Eliminating Negative Occurrences of Derived Predicates from PDDL Axioms

Claudia Grundke and Gabriele Röger

University of Basel, Switzerland
{claudia.grundke,gabriele.roeger}@unibas.ch

Abstract. Axioms are a feature of the Planning Domain Definition Language PDDL that can be considered as a generalization of database query languages such as Datalog. The PDDL standard restricts negative occurrences of predicates in axiom bodies to predicates that are directly set by actions and not derived by axioms. In the literature, authors often deviate from this limitation and only require that the set of axioms is stratifiable. Both variants can express exactly the same queries as least fixed-point logic, indicating that negative occurrences of derived predicates can be eliminated. We present the corresponding transformation.

1 Introduction

In classical planning, world states are described by a truth assignment to a finite set of ground atoms. The predicates are partitioned into *basic* and *derived* predicates. The actions may only directly affect the basic predicates, whereas the interpretation of the derived predicates is determined from the interpretation of the basic predicates by means of a logic program, consisting of so-called axioms. An axiom has the form $P(\boldsymbol{x}) \leftarrow \varphi(\boldsymbol{x})$ and expresses that the *head* $P(\boldsymbol{x})$ is true if the *body* $\varphi(\boldsymbol{x})$ is true.

Consider as an example a basic predicate E for an edge relation and a derived predicate *path*. The axiom $\mathit{path}(x, y) \leftarrow E(x, y) \vee \exists z(E(x, z) \wedge \mathit{path}(z, y))$ expresses that there is a path from x to y if there is an edge from x to y, or if x has some successor z from which there is a path to y. The axioms are evaluated by interpreting all derived atoms as false and successively making them true based on the axioms until a fixed point is reached. With this example axiom we would therefore interpret *path* as the transitive closure of the edge relation E.

The Planning Domain Definition Language PDDL is the dominant language for specifying classical planning tasks. The example axiom has the form that was introduced in PDDL 2.2 [2] and is still in effect today. It is backed up by compilability results [11] that establish that axioms increase the expressive power of PDDL. However, the PDDL standard restricts negative occurrences of predicates in axiom bodies to basic predicates, whereas the compilability analysis also permits negative occurrences of derived predicates as long as the set of axioms is *stratifiable*. This concept partitions the axioms into several strata that are successively evaluated by individual fixed-point computations. A derived

predicate may occur negatively in the body of an axiom if its interpretation has already been finalized by an earlier stratum.

Consider for an example an additional axiom $\mathit{acyclic}() \leftarrow \forall x \neg \mathit{path}(x, x)$.

The negative occurrence of derived predicate *path* in this axiom would be permitted in a stratifiable axiom program but not in PDDL 2.2.

This is not a fundamental limitation because both variants can express exactly the same queries as least fixed point logic (LFP) [10]. In this paper, we build on known transformations from LFP to directly compile away negative occurrences of derived predicates from stratifiable PDDL axioms programs. An extended version [3] with full proofs and a blow-up analysis is available on arXiv.

2 Background

We assume that the reader is familiar with first-order logic (FO). As in PDDL, we consider finite, relational vocabularies, i.e. no function symbols except for the constants, and finite structures, i.e. the universe is finite. We write $\varphi(x_1, \ldots, x_n)$ to indicate that $x_1, \ldots, x_n$ are the free variables in formula φ. We also use $\varphi(P_1, \ldots, P_m, x_1, \ldots, x_n)$, to indicate that φ does not mention predicate symbols other than $P_1, \ldots, P_m$ (treating them as free variables in a second-order formula).

An occurrence of a predicate in a formula is *positive* if it is under the scope of an even number of negations. Otherwise, it is *negative*. For example, in $\exists x \neg P(x) \wedge \neg \forall y \exists z \neg (P(y) \vee \neg P(z))$ the first occurrence of P (i.e. $P(x)$) is negative, the second one ($P(y)$) positive, and the last one again negative.

An *axiom* has the form $P(\boldsymbol{x}) \leftarrow \varphi(\boldsymbol{x})$, where $P(\boldsymbol{x})$ is a FO atom and $\varphi(\boldsymbol{x})$ is a FO formula such that $P(\boldsymbol{x})$ and $\varphi(\boldsymbol{x})$ have the same free variables $\boldsymbol{x}$. We call $P(\boldsymbol{x})$ the *head* and $\varphi(\boldsymbol{x})$ the *body* of the axiom. *Stratifiable* sets of axioms syntactically restrict sets of axioms to enable a well-defined semantics. For ease of presentation, we directly require a specific stratification. This is no limitation because all stratifications of a stratifiable set are semantically equivalent [1].

Definition 1 (Stratified Axiom Program). *A* stratified axiom program *is a finite sequence $(\Pi_1, \ldots, \Pi_n)$ of finite sets of axioms (the* strata*) such that for all $i \in \{1, \ldots, n\}$ it holds for all axioms $P(\boldsymbol{x}) \leftarrow \varphi(\boldsymbol{x})$ in Π_i that*

- *P does not occur in the head of an axiom in a stratum Π_k with $k \neq i$,*
- *P does not occur in a stratum Π_k with $k < i$,*
- *if a derived predicate P' appears positively in $\varphi(\boldsymbol{x})$ then the axioms having P' in their head are in some Π_j with $j \leq i$, and*
- *if a derived predicate P' appears negatively in $\varphi(\boldsymbol{x})$ then the axioms having P' in their head are in some Π_j with $j < i$.*

The semantics can be defined procedurally, iteratively extending a *basic state* s (interpreting the basic predicates for the universe of objects in the task, represented as a truth assignment to the ground atoms) to an *extended state* that also interprets the derived predicates. The key operation for an axiom is to consider all possible variable substitutions with objects from the universe and to

Algorithm 1 Extension of a basic state

function EXTEND(stratified axiom program $(\Pi_1, \ldots, \Pi_n)$, objects $\mathcal{O}$, basic state s_b)
  $s :=$ truth assignment to all ground atoms a with
    $s(a) = s_b(a)$ if the predicate of a is basic and $s(a) = \mathit{false}$ otherwise
  for $i \in \{1, \ldots, n\}$ **do** EXTENDSTRATUM(Π_i, $\mathcal{O}$, s) // modifies s
  return s

function EXTENDSTRATUM(stratum Π, objects $\mathcal{O}$, truth assignment s)
  while there exists a rule $\varphi \leftarrow \psi \in \Pi$ and a substitution σ of the free variables
      of ψ with objects such that $s \models \psi\{\sigma\} \wedge \neg\varphi\{\sigma\}$ **do**
    Choose such a $\varphi \leftarrow \psi$ and σ and set $s(\varphi\{\sigma\}) := \mathit{true}$

make the head true if the body is true under the current assignment. Algorithm 1 [4] extends the basic state stratum by stratum. Function EXTENDSTRATUM processes all axioms of the current stratum until it reaches a fixed point.

3 Elimination of Negative Occurrences

We build on a transformation for fixed point logic on finite structures [6, 5], based on a result for infinite structures [9]. We follow the structure by Leivant [7, 8] Transferring it to our axiom programs involves as a new contribution the handling of the *simultaneous* fixed point within each stratum.

Consider a stratified axiom program $\mathcal{P} = (\Pi_1, \ldots, \Pi_n)$ and let Π_ℓ be the earliest stratum that derives a predicate that occurs negatively in a later stratum. Let $P_1, \ldots, P_m$ be the predicates derived in Π_ℓ. We construct a program $\mathcal{P}' = (\Pi_1, \ldots, \Pi_{\ell-1}, \Pi'_\ell, \ldots, \Pi'_n)$ without negative occurrences of derived predicates from Π'_ℓ that defines the same fixed point for the predicates from $\mathcal{P}$. Repeating the process yields a program without negative occurrences of derived predicates.

For the transformation from $\mathcal{P}$ to $\mathcal{P}'$ we introduce additional predicates that can be related to the fixed-point computation for Π_ℓ. For this purpose, we subdivide this computation into several *stages*. Function EXTENDSTRATUMINSTAGES (Alg. 2) modifies parameter s exactly as EXTENDSTRATUM from Alg. 1 and can replace it within function EXTEND. The iterations of the for-loop (line 2) correspond to the different stages. For each stage, we take a snapshot of the current assignment and evaluate the bodies of the axioms only relative to this snapshot. Once a fixed point has been reached, the next stage begins with a new snapshot and we continue until the snapshot reaches a fixed point.

Consider the execution of EXTEND on a basic state s_b and the call of EXTENDSTRATUMINSTAGES for stratum Π_ℓ. Let f be the stage where the fixed point for this stratum is reached (the value of j in Algorithm 2 when it terminates is $f+1$). For an atom $P_i(\boldsymbol{a})$, we write $|\boldsymbol{a}|_{P_i}^{s_b}$ for the stage in which the truth of $P_i(\boldsymbol{a})$ is settled, i.e. $|\boldsymbol{a}|_{P_i}^{s_b}$ is the least number l such that $s_l(P_i(\boldsymbol{a}))$ is *true* in the execution of EXTENDSTRATUMINSTAGES, or $f+1$ if there is no such l.

We use these stages to define a number of auxiliary relations:

Algorithm 2 Extension for a stratum in stages

```
1: function extendStratumInStages(stratum Π, objects O, truth assignment s)
2:     for j ∈ 0, ... do
3:         s_j := copy of s
4:         while there exists a rule φ ← ψ ∈ Π and a substitution σ of the free
               variables of ψ with objects such that s_j ⊨ ψ{σ} and s ⊭ φ{σ}  do
5:             Choose such a φ ← ψ and σ and set s(φ{σ}) := true
6:         if s = s_j then return
```

Remember that m is the number of predicates occurring in a head of stratum Π_ℓ. For $i, j \in \{1, \ldots, m\}$, we define the relation $\prec^{i,j}$ such that $\boldsymbol{a} \prec^{i,j} \boldsymbol{b}$ iff $|\boldsymbol{a}|^{s_b}_{P_i} < |\boldsymbol{b}|^{s_b}_{P_j}$. This means that $P_i(\boldsymbol{a})$ is derived by extendStratumInStages in a strictly earlier iteration than $P_j(\boldsymbol{b})$, which possibly is not derived at all. Analogously, we define relation $\preceq^{i,j}$ as $\boldsymbol{a} \preceq^{i,j} \boldsymbol{b}$ iff $|\boldsymbol{a}|^{s_b}_{P_i} \leq |\boldsymbol{b}|^{s_b}_{P_j}$ and $|\boldsymbol{a}|^{s_b}_{P_i} \leq f$.

We explicitly represent the complement relations $\not\prec^{i,j}$ and $\not\preceq^{i,j}$, which are defined as $\boldsymbol{a} \not\prec^{i,j} \boldsymbol{b}$ iff $|\boldsymbol{a}|^{s_b}_{P_i} \geq |\boldsymbol{b}|^{s_b}_{P_j}$ and as $\boldsymbol{a} \not\preceq^{i,j} \boldsymbol{b}$ iff $|\boldsymbol{a}|^{s_b}_{P_i} > |\boldsymbol{b}|^{s_b}_{P_j}$ or $|\boldsymbol{a}|^{s_b}_{P_i} = f + 1$. We moreover introduce the relations $\lhd^{i,j}$ as $\boldsymbol{a} \lhd^{i,j} \boldsymbol{b}$ iff $|\boldsymbol{a}|^{s_b}_{P_i} + 1 = |\boldsymbol{b}|^{s_b}_{P_j}$.

The derivation order *within* a stage is irrelevant for the relations, so in the following, we write that $P_i(\boldsymbol{a})$ is derived *before*, *strictly before*, and *immediately before* $P_j(\boldsymbol{b})$ if $\boldsymbol{a} \preceq^{i,j} \boldsymbol{b}$, $\boldsymbol{a} \prec^{i,j} \boldsymbol{b}$, and $\boldsymbol{a} \lhd^{i,j} \boldsymbol{b}$, respectively.

Theorem 1. *The relations $\prec^{i,j}$, $\preceq^{i,j}$, $\not\prec^{i,j}$, $\not\preceq^{i,j}$ and $\lhd^{i,j}$ can be defined by a stratified axiom program with a single stratum.*

In the proof, we use the subscript ax (e.g. $\prec^{i,j}_{\text{ax}}$) to distinguish the predicates in the axioms from the relations. Moreover, we use subformulas of the form $\varphi(\boldsymbol{x})[\preceq^j \boldsymbol{y}]$. These mean that in $\varphi(\boldsymbol{x})$ every occurrence of an atom $P_k(\boldsymbol{z})$ with $k \in \{1, \ldots, m\}$ is replaced by $\boldsymbol{z} \preceq^{k,j}_{\text{ax}} \boldsymbol{y}$. Likewise for $\varphi(\boldsymbol{x})[\prec^j \boldsymbol{y}]$.

For example, for $\varphi(x, x') = \exists x''(P_1(x, x'') \wedge P_2(x'', x'))$ where P_1, P_2 are derived on stratum Π_ℓ, the formula $\varphi(x, x')[\prec^2 (y, y')]$ is $\exists x''((x, x'') \prec^{1,2}_{\text{ax}} (y, y') \wedge (x'', x') \prec^{2,2}_{\text{ax}} (y, y'))$. Intuitively, it corresponds to φ, where in the evaluation we may only use the atoms derived strictly before $P_2(y, y')$.

Similarly, we use formulas of the form $\varphi(\boldsymbol{x})[\neg \not\prec^j \boldsymbol{y}]$ that replace every occurrence of $P_k(\boldsymbol{z})$ by $\neg \boldsymbol{z} \not\prec^{k,j}_{\text{ax}} \boldsymbol{y}$. Likewise for $\varphi(\boldsymbol{x})[\neg \not\preceq^j \boldsymbol{y}]$. These will be used if we negate the formulas, so that overall all occurrences are positive again. The last kind of subformula is $\varphi(\boldsymbol{x})[\bot]$, replacing all occurrences of any $P_k(\boldsymbol{z})$ by $\bot$ (*false*). It is true, if φ is already true during the computation of the first stage.

Proof sketch. We assume w.l.o.g. that Π_ℓ contains only a single axiom for each predicate P_i. Otherwise we can combine the bodies of such axioms in a disjunction, renaming the variables accordingly. We refer to its body as $\varphi_i(\boldsymbol{x})$.

For $i, j \in \{1, \ldots, m\}$, we use the following axioms (explained below):

$$\boldsymbol{x} \prec^{i,j}_{\text{ax}} \boldsymbol{y} \leftarrow \bigvee\nolimits_{k=1}^{m} \exists \boldsymbol{z}(\boldsymbol{x} \preceq^{i,k}_{\text{ax}} \boldsymbol{z} \wedge \boldsymbol{z} \lhd^{k,j}_{\text{ax}} \boldsymbol{y}) \tag{1}$$

$$\boldsymbol{x} \preceq^{i,j}_{\text{ax}} \boldsymbol{y} \leftarrow \varphi_i(\boldsymbol{x})[\prec^j \boldsymbol{y}] \tag{2}$$

$$\boldsymbol{x} \not\prec^{i,j}_{\text{ax}} \boldsymbol{y} \leftarrow \varphi_j(\boldsymbol{y})[\bot] \vee \big(\bigvee_{k=1}^{m} \exists \boldsymbol{z}(\boldsymbol{x} \not\preceq^{i,k}_{\text{ax}} \boldsymbol{z} \wedge \boldsymbol{z} \lhd^{k,j}_{\text{ax}} \boldsymbol{y})\big) \vee \big(\bigwedge_{k=1}^{m} \forall \boldsymbol{z} \neg\varphi_k(\boldsymbol{z})[\bot]\big) \tag{3}$$

$$\boldsymbol{x} \not\preceq^{i,j}_{\text{ax}} \boldsymbol{y} \leftarrow \neg\varphi_i(\boldsymbol{x})[\neg \not\prec^{j} \boldsymbol{y}] \tag{4}$$

$$\boldsymbol{x} \lhd^{i,j}_{\text{ax}} \boldsymbol{y} \leftarrow \varphi_i(\boldsymbol{x})[\prec^{i} \boldsymbol{x}] \wedge \neg\varphi_j(\boldsymbol{y})[\neg \not\prec^{i} \boldsymbol{x}] \wedge \big(\varphi_j(\boldsymbol{y})[\preceq^{i} \boldsymbol{x}] \vee \bigwedge_{k=1}^{m} \forall \boldsymbol{z}(\neg\varphi_k(\boldsymbol{z})[\neg \not\preceq^{i} \boldsymbol{x}] \vee \varphi_k(\boldsymbol{z})[\prec^{i} \boldsymbol{x}])\big) \tag{5}$$

Note that all occurrences of the derived predicates in the bodies are positive.

Eq. (1) expresses that $P_i(\boldsymbol{x})$ is derived strictly before $P_j(\boldsymbol{y})$ if it is derived before some $P_k(\boldsymbol{z})$, which is in turn derived immediately before $P_j(\boldsymbol{y})$.

Eq. (2) states that $P_i(\boldsymbol{x})$ is derived before $P_j(\boldsymbol{y})$ because $P_i(\boldsymbol{x})$ can already be derived using only atoms that are derived strictly before $P_j(\boldsymbol{y})$.

In its three disjuncts, eq. (3) lists three possibilities why $P_i(\boldsymbol{x})$ is not derived strictly before $P_j(\boldsymbol{y})$: (a) $P_j(\boldsymbol{y})$ is already derived in stage 1, (b) there is some $P_j(\boldsymbol{z})$ derived immediately before $P_j(\boldsymbol{y})$ and $P_i(\boldsymbol{x})$ is not derived before this $P_k(\boldsymbol{z})$, so $P_i(\boldsymbol{x})$ is not derived strictly before $P_j(\boldsymbol{y})$, or (c) nothing can be derived at all, so both atoms are in the same (last) stage.

Eq. (4) states that $P_i(\boldsymbol{x})$ is not derived before $P_j(\boldsymbol{y})$ because it cannot be derived using only the atoms derived strictly before $P_j(\boldsymbol{y})$ (expressing $\prec$ as negated $\not\prec$ to avoid a negated occurrence of $\prec$ in the overall negated formula).

In its conjuncts, eq. (5) lists three requirements for $P_i(\boldsymbol{x})$ being derived immediately before $P_j(\boldsymbol{y})$: (a) $P_i(\boldsymbol{x})$ can be derived from the atoms derived strictly before $P_i(\boldsymbol{x})$, implying that it is true in the fixed point, (b) $P_j(\boldsymbol{y})$ cannot be derived from the atoms derived strictly before $P_i(\boldsymbol{x})$, implying that is not derived at the same stage as $P_i(\boldsymbol{x})$ (or earlier), and (c) $P_j(\boldsymbol{y})$ can be derived from the atoms derived before $P_i(\boldsymbol{x})$, or $P_i(\boldsymbol{x})$ was derived in the stage that reached the fixed point (and $P_j(\boldsymbol{y})$ is false in the fixed point). The last property is expressed by the requirement that all $P_k(\boldsymbol{z})$ that can be derived from the atoms derived before $P_i(\boldsymbol{x})$ can also be derived from the atoms derived *strictly* before $P_i(\boldsymbol{x})$. □

We can use these relations to eliminate negative occurrences, for example of $P_i(\boldsymbol{z})$: by definition, $\boldsymbol{z} \not\preceq^{i,i} \boldsymbol{z}$ iff $|\boldsymbol{z}|^{s_b}_{P_i} > |\boldsymbol{z}|^{s_b}_{P_i}$ or $|\boldsymbol{z}|^{s_b}_{P_i} = f+1$. Since $|\boldsymbol{z}|^{s_b}_{P_i} \not> |\boldsymbol{z}|^{s_b}_{P_i}$, this implies that $\boldsymbol{z} \not\preceq^{i,i} \boldsymbol{z}$ holds iff $P_i(\boldsymbol{z})$ is false in the fixed point of Π_ℓ (cf. definition of stage $f+1$), or equivalently, $\neg\boldsymbol{z} \not\preceq^{i,i} \boldsymbol{z}$ holds iff $P_i(\boldsymbol{z})$ is true.

We construct the desired program $\mathcal{P}' = (\Pi_1, \ldots, \Pi_{\ell-1}, \Pi'_\ell, \ldots, \Pi'_n)$ from $\mathcal{P} = (\Pi_1, \ldots, \Pi_n)$ as follows: $\Pi'_\ell = \Pi_\ell \cup \Pi_{stage}$, where Π_{stage} is the set of stage axioms for Π_ℓ as defined in the proof of Theorem 1. For $j \in \{\ell+1, \ldots, n\}$, we construct Π'_j from Π_j by replacing for all $i \in \{1, \ldots, m\}$ in each negative occurrence of $P_i(\boldsymbol{z})$ the $P_i(\boldsymbol{z})$ by $\neg\boldsymbol{z} \not\preceq^{i,i}_{\text{ax}} \boldsymbol{z}$, where $\not\preceq^{i,i}_{\text{ax}}$ is the predicate symbol for relation $\not\preceq^{i,i}$ in Π_{stage}.

Conclusion We demonstrated how one can eliminate all negative occurrences of derived predicates from stratifiable PDDL axiom programs. The extended version [3] of this paper with the full proof and algorithm as well as an example shows that the procedure only incurs a polynomial blow-up.

Acknowledgements

We have received funding for this work from the Swiss National Science Foundation (SNSF) as part of the project "Lifted and Generalized Representations for Classical Planning" (LGR-Plan).

References

1. Apt, K.R., Blair, H.A., Walker, A.: Towards a theory of declarative knowledge. In: Foundations of Deductive Databases and Logic Programming, pp. 89–148. Morgan Kaufmann (1988)
2. Edelkamp, S., Hoffmann, J.: PDDL2.2: The language for the classical part of the 4th International Planning Competition. Tech. Rep. 195, University of Freiburg, Department of Computer Science (2004)
3. Grundke, C., Röger, G.: Eliminating negative occurrences of derived predicates from PDDL axioms. arXiv:2510.14412 [cs.AI] (2025)
4. Grundke, C., Röger, G., Helmert, M.: Formal representations of classical planning domains. In: Bernardini, S., Muise, C. (eds.) Proceedings of the Thirty-Fourth International Conference on Automated Planning and Scheduling (ICAPS 2024). pp. 239–248. AAAI Press (2024)
5. Gurevich, Y., Shelah, S.: Fixed-point extensions of first-order logic. Annals of Pure and Applied Logic **32**, 265–280 (1986)
6. Immerman, N.: Relational queries computable in polynomial time. Information and Control **68**(1-3), 86–104 (1986)
7. Leivant, D.: Inductive definitions over finite structures. Information and Computation **89**(2), 95–108 (1990)
8. Libkin, L.: Elements of Finite Model Theory. Springer Berlin, Heidelberg (2004)
9. Moschovakis, Y.N.: Elementary Induction on Abstract Structures, Studies in Logic and the Foundations of Mathematics, vol. 77. Elsevier (1974)
10. Röger, G., Grundke, C.: Negated occurrences of predicates in PDDL axiom bodies. In: Proceedings of the KI 2024 Workshop on Planning, Scheduling and Design (PuK 2024) (2024)
11. Thiébaux, S., Hoffmann, J., Nebel, B.: In defense of PDDL axioms. Artificial Intelligence **168**(1–2), 38–69 (2005)