sigma_{ag}}$ (Lines 6-8) satisfies the following: histories whose runs lead to $W_{\sigma_{ag}}$ are winning and consistent with σ_{ag}; and, among the remaining histories, those whose runs lead to $W'_{\sigma_{ag}}$ are pending and consistent with σ_{ag}. CHECKBE$(\omega, \mathcal{E}, \sigma_{ag})$ checks that the agent is able to: (A) reach a final state from every state in $W_{\sigma_{ag}}$ regardless of the environment response, so that σ_{ag} is winning in every winning history consistent with it; (B) reach a final state from every state in $W'_{\sigma_{ag}} \setminus W_{\sigma_{ag}}$ for some environment response, so that σ_{ag} is weak in every pending history consistent with it. By the history-based characterization of best-effort strategies, σ_{ag} is best-effort for ω under $\mathcal{E}$ iff neither (A) nor (B) is violated.

The complexity of CHECKBE$(\omega, \mathcal{E}, \sigma_{ag})$ is dominated by constructing and solving games of doubly-exponential size in $|\omega|$ and $|\mathcal{E}|$, so that CHECKBE$(\omega, \mathcal{E}, \sigma_{ag})$ establishes 2EXPTIME membership wrt ω and $\mathcal{E}$. The size of $\mathcal{G}_{\sigma_{ag}}$ is polynomial in $|\sigma_{ag}|$, which establishes polynomial complexity wrt σ_{ag}. Hardness and computational complexity of checking if a strategy is best-effort are established in the following:

Lemma 4. *Checking if σ_{ag} is best-effort for ω under $\mathcal{E}$ is: 2*EXPTIME*-complete wrt ω and $\mathcal{E}$; and polynomial wrt σ_{ag}.*

The algorithm for checking if the agent anticipates strong passive responsibility for ω under σ_{ag} and $\mathcal{E}$ checks whether σ_{ag} is not best-effort for $\neg\omega$ under $\mathcal{E}$, written $\neg$CHECKBE$(\neg\omega, \mathcal{E}, \sigma_{ag})$. Such algorithm is sound and complete by Theorem 3 and its complexity establishes membership of checking strong passive responsibility anticipation. Hardness of strong passive responsibility anticipation follows by its relation with best-effort strategies, established in Theorem 3, and the complexity of checking if a strategy is best-effort, established in Lemma 4. Putting together membership and hardness of strong passive responsibility anticipation, we get the complexity characterization in Theorem 5.

5 Passive Responsibility Attribution

We now consider passive responsibility attribution after a history where ω occurs. Passive responsibility on histories is attributed "ex post" by counterfactually reasoning about what could have happened in the past, "fixing" the environment response to also comply with the history under consideration.

Definition 4 (Weak (resp. Strong) Passive Responsibility Attribution)**.** *The agent has weak (resp. strong) passive responsibility for ω under σ_{ag}, $\mathcal{E}$, and h such that $h \models \omega$ if: (*i*) there exists an environment strategy $\sigma_{env} \in \Sigma_{\mathcal{E}}$ such that h is consistent with σ_{env} and $h = Play(\sigma_{ag}, \sigma_{env})$; and (*ii*) there exists another agent strategy σ'_{ag} (resp. $\sigma'_{ag} \geq_{\neg\omega|\mathcal{E}} \sigma_{ag}$) such that $Play(\sigma'_{ag}, \sigma_{env}) \models \neg\omega$.*

As with passive responsibility anticipation, attribution of strong passive responsibility implies attribution of weak passive responsibility, and weak and strong passive responsibility attribution are related to dominant and best-effort strategies, respectively. The relation follows by observing that we can construct an LTL$_f$ environment specification $\mathcal{E}_h$ that captures exactly the environment strategies consistent with h. Formally, let $h = (Y_0 \cup X_0) \cdots (Y_n \cup X_n)$ be a history. That h is consistent with an environment strategy σ_{env} means that $X_i = \sigma_{env}(Y_0 \cdots Y_i)$ for every $i \geq 0$. Such behavior can be captured by the following LTL$_f$ environment specification.

$$\mathcal{E}_h = (Y_0 \supset X_0) \wedge ((Y_0 \wedge \bullet^1 Y_1) \supset \bullet^1 X_1) \wedge \\ \cdots \wedge ((Y_0 \wedge \bullet^1 Y_1 \wedge \cdots \wedge \bullet^n Y_n) \supset \bullet^n X_n)$$

It easy to see that h is consistent with σ_{env} iff σ_{env} enforces $\mathcal{E}_h$. It follows that $\mathcal{E} \wedge \mathcal{E}_h$ captures exactly the set of environment strategies enforcing $\mathcal{E}$ and consistent with h. We then derive the relation between passive responsibility attribution and dominant and best-effort strategies.

Theorem 6. *The agent is attributed weak (resp. strong) passive responsibility for ω under σ_{ag}, $\mathcal{E}$, and h iff σ_{ag} is not dominant (resp. best-effort) for $\neg\omega$ under $\mathcal{E} \wedge \mathcal{E}_h$.*

Next. we turn to the decision problem of checking weak (resp. strong) passive responsibility attribution on histories: given an LTL$_f$ formula ω, an LTL$_f$ environment specification $\mathcal{E}$, an agent strategy σ_{ag}, and an history h, we want to decide whether the agent has weak (resp. strong) passive responsibility for ω under $\mathcal{E}$, σ_{ag} and h or not. The computational complexity of the problems is established in the following.

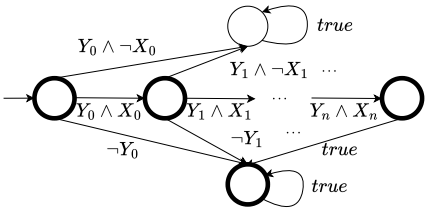

Figure 2: DFA $\mathcal{A}_{\mathcal{E}_h}$ of LTL$_f$ environment specification $\mathcal{E}_h$ that captures environment strategies σ_{env} such that history $h = (Y_0 \cup X_0) \cdots (Y_n \cup X_n)$ is consistent with σ_{env}.

Theorem 7. *Checking if the agent has weak passive responsibility for ω under σ_{ag}, $\mathcal{E}$, and h is:* PSPACE*-complete wrt ω; 2*EXPTIME*-complete wrt $\mathcal{E}$; polynomial wrt σ_{ag}; and polynomial wrt h.*

Theorem 8. *Checking if the agent has strong passive responsibility for ω under σ_{ag}, $\mathcal{E}$, and h is: 2*EXPTIME*-complete wrt ω and $\mathcal{E}$; polynomial wrt σ_{ag}; and polynomial wrt h.*

Based on Theorem 6, we can reduce checking if the agent has weak (resp. strong) passive responsibility for ω under $\mathcal{E}$, σ_{ag}, and h to checking whether σ_{ag} is *not* dominant (resp. best-effort) for $\neg\omega$ under $\mathcal{E} \wedge \mathcal{E}_h$. We can do so by adapting the algorithms in Section 4, written $\neg$CHECKDOM$(\neg\omega, \mathcal{E} \wedge \mathcal{E}_h, \sigma_{ag})$ (resp. $\neg$CHECKBE$(\neg\omega, \mathcal{E} \wedge \mathcal{E}_h, \sigma_{ag})$). The complexity of such algorithms establish membership of weak (resp. strong) passive responsibility attribution wrt ω, $\mathcal{E}$, and σ_{ag}. To obtain polynomial polynomial complexity wrt h, observe that DFA $\mathcal{A}_{\mathcal{E} \wedge \mathcal{E}_h}$ can be constructed in polynomial time as the product $\mathcal{A}_{\mathcal{E}} \times \mathcal{A}_{\mathcal{E}_h}$. In turn, $\mathcal{A}_{\mathcal{E}_h}$ can be constructed in polynomial time in the length of h as in Figure 2. To see why $\mathcal{A}_{\mathcal{E}_h}$ captures exactly the environment strategies consistent with $h = (Y_0 \cup X_0) \cdots (Y_n \cup X_n)$, observe that transitions of the form $Y_i \wedge \neg X_i$ lead to a non-final sink state, as no environment strategy σ_{env} consistent with h plays $\neg X_i$ when the agent plays $Y_0 \cdots Y_i$. Hardness of weak (resp. strong) passive responsibility attribution follows by its relation with dominant (resp. best-effort) strategies, established in Theorem 6, and hardness of checking if a strategy is dominant (resp. best-effort), established in Lemma 3 (resp. Lemma 4). Putting together membership and hardness of weak (resp. strong) passive responsibility attribution, we get the complexity characterization in Theorem 7 (resp. Theorem 8).

6 Conclusion

We observe that once these responsibility notions are characterized, they can be used not only to assess responsibility but also during strategy synthesis. This ensures the generation of strategies that avoid unintended responsibility, opening a research avenue into *responsibility-aware strategy synthesis*.

We focused on a single agent with a first-person perspective, examining responsibility from the agent's viewpoint rather than any third-person perspective. However, our work could be extended to a *multi-agent setting*, where each agent operates within an environment shaped by both the environment itself and the actions of other agents. We plan to explore these directions further in the future.

Acknowledgments

This work is supported in part by the ERC Advanced Grant WhiteMech (No. 834228), the PRIN project RIPER (No. 20203FFYLK), the PNRR MUR project FAIR (No.PE0000013), and the UKRI Erlangen AI Hub on Mathematical and Computational Foundations of AI. Emiliano Lorini is supported by the ANR projects EpiRL (grant number ANR-22-CE23-0029) and ALoRS (grant number ANR-21-CE23-0018-01). Gianmarco Parretti is supported by the Italian National Ph.D. on AI at "La Sapienza".

References

[Abarca and Broersen, 2022] A. I. R. Abarca and J. M. Broersen. A stit logic of responsibility. In *AAMAS*, pages 1717–1719. International Foundation for Autonomous Agents and Multiagent Systems (IFAAMAS), 2022.

[Aminof *et al.*, 2019] Benjamin Aminof, Giuseppe De Giacomo, Aniello Murano, and Sasha Rubin. Planning under LTL environment specifications. In *ICAPS*, pages 31–39, 2019.

[Aminof *et al.*, 2021] Benjamin Aminof, Giuseppe De Giacomo, and Sasha Rubin. Best-effort synthesis: Doing your best is not harder than giving up. In *IJCAI*, pages 1766–1772, 2021.

[Aminof *et al.*, 2023] Benjamin Aminof, Giuseppe De Giacomo, and Sasha Rubin. Reactive synthesis of dominant strategies. In *AAAI*, pages 6228–6235, 2023.

[Apt and Grädel, 2011] Krzysztof R. Apt and Erich Grädel, editors. *Lectures in Game Theory for Computer Scientists*. Cambridge University Press, 2011.

[Baier *et al.*, 2021] C. Baier, F. Funke, and R. Majumdar. A game-theoretic account of responsibility allocation. In *IJCAI*, pages 1773–1779, 2021.

[Baltag *et al.*, 2021] A. Baltag, I. Canavotto, and S. Smets. Causal agency and responsibility: A refinement of STIT logic. In Alessandro Giordani and Jacek Malinowski, editors, *Logic in High Definition, Trends in Logical Semantics*, pages 149–176. 2021.

[Bansal *et al.*, 2023] Suguman Bansal, Yong Li, Lucas M. Tabajara, Moshe Y. Vardi, and Andrew M. Wells. Model checking strategies from synthesis over finite traces. In *ATVA (1)*, volume 14215 of *Lecture Notes in Computer Science*, pages 227–247. Springer, 2023.

[Belnap *et al.*, 2001] N. Belnap, M. Perloff, and M. Xu. *Facing the Tuture: Agents and Choices in our Indeterminist World*. Oxford University Press, New York, 2001.

[Braham and van Hees, 2012] M. Braham and M. van Hees. An anatomy of moral responsibility. *Mind*, 121(483):601–634, 2012.

[Bulling and Dastani, 2013] N. Bulling and M. Dastani. Coalitional responsibility in strategic settings. In *CLIMA*, Lecture Notes in Computer Science, pages 172–189, 2013.

[Chockler and Halpern, 2004] H. Chockler and J. Halpern. Responsibility and blame: A structural-model approach. *Journal of Artificial Intelligence Research*, 22:93–115, 2004.

[Cimatti *et al.*, 2003] Alessandro Cimatti, Marco Pistore, Marco Roveri, and Paolo Traverso. Weak, strong, and strong cyclic planning via symbolic model checking. *AIJ*, 1–2(147):35–84, 2003.

[De Giacomo and Vardi, 2013] Giuseppe De Giacomo and Moshe Y. Vardi. Linear temporal logic and linear dynamic logic on finite traces. In *IJCAI*, pages 854–860, 2013.

[De Giacomo and Vardi, 2015] Giuseppe De Giacomo and Moshe Y. Vardi. Synthesis for LTL and LDL on finite traces. In *IJCAI*, pages 854–860, 2015.

[De Giacomo *et al.*, 2021] Giuseppe De Giacomo, Antonio Di Stasio, Giuseppe Perelli, and Shufang Zhu. Synthesis with mandatory stop actions. In *KR*, pages 237–246, 2021.

[Gale and Stewart, 1953] David Gale and Frank M Stewart. Infinite games with perfect information. *Contributions to the Theory of Games*, 2(245-266):2–16, 1953.

[Jansen, 2014] Nils Jansen. The idea of legal responsibility. *Oxford Journal of Legal Studies*, 34(2):221–252, 2014.

[Lorini and Mühlenbernd, 2018] E. Lorini and R. Mühlenbernd. The long-term benefits of following fairness norms under dynamics of learning and evolution. *Fundamenta Informaticae*, 158(1-3):121–148, 2018.

[Lorini and Schwarzentruber, 2011] E. Lorini and F. Schwarzentruber. A logic for reasoning about counterfactual emotions. *Artificial Intelligence*, 175(3-4):814–847, 2011.

[Lorini *et al.*, 2014] E. Lorini, D. Longin, and E. Mayor. A logical analysis of responsibility attribution: Emotions, individuals and collectives. *Journal of Logic and Computation*, 24(6):1313–1339, 2014.

[Naumov and Tao, 2021] P. Naumov and J. Tao. Two forms of responsibility in strategic games. In *IJCAI*, pages 1989–1995, 2021.

[Naumov and Tao, 2023] P. Naumov and J. Tao. Counterfactual and seeing-to-it responsibilities in strategic games. *Annals of Pure and Applied Logic*, 174(10):103353, 2023.

[Parker *et al.*, 2023] T. Parker, U. Grandi, and E. Lorini. Anticipating responsibility in multiagent planning. In *ECAI*, pages 1859–1866, 2023.

[Pnueli, 1977] Amir Pnueli. The temporal logic of programs. In *FOCS*, pages 46–57, 1977.

[Shi, 2024] Q. Shi. Responsibility in extensive form games. In *AAAI*, pages 19920–19928, 2024.

[Talbert, 2023] Matthew Talbert. Moral Responsibility. In Edward N. Zalta and Uri Nodelman, editors, *The Stanford Encyclopedia of Philosophy*. Metaphysics Research Lab, Stanford University, Fall 2023 edition, 2023.

[Vardi and Wolper, 1986] Moshe Y. Vardi and Pierre Wolper. An automata-theoretic approach to automatic program verification. In *LICS*, pages 332–344. IEEE Computer Society, 1986.

[Vincent, 2011] Nicole A. Vincent. *A Structured Taxonomy of Responsibility Concepts*, pages 15–35. Springer Netherlands, Dordrecht, 2011.

[Yazdanpanah *et al.*, 2019] V. Yazdanpanah, M. Dastani, W. Jamroga, N. Alechina, and B. Logan. Strategic responsibility under imperfect information. In *AAMAS 2019*, pages 592–600, 2019.