Relaxing is Hard: Complexity Results for Lifted Partial Order Causal Link Planning

Harrison Oates, Pascal Bercher

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

Abstract

We investigate Partial Order Causal Link (POCL) planning in the lifted setting, where plan steps are represented as parameterised action schemas. While delete relaxation has been shown to reduce the complexity of plan existence in the ground POCL setting, operating on lifted representations may at times be necessary to avoid the prohibitive cost of grounding. This motivates a separate complexity analysis for the lifted case. We formalise delete-relaxation via filter functions that selectively suppress delete effects, yielding differently strong relaxation semantics, with and without respect for pre-existing delete effects and causal links of the input plan. We prove that plan existence is EXPTIME–complete for most variants, even when the input plan is totally ordered. Further, we prove results ranging from NP–completeness to PSPACE–completeness for fixed-schema plan existence, with a tractable case when planning is done from scratch.

Introduction

Partial Order Causal Link (POCL) planning is a plan-space search framework that embodies the principle of least commitment (Weld 1994), where decisions about step ordering are delayed until absolutely necessary. By maintaining partial orderings over plan steps, POCL planners can compactly represent exponentially many valid action sequences in a single search node (Minton, Bresina, and Drummond 1994), leading to a significant reduction in the branching factor of the search space. In the lifted setting, where action schemas use variables rather than ground objects, this flexibility extends to variable assignments, allowing further deferral of commitments and further pruning of the search space (Younes and Simmons 2002).

While interest in partial order planning waned in the classical setting with the rise of better–scaling approaches, POCL methods remain intuitively attractive in domains where flexible execution, concurrency, or fault tolerance matter (Vidal and Geffner 2006; Weld 2011; Bercher and Olz 2020; Bit-Monnot and Godet 2025; Oates and Bercher 2026). For instance, POCL–style planners have been deployed in contexts such as Mars rover operations (Bresina et al. 2005) and human assistance systems (Pollack 2002; Bercher et al. 2014). Hybrid approaches have also emerged, combining partial order reasoning with state-based forward search (Coles et al. 2010). POCL planning techniques are also used in many modern hierarchical planners (Bercher, Lin, and Alford 2022), such as ARIES (Bit-Monnot 2023), OptiPlan (Firsov, Fiorino, and Pellier 2023), and FAPE (Bit-Monnot et al. 2020). Another major area where POCL plans have found sustained relevance is in plan *optimization*. In particular, POCL–based frameworks allow the fine–grained insertion and reordering of actions necessary to improve plan quality. Siddiqui and Haslum (2015), for instance, decompose a POCL plan into non-interleaving subplans, called blocks, which are then iteratively improved. Recent work has also shown that POCL plans can achieve strictly shorter execution times than Graphplan-style parallel plans (Blum and Furst 1997) for the same problem (Oates and Bercher 2026), demonstrating that their flexible ordering yields better-quality solutions under concurrent execution. A comprehensive overview of POCL's role in plan optimization is provided by Bercher, Haslum, and Muise (2024).

Most theoretical investigations of POCL planning have been limited to the *ground* (i.e., *propositional*) setting. In particular, Bercher (2021) delivered a comprehensive analysis of plan existence in ground POCL planning, establishing tight complexity bounds for both unrelaxed planning and various delete relaxations focussing on the pruning power of causal links. However, this analysis does not address the scalability challenges posed by *hard-to-ground* domains, which are problems where the grounding process leads to a combinatorial explosion in the number of facts and actions (Lauer et al. 2021; Corrêa and De Giacomo 2024). As argued by Corrêa et al. (2021), planning systems must reason over *lifted* representations to avoid this bottleneck. While lifted plan existence has been studied for classical planning (Erol, Nau, and Subrahmanian 1995), no corresponding results exist for POCL planning. This is particularly surprising given that reasoning over lifted plans was one of the main reasons for the success of and interest in early POCL planners (McAllester and Rosenblitt 1991; Weld 1994; Younes and Simmons 2002).

This paper closes that gap by extending the work of Bercher (2021) to the lifted setting. We introduce a unified framework for delete relaxations in POCL planning that systematically varies which steps are relaxed and how pre-existing causal links are treated. With this framework,

Relaxation Type	Deletes (Initial / New)	Respects Causal Links?	Complexity		Theorem(s)
			General Lifted	Fixed Schema	
None (Unrelaxed)	Original / Original	N/A	EXPSPACE-c	PSPACE-c	Thm. 1 / Thm. 6
P-relaxed	Relaxed / Original	No/Yes	EXPSPACE-c	PSPACE-c	Cor. 1 / Cor. 4
A-relaxed	Original / Relaxed	No	EXPTIME-c	NP-c	Thm. 2 / Thm. 7
AP-relaxed	Relaxed / Relaxed	No	EXPTIME-c	NP-c	Thm. 4 / Thm. 7
A-relaxed (RL)	Original / Relaxed	Yes	EXPTIME-c	NP-c	Thm. 3 / Thm. 7
AP-relaxed (RL)	Relaxed / Relaxed	Yes	EXPTIME-c	NP-c	Thm. 5 / Thm. 7
Special Case: A/AP-Relaxed with Empty Plan		No/Yes	-	P	Cor. 5

Table 1: Complexity results for PLAN EXISTENCE in lifted POCL planning. All results for the general setting hold for both partially and totally ordered input plans. "(RL)" denotes variants that respect causal links. "-c" is an abbreviation for "-complete".

we show that delete relaxation is still hard: plan existence is EXPSPACE–complete, while the more practical relaxed variants are EXPTIME–complete. Since the schema set remains fixed during search while only the partial plan grows, we also analyze the fixed-schema setting. Here, complexity drops to PSPACE–complete and NP–complete with delete relaxation, with ground input plans necessary to obtain fragments in P. Our results are summarised in Table 1.

Problem Formalization

Following Lauer et al. (2021), we define a (lifted) *classical planning problem* as a tuple $\Pi = (\mathcal{P}, \mathcal{O}, \mathcal{A}, \mathcal{I}, \mathcal{G})$ where $\mathcal{P}$ is a finite set of first-order predicates, $\mathcal{O}$ is a finite set of objects, $\mathcal{A}$ is a finite set of action schemas, $\mathcal{I}$ is the initial state, and $\mathcal{G}$ is the goal description. Each predicate $p \in \mathcal{P}$ is associated with a tuple of parameter variables V_p and has fixed arity $|V_p|$. We write $p(v_1, \ldots, v_{|V_p|})$ to explicitly indicate these parameters. Predicates can be instantiated by replacing parameters with objects in $\mathcal{O}$, or can be replaced with other variables via substitution. A predicate is *ground* if all parameters are instantiated with objects. We denote the set of ground predicates derivable from p as $p^{\mathcal{O}}$. The set of all ground predicates in the task is $\mathcal{P}^{\mathcal{O}} = \bigcup_{p \in \mathcal{P}} p^{\mathcal{O}}$. A *state* is any set $s \in 2^{\mathcal{P}^{\mathcal{O}}}$. Thus, $\mathcal{I} \in 2^{\mathcal{P}^{\mathcal{O}}}$ and $\mathcal{G} \subseteq \mathcal{P}^{\mathcal{O}}$, as the goal description implicitly defines a set of goal states $G = \{s \mid s \supseteq \mathcal{G}\}$.

An action schema $a = (V_a, pre(a), add(a), del(a))$ is a tuple consisting of a finite set of parameter variables V_a, and finite sets of preconditions $pre(a)$, add effects $add(a)$, and delete effects $del(a)$. Each of these is a finite set of predicates in $\mathcal{P}$ instantiated by substituting each variable with some element in $V_a \cup \mathcal{O}$. A parameter variable is in V_a if and only if it is in $pre(a) \cup add(a) \cup del(a)$. As with predicates, the arity of a is $|V_a|$, and each schema can be instantiated by replacing each $v \in V_a$ with some $o \in \mathcal{O}$ to obtain (ground) actions. The set of all possible actions is $\mathcal{A}^{\mathcal{O}}$. As the arity of predicates and action schemas is not bounded, $\mathcal{P}^{\mathcal{O}}$ and $\mathcal{A}^{\mathcal{O}}$ are of size exponential in the size of Π.

An action a is applicable in a state s if and only if $pre(a) \subseteq s$. If a is applicable in s, the state transition function $\gamma : \mathcal{A}^{\mathcal{O}} \times 2^{\mathcal{P}^{\mathcal{O}}} \to 2^{\mathcal{P}^{\mathcal{O}}}$ returns the successor state $\gamma(a, s) = (s \setminus del(a)) \cup add(a)$. By convention, we apply delete effects before add effects during state transitions, which ensures that any facts to be removed are eliminated from the current state before introducing new facts, maintaining consistency in the resulting state (Fikes and Nilsson 1971). A sequence of actions $\bar{a} = a_1, \ldots, a_n$ is applicable in a state s_0 if there exists a sequence of states $s_0, \ldots, s_n$ such that for all $i \in [1, n]$, it holds that a_i is applicable in state s_{i-1} and generates state $s_i = \gamma(a_i, s_{i-1})$. s_n is called the state generated by $\bar{a}$.

Definition 1 (Classical Solution to Classical Problem)**.** *An action sequence $\bar{a}$ is called a classical solution to a classical planning problem $(\mathcal{P}, \mathcal{O}, \mathcal{A}, \mathcal{I}, \mathcal{G})$ if and only if it is applicable in $\mathcal{I}$ and generates a goal state $s \supseteq \mathcal{G}$.*

We briefly introduce concepts from unification theory (Siekmann 1989) that are essential for our formalisation. A substitution σ is a mapping from variables to terms, written as $\sigma = \{v_1 \leftarrow t_1, \ldots, v_k \leftarrow t_k\}$ where each variable v_i is mapped to a term t_i for $0 \leq i \leq k$, while all other variables remain unchanged. Applying a substitution to a term t, denoted σt, replaces each variable in t according to σ. σ is called a unifier of two terms t_1 and t_2 if $\sigma t_1 = \sigma t_2$. In this case, we say that σ *unifies* t_1 and t_2. We extend substitution recursively over complex structures by applying it component-wise over tuples and element-wise over sets. For a predicate $p(v_1, \ldots, v_n)$, a substitution is applied to each argument: $\sigma p(v_1, \ldots, v_n) = p(\sigma v_1, \ldots, \sigma v_n)$. For an action schema, a substitution is applied to all preconditions, add effects, and delete effects:

$$\sigma a = (\sigma V_a, \sigma pre(a), \sigma add(a), \sigma del(a)).$$

For any set S, we apply σ element-wise:

$$\sigma S = \{\sigma x \mid x \in S\}.$$

Let $a = (V_a, pre(a), add(a), del(a))$ be an action schema with a finite set of parameter variables V_a. An action schema $a' = (V_{a'}, pre(a'), add(a'), del(a'))$ is an *action schema variation* of a if a and a' are unifiable. An action schema variation may remain fully or partially lifted, meaning that some variables may still require grounding.

Following the notation of Bercher (2021), a (partial) POCL plan is a tuple $(PS, \prec, CL, VC)$ consisting of a finite set of plan steps PS, with each plan step $(l, a) \in PS$ consisting of an action schema variation a and a unique label l. This labelling is required to differentiate between multiple occurrences of an action schema variation in the same plan. $\prec \subseteq PS \times PS$ is a strict partial order over the plan steps. Given $ps = (l, a) \in PS$, we write $pre(ps)$, $add(ps)$, and $del(ps)$ as shorthand for $pre(a)$, $add(a)$, and $del(a)$.

$CL \subseteq PS \times \mathcal{P} \times PS$ is a finite set of causal links. A causal link $(ps, f, ps') \in CL$ implies that $f \in add(ps) \cap pre(ps')$: that is, the precondition f of ps' is to be fulfilled by an effect of ps. Every causal link $(ps, f, ps') \in CL$ also implies $(ps, ps') \in \prec$. We further say that f is *protected* by that link, as f will not be deleted between these steps in any derived solution. A causal link must refer to an identical lifted predicate in both the add effect of ps and the precondition of ps'. If the predicate in $add(ps)$ and $pre(ps')$ use different variable names, a unifying substitution must be applied to both plan steps before establishing the link.

Extending our definition of substitution over a POCL plan $P = (PS, \prec, CL, VC)$ is intuitive — simply apply the substitution component-wise:

$$\sigma P = (\sigma PS, \sigma \prec, \sigma CL, \sigma VC),$$

where $\sigma ps = (l, \sigma a)$ for all $ps \in PS$.

A plan step ps'' threatens a causal link (ps, f, ps') if and only if there exists a substitution σ on the plan such that σ unifies f with some $f' \in del(ps'')$ and the transitive closure of $\prec \cup \{(ps, ps''), (ps'', ps')\}$ is a strict partial order. We say that a plan satisfying such a condition raises a *causal threat*. The transitive closure requirement ensures that the plan remains consistent and acyclic when extending the ordering constraints to capture all direct dependencies that arise when considering the threatening step ps''. A strict partial order is both irreflexive (meaning that no step precedes itself) and transitive (meaning that if $a \prec b$ and $b \prec c$, then $a \prec c$). Requiring that the closure remains a strict partial order ensures that the ordering constraints needed to recognize the causal threat do not introduce cycles. If a cycle were introduced, it would imply that some step must occur both before and after itself, making the plan infeasible. By enforcing a strict partial order, we ensure that all steps can still be arranged in a valid execution sequence.

The finite set of variable constraints VC limits the values that variables in action schema variations can take when grounded. While VC in other formalisations may express both co-designation and non-co-designation constraints, our approach simplifies VC to focus solely on non-co-designation. Co-designation requirements, such as forcing two variables to be equal or a variable to equal a constant, are managed through variable renaming or substitution instead. This has the side effect that the cardinality of VC does not grow unnecessarily, which is important for our purposes as computational complexity is measured with respect to the size of the input representation. In particular, avoiding redundant co-designation constraints prevents artificial inflation of the problem encoding. Consequently, each constraint in VC is either of the form $v_1 \neq v_2$, ensuring that the variables v_1 and v_2 are assigned different objects, or $v \neq o$, ensuring variable v is not assigned the object $o \in \mathcal{O}$. Constraints without free variables are redundant and are therefore eliminated from VC.

To encode the initial and goal states of a classical problem within a POCL framework, we must introduce two special plan steps: *init* and *goal*. The *init* step precedes all other steps in the partial ordering, while the *goal* step follows all other steps, establishing the boundaries of plan execution. These special steps are associated with similarly-named actions, which are not in $\mathcal{A}$ so cannot be inserted. *init* has no preconditions and no delete effects ($pre(init) = del(init) = \emptyset$), while its add effects correspond exactly to the initial state ($add(init) = \mathcal{I}$). The *goal* action's preconditions represent the goal conditions ($pre(goal) = \mathcal{G}$), while having no add effects or delete effects ($add(goal) = del(goal) = \emptyset$). This ensures that any valid POCL plan must begin with the conditions specified in the initial state and conclude by satisfying all goal conditions. This setup allows us to formally define when a POCL plan solves a classical planning problem.

Definition 2 (POCL Solution to Classical Problem)**.** *A POCL plan is called a POCL solution to a classical problem if and only if:*

- *every precondition of its plan steps is protected by a causal link,*
- *the plan does not raise any causal threats,*
- *all plan steps are fully ground, and*
- *the plan's set of variable constraints VC is empty.*

Definition 1 and Definition 2 are commonly known to relate in the following manner:

Proposition 1 (Solution Correspondence (Bercher 2021))**.** *Let Π be a classical planning problem. Then, Π has a classical solution if and only if it has a POCL solution.*

This equivalence follows directly from the structure of POCL plans. A POCL solution compactly encodes a set of classical solutions as any linearisation of its steps that respects the partial order yields a valid classical plan. Conversely, any classical sequence can be transformed into a totally ordered POCL Plan by introducing causal links for each precondition, which takes time polynomial in the length of the plan (Kambhampati and Kedar 1994; Muise, Beck, and McIlraith 2016).

To formalise the decision problem of determining whether a particular search node is solvable, we introduce a definition for a POCL planning problem following Bercher (2021). Unlike classical planning problems, which are defined by an initial state and goal description, our definition allows the problem to solve to be any arbitrary POCL plan. This is a natural generalisation, as every search node generated during POCL search is itself a partial plan (Younes and Simmons 2003) and so can be viewed as a POCL problem.

A *POCL planning problem* is a tuple $\Pi_{\text{POCL}} = (\mathcal{P}, \mathcal{O}, \mathcal{A}, Pl_{\mathcal{I},\mathcal{G}})$ consisting of a finite set of predicates $\mathcal{P}$, a finite set of objects $\mathcal{O}$, a finite set of action schemas $\mathcal{A}$, and an initial partial plan $Pl_{\mathcal{I},\mathcal{G}} = (PS_I, \prec_I, CL_I, VC_I)$. The problem is *totally ordered* if $\prec_I$ is a total order, and *partially ordered* if $\prec_I$ is a

Algorithm 1: Verify variable constraints in a POCL plan

Input: Initial plan $Pl_{\mathcal{I},\mathcal{G}} = (PS_I, \prec_I, CL_I, VC_I)$, solution plan $P = (PS, \prec, CL, VC)$
Output: True iff P satisfies VC_I

// Step 1: Extract variable bindings
Initialize empty substitution σ
for each plan step $(l, a) \in PS_I$ **do**
    Identify corresponding step $(l, a') \in PS$
    for each $pred_I \in \text{pre}(a) \cup \text{add}(a) \cup \text{del}(a)$ **do**
        Let $pred_I = Q(v_1, \ldots, v_n)$ and corresponding predicate $pred = Q(t_1, \ldots, t_n)$ in a'
        for each v_i such that v_i is not an object **do**
            if σv_i is defined and $\sigma v_i \neq t_i$ **then**
                return false ▷ Inconsistent binding
            Add binding $v_i \leftarrow t_i$ to σ
// Step 2: Check variable constraints
for each $(v \neq t) \in VC_I$ **do**
    if $\sigma v = \sigma t$ **then**
        return false ▷ Constraint violated
return true ▷ All constraints satisfied

partial order. With the exception of *init* and *goal*, each plan step in PS_I must be an instantiation of a schema in $\mathcal{A}$.

A solution to a POCL planning problem is any solution plan obtained by refining $Pl_{\mathcal{I},\mathcal{G}}$. Such a solution may add plan steps, causal links, or ordering constraints, but may not remove or modify any element already present in $Pl_{\mathcal{I},\mathcal{G}}$, with the exception of grounding free variables via substitution. This aligns with the general approach of refinement planning (Kambhampati 1997). Additionally, any valid refinement must satisfy the initial set of variable constraints. A polynomial-time procedure for checking constraint satisfaction is provided in Algorithm 1. Just as we defined solution criteria for classical problems, we now formalise the solution criteria for POCL problems.

Definition 3 (POCL Solution to POCL Problem)**.** *Let $\Pi_{\text{POCL}} = (\mathcal{P}, \mathcal{O}, \mathcal{A}, Pl_{\mathcal{I},\mathcal{G}})$ be a POCL planning problem and P be a POCL plan, with $P = (PS, \prec, CL, VC)$ and $Pl_{\mathcal{I},\mathcal{G}} = (PS_I, \prec_I, CL_I, VC_I)$. P is a POCL solution to Π_{POCL} if and only if:*

- *There exists a substitution σ such that $PS \supseteq \sigma PS_I$, $\prec \supseteq \sigma \prec_I$, $CL \supseteq \sigma CL_I$, and $VC = \emptyset$.*
- *The decision procedure defined in Algorithm 1 returns true.*
- *Every precondition of P's plan steps is protected by a causal link,*
- *P does not raise any causal threat, and*
- *all plan steps in P are fully ground.*

This definition builds upon Definition 2 by formalizing a solution as a valid ground refinement of the initial plan. The refinement is anchored by the plan step labels, which are treated as immutable. Since the substitution σ acts on action schemas but not their labels (i.e., $\sigma(l, a) = (l, \sigma a)$), this formalism inherently prohibits the renaming of plan steps.

This design choice is critical for ensuring that verifying a solution does not add additional complexity. If arbitrary renaming were permitted, checking if a given plan is a valid refinement would require solving the subgraph isomorphism problem, which is NP-complete (Cook 1971). In contrast, our approach ensures verification of the refinement criterion can be performed in polynomial time relative to the size of P. This is because the required substitution is implicitly defined by the groundings in the solution's steps, and the method for extracting it is straightforward: Algorithm 1 demonstrates the extraction process in its first step. We now turn our attention to delete relaxation.

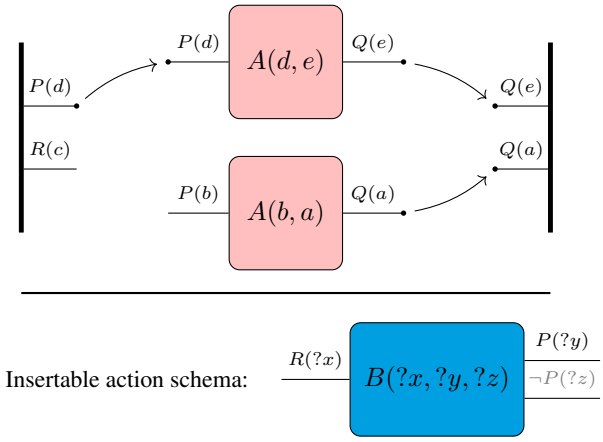

Figure 1: Initial plan with action schema B insertable and a causal link protecting $P(d)$. Under A-relaxation, $B(c, b, d)$ is delete-free and poses no threat, so can be inserted. Under A-relaxation respecting causal links, the unrelaxed $B(c, b, d)$ would delete $P(d)$, blocking insertion.

Delete Relaxation

In classical planning problems, delete relaxation is a syntactic transformation where the delete effects of all action schemas are ignored by setting $del(a) = \emptyset$ for all $a \in \mathcal{A}$ (Hoffmann and Nebel 2001). In POCL planning, however, this treatment is insufficient. Unlike progression search, where the 'current state' represents search progress, in POCL search progress is captured by the partial plan, especially causal links. Causal links serve to prune the search space by preventing a threatening step from being inserted in the interval they protect. If they are ignored in relaxation, we risk admitting plans that are impossible in the original problem. An example is given in Figure 1. A new step, when relaxed, may appear compatible with an existing causal link, but its original, unrelaxed counterpart could threaten that same link. Allowing such a step creates a *specious relaxation*: we would be admitting a path in the relaxed search space that is known to be impossible in the corresponding unrelaxed problem. This can mislead heuristics based on this relaxation by providing finite-cost estimates for what are, in fact, unsolvable subproblems.

Building on this motivation, we go beyond the case-by-case analysis of Bercher (2021) and introduce a novel unified framework that systematically characterises delete relaxation along two orthogonal dimensions. The first dimen-

sion is the scope of delete relaxation, which specifies which plan steps' delete effects are ignored: steps from the initial plan ($ps \in PS_I$), which we call **P**re-existing steps, or newly added steps ($ps \notin PS_I$), called **A**dded steps. This yields three main relaxation types: **A-relaxation** (ignore delete effects of added steps only), **P-relaxation** (ignore delete effects of pre-existing steps only), and **AP-relaxation** (ignore delete effects of all steps)[1]. The second dimension concerns the treatment of causal links, i.e, how threats against causal links are evaluated: under **ignore-links**, threats against all links are evaluated using the relaxed (filtered) delete effects, whereas under **respect-links**, Threats against initial links ($L \in CL_I$) are evaluated using the *original* delete effects (thus preserving the initial plan's integrity), while threats to new links ($L \notin CL_I$) are evaluated with the relaxed deletes. This two-dimensional framework allows us to define a single, generalized notion of a relaxed solution. We formalize the scope of relaxation with a filter function $\phi_{\mathcal{R}}$, where $\mathcal{R} \in \{\text{A, P, AP}\}$.

Definition 4 (($\mathcal{R}, \mathcal{L}$)–Relaxed Solution)**.** *Let Π_{POCL} be a POCL problem with initial plan $Pl_{\mathcal{I},\mathcal{G}} = (PS_I, \prec_I, CL_I, VC_I)$, let $\mathcal{R} \in \{$A, P, AP$\}$ be the relaxation type, and let $\mathcal{L} \in \{$ignore-links, respect-links$\}$ be the link handling policy. A POCL plan P is an $(\mathcal{R}, \mathcal{L})$–relaxed solution to Π_{POCL} if it meets all criteria from Definition 3, with the condition for causal threats modified as follows.*

A plan step ps'' threatens a causal link $L = (ps, f, ps') \in CL$ if and only if the transitive closure of $\prec \cup \{(ps, ps''), (ps'', ps')\}$ is a strict partial order, and there exists a substitution σ unifying f with some $f' \in \Delta$, where the applicable delete set Δ is defined as:

$$\Delta = \begin{cases} del(ps'') & \text{if } \mathcal{L} = \text{respect-links and } L \in CL_I \\ \phi_{\mathcal{R}}(del(ps'')) & \text{otherwise} \end{cases}$$

The filter function $\phi_{\mathcal{R}}$ is defined according to the relaxation type $\mathcal{R}$:

$$\phi_{\mathcal{R}}(del(ps)) = \begin{cases} \emptyset & \text{if } (\mathcal{R} = A \wedge ps \notin PS_I) \\ & \vee (\mathcal{R} = P \wedge ps \in PS_I) \\ & \vee (\mathcal{R} = AP) \\ del(ps) & \text{otherwise} \end{cases}$$

This unified framework allows us to define all relaxation variants in a uniform way. For readability, we adopt consistent shorthand to refer to combinations of relaxation scope and link-handling policy. A solution satisfying the (A, ignore-links) condition is simply called an *A-relaxed solution*, and similarly for P-relaxed and AP-relaxed solutions. When the causal integrity of the initial plan is protected ($\mathcal{L}$ = respect-links), we append 'that respects links' to the name (e.g., an *A-relaxed solution that respects links*). Note that the *AP-relaxation* under ignore-links corresponds to the fully relaxed setting of Bercher (2021), though our terminology makes explicit which steps are relaxed and how threats are handled.

[1]The labels *P* and *A* are designed to support a dual reading: they correspond to plan-relaxed and action-relaxed, as used by Bercher (2021). Both interpretations are semantically equivalent.

The Complexity of Plan Existence

The PLAN EXISTENCE problem is the language

$$\text{PE} = \{\langle \Pi \rangle \mid \Pi \text{ is a planning instance with a solution plan}\}.$$

Each complexity result that follows concerns a variant of this problem, obtained by specifying a solution criterion. We assume a standard encoding function $\langle \cdot \rangle$ mapping objects to strings over $\{0, 1\}$. We begin by establishing a general basis for the hardness results. Hardness for unbounded-arity POCL planning problems (relaxed or not) stems from the known hardness of classical lifted planning.

Lemma 1 (Hardness Basis)**.** *Any classical lifted planning problem $\Pi = (\mathcal{P}, \mathcal{O}, \mathcal{A}, \mathcal{I}, \mathcal{G})$ can be reduced in polynomial time to a POCL planning problem $\Pi_{\text{POCL}} = (\mathcal{P}, \mathcal{O}, \mathcal{A}, Pl_{\mathcal{I},\mathcal{G}})$ with initial plan $Pl_{\mathcal{I},\mathcal{G}} = (\{init, goal\}, \{init \prec goal\}, \emptyset, \emptyset)$.*

Proof. Follows from Proposition 1. Since $init$ and $goal$ have no delete effects and $CL_I = \emptyset$, the reduction is unaffected by any relaxation variant. □

Since (non-relaxed) classical lifted planning is EXPSPACE–complete (Erol, Nau, and Subrahmanian 1991, Theorem 5.9) and delete-relaxed classical lifted planning is EXPTIME–complete (Erol, Nau, and Subrahmanian 1991, Theorem 5.7), Lemma 1 establishes that the corresponding POCL variants inherit these lower bounds. As hardness is thus established uniformly for most variants, the proofs that follow primarily provide membership arguments, with hardness argued explicitly where it is not immediate.

General Case: Non-Relaxed and P-Relaxed Plans

We begin with the complexity of POCL planning where the delete effects of newly added actions are not relaxed. This includes the standard, non-relaxed case as well as variants where only the pre-existing steps from the initial plan are relaxed.

Compared to classical problems, POCL problems pose additional restrictions as constraints in the initial plan need to be respected, rather than just achieving some goal from an initial state. However, as we now show, this does not increase the hardness of deciding POCL problems.

Theorem 1. *Let Π_{POCL} be a totally or partially ordered POCL planning problem. Deciding whether Π_{POCL} has a solution is EXPSPACE–complete.*

Proof. Let $\Pi_{\text{POCL}} = (\mathcal{P}, \mathcal{O}, \mathcal{A}, Pl_{\mathcal{I},\mathcal{G}})$. Since predicates and action schemas have unbounded arity, $|\mathcal{P}^{\mathcal{O}}|$ and $|\mathcal{A}^{\mathcal{O}}|$ are exponential in the size of $|\langle \Pi_{\text{POCL}} \rangle|$.

Since $Pl_{\mathcal{I},\mathcal{G}}$ may be partially lifted in the input, guess a grounding $Pl^{ground}_{\mathcal{I},\mathcal{G}}$ and consider the corresponding ground problem $\Pi^{ground}_{\text{POCL}} = (\mathcal{P}^{\mathcal{O}}, \mathcal{O}, \mathcal{A}^{\mathcal{O}}, Pl^{ground}_{\mathcal{I},\mathcal{G}})$. This instance is of size exponential in $|\langle \Pi_{\text{POCL}} \rangle|$. It is known that deciding whether a ground POCL planning problem has a solution can be done in PSPACE relative to the size of the ground instance (Bercher 2021, Theorem 1). Consequently, running the PSPACE algorithm on $\Pi^{ground}_{\text{POCL}}$ uses space exponential in the size of the original lifted instance.

It remains to show that a solution to any $\Pi^{ground}_{\text{POCL}}$ corresponds to a solution of Π_{POCL}. Since each ground problem fully instantiates all free variables while preserving all causal constraints, correctness follows from the fact that the lifted plan $Pl_{\mathcal{I},\mathcal{G}}$ encodes a disjunction over its possible ground instantiations. Conversely, any solution to $Pl_{\mathcal{I},\mathcal{G}}$ must correspond to a valid grounding due to the requirement that every free variable is eventually assigned an object from $\mathcal{O}$. Thus, deciding whether Π_{POCL} has a solution is equivalent to deciding whether the guessed $\Pi^{ground}_{\text{POCL}}$ is solvable.

As the grounding phase produces an exponentially larger instance and the ground instance can be decided in PSPACE relative to its size, the overall decision procedure uses exponential space relative to $|\langle\Pi_{\text{POCL}}\rangle|$. Thus, the problem is in NEXPSPACE = EXPSPACE (Savitch 1970). □

Relaxing only the pre-existing plan steps does not fundamentally alter the problem's structure, as the core challenge of reasoning over lifted, un-relaxed new actions remains. This leads to the following immediate corollary.

Corollary 1. *Let Π_{POCL} be a POCL planning problem. Deciding whether Π_{POCL} has a (P, ignore-links)- or (P, respect-links)-relaxed solution is EXPSPACE–complete.*

Proof. The complexity established in Theorem 1 is unaffected by relaxing only pre-existing plan steps. Membership in EXPSPACE holds because the underlying procedure of solving an exponentially large ground instance in PSPACE is unchanged; ignoring deletes for a subset of steps or adding a polynomial-time check for the 'respect-links' variant which filters out threatening actions does not alter the ground solver's complexity. Hardness also follows directly from the EXPSPACE–completeness of classical lifted planning via the reduction in Lemma 1, which constructs an initial plan where the steps have no delete effects and causal links are empty. Consequently, applying either P-relaxation to this instance leaves it unchanged, preserving EXPSPACE-hardness. □

A-Relaxed Plans

Corollary 1 demonstrates that delete-relaxing the initial plan is insufficient to drop the problem's complexity. Consequently, we now consider relaxations that ignore the delete effects of newly added plan steps.

Theorem 2. *Let Π_{POCL} be a POCL planning problem. Deciding whether Π_{POCL} has an A-relaxed solution is EXPTIME–complete.*

Proof. Let the initial plan in Π_{POCL} be $Pl_{\mathcal{I},\mathcal{G}} = (PS, \prec, CL, VC)$. We describe a decision procedure that runs in exponential time and determines whether there exists a linearisation of PS such that we can insert additional delete-relaxed action schemas to make the sequence applicable. A POCL solution can be obtained from such a sequence by inserting causal links, which can be done in polynomial time.

First, observe that we must consider all possible linearisations of the plan steps in PS, of which there are at most $|PS|!$ many. This factorial growth does not exceed exponential time. For sufficiently large n, we have:

$$n! \leq n^n = 2^{n \log n} \leq 2^{n^2}, \text{ so } O(n!) \subseteq O(2^{n^2}).$$

Hence, any algorithm that runs in time $O(n!)$ also runs in time $O(2^{n^2})$, and therefore is in $\mathsf{DTIME}(2^{n^2}) \subseteq$ EXPTIME. For each linearisation, check whether it violates any ordering constraints or causal links in $Pl_{\mathcal{I},\mathcal{G}}$; this can be done in polynomial time. If it is valid, we proceed to evaluate its feasibility using lifted relaxed planning graphs, as introduced by Ridder and Fox (2014).

We construct a relaxed planning graph for each consecutive pair of plan steps, starting from $init$. At each step, build the planning graph until a fixed point is reached. Since delete effects are absent, the number of predicates in fact layers strictly increase, and the number of unique lifted predicates is bounded by $O(\mathcal{P}^{\mathcal{O}})$, where $\mathcal{P}$ is the set of predicates and $\mathcal{O}$ the set of objects. Thus, each graph takes at most exponential time to construct. If the precondition of a plan step $ps_i \in PS$ is *not* included in the last fact layer of the graph constructed for that step, this linearisation is rejected. Otherwise, we simulate the step's application by deleting its delete effects and adding its add effects to the current state. This process is repeated for all $|PS| - 1$ transitions between plan steps. If the precondition of the final $goal$ step is contained in the last fact layer of the last graph, then the linearisation admits a supporting sequence of action schemas.

Overall, we build $|PS| - 1$ planning graphs, each requiring exponential time, within a loop that runs for $O(2^{n^2})$ iterations. Hence, the total time complexity is:

$$(|PS| - 1) \cdot O(2^n) \cdot O(2^{n^2}) = O\left(n \cdot 2^{n+n^2}\right)$$
$$\subseteq O\left(2^n \cdot 2^{n+n^2}\right) \subseteq O(2^{\text{poly}(n)}) \subseteq \text{EXPTIME}. \qquad \square$$

If the input plan is restricted to be totally ordered, it is clear that the complexity of the decision procedure remains EXPTIME–complete. While enumerating linearisations is unnecessary, the membership procedure's complexity is dominated by relaxed planning graph construction and evaluation. This aligns with the EXPTIME–completeness of the delete-relaxed classical setting (Erol, Nau, and Subrahmanian 1991, Theorem 5.7).

Corollary 2. *Let Π_{POCL} be a totally ordered POCL planning problem. Deciding whether Π_{POCL} has an A-relaxed solution is EXPTIME–complete.*

Interestingly, respecting causal links does not affect the complexity of deciding plan existence for A-relaxation.

Theorem 3. *Let Π_{POCL} be a POCL planning problem. Deciding whether Π_{POCL} has an A-relaxed solution respecting causal links is EXPTIME–complete.*

Proof. We adapt the decision procedure for Theorem 2. During the construction of lifted relaxed planning graphs for new (action-relaxed) steps, we must ensure that no chosen action schema threatens any pre-existing causal link. This check can be incorporated by filtering out threatening action schemas. This adds only a polynomial overhead to each step

of the graph construction, so the overall complexity remains in EXPTIME. □

As with the previous case, requiring total order does not affect the complexity of link-respecting A-relaxation.

Corollary 3. *Let Π_{POCL} be a totally-ordered POCL planning problem. Deciding whether Π_{POCL} has an A-relaxed solution respecting causal links is EXPTIME–complete.*

AP-Relaxed Plans

Having established that A-relaxation is EXPTIME–complete, regardless of whether the relaxation respects causal links, we now investigate whether additionally ignoring delete effects in the initial plan reduces the complexity.

Theorem 4. *Let Π_{POCL} be a POCL planning problem. Deciding whether Π_{POCL} has an AP-relaxed solution is EXPTIME–complete.*

Proof. We reduce AP-relaxed plan existence to A-relaxed plan existence (Theorem 2).

Let $Pl_{\mathcal{I},\mathcal{G}} = (PS, \prec, CL, VC)$. Since ϕ_A leaves pre-existing steps' delete effects intact, we must remove them syntactically: for each $(l, a) \in PS$, replace a by $a' = (pre(a), add(a), \emptyset)$, yielding Π'_{POCL}. This is polynomial and sound, since under (AP, ignore-links) the ignore-links policy unconditionally routes threat evaluation through ϕ_{AP}, so $\Delta = \emptyset$ for every step regardless of if $L \in CL_I$ – the original deletes of pre-existing steps are never consulted.

We claim Π_{POCL} admits an AP-relaxed solution iff Π'_{POCL} admits an A-relaxed solution. For the forward direction, any AP-relaxed solution to Π_{POCL} is an A-relaxed solution to Π'_{POCL}, since the deletes A-relaxation would otherwise leave intact are now $\emptyset$ by construction. For the backward direction, any A-relaxed solution to Π'_{POCL} is an AP-relaxed solution to Π_{POCL}, since the two instances differ only in that pre-existing steps' deletes have been zeroed. By Theorem 2, AP-relaxed plan existence is in EXPTIME. □

Finally, we show that respecting causal links under AP-relaxation also preserves EXPTIME–completeness.

Theorem 5. *Let Π_{POCL} be a POCL planning problem. Deciding whether Π_{POCL} has an AP-relaxed solution respecting causal links is EXPTIME–complete.*

Proof. We extend the decision procedure from Theorem 3 to allow delete relaxation not only for inserted action schemas, but also for initial plan steps. This affects only the semantics of action application, not the structure of the refinement procedure, so incurs no additional overhead. The problem remains in EXPTIME. □

These results show that in the lifted setting, both A-relaxation and AP-relaxation remain EXPTIME–complete, even when the input plan is totally ordered and regardless of whether causal links in the initial plan are respected. This contrasts sharply with the ground setting, where AP-relaxation is solvable in polynomial time, A-relaxation is NP–complete, and totally ordered A-relaxation drops back to polynomial time. Moreover, respecting causal links in the ground setting raises the complexity of both both A- and AP-relaxation to NP–complete (Bercher 2021). In the lifted case, however, neither more aggressive delete relaxation nor causal link protection yield complexity reductions: plan existence remains EXPTIME–complete across all variants.

Fixing Action Schemas in Advance

So far, we have considered planning problems where the input can be any POCL planning problem. We now consider the fixed-schema setting, where $\mathcal{A}$ is fixed in advance and only the objects and initial plan vary across instances. This setting is useful as it reflects the structure of POCL plan search. During search, the set of available action schemas is determined by the planning domain and remains constant; only the partial plan evolves as steps, orderings, and constraints are added. That is, each search node is a POCL planning problem with the same schema set but a larger partial plan. Understanding fixed-schema complexity therefore tells us precisely how hard it is to decide solvability at each node encountered during search, making these results particularly relevant to the analysis of POCL planners.

More broadly, fixing schemas is natural whenever the problem domain is fixed but many different instances must be solved. A warehouse robot, for instance, may have only three capabilities—moving to a location, picking up a package, and placing it down—yet face a different configuration of packages and destinations each day. With the schema set fixed, the number of ground actions becomes polynomial in the number of objects, enabling tighter complexity bounds.

Theorem 6. *Let Π_{POCL} be a POCL planning problem with $\mathcal{A}$ fixed in advance. Deciding whether Π_{POCL} has a solution is PSPACE–complete.*

Proof. Membership: By assumption $\mathcal{A}$ is fixed in advance, so there is a constant k that bounds the arity of every schema in $\mathcal{A}$. Every ground action is obtained by choosing at most k objects from $\mathcal{O}$ for the schema parameters, hence there are at most $|\mathcal{O}|^k$ ground actions. Because k is constant, this is polynomial in the input size.

We therefore can construct an equivalent *ground* POCL instance with only a polynomial increase in space by enumerating all ground actions and restricting the fluent set to those that occur in the initial state, the goal, or the preconditions/effects of these ground actions. We also need to guess a grounding of the initial plan. Ground POCL planning is in PSPACE (Bercher 2021, Thm 1); the grounded instance has a solution iff the original lifted instance has a solution, so the lifted problem with fixed schemas is also in PSPACE.

Hardness: Fixed-schema classical lifted planning is PSPACE–complete (Erol, Nau, and Subrahmanian 1991, Thm. 5.14), and reduces to our problem by Lemma 1. □

Like in the general case, relaxing pre-existing plan steps does not change the difficulty of reasoning over unrelaxed action schemas. The hard part remains the causal reasoning required when some actions still have delete effects.

Corollary 4. *Let Π_{POCL} be a POCL planning problem with $\mathcal{A}$ fixed in advance. Deciding whether Π_{POCL} has a*

(P, ignore-links)- or (P, respect-links)-relaxed solution is PSPACE–complete.

If we ignore the delete effects of newly added plan steps, then we get closer toward tractability. Erol, Nau, and Subrahmanian (1991, Thm 5.14) showed that lifted classical planning with a fixed delete-relaxed schema set is in P. Unfortunately, we still need to deal with grounding the initial plan. As the arity of the schemas are bounded, there are $|\mathcal{O}|^{k \cdot |PS|}$ possible groundings of the initial plan, which is exponential in the size of the input.

Theorem 7. *Let Π_{POCL} be a POCL planning problem with $\mathcal{A}$ fixed in advance, and let $\mathcal{R} \in \{A, AP\}$. Deciding whether Π_{POCL} has an $(\mathcal{R}$, ignore-links)- or $(\mathcal{R}$, respect-links)-relaxed solution is NP–complete.*

Proof. Membership: First, construct the set of ground actions as in Theorem 6. Then, non-deterministically guess a substitution σ that grounds all variables in the initial plan $Pl_{\mathcal{I},\mathcal{G}}$. For the (AP, ignore-links) case, we decide if the resulting ground problem has a solution. This is known to be in P (Bercher 2021, Theorem 2), so the check is polynomial.

For the three other cases, the corresponding ground problem is NP–complete. To show NP–membership for the lifted problem, we guess a linearisation of the initial plan's steps and check for contradictions with existing causal links. Then, we construct a delete-relaxed planning graph in each interval between plan steps. For (A, ignore-links) and (A, respect-links) variants, delete effects from initial plan steps are applied between graphs. For link-respecting variants, we only use actions that do not threaten causal links in the initial plan. If a step's preconditions do not appear, reject. Each graph is built in polynomial time (Hoffmann and Nebel 2001), and we construct at most $|PS| - 1$ of them. Hence, we obtain NP–membership.

Hardness: We reduce from graph 3-coloring, which is well known to be NP-complete (Lovász 1973). The constructed instance has no delete effects and no causal links, so all four relaxation variants coincide; we describe the reduction once and obtain NP-hardness for each.

Suppose $G = (V, E)$ is an arbitrary undirected graph. We construct a POCL planning problem $\Pi_{\text{POCL}} = (\mathcal{P}, \mathcal{O}, \mathcal{A}, Pl_{\mathcal{I},\mathcal{G}})$ as follows. Let $\mathcal{P} = \{\mathit{colored}(?c)\}$ and $\mathcal{O} = \{r, g, b\}$, representing the three available colors. Define a single action schema $\mathit{AssignColor}(?c) = (\{?c\}, \emptyset, \{\mathit{colored}(?c)\}, \emptyset)$ in $\mathcal{A}$. For each vertex $u \in V$, create a plan step $(u, \mathit{AssignColor}(?c_u))$ in PS, where $?c_u$ is a fresh variable associated with vertex u. For every edge $(u, v) \in E$, add the non-co-designation constraint $(?c_u \neq ?c_v)$ to VC. Set $\mathcal{I} = \emptyset$, $\mathcal{G} = \emptyset$, and $CL = \emptyset$, and include the standard *init* and *goal* steps with the usual ordering constraints. The construction is clearly polynomial in $|V| + |E|$.

Since each schema has no preconditions and no delete effects, every grounding of every action is trivially applicable, and causal threats cannot arise. Moreover, because $\mathcal{G} = \emptyset$, no causal links are required in a solution. Thus, the sole constraint on solvability is that the initial plan steps must admit a grounding that satisfies VC. A grounding σ assigns each variable $?c_u$ to some color in $\{r, g, b\}$. The constraint $?c_u \neq ?c_v$ is satisfied under σ if and only if $\sigma(?c_u) \neq \sigma(?c_v)$ – that is, adjacent vertices receive distinct colors. Therefore, a valid grounding exists precisely when G admits a 3-coloring. Finally, note that inserting additional steps cannot circumvent these constraints: any solution must still ground all steps from the initial plan while respecting VC. □

The root cause of hardness we have identified for the delete-relaxed fixed-$\mathcal{A}$ case has come from needing to ground the input plan. If we restrict the input plan to be ground, delete-relaxed fixed-$\mathcal{A}$ lifted problems reduce to their non-fixed ground POCL counterparts. Unlike in the unrestricted lifted setting, requiring input plans to be totally-ordered makes a difference in the delete-relaxed ground setting, and is sufficient to obtain membership in P (Bercher 2021). Most notably for our purposes, this is the case when planning from an initial state and goal description alone, aligning with the established bounds of Erol, Nau, and Subrahmanian (1991, Thm. 5.14):

Corollary 5. *Let Π_{POCL} be a POCL planning problem with $\mathcal{A}$ fixed in advance, and require the initial partial plan $Pl_{\mathcal{I},\mathcal{G}} = (\{init, goal\}, \{init \prec goal\}, \emptyset, \emptyset)$. Then, deciding if Π_{POCL} has an $(\mathcal{R}, \mathcal{L})$-relaxed solution is in P for each $\mathcal{R} \in \{A, AP\}$ and $\mathcal{L} \in \{$ignore-links, respect-links$\}$.*

Our membership arguments for the fixed-$\mathcal{A}$ setting rely on the fact that a fixed schema set implies a constant bound on arity, which keeps the number of ground actions polynomial. Consequently, all complexity results in this section generalise from a fixed $\mathcal{A}$ to any POCL problem where schema arity is bounded by a constant k. Specifically, the unrelaxed and P-relaxed cases remain PSPACE–complete, while the A- and AP-relaxed cases remain NP–complete, or in P if the initial plan is ground and totally-ordered.

Conclusion

We comprehensively examined the complexity of lifted POCL plan existence, focusing on delete-relaxation through a novel two-dimensional framework. Our results show that delete-relaxation on its own is insufficient to obtain tractability, only lowering complexity from EXPSPACE–complete to EXPTIME–complete. Tractability requires further restrictions: fixing action schemas in advance reduces the complexity to NP–complete for relaxed planning, and only when the initial plan is additionally ground and totally-ordered do all relaxations become solvable in polynomial time. In short, relaxing is hard, and only by staying grounded do we have a chance at tractability.

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

Bercher, P. 2021. A Closer Look at Causal Links: Complexity Results for Delete-Relaxation in Partial Order Causal Link (POCL) Planning. In *ICAPS 2021*, 36–45. AAAI Press.

Bercher, P.; Biundo, S.; Geier, T.; Hörnle, T.; Nothdurft, F.; Richter, F.; and Schattenberg, B. 2014. Plan, Repair, Execute, Explain – How Planning Helps to Assemble your Home Theater. In *ICAPS 2014*, 386–394. AAAI Press.

Bercher, P.; Haslum, P.; and Muise, C. 2024. A Survey on Plan Optimization. In *IJCAI 2024*, 7941–7950. IJCAI Organization.

Bercher, P.; Lin, S.; and Alford, R. 2022. Tight Bounds for Hybrid Planning. In *IJCAI 2022*, 4597–4605. IJCAI Organization.

Bercher, P.; and Olz, C. 2020. POP $\equiv$ POCL, Right? Complexity Results for Partial Order (Causal Link) Makespan Minimization. In *AAAI 2020*, 9785–9793. AAAI Press.

Bit-Monnot, A. 2023. Experimenting with Lifted Plan-Space Planning as Scheduling: Aries in the 2023 IPC. In *IPC 2023 (HTN Track): Planner and Domain Abstracts*, 7–9.

Bit-Monnot, A.; Ghallab, M.; Ingrand, F.; and Smith, D. E. 2020. FAPE: A Constraint-based Planner for Generative and Hierarchical Temporal Planning. arXiv preprint arXiv:2010.13121. arXiv:2010.13121.

Bit-Monnot, A.; and Godet, R. 2025. Towards Canonical and Minimal Solutions in a Constraint-Based Plan-Space Planner. In *ECAI 2025*, 4678–4685. IOS Press.

Blum, A. L.; and Furst, M. L. 1997. Fast Planning through Planning Graph Analysis. *AIJ*, 90(1): 281–300.

Bresina, J. L.; Jónsson, A. K.; Morris, P. H.; and Rajan, K. 2005. Activity Planning for the Mars Exploration Rovers. In *ICAPS 2005*, 40–49. AAAI Press.

Coles, A.; Coles, A.; Fox, M.; and Long, D. 2010. Forward-Chaining Partial-Order Planning. In *ICAPS 2010*, 42–49. AAAI Press.

Cook, S. A. 1971. The Complexity of Theorem-Proving Procedures. In *STOC 1971*, 151–158. Association for Computing Machinery.

Corrêa, A. B.; and De Giacomo, G. 2024. Lifted Planning: Recent Advances in Planning Using First-Order Representations. In *IJCAI 2024*, 8010–8019. IJCAI Organization.

Corrêa, A. B.; Francès, G.; Pommerening, F.; and Helmert, M. 2021. Delete-Relaxation Heuristics for Lifted Classical Planning. In *ICAPS 2021*, 94–102. AAAI Press.

Erol, K.; Nau, D. S.; and Subrahmanian, V. S. 1991. Complexity, Decidability and Undecidability Results for Domain-Independent Planning. Technical Report CS-TR-2797, UMIACS-TR-91-154, SRC-TR-91-96, Univ. Maryland.

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

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

Firsov, O.; Fiorino, H.; and Pellier, D. 2023. OptiPlan - a CSP-based partial order HTN planner. In *IPC 2023 (HTN Track): Planner and Domain Abstracts*, 10–11.

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

Kambhampati, S. 1997. Refinement Planning as a Unifying Framework for Plan Synthesis. *AI Mag.*, 18(2): 67–97.

Kambhampati, S.; and Kedar, S. 1994. A Unified Framework for Explanation-Based Generalization of Partially Ordered and Partially Instantiated Plans. *AIJ*, 67(1): 29–70.

Lauer, P.; Torralba, A.; Fišer, D.; Höller, D.; Wichlacz, J.; and Hoffmann, J. 2021. Polynomial-Time in PDDL Input Size: Making the Delete Relaxation Feasible for Lifted Planning. In *IJCAI 2021*, 4119–4126. IJCAI Organization.

Lovász, L. 1973. Coverings and colorings of hypergraphs. In *Proc. 4th Southeastern Conf. on Comb.*, 3–12. Utilitas Math.

McAllester, D.; and Rosenblitt, D. 1991. Systematic Nonlinear Planning. In *AAAI 1991*, 634–639. AAAI Press.

Minton, S.; Bresina, J.; and Drummond, M. 1994. Total-Order and Partial-Order Planning: A Comparative Analysis. *JAIR*, 2: 227–262.

Muise, C.; Beck, J. C.; and McIlraith, S. A. 2016. Optimal Partial-Order Plan Relaxation via MaxSAT. *JAIR*, 57: 113–149.

Oates, H.; and Bercher, P. 2026. Makespan Investigations of Sequential, Parallel, PO, and POCL Plans. In *AAAI 2026*. AAAI Press.

Pollack, M. E. 2002. Planning Technology for Intelligent Cognitive Orthotics. In *AIPS 2002*, 322–332. AAAI Press.

Ridder, B.; and Fox, M. 2014. Heuristic Evaluation Based on Lifted Relaxed Planning Graphs. In *ICAPS 2014*, 244–252. AAAI Press.

Savitch, W. J. 1970. Relationships between Nondeterministic and Deterministic Tape Complexities. *J. Comput. Syst. Sci*, 4(2): 177–192.

Siddiqui, F. H.; and Haslum, P. 2015. Continuing Plan Quality Optimisation. *JAIR*, 54: 369–435.

Siekmann, J. H. 1989. Unification Theory. *J. Symb. Comput.*, 7(3): 207–274.

Vidal, V.; and Geffner, H. 2006. Branching and Pruning: An Optimal Temporal POCL Planner Based on Constraint Programming. *AIJ*, 170(3): 298–335.

Weld, D. S. 1994. An Introduction to Least Commitment Planning. *AI Mag.*, 15(4): 27–27.

Weld, D. S. 2011. AAAI-10 Classic Paper Award: Systematic Nonlinear Planning A Commentary. *AI Mag.*, 32(1): 101–101.

Younes, H. L. S.; and Simmons, R. G. 2002. On the Role of Ground Actions in Refinement Planning. In *AIPS 2002*, 54–61. AAAI Press.

Younes, H. L. S.; and Simmons, R. G. 2003. VHPOP: Versatile Heuristic Partial Order Planner. *JAIR*, 20: 405–430.