Delete Relaxation with Axioms

Travis Rivera Petit, Simon Dold, David Speck, Malte Helmert

University of Basel, Basel, Switzerland
{travis.riverapetit,simon.dold@unibas.ch,davidjakob.speck,malte.helmert}@unibas.ch

Abstract

Existing approaches to delete-relaxation heuristics in classical planning with axioms are based on task transformations, three-valued logic, or answer set programming. We introduce three additional, novel methods that follow the task-transformation approach, internally compiling axioms away. We show that existing task-transformation approaches yield unsafe heuristics and conjecture a well-founded bridge to the three-valued logic approach, resulting in a dominance hierarchy over existing and new methods. Experimental evaluations show that our delete-relaxation methods perform favorably in overall coverage compared to the existing approaches, for both optimal planning with $h^{\max}$ and satisficing planning with h^{FF}.

1 Introduction

Classical planning studies deterministic, fully observable, single-agent problems, aiming to transform an initial state into a goal state via a sequence of operators. Axioms are an important modeling tool, enabling compact representations of complex properties such as recursive dependencies and reachability conditions (e.g., Thiébaux, Hoffmann, and Nebel 2005). Beyond improving modeling convenience, axioms strictly increase the expressiveness of planning formalisms (Thiébaux, Hoffmann, and Nebel 2005; Hoffmann and Edelkamp 2005; Grundke, Röger, and Helmert 2024).

Several works study planning with axioms, including heuristic search (Thiébaux, Hoffmann, and Nebel 2005; Ivankovic and Haslum 2015), symbolic search (Speck et al. 2019, 2025), and SAT-based planning (Behnke, Speck, and Gnad 2025). We focus on heuristic search, a common approach to classical planning, with delete relaxation being widely used to derive informative heuristics by ignoring negative effects (Bonet and Geffner 1999; Hoffmann and Nebel 2001). Ivankovic and Haslum (2015) relate delete relaxation to monotonic relaxation, compute heuristics using three-valued logic h^+_{3VL} and answer set programming h^+_{ASP}, and show that h^+_{3VL} is dominated by h^+_{ASP}.

We introduce three admissible delete-relaxation heuristics, h^+_{NA}, h^+_{CA}, and h^+_{UR}, that support axioms via compilation leveraging zero-cost operators. Moreover, we identify issues in existing systems (Thiébaux, Hoffmann, and Nebel 2005; Helmert 2006) and prove that the underlying approach is incomplete by showing that their heuristics are not safe.

$$h^+_{\text{NA}} \quad \leq \quad h^+_{\text{CA}} \quad \leq \quad h^+_{\text{UR}} \quad \stackrel{?}{=} \quad h^+_{\text{3VL}} \quad \leq \quad h^+_{\text{ASP}}$$

Figure 1: A hierarchy of delete-relaxation heuristics highlighting a conjectured bridge between different approaches.

Furthermore, we extend the domination hierachy by Ivankovic and Haslum (2015) with our delete-relaxation heuristics and conjecture the equality of h^+_{UR} and h^+_{3VL} (Figure 1). This conjecture is also strongly supported by our empirical findings. Finally, our experiments show that our novel heuristics perform favorably for optimal and satisficing planning compared to the state of the art.

2 Background

For a set of propositional variables $\mathcal{V}$, we denote by $\mathcal{L}(\mathcal{V})$ the set of all formulae that can be obtained using the connectives $\wedge$ (conjunction), $\vee$ (disjunction), and $\neg$ (negation). A literal ℓ is either a variable $v \in \mathcal{V}$ or its negation $\neg v$. With $\mathcal{C}(\mathcal{V}) \subseteq \mathcal{L}(\mathcal{V})$, we refer to the set of conjunctions of literals as well as $\top$ (the formula that always evaluates to $\mathbf{T}$) and $\bot$ (the formula that always evaluates to $\mathbf{F}$). We assume familiarity with the standard rules of propositional logic (Kleine Büning and Lettmann 1999).

We consider classical planning tasks with axioms (e.g., Thiébaux, Hoffmann, and Nebel 2005) and arbitrary conditions and conditional effects over propositional variables (e.g., Rintanen 2008).

Definition 1 (Planning Task)**.** A *planning task* is a tuple $\Pi = \langle \mathcal{V}, \mathcal{D}, \mathcal{O}, \mathcal{A}, \mathcal{I}, \gamma \rangle$. $\mathcal{V}$ is a finite set of *basic variables* and $\mathcal{D}$ is a finite set of *derived variables* with $\mathcal{V} \cap \mathcal{D} = \emptyset$. $\mathcal{O}$ is a finite set of *operators*, where each *operator* $o = \langle \text{pre}(o), \text{eff}(o), \text{cost}(o) \rangle$ consists of a *precondition* $\text{pre}(o) \in \mathcal{L}(\mathcal{V} \cup \mathcal{D})$, an *effect* $\text{eff}(o)$, and a *cost* $\text{cost}(o) \in \mathbb{R}_{\geq 0}$. Effects are defined inductively: 1) $\top$ is the *empty effect*, 2) v is a *positive atomic effect* and $\neg v$ is a *negative atomic effect* for $v \in \mathcal{V}$, 3) $(e \wedge e')$ is a *conjunctive effect* if e, e' are effects, and 4) $(\chi \rhd e)$ is a *conditional effect* if e is an effect and $\chi \in \mathcal{L}(\mathcal{V} \cup \mathcal{D})$ is a formula, called the *effect condition*.We sometimes omit parentheses and write $e \wedge e'$ and $\chi \rhd e$ instead of $(e \wedge e')$ and $(\chi \rhd e)$. $\mathcal{A}$ is a finite set of *axioms*. Each *axiom* a is of the form $d \leftarrow \varphi$, where $\text{head}(a) = d \in \mathcal{D}$, is called the *head* of a, and $\text{body}(a) = \varphi \in \mathcal{C}(\mathcal{V} \cup \mathcal{D})$, is called the

body of a. A *(basic) state* $s : \mathcal{V} \to \{\mathbf{T}, \mathbf{F}\}$ is a complete truth assignment over the basic variables, while an *extended state* $s' : \mathcal{V} \cup \mathcal{D} \to \{\mathbf{T}, \mathbf{F}\}$ is a complete truth assignment over both the basic and derived variables. We denote by $\mathcal{S}(\Pi)$ the set of all basic states. The *initial state* $\mathcal{I} \in \mathcal{S}(\Pi)$ is a basic state, while the *goal condition* $\gamma \in \mathcal{L}(\mathcal{V} \cup \mathcal{D})$ is a formula.

Axioms specify how derived variables are inferred from basic variables. We only consider planning tasks whose sets of axioms are *stratifiable*. This property ensures a unique and efficient derivation (Thiébaux, Hoffmann, and Nebel 2005). Let $\text{level} : \mathcal{D} \to \mathbb{N}_0$ be a function assigning a level to each derived variable. It induces a partition $\{\mathcal{A}_0, \ldots, \mathcal{A}_m\}$ of $\mathcal{A}$ into strata where $m = \max\{\text{level}(d) \mid d \in \mathcal{D}\}$ and $\mathcal{A}_i = \{d \leftarrow \varphi \mid \text{level}(d) = i\}$ for $i = 0, \ldots, m$. We call level a stratification of $\mathcal{A}$ if for every $a \in \mathcal{A}$ with head d and for every derived variable d', the following conditions hold: 1) if d' appears in the body of a then $\text{level}(d') \leq \text{level}(d)$, and 2) if d' appears negated (i.e., as $\neg d'$) in the body of a, then $\text{level}(d') < \text{level}(d)$. The set $\mathcal{A}$ is stratifiable if such a stratification exists. We define $\mathcal{A}_{\leq n} = \bigcup_{i \leq n} \mathcal{A}_i$.

Given a basic state $s \in \mathcal{S}(\Pi)$, the extended state $\mathcal{A}[s]$ is uniquely defined by the standard stratified semantics (Apt, Blair, and Walker 1988; Thiébaux, Hoffmann, and Nebel 2005). Algorithm 1 computes the values of derived variables using negation-as-failure. The values of the basic variables remain unchanged, i.e., $\mathcal{A}[s](v) = s(v)$ for all $v \in \mathcal{V}$. Initially, each derived variable $d \in \mathcal{D}$ is assigned the default value $\mathbf{F}$. Then, a fixed-point computation is performed for each axiom layer $\mathcal{A}_i$, in order, to derive the values of the derived variables $\mathcal{D}_i$.

Algorithm 1: Axiom evaluation (Helmert 2006)

Input: State s and $\{\mathcal{A}_0, \ldots, \mathcal{A}_m\}$ induced by level and $\mathcal{A}$
Output: Extended state $\mathcal{A}[s]$

for all $v \in \mathcal{V} \cup \mathcal{D}$ **do**
  $\mathcal{A}[s](v) := \begin{cases} s(v) & \text{if } v \in \mathcal{V} \\ \mathbf{F} & \text{otherwise} \end{cases}$
end for
for $i = 0$ **to** m **do**
  while there exists $d \leftarrow \varphi \in \mathcal{A}_i$ such that $\mathcal{A}[s](d) = \mathbf{F}$ and $\mathcal{A}[s] \models \varphi$ **do**
    $\mathcal{A}[s](d) := \mathbf{T}$
  end while
end for
return $\mathcal{A}[s]$

We define implication criteria, adapted from Rintanen (2008), which specify when an effect yields a literal.

Definition 2 (Implication Criterion)**.** Let ℓ be an atomic effect and e be an effect. The *implication criterion* $\text{impc}(e, \ell)$, under which the atomic effect ℓ is triggered by the effect e, is defined as: 1) $\text{impc}(\top, \ell) = \bot$, 2) $\text{impc}(\ell, \ell) = \top$, 3) $\text{impc}(\ell', \ell) = \bot$, if ℓ' is an atomic effect with $\ell' \neq \ell$, 4) $\text{impc}(e \wedge e', \ell) = \text{impc}(e, \ell) \vee \text{impc}(e', \ell)$, and 5) $\text{impc}(\chi \rhd e, \ell) = \chi \wedge \text{impc}(e, \ell)$.

We often refer to the *conditions* of a planning task Π, which are all the formulae appearing in Π. That is, each precondition, effect condition, goal condition, or axiom body is a condition of Π.

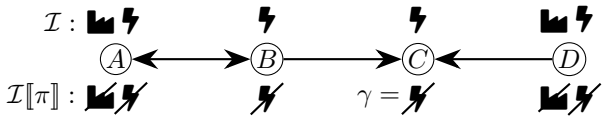

Figure 2: Visualization of the running Example 1.

Using implication criteria, we define effect and operator application, and the resulting successor states, as follows.

Definition 3 (Application of Effects and Operators)**.** Let $s \in \mathcal{S}(\Pi)$ be a state, let e be an effect, and let $\mathcal{A}$ be a set of axioms. Applying e in s yields the *resulting state* $s[\![e]\!]$, defined by

$$s[\![e]\!](v) = \begin{cases} \mathbf{T} & \text{if } \mathcal{A}[s] \models \text{impc}(e, v), \\ \mathbf{F} & \text{if } \mathcal{A}[s] \models \text{impc}(e, \neg v) \wedge \neg\, \text{impc}(e, v), \\ s(v) & \text{otherwise.} \end{cases}$$

The resulting state of applying an operator o in s is defined as $s[\![o]\!] = s[\![\text{eff}(o)]\!]$. An operator o is *applicable* in s if $\mathcal{A}[s] \models \text{pre}(o)$.

We say that two effects e and e' are *equivalent* if $s[\![e]\!] = s[\![e']\!]$ for all states $s \in \mathcal{S}(\Pi)$. Examples of equivalent effects include $e = v$ and $e' = \neg v \rhd v$ as well as $e = a \rhd (b \rhd c)$ and $e' = (a \wedge b) \rhd c$. Given the semantics of a planning task, we define plans and their associated costs as follows.

Definition 4 (Plan, Optimal Plan, Plan Cost)**.** A sequence of operators $\pi = \langle o_1, \ldots, o_n \rangle$ is applicable in a state $s \in \mathcal{S}(\Pi)$ if o_i is applicable in $s[\![o_1, \ldots, o_{i-1}]\!]$ for all $i = 1, \ldots, n$, where $s[\![o_1, \ldots, o_i]\!] = s[\![o_1]\!] \cdots [\![o_i]\!]$. We denote the resulting state by $s[\![\pi]\!]$. If $s[\![\pi]\!] \models \gamma$, and each member of π is in $\mathcal{O}$, then π is an *s-plan*. If s is the initial state of Π, then π is a *plan* for Π. The *cost* of π is $\sum_{i=1}^{n} \text{cost}(o_i)$. A plan π is *optimal* for a state s if no cheaper s-plan exists. The objective of satisficing planning is to find a plan for a task Π, whereas optimal planning aims to find an optimal plan.

The following example illustrates planning tasks with axioms.

Example 1. Consider a planning task Π^{Expl} modeling an energy network with four nodes $N = \{A, B, C, D\}$ (Figure 2). Nodes A and D have generators. Energy flow is represented by derived variables $\mathcal{D} = \{\textit{eflow-}n \mid n \in N\}$, and generators at the left and right nodes by basic variables $\mathcal{V} = \{\textit{on-}A, \textit{on-}D\}$. There are two operators with cost 1, *toggle-A* and *toggle-D*, both with precondition $\top$ and effects $(\textit{on-}A \rhd \neg \textit{on-}A) \wedge (\neg \textit{on-}A \rhd \textit{on-}A)$ and $(\textit{on-}D \rhd \neg \textit{on-}D) \wedge (\neg \textit{on-}D \rhd \textit{on-}D)$. Energy flow is defined by the following axioms:

$a_1 := \textit{eflow-}A \leftarrow \textit{on-}A$ $\quad$ $a_2 := \textit{eflow-}D \leftarrow \textit{on-}D$
$a_3 := \textit{eflow-}A \leftarrow \textit{eflow-}B$ $\quad$ $a_4 := \textit{eflow-}B \leftarrow \textit{eflow-}A$
$a_5 := \textit{eflow-}C \leftarrow \textit{eflow-}B$ $\quad$ $a_6 := \textit{eflow-}C \leftarrow \textit{eflow-}D$

Since all axiom bodies are positive, a single stratum suffices, i.e., $\text{level}(d) = 0$ for all $d \in \mathcal{D}$. In the initial state $\mathcal{I}$, both $\textit{on-}A$ and $\textit{on-}D$ hold, and the goal is to turn off the energy at

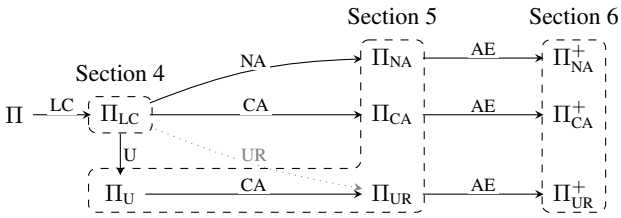

Figure 3: Illustration of the steps for computing delete-relaxation heuristics for planning tasks with axioms. We first perform a **l**iteral-**c**onjunction transformation (LC) and then choose one of the following delete relaxations: **n**egation **a**pproximation (NA), **c**ycle **a**pproximation (CA), **u**nrolling **r**elaxation (UR). Computing UR requires an additional **u**nrolling step U. Next, all axioms are compiled by **a**xiom **e**limination (AE) into zero-cost operators, yielding the relaxed tasks Π^+_{NA}, Π^+_{CA}, and Π^+_{UR} on which we can directly compute delete-relaxation heuristics such as $h^{\max}$, h^{add}, or h^{FF}.

node C for maintenance, i.e., $\gamma = \neg \mathit{eflow}\text{-}C$. The sequence $\pi = \langle \mathit{toggle}\text{-}A, \mathit{toggle}\text{-}D \rangle$ is an optimal plan.

We focus on heuristic search (Pearl 1984), which is one of the most successful approaches to optimal and satisficing classical planning. A *heuristic* h_Π of a planning task Π is a function $h_\Pi : \mathcal{S}(\Pi) \to \mathbb{R}_{\geq 0} \cup \{\infty\}$ that estimates the cost of an s-plan for each state $s \in \mathcal{S}(\Pi)$.[1] The *perfect heuristic* h^* maps each state s to the cost of an optimal s-plan, or to ∞ if no such plan exists. A heuristic h is called *safe* if $h(s)$ is finite for every state $s \in \mathcal{S}(\Pi)$ for which an s-plan exists. A heuristic h is called *admissible* if it never overestimates the optimal cost of reaching a goal state, i.e., $h(s) \leq h^*(s)$ for all $s \in \mathcal{S}(\Pi)$. We say that an admissible heuristic h_1 *dominates* another admissible heuristic h_2, written $h_1 \geq h_2$, if it provides at least as accurate cost estimates as h_2 for every state, i.e., $h_1(s) \geq h_2(s)$ for all $s \in \mathcal{S}(\Pi)$.

3 Delete Relaxation

We describe delete-relaxation heuristics as a sequence of task transformations, as shown in Figure 3. It allows us to obtain a delete-relaxed planning task relaxed via negation, cycle or unrolling relaxation. This framework yields delete-relaxed planning tasks based on negation approximation, cycle approximation, or unrolling relaxation. Standard classical planning methods can then be applied to these transformed tasks to compute heuristics such as h^+ and $h^{\max}$. The described pipeline and underlying idea are the same as in Thiébaux, Hoffmann, and Nebel (2005) and Helmert (2006), with the difference being our novel transformations NA, CA and UR, which avoid issues present in earlier approaches that can lead to unsafe heuristics (as we will prove later). The new tasks consist of new sets of variables, operators and axioms. We define task transformations and the property of admissibility as follows.

[1] We sometimes write h instead of h_Π when clear in context.

Definition 5 (Task transformation, Optimal-Cost Task Transformation, Admissible Task Transformation)**.** A *task transformation* f is a function which takes as input a planning task $\Pi = \langle \mathcal{V}, \mathcal{D}, \mathcal{O}, \mathcal{A}, \mathcal{I}, \gamma \rangle$ and produces a new planning task $f(\Pi) = \Pi' = \langle \mathcal{V}', \mathcal{D}', \mathcal{O}', \mathcal{A}', \mathcal{I}', \gamma' \rangle$. We call the task transformation *optimal-cost preserving* iff ${h^*}_{\Pi'}(\mathcal{I}') = {h^*}_{\Pi}(\mathcal{I})$ and *admissible* iff ${h^*}_{\Pi'}(\mathcal{I}') \leq {h^*}_{\Pi}(\mathcal{I})$.

Admissible transformations allow us to propagate admissible heuristics from $f(\Pi)$ to Π by setting $h_\Pi(s) := h_{f(\Pi)}(f(s))$ when considering s as the initial state. We will argue that each of the task transformations in Sections 5.1, 5.2, and 5.3 are admissible, while the task transformations from Sections 4 and 6 are optimal-cost preserving. This gives rise to the delete-relaxation heuristics h^+_{NA}, h^+_{CA} and h^+_{UR}.

4 Literal-Conjunctive Form

This section introduces a transformation of planning tasks with complex conditions, effects, and axioms into a normal form where all conditions are conjunctions of literals. The idea is to flatten nested effects and to replace complex conditions with derived variables and axioms that evaluate them.

Definition 6 (Flat normal form)**.** An effect is *simple* if it is either an atomic effect or an effect of the form $\chi \rhd e$, where e is an atomic effect. An effect is *flat* if it is the conjunction of zero or more simple effects, and none of the simple effects include the same atomic effect. An operator o is *flat* if $\text{eff}(o)$ is flat. A planning task is *flat* if all of its operators are flat.

We now introduce the conjunctive condition normal form. It characterizes flat planning tasks whose conditions are conjunctions of literals.

Definition 7 (Literal-Conjunctive Form)**.** A planning task Π is in *literal-conjunctive (LC)* form if it is flat and all conditions are conjunctions of literals.

Having defined the literal-conjunctive form for a planning task, we now describe how to transform a given planning task into an equivalent one in this form. We start by defining a transformation flat that produces an equivalent flat effect.

Definition 8. For an effect e, the *flattening* of e is $\text{flat}(e) = \bigwedge_{v \in \mathcal{V}} \big(\text{impc}(e, v) \rhd v\big) \wedge \bigwedge_{v \in \mathcal{V}} \big(\text{impc}(e, \neg v) \rhd \neg v\big)$.

A flattened effect $\text{flat}(e)$ is a conjunctive effect consisting of one conditional effect for each literal ℓ. We can faithfully replace each effect e in an operator of Π by $\text{flat}(e)$. This is because if e makes ℓ true, then $\text{impc}(e, \ell)$ holds in the current state and thus the conditional effect $\text{impc}(e, \ell) \rhd \ell$ triggers. The same argument holds in the other direction.

Proposition 1. Each effect e is equivalent to $\text{flat}(e)$ and $\text{flat}(e)$ is flat.

Next, we turn to the transformation of conditions into conjunctions of literals. We present two alternative approaches for this purpose.

Approach One: DNF-based. Conditions are transformed into disjunctive normal form (DNF). If an action precondition has multiple terms, each term defines a separate operator. If an effect condition has multiple terms, each term defines a separate conditional effect. If an axiom body has multiple

terms, each term defines a separate axiom with the same head. This approach to handling conditions is used in Fast Downward (Helmert 2006, 2009) and can incur an exponential blow-up due to the size of the DNF. While complex goal conditions could be handled similarly by introducing a new variable and zero-cost dummy operators, Fast Downward avoids such operators. If the goal is not a simple conjunction, it is replaced by a new variable and multiple axioms that derive it, which are then handled by the DNF-based approach.

Approach Two: Tseitin-based. As a second approach, we propose a Tseitin-based transformation. Each condition φ is converted into negation normal form (NNF). For each innermost disjunction $\psi = \bigvee_{i=1}^{n} \psi_i$, we introduce a new derived variable d_ψ, replace ψ by d_ψ, and add axioms $d_\psi \leftarrow \psi_i$. This process is repeated until no disjunctions remain. With $\mathcal{A}_\varphi$ and $\mathcal{D}_\varphi$ we denote the set of axioms and derived variables produced by this process. The transformation is specified in Algorithm 2 and mimics the well-known *Tseitin transformation* (Tseitin 1968).

Algorithm 2: Tseitin-based transformation

Input: φ
Output: $\varphi^{\text{Tseitin}}, \mathcal{A}_\varphi, \mathcal{D}_\varphi$
1: $\varphi^{\text{Tseitin}} := \text{NNF}(\varphi)$
2: $\mathcal{A}_\varphi := \emptyset$
3: $\mathcal{D}_\varphi := \emptyset$
4: **while** there is a subformula ψ of φ^{Tseitin} of the form $\psi = \psi_1 \vee \cdots \vee \psi_n$ with $n \geq 2$, and each ψ_i is a conjunction of literals **do**
5: $\quad \mathcal{D}_\varphi := \mathcal{D}_\varphi \cup \{d_\psi\}$
6: $\quad \mathcal{A}_\varphi := \mathcal{A}_\varphi \cup \{\, d_\psi \leftarrow \psi_i \mid 1 \leq i \leq n \,\}$
7: $\quad \varphi^{\text{Tseitin}} := \text{replace}(d_\psi, \psi, \varphi^{\text{Tseitin}})$
8: **end while**
9: **return** $\langle \varphi^{\text{Tseitin}}, \mathcal{A}_\varphi, \mathcal{D}_\varphi \rangle$

Example 2. Let $\varphi = x \wedge \neg(y \wedge \neg z)$ be a precondition of an operator o. The *DNF-based approach* transforms φ into DNF: $(x \wedge \neg y) \vee (x \wedge z)$, and then produces two copies o_1, o_2 of operator o with $\text{pre}(o_1) = x \wedge \neg y$ and $\text{pre}(o_2) = x \wedge z$. The *Tseitin-based approach* transforms φ into negation normal form: $x \wedge (\neg y \vee z)$. Then it introduces $d_{\neg y \vee z}$ with axioms $d_{\neg y \vee z} \leftarrow \neg y$ and $d_{\neg y \vee z} \leftarrow z$, and replaces the condition by $x \wedge d_{\neg y \vee z}$.

The *LC transformation* executes first the flat transformation on the effect of each operator, then either the Tseitin-based or the DNF-based transformation on each condition. We can transform any planning task Π into an equivalent task $\text{LC}(\Pi) = \Pi_{\text{LC}}$ in *LC form* using the LC transformation. These transformations are by construction optimal-cost preserving. The Tseitin-based approach guarantees a polynomial transformation, whereas the DNF-based approach can incur an exponential blow-up. As we will see in the experiments, in most tasks the condition structure is simple enough that the naive DNF-based approach is feasible. However, there are also domains with more complex structures where a naive transformation to DNF is infeasible.

5 Delete-Free Forms

This section introduces positive planning tasks and presents three transformations that map planning tasks in literal-conjunctive form to a delete-free form, which forms the basis of delete relaxation. Delete relaxation was originally introduced for STRIPS tasks, where it is obtained by removing all delete effects from operators (Bonet and Geffner 2001). We generalize this notion to planning tasks with axioms.

Definition 9 (Positive Formula, Delete-Free Effect, Delete-Free Planning Task)**.** A logical formula is *positive* if it does not contain the negation symbol $\neg$. An effect is *delete-free* if its effect conditions are positive and its atomic effects do not contain $\neg$. A planning task is *delete-free* if all its conditions are positive and all its effects are delete-free.

In all three transformations we make use of the following idea. For each variable $v \in \mathcal{V} \cup \mathcal{D}$, we introduce a new variable $\hat{v}$, called the *antagonist* of v. The transformations NA, CA and UR differ by the conditions we impose to infer the truth values of the antagonists derived variables. Previous work also introduces antagonist variables at the lifted level (Gazen and Knoblock 1997), which is used to compile away negated preconditions and negated effect conditions. In contrast, we extend this idea to also eliminate negated occurrences in axiom bodies. If X is a set of variables we denote with $\hat{X}$ the set $\{\hat{x} \mid x \in X\}$. We introduce a transformation which we use to transform all conditions into their positive counterparts.

Definition 10 (Antagonist Transformation)**.** Let $\varphi \in \mathcal{L}(\mathcal{V} \cup \mathcal{D})$ be a formula in negation normal form. The *antagonist transformation* $T(\varphi) \in \mathcal{L}(\mathcal{V} \cup \hat{\mathcal{V}} \cup \mathcal{D} \cup \hat{\mathcal{D}})$ of φ is obtained by replacing each negative occurrence $\neg x$ of a variable by $\hat{x}$.

We extend the previous definitions to include effects, operators, axioms, plans, and states. If $e = \bigwedge_i (\chi_i \rhd \ell_i)$ is a flat effect, then $T(e)$ replaces each $\chi_i \rhd v$ with $T(\chi_i) \rhd v$ and each $\chi_i \rhd \neg v$ with $T(\chi_i) \rhd \hat{v}$. For an operator o, set $T(o) = \langle T(\text{pre}(o)), T(\text{eff}(o)), \text{cost}(o) \rangle$. If $a = d \leftarrow \varphi$ is an axiom then $T(a) = d \leftarrow T(\varphi)$, and if $\pi = \langle o_1, \ldots o_n \rangle$, then $T(\pi) = \langle T(o_1), \ldots, T(o_n) \rangle$. The antagonist transformation $T(s)$ of a state s is the map $\mathcal{V} \cup \hat{\mathcal{V}} \to \{\mathbf{F}, \mathbf{T}\}$ with $T(s)(v) = s(v)$ and $T(s)(\hat{v}) = \neg s(v)$[2] for each $v \in \mathcal{V}$.

5.1 Negation Approximation

The negation approximation is a transformation that uses the antagonist transformation to make all conditions positive and adds trivial axiom rules $\hat{d} \leftarrow \top$ for each derived variable d.

Definition 11 (Negation Approximation)**.** Let $\Pi = \langle \mathcal{V}, \mathcal{D}, \mathcal{O}, \mathcal{A}, \mathcal{I}, \gamma \rangle$ be a planning task in LC form. The *negation approximation* (NA) of Π is the planning task given by $\mathcal{V}_{\text{NA}} = \mathcal{V} \cup \hat{\mathcal{V}}$, $\mathcal{D}_{\text{NA}} = \mathcal{D} \cup \hat{\mathcal{D}}$, $\mathcal{O}_{\text{NA}} = \{T(o) \mid o \in \mathcal{O}\}$, $\mathcal{A}_{\text{NA}} = \{T(a) \mid a \in \mathcal{A}\} \cup \{\hat{d} \leftarrow \top \mid d \in \mathcal{D}\}$, $\mathcal{I}_{\text{NA}} = T(\mathcal{I})$, and $\gamma_{\text{NA}} = T(\gamma)$.

Theorem 1. NA is an admissible transformation for all tasks in LC form.

[2] We use the convention $\neg\mathbf{T} = \mathbf{F}$ and $\neg\mathbf{F} = \mathbf{T}$.

Proof sketch. Let Π be a planning task in LC form and $\pi = \langle o_1, \dots, o_n \rangle$ be a plan for Π inducing state sequence $\langle s_0, \dots, s_n \rangle$. We argue that $T(\pi)$ is a plan for Π_{NA}. Let $\langle s'_0, \dots, s'_n \rangle$ be the states in Π_{NA} induced by the effects of the operators in $T(\pi)$. For each variable $v \in \mathcal{V} \cup \mathcal{D}$ it holds by construction that $\mathcal{A}[s_i](v) = \mathbf{T}$ implies $\mathcal{A}_{\text{NA}}[s'_i](v) = \mathbf{T}$ for all $0 \leq i \leq n$. Negative literals in conditions are transformed into antagonist variables $\hat{v}$, which are always true in Π_{NA} by construction. Thus, all transformed negative conditions are satisfied. Hence, for each $1 \leq i \leq n$ operator $T(o_i)$ is applicable in s'_{i-1}, and $s'_n \models T(\gamma)$.

5.2 Cycle Approximation

The negation approximation NA assigns $\hat{d} \leftarrow \top$ for all derived variables, resulting in a coarse approximation of the original task. We improve this construction by observing that, for cyclically independent variables, $\hat{d}$ can instead be handled directly while preserving admissibility.

Definition 12 (Cyclic Dependency)**.** A variable d is *cyclically independent* if there exists a stratification of $\mathcal{A}$ in which $\text{level}(d) \neq \text{level}(d')$ for all $d' \in \mathcal{D} \setminus \{d\}$. A variable that is not cyclically independent is *cyclically dependent*.

The previous definition induces a partition of $\mathcal{D}$ into cyclically dependent and independent variables, denoted by Dep and Ind. For a derived variable d, let $\text{unsat}(d) = \bigwedge_{a \in \mathcal{A}: \text{head}(a) = d} \neg \text{body}(a)$ be a formula which characterizes when no axiom with head d has a satisfied body. The following lemma shows that $\text{unsat}(d)$ can be inferred directly for cyclically independent variables. Intuitively, the lemma shows that since d is cyclically independent, its value depends only on variables from lower strata. Therefore, its falsity implies that none of its defining axioms can be satisfied.

Lemma 1. Let Π be a planning task in LC form, $s \in \mathcal{S}(\Pi)$ and $d \in Ind$. Then $\mathcal{A}[s](d) = \mathbf{F}$ iff $\mathcal{A}_{<\text{level}(d)}[s] \models \text{unsat}(d)$.

Proof. If $\mathcal{A}[s](d) = \mathbf{F}$, then in particular $\mathcal{A}_{<\text{level}(d)}[s](d) = \mathbf{F}$ so $\mathcal{A}_{<\text{level}(d)}[s] \models \text{unsat}(d)$. We now consider the case $\mathcal{A}[s](d) = \mathbf{T}$. Choose a stratification of $\mathcal{A}$ such that $\text{level}(d) \neq \text{level}(d')$ for all $d' \in \mathcal{D} \setminus \{d\}$, which exists since d is cyclically independent. Note that since d is in a single stratum, $\mathcal{A}[s](d) = \mathbf{T}$ iff there exists an axiom $a \in \mathcal{A}_{\text{level}(d)}$ with $\text{head}(a) = d$ and $\mathcal{A}_{<\text{level}(d)}[s] \models \text{body}(a)$. In this case, all derived variables v_i appearing in $\text{body}(a)$ satisfy $\text{level}(v_i) < \text{level}(d)$. In particular, we get $\mathcal{A}_{<\text{level}(d)}[s](v_i) = \mathcal{A}_{\leq\text{level}(d)}[s](v_i)$ and therefore $\mathcal{A}_{<\text{level}(d)}[s] \models \text{body}(a)$. □

Note that the lemma does not hold if d is cyclically dependent. This observation is not addressed in Helmert (2006), which leads to unsafe behavior in the delete relaxation of the Fast Downward heuristic h^+ (see Theorem 5).

We reuse the Tseitin-based transformation from Section 4 to convert $\text{unsat}(d)$ into a conjunction of literals. The derived variables of the resulting task include $\mathcal{D}$, their antagonists, the auxiliary variables introduced by the Tseitin transformation for cyclically independent variables, and their antagonists. We consider four kinds of axioms: (i) *transformed axioms*, obtained by applying the antagonist transformation to the original axioms; (ii) *antagonist axioms for independent variables*, where $\text{unsat}(d)$ is encoded using the Tseitin transformation; (iii) *auxiliary axioms*, introduced by the Tseitin transformation; and (iv) *antagonist axioms for dependent variables*, which we define as $\hat{d} \leftarrow \top$. This encodes the falsity of cyclically independent variables using their antagonist variables, while cyclically dependent variables are approximated as in the NA transformation.

Definition 13 (Cycle Approximation)**.** Let $\Pi = \langle \mathcal{V}, \mathcal{D}, \mathcal{O}, \mathcal{A}, \mathcal{I}, \gamma \rangle$ be a planning task in LC form. The *cycle approximation* (CA) of Π is the planning task given by $\mathcal{V}_{\text{CA}} = \mathcal{V} \cup \hat{\mathcal{V}}$, $\mathcal{D}_{\text{CA}} = \mathcal{D} \cup \hat{\mathcal{D}} \cup \bigcup_{d \in Ind} \mathcal{D}_{\text{unsat}(d)} \cup \bigcup_{d \in Ind} \widehat{\mathcal{D}_{\text{unsat}(d)}}$, $\mathcal{O}_{\text{CA}} = \{T(o) \mid o \in \mathcal{O}\}$, $\mathcal{A}_{\text{CA}}$ is the union of the sets *Transformed-*$\mathcal{A}$, *Antagonist-indep*, *Aux*, and *Antagonist-dep*, where *Transformed-*$\mathcal{A} = \{T(a) \mid a \in \mathcal{A}\}$, *Antagonist-indep* $= \{\hat{d} \leftarrow T(\text{unsat}(d)^{\text{Tseitin}}) \mid d \in Ind\}$, *Aux* $= \{T(a) \mid a \in \mathcal{A}_\varphi$ for some $d \leftarrow \varphi \in \mathcal{A}$ with $d \in Ind\}$, and *Antagonist-dep* $= \{\hat{d} \leftarrow \top \mid d \in Dep\}$. Finally, the initial state is $\mathcal{I}_{\text{CA}} = T(\mathcal{I})$, and the goal $\gamma_{\text{CA}} = T(\gamma)$.

Similarly to NA, cyclically dependent antagonist derived variables $\hat{d}$ are assigned trivial axioms $\hat{d} \leftarrow \top$. In contrast to NA, cyclically independent derived variables $\hat{d}$ can be false in the derived state $\mathcal{A}_{\text{CA}}[T(s)]$. They satisfy the following property.

Proposition 2. Let Π be a planning task in LC form, $s \in \mathcal{S}(\Pi)$ and $d \in Ind$. If $\mathcal{A}_{<\text{level}(d)}[s] \models \text{unsat}(d)$ then $\mathcal{A}_{\text{CA}}[T(s)](\hat{d}) = \mathbf{T}$. Otherwise, if $\mathcal{A}_{<\text{level}(d)}[s] \not\models \text{unsat}(d)$ then $\mathcal{A}_{\text{CA}}[T(s)](d) = \mathbf{T}$.

Combining this property, Lemma 1, which characterizes the correct handling of falsity for cyclically independent variables, and the argument of Theorem 1, we obtain the following result.

Theorem 2. CA is an admissible transformation for all tasks in LC form.

5.3 Unrolling

The cycle approximation fails to capture the behavior of cyclically dependent derived variables. To address this, we first eliminate cyclic dependencies using *time unrolling* (e.g., Behnke, Speck, and Gnad 2025).

For a derived variable d, let $\text{cyc}(d)$ denote the set of derived variables $d' \in \mathcal{D}$ such that $\text{level}(d) = \text{level}(d')$ holds for every stratification of $\mathcal{A}$. Note that $d \in \text{cyc}(d)$, so $\text{cyc}(d)$ is never empty. The transformation introduces time-indexed copies $d^1, \dots, d^{|\text{cyc}(d)|}$ for each derived variable $d \in \mathcal{D}$. Intuitively, d^t represents the value of d after t rounds of axiom evaluation affecting variables in $\text{cyc}(d)$. Each axiom a is replaced by a set of axioms $\text{unroll}(a)$, and for each derived variable d, an additional linking axiom $d \leftarrow d^{|\text{cyc}(d)|}$ is introduced.

In the following definition, $\text{replace}(x \in S, x', \varphi)$ denotes the formula obtained by replacing each occurrence of a variable $x \in S$ in φ by x'.

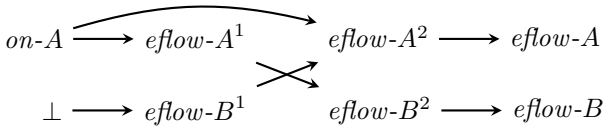

Figure 4: Axioms created for the cyclically dependent variables of the task Π^{Expl} by the unrolling approach U.

Definition 14 (Unrolling)**.** Let $\Pi = \langle \mathcal{V}, \mathcal{D}, \mathcal{O}, \mathcal{A}, \mathcal{I}, \gamma \rangle$ be a planning task in LC form. Let $a = d \leftarrow \varphi$ be an axiom. The unrolling of a is given by

$$\begin{aligned}\text{unroll}(a) = \{d^1 \leftarrow \text{replace}(x \in \text{cyc}(d), \bot, \varphi)\} \cup \\ \{d^t \leftarrow \text{replace}(x \in \text{cyc}(d), x^{t-1}, \varphi) \\ \mid 2 \leq t \leq |\,\text{cyc}(d)|\}.\end{aligned}$$

The unrolling (U) of Π is the task given by $\mathcal{V}_{\text{U}} = \mathcal{V}$, $\mathcal{D}_{\text{U}} = \mathcal{D} \cup \{d^t \mid d \in \mathcal{D},\ 1 \leq t \leq |\,\text{cyc}(d)|\}$, $\mathcal{O}_{\text{U}} = \mathcal{O}$, $\mathcal{A}_{\text{U}} = \bigcup_{a \in \mathcal{A}} \text{unroll}(a) \cup \{d \leftarrow d^{|\,\text{cyc}(d)|} \mid d \in \mathcal{D}\}$, $\mathcal{I}_{\text{U}} = \mathcal{I}$, and $\gamma_{\text{U}} = \gamma$.

The result of unrolling a task in LC form is again a task in LC form and we can apply the cycle approximation to it.

Definition 15 (Unrolling Relaxation)**.** The unrolling relaxation UR of a planning task Π in LC form is $\text{UR}(\Pi) = \text{CA}(\text{U}(\Pi))$.

Since U is optimal-cost preserving and CA is admissible, we obtain the following result.

Theorem 3. The task transformation UR is an admissible transformation for all tasks in LC form.

5.4 Examples

Consider the example task Π^{Expl} (Example 1).

Negation Approximation The negation approximation $\text{NA}(\Pi^{\text{Expl}})$ extends the set of basic and derived variables with $\mathcal{V}_{\text{NA}} = \mathcal{V} \cup \{\widehat{on\text{-}A}, \widehat{on\text{-}B}\}$ and $\mathcal{D}_{\text{NA}} = \mathcal{D} \cup \{\widehat{eflow\text{-}n} \mid n \in \{A, B, C, D\}\}$. It consists of the antagonist-transformed set of axioms in $\mathcal{A}$ and the trivial antagonist axioms $\widehat{eflow\text{-}A} \leftarrow \top$, $\widehat{eflow\text{-}B} \leftarrow \top$, $\widehat{eflow\text{-}C} \leftarrow \top$ and $\widehat{eflow\text{-}D} \leftarrow \top$. Its operators are $T(\textit{toggle-A}) = \langle \top, on\text{-}A \rhd \widehat{on\text{-}A} \wedge \widehat{on\text{-}A} \rhd on\text{-}A \rangle$ and $T(\textit{toggle-D}) = \langle \top, on\text{-}D \rhd \widehat{on\text{-}D} \wedge \widehat{on\text{-}D} \rhd on\text{-}D \rangle$. The initial state is $\mathcal{I}_{\text{NA}} = \{on\text{-}A \mapsto \mathbf{T}, \widehat{on\text{-}A} \mapsto \mathbf{F}, on\text{-}B \mapsto \mathbf{T}, \widehat{on\text{-}B} \mapsto \mathbf{F}\}$, and its goal $\gamma_{\text{NA}} = \widehat{eflow\text{-}C}$.

Cycle Approximation The cycle approximation $\text{CA}(\Pi^{\text{Expl}})$ differs from $\text{NA}(\Pi^{\text{Expl}})$ only in its antagonist axioms. Note that $eflow\text{-}A \in \text{cyc}(eflow\text{-}B)$. Accordingly, the antagonist axioms become $\widehat{eflow\text{-}A} \leftarrow \top$, $\widehat{eflow\text{-}B} \leftarrow \top$, $\widehat{eflow\text{-}C} \leftarrow \widehat{eflow\text{-}B} \wedge \widehat{eflow\text{-}D}$, and $\widehat{eflow\text{-}D} \leftarrow \widehat{on\text{-}D}$.

Unrolling Relaxation The unrolling relaxation addresses the issue that the axioms $\widehat{eflow\text{-}A} \leftarrow \top$ and $\widehat{eflow\text{-}B} \leftarrow \top$ do not capture the semantics of $\neg eflow\text{-}A$ and $\neg eflow\text{-}B$ in Π^{Expl}. This is achieved by first constructing $\text{U}(\Pi^{\text{Expl}})$, which compiles away the cyclic dependency between $eflow\text{-}A$ and $eflow\text{-}B$ via the unrolling shown in Figure 4. Finally, the unrolling relaxation computes CA on top of U. Here, instead of the trivial axioms $\widehat{eflow\text{-}A} \leftarrow \top$, $\widehat{eflow\text{-}B} \leftarrow \top$, we obtain $\widehat{eflow\text{-}A} \leftarrow \widehat{eflow\text{-}A^2}$ and $\widehat{eflow\text{-}B} \leftarrow \widehat{eflow\text{-}B^2}$.

The costs of the optimal plans are

$$0 = h^*_{\Pi^{\text{Expl}}_{\text{NA}}}(\mathcal{I}_{\text{NA}}) \ < \ h^*_{\Pi^{\text{Expl}}_{\text{CA}}}(\mathcal{I}_{\text{CA}}) \ < \ h^*_{\Pi^{\text{Expl}}_{\text{UR}}}(\mathcal{I}_{\text{UR}}) = 2.$$

6 Axiom Elimination

In the final step of the pipeline, we take a negation-free planning task and compile its axioms into zero-cost operators. That is, each axiom $d \leftarrow \varphi \in \mathcal{A}$ is converted into an operator $\langle \varphi, d, 0 \rangle$. The idea of compiling axioms into operators was introduced by Thiébaux, Hoffmann, and Nebel (2005) in the context of the h^{FF} heuristic, where it was used as an admissible approximation. In contrast, we show that for negation-free planning tasks, this transformation is an optimal-cost preserving transformation. Axioms differ from zero-cost operators in two major ways. First, axioms follow the negation-as-failure semantics, where on each state, all derived variables are set to **F** and then have their truth values updated according to Algorithm 1. Second, axioms must be applied in a fixed-point computation, while the application of operators is optional. We show that for negation-free planning tasks, this makes no difference. After compiling axioms as zero-cost operators we may compute any delete-relaxed heuristic for tasks without axioms in the usual way.

Definition 16 (Axiom Elimination)**.** Let $\Pi = \langle \mathcal{V}, \mathcal{D}, \mathcal{O}, \mathcal{A}, \mathcal{I}, \gamma \rangle$ be a negation-free planning task. The *axiom elimination* (AE) of Π is the task given by $\mathcal{V}_{\text{AE}} = \mathcal{V} \cup \mathcal{D}$, $\mathcal{D}_{\text{AE}} = \emptyset$, $\mathcal{O}_{\text{AE}} = \mathcal{O} \cup \{\langle \text{body}(a), \text{head}(a), 0 \rangle \mid a \in \mathcal{A}\}$, $\mathcal{A}_{\text{AE}} = \emptyset$, $\mathcal{I}_{\text{AE}}(v) = \mathcal{I}(v)$ for $v \in \mathcal{V}$, $\mathcal{I}_{\text{AE}}(d) = \mathbf{F}$ for $d \in \mathcal{D}$, and $\gamma_{\text{AE}} = \gamma$.

Theorem 4 (Axiom Elimination Theorem)**.** The axiom elimination is an optimal-cost preserving transformation for all negation-free tasks.

Proof sketch. Fix a negation-free planning task $\Pi = \langle \mathcal{V}, \mathcal{D}, \mathcal{O}, \mathcal{A}, \mathcal{I}, \gamma \rangle$. For a state $s \in \mathcal{S}(\Pi)$ let $t(s) = \langle a_1, \ldots, a_m \rangle$ be the trace of axioms in Algorithm 1. That is, $t(s)$ is the sequence of compiled axioms (zero-cost operators) that fire when using this algorithm. We show that $h^*_{\Pi}(\mathcal{I}) \geq h^*_{\Pi_{\text{AE}}}(\mathcal{I}_{\text{AE}})$ and that $h^*_{\Pi}(\mathcal{I}) \leq h^*_{\Pi_{\text{AE}}}(\mathcal{I}_{\text{AE}})$.

- Proof that $h^*_{\Pi}(\mathcal{I}) \geq h^*_{\Pi_{\text{AE}}}(\mathcal{I}_{\text{AE}})$:
  Let $\pi = \langle o_1, \ldots, o_n \rangle$ be an optimal plan in Π. We construct an $\mathcal{I}_{\text{AE}}$-plan π_{AE} with $\text{cost}(\pi_{\text{AE}}) = \text{cost}(\pi)$. The plan π induces a list of states $\langle s_0, s_1, \ldots, s_n \rangle$ where $s_{i+1} = s_i[\![o_{i+1}]\!]$ for $i = 0, \ldots, n-1$. By the definition of t and Π_{AE} we immediately get that $\pi_{\text{AE}} := \langle t(s_0), o_1, t(s_1) \ldots o_n, t(s_n) \rangle$ is a plan for Π_{AE}. Since each each axiom is treated as a zero-cost operator, we get $\text{cost}(\Pi_{\text{AE}}) = \text{cost}(\pi)$ as desired.
- Proof that $h^*_{\Pi}(\mathcal{I}) \leq h^*_{\Pi_{\text{AE}}}(\mathcal{I}_{\text{AE}})$:
  Let π_{AE} be an optimal plan in Π_{AE}. We construct a plan π with $\text{cost}(\pi) = \text{cost}(\pi_{\text{AE}})$. First, we define an axiom section S as a (possibly empty) list of compiled axioms. This allows us to split π_{AE} into $\pi_{\text{AE}} = \langle S_0, o_1, S_1, \ldots, o_n, S_n \rangle$. Note that each member of S_0 is also in $t(s_0)$.

```
(define (domain ffx-unsolvable)
  (:requirements :strips
     :derived-predicates
     :negative-preconditions)
  (:predicates (v) (p) (q) (r))

  ; not v -> p <-> q <-> r
  (:derived (p) (not (v)))
  (:derived (p) (q))
  (:derived (q) (p))
  (:derived (r) (q))
  (:derived (q) (r))

  (:action set-v
   :effect (v)
  )
)
```

```
(define (problem P)
  (:domain ffx-unsolvable)
  (:init (not (v)))
  (:goal (not (r)))
)
```

Figure 5: PDDL files illustrating a case where $h^{\text{FF}}_{\text{X}}(\mathcal{I}) = \infty$ although a plan exists.

Therefore, since Π is negation-free, the sequence $\langle t(s_0), o_1, S_1, o_2, S_2, \ldots, o_n, S_n\rangle$ is also an optimal plan for Π_{AE}. With a straightforward induction argument, it follows that $\langle t(s_0), o_1, t(s_0[\![o_1]\!]), \ldots o_n, t(s_0[\![o_1, \ldots, o_n]\!])\rangle$ is an optimal plan for Π_{AE}. In particular, this means that $\pi := \langle o_1, \ldots o_n\rangle$ is a plan with the same cost as π_{AE}.

7 Comparison to Other Approaches

We conjecture that the unrolling relaxation heuristic h^+_{UR} yields the same heuristic function as the three-valued heuristic h^+_{3VL} of Ivankovic and Haslum (2015). While initially surprising, the intuitive explanation is as follows: in h^+_{3VL}, starting from a relaxed basic state, the values of the derived variables can take three possible values: true, false, and unknown. If a derived variable d is true in the h^+_{3VL} approach, this corresponds in our transformation to d being true and its antagonist $\hat{d}$ being false. Similarly, if d is false in the h^+_{3VL} approach, it means in our framework that d is false and its antagonist $\hat{d}$ is true. Finally, if d is unknown in h^+_{3VL}, this corresponds to both d and $\hat{d}$ being true.

Conjecture 1. The heuristics h^+_{UR} and h^+_{3VL} are equivalent, i.e., $h^+_{\text{UR}}(s) = h^+_{\text{3VL}}(s)$ for each task Π and state $s \in \mathcal{S}(\Pi)$.

We now show that other delete relaxations from the literature are incomparable to ours, as they can produce unsafe heuristic estimates. Specifically, these relaxations can assign an infinite cost to a state s even when an s-plan exists. This observation extends directly to the practically computed versions of these heuristics, namely $h^{\max}$ and h^{FF}.

Theorem 5. The delete relaxation presented in Helmert (2006) yields heuristics that are not safe.

Proof. In Helmert (2006), a derived variable d in the delete relaxation can take on negative values (here represented by $\hat{d}$) based on the transitions in the extended domain transition graph.[3] Such transitions are induced by the negation of the axiom bodies that make d true. This is identical to the approaches we presented, but it does not address cyclic dependencies by unrolling or trivially setting them to true. Thus, when considering the planning task Π^{Expl} from Example 1, we obtain for the goal variable of the delete relaxation $\widehat{\mathit{eflow}\text{-}C}$ the following transition to make it true: $\widehat{\mathit{eflow}\text{-}C} \leftarrow \widehat{\mathit{eflow}\text{-}B} \wedge \widehat{\mathit{eflow}\text{-}D}$. For $\widehat{\mathit{eflow}\text{-}B}$ to be true, we have the transition $\widehat{\mathit{eflow}\text{-}B} \leftarrow \widehat{\mathit{eflow}\text{-}A}$. So, for the goal $\widehat{\mathit{eflow}\text{-}C}$ in the delete relaxation, $\widehat{\mathit{eflow}\text{-}A}$ has to be true, which has the transition $\widehat{\mathit{eflow}\text{-}A} \leftarrow \widehat{\mathit{on}\text{-}A} \wedge \widehat{\mathit{eflow}\text{-}B}$. We see that for $\widehat{\mathit{eflow}\text{-}B}$ to become true, $\widehat{\mathit{eflow}\text{-}A}$ must be true, and vice versa: for $\widehat{\mathit{eflow}\text{-}A}$ to become true, $\widehat{\mathit{eflow}\text{-}B}$ must be true. However, in the initial state, where $\mathit{eflow}\text{-}A$ and $\mathit{eflow}\text{-}B$ are true, both $\widehat{\mathit{eflow}\text{-}A}$ and $\widehat{\mathit{eflow}\text{-}B}$ are false. Thus, since $\widehat{\mathit{eflow}\text{-}A}$ and $\widehat{\mathit{eflow}\text{-}B}$ are dependent on each other, they can never become true in the delete relaxation. Hence, the delete relaxation has no plan, although the original task has one, from which it directly follows that h^+ based on this approach is not safe. □

A similar unsafe behavior occurs in the FFX planner (Thiébaux, Hoffmann, and Nebel 2005). In the FFX planner, antagonist variables are introduced, and the task is transformed into positive normal form by adding axioms that infer the truth values of the antagonist-derived variables. However, the implemented method breaks down when a variable has more than one other variable that is cyclically dependent on it. In particular, FFX can report tasks to be unsolvable because the h^{FF}_{X} heuristic returns ∞ for states where $h^*(s)$ is in fact finite. One such PDDL instance is depicted in Figure 5.

8 Experiments

We implemented our delete relaxation approaches in Fast Downward (Helmert 2006), computing $h^{\max}$ (Bonet and Geffner 2001) for optimal and h^{FF} (Hoffmann and Nebel 2001) for satisficing planning. Our benchmark set includes domains with axioms and conditional effects (e.g., Thiébaux and Cordier 2001; Kominis and Geffner 2015; Miura and Fukunaga 2017; Borgwardt et al. 2022; Bofill et al. 2023), including IPC 2004 domains (Hoffmann and Edelkamp 2005). All experiments use a time limit of 30 minutes and 3.5 GB of memory per run. Our code and data are available online (Petit et al. 2026).

We want to highlight that, beyond heuristic search, other planning approaches support axioms as well: symbolic search (Speck et al. 2019; Speck, Seipp, and Torralba 2025), SAT-based planning (Behnke, Speck, and Gnad 2025), and ASP-based planning (Miura and Fukunaga 2017; Dimopoulos et al.

[3] For consistency, we present the result using antagonist variables for the delete relaxation, although in Helmert (2006) this is handled by allowing variables to have multiple values (true or false). Conceptually, this makes no difference for our argument.

Domain	Tseitin LC				DNF LC				IH 2015	
	h^{blind}	$h^{\text{max}}_{\text{NA}}$	$h^{\text{max}}_{\text{CA}}$	$h^{\text{max}}_{\text{UR}}$	h^{blind}	$h^{\text{max}}_{\text{NA}}$	$h^{\text{max}}_{\text{CA}}$	$h^{\text{max}}_{\text{UR}}$	$h^{\text{max}}_{\text{3VL}}$	$h^{\text{max}}_{\text{ASP}}$
Blocker (9)	**7**	**7**	6	6	**7**	**7**	**7**	6	6	5
Blocks (105)	**54**	**54**	**54**	**54**	**54**	**54**	**54**	**54**	52	33
Collab (3)	**3**	**3**	**3**	**3**	1	1	1	1	1	1
GED (266)	**16**	**16**	**16**	**16**	**16**	**16**	**16**	**16**	**16**	12
Ghosh-Etal (107)	14	14	14	14	**16**	15	15	15	7	2
Grid-Axioms (5)	1	**3**	**3**	**3**	1	**3**	**3**	**3**	2	1
Horn-DL (271)	226	225	223	**230**	**230**	228	227	**230**	208	98
Miconic-Axioms (150)	55	55	55	55	**60**	**60**	**60**	**60**	55	45
Muddy (12)	**9**	**9**	**9**	**9**	2	2	2	2	2	2
Opt-Telegraphs (48)	**2**	**2**	**2**	**2**	**2**	**2**	**2**	**2**	**2**	1
Philosophers (48)	**5**	**5**	**5**	**5**	**5**	**5**	**5**	**5**	**5**	3
PSR (100)	48	48	48	43	**50**	49	49	48	46	24
Snowman (51)	27	**29**	**29**	**29**	26	**29**	28	**29**	26	6
Social-Planning (2)	**2**	**2**	**2**	**2**	**2**	**2**	**2**	**2**	**2**	1
Sokoban-Axioms (30)	24	**27**	**27**	**27**	24	**27**	**27**	**27**	21	5
Sum (5)	3	**4**	**4**	**4**	1	1	1	1	1	1
Word-Rooms (5)	**5**	**5**	**5**	**5**	2	2	2	2	2	0
Sum (1217)	501	**507**	505	**507**	499	503	501	506	454	240

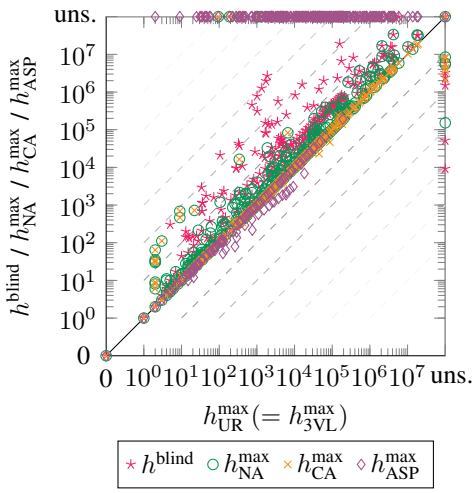

Figure 6: The number of solved tasks (left) and a comparison of the necessary node expansions (right) for A* using the h^{blind} and h^{max} heuristics computed with different delete relaxations. In the table, we report results using both a Tseitin-based and a DNF-based variant for transforming a planning task into literal-conjunctive form, which is required for our heuristics $h^{\text{max}}_{\text{NA}}$ (negation approximation), $h^{\text{max}}_{\text{CA}}$ (cycle approximation), and $h^{\text{max}}_{\text{UR}}$ (unrolling). The heuristics $h^{\text{max}}_{\text{3VL}}$ (three-valued logic) and $h^{\text{max}}_{\text{ASP}}$ (answer-set programming) are taken from Ivankovic and Haslum (2015) and executed using their planner.

2019). While these approaches show strong empirical results, here we compare against other heuristic search methods to evaluate whether we improve over existing approaches within the same paradigm.

We use the following optimizations for CA and UR. First, when computing each $\text{unsat}(\varphi)^{\text{Tseitin}}$, we make use of formula caching to reduce the number of new axioms. Second, to speed up the computation of cyclic dependency checks, we stratify tasks using the *max* strategy.[4] Under this stratification, cyclic dependencies can be identified by $d' \in \text{cyc}(d)$ iff $\text{level}(d) = \text{level}(d')$. Third, antagonist derived variables and antagonist axioms are only introduced when needed.

8.1 Optimal Planning

We use A* (Hart, Nilsson, and Raphael 1968) with h^{max} (Bonet and Geffner 1999) under different delete relaxations: $h^{\text{max}}_{\text{NA}}$, $h^{\text{max}}_{\text{CA}}$, and $h^{\text{max}}_{\text{UR}}$. As a baseline, h^{blind} returns the minimal operator cost in non-goal states and zero otherwise. We compare against $h^{\text{max}}_{\text{3VL}}$ and $h^{\text{max}}_{\text{ASP}}$ (Ivankovic and Haslum 2015), and evaluate both Tseitin and DNF-based transformations into LC form.[5]

Figure 6 reports coverage. The table shows the number of solved tasks for each domain (with some rows aggregating domains by topic), as well as the overall coverage of the different approaches. We first observe that the Tseitin-based transformation into LC form generally performs slightly better than the DNF-based transformation. However, in the Miconic-Axioms domain, all DNF-based variants solve more tasks than their Tseitin-based counterparts. In this case, the conditions are simple enough to be handled efficiently without additional encoding, whereas the introduction of many derived variables incurs unnecessary overhead. Overall, $h^{\text{max}}_{\text{UR}}$ and $h^{\text{max}}_{\text{CA}}$ achieve the best coverage. However, in some domains (e.g., PSR), their computational overhead outweighs their benefits, and simpler heuristics, including h^{blind}, perform better. Finally, the plot in Figure 6 shows that the necessary node expansions of $h^{\text{max}}_{\text{UR}}$ and $h^{\text{max}}_{\text{3VL}}$ are identical, which supports our conjecture of equivalence and the established dominance relations between the various delete relaxations.

[4] In Fast Downward, this is enabled with the command `--translate-options --layer-strategy max`.

[5] The implementation of (Ivankovic and Haslum 2015) relies on a version of Fast Downward with a translator bug. We patched this issue to ensure a fair comparison.

Domain	Lazy Greedy			+ Pref. Ops.			FFX
	$h^{\text{FF}}_{\text{NA}}$	$h^{\text{FF}}_{\text{CA}}$	$h^{\text{FF}}_{\text{UR}}$	$h^{\text{FF}}_{\text{NA}}$	$h^{\text{FF}}_{\text{CA}}$	$h^{\text{FF}}_{\text{UR}}$	h^{FF}_{X}
Sum (900)	640	692	687	674	718	**720**	400

Table 1: The number of solved tasks for lazy greedy search (with and without preferred operators) using the h^{FF} heuristic computed with our delete relaxations (we only report results for the Tseitin-based literal-conjunctive form). The h^{FF}_{X} configuration runs the FFX planner, which implements enforced hill-climbing with the h^{FF}_{X} heuristic.

8.2 Satisficing Planning

We use lazy greedy search with h^{FF} (Hoffmann and Nebel 2001), both with and without preferred operators (Röger and Helmert 2010), and evaluate $h^{\text{FF}}_{\text{NA}}$, $h^{\text{FF}}_{\text{CA}}$, and $h^{\text{FF}}_{\text{UR}}$ under unit costs. Results are compared against the FF-X planner using

h^{FF}_{X} (Thiébaux, Hoffmann, and Nebel 2005). The Snowman and GED domains are excluded, since FF-X does not support action costs, to ensure a fair comparison.

Table 1 shows coverage using Tseitin-based transformation into LC form. The use of preferred operators significantly increases the number of solved tasks. FFX encounters translator/parser issues in one domain on verifying systems (Ghosh, Dasgupta, and Ramesh 2015), so these results should be interpreted with some caution. Nevertheless, FFX has the highest coverage in the optical telegraph domain, while our new methods yield the highest coverage in all other domains.

9 Conclusions

We introduced three delete relaxations based on task transformations for classical planning with axioms. Theoretically, we conjecture that one matches the optimal delete-relaxation heuristic of Ivankovic and Haslum (2015), yielding a hierarchy that relates our approaches to existing methods based on three-valued logic and answer-set programming. Empirically, the resulting $h^{\max}$ and h^{FF} variants achieve strong performance in both optimal and satisficing planning. One advantage of our approach is that it is realized entirely as a task transformation that compiles away axioms. As a result, once the transformation has been applied, the standard delete-relaxation heuristics such as h^{+}, $h^{\max}$, h^{add}, and h^{FF} become immediately applicable without requiring modifications. For future work, we plan to study other heuristics, such as landmark heuristics (e.g., Helmert and Domshlak 2009; Richter and Westphal 2010; Büchner et al. 2023), and extend delete-relaxation heuristics to lifted planning with axioms.

10 Acknowledgments

This work was funded by the Swiss National Science Foundation (SNSF) as part of the project "Unifying the Theory and Algorithms of Factored State-Space Search" (UTA).

References

Apt, K. R.; Blair, H. A.; and Walker, A. 1988. Towards a Theory of Declarative Knowledge. In *Foundations of Deductive Databases and Logic Programming*, 89–148. Morgan Kaufmann.

Behnke, G.; Speck, D.; and Gnad, D. 2025. AxSAT – Bringing Axioms to SAT Planning. In Casini, G.; Dundua, B.; and Kutsia, T., eds., *Logics in Artificial Intelligence*, 77–93. Springer Nature Switzerland.

Bofill, M.; Borralleras, C.; Espasa, J.; Martín, G.; Patow, G.; and Villaret, M. 2023. A Good Snowman is Hard to Plan. arXiv:2310.01471 [cs.AI].

Bonet, B.; and Geffner, H. 1999. Planning as Heuristic Search: New Results. In *Proc. ECP 1999*, 360–372.

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

Borgwardt, S.; Hoffmann, J.; Kovtunova, A.; Krötzsch, M.; Nebel, B.; and Steinmetz, M. 2022. Expressivity of Planning with Horn Description Logic Ontologies. In *Proc. AAAI 2022*, 5503–5511.

Büchner, C.; Eriksson, S.; Keller, T.; and Helmert, M. 2023. Landmark Progression in Heuristic Search. In *Proc. ICAPS 2023*, 70–79.

Dimopoulos, Y.; Gebser, M.; Lühne, P.; Romero, J.; and Schaub, T. 2019. plasp 3: Towards Effective ASP Planning. *Theory Pract. Log. Program.*, 19(3): 477–504.

Gazen, B. C.; and Knoblock, C. A. 1997. Combining the Expressivity of UCPOP with the Efficiency of Graphplan. In *Proc. ECP 1997*, 221–233.

Ghosh, K.; Dasgupta, P.; and Ramesh, S. 2015. Automated Planning as an Early Verification Tool for Distributed Control. *Journal of Automated Reasoning*, 54(1): 31–68.

Grundke, C.; Röger, G.; and Helmert, M. 2024. Formal Representations of Classical Planning Domains. In *Proc. ICAPS 2024*, 239–248.

Hart, P. E.; Nilsson, N. J.; and Raphael, B. 1968. A Formal Basis for the Heuristic Determination of Minimum Cost Paths. *IEEE Transactions on Systems Science and Cybernetics*, 4(2): 100–107.

Helmert, M. 2006. The Fast Downward Planning System. 26: 191–246.

Helmert, M. 2009. Concise Finite-Domain Representations for PDDL Planning Tasks. 173: 503–535.

Helmert, M.; and Domshlak, C. 2009. Landmarks, Critical Paths and Abstractions: What's the Difference Anyway? In *Proc. ICAPS 2009*, 162–169.

Hoffmann, J.; and Edelkamp, S. 2005. The Deterministic Part of IPC-4: An Overview. 24: 519–579.

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

Ivankovic, F.; and Haslum, P. 2015. Optimal Planning with Axioms. In *Proc. IJCAI 2015*, 1580–1586.

Kleine Büning, H.; and Lettmann, T. 1999. *Propositional Logic: Deduction and Algorithms*, volume 48 of *Cambridge Tracts in Theoretical Computer Science*. Cambridge University Press.

Kominis, F.; and Geffner, H. 2015. Beliefs in Multiagent Planning: From One Agent to Many. In *Proc. ICAPS 2015*, 147–155.

Miura, S.; and Fukunaga, A. 2017. Automatic Extraction of Axioms for Planning. In *Proc. ICAPS 2017*, 218–227.

Pearl, J. 1984. *Heuristics: Intelligent Search Strategies for Computer Problem Solving*. Addison-Wesley.

Petit, T. R.; Dold, S.; Speck, D.; and Helmert, M. 2026. Code, benchmarks and data for the paper "Delete Relaxation With Axioms". https://doi.org/10.5281/zenodo.20607694.

Richter, S.; and Westphal, M. 2010. The LAMA Planner: Guiding Cost-Based Anytime Planning with Landmarks. 39: 127–177.

Rintanen, J. 2008. Regression for Classical and Nondeterministic Planning. In *Proc. ECAI 2008*, 568–572.

Röger, G.; and Helmert, M. 2010. The More, the Merrier: Combining Heuristic Estimators for Satisficing Planning. In *Proc. ICAPS 2010*, 246–249.

Speck, D.; Geißer, F.; Mattmüller, R.; and Torralba, Á. 2019. Symbolic Planning with Axioms. In *Proc. ICAPS 2019*, 464–472.

Speck, D.; Hecher, M.; Gnad, D.; Fichte, J. K.; and Corrêa, A. B. 2025. Counting and Reasoning with Plans. In *Proc. AAAI 2025*, 26688–26696.

Speck, D.; Seipp, J.; and Torralba, Á. 2025. Symbolic Search for Cost-Optimal Planning with Expressive Model Extensions. 82: 1349–1405.

Thiébaux, S.; and Cordier, M.-O. 2001. Supply Restoration in Power Distribution Systems — A Benchmark for Planning Under Uncertainty. In *Proc. ECP 2001*, 196–202.

Thiébaux, S.; Hoffmann, J.; and Nebel, B. 2005. In Defense of PDDL Axioms. 168(1–2): 38–69.

Tseitin, G. 1968. On the Complexity of Derivation in the Propositional Calculus. In *Studies in Constructive Mathematics and Mathematical Logic, Part II*, 115–125. Consultants Bureau, New York. English Translation.