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

Counterfactual Scenarios for Automated Planning

Nicola Gigante[1] , **Francesco Leofante**[2] , **Andrea Micheli**[3]
[1]Free University of Bozen-Bolzano, Bolzano, Italy
[2]Department of Computing, Imperial College London, UK
[3]Fondazione Bruno Kessler, Trento, Italy
nicola.gigante@unibz.it, f.leofante@imperial.ac.uk, amicheli@fbk.eu

Abstract

Counterfactual Explanations (CEs) are a powerful technique used to explain Machine Learning models by showing how the input to a model should be minimally changed for the model to produce a different output. Similar proposals have been made in the context of Automated Planning, where CEs have been characterised in terms of minimal modifications to an existing plan that would result in the satisfaction of a different goal. While such explanations may help diagnose faults and reason about the characteristics of a plan, they fail to capture higher-level properties of the problem being solved. To address this limitation, we propose a novel explanation paradigm that is based on counterfactual scenarios. In particular, given a planning problem P and an LTL$_f$ formula ψ defining desired properties of a plan, counterfactual scenarios identify minimal modifications to P such that it admits plans that comply with ψ. In this paper, we present two qualitative instantiations of counterfactual scenarios based on an explicit quantification over plans that must satisfy ψ. We then characterise the computational complexity of generating such counterfactual scenarios when different types of changes are allowed on P. We show that producing counterfactual scenarios is often only as expensive as computing a plan for P, thus demonstrating the practical viability of our proposal and ultimately providing a framework to construct practical algorithms in this area.

1 Introduction

The widespread adoption of AI solutions for consequential decision-making tasks has fuelled considerable interest in Explainable AI (XAI). This is particularly true in the area of Machine Learning (ML), where black-box models are often deployed in applications such as credit risk analysis (FICO Community 2019) or bail approval (ProPublica 2016). In these settings, most approaches focus on explaining single-shot decisions (*i.e.*, predictions) produced by ML models but are often unable to explain more sophisticated decision-making tasks involving multiple reasoning steps.

The planning community has long recognised the importance of providing explanations for sequential decision-making tasks (Chakraborti, Sreedharan, and Kambhampati 2020). As a result, a host of explainability approaches have been proposed. Examples include model reconciliation (Chakraborti et al. 2017; Sreedharan, Chakraborti, and Kambhampati 2018), which focuses on resolving potential discrepancies between an AI agent's internal model and a human's mental model, and contrastive explanations (Krarup et al. 2021; Krarup et al. 2024; Hoffmann and Magazzeni 2019), which highlight differences between a plan generated by an AI and a user-suggested alternative, showing why the former was preferred.

Much less attention has been devoted to counterfactual explanations, a popular explanation framework that is favoured in XAI due to its intelligibility and alignment with human reasoning (Byrne 2019). Counterfactuals are actively studied in ML, where they are typically defined in terms of minimally altered inputs for which the ML model gives a different, more desirable output from that of the original input (Karimi et al. 2023). Echoing this idea, recent work in planning proposed to characterise counterfactuals in terms of minimal modifications to an existing plan that would result in the satisfaction of a different goal (Belle 2023).

Counterfactuals defined in this way are well suited to reason about *local* properties of a plan, *i.e.*, diagnose faults in it and potentially identify avenues for repair showing how a plan would need to change for a desired outcome to be attained. However, in many practical applications, users are often interested in understanding how characteristics of a planning problem may affect the quality of plans that can be produced. In such cases, local counterfactual explanations are of little use as they fail to capture properties of the planning problem being solved, leaving many unanswered questions about the fundamental relationships between problem structure and plan properties.

Contributions. In this paper, we fill this gap and propose a novel explanation paradigm based on the concept of *counterfactual scenarios* for automated planning problems. Differently from existing proposals, our explanations rely on counterfactual modifications of the planning problem itself, thus pointing users of planning software to what would need to be changed in the original problem formulation for a given course of action to be observed. More formally, given a planning problem P and an LTL$_f$ (De Giacomo and Vardi 2013) formula ψ defining desired properties of a plan, counterfactual scenarios identify minimal modifications to P such that it admits plans that comply with ψ. We investigate the computational complexity of generating such explanations under counterfactual modifications that can alter the initial state of P, the structure of its actions or its goals.

We focus our analysis on two qualitative variants of counterfactuals scenarios: *existential counterfactual scenarios* ($\exists$), which identify modifications to P such that it admits *at least one plan* satisfying ψ, and *universal counterfactual scenarios* ($\forall$), which modify P in such a way that *all its valid plans* satisfy ψ. We demonstrate the feasibility of our proposal by showing that generating $\exists$ and $\forall$ counterfactual scenarios often has a computational complexity comparable to finding a plan for the original problem, thus providing a foundation for developing practical algorithms in this domain.

Related work. The challenge of explaining AI behaviours in sequential decision-making settings has gained considerable attention (see, *e.g.*, (Baier et al. 2025) for a recent account). As a result, a host of approaches have been proposed both in model-based and model-free settings (see, *e.g.*, Chakraborti, Sreedharan, and Kambhampati (2020) and Milani et al. (2024) for a general overview). In the following, we focus on counterfactual explanations for sequential decision-making tasks, but refer the reader to Section 7 for a broader discussion of related work in the planning literature.

While research on counterfactuals for sequential decision-making is still in its infancy, several proposals have already been put forward. For instance, counterfactuals for automated planning problems have been defined by Belle (2023) as minimal modifications to an existing plan that would result in the satisfaction of a different goal. This definition echoes existing characterisations of counterfactuals in ML (Wachter, Mittelstadt, and Russell 2017) and is closely related to the literature on plan repair (Fox et al. 2006). However, differently from the aforementioned definition, our counterfactuals identify changes *in the planning problem itself* and can thus be used to point to how changes in the structure of a problem may affect plan properties.

Counterfactuals have also been investigated in the context of sequential decision-making problems modelled by Markov Decision Processes (MDPs). For instance, Kobialka et al. (2025) studied the problem of generating counterfactual strategies for MDPs by computing minimal modifications to an original strategy that would lead to reaching desirable states high probability. Similar efforts have been made in the Reinforcement Learning arena, as summarised by Gajcin and Dusparic (2024). While the works above propose notions that vary based on the specific changes that can be effected on a policy, they still frame counterfactuals as minimally altered policies, differently from our counterfactual scenarios. Finally, causality-based counterfactuals for MDPs have been investigated (Tsirtsis, De, and Rodriguez 2021; Kazemi et al. 2024). In this line of work, counterfactual explanations are defined in terms of paths that diverge by at most k actions from a given initial path in the MDP, again marking a key difference from our proposal.

Structure of the paper. In the next section, we motivate and explain the need for counterfactual explanations whose scope extends beyond existing proposals, while Section 3 provides some background concepts on planning and LTL_f. Then, Section 4 presents our novel notion of counterfactual scenarios, and in Section 5 we present three concrete classes of counterfactuals that we are interested in analysing.

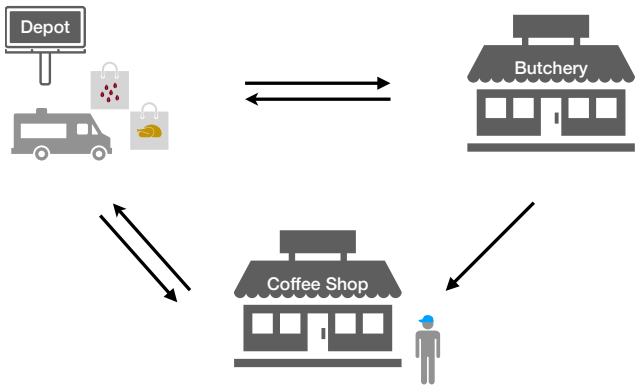

Figure 1: The food delivery domain.

Section 6 proves computational complexity results for such classes. Finally, we discuss the implications of our results and draw our conclusions in Section 7 and Section 8.

2 A Motivating Example

To see what makes counterfactual scenarios interesting and useful, let us consider an example based on the food delivery domain originally formulated by Krarup et al. (2024).

Example 1. *The domain features one driver, one truck and three locations: a depot, a butchery, and a coffee shop. All locations are connected by roads with no pedestrian access, modelled by a predicate* `link(?from,?to)` *(links are directed). Both the goods and the truck are initially located at the depot (*`at(coffee,depot)`, `at(meat,depot)`, `at(truck,depot)`*), while the driver is located at the coffee shop (*`at(driver,coffee shop)`*). The goal is for the driver to deliver meat to the butchery and coffee beans to the coffee shop (*`at(meat,butchery)` $\wedge$ `at(coffee,coffee shop)`*). To achieve this goal, the following actions can be performed:*

- `load(?item,?truck,?loc)`*: loads an item onto the truck, provided both are at the same location. When an item is loaded, it is removed from its location and is marked as stored inside the truck (*$\neg$`at(?item,?truck)` $\wedge$ `in(?item,?truck)`*);*
- `unload(?item,?truck,?loc)`*: removes an item from the truck, making it available at the location where it was unloaded;*
- `drive(?truck,?from,?to,?driver)`*: allows to drive the truck between two specified locations, provided that they are connected by a road,* i.e., `link(?from,?to)` *is true. The driver and the truck must start from the same location for this action to be executable.*

A visual representation of this problem is shown in Figure 1.

Existing approaches to define counterfactual explanations for sequential problems assume the availability of an initial action sequence from which the counterfactual explanation can be obtained. However, there may be cases where the planning problem does not admit a solution and the problem

of generating counterfactuals for such instances becomes ill-defined. For instance, the planning problem described in Example 1 does not admit a solution, as the driver has no way to reach the truck and carry out deliveries. As a consequence, no initial action sequence can be determined to seed the search for a counterfactual explanation. In situations where a solution is unattainable, our *existential counterfactual scenarios* can help identify modifications to the planning problem that would make the problem solvable, as demonstrated in Example 2

Example 2. *The planning problem described in Example 1 does not admit a solution, as the driver is initially located away from the truck with no way to reach it. In such cases, an existential counterfactual scenario highlights how the initial problem should be changed for it to admit a solution. A possible existential counterfactual scenario can therefore be obtained by changing the initial location of the driver from* `at(driver, coffee-shop)` *to* `at(driver, depot)`*. As a result, the new planning problem obtained by implementing this change admits (at least) one plan:*

- `load(meat,truck,depot)`
- `load(coffee,truck,depot)`
- `drive(truck,depot,butchery,driver)`
- `unload(meat,truck,butchery)`
- `drive(truck,butchery,coffee-shop,driver)`
- `unload(coffee,truck,coffee shop)`

The counterfactual scenario in Example 2 helps users of planning software deal with unsolvable planning problems, potentially identifying changes in the problem formulation that determined unsolvability. These kind of explanations can be used to complement existing efforts, *e.g.*, (Eriksson, Röger, and Helmert 2018; Eriksson and Helmert 2020) and improve the user experience, debugging process, or overall effectiveness of planning tools. However, existential counterfactual scenarios also prove useful when planning problems are solvable to begin with, as discussed in Example 3.

Example 3. *Consider the planning problem given by the existential counterfactual scenario in Example 2. Suppose now that the user requires that in some plans, the truck visits the coffee shop before making a delivery to the butchery. This can be captured by the following logical specification:*

$$\psi \coloneqq \Box\big(\mathtt{at(truck, coffee{-}shop)} \rightarrow \bigcirc\, \mathtt{at(truck, butchery)}\big) \\ \wedge \big(\neg \mathtt{at(truck, butchery)}\, \mathcal{U}\, \mathtt{at(truck, coffee{-}shop)}\big)$$

Satisfying this requirement is impossible in the original domain, as driving from the coffee shop to the butchery would require a (directed) connection between the two locations. This fact can be highlighted by means of an existential counterfactual scenario, showing how the preconditions of the action `drive` *should be minimally changed for some valid plans to satisfy ψ. A possible solution to the problem corresponds to a modified planning problem where the precondition of the action is weakened to:*

$$\left\{\begin{array}{l}\mathtt{at(driver, coffee{-}shop)} \wedge \\ \mathtt{at(truck, coffee{-}shop)} \wedge \\ \mathtt{link(coffee{-}shop, butchery)}\end{array}\right\}$$

With this change, the truck is allowed to visit the butchery after the coffee shop in some of the plans admitted by the existential counterfactual scenario.

Example 3 showed how existential counterfactual scenarios can result in changes that bring about *at least one* plan satisfying user requirements. However, depending on the application, users may be interested in changes that constrain the planning problem to have *all* its valid solutions comply with a specification. Our *universal counterfactual scenarios* can achieve this, as demonstrated in Example 4.

Example 4. *Consider the existential counterfactual scenario from Example 2. Suppose now that the user requires that in all plans, the truck should always go back to the depot (eventually) after completing all deliveries. This can be expressed by the following logical specification:*

$$\psi \coloneqq \Box\left[\begin{array}{l}\left(\begin{array}{l}\mathtt{at(meat, butchery)} \wedge \\ \mathtt{at(coffee, coffee{-}shop)}\end{array}\right) \rightarrow \\ \rightarrow \Diamond \mathtt{at(truck, depot)}\end{array}\right]$$

Universal counterfactual scenarios can help identifying changes in the planning problem that would satisfy the requirement above. For instance, strengthening the goal to:

$$\left\{\begin{array}{l}\mathtt{at(meat, butchery)} \wedge \\ \mathtt{at(coffee, coffee{-}shop)} \wedge \\ \mathtt{at(truck, depot)}\end{array}\right\}$$

would result in the the truck eventually visiting the depot in all plans admitted by the modified planning problem.

This section introduced the intuition behind counterfactual scenarios, highlighting some use cases. What follows will formally define our explanation framework and present a detailed analysis of the computational complexity of deciding the existence of such explanations.

3 Preliminaries

For the sake of this paper, we focus on classical planning, adopting a general formulation with arbitrary formulae as preconditions. While the definitions below easily generalize to numeric planning, we restrict ourselves to the classical, finite-state case because numeric planning is in general undecidable thus making all the explainability problems addressed in this paper trivially undecidable as well.

Definition 1 (Planning problem)**.** *A (ground)* ***planning problem*** *is a tuple $P = \langle F, A, I, G\rangle$ where:*

1. *F is a set of Boolean fluents;*
2. *A is a set of actions, where each $a \in A$ comes with:*
   (a) *a precondition pre_a expressed as a formula over F; and*
   (b) *a set of effects eff_a of the form $f := v$ where $f \in F$ and $v \in \{\top, \bot\}$;*[1]
3. *$I : F \rightarrow \{\top, \bot\}$ is the initial state encoded as a total assignment of truth values to fluents in F;*
4. *G is the goal condition expressed as a formula over F.*

We indicate with $\mathcal{P}$ the set of all possible planning problems.

[1] For the sake of simplicity, we assume that in every action there is at most one effect for a fluent f.

Definition 2 (Plan state)**.** *A **state** for a planning problem $P = \langle F, A, I, G\rangle$ is a total assignment $s : F \to \{\top, \bot\}$ of values to fluents.*

Definition 3 (Applicable action)**.** *Given a planning problem $P = \langle F, A, I, G\rangle$, an action $a \in A$ is **applicable** in state s if the formula pre_a is satisfied by the assignment of s.*

Definition 4 (Successor state)**.** *Given a planning problem $P = \langle F, A, I, G\rangle$ and a state s, the **successor** $a(s)$ **of an applicable action** $a \in A$ is a state such that:*

$$a(s)(f) \coloneqq \begin{cases} v & \text{if } f := v \in \mathit{eff}_a \\ s(f) & \text{otherwise} \end{cases}$$

Definition 5 (Sequential plan)**.** *A **sequential plan** for a planning problem $P = \langle F, A, I, G\rangle$ is a sequence of actions $\pi = \langle a_0, \ldots, a_{n-1}\rangle$ where $a_i \in A$. When denoting $s_0 = I$ and $s_{i+1} = a_i(s_i)$, we say π is **valid** if every a_i is applicable in s_{i-1} and $s_n \models G$.*

Given a planning problem $P = \langle F, A, I, G\rangle$, we write Π_P for the set of all valid plans. Note that there is no reason for a plan in *classical* planning for visiting twice the same state, as any such plan can be cut into a shorter one that does not. For this reason, we assume *w.l.o.g.* all plans in Π_P to be devoid of state loops.

Our counterfactual scenarios are based on the notion of satisfaction of an LTL$_f$ formula (De Giacomo and Vardi 2013). LTL$_f$ is a propositional modal logic interpreted over *finite words*. We consider here the standard syntax of LTL$_f$ where a formula ψ is defined by the following grammar:

$$\psi \coloneqq p \mid \neg\psi \mid \psi \vee \psi \mid \psi \wedge \psi \mid \bigcirc \psi \mid \widetilde{\bigcirc}\psi \mid \psi \,\mathcal{U}\, \psi \mid \Diamond \psi \mid \Box \psi$$

where $p \in F$ with F being some finite set of *propositions*.

The semantics is defined as usual in the literature (we omit the formal definition because of space concerns), where: $\bigcirc\psi$ and $\widetilde{\bigcirc}\psi$ are the *strong* and *weak tomorrow* operators which state the truth of ψ at the next state (only if it exists, in the case of the weak operator); $\psi_1 \,\mathcal{U}\, \psi_2$ is the *until* operator stating that ψ_2 will happen in the future and ψ_1 holds everywhere until then; $\Diamond\psi$ and $\Box\psi$ are the *eventually* and *globally* operators, definable respectively as $\Diamond\psi \equiv \top \,\mathcal{U}\, \psi$ and $\Box\psi \equiv \neg\Diamond\neg\psi$, which mandates ψ to hold somewhere in the future or always in the future.

We recall that the *satisfiability* and *validity* problems for LTL$_f$, *i.e.*, deciding respectively whether there exists a word satisfying a given formula or whether all words satisfy a given formula, are both PSPACE-complete (De Giacomo and Vardi 2013), which also happens to be the same computational complexity of the plan existence problem in classical planning (Bylander 1994).

With a slight abuse of notation, we say that a plan π satisfies an LTL$_f$ formula ψ if $\langle s_0, s_1, \ldots, s_n\rangle \models \psi$, where the s_i are defined as in Definition 5.

4 Abstract Counterfactual Scenarios

In this section, we introduce a general and abstract formulation of our counterfactual scenarios which can be instantiated into several different kind of explanations depending on the case at hand. In Section 5 we instantiate the framework considering some interesting concrete classes of counterfactuals which then will be studied in Section 6 from a computational complexity perspective.

In the following, let $\mathbb{P}$ denote the class of all the possible *planning problems* as in Definition 1.

Definition 6 (Possible changes)**.** *A **possible-change relation** is a relation $C \subseteq \mathbb{P} \times \mathbb{P}$.*

Intuitively, a possible-change relation describes abstractly which classes of *changes* are admitted. Then, a *counterfactual scenario* can be defined as a planning problem that can be obtained by a minimal sequence of admitted changes. This notion is formally defined by introducing an abstract *transition system* whose worlds are planning problems.

Definition 7 (Counterfactual transition system)**.** *Let P be a classical planning problem, and let C be a possible-change relation. A **counterfactual transition system** for P and C is the transition system $\mathcal{C}(P, C) \coloneqq \langle \mathbb{P}, P, C\rangle$ where:*

1. *$\mathbb{P}$ is the set of worlds of the transition system;*
2. *P is the initial world of the system.*
3. *C is the accessibility relation of the system;*

A path γ in $\mathcal{C}(P, C)$ is a sequence of problems $\langle P_0, P_1, \ldots, P_n\rangle$ where $P_0 = P$ and $(P_i, P_{i+1}) \in C$ for each $i \in [0, n-1]$. The *cost* of a path is its length $|\gamma|$.

Given a planning problem P' and a possible-change relation C, we denote as $\Gamma_C(P')$ all the paths ending in P' in $\mathcal{C}(P, C)$. We denote as $\Delta_C^P(P') = \min_{\gamma \in \Gamma_C(P')} |\gamma|$ the minimum cost of reaching P'.

Definition 8. *Given a planning problem $P = \langle F, A, I, G\rangle$, an LTL$_f$ formula ψ defined over the alphabet F and a possible-change relation C, an **existential counterfactual scenario** is a planning problem $\Xi_\exists(P, \psi, C)$ such that:*

$$\Xi_\exists(P, \psi, C) \coloneqq \operatorname*{arg\,min}_{\substack{P' \in \mathbb{P} \text{ s.t.} \\ \exists \pi' \in \Pi_{P'} \text{ s.t.} \\ \pi' \models \psi}} \Delta_C^P(P')$$

*Similarly, a **universal counterfactual scenario** is a planning problem $\Xi_\forall(P, \psi, C)$ such that:*

$$\Xi_\forall(P, \psi, C) \coloneqq \operatorname*{arg\,min}_{\substack{P' \in \mathbb{P} \text{ s.t.} \\ \Pi_{P'} \neq \emptyset \text{ and} \\ \forall \pi' \in \Pi_{P'}\, \pi' \models \psi}} \Delta_C^P(P')$$

Note that existential and universal counterfactuals might not exist if there is no problem P' reachable in $\mathcal{C}(P, C)$ admitting a valid plan that satisfies ψ.

Moreover, our definitions do not rule out the possibility that a counterfactual scenario for a problem P is P itself. However, this behaviour is unlikely to happen in practical settings and depends on the type of counterfactual scenario being sought. Typically, users of planning tools are interested in explaining a plan π *after* it has been generated by a planner. When searching for an existential counterfactual scenario, it is possible for $\Xi_\exists(P, \psi, C)$ to be equal to P, because the planner might have picked a plan not satisfying ψ non-deterministically from the pool of existing plans. However, if the user is interested in universal counterfactual scenarios, then $\Xi_\forall(P, \psi, C)$ for P cannot be P itself since the existence of π would violate Definition 8.

5 Concrete Classes of Counterfactuals

The definitions of existential and universal counterfactual scenarios given in the previous section are intentionally quite abstract and general, and in applications they need to be instantiated over concrete classes of possible-change relations. In this section, we describe three possible such instantiations, which define as many different classes of counterfactual scenarios that are interesting in practice. Then, Section 6 will study the associated decision problems from a complexity-theoretic standpoint. In particular, we focus on counterfactuals where we allow changes in the *initial state*, the *goal conditions* or the *action preconditions*. For the sake of this paper, we exclude the possibility of modifying the effects of actions for two reasons. First, for counterfactuals to be useful in practice, they need to be *plausible*: they should represent scenarios that could realistically occur or be implemented, rather than propose unachievable changes to an action's effects (*e.g.*, proposing the complete removal of all resource consumption following the execution of an action, which might be practically impossible). Second, from a technical standpoint, one needs to be very careful when choosing which effects are allowed to be changed to avoid constructing trivial counterfactuals (*e.g.*, achieving the goal from the first action by adding a goal state itself goal as an effect). Nonetheless, the framework introduced in the previous section can easily accommodate the change of effects and one can describe the plausibility constraints in the possible-change relation.

We start from the simplest instance of our framework, where only changes to the *initial state* are allowed. To formalise this requirement, we define a possible-change relation that admits problems that differ for the initial state only. Let F be a set of fluents, then:

$$C_{init} \coloneqq \left\{ (P, P') \,\middle|\, \begin{array}{c} P = \langle F, A, I, G\rangle,\ P' = \langle F, A, I', G\rangle \\ I = I' \text{ except at most one fluent} \end{array} \right\}$$

Note that single edits can change the assignment of at most one single fluent in the initial state, so the distance between two different initial states is the minimum number of edits needed to change one into the other.

A slightly more involved definition arises when one wants to admit changes to the *goal conditions*. In contrast to initial states, goals are expressed as arbitrary Boolean formulas, therefore a slight change may be more semantically relevant than it appears. To capture this nuance, when comparing two different goal conditions we count the number of *differing truth assignments* of the conditions, and we only admit to add or remove one truth assignment at a time, obtaining the following definition:

$$C_{goal} \coloneqq \left\{ (P, P') \,\middle|\, \begin{array}{c} P = \langle F, A, I, G\rangle,\ P' = \langle F, A, I, G'\rangle \\ \#\big[(G \wedge \neg G') \vee (G' \wedge \neg G)\big] = 1 \end{array} \right\}$$

where $\#[\phi]$ is the number of *truth assignments* of ϕ.

Note that the definition of C_{goal} considers both changes that *strengthen* the goal condition, *i.e.*, that remove goal states, and those that *weaken* the condition, *i.e.*, that add goal states. However, note that, for *existential* counterfactuals, *strengthening* the goal is never useful, because if a plan satisfying ψ does not exist, *removing* goal states will not help. Conversely, *weakening* the goal is always sufficient, because adding a state reachable by a valid plan always costs less than adding it and removing something else.

For *universal* counterfactuals, instead, *weakening* is never useful, because if a plan *not* satisfying ψ exists, it cannot disappear by *adding* goal states. However, note that strengthening the goal is not the only option in the universal case, because a counterfactual may be found by considering an entirely disjoint set of goal states.

A class of counterfactual scenarios that is apparently similar to the latter is that obtained by admitting changes to *actions' preconditions*. In this case we define a possible-change relation that admits only problems where the only change is the precondition of a single action where a single truth assignment has been added or removed.

$$C_{act} \coloneqq \left\{ (P, P') \,\middle|\, \begin{array}{c} P = \langle F, A, I, G\rangle,\ P' = \langle F, A', I, G\rangle \\ A \setminus A' = A' \setminus A = \{a\} \\ \text{eff}_a = \text{eff}_{a'} \\ \#\left[\begin{array}{c} \vee\ (\text{pre}_a \wedge \neg\text{pre}_{a'}) \\ (\text{pre}_{a'} \wedge \neg\text{pre}_a) \end{array}\right] = 1 \end{array} \right\}$$

Note that for this class of counterfactuals we can observe, similarly to the previous case, that in the *existential* case only *weakening* the preconditions make sense. Although the definitions of C_{goal} and C_{act} present many similarities, shortly we will see in Section 6 that they are computationally quite different.

Defining plausible scenarios. In concrete applications, not all possible changes in either the initial state, the goal or the action preconditions might be acceptable. For instance, in Example 1, `at(truck, depot)` and `at(truck, butchery)` cannot be both initially true, otherwise the model would allow the truck to be in two locations at the same time. Hence, one needs to impose additional *plausibility constraints* to control which changes are allowed to the planning problem. This can be easily achieved by leveraging the expressive power of LTL_f. In the following, let $\phi_{init}, \phi_{act}, \phi_{goal}$ denote three propositional formulae defining plausibility constraints for the initial state, action preconditions and goal conditions respectively. To limit the changes in C_{init}, we simply conjoin the plausibility constraints (without temporal operators) in ϕ_{init} with ψ; in this way, we obviously constrain the possible initial states, because of the semantics of LTL_f. For C_{goal}, we conjoin the formula $\Diamond(\circledcirc\bot \wedge \phi_{goal})$ into ψ. Essentially, we are forcing ϕ_{goal} to hold on all the final states of a plan (where $\circledcirc\bot$ holds). Finally, if we want to impose a propositional plausibility constraint on the precondition of an action a, we assume *w.l.o.g.* that there exists a fluent f_a that is set to true by the effects of action a and falsified by the effects of all the other actions. Then, we add the formula $\Box(\bigcirc f_a \rightarrow \phi_{act})$, forcing the formula ϕ_{act} to hold in the state where action a

	Initial State C_{init}	**Goals** C_{goal}	**Action Preconditions** C_{act}
$\Xi_\exists$	PSPACE-c. Theorem 2	PSPACE-c. Theorem 3	PSPACE-c. Theorem 5
$\Xi_\forall$	PSPACE-c. Theorem 2	PSPACE-c. Theorem 4	$\in$ NEXP$^{\text{NP}}$ Theorem 6

Table 1: Overview of the computational complexity results.

has been applied. In all three cases above, imposing plausibility constraints can be done by simply extending the formula ψ without complicating the problem further. As such, we do not need to explicitly consider plausibility constraints in the theoretical analysis presented in the next section.

6 The Hardness of Finding Counterfactuals

In this section we study the *computational complexity* of finding counterfactuals of the kinds introduced in Section 5. We will see that the decision problem for most of them surprisingly remains PSPACE-complete, which is the same complexity as finding a plan in the first place. A complete picture of the results is available in Table 1.

In our framework, the quality of plans is evaluated in terms of the satisfaction of an LTL$_f$ formula ψ over the fluents F. Therefore, we need a way to link the two worlds of planning and LTL$_f$. To this end, we can leverage a well-known result stating that any classical planning problem can be translated into a suitable LTL$_f$ formula that represents the plans of the problem and the states visited by those plans.

Proposition 1 (Cialdea Mayer et al. (2007))**.** *For any classical planning problem $P = \langle F, A, I, G\rangle$ there exist:*

1. *an LTL$_f$ formula $[P]$ over the alphabet $A \cup F$ such that from any model of $[P]$ one can extract in polynomial time a plan for P (not necessarily reaching the goal);*
2. *an LTL$_f$ formula $[P]_G$ over the alphabet $A \cup F$ such that from any model of $[P]_G$ one can extract in polynomial time a solution plan for P (*i.e.*, reaching the goal);*

The extraction in polynomial time is actually a simple projection of the propositions from A in the model of the LTL$_f$ formula: if proposition a is true at a given time point it means action a is executed in that step. Similarly, the propositions from F encode the states visited by the plan.

Since the counterfactual scenarios are *minimal* objects, minimizing the number of edits, finding one is an *optimization* problem. Therefore, as is customary in complexity theory, our analysis is concerned with the *decision problem* associated with such an optimization problem.

Definition 9 (Counterfactual scenario existence problem)**.** *Let $C \subseteq \mathbb{P}\times\mathbb{P}$ be a possible-change relation. The **existential counterfactual scenario existence problem** under C, denoted $\text{CSEP}_\exists(C)$, is the problem of deciding, given a planning problem P, an LTL$_f$ formula ψ, and a budget $K \geq 0$, whether $\Xi_\exists(P, \psi, C)$ exists and $\Delta_C^P(\Xi_\exists(P, \psi, C)) \leq K$.*

*The **universal** counterfactual scenario existence problem under C, denoted $\text{CSEP}_\forall(C)$, is defined similarly over $\Xi_\forall(P, \psi, C)$.*

Our first result is that counterfactual scenario existence problems are, in general, PSPACE-hard.

Theorem 1. *Let F be a set of fluents and $C \subseteq \mathbb{P} \times \mathbb{P}$ be a possible-change relation over F. Then, both $\text{CSEP}_\exists(C)$ and $\text{CSEP}_\forall(C)$ are **PSPACE-hard**.*

Proof. We go by reduction from the plan existence problem of classical planning which is known to be PSPACE-complete (Bylander 1994). Given $P = \langle F, A, I, G\rangle$, it is sufficient to ask for an existential or universal counterfactual scenario for a trivial formula $\psi := \top$ and a cost $K = 0$. The formula is trivially always satisfied, and the zero cost ensure to be unable to make any change to P. Then, since both the existential and the universal definitions require the counterfactual to have at least a solution, the counterfactual exists if and only if P has a solution in the first place. □

Following Theorem 1, we can now focus specifically on showing that the decision problems for most of our particular classes of counterfactual scenarios *belong* to PSPACE and are therefore PSPACE-complete.

As in Section 5, we start from the class of counterfactuals where only changes to the *initial state* are admitted.

Theorem 2. *Both $\text{CSEP}_\exists(C_{init})$ and $\text{CSEP}_\forall(C_{init})$ are **PSPACE-complete**.*

Proof. Let $P = \langle F, A, I, G\rangle$, ψ an LTL$_f$ formula, and let $K \geq 0$ be the cost. We proceed as follows. If $K = 0$, we define $I' = I$. Otherwise, we loop through all the possible alternative initial states I' which differs from I of at most K fluents. In both cases, for all the considered I', we test the satisfiability of $\psi \wedge [P]_G$, in the case of $\text{CSEP}_\exists(C_{init})$, or the validity of $[P]_G \rightarrow \psi$ in the case of $\text{CSEP}_\forall(C_{init})$ (see also Proposition 1). If we find any I' for which the test is successful, we found an existential (or universal) counterfactual, otherwise we reply negatively. Since satisfiability and validity for LTL$_f$ is PSPACE-complete, and we only need an additional polynomial number of bits to count all the possible exponential initial states I', the above procedure uses polynomial space, so $\text{CSEP}_\exists(C_{init})$ and $\text{CSEP}_\forall(C_{init})$ are PSPACE-complete. □

Let us now consider counterfactuals where the *goal condition* can be edited. Here, the existential and universal counterfactuals need to be addressed separately.

Theorem 3. $\text{CSEP}_\exists(C_{goal})$ *is **PSPACE-complete**.*

Proof. In this class of counterfactuals, changing the goal can be done by adding or removing truth assignments of the goal condition. Recall that, as observed in Section 5, for the existential counterfactuals, only *weakening* the goal makes sense, *i.e.*, *adding* truth assignments.

Let $\theta := \psi \wedge [P]$ (see Proposition 1), and we test the satisfiability of θ (which can be done in polynomial space, as we recall). Then:

Algorithm 1 Algorithm for $\mathrm{CSEP}_\forall(C_{goal})$

Require: $P = \langle F, A, I, G\rangle$, ψ LTL_f, $K \geq 0$
1: $\mathrm{counter} \leftarrow 0$
2: **for** every truth assignment $\nu \models G$ **do**
3: **if** $[P]_\nu \wedge \neg\psi$ is satisfiable **then**
4: $\mathrm{counter} \leftarrow \mathrm{counter} + 1$
5: **end if**
6: **end for**
7: **if** $\mathrm{counter} > K$ **then**
8: **return** *false*
9: **else if** $\mathrm{counter} = K$ **then**
10: **return** whether $[P]_G \wedge \psi$ is satisfiable
11: **else**
12: **return** whether $[P] \wedge \psi$ is satisfiable
13: **end if**

1. if $K = 0$, and θ is satisfiable, P is an existential counterfactual for itself of cost zero, so we reply positively; otherwise, we reply negatively;
2. if $K > 0$, and θ is satisfiable, then we extract the last state s reached by the corresponding model of θ, and we define $P' = \langle F, A, I, G'\rangle$ where $G' := G \vee s$; then we know a plan for P' satisfying ψ exists; if θ is unsatisfiable, we reply negatively.

All of the above can be done in polynomial space, therefore $\mathrm{CSEP}_\exists(C_{goal})$ is PSPACE-complete. □

The case of $\mathrm{CSEP}_\forall(C_{goal})$ is a bit more involved. Our solution is shown in Algorithm 1. Intuitively, the algorithm counts how many goal states are reachable by plans that do *not* satisfy ψ (lines 2 to 6). This amount of goal states have to be removed from the goal in any case, so we do not actually *store* them, we only count how many they are. Then, if the amount is greater than the maximum allowed cost, we reply negatively (line 7). Otherwise, we have to be careful. The definition of universal counterfactual (Definition 8) forbids trivial solutions where the problem has no plans at all (and hence trivially all their plans would satisfy ψ). So we need to check whether there actually exists at least a plan satisfying ψ. However, if we already reached the maximum allowed cost (line 9), such a plan must reach the original goal condition, because we do not have margin to add anything within the cost, so return positively if and only if a plan of the original problem exists that satisfies ψ (line 10). If instead the amount of assignments to be removed is strictly below the allowed cost (line 11), we have margin to ensure the set of solution plans is not empty, so we reply positively if and only if there is a plan reaching any goal state while satisfying ψ (line 12). Let us now prove formally the soundness of Algorithm 1.

Theorem 4. $\mathrm{CSEP}_\forall(C_{goal})$ *is* ***PSPACE-complete***.

Proof. We want to prove that Algorithm 1 returns *true* if and only if there exists a universal counterfactual scenario $\Xi_\forall(P, \psi, C_{goal})$ of cost less than or equal to K. So suppose that the latter is the case and let $\Xi_\forall(P, \psi, C_{goal}) = \langle F, A, I, G'\rangle$. In this proof, with some abuse of notation, we will denote as G and G' the *set of goal states* that satisfy the G and G' formulas. As one can easily go from one representation to the other, this should not cause any ambiguity.

Let us now define as $S \subseteq G$ the subset of goal states reached by any plan of P that does *not* satisfy ψ. Note that such goal states must in any case be removed from G so the cost of $\Xi_\forall(P, \psi, C_{goal})$ is at least $|S|$. Since the cost of $\Xi_\forall(P, \psi, C_{goal})$ stays within K we know $|S| \leq K$, otherwise we would need to overflow the cost to edit G to G'. Therefore the $\mathrm{counter}$ variable at the end of the *for loop* in Algorithm 1 must be less than or equal to K so the algorithm does *not* reply *false* in Line 8. So now we have two cases:

1. if $|S| = K$, we know $G' = G \setminus S$, because we can only afford to remove S and not to edit G in any other way. Now, since $\Xi_\forall(P, \psi, C_{goal})$ is a universal counterfactual and by definition it has at least one plan satisfying ψ, the check at Line 10 returns *true*.
2. If $|S| < K$, then $G' = G \setminus S \cup S'$ where S' is some set of goal states that G' adds with regards to G. Then, since $\Xi_\forall(P, \psi, C_{goal})$ is a universal counterfactual and by definition it has at least one plan satisfying ψ, the check at Line 12 returns *true*.

Vice versa, suppose Algorithm 1 returns *true*. We build a set G' of goal states that will define the counterfactual $\Xi_\forall(P, \psi, C_{goal}) = \langle F, A, I, G'\rangle$. Let $S \subseteq G$ be the subset of goal states reached by any plan of P that does *not* satisfy ψ. Its size $|S|$ is the value the variable $\mathrm{counter}$ holds at line 6. Since we return *true* we know $|S| \leq K$, so we have two cases:

1. we return *true* at line 10, hence $|S| = K$ and $[P]_G \wedge \psi$ is satisfiable. So we set $G' = G \setminus S$ to build our counterfactual: any plan reaching a goal state in G' must satisfy ψ (because it *does not* satisfy $\neg\psi$), and we know the set of plans is not empty because $[P]_G \wedge \psi$ is satisfiable.
2. We return *true* at line 12, hence $|S| < K$ and $[P] \wedge \psi$ is satisfiable. So there exists a plan of P (not necessarily reaching G, see Proposition 1) that satisfies ψ. Let s be the goal state reached by such a plan. We set $G' = G \setminus S \cup \{s\}$ to build our counterfactual: in this way we ensure the set of plans satisfying ψ is not empty, and we stay below the cost K. □

Let us now focus on the last class of counterfactuals considered in Section 5, namely those where *actions' preconditions* are allowed to change. As we will see, even though the existential counterfactuals are again relatively easy to find (*i.e.*, still in polynomial space), *universal* counterfactuals may be more involved. We start with the easier case, which nevertheless requires some additional background. Given a finite set of propositions F, let $\omega : F \to \mathbb{N}$ be a *weighting function* for F. Then, given an LTL_f formula over F, one may ask for a word of *minimum weight*, where the weight of a word is the sum of the weights of all the propositions true in any time step. Then, the following is known:

Proposition 2 (Dodaro, Fionda, and Greco (2022))**.** *Deciding whether an LTL_f formula has a model of weight at most*

k *is* PSPACE-complete. *A minimal model can be produced in polynomial space as well.*

The above proposition is at the core of the following solution for $\mathrm{CSEP}_{\exists}(C_{act})$.

Theorem 5. $\mathrm{CSEP}_{\exists}(C_{act})$ *is* ***PSPACE-complete****.*

Proof. Let $P = \langle F, A, I, G\rangle$ be a planning problem, ψ an LTL_f formula, and $K \geq 0$. Recall from an observation in Section 5 that it only makes sense to look for counterfactuals that *weaken* the precondition of some actions. We proceed as follows. We define A_{relax} as a set of actions equal to A except that each action has its precondition set to *true*. Then, we let $P_{relax} = \langle F, A \cup A_{relax}, I, G\rangle$, and we define a weighting function ω such that $\omega(a) = 0$ for all $a \in A$ and $\omega(a) = 1$ for all $a \in A_{relax}$. Then, we consider the formula $\theta \coloneqq [P_{relax}]_G \wedge \psi$, and look for a model of *minimal weight* according to ω. Recall from Section 3 that this can be done in polynomial space. Then, we return *true* if and only if θ is satisfiable and the minimal model has weight at most K.

To see that the above procedure is correct, suppose on one hand that θ is indeed satisfiable with a minimal model of weight $L \leq K$. We can extract each occurrence of an action a from A_{relax} in the model, and define as S_a the set of all the states the action a was applied to in the model. We then define our counterfactual $\Xi_{\exists}(P, \psi, C_{act}) = \langle F, A', I, G\rangle$ where A' is defined as A except that the precondition of each $a \in A$ is *disjuncted* with all the states in S_a. Now, note that this is an existential counterfactual because, by construction, there is a plan of $\Xi_{\exists}(P, \psi, C_{act})$ satisfying ψ. It also has cost $L \leq K$, by the way we defined ω.

Vice versa, suppose a counterfactual $\Xi_{\exists}(P, \psi, C_{act}) = \langle F, A', I, G\rangle$ exists with cost $L \leq K$. Then, let S_a be the set of states added to pre_a in A'. Let π be the plan that witnesses the counterfactual, *i.e.*, a plan that satisfies ψ. We can assume *w.l.o.g.* that π visits all the states in S_a at least once, otherwise there would be a counterfactual of lower cost built by *not adding* the non-visited state to pre_a. Then, we can construct a model of θ of minimum weight by setting proposition a_{relax} or a to true respectively each time an action a is applied in π to a state $s \in S_a$ or $s \notin S_a$. Since each usage of an a_{relax} has cost 1, the cost of the model is $L \leq K$. □

The complexity of $\mathrm{CSEP}_{\forall}(C_{act})$ concludes this section. However, this case is more involved. To proceed, let us recall a standard ingredient of complexity theory. *Oracle machines* are a standard tool (see *e.g.*, Arora and Barak (2009)) for the study of the relationship between complexity classes. An Oracle machine is a Turing machine with an associated *oracle language* O. At any time, the machine can query whether $s \in O$ for some string s and get the answer in *constant time*. When we study the computational complexity of a problem, by counting the oracle computation as a single computation step we isolate the main algorithm solving the problem from its sub-problems, clarifying the relationship between them.

What we can show here is that $\mathrm{CSEP}_{\forall}(C_{act})$ belongs to the class $\mathrm{NEXP}^{\mathrm{NP}}$, which is the class of problems solvable in *nondeterministic exponential time* when given access to an oracle for an NP problem. This class is also often denoted as Σ_2^{EXP}, referring to its second place in the *exponential hierarchy*. Note that $\mathrm{NEXP} \subseteq \mathrm{NEXP}^{\mathrm{NP}} \subseteq \mathrm{EXPSPACE}$ but it is not known whether $\mathrm{NEXP} = \mathrm{NEXP}^{\mathrm{NP}}$ unless $\mathrm{P} = \mathrm{NP}$.

In our case, the NP problem to be used as oracle is SAT, *i.e.*, *propositional satisfiability*, the prototypical NP-complete problem. We use a call to a SAT oracle to test *bounded* LTL_f *satisfiability*, *i.e.*, satisfiability of an LTL_f formula restricted to models of bounded length. Many approaches exist in literature to solve LTL and LTL_f satisfiability (or equivalently model checking) through an iteration of SAT queries (Clarke et al. 2001; Geatti et al. 2024). In particular, we can state the following.

Proposition 3 (Geatti et al. (2024))**.** *Given an* LTL_f *formula* ψ *and a bound* $B > 0$*, in time polynomial in* B *and in the size of* ψ *one can produce a* propositional *formula* $\mathrm{enc}(\psi, B)$ *such that* $\mathrm{enc}(\psi, B)$ *is* satisfiable *if and only if* ψ *has a model of length at most* B*.*

With Proposition 3 in place we can proceed.

Theorem 6. $\mathrm{CSEP}_{\forall}(C_{act})$ *belongs to* ***NEXP$^{\mathrm{NP}}$****.*

Proof. The solution is shown in Algorithm 2. Following the definition of universal counterfactual, we nondeterministically guess a number of actions to be edited and the states to be added or removed and we build a candidate counterfactual P'. Then we need to check first if the problem has any solution plan at all, and then if all its solution plans satisfy ψ. For both checks we invoke the NP oracle, whose calls are underlined in Algorithm 2:

1. the first check tests the satisfiability of $\mathrm{enc}([P']_G, 2^{|F|})$, which by Proposition 3 corresponds to testing the satisfiability of $[P']_G$ over models of length at most $2^{|F|}$. We know this bound is sufficient because solution plans of classical planning problems do not have any reason to visit the same state twice, and after $2^{|F|}$ steps we are sure to find a repetition.
2. the second check tests the satisfiability of the formula $\mathrm{enc}([P']_G \wedge \neg\psi, 2^{|F|})$ which by Proposition 3 corresponds to the satisfiability of $[P']_G \wedge \neg\psi$ over models of length at most $2^{|F|}$, a bound that is chosen for the same reason outlined above. Since we accept when $[P']_G \wedge \neg\psi$ is *not* satisfiable this corresponds to checking the validity of $[P']_G \rightarrow \psi$, which corresponds to the fact that all solution plans of P' satisfy ψ. □

Theorem 6 shows that $\mathrm{CSEP}_{\forall}(C_{act})$ has a $\mathrm{NEXP}^{\mathrm{NP}}$ upper bound, but a matching lower bound is still missing and not trivial to prove. Theorem 1 shows that the class of $\mathrm{CSEP}_{\exists}(C)$ and $\mathrm{CSEP}_{\forall}(C)$ problems have a PSPACE *core* that cannot be avoided, made of the interconnection of two PSPACE-complete problems, namely classical planning and LTL_f satisfiability. However, the possible-change relation C can be arbitrarily complex. For the practically relevant classes of counterfactuals that we chose to study, the complexity has turned out to be relatively low, but theoretically one may devise classes of possible-change relations that easily cause complexity to increase arbitrarily, as evidence of the formal flexibility of our framework.

Algorithm 2 Algorithm for $\mathrm{CSEP}_\forall(C_{act})$

Require: $P = \langle F, A, I, G\rangle$, ψ LTL_f, $K \geq 0$
for $i \leftarrow 1$ to K **do**
  guess action a **and** state s
  guess $add \in \{true, false\}$
  if *add* **then**
    $\mathrm{pre}'_a \leftarrow \mathrm{pre}_a \vee s$
  else
    $\mathrm{pre}'_a \leftarrow \mathrm{pre}_a \wedge \neg s$
  end if
end for
if $\underline{\mathrm{enc}([P']_G, 2^{|F|}) \text{ is satisfiable and}}$
  $\underline{\mathrm{enc}([P']_G \wedge \neg\psi, 2^{|F|}) \text{ is } \textit{not} \text{ satisfiable}}$
then
  accept
else
  reject
end if

7 Discussion

This paper introduced counterfactual scenarios, a novel explanation paradigm for automated planning problems. Differently from existing proposals, our explanations identify minimal modifications to a planning problem such that it admits plans that comply with user-specified requirements, here captured by means of LTL_f formulae. Given that our proposal is the first of its kind, in this paper we set to investigate the computational complexity of generating this type of explanations and explored potential theoretical barriers to our proposal. Our analysis revealed that computing counterfactual scenarios is, in most cases, only as expensive as solving planning problems. This somewhat surprising result suggests that, for a significant class of problems, the additional step of identifying changes to a planning problem necessary to achieve a desired outcome does not introduce a prohibitive increase in complexity.

Our counterfactual scenarios are strongly related to the literature on model repair. For instance, work by Lin and Bercher addresses a setting where a flawed domain is repaired to fit a plan that serves as a witness (Lin and Bercher 2021; Lin and Bercher 2023), often accounting for a notion of minimality of change (Lin et al. 2025). Although this line of work bears some similarities to our proposal, here we considered the more general problem of generating modifications that comply with an arbitrary LTL_f formula, rather than fixing existing domains based on specific plan examples. This is a major departure point, as our counterfactual scenarios can accommodate complex behaviours captured by abstract and highly expressive LTL_f formulae, and are guaranteed to return modified domains where all (or at least one) plans adhere to such behaviours. This profound technical difference is also reflected in the divergent computational complexities we obtained: most of our problems are PSPACE-complete, while the setting considered by Lin and Bercher typically yields NP-complete problems. Another critical difference between our work and model repair is the inherent plausibility requirement. Our counterfactuals are designed to generate actionable and plausible insights, pointing to what would need to be changed in a planning domain for a given behaviour to be observed. As such, changes suggested by our counterfactuals are not necessarily aimed at fixing a flawed domain; rather, they must be plausible (according to a domain-dependent notion of plausibility). This crucial requirement motivated our initial choice of using LTL_f formulae to capture plausibility and marks another fundamental difference from work on model repair.

Our work fits within the broader scope of explainable automated planning and introduces a novel explanation paradigm that takes a fundamentally different approach than both model reconciliation and contrastive explanations. Unlike model reconciliation, our counterfactual scenarios do not operate on the premise of reconciling a human's mental model with an agent's. Our approach focuses on identifying systematic modifications to a planning problem that would enable or enforce specific properties on the resulting plans, as captured by an LTL_f formula. As such, our counterfactuals are prescriptive and prospective, answering the question "How can I change the planning problem so that plans satisfying a given property become possible?". This marks a crucial difference from contrastive explanations, which have typically been defined to address the question: "Why did the agent choose action A instead of action B in a plan?". Contrastive explanations are inherently local and retrospective, providing a justification for a specific choice made for a given planning problem. Our explanations have a global scope, identifying modifications that affect the entire planning problem space to enable or enforce a class of solutions characterized by an LTL_f formula. This is particularly evident in the universal setting, where there is a sharp contrast with the local and specific comparison inherent in contrastive explanations.

8 Future Work

The findings presented here have profound implications for the practical application of our counterfactuals, suggesting that existing planning algorithms could be leveraged to support the generation of counterfactual scenarios. Our future work will focus on the development of efficient algorithms for the generation of counterfactual scenarios. Our efforts will initially concentrate on the classical planning setting and the use of LTL_f specifications as described in this paper. Given the strong link with model repair, we believe it would be interesting to investigate whether some of the algorithmic techniques developed in that space could be adapted to compute counterfactual scenarios, particularly if plausibility constraints could be integrated. Once a robust algorithmic foundation is established for this setting, it would be interesting to explore the generation of counterfactual scenarios for different planning formalisms, such as those addressing numeric domains, as well as considering alternative specification languages over finite traces, such as CTL for probabilistic systems (Baier and Katoen 2008).

Acknowledgments

Andrea Micheli has been supported by the STEP-RL project funded by the European Research Council under GA n. 101115870. Francesco Leofante was supported by Imperial College under the Imperial College Research Fellowship scheme. Nicola Gigante was partially funded by the NextGenerationEU FAIR PE0000013 project MAIPM (CUP C63C22000770006).

References

Arora, S., and Barak, B. 2009. *Computational Complexity - A Modern Approach*. Cambridge University Press.

Baier, C., and Katoen, J. 2008. *Principles of model checking*. MIT Press.

Baier, H.; Keane, M. T.; Sreedharan, S.; Tulli, S.; Verma, A.; and Vasileiou, S. L. 2025. Explainable AI for Sequential Decision Making (Dagstuhl Seminar 24372). *Dagstuhl Reports* 14(9):67–103.

Belle, V. 2023. Counterfactual explanations as plans. In *The 39th International Conference on Logic Programming. Open Publishing Association*.

Bylander, T. 1994. The computational complexity of propositional STRIPS planning. *Artif. Intell.* 69(1-2):165–204.

Byrne, R. M. J. 2019. Counterfactuals in explainable artificial intelligence (XAI): evidence from human reasoning. In Kraus, S., ed., *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI 2019, Macao, China, August 10-16, 2019*, 6276–6282. ijcai.org.

Chakraborti, T.; Sreedharan, S.; Zhang, Y.; and Kambhampati, S. 2017. Plan explanations as model reconciliation: Moving beyond explanation as soliloquy. In Sierra, C., ed., *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI 2017, Melbourne, Australia, August 19-25, 2017*, 156–163. ijcai.org.

Chakraborti, T.; Sreedharan, S.; and Kambhampati, S. 2020. The emerging landscape of explainable automated planning & decision making. In *Proceedings of IJCAI 2020*, 4803–4811. ijcai.org.

Cialdea Mayer, M.; Limongelli, C.; Orlandini, A.; and Poggioni, V. 2007. Linear temporal logic as an executable semantics for planning languages. *J. Log. Lang. Inf.* 16(1):63–89.

Clarke, E. M.; Biere, A.; Raimi, R.; and Zhu, Y. 2001. Bounded model checking using satisfiability solving. *Formal Methods Syst. Des.* 19(1):7–34.

De Giacomo, G., and Vardi, M. Y. 2013. Linear temporal logic and linear dynamic logic on finite traces. In Rossi, F., ed., *IJCAI 2013, Proceedings of the 23rd International Joint Conference on Artificial Intelligence*, 854–860. IJCAI/AAAI.

Dodaro, C.; Fionda, V.; and Greco, G. 2022. LTL on weighted finite traces: Formal foundations and algorithms. In Raedt, L. D., ed., *Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence, IJCAI 2022, Vienna, Austria, 23-29 July 2022*, 2606–2612. ijcai.org.

Eriksson, S., and Helmert, M. 2020. Certified unsolvability for SAT planning with property directed reachability. In Beck, J. C.; Buffet, O.; Hoffmann, J.; Karpas, E.; and Sohrabi, S., eds., *Proceedings of the Thirtieth International Conference on Automated Planning and Scheduling, Nancy, France, October 26-30, 2020*, 90–100. AAAI Press.

Eriksson, S.; Röger, G.; and Helmert, M. 2018. Inductive certificates of unsolvability for domain-independent planning. In Lang, J., ed., *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI 2018, July 13-19, 2018, Stockholm, Sweden*, 5244–5248. ijcai.org.

FICO Community. 2019. Explainable Machine Learning Challenge. https://community.fico.com/s/explainable-machine-learning-challenge.

Fox, M.; Gerevini, A.; Long, D.; and Serina, I. 2006. Plan stability: Replanning versus plan repair. In Long, D.; Smith, S. F.; Borrajo, D.; and McCluskey, L., eds., *Proceedings of the Sixteenth International Conference on Automated Planning and Scheduling, ICAPS 2006, Cumbria, UK, June 6-10, 2006*, 212–221. AAAI.

Gajcin, J., and Dusparic, I. 2024. Redefining counterfactual explanations for reinforcement learning: Overview, challenges and opportunities. *ACM Comput. Surv.* 56(9):219:1–219:33.

Geatti, L.; Gigante, N.; Montanari, A.; and Venturato, G. 2024. SAT meets tableaux for linear temporal logic satisfiability. *J. Autom. Reason.* 68(2):6.

Hoffmann, J., and Magazzeni, D. 2019. Explainable AI planning (XAIP): overview and the case of contrastive explanation (extended abstract). In *Reasoning Web. Explainable Artificial Intelligence.*, volume 11810 of *Lecture Notes in Computer Science*, 277–282. Springer.

Karimi, A.; Barthe, G.; Schölkopf, B.; and Valera, I. 2023. A survey of algorithmic recourse: Contrastive explanations and consequential recommendations. *ACM Comput. Surv.* 55(5):95:1–95:29.

Kazemi, M.; Lally, J.; Tishchenko, E.; Chockler, H.; and Paoletti, N. 2024. Counterfactual influence in markov decision processes. *CoRR* abs/2402.08514.

Kobialka, P.; Gerlach, L.; Leofante, F.; Ábrahám, E.; Tarifa, S. L. T.; and Johnsen, E. B. 2025. Counterfactual strategies for markov decision processes. *CoRR* abs/2505.09412.

Krarup, B.; Krivic, S.; Magazzeni, D.; Long, D.; Cashmore, M.; and Smith, D. E. 2021. Contrastive explanations of plans through model restrictions. *J. Artif. Intell. Res.* 72:533–612.

Krarup, B.; Coles, A. J.; Long, D.; and Smith, D. E. 2024. Explaining plan quality differences. In *Proceedings of the Thirty-Fourth International Conference on Automated Planning and Scheduling (ICAPS)*, 324–332. AAAI Press.

Lin, S., and Bercher, P. 2021. Change the world - how hard can that be? on the computational complexity of fixing

planning models. In *Proceedings of the IJCAI 2021*, 4152–4159. ijcai.org.

Lin, S., and Bercher, P. 2023. Was fixing this really that hard? on the complexity of correcting HTN domains. In *Proceedings of AAAI 2023*, 12032–12040. AAAI Press.

Lin, S.; Grastien, A.; Shome, R.; and Bercher, P. 2025. Told you that will not work: Optimal corrections to planning domains using counter-example plans. In *Proceedings of AAAI 2025*, 26596–26604. AAAI Press.

Milani, S.; Topin, N.; Veloso, M.; and Fang, F. 2024. Explainable reinforcement learning: A survey and comparative review. *ACM Comput. Surv.* 56(7):168:1–168:36.

ProPublica. 2016. How We Analyzed the COMPAS Recidivism Algorithm . https://www.propublica.org/article/how-we-analyzed-the-compas-recidivism-algorithm.

Sreedharan, S.; Chakraborti, T.; and Kambhampati, S. 2018. Handling model uncertainty and multiplicity in explanations via model reconciliation. In *Proceedings of ICAPS 2018*, 518–526. AAAI Press.

Tsirtsis, S.; De, A.; and Rodriguez, M. 2021. Counterfactual explanations in sequential decision making under uncertainty. In *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*, 30127–30139.

Wachter, S.; Mittelstadt, B.; and Russell, C. 2017. Counterfactual explanations without opening the black box: Automated decisions and the GDPR. *Harv. JL & Tech.* 31:841.