Complexity Results for Fixing Classical Models Using LTL to Express Which Solutions Are (Un)Desired

Huanghua Sheng, Pascal Bercher

School of Computing, The Australian National University, Canberra, Australia
huanghua.sheng@anu.edu.au, pascal.bercher@anu.edu.au

Abstract

Model repair is the task of modifying a given planning model so that it satisfies a set of validity constraints. Prior frameworks encode constraints as finite sets of desired and undesired plans, which cannot capture properties of infinite sets of plans or patterns shared by *all* (non)solutions, thereby limiting their usefulness. We propose a unified formalism that subsumes prior approaches by incorporating validity constraints specified in Linear Temporal Logic over process traces (LTL_p), thereby capturing temporal and structural properties of plans. We further present a comprehensive complexity analysis that begins with a highly restricted fragment of LTL_p and progressively lifts restrictions, tracking how increasing expressivity impacts the complexity of model repair. This yields a complexity landscape ranging from NP through Σ_2^P up to PSPACE. We also highlight connections between our framework and some well-studied problems, including planning with temporally extended goals (TEGs), model repair using desired and undesired plans, the model reconciliation explanation (MRE) problem, and counterfactual scenarios for automated planning.

Introduction

Automated planning is concerned with finding an action trace (also called a plan) that achieves a desired goal from an initial state (Ghallab, Nau, and Traverso 2004). In this paper, we focus on STRIPS planning, a classical framework introduced by Fikes and Nilsson (1971) which has been extensively studied (Bylander 1994; Erol, Nau, and Subrahmanian 1995; Bonet and Geffner 2001; Hoffmann and Nebel 2001; Helmert 2006). Despite its maturity, building an accurate STRIPS planning model is still a challenging task: even small mismatches between the model and the real world may lead to unacceptable failures, such as having valid plans rejected and invalid ones accepted. Recent work on formal representations of planning domains further highlights this difficulty (Grundke, Röger, and Helmert 2024). A natural way to address this challenge is to automatically repair the model so that it becomes consistent with our existing knowledge. This idea has been widely applied in other areas, including ontology engineering (Baader and Wassermann 2024) and formal verification (Chatzieleftheriou et al. 2015).

Model repair formalizes this idea: given a (possibly) flawed model and a set of *constraints* as domain knowledge, the goal is to modify the model so that the resulting model satisfies these constraints (Bercher, Sreedharan, and Vallati 2025). Existing model repair frameworks typically encode constraints as finite sets of positive plans (plans that work in the real world but might not in the model) and negative plans (plans that do not work in the real world but might succeed in the model) (Zhuo, Nguyen, and Kambhampati 2013; Aineto, Jiménez Celorrio, and Onaindia 2019; Lin and Bercher 2021; Lin, Grastien, and Bercher 2023; Bavandpour et al. 2025; Gragera et al. 2025; Lin et al. 2025). There are also approaches that instead repair models of unsolvable planning problems without additional plans, relying only on (possibly specified) preferences over candidate repairs (Aineto, Jiménez Celorrio, and Onaindia 2019; Sreedharan, Chakraborti, and Kambhampati 2021; Gragera et al. 2023; Welt et al. 2025).

Although plan-based constraints incorporate empirical examples into model repair, they can express only inclusion and exclusion over *finite* sets of plans, which limits their expressivity. For example, the requirement "whenever action a occurs, action b eventually follows" cannot be captured by a finite list, because it ranges over infinitely many plans. To provide additional expressivity, we extend existing frameworks to handle plan patterns based on a language-theoretic perspective, viewing sets of plans as formal languages (Höller et al. 2014, 2016; Lin and Bercher 2022). We focus on patterns over action traces because in many practical scenarios, the most reliable information is an executed action log, whereas the corresponding state information may be unavailable under partial observability or simply too cumbersome to specify. This perspective is shared by a complementary line of work that studies qualitative and quantitative properties of plan spaces (Gnad et al. 2025; Kornherr et al. 2025; Speck et al. 2025). A related line of work studies counterfactual scenarios for planning (Gigante, Leofante, and Micheli 2025); that setting considers changes to the initial state, goal description, and action preconditions.

Concretely, we achieve our goal by replacing finite lists with a more expressive encoding capable of representing infinite languages: *Linear Temporal Logic over process traces* (LTL_p) (Fionda and Greco 2016, 2018). Its precursor, Linear Temporal Logic over finite traces (LTL_f) (De Giacomo

and Vardi 2013), was introduced to express temporal properties over finite traces and has been widely used to model or constrain planning behavior (Simon and Röger 2015; Camacho and McIlraith 2019). Recent work by Gigante, Leofante, and Micheli (2025) also uses LTL_f in the context of counterfactual scenarios to specify properties of state traces. As a variant of LTL_f, LTL_p is interpreted over *process* traces, i.e., finite traces where *exactly one* proposition holds at each time point, making it perfectly suitable for reasoning about sequential plans. LTL_p is capable of expressing plan patterns, enabling a concise representation of (possibly infinitely many) plans that share a common pattern.

Moreover, constraints in existing frameworks can only specify which plans should be included or excluded. We lift this restriction by introducing *constraint types* that capture richer set-theoretic relations (e.g., subset and intersection) between (i) the set of plans that solve the planning problem and (ii) the set of plans that satisfy the specified patterns.

After presenting the framework, we develop a comprehensive complexity analysis of model repair across various settings. We begin with a restricted fragment of LTL_p, demonstrating how the lower expressivity of the fragment affects the complexity of model repair, and then relax these restrictions to analyze the resulting shifts in complexity. We also draw connections between our model repair framework and several closely related problems, including planning with temporally extended goals (TEGs) (Baier and McIlraith 2006a,b), model repair using positive and negative plans (Lin and Bercher 2021; Lin et al. 2025), the model reconciliation explanation (MRE) problem (Sreedharan, Chakraborti, and Kambhampati 2021; Sreedharan, Bercher, and Kambhampati 2022), and recent work on counterfactual scenarios for planning (Gigante, Leofante, and Micheli 2025).

Background

In this section, we present the preliminaries used throughout this paper. We begin with formal languages and the STRIPS planning formalism, then introduce LTL_p and two fragments of it. Finally, we review the relevant complexity classes.

Formal Languages

Throughout this paper, we adopt the notation of Hopcroft, Motwani, and Ullman (2001) regarding formal languages and automata. An *alphabet* is a finite and nonempty set of symbols, and a *word* is a finite sequence of symbols from some alphabet. For example, $\Sigma = \{a, b, \dots, z\}$ is the set of all lower-case letters, and $\langle abc \rangle$ is a word from Σ. The empty word, denoted by ε, is the unique word of length 0. We denote by Σ^* the set of all words over an alphabet Σ. If Σ is an alphabet, then $L \subseteq \Sigma^*$ is a *formal language* over Σ.

STRIPS Planning Formalism

The STRIPS planning framework was introduced by Fikes and Nilsson (1971). In this paper, we follow the formalism presented by Lin and Bercher (2022).

Syntax. A *model* M in STRIPS planning is a triple (F, A, δ), where F is a finite set of facts (also called state variables, fluents, or propositions); A is a finite set of action names (we use "action name" and "action" interchangeably when unambiguous); and $\delta \colon A \to 2^F \times 2^F \times 2^F$ is a function that maps each $a \in A$ to its preconditions, add effects, and delete effects, i.e., $\delta(a) = (\mathit{prec}(a), \mathit{add}(a), \mathit{del}(a))$.

Semantics. In STRIPS planning, a planning state $s \in 2^F$ is a subset of facts; when unambiguous, we simply say "state". An action a is applicable in a state s iff $\mathit{prec}(a) \subseteq s$. If a is applicable in s, then applying a in s yields a new state $s' = (s \setminus \mathit{del}(a)) \cup \mathit{add}(a)$, written $\gamma(a, s) = s'$, where γ is the state transition function. Applying an action trace $\pi = \langle a_1 \dots a_n \rangle$ in state s_0 induces a state trace $\overline{s} = \langle s_0 s_1 \dots s_n \rangle$ such that for every i with $1 \leq i \leq n$, a_i is applicable in s_{i-1} and s_i is the consequence of applying a_i in s_{i-1}. By a slight abuse of notation, we write $\gamma(\pi, s_0) = s_n$ to denote that applying π in s_0 yields s_n.

A STRIPS planning problem $\mathcal{P}$ for a model M is a triple (M, s_0, g), where $s_0 \in 2^F$ is the initial state, and $g \subseteq F$ is the goal description. An action trace π is a solution (or plan) to a STRIPS planning problem $\mathcal{P} = (M, s_0, g)$ iff π is applicable in s_0 and $\gamma(\pi, s_0) \supseteq g$, i.e., π is executable and it achieves the goal. We write $\pi \models \mathcal{P}$ iff π solves $\mathcal{P}$.

Language. Given a STRIPS planning problem $\mathcal{P} = (M, s_0, g)$, its language is defined as $L(\mathcal{P}) = \{\pi \mid \pi \models \mathcal{P}\} \setminus \{\varepsilon\}$. In other words, the language of $\mathcal{P}$ is the set of all its non-empty solutions (Höller et al. 2014, 2016; Lin and Bercher 2022). In this paper, we exclude the empty action trace ε to avoid the trivial case where $g \subseteq s_0$ and to align with the LTL_p semantics, which we will discuss later.

LTL_p

LTL_p is a variant of Linear Temporal Logic tailored to reasoning about processes (formalized below; see Fionda and Greco (2016, 2018)); we therefore adopt it to reason about action traces, with some slight adaptations.

Syntax. Let P be a finite set of (atomic) propositions. LTL_p formulae are built from $a \in P$ using Boolean connectives and the temporal operators X (next) and U (until):

$$\varphi ::= \top \mid a \mid \varphi_1 \wedge \varphi_2 \mid \neg\varphi \mid \mathsf{X}\,\varphi \mid \varphi_1 \,\mathsf{U}\, \varphi_2$$

Semantics. LTL_p is interpreted over *process* traces, where exactly one proposition holds at each time point. A process trace is a finite sequence $\pi = \langle a_0 \dots a_{n-1} \rangle$ of length $n \geq 1$ such that, for each $0 \leq i < n$, we have $a_i \in P$. For $0 \leq i < n$, let $\pi_i = \langle a_i \dots a_{n-1} \rangle$ denote the suffix of π starting at time point i. The satisfaction relation $\models$ is defined inductively as follows. For $0 \leq i < n$:

- $\pi_i \models \top$ • $\pi_i \models a$ iff $a = a_i$ • $\pi_i \models \neg\varphi$ iff $\pi_i \not\models \varphi$
- $\pi_i \models \varphi_1 \wedge \varphi_2$ iff $\pi_i \models \varphi_1$ and $\pi_i \models \varphi_2$.
- $\pi_i \models \mathsf{X}\,\varphi$ iff $i < n - 1$ and $\pi_{i+1} \models \varphi$.
- $\pi_i \models \varphi_1 \,\mathsf{U}\, \varphi_2$ iff $\exists\, j$ with $i \leq j < n$ such that $\pi_j \models \varphi_2$ and, $\forall\, k$ with $i \leq k < j$ it holds that $\pi_k \models \varphi_1$.

For convenience, we adopt some standard abbreviations, namely, $\bot$, $\vee$, $\rightarrow$, F (eventually), G (globally), L (lastly), and X^n (n-step next). They are defined as follows:

- $\bot \equiv \neg\top$
- $\varphi_1 \vee \varphi_2 \equiv \neg(\neg\varphi_1 \wedge \neg\varphi_2)$
- $\varphi_1 \to \varphi_2 \equiv \neg\varphi_1 \vee \varphi_2$
- $\mathsf{F}\,\varphi \equiv \top\,\mathsf{U}\,\varphi$
- $\mathsf{G}\,\varphi \equiv \neg(\top\,\mathsf{U}\,\neg\varphi)$
- $\mathsf{L} \equiv \neg\mathsf{X}\top$
- $\mathsf{X}^n\varphi \equiv \underbrace{\mathsf{X}\ldots\mathsf{X}}_{n}\varphi$

Note that in LTL_p, L is true only at the final time point.

Language. Given an LTL_p formula φ, its language is defined as $L(\varphi) = \{\pi \mid \pi \models \varphi\}$, i.e., the set of all process traces that satisfy φ. For example, the LTL_p formula $a \wedge \mathsf{X}a \wedge \mathsf{XL}$ is satisfied only by $\langle aa\rangle$, hence we have $L(a \wedge \mathsf{X}a \wedge \mathsf{XL}) = \{\langle aa\rangle\}$. Since LTL_p is evaluated at the initial time point of a trace, we only consider traces of length at least 1; in particular, $\varepsilon \notin L(\varphi)$ for any LTL_p formula φ.

Note that the language of a STRIPS planning problem is defined over actions as its alphabet, while that of an LTL_p formula is defined over propositions. We thus have the following correspondence shown in Table 1. When unambiguous, we freely identify these notations.

Formal Language	STRIPS Planning	LTL_p
Alphabet	Actions	Propositions
Word	Action Trace	Process Trace

Table 1: Correspondence between concepts of formal language, STRIPS planning problem, and LTL_p.

LTL_p Fragments. In addition to full LTL_p, we consider and define two novel fragments of LTL_p:

1. $\mathrm{LTL}_\mathsf{p}^{\mathsf{list}}$, the fragment that captures the behavior of finite plan sets. A formula φ in this fragment has the form
$$\bigvee_{i=1}^{n}(a_{i,1} \wedge \mathsf{X}a_{i,2} \wedge \cdots \wedge \mathsf{X}^{l_i-1}a_{i,l_i} \wedge \mathsf{X}^{l_i-1}\mathsf{L})$$
where n is the number of disjuncts in φ and l_i is the length of the ith plan (we allow $n = 0$, in which case $\varphi = \bot$ and $L(\varphi) = \emptyset$). This fragment enforces (i) what action occurs at each time point and (ii) the length of each plan. Consequently, every formula φ in this fragment satisfies (i) $|L(\varphi)| = n$ (assuming no duplicate disjuncts in φ) and (ii) $\max_{\pi\in L(\varphi)} |\pi| = \max_{1\le i\le n} l_i$. In other words, both the number of plans in the list and the maximum plan length are bounded linearly in the input size. $\mathrm{LTL}_\mathsf{p}^{\mathsf{list}}$ precisely captures the behavior of finite plan sets and therefore has the same expressive power as the list-based encoding used in previous model repair frameworks (Zhuo, Nguyen, and Kambhampati 2013; Aineto, Jiménez Celorrio, and Onaindia 2019; Lin and Bercher 2021; Lin, Grastien, and Bercher 2023; Lin et al. 2025). For example, a finite plan set $\{\langle caa\rangle, \langle cab\rangle, \langle cba\rangle, \langle cbb\rangle\}$ over actions $\{a, b, c\}$ can be translated into $(c \wedge \mathsf{X}a \wedge \mathsf{XX}a \wedge \mathsf{XXL}) \vee (c \wedge \mathsf{X}a \wedge \mathsf{XX}b \wedge \mathsf{XXL}) \vee (c \wedge \mathsf{X}b \wedge \mathsf{XX}a \wedge \mathsf{XXL}) \vee (c \wedge \mathsf{X}b \wedge \mathsf{XX}b \wedge \mathsf{XXL})$ and *vice versa*, as they define the same language.
2. $\mathrm{LTL}_\mathsf{p}^{\mathsf{len}}$, the fragment that captures the behavior of finite sets of plans subject to a unarily encoded upper bound on plan length (so that all plans have length linear in the input size), optionally with additional specifications. Equivalently, $\mathrm{LTL}_\mathsf{p}^{\mathsf{len}}$ can be viewed as an abstraction of $\mathrm{LTL}_\mathsf{p}^{\mathsf{list}}$ that does not commit to a specific action at each time point. A formula in this fragment has the form
$$\bigvee_{i=1}^{n}(\psi_i \wedge \mathsf{X}^{l_i-1}\mathsf{L})$$
where each ψ_i can be any LTL_p formula, and the meanings of n and l_i remain as before. By definition, every $\mathrm{LTL}_\mathsf{p}^{\mathsf{list}}$ formula is an $\mathrm{LTL}_\mathsf{p}^{\mathsf{len}}$ formula, including $\bot$. Compared to $\mathrm{LTL}_\mathsf{p}^{\mathsf{list}}$, this fragment enforces only a linear length bound for each plan and lifts the restriction on which action occurs at each time point. Consequently, every formula φ in this fragment satisfies (i) $|L(\varphi)| \le \sum_i^n |P|^{l_i}$, where $|P|$ is the number of propositions, and (ii) $\max_{\pi\in L(\varphi)} |\pi| = \max_{1\le i\le n} l_i$. Note that, just like $\mathrm{LTL}_\mathsf{p}^{\mathsf{list}}$, languages defined by $\mathrm{LTL}_\mathsf{p}^{\mathsf{len}}$ formulae remain finite, and each word again has length at most linear in the input size (recall that X^{l_i-1} abbreviates $\underbrace{\mathsf{X}\ldots\mathsf{X}}_{l_i-1}$). However, since words are no longer enumerated explicitly, the cardinality $|L(\varphi)|$ can grow exponentially in the input size, yielding greater flexibility in specifying plan patterns. We recall the example from $\mathrm{LTL}_\mathsf{p}^{\mathsf{list}}$, namely $\{\langle caa\rangle, \langle cab\rangle, \langle cba\rangle, \langle cbb\rangle\}$, which consists exactly of those action traces of length three that start with c and contain no further occurrence of c. In $\mathrm{LTL}_\mathsf{p}^{\mathsf{len}}$, the same language can be succinctly encoded as $c \wedge \mathsf{X}\neg\mathsf{F}c \wedge \mathsf{XXL}$.

Comparison. Table 2 summarizes the upper bounds on the number of plans ($\max|L(\varphi)|$) and on the plan length ($\max|\pi|$) for $\mathrm{LTL}_\mathsf{p}^{\mathsf{list}}$, $\mathrm{LTL}_\mathsf{p}^{\mathsf{len}}$, and (full) LTL_p formulae.

	$\mathrm{LTL}_\mathsf{p}^{\mathsf{list}}$	$\mathrm{LTL}_\mathsf{p}^{\mathsf{len}}$	LTL_p
$\max\|L(\varphi)\|$	n	$\sum_i^n \|P\|^{l_i}$	∞
$\max\|\pi\|$	$\max_{1\le i\le n} l_i$	$\max_{1\le i\le n} l_i$	no finite bound

Table 2: Upper bounds on language size and maximum plan length for LTL_p and its fragments.

Relevant Complexity Classes

Recall that co-NP denotes the class of complements of NP languages, i.e., $L \in$ co-NP iff $\Sigma^* \setminus L \in$ NP. An NP oracle is a black-box machine that decides a fixed NP language in one step; consequently, any NP or co-NP language can be decided by some NP oracle in one step (for the co-NP case, we simply flip the answer). Σ_2^{p} is the class of languages decidable in polynomial time by a nondeterministic Turing machine with access to an NP oracle. We also recall that NP $\subseteq \Sigma_2^{\mathrm{p}} \subseteq$ PSPACE, and it is widely conjectured that these containments are proper.

Model Repair Framework

Model repair is the task of modifying a model to satisfy a given set of constraints (Bercher, Sreedharan, and Vallati

2025). In this work, we focus on repairs to action descriptions and first define *atomic repairs*.

Definition 1. *Let $M = (F, A, \delta)$ be a model. An atomic repair is a triple $r = (a, t, f)$, where $a \in A$ is an action, $t \in \{prec^+, prec^-, add^+, add^-, del^+, del^-\}$ is a repair type, and $f \in F$ is a fact.*

We define the semantics of atomic repairs as follows:

Definition 2. *Applying an atomic repair r to a model M yields a new model M' in which the corresponding action name maps to the modified preconditions and effects; we write this as $M \Rightarrow_r M'$. For any action name $a \in A$ and fact $f \in F$, the semantics of atomic repairs are:*

- *If $r = (a, prec^+, f)$, then the preconditions of a are updated to $prec(a) \cup \{f\}$.*
- *If $r = (a, prec^-, f)$, then the preconditions of a are updated to $prec(a) \setminus \{f\}$.*
- *The semantics of the remaining cases (add and delete effects) are defined analogously.*

Note that repairs may interfere (e.g., by adding and deleting the same fact), so the order of application may alter the resulting model. To avoid such cases, following Lin et al. (2025), we define *valid* repair sets as follows:

Definition 3. *A finite set Δ of atomic repairs is* valid *iff (i) the order of applying the repairs in Δ to a model does not affect the resulting model, i.e., no repair in Δ would undo another, and (ii) the set does not contain redundant repairs, e.g., adding positive preconditions that already exist.*

Applying a valid repair set Δ to a model M amounts to applying all elements of Δ to M in an arbitrary order; this yields a unique repaired model M', which we denote by $M \Rightarrow_\Delta M'$. Note that, for any model $M = (F, A, \delta)$, every valid repair set Δ satisfies $|\Delta| \leq 3|A||F|$, since redundant repairs are disallowed by definition.

Having formalized the repair procedure, we now introduce *constraints* to specify the desired behavior of repaired models.

Definition 4. *A constraint c for a model $M = (F, A, \delta)$ is a 4-tuple $c = (s_0, g, \varphi, t)$, where $s_0 \in 2^F$ is the initial state, $g \subseteq F$ is the goal, φ is an LTL$_\mathsf{p}$ formula, and $t \in \{positive, negative, existential, universal\}$ is a constraint type.*

The constraint types capture the following intuitions about the behavior of the desired model:

- Positive: action traces with the pattern must be plans.
- Negative: action traces with the pattern must not be plans.
- Existential: there exists a plan with the pattern.
- Universal: all plans must have the pattern.

Formally, their semantics are defined as follows:

Definition 5. *Let M be a model and $c = (s_0, g, \varphi, t)$ be a constraint. Let $\mathcal{P} = (M, s_0, g)$ be the STRIPS planning problem induced by M and c. The satisfaction relation between M and c is given by:*

- *If $t = positive$, then $M \models c$ iff $L(\mathcal{P}) \supseteq L(\varphi)$.*
- *If $t = negative$, then $M \models c$ iff $L(\mathcal{P}) \cap L(\varphi) = \emptyset$.*
- *If $t = existential$, then $M \models c$ iff $L(\mathcal{P}) \cap L(\varphi) \neq \emptyset$.*
- *If $t = universal$, then $M \models c$ iff $L(\mathcal{P}) \subseteq L(\varphi)$.*

We furthermore define that for a set of constraints C, $M \models C$ iff $M \models c$ for every constraint $c \in C$.

Existential and universal constraints are also considered by Gigante, Leofante, and Micheli (2025) in the context of counterfactual scenarios, where constraints are specified over state traces induced by plans.

Building on the above, we formally define the unbounded and k-bounded model repair problems as follows:

Definition 6. *An unbounded model repair problem is a pair $\mathcal{MR} = (M, C)$, where M is a model and C is a finite set of constraints. A solution to $\mathcal{MR}$ is a valid repair set Δ such that $M \Rightarrow_\Delta M'$ and $M' \models C$.*

In this paper, we do not study the unbounded case; instead, we focus on the k-bounded case, where the number of repairs is limited by a given repair budget k:

Definition 7. *A k-bounded model repair problem is a pair $\mathcal{MR}_k = (\mathcal{MR}, k)$, where $k \in \mathbb{N}$. A solution to $\mathcal{MR}_k$ is a valid repair set Δ that solves $\mathcal{MR}$ and satisfies $|\Delta| \leq k$. We say that a k-bounded model repair problem $\mathcal{MR}_k$ is solvable iff it has a solution.*

Although k is binarily encoded and may be exponentially large, every valid repair set Δ has size at most $3|A||F|$. Therefore, every solution to an unbounded or k-bounded model repair instance has size polynomial in the input.

In addition to the general setting, which allows arbitrary mixtures of constraint types, we also consider, for each constraint type t, the simpler t-only setting in which all constraints are of type t.

Definition 8. *Let $t \in \{positive, negative, existential, universal\}$ be a constraint type. An unbounded model repair problem $\mathcal{MR} = (M, C)$ is* t-only *iff for every constraint $c \in C$, the constraint type of c is t. A k-bounded model repair problem $\mathcal{MR}_k = (\mathcal{MR}, k)$ is* t-only *iff $\mathcal{MR}$ is t-only.*

To give readers a better intuition, we present a concrete Blocksworld example in the supplementary material (Sheng and Bercher 2026). In the example, we use the following constraint to avoid double pick in any situation, which is a common physical constraint in the real world:

$$c = (s_0, \emptyset, \mathsf{G}(pick \rightarrow \mathsf{X}(\neg pick \,\mathsf{U}\, drop)), universal).$$

Complexity of Model Repair Problems

In this section, we study the computational complexity of the k-bounded model repair problem under various settings. An overview of complexity results is given in Table 3. We present the results in the following order: from the most restricted fragment LTL$_\mathsf{p}^{\mathsf{list}}$, through LTL$_\mathsf{p}^{\mathsf{len}}$, to full LTL$_\mathsf{p}$; within each fragment, we first treat the t-only cases and then the general case. While Gigante, Leofante, and Micheli (2025) also study the complexity of a broad range of model changes, their complexity analysis is restricted to changes to action preconditions, the initial state, or the goal description.

We begin with the most restricted setting, where constraint languages are encoded as LTL$_\mathsf{p}^{\mathsf{list}}$ formulae, which can be translated into explicit finite plan sets in linear time.

Encoding	$\max \lvert L(\varphi)\rvert$	$\max \lvert\pi\rvert$	Type	Complexity	Theorem
$\text{LTL}_\text{p}^\text{list}$	n	$\max_{1\le i\le n} l_i$	positive	NP-c	(Lin and Bercher 2021, Thm. 13)
			negative	NP-c	(Lin et al. 2025, Thm. 2)
			existential	NP-c	Thm. 1
			universal	PSPACE-c	Thm. 2
			*	PSPACE-c	Thm. 3
$\text{LTL}_\text{p}^\text{len}$	$\sum_i^n \lvert P\rvert^{l_i}$	$\max_{1\le i\le n} l_i$	positive	Σ_2^p-c	Thm. 4
			negative	Σ_2^p-c	Thm. 5
			existential	NP-c	Thm. 6
			universal	PSPACE-c	Thm. 2
			*	PSPACE-c	Thm. 7
LTL_p	∞	no finite bound	positive	PSPACE-c	Thm. 8
			negative	PSPACE-c	Thm. 9
			existential	PSPACE-c	Thm. 10
			universal	PSPACE-c	Thm. 2
			*	PSPACE-c	Thm. 11

Table 3: Overview of the investigated k-bounded model repair problems, their settings, complexities, and corresponding theorems. Here, n is the number of disjuncts in φ, l_i is the length of the ith action trace, and $|P|$ is the number of propositions; all are linear in the input size. "*" denotes all combinations of the investigated constraint types (the general case). A complementary view highlighting the impact of the constraint types is provided in the supplementary material (Sheng and Bercher 2026).

Leveraging existing results, we obtain the following corollaries for the positive-only and negative-only cases:

Corollary 1. *The positive-only k-bounded model repair problem using $\text{LTL}_\text{p}^\text{list}$ is* NP*-complete, even when restricted to instances with a single constraint whose language is a singleton (i.e., $|L(\varphi)| = 1$).*

Proof. Lin and Bercher (2021, Thm. 13) show that the k-bounded model repair problem using positive plans (action traces that are supposed to be solutions, i.e., white-list plans) is NP-complete even when the list contains only one plan. In our positive-only case, the constraint requires that every listed action trace is a solution, which coincides with the requirement of positive plans; moreover, $\text{LTL}_\text{p}^\text{list}$ formulae and finite plan sets are trivially inter-translatable. For example, $(a \wedge \mathsf{X}b \wedge \mathsf{XL}) \vee (a \wedge \mathsf{X}b \wedge \mathsf{XX}c \wedge \mathsf{XXL})$ has the same language as the list $\{\langle ab\rangle, \langle abc\rangle\}$. Thus, the two problems are equivalent (via a trivial reduction in both directions), and NP-completeness follows directly for both the unrestricted case and the case with only one constraint whose language is a singleton. □

Corollary 2. *The negative-only k-bounded model repair problem using $\text{LTL}_\text{p}^\text{list}$ is* NP*-complete; this holds even when for every constraint, the language contains only words of length 1 (i.e., $|\pi| = 1$ for all $\pi \in L(\varphi)$).*

Proof. Membership: Guess a valid repair set Δ and apply it to M to obtain the repaired model M'. For each constraint $c = (s_0, g, \varphi, \text{negative})$, since φ is an $\text{LTL}_\text{p}^\text{list}$ formula, we can enumerate all action traces in $L(\varphi)$. For each action trace $\pi \in L(\varphi)$, we simulate π in M' from s_0, which takes polynomial time. By definition, with $\mathcal{P} = (M', s_0, g)$, we have $M' \models c$ iff $L(\mathcal{P}) \cap L(\varphi) = \emptyset$, so c is satisfied iff none of the simulated action traces is a solution to $\mathcal{P}$. Summed over all $c \in C$, the verification runs in polynomial time. Therefore, the problem is in NP. In particular, NP-membership also holds under the restriction that $|\pi| = 1$ for all $\pi \in L(\varphi)$.

Hardness: Lin et al. (2025, Thm. 2) show that the k-bounded model repair problem using negative plans (i.e., action traces that must *not* be solutions) is NP-hard, even when every action trace in the list is of length 1. In our negative-only case, the constraints require that no listed action trace is a solution to $\mathcal{P}$, which coincides with the requirement of negative plans. Note that, in their definition of negative plans (Lin et al. 2025, Def. 4), there is an additional parameter i indicating the first failing step; this mismatch is harmless here: for action traces of length 1, this parameter is vacuous, as such action traces can only fail at the first step. Thus, we reduce the k-bounded model repair problem using negative plans of length 1 to our negative-only case under the same restriction by translating the given lists into $\text{LTL}_\text{p}^\text{list}$ formulae. This reduction yields NP-hardness under the restriction ($|\pi| = 1$), and thus also in the unrestricted case. □

We proceed to the existential-only case:

Theorem 1. *The existential-only k-bounded model repair problem using $\text{LTL}_\text{p}^\text{list}$ is* NP*-complete, even when restricted to instances with a single constraint whose language is a singleton (i.e., $|L(\varphi)| = 1$).*

Proof. Membership: Guess a valid repair set Δ and obtain the repaired model M'. For an existential constraint c with formula φ, recall that c is satisfied iff $L(\mathcal{P}) \cap L(\varphi) \neq \emptyset$ where $\mathcal{P} = (M', s_0, g)$. Since $L(\varphi)$ can be regarded as a finite list of action traces, we can iterate over this list and, for each action trace, check in polynomial time whether it is a solution to $\mathcal{P}$. Thus, whether M' satisfies all constraints can be decided in polynomial time, and the problem is in NP.

Hardness: Reduce from the positive-only case with only one constraint whose language is a singleton, which is

NP-hard (Cor. 1). Observe that, when $L(\varphi) = \{\pi\}$ and $\mathcal{P} = (M', s_0, g)$, we have $L(\mathcal{P}) \supseteq L(\varphi) \iff \{\pi\} \subseteq L(\mathcal{P}) \iff \pi \in L(\mathcal{P}) \iff L(\mathcal{P}) \cap L(\varphi) \neq \emptyset$. Hence, NP-hardness holds already under the restriction $|L(\varphi)| = 1$, and therefore also in the unrestricted case. □

The complexity rises to PSPACE in the universal-only case; moreover, this bound is independent of the choice of LTL_p fragment.

Theorem 2. *The universal-only k-bounded model repair problem is* PSPACE*-complete, regardless of whether it uses $\text{LTL}_\text{p}^\text{list}$, $\text{LTL}_\text{p}^\text{len}$, or LTL_p.*

Proof. Membership: We consider LTL_p first. Guess a valid repair set Δ and obtain the repaired model M'; this repair step is in NP. To decide whether M' satisfies all constraints, it suffices to consider one constraint at a time. Fix a constraint $c = (s_0, g, \varphi, \text{universal})$ and let $\mathcal{P} = (M', s_0, g)$. By definition, $M' \models c$ iff every solution to $\mathcal{P}$ satisfies φ, i.e., $L(\mathcal{P}) \subseteq L(\varphi)$. Equivalently, c is violated iff there exists a *counterexample* plan $\pi \in L(\mathcal{P}) \setminus L(\varphi)$, i.e., a solution to $\mathcal{P}$ that does not satisfy φ. Since PSPACE = co-PSPACE (Savitch 1970), it suffices to check the complementary question: whether such a counterexample exists. To do so in polynomial space, we adapt algorithms for planning with temporally extended goals (TEGs) (Bacchus and Kabanza 1996; De Giacomo and Vardi 1999). Informally, our procedure can be viewed as running STRIPS plan search and LTL_p trace search simultaneously. The key idea is to monitor the run *on the fly* instead of materializing it in full. Suppose such a counterexample π exists. We incrementally generate π, guessing one action at a time and maintaining only (i) the current planning state and (ii) a "progressed" LTL_p formula, as in planning with TEGs. Note that we do not store the entire formula (which could blow up exponentially in the length of π). Instead, following standard LTL model checking techniques, we represent it as a configuration over the closure of φ, thereby tracking only the *remaining* obligations (Clarke et al. 2018, Chapter 4.6). After each action, we update these two components and discard the executed action. When we nondeterministically decide to stop extending π, we check whether (i) the current planning state satisfies g and (ii) the current configuration shows that φ is not satisfied; in that case, the guessed π is a counterexample.

Note that nondeterminism is used throughout the procedure; this does not change the complexity, since PSPACE = NPSPACE (Savitch 1970). Therefore, the LTL_p case is in PSPACE; the same membership follows for $\text{LTL}_\text{p}^\text{list}$ and $\text{LTL}_\text{p}^\text{len}$ since they are fragments of LTL_p.

Hardness: We reduce from deciding whether a STRIPS planning problem has no non-empty plan. Its complement is PSPACE-complete, since plan existence is PSPACE-complete (Bylander 1994), and excluding the empty plan does not change the complexity, as one can enforce this by introducing a dummy fact f and a dummy action whose precondition is the original goal description and whose sole add effect is f, while replacing the original goal description by $\{f\}$. Hence, the problem itself is PSPACE-complete, since PSPACE is closed under complement. Given a STRIPS planning instance $\mathcal{P} = (M, s_0, g)$, we construct a model repair instance: $\mathcal{MR}_k = ((M, \{(s_0, g, \varphi, \text{universal})\}), k)$, where $\varphi = \bot$ and $k = 0$ so that no repair is permitted; hence, the repaired model can only be M itself. Recall that a universal constraint requires that every non-empty solution to $\mathcal{P}$ satisfies φ; with $\varphi = \bot$, this holds iff $\mathcal{P}$ has no non-empty plan. Therefore, $\mathcal{MR}_k$ is solvable iff $\mathcal{P}$ has no non-empty plan. Recall that $\bot$ is an $\text{LTL}_\text{p}^\text{list}$ formula with 0 disjuncts; PSPACE-hardness thus follows for all three cases ($\text{LTL}_\text{p}^\text{list}$, $\text{LTL}_\text{p}^\text{len}$, and LTL_p). □

Note that the universal counterfactual setting of Gigante, Leofante, and Micheli (2025) is closely related to ours, as it also concerns modifications of a planning model subject to a universal temporal constraint. Their Thm. 6 provides an upper bound for that related setting, but the bound is not known to be tight. However, because their underlying formalism and repair operations differ from ours, our PSPACE result does not directly apply and therefore does not imply an improved upper bound for their problem.

We move on to the general case:

Theorem 3. *The (general) k-bounded model repair problem using $\text{LTL}_\text{p}^\text{list}$ is* PSPACE*-complete.*

Proof. Membership: Guess a valid repair set Δ and obtain the repaired model M'. Then, for each constraint $c \in C$, check whether $M' \models c$. By Cors. 1–2 and Thms. 1–2, this verification can be carried out in polynomial space for every constraint type. Processing the constraints one by one and reusing space yields a PSPACE procedure.

Hardness: The universal-only case is a special case of the general problem and is PSPACE-hard; thus the general problem is also PSPACE-hard. □

Before moving on, we recall the two-quantifier fragment of quantified Boolean formulae (QBF). Let $X = \{x_1, \ldots, x_m\}$ and $Y = \{y_1, \ldots, y_n\}$ be disjoint sets of Boolean variables, and let

$$\Psi \;=\; \exists X \, \forall Y \; \psi(X, Y),$$

where ψ is a propositional formula over $X \cup Y$. Semantically, Ψ is true iff there exists an assignment to X such that for every assignment to Y, $\psi(X, Y)$ is true. The problem of deciding whether a given Ψ is true is denoted by QSAT_2.

Recall that a *literal* is a variable or its negation; a *clause* is a disjunction of literals; and a *term* is a conjunction of literals. A propositional formula is in 3-CNF (conjunctive normal form) if it is a conjunction of clauses, each with three literals, and in 3-DNF (disjunctive normal form) if it is a disjunction of terms, each with three literals. A QSAT_2 instance is said to be in 3-DNF if its propositional part ψ is in 3-DNF. We denote this restriction by $\text{QSAT}_2 \cap$ 3-DNF. This problem is Σ_2^p-complete (Stockmeyer 1976) and will serve as a source problem in our hardness proofs.

We now turn to $\text{LTL}_\text{p}^\text{len}$, the fragment that imposes only (unarily encoded) length bounds on action traces. Consequently, for any $\text{LTL}_\text{p}^\text{len}$ formula φ, every word in $L(\varphi)$ has length at most linear in the input size, while $|L(\varphi)|$ may be exponential, making a direct translation to explicit lists (as in $\text{LTL}_\text{p}^\text{list}$) impractical. We begin with the positive-only case:

Theorem 4. *The positive-only k-bounded model repair problem using LTL$_\mathsf{p}^{\mathsf{len}}$ is Σ_2^p-complete.*

Proof. Membership: Guess a valid repair set Δ, and then check whether there exists a counterexample action trace of linear length. In this case, a counterexample action trace π is one that satisfies φ but does not solve $\mathcal{P}$. Since the length of π is linearly bounded, its existence can be decided by an NP oracle in one step. We accept iff the oracle answers no. Therefore, the problem is in Σ_2^p.

Hardness: We reduce from QSAT$_2 \cap$ 3-DNF. Fix an instance $\Psi = \exists X \forall Y \psi(X, Y)$ and write $\psi \equiv \bigvee_{i=1}^{|T|} T_i$, where each T_i is a term with three literals and $|T|$ is the number of terms. We construct a positive-only k-bounded model repair instance as follows:

Facts. Let $F = \{f\}$. The sole fact f is used to mark the assignments to X that are not chosen.

Actions. We introduce the following actions:

1. For each $i \in \{1, \dots, |X|\}$, two X-actions encoding the assignment to X: $\delta(a_{setX}^{i,\top}) = \delta(a_{setX}^{i,\bot}) = (\emptyset, \emptyset, \emptyset)$.
2. For each $i \in \{1, \dots, |Y|\}$, two Y-actions encoding the assignment to Y: $\delta(a_{setY}^{i,\top}) = \delta(a_{setY}^{i,\bot}) = (\emptyset, \emptyset, \emptyset)$.
3. A check-action used to block action traces corresponding to the chosen assignment to X: $\delta(a_{check}) = (\{f\}, \emptyset, \emptyset)$.

Constraints and model repair instance. We construct an LTL$_\mathsf{p}^{\mathsf{len}}$ formula φ to specify the pattern of action traces that must be solutions to the repaired planning problem:

$$\begin{aligned}\varphi \;=\; & \bigwedge_{i=1}^{|X|} \mathsf{X}^{i-1}\big(a_{setX}^{i,\top} \vee a_{setX}^{i,\bot}\big) \\ & \wedge \bigwedge_{i=1}^{|Y|} \mathsf{X}^{|X|+i-1}\big(a_{setY}^{i,\top} \vee a_{setY}^{i,\bot}\big) \\ & \wedge \mathsf{X}^{|X|+|Y|} a_{check} \\ & \wedge \bigwedge_{i=1}^{|T|} \big(\mathsf{F}\, action(i,1) \\ & \qquad \wedge\; \mathsf{F}\, action(i,2) \\ & \qquad\quad \wedge\; \mathsf{F}\, action(i,3) \to \neg \mathsf{F} a_{check}\big) \\ & \wedge\; \mathsf{X}^{|X|+|Y|}\mathsf{L},\end{aligned}$$

where the function *action* maps each pair (i, j) to the action corresponding to the j-th literal of the i-th term in ψ. For example, if $T_i = x_1 \wedge y_2 \wedge \neg x_3$, then we have $action(i, 1) = a_{setX}^{1,\top}$, $action(i, 2) = a_{setY}^{2,\top}$, and $action(i, 3) = a_{setX}^{3,\bot}$.

We now construct our main constraint

$$c = (s_0, g, \varphi, \text{positive}) \quad \text{with } s_0 = g = \emptyset,$$

which enforces the main requirement $L(\mathcal{P}) \supseteq L(\varphi)$.

Next, for each $x_i \in X$, we introduce a formula φ_i. Let

$$\varphi_i \;=\; a_{setX}^{i,\top} \;\wedge\; \mathsf{X}\, a_{setX}^{i,\bot} \;\wedge\; \mathsf{XL},$$

and for each i, we add a positive constraint

$$c_x^i = (s_0, g, \varphi_i, \text{positive}) \quad \text{with } s_0 = \emptyset \text{ and } g = \{f\}.$$

Let $M = (F, A, \delta)$ be the model to be repaired, and let

$$C \;=\; \{c\} \;\cup\; \{c_x^i \mid 1 \le i \le |X|\}$$

be the set of constraints. We construct the positive-only model repair instance $\mathcal{MR}_k = \big((M, C),\, k\big)$ with $k = |X|$.

Observe that Ψ is true iff the constructed positive-only model repair instance $\mathcal{MR}_k$ is solvable; Σ_2^p-hardness thus follows. We defer the detailed proof and an example to the supplementary material (Sheng and Bercher 2026). □

We move on to the negative-only case:

Theorem 5. *The negative-only k-bounded model repair problem using LTL$_\mathsf{p}^{\mathsf{len}}$ is Σ_2^p-complete.*

Proof. Membership: Guess a valid repair set Δ, then verify whether a counterexample plan of linear length exists. A counterexample plan here is a plan π that satisfies φ and solves $\mathcal{P}$. The existence of such a counterexample plan can be decided by an NP oracle in one step, and we accept iff this oracle answers no. Therefore, the problem is in Σ_2^p.

Hardness: Our hardness proof adapts the one presented by Sreedharan, Bercher, and Kambhampati (2022) for the model reconciliation explanation (MRE) problem.

We reduce from QSAT$_2 \cap$ 3-DNF to our problem. Let $\Psi = \exists X\, \forall Y\, \psi(X, Y)$ with ψ in 3-DNF. Then

$$\Psi \iff \exists X\, \neg \exists Y\, \neg \psi(X, Y).$$

By De Morgan's laws, $\neg\psi$ can be converted into an equivalent 3-CNF formula in linear time. Write $\neg\psi \equiv \bigwedge_{i=1}^{|C|} C_i$, where each C_i is a 3-literal clause over $X \cup Y$. We construct a negative-only k-bounded model repair instance as follows:

Facts. For each $x_i \in X$, create $f_{X,i}^\top$ and $f_{X,i}^\bot$; for each $y_j \in Y$, create $f_{Y,j}^\top$ and $f_{Y,j}^\bot$. We use these facts to simulate truth values of variables, and let F be the set of all these facts.

Actions. We introduce the following actions:

1. a_{setX} with $\delta(a_{setX}) = (\emptyset, \emptyset, \emptyset)$.
2. For each $i \in \{1, \dots, |X|\}$ and $j \in \{1, \dots, |X| + 1\}$, an action ensuring that the assignment to X is well-defined: $\delta(a_{checkX}^{i,j}) = (\{f_{X,i}^\top, f_{X,i}^\bot\}, \emptyset, \emptyset)$.
3. For each $i \in \{1, \dots, |Y|\}$, two actions encoding the assignment to Y: $\delta(a_{setY}^{i,\top}) = (\emptyset, \{f_{Y,i}^\top\}, \{f_{Y,i}^\bot\})$ and $\delta(a_{setY}^{i,\bot}) = (\emptyset, \{f_{Y,i}^\bot\}, \{f_{Y,i}^\top\})$.
4. For each clause $C_i = l_{i,1} \vee l_{i,2} \vee l_{i,3}$, introduce three actions $a_{clause}^{i,j}$ ($j \in \{1, 2, 3\}$) encoding whether the clause is satisfied under the current assignment to X and Y: $\delta(a_{clause}^{i,j}) = (\{fact(l_{i,j})\}, \emptyset, \emptyset)$, where the function *fact* maps each literal to its corresponding fact, e.g., $fact(x_i) = f_{X,i}^\top$, $fact(\neg y_i) = f_{Y,i}^\bot$.

Constraint and model repair instance. The following formula encodes the action traces that must be excluded:

$$\begin{aligned}\varphi \;=\; & \Big(a_{setX} \;\wedge\; \mathsf{X}\big(\bigvee_{i=1}^{|X|} \bigvee_{j=1}^{|X|+1} a_{checkX}^{i,j}\big) \;\wedge\; \mathsf{XL}\Big) \\ & \vee \Big(a_{setX} \;\wedge\; \bigwedge_{i=1}^{|Y|} \mathsf{X}^i \big(a_{setY}^{i,\top} \vee a_{setY}^{i,\bot}\big)\end{aligned}$$

$$\wedge \bigwedge_{i=1}^{|C|} \mathsf{X}^{|Y|+i}\big(a^{i,1}_{clause} \vee a^{i,2}_{clause} \vee a^{i,3}_{clause}\big) \wedge \mathsf{X}^{|Y|+|C|}\mathsf{L}\Big).$$

Let $M = (F, A, \delta)$ be the model to be repaired, and let

$$c = (s_0, g, \varphi, \text{negative}),$$

where $s_0 = F$ and $g = \emptyset$, be the only constraint. We then construct $\mathcal{MR}_k = ((M, \{c\}), k)$ with $k = |X|$.

One can verify that Ψ is true iff $\mathcal{MR}_k$ is solvable, which establishes Σ_2^{p}-hardness. The detailed proof is deferred to the supplementary material (Sheng and Bercher 2026). □

We now consider the existential-only case:

Theorem 6. *The existential-only k-bounded model repair problem using LTL$_\mathsf{p}^{\mathsf{len}}$ is* NP*-complete.*

Proof. Membership: Guess (i) a valid repair set Δ such that $M \Rightarrow_\Delta M'$, and (ii) an example plan π of linear length; together they constitute a polynomial-size certificate. Recall that for M' to satisfy an existential constraint, it suffices to exhibit an example plan that solves the repaired planning problem and satisfies φ. Since φ is an LTL$_\mathsf{p}^{\mathsf{len}}$ formula, any such plan has linear length. Thus, we can check in polynomial time that (i) π is a solution to the repaired planning problem, and (ii) $\pi \models \varphi$. Since the problem is solvable iff there exists a polynomial-size certificate verifiable in polynomial time (Δ and π), we obtain NP-membership.

Hardness: The existential-only k-bounded model repair problem using LTL$_\mathsf{p}^{\mathsf{list}}$ is a special case, and is NP-complete (Thm. 1); NP-hardness of this problem thus follows. □

The general case now follows:

Theorem 7. *The (general) k-bounded model repair problem using LTL$_\mathsf{p}^{\mathsf{len}}$ is* PSPACE*-complete.*

Proof. Membership: Guess a valid repair set Δ and verify the repaired model against each constraint; by Thms. 2 and 4–6, such verification can be performed in polynomial space regardless of the constraint type. Hence, the problem is in PSPACE.

Hardness: The (general) k-bounded model repair problem using LTL$_\mathsf{p}^{\mathsf{list}}$ is a special case of this problem (since LTL$_\mathsf{p}^{\mathsf{list}}$ is a special case of LTL$_\mathsf{p}^{\mathsf{len}}$) and is PSPACE-complete by Thm. 3. PSPACE-hardness thus follows. □

We now turn to full LTL$_\mathsf{p}$, starting with positive-only:

Theorem 8. *The positive-only k-bounded model repair problem using LTL$_\mathsf{p}$ is* PSPACE*-complete.*

Proof. Membership: Guess a valid repair set Δ and obtain the repaired model M'. Recall that for M' to satisfy a positive constraint, we require that $L(\mathcal{P}) \supseteq L(\varphi)$, i.e., there exists no counterexample action trace π that (i) satisfies φ and (ii) does not solve $\mathcal{P}$. To check this in polynomial space, we run an on-the-fly search for a counterexample: nondeterministically generate π one action at a time (so that we can reuse space), and verify whether (i) π fails to solve $\mathcal{P}$ and (ii) $\pi \models \varphi$. This yields an NPSPACE procedure for finding a counterexample; since PSPACE = NPSPACE (Savitch 1970), the complementary problem, i.e., accept iff no counterexample exists, is in PSPACE.

Hardness: Reduce from LTL$_\mathsf{p}$ unsatisfiability (given an LTL$_\mathsf{p}$ formula φ, decide whether $L(\varphi) = \emptyset$), which is PSPACE-complete since its complement, LTL$_\mathsf{p}$ satisfiability, is PSPACE-complete (Fionda and Greco 2016). Let φ be an arbitrary LTL$_\mathsf{p}$ formula. We construct a STRIPS planning model $M = (F, A, \delta)$ with $F \neq \emptyset$ and $A = \emptyset$. Let $C = \{c\}$ where $c = (s_0, g, \varphi, \text{positive})$, $s_0 = \emptyset$, and $g = F$. Consider the model repair instance $\mathcal{MR}_k = ((M, C), k)$ with $k = 0$ (so that no repair is allowed). Since $A = \emptyset$ and $s_0 \subsetneq g$, the planning problem $\mathcal{P} = (M, s_0, g)$ has no solution, hence $L(\mathcal{P}) = \emptyset$. By the semantics of positive constraints, we have $M \models c \iff L(\mathcal{P}) \supseteq L(\varphi) \iff \emptyset \supseteq L(\varphi) \iff L(\varphi) = \emptyset$. Therefore, $\mathcal{MR}_k$ is solvable iff φ is unsatisfiable. PSPACE-hardness thus follows. □

The remaining cases follow by straightforward adaptations of the preceding arguments, and we therefore omit the detailed proofs. For the negative-only and existential-only cases, we reuse the on-the-fly decision procedure from Thm. 8, but now search for a plan in $L(\mathcal{P}) \cap L(\varphi)$. The general case again proceeds by reusing space across constraints.

Theorem 9. *The negative-only k-bounded model repair problem using LTL$_\mathsf{p}$ is* PSPACE*-complete.*

Theorem 10. *The existential-only k-bounded model repair problem using LTL$_\mathsf{p}$ is* PSPACE*-complete.*

Theorem 11. *The (general) k-bounded model repair problem using LTL$_\mathsf{p}$ is* PSPACE*-complete.*

Our result for Thm. 10 is complemented by Gigante, Leofante, and Micheli (2025, Thm. 5), who show PSPACE-completeness for the related setting in which constraints are specified over state traces and repairs are restricted to action preconditions.

Conclusion

In this paper, we propose a unified framework for model repair in STRIPS planning that (i) replaces finite plan sets with constraint languages expressed in LTL$_\mathsf{p}$, thereby allowing the specification of infinitely many plans that share a common *pattern*, and (ii) generalizes constraint semantics via *constraint types*. This framework subsumes existing plan-based approaches and systematically captures the main classes of model repair requirements identified by Bercher, Sreedharan, and Vallati (2025). Our complexity analysis identifies a hierarchy of hardness that is governed by two independent sources: (i) the expressivity of the underlying LTL$_\mathsf{p}$ fragment and (ii) the constraint types that are allowed. In particular, depending on the chosen setting, the complexity ranges from NP through Σ_2^{p} up to PSPACE. These results clarify the computational price of increased temporal expressivity and semantic flexibility. We further elucidate the connections between our framework and existing model repair paradigms, revealing them as special cases within a unified schema and providing a common foundation for future generalizations. A natural next step is to combine our framework with that of Gigante, Leofante, and Micheli (2025), reasoning over action and state traces jointly.

Acknowledgments

Pascal Bercher is the recipient of an Australian Research Council (ARC) Discovery Early Career Researcher Award (DECRA), project number DE240101245, funded by the Australian Government.

References

Aineto, D.; Jiménez Celorrio, S.; and Onaindia, E. 2019. Learning Action Models with Minimal Observability. *Artificial Intelligence*, 275: 104–137.

Baader, F.; and Wassermann, R. 2024. Contractions Based on Optimal Repairs. In *Proceedings of the 21st International Conference on Principles of Knowledge Representation and Reasoning (KR 2024)*, 94–105. IJCAI.

Bacchus, F.; and Kabanza, F. 1996. Planning for Temporally Extended Goals. In *Proceedings of the 13th AAAI Conference on Artificial Intelligence (AAAI 1996)*, 1215–1222. AAAI Press.

Baier, J. A.; and McIlraith, S. A. 2006a. Planning with First-Order Temporally Extended Goals Using Heuristic Search. In *Proceedings of the 21st AAAI Conference on Artificial Intelligence (AAAI 2006)*, 788–795. AAAI Press.

Baier, J. A.; and McIlraith, S. A. 2006b. Planning with Temporally Extended Goals using Heuristic Search. In *Proceedings of the 16th International Conference on Automated Planning and Scheduling (ICAPS 2006)*, 342–345. AAAI Press.

Bavandpour, N. K.; Lauer, P.; Lin, S.; and Bercher, P. 2025. Repairing Planning Domains Based on Lifted Test Plans. In *Proceedings of the 28th European Conference on Artificial Intelligence (ECAI 2025)*, 4774–4781. IOS Press.

Bercher, P.; Sreedharan, S.; and Vallati, M. 2025. A Survey on Model Repair in AI Planning. In *Proceedings of the 34th International Joint Conference on Artificial Intelligence (IJCAI 2025)*, 10371–10380. IJCAI.

Bonet, B.; and Geffner, H. 2001. Planning as Heuristic Search. *Artificial Intelligence*, 129(1): 5–33.

Bylander, T. 1994. The Computational Complexity of Propositional STRIPS Planning. *Artificial Intelligence*, 69(1): 165–204.

Camacho, A.; and McIlraith, S. A. 2019. Learning Interpretable Models Expressed in Linear Temporal Logic. In *Proceedings of the 29th International Conference on Automated Planning and Scheduling (ICAPS 2019)*, 621–630. AAAI Press.

Chatzieleftheriou, G.; Bonakdarpour, B.; Katsaros, P.; and Smolka, S. A. 2015. Abstract Model Repair. *Logical Methods in Computer Science*, 11(3).

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

De Giacomo, G.; and Vardi, M. Y. 1999. Automata-Theoretic Approach to Planning for Temporally Extended Goals. In *Proceedings of the 5th European Conference on Planning (ECP 1999)*, 226–238. Springer.

De Giacomo, G.; and Vardi, M. Y. 2013. Linear Temporal Logic and Linear Dynamic Logic on Finite Traces. In *Proceedings of the 23rd International Joint Conference on Artificial Intelligence (IJCAI 2013)*, 854–860. AAAI Press.

Erol, K.; Nau, D. S.; and Subrahmanian, V. S. 1995. Complexity, Decidability and Undecidability Results for Domain-Independent Planning. *Artificial Intelligence*, 76(1-2): 75–88.

Fikes, R. E.; and Nilsson, N. J. 1971. STRIPS: A New Approach to the Application of Theorem Proving to Problem Solving. *Artificial Intelligence*, 2(3-4): 189–208.

Fionda, V.; and Greco, G. 2016. The Complexity of LTL on Finite Traces: Hard and Easy Fragments. In *Proceedings of the 30th AAAI Conference on Artificial Intelligence (AAAI 2016)*, 971–977. AAAI Press.

Fionda, V.; and Greco, G. 2018. LTL on Finite and Process Traces: Complexity Results and a Practical Reasoner. *Journal of Artificial Intelligence Research*, 63: 557–623.

Ghallab, M.; Nau, D. S.; and Traverso, P. 2004. *Automated Planning: Theory and Practice*. Elsevier.

Gigante, N.; Leofante, F.; and Micheli, A. 2025. Counterfactual Scenarios for Automated Planning. In *Proceedings of the 22nd International Conference on Principles of Knowledge Representation and Reasoning (KR 2025)*, 794–804. IJCAI.

Gnad, D.; Hecher, M.; Gaggl, S. A.; Rusovac, D.; Speck, D.; and Fichte, J. K. 2025. Interactive Exploration of Plan Spaces. In *Proceedings of the 22nd International Conference on Principles of Knowledge Representation and Reasoning (KR 2025)*, 599–609. IJCAI.

Gragera, A.; Fuentetaja, R.; Olaya, Á. G.; and Fernández, F. 2023. A Planning Approach to Repair Domains with Incomplete Action Effects. In *Proceedings of the 33rd International Conference on Automated Planning and Scheduling (ICAPS 2023)*, 153–161. AAAI Press.

Gragera, A.; Fuentetaja, R.; Olaya, Á. G.; and Fernández, F. 2025. On the Gains from Using Action Observations in Domain Repair. In *Proceedings of the 35th International Conference on Automated Planning and Scheduling (ICAPS 2025)*, 343–347. AAAI Press.

Grundke, C.; Röger, G.; and Helmert, M. 2024. Formal Representations of Classical Planning Domains. In *Proceedings of the 34th International Conference on Automated Planning and Scheduling (ICAPS 2024)*, volume 34, 239–248. AAAI Press.

Helmert, M. 2006. The Fast Downward Planning System. *Journal of Artificial Intelligence Research*, 26: 191–246.

Hoffmann, J.; and Nebel, B. 2001. The FF Planning System: Fast Plan Generation Through Heuristic Search. *Journal of Artificial Intelligence Research*, 14: 253–302.

Höller, D.; Behnke, G.; Bercher, P.; and Biundo, S. 2014. Language Classification of Hierarchical Planning Problems. In *Proceedings of the 21st European Conference on Artificial Intelligence (ECAI 2014)*, 447–452. IOS Press.

Höller, D.; Behnke, G.; Bercher, P.; and Biundo, S. 2016. Assessing the Expressivity of Planning Formalisms through

the Comparison to Formal Languages. In *Proceedings of the 26th International Conference on Automated Planning and Scheduling (ICAPS 2016)*, 158–165. AAAI Press.

Hopcroft, J. E.; Motwani, R.; and Ullman, J. D. 2001. *Introduction to automata theory, languages, and computation, 2nd Edition*. Addison-Wesley-Longman.

Kornherr, M.; Correa, A. B.; Gaggl, S. A.; Hecher, M.; Rusovac, D.; Speck, D.; Fichte, J. K.; and Gnad, D. 2025. Graphical Navigation in Solution Spaces using PlanPilot. In *System Demonstrations and Exhibits Program at the 35th International Conference on Automated Planning and Scheduling (ICAPS 2025)*.

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 30th International Joint Conference on Artificial Intelligence (IJCAI 2021)*, 4152–4159. IJCAI.

Lin, S.; and Bercher, P. 2022. On the Expressive Power of Planning Formalisms in Conjunction with LTL. In *Proceedings of the 32nd International Conference on Automated Planning and Scheduling (ICAPS 2022)*, 231–240. AAAI Press.

Lin, S.; Grastien, A.; and Bercher, P. 2023. Towards Automated Modeling Assistance: An Efficient Approach for Repairing Flawed Planning Domains. In *Proceedings of the 37th AAAI Conference on Artificial Intelligence (AAAI 2023)*, 12022–12031. 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 the 39th AAAI Conference on Artificial Intelligence (AAAI 2025)*, 26596–26604. AAAI Press.

Savitch, W. J. 1970. Relationships Between Nondeterministic and Deterministic Tape Complexities. *Journal of Computer and System Sciences*, 4(2): 177–192.

Sheng, H.; and Bercher, P. 2026. Supplementary Material for "Complexity Results for Fixing Classical Models Using LTL to Express Which Solutions Are (Un)Desired". DOI: 10.25911/5ZWE-4K22.

Simon, S.; and Röger, G. 2015. Finding and Exploiting LTL Trajectory Constraints in Heuristic Search. In *Proceedings of the 8th Annual Symposium on Combinatorial Search (SoCS 2015)*, 113–121. AAAI Press.

Speck, D.; Hecher, M.; Gnad, D.; Fichte, J. K.; and Corrêa, A. B. 2025. Counting and Reasoning with Plans. In *Proceedings of the 39th AAAI Conference on Artificial Intelligence (AAAI 2025)*, 26688–26696. AAAI Press.

Sreedharan, S.; Bercher, P.; and Kambhampati, S. 2022. On the Computational Complexity of Model Reconciliations. In *Proceedings of the 31st International Joint Conference on Artificial Intelligence (IJCAI 2022)*, 4657–4664. IJCAI.

Sreedharan, S.; Chakraborti, T.; and Kambhampati, S. 2021. Foundations of Explanations as Model Reconciliation. *Artificial Intelligence*, 301: 103558.

Stockmeyer, L. J. 1976. The Polynomial-Time Hierarchy. *Theoretical Computer Science*, 3(1): 1–22.

Welt, M.; Lodemann, A.; Olz, C.; Bercher, P.; and Glimm, B. 2025. Calculating Optimal Corrections for Unsolvable Planning Problems. In *Proceedings of the 28th European Conference on Artificial Intelligence (ECAI 2025)*, 4637–4644. IOS Press.

Zhuo, H. H.; Nguyen, T.; and Kambhampati, S. 2013. Refining Incomplete Planning Domain Models Through Plan Traces. In *Proceedings of the 23rd International Joint Conference on Artificial Intelligence (IJCAI 2013)*, 2451–2457. AAAI Press.