Negated Occurrences of Predicates in PDDL Axiom Bodies

Gabriele Röger and Claudia Grundke
University of Basel
Basel, Switzerland
{gabriele.roeger, claudia.grundke}@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 relate the axioms to fixed point logics to show that this aspect does not affect what queries can be expressed by the axioms.

1 Introduction

A world state in planning is specified by the interpretation of a relational vocabulary, i.e. a vocabulary consisting only of constant and predicate symbols. The predicates are partitioned into *basic* and *derived* predicates. The interpretation of basic predicates is directly affected by the actions of the agent. In contrast, the interpretation of the derived predicates is derived from this 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 *edge* and a derived predicate *reachable*. The axiom

$$\mathit{reachable}(x, y) \leftarrow \mathit{edge}(x, y) \vee \exists z(\mathit{edge}(x, z) \wedge \mathit{reachable}(z, y)) \tag{1}$$

expresses that y is reachable from x if there is an edge from x to y or if x has some successor z from which y is reachable. 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 *reachable* as the transitive closure of the *edge* relation.

A variant of axioms has already been introduced in version 1.2 of the Planning Domain Definition Language PDDL (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 also still in effect nowadays. This last revision was backed up by work by Thiébaux et al. (2005) that established that axioms increase the expressive power of PDDL and cannot be compiled away without a worst-case super-polynomial increase of the task representation or plan size.

Interestingly, the official definition and the variant in this paper differ in an aspect, namely the restriction on negated occurrences of predicates in axiom bodies. We call the axiom formalism from the official PDDL 2.2 definition AP_0 and the more general one from the compilability analysis AP. In AP_0, only basic predicates may occur negated in axiom bodies, whereas AP also permits negated 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 negated 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 $\mathit{acyclic}() \leftarrow \forall x \neg \mathit{reachable}(x, x)$. The negated occurrence of derived predicate *reachable* in this axiom would be permitted in AP but not in AP_0.

Thiébaux et al. (2005) were aware of this difference of axiom formalisms but did not consider this restriction on the axiom language in their analysis (but they did consider the impact of the stronger restrictions to DATALOG and non-recursive DATALOG).

The restriction of negation to basic predicates was not uncommon at the time of the definition of PDDL 2.2 (e.g. Gazen and Knoblock, 1997; Coles and Smith, 2007) but in the past decade it became more common to permit (stratifiable) negated occurrences of derived predicates (e.g. Ivankovic and Haslum, 2015; Miura and Fukunaga, 2017; Speck et al., 2019; Grundke et al., 2024; Speck and Gnad, 2024), although sometimes only restricted to a variant of stratified DATALOG.

In this paper, we will clarify the question whether AP is fundamentally more expressive than AP_0 and show that this is not the case. As a side-result we will establish that both formalisms are strictly more expressive than stratified DATALOG.

In contrast to Thiébaux et al. (2005), we consider axiom languages independently from planning tasks, so we do not consider compilations that can transfer part of the axiom evaluation into a sequence of action applications. Since our analysis builds on known results for least fixed point logic, we will first provide the necessary background, formally introducing this logic and the axiom formalisms AP and AP_0. Afterwards we establish our main result. We conclude with a discussion of possible implications and future work.

2 Background

We assume that the reader is familiar with first-order logic (FO). We only consider finite vocabularies σ that moreover are relational, i.e. they do not contain any function symbols except for the constants.

For a given FO-vocabulary σ, a *σ-structure* $\mathfrak{A}$ consists of a *universe* U and an interpretation of each symbol of σ: each constant symbol c is interpreted as a $c^{\mathfrak{A}} \in U$ and each k-ary predicate P is interpreted by a set $P^{\mathfrak{A}} \subseteq U^k$. We only consider finite structures, i.e. the universe U is finite.

We write $\varphi(x_1, \ldots, x_n)$ to indicate that $x_1, \ldots, x_n$ are the free variables in φ. A variable assignment α maps free variables to elements of the universe. We write $\mathfrak{A} \models \varphi(o_1, \ldots, o_n)$ to indicate that $\varphi(x_1, \ldots, x_n)$ is true under $\mathfrak{A}$ and the variable assignment that maps x_i to o_i, using the usual semantics of FO.

We call a formula φ *positive in predicate* P if every occurrence of P in φ is under the scope of an even number of negations. Otherwise, we say that P occurs negatively in φ. In the planning literature (e.g. Thiébaux et al., 2005; Edelkamp and Hoffmann, 2004), the same concept of negative occurrences of P is alternatively described as negated appearances in the negation normal form of φ.

A FO formula $\varphi(x_1, \ldots, x_n)$ for vocabulary σ specifies a *query* that associates each σ-structure $\mathfrak{A}$ (with universe U) with a subset of U^n as

$$\varphi(\mathfrak{A}) = \{(o_1, \ldots, o_n) \in U^n \mid \mathfrak{A} \models \varphi(o_1, \ldots, o_n)\}.^1$$

For two formalisms X and Y, we will write $X \leq Y$ to express that every X-query Q_X has an equivalent Y-query Q_Y, i.e. both associate every finite structure $\mathfrak{A}$ with the same set ($Q_X(\mathfrak{A}) = Q_Y(\mathfrak{A})$). It is easy to see from this definition that $\leq$ is transitive. We write $X = Y$ if $X \leq Y$ and $Y \leq X$, which means that both formalisms can define the same queries.

2.1 Fixed Point Logics

Before we introduce the relevant fixed point logics, we first introduce fixed points as a property of operators of the form $F : \mathcal{P}(S) \to \mathcal{P}(S)$, where S is a finite set and $\mathcal{P}(S)$ denotes its power set.

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.

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

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 some $n \in \mathbb{N}_0$ such that $X_n = X_\infty$.

A set $X \subseteq S$ with $X = F(X)$ is called a *fixed point* of F. It is the *least* fixed point $\mathbf{lfp}(F)$ if every other fixed point Y of F is a superset of X. Every monotone operator F has a least fixed point $\mathbf{lfp}(F) = \bigcap\{X \mid X = F(X)\}$ and it holds that $\mathbf{lfp}(F) = X_\infty$ (e.g. Libkin, 2004, Thm. 10.2).

For a structure $\mathfrak{A}$ with finite universe U, let $\varphi(P, \vec{x})$, where P is a k-ary predicate and $|\vec{x}| = k$, be a formula that is positive in P. The formula induces an operator $F_\varphi : \mathcal{P}(U^k) \to \mathcal{P}(U^k)$ defined as $F_\varphi(X) = \{\vec{o} \mid \mathfrak{A} \models \varphi(P/X, \vec{o})\}$, where $\varphi(P/X, \vec{o})$ means that P is interpreted as X in φ. If $\varphi(P, \vec{x})$ is positive in P then F_φ is monotone (Libkin, 2004, Lemma 10.7).

Consider as an example formula $\varphi(\mathit{above}, x, y) := \mathit{on}(x, y) \vee \exists z(\mathit{on}(x, z) \wedge \mathit{above}(z, y))$ and a structure $\mathfrak{A}$ with universe $U = \{a, b, c, d\}$, where *on* is interpreted as $\{(a, b), (b, c), (c, d)\}$.

For $X_0 = \emptyset$, we get $X_1 := F_\varphi(X_0) = \{(a, b), (b, c), (c, d)\}$ because these are the tuples for which *on* is true. The existential subformula of φ is false for all $z \in U$ because *above* is interpreted as the empty set X_0 and thus false for all tuples. Applying the same operator to X_1, we get $X_2 := F_\varphi(X_1) = \{(a, b), (b, c), (c, d), (a, c), (b, d)\}$. For example, (a, c) is included in X_2 because U contains element b, for which $\mathit{on}(a, b)$ is true (*on* being interpreted by $\mathfrak{A}$) and $\mathit{above}(b, c)$ is true (*above* being interpreted as X_1). If we continue the sequence, we get $X_3 = X_2 \cup \{(a, d)\}$ and $X_4 = X_3$. Since the sequence reached a fixed point, we know that $X_\infty = \mathbf{lfp}(F_\varphi) = X_3$.

We now are 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 $\vec{t}$.

The semantics is defined as follows: $\mathfrak{A} \models [\mathbf{lfp}_{P,\vec{x}}\varphi(P, \vec{x})](\vec{o})$ *iff* $\vec{o} \in \mathbf{lfp}(F_\varphi)$.

Continuing the blocksworld example, we can see that the structure $\mathfrak{A}$ is a model of LFP formula $[\mathbf{lfp}_{\mathit{above},x,y}\varphi(\mathit{above}, x, y)](a, d)$ and of LFP formula $\neg\exists z[\mathbf{lfp}_{\mathit{above},x,y}\varphi(\mathit{above}, x, y)](z, z)$, expressing that no object is above itself.

2.1.1 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:

Consider a relational vocabulary σ and let $P_1, \ldots, P_n$ be new predicate symbols, where the arity of P_i is k_i. 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 over $\sigma \cup \{P_1, \ldots, P_n\}$ that are all positive in all P_is.

For a structure $\mathfrak{A}$ over σ with universe U, each formula φ_i induces an operator $F_i : \mathcal{P}(U^{k_1}) \times \cdots \times P(U^{k_n}) \to \mathcal{P}(U^{k_i})$ defined as $F_i(X_1, \ldots, X_n) = \{\vec{o} \mid \mathfrak{A} \models \varphi_i(P_1/X_1, \ldots, P_n/X_n, \vec{o})\}$.

We define a sequence of vectors from $\mathcal{P}(U^{k_1}) \times \cdots \times P(U^{k_n})$ by $\vec{X}_0 = (\emptyset, \ldots, \emptyset)$ and $\vec{X}_{i+1} = (F_1(\vec{X}_i), \ldots, F_n(\vec{X}_i))$. 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 $\mathfrak{A} \models [\mathbf{lfp}_{P_i,\Phi}](\vec{i})$ iff $\vec{o}$ is an element of the i-th component of $\vec{X}_\infty$.

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

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

2.1.2 $\mathbf{LFP}_0$

Logic LFP_0 is a syntactically very limited fragment of LFP-sim and 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_is. 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 very restrictive, it can be shown that it is as expressive as full LFP:

Theorem 2 (Libkin (2004), Corollary 10.13)**.** $\text{LFP}_0 = \text{LFP}$.

2.2 Axiom Programs

We first define the general syntactic form of axioms and introduce some terminology.

Definition 2 (General Axiom Program)**.** *A* general axiom program Π *over a vocabulary σ is a finite set of* axioms *of the form $P(\vec{x}) \leftarrow \varphi(\vec{x})$, where $P(\vec{x})$ is a* FO *atom over σ and $\varphi(\vec{x})$ is a* FO *formula over σ such that $P(\vec{x})$ and $\phi(\vec{x})$ have the same free variables $\vec{x}$.*

We call $P(\vec{x})$ the head *and $\varphi(\vec{x})$ the* body *of the axiom.*

Every predicate symbol occurring in the head of an axiom is derived*, all the other symbols in σ are* basic*. We write σ_{derived} for the derived predicates and σ_{basic} for the basic vocabulary.*

In this paper, we use the established terminology from the planning community. In the related context of database theory, the basic predicates are called *extensional* predicates and the derived predicates *intensional* predicates.

In planning, the most general considered type of axiom programs are *stratifiable* axiom programs that restrict the syntax of general axioms programs to 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. Our definition is in this respect analogous to the definition of stratified DATALOG by Ebbinghaus and Flum (1995).

Definition 3 (Stratified Axiom Program, AP)**.** *An* axiom stratum *is a general axiom program, in which all bodies are positive in all derived predicates.*

A stratified axiom program *is a finite sequence $(\Pi_1, \ldots, \Pi_n)$ of axiom strata over vocabularies $\sigma_1, \ldots, \sigma_n$, where $(\sigma_i)_{\text{basic}} = \sigma_{i-1}$ for $i > 1$.*

We refer to the class of all stratified axiom programs over finite vocabularies as AP*.*

The condition on the vocabularies corresponds to the requirement that the program is stratified, i.e. for all axioms $P(\vec{x}) \leftarrow \varphi(\vec{x})$ in stratum Π_i it holds that

- if P' appears positively in $\varphi(\vec{x})$ then all axioms having P' in their head are in $\bigcup_{k=1}^{i} \Pi_k$, and
- if P' appears negatively in $\varphi(\vec{x})$ then all axioms having P' in their head are in $\bigcup_{k=1}^{i-1} \Pi_k$.

Intuitively, this definition ensures that recursion is not allowed through negation.

For the semantics, we first consider a single axiom stratum Π over vocabulary σ. Let $\mathfrak{A}$ be a σ_{basic}-structure with universe U. The stratum extends $\mathfrak{A}$ to a σ-structure $\mathfrak{A}[\Pi]$ with the same universe U via a simultaneous fixed point as follows: Let $P_1, \ldots, P_m$ be the derived predicates from σ_{derived} with arities $k_1, \ldots, k_m$, respectively. We define an operator $F : \mathcal{P}(U^{k_1}) \times \cdots \times P(U^{k_m}) \to \mathcal{P}(U^{k_1}) \times \cdots \times P(U^{k_m})$ as $F(X_1, \ldots, X_m) = (F_1(X_1, \ldots, X_m), \ldots, F_m(X_1, \ldots, X_m))$ where $F_i(X_1, \ldots, X_m) = \bigcup_{P_i(\vec{x}) \leftarrow \varphi(P_1, \ldots, P_m, \vec{x}) \in \Pi} \{\vec{o} \mid \mathfrak{A} \models \varphi(P_1/X_1, \ldots, P_m/X_m, \vec{o})\}$.

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

Stratum Π extends σ_{basic}-structure $\mathfrak{A}$ to σ-structure $\mathfrak{A}[\Pi]$, interpreting every symbol from $\mathfrak{A}$ as in $\mathfrak{A}$ and each derived predicate P_i as the i-th component of the least fixed point of F.

For an entire stratified axiom program $\mathcal{P} = (\Pi_1, \ldots, \Pi_n)$ over vocabularies $\sigma_1, \ldots, \sigma_n$, this extension is applied stratum by stratum, extending a $(\sigma_1)_{\text{basic}}$-structure $\mathfrak{A}$ to a σ_n-structure $\mathfrak{A}[\mathcal{P}]$ as $\mathfrak{A}[\mathcal{P}] = (\ldots(\mathfrak{A}[\Pi_1])[\Pi_2]\ldots)[\Pi_n]$.

A query refers to the interpretation of a specific predicate in the final structure:

Definition 4 (Axiom query)**.** *An* axiom query *over vocabularies $\sigma_1, \ldots, \sigma_n$ is a pair $\mathcal{Q} = (\mathcal{P}, P)$, where $\mathcal{P}$ is a stratified axiom program over these vocabularies and P is a derived predicate from some stratum (i.e. a predicate symbol in $\sigma_n \setminus (\sigma_1)_{\text{basic}}$) with arity k.*

The query associates each $(\sigma_1)_{\text{basic}}$-structure $\mathfrak{A}$ (with universe U) with a set $\mathcal{Q}(\mathfrak{A}) \subseteq U^k$ as the interpretation of P in $\mathfrak{A}[\mathcal{P}]$, i.e. $\mathcal{Q}(\mathfrak{A}) = P^{\mathfrak{A}[\mathcal{P}]}$.

The PDDL standard – as defined for PDDL 2.2 by Edelkamp and Hoffmann (2004) – only permits negative occurrences of predicates that are set by actions directly and not evaluated by means of the axioms. In our notation, this corresponds to restricting negative occurrences of predicates in axiom bodies to basic predicates of the first stratum. Under this restriction all axioms can be combined into a single stratum (Thiébaux et al., 2005) and every stratified axiom program with a single stratum respects this restriction, so 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 et al., 1995).

Definition 5 (Semipositive Axiom Program, AP_0)**.** *A* semipositive *axiom program is a stratified axiom program (Π) that consists of a single stratum.*

We refer to the class of all semipositive axiom programs over finite vocabularies as AP_0.

We can now turn to the question whether AP is strictly more expressive than AP_0.

3 Equivalence of AP and AP_0

Since semipositive axiom programs are a special case of stratified axiom programs, the following lemma is trivial.

Lemma 1. $\text{AP}_0 \leq \text{AP}$

Overall, we want to establish that indeed $\text{AP}_0 = \text{AP}$. We will achieve this indirectly, relating the formalisms to fixed-point logics. We will separately establish that $\text{LFP}_0 \leq \text{AP}_0$ (Lemma 2) and that $\text{AP} \leq \text{LFP-sim}$ (Lemma 3). With $\text{LFP-sim} = \text{LFP} = \text{LFP}_0$ from Theorems 1 and 2, this implies that $\text{AP} \leq \text{AP}_0$, so overall we get the equality of AP_0 and AP and as a side result their equality to LFP.

Lemma 2. $\text{LFP}_0 \leq \text{AP}_0$.

Proof. We show how we can translate every LFP_0 query φ over vocabulary σ into an equivalent AP_0 query $\mathcal{Q} = (\mathcal{P}, Q)$, i.e. for any finite σ-structure $\mathfrak{A}$ it holds that $\mathcal{Q}(\mathfrak{A}) = \varphi(\mathfrak{A})$.

Let $\varphi(\vec{x})$ be an arbitrary LFP_0 formula over some vocabulary σ. 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 $\mathcal{P} = (\{Q(\vec{x}) \leftarrow \varphi(\vec{x})\})$ with a single axiom. Since $\varphi(\vec{x})$ does not mention the only derived predicate Q, this program is trivially stratified. Since it consists of a single stratum, it is semipositive. It is easy to see that the query $\mathcal{Q} = (\mathcal{P}, Q)$ is equivalent to the query $\varphi(\vec{x})$.

If $\varphi(\vec{x})$ has the form $[\mathbf{lfp}_{P_\ell,\Phi}](\vec{x})$ with $\Phi = \{\varphi_1(P_1, \ldots, P_n, \vec{x}_1), \ldots, \varphi_n(P_1, \ldots, P_n, \vec{x}_n)\}$, where each P_i has arity k_i, we construct a program with a single stratum Π_1 as follows:

For each formula $\varphi_i(P_1, \ldots, P_n, \vec{x}_i) \in \Phi$, stratum Π_1 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 $\mathcal{P} := (\Pi_1)$ is stratified and semipositive. The AP_0 query $(\mathcal{P}, P_\ell)$ is equivalent to $\varphi(\vec{x})$. □

Lemma 3. $\text{AP} \leq \text{LFP-sim}$.

Proof. We show how we can translate every AP query $\mathcal{Q}$ over $\sigma_1, \ldots, \sigma_n$ into an equivalent LFP-sim formula φ, i.e. on any finite $(\sigma_1)_{\text{basic}}$-structure $\mathfrak{A}$ it holds that $\mathcal{Q}(\mathfrak{A}) = \varphi(\mathfrak{A})$.

Let $\mathcal{Q} = (\Pi, Q)$ be an arbitrary AP query over $\sigma_1, \ldots, \sigma_n$ with $\Pi = (\Pi_1, \ldots, \Pi_n)$ and let $\mathfrak{A}$ be a finite $(\sigma_1)_{\text{basic}}$-structure. The following construction does not depend on $\mathfrak{A}$.

We proof the statement by an induction over the strata.

We first only consider the first stratum Π_1. Let $P_1, \ldots, P_n$ be the derived predicates from this stratum and let $\Pi_1^{P_i} \subseteq \Pi_1$ contain the axioms that mention P_i in their head. 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.

For each derived predicate P_i, let

$$\psi_{P_i}(P_1, \ldots, P_n, \vec{x}) := \bigvee_{P_i(\vec{x}) \leftarrow \varphi(P_1, \ldots, P_n, \vec{x}) \in \Pi_1^{P_i}} \varphi(P_1, \ldots, P_n, \vec{x}) \tag{2}$$

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

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

Thus $\chi_{P_i}(\vec{t}) := [\mathbf{lfp}_{P_i, \Psi}](\vec{t})$ is an LFP-sim formula and it is easy to see from the semantics of axiom programs that $((\Pi_1), P_i)(\mathfrak{A}) = \chi_{P_i}(\mathfrak{A})$: the crucial insight is that the union in the operator $F_i(X_1, \ldots, X_m) = \bigcup_{P_i(\vec{x}) \leftarrow \varphi(P_1, \ldots, P_n, \vec{x}) \in \Pi} \{\vec{o} \mid \mathfrak{A} \models \varphi(P_1/X_1, \ldots, P_n/X_n, \vec{o})\}$ from the semantics of an axiom stratum is matched by the disjunction in χ_{P_i}, so that $F_i(X_1, \ldots, X_m) = \{\vec{o} \mid \mathfrak{A} \models \chi_{P_i}(P_1/X_1, \ldots, P_n/X_n, \vec{o})\}$, the latter set being the corresponding operator from the semantics of a simultaneous fixed point in LFP-sim.

Moreover, for all $1 < j \leq n$ it holds that $((\Pi_1, \ldots, \Pi_j), P_i)(\mathfrak{A}) = \chi_{P_i}(\mathfrak{A})$ because later strata do not change the interpretation of P_i.

Suppose that for every derived predicate P of an axiom stratum Π_ℓ with $\ell < m$, there is a LFP-sim formula $\chi_P(\vec{t})$ such that for all $\ell \leq j \leq n$ it holds that $((\Pi_1, \ldots, \Pi_j), P)(\mathfrak{A}) = \chi_P(\mathfrak{A})$ (induction hypothesis).

We show that then there also is such a formula χ_P for every derived predicate P of stratum Π_m. For this purpose, we construct for each $P \in (\sigma_m)_{\text{derived}}$ a formula ψ_P as in equation 2 with the difference that for every occurrence of an atom $P'(\vec{t})$ where P' is a derived predicate from an earlier stratum, we use instead formula $\chi_{P'}(\vec{t})$ from the induction hypothesis. With $\Psi_m := \{\psi_P \mid P \in (\sigma_m)_{\text{derived}}\}$, we can define LFP-sim-formula $\chi_P(\vec{t}) := [\mathbf{lfp}_{P, \Psi_m}](\vec{t})$ as the desired formula with $((\Pi_1, \ldots, \Pi_j), P)(\mathfrak{A}) = \chi_P(\mathfrak{A})$ for all $m \leq j \leq n$.

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

With the discussion at the beginning of this section we immediately get the main result of this paper as a corollary of Lemmas 1, 2 and 3:

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

4 Discussion

We considered the question whether permitting several strata in PDDL axioms allows us to express queries that could not be expressed with only a single stratum.

We answered this question negatively by means of the side result that both variants are equivalent to the least fixed point logic LFP (on finite structures).

This side result is interesting in its own right because 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, restricting the bodies to (implicitly) existentially quantified conjunctions of literals.

This insight reveals a limitation of the Fast Downward planning system (Helmert, 2006), which in a preprocessing phase translates the input task to a normal form where axioms are stratified DATALOG rules (Helmert, 2009). According to our results, this is not always possible and, indeed, a closer look at Fast Downward's transformation reveals that it can lead to non-stratifiable programs.

For an example, consider axiom $A := S(x) \leftarrow \exists y(M(x,y) \wedge \forall z(\neg M(y,z) \vee S(z)))$.[2] The transformation 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 $A' := S(x) \leftarrow \exists y(M(x,y) \wedge \neg R(z,y))$. 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 et al. (2005).

Since this issue did never materialize from a practical perspective and indeed does not occur on the benchmark instances of the international planning competitions, a natural question is whether the PDDL standard is not unnecessarily generous.

Axioms contribute to the expressive power of PDDL (under certain compilabilty requirements) and unless $\mathsf{EXPTIME} = \mathsf{PSPACE}$ cannot be compiled away without a super-polynomial increase in plan or task representation size. This is already true if they are restricted to pure DATALOG axioms (Thiébaux et al., 2005, Thm. 3) but there can be different levels of expressivity (wrt. compilability) between pure DATALOG and AP.

If we return from the compilability of entire planning tasks to the equivalence of the axiom and fixed-point formalisms, we find some additional interesting pointers in the literature.

Stratified DATALOG programs can be separated into different fragments, depending on their breadth, i.e. the maximal number of occurrences of derived predicates in axiom bodies from the same stratum. If these are not permitted (breadth 0), the formalism is equivalent to FO[3]; with breadth 1 it is equivalent to FO extended with a transitive closure operator, and for breadth ≥ 2 the resulting formalism is equivalent to general stratified DATALOG (Ebbinghaus and Flum, 1995, Cor. 8.1.3).

Since stratified DATALOG is the target formalism of the Fast Downward compilation, there is some empirical evidence that this formalism would already be sufficient for many interesting practical purposes, and it also is the considered formalism of some recent papers on axioms in planning (e.g. Miura and Fukunaga, 2017; Speck et al., 2019). However, with its restriction to conjunctions of literals in rule bodies, it is a bit cumbersome from a modelling aspect. An interesting starting point for an alternative formalism would be the DATALOG(FO,$\forall$) (Grädel, 1991) that extends DATALOG to first-order formulas over the basic vocabulary and a restricted form of universal quantification involving derived predicates. It is a proper subset of stratified DATALOG (Grädel, 1991) and it would be interesting what other, similar formalisms have been studied in the literature.

5 Conclusion

We have shown that the most general form of axiom programs in planning is equivalent to the variant from the PDDL standard and more expressive than a common restriction to stratified DATALOG.

In this paper, we presented preliminary work and our analysis was purely focusing on the axiom language and on the fundamental question whether there are queries that can be expressed in AP but not in AP_0, no matter the representation size. In future work, we will also take representation size into consideration and analyse the computational cost of a compilation.

[2] The 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.

[3] This is consistent with the result by Thiébaux et al. (2005, Thm. 4) that this non-recursive stratified DATALOG does not increase the expressive power of PDDL but can be compiled away.

References

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

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., editors, *Recent Advances in AI Planning. 4th European Conference on Planning (ECP 1997)*, volume 1348 of *Lecture Notes in Artificial Intelligence*, pages 221–233. Springer-Verlag.

Grädel, E. (1991). On transitive closure logic. In Börger, E., Jäger, G., Büning, H. K., and Richter, M. M., editors, *Proceedings of the 5th Workshop on Computer Science Logic (CSL 1991)*, volume 626 of *Lecture Notes in Computer Science*, pages 149–163. Springer.

Grundke, C., Röger, G., and Helmert, M. (2024). Formal representations of classical planning domains. In Bernardini, S. and Muise, C., editors, *Proceedings of the Thirty-Fourth International Conference on Automated Planning and Scheduling (ICAPS 2024)*, pages 239–248. AAAI Press.

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.

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

Kolaitis, P. G. (1991). The expressive power of stratified programs. *Information and Computation*, 90(1):50–66.

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., editors, *Proceedings of the Twenty-Seventh International Conference on Automated Planning and Scheduling (ICAPS 2017)*, pages 218–227. AAAI Press.

Speck, D., Geißer, F., Mattmüller, R., and Torralba, Á. (2019). Symbolic planning with axioms. In Lipovetzky, N., Onaindia, E., and Smith, D. E., editors, *Proceedings of the Twenty-Ninth International Conference on Automated Planning and Scheduling (ICAPS 2019)*, pages 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., editors, *Proceedings of the Thirty-Fourth International Conference on Automated Planning and Scheduling (ICAPS 2024)*, pages 546–554. AAAI Press.

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