PDDL Axioms Are Equivalent to Least Fixed Point Logic

Claudia Grundke, Gabriele Röger

University of Basel
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. We show that both variants can express exactly the same queries as least fixed point logic. They are thus strictly more expressive than stratified Datalog, which aligns with another restriction on axioms occasionally considered in the planning literature. Complementing this theoretical analysis, we also present a compilation that eliminates negative occurrences of derived predicates from PDDL axioms.

1 Introduction

The aim in classical automated planning is to reach a desirable world state from the current world state by applying a suitable sequence of actions. World states are represented as sets of ground atoms that are true in the state. The set of all ground atoms of the planning task is defined by a finite set of predicates, each with an associated arity, and the finite set of constants (or objects) of the task. Ground atoms that do not occur in the set representation of a state s are false in s.

The predicates are partitioned into *basic* and *derived* predicates. The actions only directly affect the truth of 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 $P(\vec{x}) \leftarrow \varphi(\vec{x})$ and expresses that the *head* $P(\vec{x})$ is true if the *body* $\varphi(\vec{x})$ is true.

Consider as an example a basic predicate *E* for an edge relation and a derived predicate *path*. The axiom

$$\mathit{path}(x, y) \leftarrow \mathit{E}(x, y) \vee \exists z(\mathit{E}(x, z) \wedge \mathit{path}(z, y)) \qquad (1)$$

expresses that there is a path from x to y if there is a direct edge from x to y, or if there is an edge from x to a successor z from which there is a path to y. Axioms are evaluated by initializing 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*.

Capturing indirect effects of action applications, axioms facilitate the modelling of planning domains and can lead to a more efficient search for a plan (Ivankovic and Haslum 2015). The Planning Domain Definition Language PDDL is the dominant language for specifying classical planning tasks. A variant of axioms has already been introduced in PDDL 1.2 (McDermott et al. 1998) but has been dropped again in PDDL 2.1 (Fox and Long 2003) because the semantics were not clear. The previous example axiom corresponds to the form that was reintroduced in PDDL 2.2 (Edelkamp and Hoffmann 2004) and is still in effect in today's PDDL standard. This last revision is backed up by work by Thiébaux, Hoffmann, and Nebel (2005) that established that axioms increase the expressive power of PDDL and cannot be compiled away without a worst-case superpolynomial increase of the task representation or plan size.

Interestingly, the official definition differs from the variant considered in that paper in an aspect, namely by the restriction on negative occurrences of predicates in axiom bodies. We call the axiom formalism from the official PDDL definition AP_0 and the more general one from the compilability analysis AP. In AP_0, only basic predicates may occur negatively in axiom bodies, whereas AP also permits negative occurrences of derived predicates as long as the set of axioms is *stratifiable*. This concept allows to partition 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 as an example in addition to axiom (1) an axiom

$$\mathit{acyclic}() \leftarrow \forall x \neg \mathit{path}(x, x). \qquad (2)$$

The negative occurrence of derived predicate *path* in this axiom would be permitted in AP but not in AP_0.

Thiébaux, Hoffmann, and Nebel (2005) were aware of this difference of axiom formalisms but did not consider this restriction on the axiom language in their analysis.

The literature considers both variants. Some works adopt the restrictions of the PDDL standard and use AP_0 (Gazen and Knoblock 1997; Coles and Smith 2007; Gerevini, Saetti, and Serina 2011), while others rely on the more general concept of stratifiability and consider AP (Helmert 2009; Borgwardt et al. 2022; Grundke, Röger, and Helmert 2024).

The planning literature also considers planning tasks in a propositional or finite-domain representation (FDR), with *ground* actions and axioms (i.e. axioms do not mention any variables). In this context, it is prevalent to permit (stratifiable) negative occurrences of derived predicates (Ivankovic and Haslum 2015), although sometimes only restricted to an FDR variant of (ground) stratified Datalog (Miura and Fukunaga 2017; Speck et al. 2019; Speck and Gnad 2024). Since we focus on lifted axiom formalisms, these formalisms are not in the scope of our analysis.

In this paper, we make the following contributions:

- In Section 3, we show that both variants AP and AP_0 can express exactly the same queries as least fixed point logic.
- Based on this insight, we clarify their relationship to stratified Datalog in Section 4.
- The theoretical result indicates that negative occurrences of derived predicates can be eliminated. We present the corresponding compilation in Section 5.

Since we build on known results for least fixed point logic, we first provide the necessary background, formally introducing this logic and the axiom formalisms AP and AP_0.

2 Background

We assume that the reader is familiar with first-order logic (FO). We only consider first-order vocabularies with a finite set of predicates and constants and without function symbols (besides the constants). We also only consider *finite* structures $\mathcal{S}$ that interpret all predicates and constant symbols for a finite universe U.

We write $\varphi(x_1, \dots, x_n)$ to indicate that $x_1, \dots, x_n$ are the free variables in formula φ. A variable assignment maps the free variables to elements of the universe. We write $\mathcal{S} \models \varphi(o_1, \dots, o_n)$ to indicate that $\varphi(x_1, \dots, x_n)$ is true under $\mathcal{S}$ and the variable assignment that maps x_i to $o_i \in U$, using the usual semantics of FO. We also write $\vec{x}$ and $\vec{o}$ for vectors of variables and objects, leaving their length implicit.

Fixed point logic can define relations that are not already interpreted by the given structure. We write $\varphi(P_1, \dots, P_m, x_1, \dots, x_n)$ to indicate that predicates $P_1, \dots, P_m$ in formula φ refer to such relations (and $x_1, \dots, x_n$ are free variables).

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 ($P(z)$) again negative. In the planning literature (Thiébaux, Hoffmann, and Nebel 2005; Edelkamp and Hoffmann 2004), the same concept of negative occurrences is also described as negated appearances in the negation normal form of φ.

Expressiveness

We will compare the expressiveness of different formalisms based on which *queries* they can express. A query associates a structure $\mathcal{S}$ (with universe U) with a subset of $U \times \cdots \times U$.

For logic formalisms such as least fixed point logic, a query is given by a formula $\varphi(x_1, \dots, x_n)$ with free variables $x_1, \dots, x_n$ and defines the mapping $\varphi(\mathcal{S}) = \{\langle o_1, \dots, o_n \rangle \in U^n \mid \mathcal{S} \models \varphi(o_1, \dots, o_n)\}$.[1]

We will later also introduce a form of queries for the axiom formalisms under study.

For two formalisms X and Y, we write $X \leq Y$ to express that every X-query Q_X has an equivalent Y-query Q_Y, i.e. both associate every given finite structure $\mathcal{S}$ with the same set. In that case we say that Y is *at least as expressive* as X. It is easy to see that $\leq$ is transitive. We write $X = Y$ and call them *equally expressive* if $X \leq Y$ and $Y \leq X$, which means that both formalisms can define the same queries. A formalism Y is *strictly more expressive* than a formalism X, written $X < Y$, if $X \leq Y$ and $X \neq Y$.

Fixed Points

The formal definition of least fixed point logic and the semantics of PDDL axioms rely on the general concept of fixed points. Fixed points are a property of operators of the form $F : \mathcal{P}(S) \to \mathcal{P}(S)$, where S is a set and $\mathcal{P}(S)$ denotes its power set. An operator $F : \mathcal{P}(S) \to \mathcal{P}(S)$ is called

- *monotone* if $X \subseteq Y$ implies $F(X) \subseteq F(Y)$, and
- *inductive* if the sequence $X_0 = \emptyset$, $X_{i+1} = F(X_i)$ is increasing, i.e. $X_i \subseteq X_{i+1}$ for all i.

Every monotone operator is inductive and for inductive operators we define $X_\infty = \bigcup_{i=0}^{\infty} X_i$. We only consider finite sets S, so there always is an $n \in \mathbb{N}_0$ with $X_n = X_\infty$.

We call a set $X \subseteq S$ a *fixed point* of operator F if $X = F(X)$. It is the *least* fixed point $\mathbf{lfp}(F)$ if every other fixed point Y of F is a superset of X. We will later use the fact that every monotone operator F has a least fixed point $\mathbf{lfp}(F) = \bigcap\{X \mid X = F(X)\}$ and that $\mathbf{lfp}(F) = X_\infty$ (e.g. Libkin 2004, Theorem 10.2).

Least Fixed Point Logic

In least fixed point logic (LFP) we can introduce a new relation as the least fixed point of an operator that is defined in terms of a formula.

Let P be a new predicate symbol of arity k, i.e. P is not interpreted by $\mathcal{S}$, and let $\varphi(P, x_1, \dots, x_k)$ be a formula that is positive in P.

It induces an operator $F_\varphi : \mathcal{P}(U^k) \to \mathcal{P}(U^k)$ defined as

$$F_\varphi(X) = \{\langle o_1, \dots, o_k \rangle \mid \mathcal{S} \models \varphi(P/X, o_1, \dots, o_k)\},$$

where P/X means that P is interpreted as X in φ. If $\varphi(P, x_1, \dots, x_k)$ is positive in P then F_φ is monotone (Libkin 2004, Lemma 10.7).

For illustration, we demonstrate the computation of the least fixed point of F_φ for the example formula

$$\varphi(\mathit{path}, x, y) := E(x, y) \vee \exists z (E(x, z) \wedge \mathit{path}(z, y))$$

and a structure $\mathcal{S}$ with universe $U = \{a, b, c, d\}$ which interprets E as $\{\langle a, b \rangle, \langle b, c \rangle, \langle c, d \rangle\}$. Starting from $X_0 = \emptyset$, we collect all tuples $\langle o_1, o_2 \rangle \in U \times U$ for which $\mathcal{S} \models \varphi(\mathit{path}/\emptyset, o_1, o_2)$ holds. This is the case only for the tuples for which E is true. The existential subformula of φ is false for all $z \in U$ because *path* is interpreted as the empty set X_0

[1] We follow Libkin (2004) in overloading notation.

and thus false for all tuples. So, we get $X_1 := F_\varphi(X_0) = \{\langle a,b\rangle, \langle b,c\rangle, \langle c,d\rangle\}$.

Applying the operator F_φ to X_1, we get $X_2 := F_\varphi(X_1) = \{\langle a,b\rangle, \langle b,c\rangle, \langle c,d\rangle, \langle a,c\rangle, \langle b,d\rangle\}$. For instance, $\langle a,c\rangle$ is included in X_2 because $\mathcal{S} \models \varphi(\mathit{path}/X_1, a, c)$ holds, i.e. because U contains element b, for which $E(a,b)$ is true (interpreted by $\mathcal{S}$) and $\mathit{path}(b,c)$ is true (*path* being interpreted as X_1). If we continue the sequence, we get $X_3 = X_2 \cup \{\langle a,d\rangle\}$ and $X_4 = X_3$. The sequence reached a fixed point and we know that $X_\infty = \mathbf{lfp}(F_\varphi) = X_3$.

We are now prepared to define the logic LFP:

Definition 1 (Libkin 2004, Definition 10.6)**.** *The logic* LFP *extends* FO *with the following formation rule:*

- *If $\varphi(P, \vec{x})$ is a formula positive in P, where P is k-ary, and $\vec{t}$ is a tuple of terms, where $|\vec{x}| = |\vec{t}| = k$, then*

$$[\mathbf{lfp}_{P,\vec{x}}\varphi(P,\vec{x})](\vec{t})$$

is a formula, whose free variables are those of $\vec{t}$.

Semantically, $\mathcal{S} \models [\mathbf{lfp}_{P,\vec{x}}\varphi(P,\vec{x})](\vec{o})$ iff $\vec{o} \in \mathbf{lfp}(F_\varphi)$.

Continuing the graph example, we can see that the LFP formula $[\mathbf{lfp}_{\mathit{path},x,y}\varphi(\mathit{path},x,y)](a,d)$ holds in $\mathcal{S}$ because $\langle a,d\rangle \in X_3 = X_\infty$. As X_∞ is irreflexive, $\mathcal{S}$ also satisfies formula $\neg\exists z[\mathbf{lfp}_{\mathit{path},x,y}\varphi(\mathit{path},x,y)](z,z)$, expressing that a graph with edges as in the interpretation of E by $\mathcal{S}$ is acyclic.

Simultaneous Fixed Points In the previous definitions, the fixed point only iterates a single predicate. The formalism can be extended to simultaneous fixed points as follows:

Let $P_1, \ldots, P_n$ be new predicate symbols, where the arity of P_i is k_i and let $\vec{x}_i$ be a k_i-tuple of variables. Consider a set $\Phi = \{\varphi_1(P_1, \ldots, P_n, \vec{x}_1), \ldots, \varphi_n(P_1, \ldots, P_n, \vec{x}_n)\}$ of formulas that are positive in all predicates P_i.

Each formula φ_i defines an operator $F_i : \mathcal{P}(U^{k_1}) \times \cdots \times \mathcal{P}(U^{k_n}) \to \mathcal{P}(U^{k_i})$ as

$$F_i(X_1, \ldots, X_n) = \{\vec{o} \mid \mathcal{S} \models \varphi_i(P_1/X_1, \ldots, P_n/X_n, \vec{o})\}.$$

These operators induce a sequence of vectors from $\mathcal{P}(U^{k_1}) \times \cdots \times \mathcal{P}(U^{k_n})$ by $\vec{X}_0 = \langle \emptyset, \ldots, \emptyset\rangle$ and $\vec{X}_{i+1} = \langle F_1(\vec{X}_i), \ldots, F_n(\vec{X}_i)\rangle$. This sequence has a fixed point, which we denote by $\vec{X}_\infty$.

The logic LFP-sim extends FO with the formation rule $[\mathbf{lfp}_{P_i,\Phi}](\vec{t})$, where $\vec{t}$ is a tuple of length k_i. The semantics is that $\mathcal{S} \models [\mathbf{lfp}_{P_i,\Phi}](\vec{o})$ iff $\vec{o}$ is an element of the i-th component of $\vec{X}_\infty$.

Consider for example P_1 and P_2 with arity 1 and 2, respectively, and $\Phi = \{Q(x) \vee \exists y P_2(y,x), P_1(x) \wedge R(x,y)\}$. For $\mathcal{S}$ with universe $\{a,b,c,d\}$ such that exactly $R(a,b)$, $R(b,c)$, $R(c,d)$ and $Q(b)$ are true, we get the following sequence for the $\vec{X}_i$: $\vec{X}_1 = \langle\{\langle b\rangle\}, \emptyset\rangle$, $\vec{X}_2 = \langle\{\langle b\rangle\}, \{\langle b,c\rangle\}\rangle$, $\vec{X}_3 = \langle\{\langle b\rangle, \langle c\rangle\}, \{\langle b,c\rangle\}\rangle$, $\vec{X}_4 = \langle\{\langle b\rangle, \langle c\rangle\}, \{\langle b,c\rangle, \langle c,d\rangle\}\rangle$, $\vec{X}_5 = \langle\{\langle b\rangle, \langle c\rangle, \langle d\rangle\}, \{\langle b,c\rangle, \langle c,d\rangle\}\rangle = \vec{X}_6$. So for instance, $\mathcal{S} \models [\mathbf{lfp}_{P_1,\Phi}]\langle b\rangle$ and $\mathcal{S} \models \neg[\mathbf{lfp}_{P_2,\Phi}](a,b)$.

Simultaneous iteration of several predicates does not increase the expressive power compared to the iteration of a single predicate:

Theorem 1 (Libkin (2004), Cor. 10.8)**.** LFP-sim = LFP.

Axiom Programs

A planning domain defines a finite relational FO-vocabulary (i.e. one containing only predicates and constants), a set of actions and an axiom program. We skip the specifics of the actions as they are not relevant for this work. Instead we focus on the predicates, states and axioms. The predicates are partitioned into the *basic* and *derived* predicates. The derived predicates are those that occur in the heads of axioms. In the context of our work, we treat states as structures that interpret the predicates over a fixed universe U, consisting of the objects of the planning task. A state can alternatively be seen as the set of true ground atoms or a truth assignment to the ground atoms.

A *basic* state (or *basic* structure) only considers the basic predicates, whose interpretation is directly defined by the action applications. It gets extended to a full state (considering all predicates) by means of the axioms.

Definition 2 (Axiom)**.** *An* axiom *has the form*

$$P(\vec{x}) \leftarrow \varphi(\vec{x}),$$

where $P(\vec{x})$ is a FO *atom and $\varphi(\vec{x})$ is a* FO *formula such that $P(\vec{x})$ and $\varphi(\vec{x})$ have the same free variables $\vec{x}$.*

We call $P(\vec{x})$ the head *and $\varphi(\vec{x})$ the* body *of the axiom. The axiom* affects *the head predicate P. A set Π of axioms* affects *P if some axiom in Π affects P.*

The most general type of axiom programs commonly considered in planning are *stratifiable* axiom programs that enable a well-defined semantics. For ease of presentation, we directly require a specific stratification. This does not limit the scope of our work because all stratifiable axiom programs can be represented in this form and all stratifications of a stratifiable program are semantically equivalent (Apt, Blair, and Walker 1988, Thm. 11). Our definition is in this respect analogous to the definition of stratified Datalog by Ebbinghaus and Flum (1995). Intuitively, stratification prohibits recursion through negation.

Definition 3 (Stratified Axiom Program, AP)**.** *A* stratified axiom program *is a finite sequence $\langle \Pi_1, \ldots, \Pi_n\rangle$, where*

- *each Π_i is a finite set of axioms, called a* stratum*,*
- *all axioms affecting the same predicate P are in the same stratum, and*
- *for all strata Π_i and $P(\vec{x}) \leftarrow \varphi(\vec{x})$ in Π_i it holds that*
  - *if a predicate Q appears positively in $\varphi(\vec{x})$ then the axioms affecting Q are in $\bigcup_{k=1}^{i} \Pi_k$, and*
  - *if a predicate Q appears negatively in $\varphi(\vec{x})$ then the axioms affecting Q are in $\bigcup_{k=1}^{i-1} \Pi_k$.*

We refer to the class of all stratified axiom programs as AP.

The PDDL standard (Edelkamp and Hoffmann 2004) only permits negative occurrences of *basic* predicates in axiom bodies. Under this restriction all axioms can be combined into a single stratum (Thiébaux, Hoffmann, and Nebel 2005) and every stratified axiom program with a single stratum respects this restriction. We use this property to define the second axiom formalism that we want to consider.

We call such axiom programs *semipositive*, following the terminology for the analogous restriction in Datalog with negation (Abiteboul, Hull, and Vianu 1995).

Definition 4 (Semipositive Axiom Program, AP_0)**.** *A* semipositive *axiom program is a stratified axiom program $\langle\Pi\rangle$ that consists of a single stratum. We refer to the class of all semipositive axiom programs as* AP_0.

For the semantics of axiom programs, we first consider a single axiom stratum Π. Let $P_1, \ldots, P_m$ with arities $k_1, \ldots, k_m$ be the predicates affected by the axioms in Π and let $\mathcal{S}$ be a structure that interprets all predicates in Π except for $P_1, \ldots, P_m$ on some finite universe U. In the overall context, for the first stratum, $\mathcal{S}$ will be a basic state. For later strata, it will in addition interpret all predicates affected by earlier strata in the axiom program.

Stratum Π extends $\mathcal{S}$ to a structure $\mathcal{S}[\Pi]$ that also interprets the predicates $P_1, \ldots, P_m$ affected by Π, while preserving the interpretation of all other symbols from $\mathcal{S}$.

Conceptually, we consider all ground instantiations of the axioms in the stratum and make the head true if the body is true under $\mathcal{S}$ plus the already derived head atoms until we reach a fixed point. The interpretation of P_i is then the set of all derived ground atoms with predicate P_i.

Formally, the interpretation for $P_1, \ldots, P_m$ can be defined by the least simultaneous fixed point of the operator $F : \mathcal{P}(U^{k_1}) \times \cdots \times \mathcal{P}(U^{k_m}) \to \mathcal{P}(U^{k_1}) \times \cdots \times \mathcal{P}(U^{k_m})$, where $F(X_1, \ldots, X_m) = \langle F_1(X_1, \ldots, X_m), \ldots, F_m(X_1, \ldots, X_m)\rangle$ with

$$F_i(X_1, \ldots, X_m) = \bigcup_{P_i(\vec{x}) \leftarrow \varphi(P_1, \ldots, P_m, \vec{x}) \in \Pi} \{\vec{o} \mid \mathcal{S} \models \varphi(P_1/X_1, \ldots, P_m/X_m, \vec{o})\}.$$

If stratum Π contains for derived predicate P_i only one axiom $P_i(\vec{x}) \leftarrow \varphi(P_1, \ldots, P_m, \vec{x})$ then F_i simplifies to $F_i(X_1, \ldots, X_m) = \{\vec{o} \mid \mathcal{S} \models \varphi(P_1/X_1, \ldots, P_m/X_m, \vec{o})\}$.

Since F is monotone, the sequence of vectors from $\mathcal{P}(U^{k_1}) \times \cdots \times P(U^{k_m})$ defined by $\vec{X}_0 = \langle\emptyset, \ldots, \emptyset\rangle$ and $\vec{X}_{i+1} = F(\vec{X}_i)$ reaches a fixed point $\vec{X}_\infty$ that corresponds to the least fixed point of F.

The i-th component of this least fixed point defines the interpretation of predicate P_i in $\mathcal{S}[\Pi]$.

An entire stratified axiom program $\mathcal{P} = (\Pi_1, \ldots, \Pi_n)$ extends a basic state $\mathcal{S}$ stratum by stratum to

$$\mathcal{S}[\mathcal{P}] = (\ldots(\mathcal{S}[\Pi_1])[\Pi_2]\ldots)[\Pi_n].$$

We can now turn to the question whether AP is strictly more expressive than AP_0 and how both compare to LFP.

3 Equivalence of AP and AP_0 to LFP

To compare the expressiveness of the class of stratified axiom programs to least fixed point logic, we need to define the notion of a query for this formalism. For the related Datalog formalism, Ebbinghaus and Flum (1995) define a concept of Datalog formulas. We use the same concept for axioms queries.

Definition 5 (Axiom query)**.** *An* axiom query *is a pair $\mathcal{Q} = \langle\mathcal{P}, P\rangle$, where $\mathcal{P}$ is a stratified axiom program and P is a predicate with arity k affected by some stratum of $\mathcal{P}$.*

For basic structure $\mathcal{S}$ with universe U,

$$\mathcal{Q}(\mathcal{S}) = \{\langle o_1, \ldots, o_k\rangle \in U^k \mid \mathcal{S}[\mathcal{P}] \models P(o_1, \ldots, o_k)\}.$$

The query determines which ground atoms for predicate P can be derived from a given basic state.

Since semipositive axiom programs are a special case of stratified axiom programs, our first theorem is trivial.

Theorem 2. $\mathrm{AP}_0 \leq \mathrm{AP}$

Overall, we will establish that indeed $\mathrm{AP}_0 = \mathrm{AP} = \mathrm{LFP}$.

For this purpose, we consider a syntactically very limited fragment of LFP-sim called LFP_0 that extends FO as follows: Let Φ be a finite set of FO formulas $\varphi(P_1, \ldots, P_n, \vec{x})$ that are positive in all P_i. Then $[\mathbf{lfp}_{P_i,\Phi}](\vec{x})$ is an LFP_0 formula. Note that this is not a formation rule as in Definition 1, so these formulas may not be further combined with Boolean connectives, quantification or fixed point operators.

Although LFP_0 is syntactically very restrictive, it can be shown that it is as expressive as full LFP:

Theorem 3 (Libkin (2004), Cor. 10.13)**.** $\mathrm{LFP}_0 = \mathrm{LFP}$.

We will separately establish that $\mathrm{LFP}_0 \leq \mathrm{AP}_0$ (Theorem 4) and that $\mathrm{AP} \leq$ LFP-sim (Theorem 5). With Theorem 2 and LFP-sim $= \mathrm{LFP} = \mathrm{LFP}_0$ from Theorems 1 and 3, this implies that $\mathrm{LFP} \leq \mathrm{AP}_0 \leq \mathrm{AP} \leq \mathrm{LFP}$, establishing the equality of the formalisms.

Theorem 4. $\mathrm{LFP}_0 \leq \mathrm{AP}_0$.

Proof. We show how we can translate every LFP_0 query $\varphi(\vec{x})$ into an equivalent AP_0 query $\mathcal{Q} = (\mathcal{P}, Q)$.

Let $\varphi(\vec{x})$ be an arbitrary LFP_0 formula. Formula $\varphi(\vec{x})$ is either a FO formula or it has the form $[\mathbf{lfp}_{P_\ell,\Phi}](\vec{x})$, where Φ is a finite set of FO formulas.

If $\varphi(\vec{x})$ is a FO formula, we introduce a new predicate Q with arity $|\vec{x}|$ and use an axiom program

$$\mathcal{P} = \langle\{Q(\vec{x}) \leftarrow \varphi(\vec{x})\}\rangle$$

with a single axiom. Since Q is the only derived predicate and $\varphi(\vec{x})$ does not mention it, this program is trivially stratified. Since it consists of a single stratum, it is semipositive. It is easy to see that the query $\mathcal{Q} = \langle\mathcal{P}, Q\rangle$ is equivalent to the query $\varphi(\vec{x})$.

If $\varphi(\vec{x})$ has the form $[\mathbf{lfp}_{P_\ell,\Phi}](\vec{x})$ with $\Phi = \{\varphi_1(P_1, \ldots, P_n, \vec{x}_1), \ldots, \varphi_n(P_1, \ldots, P_n, \vec{x}_n)\}$, we construct a program with a single stratum Π as follows:

For each formula $\varphi_i(P_1, \ldots, P_n, \vec{x}_i) \in \Phi$, stratum Π contains an axiom $P_i(\vec{x}_i) \leftarrow \varphi_i(P_1, \ldots, P_n, \vec{x}_i)$. Note that for φ being an LFP_0 formula, each φ_i must be positive in all predicates $P_1, \ldots, P_n$, so the axiom program $\mathcal{P} := \langle\Pi\rangle$ is stratified and semipositive. The AP_0 query $\langle\mathcal{P}, P_\ell\rangle$ is equivalent to $\varphi(\vec{x})$. □

Theorem 5. $\mathrm{AP} \leq$ LFP-sim.

Proof. Let $\mathcal{Q} = \langle\mathcal{P}, Q\rangle$ be an AP query with $\mathcal{P} = \langle\Pi_1, \ldots, \Pi_n\rangle$. We show how we can translate $\mathcal{Q}$ into an equivalent LFP-sim formula φ.

We proof the statement by an induction over the strata.

We first only consider the first stratum Π_1. Let $P_1, \ldots, P_m$ be the derived predicates affected by this stratum. In a first step, we combine all axioms affecting the same predicate into a single axiom. For $i \in \{1, \ldots, m\}$,

let $\Pi_1^{P_i} \subseteq \Pi_1$ consist of the axioms that affect P_i. We can assume w.l.o.g. that all these axioms have the form $P_i(\vec{x}) \leftarrow \varphi(P_1, \ldots, P_n, \vec{x})$, using the same vector $\vec{x}$ of free variables. Otherwise we can rename the variables accordingly, if necessary replacing bound variables with fresh variables.

For each derived predicate P_i, let

$$\psi_{P_i}(P_1, \ldots, P_m, \vec{x}) := \bigvee_{P_i(\vec{x}) \leftarrow \varphi(P_1, \ldots, P_m, \vec{x}) \in \Pi_1^{P_i}} \varphi(P_1, \ldots, P_m, \vec{x}) \quad (3)$$

and define formula set $\Psi := \{\psi_{P_i}(P_1, \ldots, P_m, \vec{x}) \mid i \in \{1, \ldots, m\}\}$.

Since Π_1 is an axiom stratum, all bodies in its axioms are positive in $P_1, \ldots, P_m$ and consequently also all formulas $\psi_{P_i}(P_1, \ldots, P_m, \vec{x})$ are positive in all these predicates.

Thus $\chi_{P_i}(\vec{t}) := [\mathbf{lfp}_{P_i,\Psi}](\vec{t})$ is an LFP-sim formula. It has the property that for all structures $\mathcal{S}$ it holds that $\langle\langle \Pi_1\rangle, P_i\rangle(\mathcal{S}) = \chi_{P_i}(\mathcal{S})$, which can be seen from the semantics of axiom programs and least fixed point logic. Both induce the same operator for the simultaneous fixed point computation. The crucial insight is that the union in

$$F_i(X_1, \ldots, X_m) = \bigcup_{P_i(\vec{x}) \leftarrow \varphi(P_1, \ldots, P_m, \vec{x}) \in \Pi} \{\vec{o} \mid \mathcal{S} \models \varphi(P_1/X_1, \ldots, P_m/X_m, \vec{o})\}.$$

from the semantics of an axiom stratum is matched by the disjunction in equation (3) for ψ_{P_i}. Thus, F_i is equivalent to $F_i(X_1, \ldots, X_m) = \{\vec{o} \mid \mathcal{S} \models \psi_{P_i}(P_1/X_1, \ldots, P_m/X_m, \vec{o})\}$ which is the corresponding function in the definition of the simultaneous fixed point in LFP-sim. Since later strata do not change the interpretation of P_i, it holds also for all other j with $1 < j \leq n$ that $\langle\langle \Pi_1, \ldots, \Pi_j\rangle, P_i\rangle(\mathcal{S}) = \chi_{P_i}(\mathcal{S})$.

For the induction hypothesis, suppose that for every derived predicate P affected by an axiom stratum Π_ℓ with $\ell < l$, there is an LFP-sim formula $\chi_P(\vec{t})$ such that for all $\ell \leq j \leq n$ it holds for all structures $\mathcal{S}$ that $\langle\langle \Pi_1, \ldots, \Pi_j\rangle, P\rangle(\mathcal{S}) = \chi_P(\mathcal{S})$.

We show that then there is also such a formula χ_P for every predicate P affected by stratum Π_l. For this purpose, we construct for each such P a formula ψ_P as in equation 3 with the difference that for every occurrence of an atom $P'(\vec{t'})$ where P' is affected by an earlier stratum, we use instead formula $\chi_{P'}(\vec{t'})$ from the induction hypothesis. With $\Psi_l := \{\psi_P \mid P \text{ is affected by } \Pi_l\}$, we can define LFP-sim-formula $\chi_P(\vec{t}) := [\mathbf{lfp}_{P,\Psi_l}](\vec{t})$ as the desired formula with $\langle\langle \Pi_1, \ldots, \Pi_j\rangle, P\rangle(\mathcal{S}) = \chi_P(\mathcal{S})$ for all $l \leq j \leq n$ for all structures $\mathcal{S}$.

We have shown that for every derived predicate P there is an equivalent LFP-sim formula χ_P with $\langle \mathcal{P}, P\rangle(\mathcal{S}) = \chi_P(\mathcal{S})$. So this is also true for predicate Q and we have overall shown that for every AP query there is an equivalent LFP-sim query. □

Consider as an example an AP program $\mathcal{P}$ with axioms

$$P(x, y, z) \leftarrow E(x, y) \wedge (x \neq y) \wedge (y \neq z) \quad (4)$$
$$P(x, y, z) \leftarrow \exists u(E(x, u) \wedge P(u, y, z) \wedge (x \neq z)) \quad (5)$$
$$Q(x, y) \leftarrow E(x, y) \vee \exists u(R(x, u) \wedge E(u, y) \wedge P(x, u, y)) \quad (6)$$
$$R(x, y) \leftarrow \exists u(Q(x, u) \wedge Q(u, y) \wedge P(x, u, y)) \quad (7)$$
$$S(x) \leftarrow \neg Q(x, x), \quad (8)$$

with strata $\langle \Pi_1 = \{(4), (5)\}, \Pi_2 = \{(6), (7)\}, \Pi_3 = \{(8)\}\rangle$. For stratum Π_1, we use formula set $\Psi_1 = \{\varphi_P(P, x, y, z)\}$ with $\varphi_P(P, x, y, z) = (E(x, y) \wedge (x \neq y) \wedge (y \neq z)) \vee \exists u(E(x, u) \wedge P(u, y, z) \wedge (x \neq z))$. For stratum Π_2, we use set $\Psi_2 = \{\varphi_Q(Q, R, x, y), \varphi_R(Q, R, x, y)\}$ with $\varphi_Q(Q, R, x, y) = E(x, y) \vee \exists u(R(x, u) \wedge E(u, y) \wedge [\mathbf{lfp}_{P,\Psi_1}](x, u, y))$ and $\varphi_R(Q, R, x, y) = \exists u(Q(x, u) \wedge Q(u, y) \wedge [\mathbf{lfp}_{P,\Psi_1}](x, u, y))$. For the last stratum, we get $\Psi_3 = \{\varphi_S(x)\}$ with $\varphi_S(x) = \neg[\mathbf{lfp}_{Q,\Psi_2}](x, x)$. Now for all $\mathcal{S}$ (interpreting E), $([\mathbf{lfp}_{Q,\Psi_3}](x))(\mathcal{S}) = \langle \mathcal{P}, Q\rangle(\mathcal{S})$.

With the discussion at the beginning of this section we get the main result of this paper as a corollary of Theorems 2, 4 and 5:

Corollary 1. $\text{AP} = \text{AP}_0 = \text{LFP}$.

In the next section, we will use this result to relate AP and AP_0 to another frequently studied axiom formalism in planning, addressing a common misconception.

4 Relationship to Stratified Datalog

LFP is a well-studied formalism. For instance, it is known that *bounded* fixed-point logic (also known as stratified fixed-point logic) BFP is strictly weaker than LFP (Ebbinghaus and Flum 1995, Th. 7.7.2). This logic BFP in turn is equally expressive as *stratified Datalog* (Ebbinghaus and Flum 1995, Th. 8.1.1), which is a formalism very similar to AP, but restricting the bodies to (implicitly) existentially quantified *conjunctions* of literals.

This result on stratified Datalog is relevant in the context of axioms in planning. The Fast Downward planning system (Helmert 2006) translates the input task in a preprocessing phase to a normal form where axioms are stratified Datalog rules (Helmert 2009). The difference in expressiveness reveals a fundamental limitation of this translation because according to our results, it is not always possible. Indeed, a closer look at Fast Downward's transformation reveals that it can lead to non-stratifiable programs.

As an example, consider axiom $A := S(x) \leftarrow \exists y(M(x, y) \wedge \forall z(M(y, z) \rightarrow S(z)))$. This example is taken from the proof by Kolaitis (1991) where he establishes that fixed point logic has a higher expressive power than stratified Datalog by means of a game-tree separation. The axiom expresses that player 1 has a winning strategy from position x if she can move to a position y such that for all possible moves from y to z by player 2, player 1 has a winning strategy from z. The transformation by Fast Downward would introduce a new predicate $R(y)$ with axiom $B := R(y) \leftarrow \exists z(M(y, z) \wedge \neg S(z))$ and replace the axiom A with $A' := S(x) \leftarrow \exists y(M(x, y) \wedge \neg R(y))$. Now S

occurs negatively in the body of axiom B, so axiom A' must be in a strictly earlier stratum than B. But since R has a negative occurrence in the body of axiom A', axiom B must be on a strictly earlier stratum than A'. Overall, we see that a stratification is no longer possible after this step of the transformation. In this example, the problem occurs because the head predicate S occurs again under the scope of an universal quantifier in the body.

While a general transformation from AP to stratified Datalog is not possible according to our theoretical results, the transformation by Fast Downward must only be correct for a specific given task, which has a fixed finite universe. Thus it would be possible to expand universal quantifiers in axiom bodies for the entire universe (for the prize of a blow-up in the representation size). It could also be possible to apply a transformation that shifts part of the axiom evaluations into forced action applications, similar to the compilations by Thiébaux, Hoffmann, and Nebel (2005).

Also the work by Thiébaux, Hoffmann, and Nebel (2005) needs to be revisited in the light of our theoretical findings. They write:

"*Furthermore, we will often restrict the form of the right-hand side of axioms to a particularly simple form, namely, conjunctions of atoms, where all variables not appearing on the left-hand side of the axiom are implicitly existentially quantified. Such axioms are syntactically identical to DATALOG programs and for this reason we will call such axioms to be in DATALOG form. If the axioms contain negation, we say that they are in DATALOG$^\neg$ form. It is now a well-known fact from database theory that first-order queries can be rewritten into DATALOG$^\neg$ programs, which are linear in the size of the original formula (Abiteboul, Hull, and Vianu 1995). For this reason, we can concentrate in the following on stratified axioms in DATALOG$^\neg$ form as the most expressive axiom language.*"

According to our theoretical result, this formalism is strictly less expressive than general PDDL axiom programs, so something must be wrong with the rewriting argument.

Indeed, FO queries can be rewritten to non-recursive Datalog with negation. This is a special case of stratified Datalog, where the definition of no predicate depends (also indirectly) on itself.

The argument that arbitrary first-order formulas φ in axiom bodies can be replaced by a set of axioms that evaluate φ is tempting, but there are two issues: First, the rewriting can require several strata, but we need to be able to put all these axioms into the same stratum as the rewritten axiom body. Second, first-order queries have no notion of recursion and the rewriting evaluates the formula relative to a structure that already interprets *all* predicates. In this scenario it is permitted to introduce negated occurrences of predicates, which is not the case in our setting if the predicate is derived on the same stratum. Considering again $S(x) \leftarrow \exists y(M(x,y) \wedge \forall z(\neg M(y,z) \vee S(z)))$: the two axioms $Q(x) \leftarrow \exists y(M(x,y) \wedge \neg R(y))$ and $R(y) \leftarrow \exists z(M(y,z) \wedge \neg S(z))$ correspond to such a rewriting of the body and we would combine them with $S(x) \leftarrow Q(x)$ in one stratum. We see directly that the negated occurrence of S in the axiom affecting P breaks the stratification.

Note that this insight does not threaten the non-compilability results in the paper by Thiébaux, Hoffmann, and Nebel, so the central finding that axioms add to the expressive power of PDDL stays valid.

5 Eliminating Negative Occurrences

Another implication of our theoretical result is that every AP program has an equivalent AP_0 program, meaning negative occurrences of derived predicates can be eliminated from axiom bodies. This section presents such a compilation.

Our theoretical result builds on the fact that LFP_0 and LFP are equivalent, which was established by means of an actual compilation from LFP to LFP_0. We extract its core idea and transfer it to the axiom formalisms in planning.

The original result for fixed point logic goes back to Moschovakis (Moschovakis 1974) for infinite structures and was adapted to finite structures by Immerman (Immerman 1986) and Gurevich (Gurevich and Shelah 1986). We follow the structure by Leivant (Leivant 1990) as presented by Libkin (Libkin 2004, Cor. 10.13). This requires as a new contribution to directly handle the *simultaneous* fixed point within each stratum.

We will transform a given AP program $\mathcal{P} = \langle \Pi_1, \ldots, \Pi_n \rangle$ to a program $\mathcal{P}' = \langle \Pi'_1, \ldots, \Pi'_n \rangle$ that does not contain negative occurrences of derived predicates. Then all strata $\Pi'_1, \ldots, \Pi'_n$ can be combined into a single stratum, resulting in an equivalent AP_0 program.

For every derived predicate P of $\mathcal{P}$ and every basic structure $\mathcal{S}$ the structures $\mathcal{S}[\mathcal{P}]$ and $\mathcal{S}[\mathcal{P}']$ will interpret P equivalently, but program $\mathcal{P}'$ will use additional derived predicates. The idea is to introduce for each derived predicate P of $\mathcal{P}$ a new derived predicate $\bar{P}$ of the same arity for its complement, i.e. $\mathcal{S}[\mathcal{P}'] \models \bar{P}(\vec{t})$ iff $\mathcal{S}[\mathcal{P}'] \not\models P(\vec{t})$ (or equivalently $\mathcal{S}[\mathcal{P}] \not\models P(\vec{t})$). We use these complement predicates to replace negative occurrences $P(\vec{x})$ of P with the equivalent $\neg\bar{P}(\vec{x})$. Since $P(\vec{x})$ is under the scope of an odd number of negations, $\bar{P}(\vec{x})$ is then under the scope of an even number, so this is a positive occurrence of $\bar{P}$.

To achieve the intended interpretation of the complement predicates, we will need a number of auxiliary derived predicates to "analyze" what is *not* derived on each stratum.

The additional predicates are related to the fixed-point computation for a single stratum and we introduce them separately for each stratum Π_ℓ of $\mathcal{P}$.

Let $P_1, \ldots, P_m$ be the predicates affected by Π_ℓ. Consider an arbitrary basic structure $\mathcal{S}$ and the extension of $\mathcal{S}[\langle \Pi_1, \ldots, \Pi_{\ell-1} \rangle]$ to $\mathcal{S}[\langle \Pi_1, \ldots, \Pi_\ell \rangle]$, processing Π_ℓ. It is based on a sequence of vectors with $\vec{X}_0 = \langle \emptyset, \ldots, \emptyset \rangle$ and $\vec{X}_{i+1} = F(\vec{X}_i)$ and determines the interpretation of each P_i ($1 \leq i \leq m$) as the i-th component in the fixed point $\vec{X}_\infty$ of this sequence (cf. the semantics of PDDL axiom programs in Section 2).

We use the same sequence of vectors to associate the ground atoms of these predicates with different stages. We say that a ground atom $P_i(\vec{a})$ is *derived in stage* l if $\vec{a}$ occurs in the i-th component of X_l but *not* already in the i-th component of X_{l-1}. If an atom is never derived, i.e. if it is false in the interpretation defined by $\vec{X}_\infty$, then there is no such l.

Let f be the stage where the fixed point for stratum Π_ℓ is reached, i.e. f is the least number for which $\vec{X}_f = \vec{X}_\infty$. For an atom $P_i(\vec{a})$, we write $|\vec{a}|^{\mathcal{S}}_{P_i}$ to refer to the stage l in which the atom $P_i(\vec{a})$ is derived, or to $f+1$ if there is no such l.

Before turning to the additional auxiliary predicates we use the stages to define a number of auxiliary relations that will form the basis of these predicates. Remember that m is the number of predicates affected by stratum Π_ℓ. For $i, j \in \{1, \ldots, m\}$, we define the relation $\prec^{i,j}$ such that

$$\vec{a} \prec^{i,j} \vec{b} \text{ iff } |\vec{a}|^{\mathcal{S}}_{P_i} < |\vec{b}|^{\mathcal{S}}_{P_j}. \tag{9}$$

This means that $P_i(\vec{a})$ is derived in a strictly earlier stage than $P_j(\vec{b})$, which possibly is not derived at all.

Analogously, we define relation $\preceq^{i,j}$ as

$$\vec{a} \preceq^{i,j} \vec{b} \text{ iff } |\vec{a}|^{\mathcal{S}}_{P_i} \leq |\vec{b}|^{\mathcal{S}}_{P_j} \text{ and } |\vec{a}|^{\mathcal{S}}_{P_i} \leq f. \tag{10}$$

This means that $P_i(\vec{a})$ is derived in some stage and that this happens at latest in the stage in which $P_j(\vec{b})$ is derived (if the latter is derived at all).

We also explicitly represent the complement relations $\not\prec^{i,j}$ and $\not\preceq^{i,j}$, which are defined as

$$\vec{a} \not\prec^{i,j} \vec{b} \text{ iff } |\vec{a}|^{\mathcal{S}}_{P_i} \geq |\vec{b}|^{\mathcal{S}}_{P_j} \tag{11}$$

and as

$$\vec{a} \not\preceq^{i,j} \vec{b} \text{ iff } |\vec{a}|^{\mathcal{S}}_{P_i} > |\vec{b}|^{\mathcal{S}}_{P_j} \text{ or } |\vec{a}|^{\mathcal{S}}_{P_i} = f+1. \tag{12}$$

Lastly, we introduce the relation $\lhd^{i,j}$ as

$$\vec{a} \lhd^{i,j} \vec{b} \text{ iff } |\vec{a}|^{\mathcal{S}}_{P_i} + 1 = |\vec{b}|^{\mathcal{S}}_{P_j}. \tag{13}$$

In the following, we write that $P_i(\vec{a})$ is derived *before*, *strictly before*, and *immediately before* $P_j(\vec{b})$ if $\vec{a} \preceq^{i,j} \vec{b}$, $\vec{a} \prec^{i,j} \vec{b}$, and $\vec{a} \lhd^{i,j} \vec{b}$, respectively.

In the proof of Theorem 7 we will show how to express these relations by means of axioms. For this purpose we introduce corresponding predicate symbols with subscript ax (e.g. $\prec^{i,j}_{\text{ax}}$) to distinguish these new *stage predicates* from the corresponding *stage relations*, which are induced by a specific structure. In the proof of Theorem 7, we will present *stage axioms* that interpret these predicates for every basic structure as the corresponding stage relation. We will also see that these axioms do not introduce negative occurrences of any of these predicates.

Before we get to these details, we explain how the stage relations are relevant for the overall compilation: For each predicate P_i affected by Π_ℓ, we can achieve the intended meaning of auxiliary predicate $\bar{P}_i$ as its complement predicate ($\bar{P}_i(\vec{x})$ being true iff $P_i(\vec{x})$ is false) by an additional *complement axiom*

$$\bar{P}_i(\vec{x}) \leftarrow \vec{x} \not\preceq^{i,i}_{\text{ax}} \vec{x}. \tag{14}$$

The following theorem provides the theoretical basis for these complement axioms.

Theorem 6. *Let Π be a stratum of an axiom program $\mathcal{P}$, and P_i be a predicate affected by Π. Let $\mathcal{S}$ be a basic structure for $\mathcal{P}$ and relation $\not\preceq^{i,i}$ be defined as in equation* (12)*.*

For all $\vec{a}$ (of the arity of P_i) it holds that $\vec{a} \not\preceq^{i,i} \vec{a}$ iff $\mathcal{S}[\mathcal{P}] \not\models P_i(\vec{a})$.

Algorithm 1 Elimination procedure

```
1: function ELIMINATE(AP program P = ⟨Π_1, ..., Π_n⟩)
2:     for Π in Π_1, ..., Π_{n-1} do
3:         affected := {P | P is affected by Π}
4:         for each P_i, P_j ∈ affected do
5:             add the axioms for ≺^{i,j}_ax, ⪯^{i,j}_ax, ⊀^{i,j}_ax, ⋠^{i,j}_ax,
                                   and ◁^{i,j}_ax to Π
6:         for each P_i ∈ affected do
7:             add axiom P̄_i(x⃗) ← x⃗ ⋠^{i,i}_ax x⃗ to Π
8:     for each derived P_i occurring negatively in P do
9:         replace every negative occurrence P_i(x⃗) in
           P with ¬P̄_i(x⃗)
10:    return P
```

Proof. "⇒" Since $\vec{a} \not\preceq^{i,i} \vec{a}$ iff $|\vec{a}|^{\mathcal{S}}_{P_i} > |\vec{a}|^{\mathcal{S}}_{P_i}$ or $|\vec{a}|^{\mathcal{S}}_{P_i} = f+1$, and trivially $|\vec{a}|^{\mathcal{S}}_{P_i} \not> |\vec{a}|^{\mathcal{S}}_{P_i}$, it must hold that $|\vec{a}|^{\mathcal{S}}_{P_i} = f+1$. By the definition of the stages, this means that $P_i(\vec{a})$ is not in the fixed point for Π and thus $\mathcal{S}[\mathcal{P}] \not\models P_i(\vec{a})$.

"⇐" If $\mathcal{S}[\mathcal{P}] \not\models P_i(\vec{a})$ then $|\vec{a}|^{\mathcal{S}}_{P_i} = f+1$, so $\vec{a} \not\preceq^{i,i} \vec{a}$. □

Based on the complement axioms we can transform a given axiom program $\mathcal{P} = \langle \Pi_1, \ldots, \Pi_n \rangle$ as shown in Algorithm 1. In a first pass, we add to each stratum the stage axioms, as defined in the proof of Theorem 7, and the complement axioms, as in Equation (14). We can skip the last stratum because its affected predicates cannot occur negatively in a stratified program. In a second pass, we can then eliminate all negative occurrences of derived predicates, replacing them with their negated complement predicates.

We have now presented all components of the compilation except for the stage axioms. We conclude by showing that the stage relations can indeed be expressed by axioms.

Theorem 7. *The relations $\prec^{i,j}$, $\preceq^{i,j}$, $\not\prec^{i,j}$, $\not\preceq^{i,j}$ and $\lhd^{i,j}$ can be defined by an* AP_0 *program.*

In the following proof, we will use subformulas of the form $\varphi(\vec{x})[\preceq^j \vec{y}]$. These mean that in $\varphi(\vec{x})$ every occurrence of an atom $P_k(\vec{z})$ with $k \in \{1, \ldots, m\}$ is replaced by $\vec{z} \preceq^{k,j}_{\text{ax}} \vec{y}$. Likewise for $\varphi(\vec{x})[\prec^j \vec{y}]$.

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 $\varphi(x, x')[\preceq^2(y, y')]$ is $\exists x''((x, x'') \preceq^{1,2}_{\text{ax}} (y, y') \wedge (x'', x') \preceq^{2,2}_{\text{ax}} (y, y'))$. It corresponds to φ, where in the evaluation we may only use the atoms derived before $P_2(y, y')$.

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

Proof sketch. Let $P_1, \ldots, P_m$ be the predicates affected by a stratum Π_ℓ. We assume w.l.o.g. that all axioms in stratum Π_ℓ use distinct variables and that Π_ℓ contains for each P_i

only a single axiom that affects P_i. Otherwise we can combine the bodies of such axioms in a disjunction, renaming the variables accordingly. We refer to the body of the axiom affecting P_i as $\varphi_i(\vec{x})$.

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

$$\vec{x} \prec^{i,j}_{\text{ax}} \vec{y} \leftarrow \bigvee_{k=1}^{m} \exists \vec{z}(\vec{x} \preceq^{i,k}_{\text{ax}} \vec{z} \wedge \vec{z} \lhd^{k,j}_{\text{ax}} \vec{y}) \tag{15}$$

$$\vec{x} \preceq^{i,j}_{\text{ax}} \vec{y} \leftarrow \varphi_i(\vec{x})[\prec^j \vec{y}] \tag{16}$$

$$\begin{aligned} \vec{x} \not\prec^{i,j}_{\text{ax}} \vec{y} \leftarrow\ & \varphi_j(\vec{y})[\bot] \vee \\ & \Big(\bigvee_{k=1}^{m} \exists \vec{z}(\vec{x} \not\preceq^{i,k}_{\text{ax}} \vec{z} \wedge \vec{z} \lhd^{k,j}_{\text{ax}} \vec{y})\Big) \vee \\ & \Big(\bigwedge_{k=1}^{m} \forall \vec{z} \neg \varphi_k(\vec{z})[\bot]\Big) \end{aligned} \tag{17}$$

$$\vec{x} \not\preceq^{i,j}_{\text{ax}} \vec{y} \leftarrow \neg\varphi_i(\vec{x})[\neg \not\prec^j \vec{y}] \tag{18}$$

$$\begin{aligned} \vec{x} \lhd^{i,j}_{\text{ax}} \vec{y} \leftarrow\ & \varphi_i(\vec{x})[\prec^i \vec{x}] \wedge \neg\varphi_j(\vec{y})[\neg \not\prec^i \vec{x}] \wedge \\ & \big(\varphi_j(\vec{y})[\preceq^i \vec{x}] \vee \\ & \bigwedge_{k=1}^{m} \forall \vec{z}(\neg\varphi_k(\vec{z})[\neg \not\preceq^i \vec{x}] \vee \varphi_k(\vec{z})[\prec^i \vec{x}])\big) \end{aligned} \tag{19}$$

Note that these axioms have no negative occurrences of a stage predicate or of a predicate affected by Π_ℓ.

Eq. (15) expresses that $P_i(\vec{x})$ is derived strictly before $P_j(\vec{y})$ if it is derived before some $P_k(\vec{z})$, which is in turn derived immediately before $P_j(\vec{y})$.

Eq. (16) states that $P_i(\vec{x})$ is derived before $P_j(\vec{y})$ because $P_i(\vec{x})$ can already be derived using only atoms that are derived strictly before $P_j(\vec{y})$.

In its three disjuncts, eq. (17) lists three possibilities why $P_i(\vec{x})$ is not derived strictly before $P_j(\vec{y})$: (a) $P_j(\vec{y})$ is already derived in stage 1, (b) there is some $P_j(\vec{z})$ derived immediately before $P_j(\vec{y})$ and $P_i(\vec{x})$ is not derived before this $P_k(\vec{z})$, so $P_i(\vec{x})$ is not derived strictly before $P_j(\vec{y})$, or (c) nothing can be derived at all, so both atoms are in the same stage $f + 1$.

Eq. (18) states that $P_i(\vec{x})$ is not derived before $P_j(\vec{y})$ because it cannot be derived using only the atoms derived strictly before $P_j(\vec{y})$ (expressing $\prec$ as negated $\not\prec$ to avoid a negated occurrence of $\prec$ in the overall negated formula).

In its conjuncts, eq. (19) lists three requirements for $P_i(\vec{x})$ being derived immediately before $P_j(\vec{y})$: (a) $P_i(\vec{x})$ can be derived from the atoms derived strictly before $P_i(\vec{x})$, implying that it is true in the fixed point, (b) $P_j(\vec{y})$ cannot be derived from the atoms derived strictly before $P_i(\vec{x})$, implying that is not derived at the same stage as $P_i(\vec{x})$ (or earlier), and (c) $P_j(\vec{y})$ can be derived from the atoms derived before $P_i(\vec{x})$, or $P_i(\vec{x})$ was derived in the stage that reached the fixed point (and $P_j(\vec{y})$ is false in the fixed point). The last property is expressed by the requirement that all $P_k(\vec{z})$ that can be derived from the atoms derived before $P_i(\vec{x})$ can also be derived from the atoms derived *strictly* before $P_i(\vec{x})$.

The full proof is included in the extended version of this paper (Grundke and Röger 2026). □

The theoretical result in Section 3 implied that negative occurrences can be eliminated, but it did not explain how this can be achieved. Addressing this gap in understanding was our primary motivation for presenting this transformation. For this reason, we prioritized clarity over minimizing the size of the resulting axiom program.

There is a number of possible straight-forward improvements: we only introduced the complement predicates for instructional reasons, but could easily omit them and the complement axioms by using directly the body of these axioms in the replacement of negative occurrences; instead of introducing the stage axioms for *all* predicates in each stratum, we could analyze which of these are necessary based on the predicates with actual negative occurrences; last, many stage axioms contain the same subexpressions (e.g. the large conjunction in Eq. (19) does not depend on j). These could be represented only once by means of additional axioms.

6 Conclusion and Future Work

We considered three axiom formalism used in automated planning. The most general one, AP, allows arbitrary first-order formulas in axiom bodies as long as the axioms are stratifiable. The other two impose different syntactic restrictions: AP_0 forbids negative occurrences of derived predicates in axiom bodies or, equivalently, only permits a single stratum. Stratified Datalog restricts bodies to existentially quantified conjunctions of literals.

As a measure for expressiveness, we compared which queries these formalisms can express. We showed that AP and AP_0 are equally expressive, matching the expressive power of least fixed point logic, and thus strictly exceeding the expressiveness of stratified Datalog. This is the strongest possible negative result: Regardless of what time and space we can use in the compilation, some AP and AP_0 programs cannot be expressed in stratified Datalog.

Unlike Thiébaux, Hoffmann, and Nebel (2005), we examined axiom languages independently of the other components of the planning task and therefore did not consider compilations that shift part of the axiom evaluation into action applications. Using the corresponding compilation-scheme framework (Nebel 2000; Thiébaux, Hoffmann, and Nebel 2005), it may still be possible to compile tasks with AP axioms into tasks with stratified Datalog axioms if a non-constant increase of the plan length is allowed.

However, our positive result on the equivalence of AP and AP_0 also applies in this setting: the elimination procedure from Section 5 directly gives rise to a polynomial-time compilation scheme that preserves plan size exactly.

Our primary motivation for the elimination procedure was to obtain a deeper theoretical understanding. In future work, we will also evaluate its practical usefulness empirically. A potential obstacle is the induced blow-up: Although it is only polynomial on the lifted level, it may be prohibitive for planners that ground the axioms. The arity of derived predicates is a particularly critical factor, as the maximum arity will inevitably double for many axiom programs.

Moreover, the compilation introduces auxiliary predicates that are relevant only during the axiom evaluation but not for action applicability or for satisfying the goal. Such atoms increase the size of the state representation in the planning task. In forward state-space search, however, storing such atoms can be avoided, a general optimization that should be especially beneficial when using the compilation.

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

Abiteboul, S.; Hull, R.; and Vianu, V. 1995. *Foundations of Databases*. Addison-Wesley.

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.

Borgwardt, S.; Hoffmann, J.; Kovtunova, A.; Krötzsch, M.; Nebel, B.; and Steinmetz, M. 2022. Expressivity of Planning with Horn Description Logic Ontologies. In Honavar, V.; and Spaan, M., eds., *Proceedings of the Thirty-Sixth AAAI Conference on Artificial Intelligence (AAAI 2022)*, 5503–5511. AAAI Press.

Coles, A.; and Smith, A. 2007. Marvin: A Heuristic Search Planner with Online Macro-Action Learning. *Journal of Artificial Intelligence Research*, 28: 119–156.

Ebbinghaus, H.-D.; and Flum, J. 1995. *Finite Model Theory*. Springer-Verlag.

Edelkamp, S.; and Hoffmann, J. 2004. PDDL2.2: The Language for the Classical Part of the 4th International Planning Competition. Technical Report 195, University of Freiburg, Department of Computer Science.

Fox, M.; and Long, D. 2003. PDDL2.1: An Extension to PDDL for Expressing Temporal Planning Domains. *Journal of Artificial Intelligence Research*, 20: 61–124.

Gazen, B. C.; and Knoblock, C. A. 1997. Combining the Expressivity of UCPOP with the Efficiency of Graphplan. In Steel, S.; and Alami, R., eds., *Recent Advances in AI Planning. 4th European Conference on Planning (ECP 1997)*, volume 1348 of *Lecture Notes in Artificial Intelligence*, 221–233. Springer-Verlag.

Gerevini, A. E.; Saetti, A.; and Serina, I. 2011. Planning in Domains with Derived Predicates Through Rule-action Graphs and Local Search. *Annals of Mathematics and Artificial Intelligence*, 62(3,4): 259–298.

Grundke, C.; and Röger, G. 2026. PDDL Axioms Are Equivalent to Least Fixed Point Logic (Extended Version). arXiv:2510.14412v2 [cs.AI].

Grundke, C.; Röger, G.; and Helmert, M. 2024. Formal Representations of Classical Planning Domains. In Bernardini, S.; and Muise, C., eds., *Proceedings of the Thirty-Fourth International Conference on Automated Planning and Scheduling (ICAPS 2024)*, 239–248. AAAI Press.

Gurevich, Y.; and Shelah, S. 1986. Fixed-point Extensions of First-order Logic. *Annals of Pure and Applied Logic*, 32: 265–280.

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

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

Immerman, N. 1986. Relational Queries Computable in Polynomial Time. *Information and Control*, 68(1-3): 86–104.

Ivankovic, F.; and Haslum, P. 2015. Optimal Planning with Axioms. In Yang, Q.; and Wooldridge, M., eds., *Proceedings of the 24th International Joint Conference on Artificial Intelligence (IJCAI 2015)*, 1580–1586. AAAI Press.

Kolaitis, P. G. 1991. The Expressive Power of Stratified Programs. *Information and Computation*, 90(1): 50–66.

Leivant, D. 1990. Inductive Definitions over Finite Structures. *Information and Computation*, 89(2): 95–108.

Libkin, L. 2004. *Elements of Finite Model Theory*. Springer Berlin, Heidelberg.

McDermott, D.; Ghallab, M.; Howe, A.; Knoblock, C.; Ram, A.; Veloso, M.; Weld, D.; and Wilkins, D. 1998. PDDL – The Planning Domain Definition Language – Version 1.2. Technical Report CVC TR-98-003/DCS TR-1165, Yale Center for Computational Vision and Control, Yale University.

Miura, S.; and Fukunaga, A. 2017. Automatic Extraction of Axioms for Planning. In Barbulescu, L.; Frank, J.; Mausam; and Smith, S. F., eds., *Proceedings of the Twenty-Seventh International Conference on Automated Planning and Scheduling (ICAPS 2017)*, 218–227. AAAI Press.

Moschovakis, Y. N. 1974. *Elementary Induction on Abstract Structures*, volume 77 of *Studies in Logic and the Foundations of Mathematics*. Elsevier.

Nebel, B. 2000. On the Compilability and Expressive Power of Propositional Planning Formalisms. *Journal of Artificial Intelligence Research*, 12: 271–315.

Speck, D.; Geißer, F.; Mattmüller, R.; and Torralba, Á. 2019. Symbolic Planning with Axioms. In Lipovetzky, N.; Onaindia, E.; and Smith, D. E., eds., *Proceedings of the Twenty-Ninth International Conference on Automated Planning and Scheduling (ICAPS 2019)*, 464–472. AAAI Press.

Speck, D.; and Gnad, D. 2024. Decoupled Search for the Masses: A Novel Task Transformation for Classical Planning. In Bernardini, S.; and Muise, C., eds., *Proceedings of the Thirty-Fourth International Conference on Automated Planning and Scheduling (ICAPS 2024)*, 546–554. AAAI Press.

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