Proceedings of the Twenty-Seventh International Conference on Automated Planning and Scheduling (ICAPS 2017)

Automatic Extraction of Axioms for Planning

Shuwa Miura, Alex Fukunaga
Department of General Systems Studies
Graduate School of Arts and Sciences
The University of Tokyo
miura-shuwa@g.ecc.u-tokyo.ac.jp, fukunaga@idea.c.u-tokyo.ac.jp

Abstract

Axioms can be used to model derived predicates in domain-independent planning models. Formulating models which use axioms can sometimes result in problems with much smaller search spaces than the original model. We propose a method for automatically extracting a particular class of axioms from standard STRIPS PDDL models. More specifically, we identify operators whose effects become irrelevant given some other operator, and generate axioms that capture this relationship. We show that this algorithm can be used to successfully extract axioms from standard IPC benchmark instances, and show that the extracted axioms can be used to significantly improve the performance of satisficing planners.

1 Introduction

In the most commonly studied classical planning models, all changes to the world are direct effects of operators. However, it is possible to model some effects as indirect effects which can be inferred from a set of basic state variables. Such *derived predicates* can be expressed in modeling languages such as PDDL and formalisms such as SAS+ as *axioms*, logical rules which define how the derived predicates follow from basic variables. Planners have supported various forms of derived predicates since relatively early systems (Manna and Waldinger 1987; Barrett et al. 1995), and PDDL has supported axioms which specify derived predicates as a logic program with negation-as-failure semantics since version 2.2 (Edelkamp and Hoffmann 2004)

Previous work on derived predicates and axioms for planning has focused on the advantages of expressivity (compactness) of domain modeling using axioms, (Thiébaux, Hoffmann, and Nebel 2005), as well as search algorithms which are aware of axioms (Coles and Smith 2007; Gerevini, Saetti, and Serina 2011; Ivankovic and Haslum 2015b). Previous work has shown that axioms can be exploited by a search algorithm to solve problems more efficiently, compared to a version of the problem without explicit axioms.

Consider the single-agent puzzle Sokoban, in which the player pushes stones around in a maze. The goal is to push all the stones to their destinations. The standard PDDL formulation of Sokoban used in the International Planning Competition(IPC) consists of two kinds of operators, push and move. push lets the player push a box in one direction, while move moves the player into an unoccupied location.

Ivankovic and Haslum (2015a) proposed a new formulation of Sokoban with axioms and showed that this leads to a problem with a smaller search space (Ivankovic and Haslum 2015b). They remove the move operators entirely, and introduce axioms to check whether the player can reach a box to push it. The reformulated push operators now have a derived predicate reachable(loc) instead of at-player=loc as their precondition. The values of the derived predicates are determined by the following axioms:

1. reachable(loc) ← at-player=loc
2. reachable(to) ← reachable(from), clear(to), connected(from,to)

Intuitively, the first axiom means that the current location of the player is reachable. The second axiom means a clear location next to a reachable location is also reachable. Figure1 illustrates the search space with and without axioms. With axioms, the search space only has the transitions indicated by the black arrows. The transitions indicated by the white arrows are captured by axioms as reachability analysis.

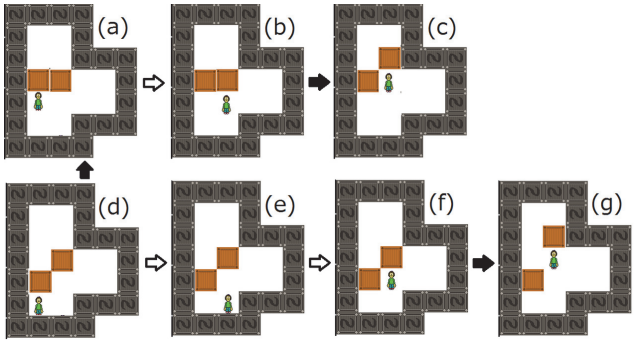

Figure 1: The search space of the Sokoban domain.

While previous work focused on axioms as expressive domain modeling constructs used by human domain modelers, this paper shows that axioms can also be viewed as structure to be discovered and exploited. We investigate a *completely automated*, reformulation approach which extracts axioms from standard PDDL domains, solves the reformulated problem instance with an axiom-aware planner, and

then converts the solution to the axiom-reformulated problem back into a valid plan for the original problem instance. For example, in Sokoban, the standard domain with the push and move operators is automatically reformulated to a domain without the move operator. After finding a plan which achieves the goals in this reformulated state space, which consists solely of push, we then decode the plan to generate a valid plan for the original domain (with move and push).

We propose methods for automatically extracting two classes of axioms from standard (axiom-free) SAS+(STRIPS) planning models. Our first method (Sec. 3) partitions the operators in a SAS+ model into observable operators, which are in some sense "fundamental" operators, and τ-operators, which are "auxiliary" operators. Given a list observable operators to execute, a set of τ-operators to be executed is implied. We reformulate the problem such that the preconditions of the observable operators are derived predicates which are established by τ-operators, and axioms representing how these derived predicates are implied are added to the domain. Our second method (Sec. 4) extracts axioms based on groups of fluents where exactly one of the fluents must be true at all times. We generate ϵ-axioms which describe the truth value of one member of such a "exactly-1" group in terms of the other members of the group. Sec. 5 describes the procedure for converting solutions to the reformulated problems to solutions for the original domain.

We show that τ-operators and ϵ-axioms can be extracted from half of the STRIPS + action costs IPC benchmark domains (Sec. 6). We then evaluate our reformulation on various planners (Sec. 7) and show that the axioms extracted from IPC standard benchmarks can be exploited to speed up satisficing planning, for model-based planning (integer programming and answer set programming).

2 Review of SAS+ and Axioms

We adopt the definition of axiom-enhanced SAS+ used in Helmert (2009) and Ivankovic and Haslum (2015b).

Definition 1. A SAS+ problem with axioms Π is a tuple (V, O, I, G, U, A) where
- V is a set of *primary variables*. Each variable v_i has a finite domain of values $D(v_i)$.
- O is a set of operators. Each operator o has a precondition (pre(o)) and an effect (eff(o)), which consists of variable assignments of the form $v_i{=}x$ where $x{\in}D(v_i)$. We abbreviate $v_i{=}1$ as v_i when we know the variable is binary. An operator o is applicable in a state s iff s satisfies pre(o). The resultant state $o(s)$ has the assignments specified in eff(o). We abbreviate a sequence of application of operators o_1,o_2,...,o_n as $o_1o_2...o_n(s)$. cost(o) is the associated cost of the operator o.
- I is an initial assignment over primary variables.
- G is a partial assignment over variables that specifies the goal conditions.
- U is a set of *secondary variables*. Secondary variables are binary and do not appear in operator effects. Their values are determined by axioms after each operator execution.
- A is a set of axioms with form $u{\leftarrow}\phi$ where u (head) is a secondary variable and ϕ (body) is a conjunction of positive and negative variable assignments. Intuitively, an axiom requires u to be true when ϕ holds. Unless implied by an axiom, u is false by default (negation by failure semantics). $A(s)$ denotes the result of evaluating a set of axioms A in a state s.

Axioms and primary variables can be seen as forming a logic program. Formally, the values of secondary variables are determined by the stable model (c.f., Gelfond and Kahl 2014) of the program.

The set of axioms must be *stratified*. A set of axioms is stratifiable if and only if there is a mapping l from U to $\{0, ..., m\}$ such that
- for every axiom a that has u_i as its head, for every u_j that appear in its body, $l(u_j){\leq}l(u_i)$
- for every axiom a that has u_i as its head, for every not u_j that appear in its body, $l(u_j){<}l(u_i)$

When A is stratified, $A(s)$, the result of axiom evaluation, is guaranteed to be unique (Apt, Blair, and Walker 1988).

A solution (*plan*) to Π is an applicable sequence of operators $o_0, ..., o_n$ that maps I into a state where the G holds.

3 τ-Axiom Extraction: Representing Internal Transitions as Axioms

We extract operators whose effects become irrelevant given some other operator, and express this structure as axioms. For example, in the aforementioned Sokoban example, we need move operators only to check if the player can get behind a stone to push it. How the player reaches there does not matter since the resulting state will always be the same. In this sense, the states visited by the player's move are "indistinguishable" with respect to the push operator. We formally capture this intuition as follows.

Definition 2. A pair of states s,s' are *distinguishable* with respect to an operator o if and only if o is applicable in both states and the resulting states differ ($o(s){\neq}o(s')$). Otherwise, s,s' are *indistinguishable*.

Definition 3. A set of states S is indistinguishable with respect to o if and only if every pair of states in S are indistinguishable with respect to o.

In Figure 1, states (b) and (f) are distinguishable w.r.t. the grounded push operator since the resulting states ((c) and (g)) differ. However, states (d),(e) and (f) are indistinguishable w.r.t. the grounded push operator since the push operator is applicable only in (f).

Using the notion of indistinguishability, we identify a set of move-like operators and push-like operators.

Definition 4. Let $\mathcal{T}(s)$ be a set of states reachable from s using only the operators in $\mathcal{T}{\subset}O$, We call a set of operators $\mathcal{T}{\subsetneq}O$ the *τ-operators*, if and only if for every state s, $\mathcal{T}(s)$ is indistinguishable with respect to every operator in $\bar{\mathcal{T}}$. We call $\bar{\mathcal{T}}$ the *observable operators*. We denote a sequence of zero or more consecutive τ-operators as τ^*.

Theorem 1. Given a state s and an applicable sequence of operators $\tau^* o$ and $\tau^{*'} o$ (o is an observable operator), $\tau^* o(s){=}\tau^{*'} o(s)$.

Proof. Suppose $\tau^* o(s) \neq \tau^{*'} o(s)$. This means $\tau^*(s)$ and $\tau^{*'}(s)$ are distinguishable w.r.t. an observable operator o, which contradicts Definition 4. □

Now that we know τ-operators are just checking if observable operators are applicable rather than making change of their own, we want to express τ-operators as axioms. Below, we show how to 1. identify τ and observable operators 2. convert τ-operators into axioms which correctly infer applicability of observable operators, and 3. modify observable operators accordingly for the reformulated domain.

3.1 τ-Axiom Extraction

We use the multi-valued semantics of a SAS+ problem to identify τ-operators using the notion of dominance.

Definition 5. Let effvar(o) and prevar(o) be the set of variables that appear in eff(o) and pre(o) respectively. An operator o *dominates* o' if and only if effvar(o') $\subseteq$ prevar(o).

For example, consider the following (simplified) SAS+ formulation of Sokoban, with two actions,

move(from,to): pre(move) = [at-player = from] $\wedge$ clear(to), eff(move) = [at-player = to] , and;

push(from,to,dest) : pre(push) = [at-player = from] $\wedge$ [at-stone = to] $\wedge$ clear(dest), eff(push) = [at-player = to] $\wedge$ [at-stone = dest].

Since effvar(move(from,to)) = {at-player} $\subseteq$ prevar(push(from,to,dest)) = {at-player,at-stone,clear(dest)}, push(from,to,dest) dominates move(from,to).

Theorem 2. Given a set of operators $\mathcal{T}$, if every operator in $\mathcal{T}$ is dominated by every operator in $\bar{\mathcal{T}}$, then $\mathcal{T}$ is a set of τ-operators.

Proof. Let $\mathcal{T}$ be a set of operators such that every operator in $\mathcal{T}$ is dominated by every operator in $\bar{\mathcal{T}}$. Given a state s and an operator $o \in \bar{\mathcal{T}}$, assume $s', s'' \in \mathcal{T}(s)$ are distinguishable w.r.t. o. Since o is applicable in both s' and s'', they must have the same values for all variables in $pre(o)$, and there must exist some variable v not in $pre(o)$ whose value differs in s' and s''. However, by the definition of dominance, operators in $\mathcal{T}$ cannot change the values of variable not in pre(o), so there can be no such v, a contradiction. □

We denote the set of all variables appearing in the effects of all of the τ-operators as V_τ. For the Sokoban example, V_τ consists of one variable, at-player. Since τ-operators only change variables in V_τ, states reachable from a state s can be characterized by assignments over V_τ. We call these the *internal states* of s. There can be multiple partitions for a given set of operators O satisfying the definition of τ-operators. For now, we are interested in finding one such partition.

Graph-Based τ-Axiom Extraction Since the dominance relation is defined over pairs of operators, one approach is to use a graph of the relationships between operators.

Definition 6. The dominance graph for a SAS+ problem Π is a directed graph $G=(V, E)$ such that (1) $V=O$, (2) (o,o') $\in E$ iff o dominates o'.

If a set of vertices V can be partitioned into $\mathcal{T}$ and $\bar{\mathcal{T}}$ such that there is an edge from every vertex in $\bar{\mathcal{T}}$ to every vertex in $\mathcal{T}$, then the operators corresponding to $\mathcal{T}$ and $\bar{\mathcal{T}}$ are τ-operators and observable operators, respectively. To determine if such a partition exists for a given graph $G=(V, E)$, we examine strong-connectedness of its complement graph $\bar{G}$. The strongly connected components (SCCs) of G can be computed in $\mathcal{O}(|V| + |E|)$ time (Tarjan 1972). Let $\bar{\mathcal{T}}$ be the vertices in a sink (with no out-going edges) SCC of the complement graph $\bar{G}$. Then there is an edge from every vertex in $\bar{\mathcal{T}}$ to every vertex in $\mathcal{T}$ on the graph G. This is because there is no edge from a vertex in $\bar{\mathcal{T}}$ to a vertex in $\mathcal{T}$ on $\bar{G}$. If a partition exists, then the τ-operators and observable operators have been found. $\mathcal{T}$ found this way is maximal w.r.t. set inclusion. Suppose we add $o \in \bar{\mathcal{T}}$ to $\mathcal{T}$. Since $\bar{\mathcal{T}}$ is a strongly connected component, there is some $o' \in \bar{\mathcal{T}}$ that has an edge to o on the complement graph $\bar{G}$, which means there is no edge from o' to o in the original graph.

Since the number of edges of a graph is at most $|V^2|$, this algorithm runs in $\mathcal{O}(|V|^2)$ time and space.

Variable-Based τ-Axiom Extraction Since our dominance relation relies on multi-valued variables, we can search over subsets of variables instead of partitions of operators, using a greedy algorithm that keeps track of all the candidates for V_τ. The initial set of candidates consists of prevar(o) for every operator o. This is because a set of variables that appear in effects of tau-operators, V_τ, if it exists, must be a subset of the set s of variables in preconditions of an operator. As we iterate through every operator o, we check if for every candidate c, either prevar(o) $\supseteq c$ or effvar(o) $\subseteq c$ is true, and if so, keep the candidate for the next iteration. The two conditions correspond to o being an observable operator and a τ-operator respectively. When a candidate c is not viable for an operator o we add prevar(o) $\cap\, c$ and effvar(o) $\cap\, c$ to the next candidates. After the last iteration, we pick a candidate c of the largest cardinality. Every operator o with effvar(o) $\subseteq c$ becomes a τ-operator. Although in the worst case there may be an exponential number of candidates (all the subsets of the initial candidates), we show in Section 6 that this algorithm usually outperforms the graph-based algorithm.

3.2 Encoding

Given a SAS+ problem $\Pi=(V, O, I, G, U=\phi, A=\phi)$, τ-operators $\mathcal{T} \subsetneq O$ with every operator in $\mathcal{T}$ dominated by every operator in $\bar{\mathcal{T}}$, we reformulate Π into a new SAS+ problem with axioms $\Pi'=(V', O', I', G', U', A')$ as follows.

- $V'=V$, $I'=I$.
- For every internal state r, we introduce a secondary variable $u_r \in U'$. We can also denote u_r as $u_{\{v_1=x_1,...,v_n=x_n\}}$, where $\{v_1=x_1, ..., v_n=x_n\}$ is the assignments on V_τ corresponding to r. Since the current state is also a reachable internal state, we introduce an axiom $[u_{\{v_1=x_1,...,v_n=x_n\}} \leftarrow v_1=x_1, ..., v_n=x_n] \in A'$. For Sokoban, this introduces an axiom $[u_{\text{at-player=loc}} \leftarrow \text{at-player=loc}]$, which means that the current location is always reachable.
- Goal conditions on variables not in V_τ remain, so $\{v_i = x | [v_i=x] \in G, v_i \notin V_\tau\} \subseteq G'$. To see if the goal conditions

on V_τ are achieved, we introduce a secondary variable g and a new goal condition $g{\in}G'$. $[g{\leftarrow}u_r]{\in}A'$ for all the internal states that satisfy the goal conditions.

- For $o{\in}\bar{\mathcal{T}}$, there is a reformulated operator $o'{\in}O'$ such that:
  - $\mathrm{pre}(o')$ consists of $\{v_i{=}x | v_i{\notin}V_\tau, [v_i{=}x]{\in}\mathrm{pre}(o)\}$ and $u_{\{v_1=x_1,...,v_n=x_n\}}$ where $\{v_1{=}x_1, ..., v_n{=}x_n\}$ are the preconditions on V_τ. Note that $o{\in}\bar{\mathcal{T}}$ always has the preconditions on V_τ. Intuitively, this means that we need to reach $\{v_1{=}x_1, ..., v_n{=}x_n\}$ to apply o. In Sokoban, while the original push had pre(push) = [at-player = from] $\wedge$ [at-stone = to] $\wedge$ clear(dest), the reformulated operator now has pre(push') = $u_{\text{at-player = from}} \wedge$ [at-stone = to] $\wedge$ clear(dest),
  - $\mathrm{eff}(o')$ keeps the original effects $\{v_i{=}x | v_i{\in}V_\tau, [v_i{=}x]{\in}\mathrm{eff}(o)\}$. Additionally, now that we have extracted the τ-operators, we only check whether a certain value for V_τ is reachable instead of actually modifying the variables. Hence, once o' is applied, it needs $\{v_i{=}x | [v_i{=}x]{\in}\mathrm{pre}(o), v_i{\in}V_\tau, v_i{\notin}\mathrm{effvar}(o')\}$ as its effect to "restore" values for V_τ.
- For every pair of internal states r, r', if there exists $o{\in}\mathcal{T}$ with $r'{=}o(r)$ (with respect to variables on V_τ), we introduce an axiom $[u_{r'}{\leftarrow}u_r, v_{n+1}{=}x_{n+1}, ..., v_m{=}x_m]{\in}A'$, where $v_{n+1}, ..., v_m$ are preconditions not in V_τ. For Sokoban, this introduces the axiom $[u_{\text{at-player=to}}{\leftarrow}u_{\text{at-player=from}}, \text{clear(to)}, \text{connected(from,to)}]$.

We show that this encoding is correct.

Lemma 1. For every state $A'(s)$ in Π', $u_r{=}1$ if and only if an internal state r is reachable from s in the original problem Π using only τ-operators.

Proof Sketch. We show this by induction. For $\{v_1{=}x_1, ..., v_n{=}x_n\}$ which is already true in s, the lemma holds due to the axiom $u_{\{v_1=x_1,...,v_n=x_n\}}{\leftarrow}v_1{=}x_1, ..., v_n{=}x_n$. If the lemma holds for every internal state r reachable in less than k steps, it also holds for every r' achievable in less than k+1 steps because of $[u_{r'}{\leftarrow}u_r, v_{n+1}{=}x_{n+1}, ..., v_m{=}x_m]$. □

Lemma 2. Given a state s, an applicable sequence τ^*o, and the reformulated operator o', $\tau^*o(s){=}o'(s)$. (*Proof*: immediate from construction of o'.)

Theorem 3. Π' has a plan if and only if Π has a plan.

Proof Sketch. If Π has a plan p, a reformulated plan p' with all τ-operators removed is a plan for Π'. For o' in p', $u_r{\in}$ $\mathrm{pre}(o')$ is true by Lemma 1, since r is reachable using τ^* in p. By Lemma 2, the result of applying τ^*o and o' will be the same. Hence, p' obtained this way is a plan for Π'. It is easy to show that p' also achieves the goals.

Conversely, if Π' has a plan p', there also exists a plan p for Π. p has operators corresponding to p' except that the gaps between the observable operators need to be filled with τ-operators to reach the internal state r where the next observable operator o is applicable. Since u_r is true when applying o' in p', by Lemma 1, there is always a sequence of τ^* that leads to r. By Lemma 2, the result of applying τ^*o and o' will be the same. By Theorem 1, the choice of τ^* doesn't matter. Hence, p is a plan for Π □

4 ϵ-Group Axioms

Our next method identifies a class of redundant fluents whose values can be determined from the other fluents, and express them as secondary variables. For example, in blocks, whether or not block a's surface is clear can be inferred from the on relationships among the blocks (clear(a) ←not holding(a), not on(b,a)...).

We now propose a new algorithm which transforms "inessential" variables such as clear(a) into secondary variables whose values are determined by a new type of axiom, *ϵ-axioms*. Since identifying all inessential variables is computationally too hard, we capture a particular subset of those variables using exactly-1 invariants (ϵ-group) which has been used in previous works such as (Edelkamp and Helmert 2000) and (Alcázar and Torralba 2015).

Definition 7. A set of fluents g is an *ϵ-group* iff exactly one member of g is true in every reachable state.

Every SAS+ variable is an ϵ-group, since variables have exactly one value at a time, e.g., in blocks variables for the locations for the block capture the fact that they must be at exactly one location at time. However, some ϵ-groups are not captured by variables. For example, g={holding(a),on(b,a),...,clear(a)} is an ϵ-group which does not correspond to a SAS+ variable. Since exactly one of g holds, and the fluents other than clear(a) are represented by other variables, we can derive an axiom [clear(a) ← not holding(a), not on(b,a),...] which determines when block a is clear. A similar idea can be found in (Haslum 2007), where clear(a) is replaced with the formula (¬holding(a) ∧ ¬on(b,a) ...) instead of axioms.

Unlike in a mutex group, set of fluents of which at most one is true in every state, at least one of the fluents in an ϵ-group must be true. We find ϵ-groups by iterating through all of the mutex groups found by the PDDL-to-SAS+ translator (Helmert 2009) and filtering as follows. A mutex group g is an ϵ-group if: (1) One of the fluents is true in the initial state; and (2) For every operator o that might delete one of the fluents in g, $|\mathrm{del}(o,g)| \leq |\mathrm{add}(o,g)|$, where add($o$,$g$) and del($o$,$g$) are a set of fluents in g that o might add and delete, respectively. The latter constraint ensures that the number of true fluents in a group never decreases.

Next, we show how to produce a problem with fewer primary variables using ϵ-axioms. We first review the widely used translation procedure from PDDL (STRIPS) to SAS+ by Helmert (2009). The candidates for the SAS+ variables are mutex groups. The translator first finds the mutex groups that can be found in tractable time, and then augments every such group with the special value "none" It then seeks to cover the set of all fluents using the fewest mutex+none groups (subsets of fluents). Since this set covering problem is NP-hard, Helmert's algorithm greedily selects the mutex+none group with the largest cardinality until all fluents are covered. On blocks, this greedy strategy chooses the locations of the blocks as variables, leaving clear(block) fluents uncovered. The resulting SAS+ problem has a binary variable for each clear(block) fluent.

Instead of generating SAS+ variables by greedily covering the fluents using mutex+none groups as in (Helmert

2009), we use the following algorithm to seek a SAS+ problem with the minimal number of primary variables, while representing inessential variables with axiom. We cover the fluents with ϵ-groups, using an integer program (IP). The key idea is that we can leave one fluent from every ϵ-group uncovered, because its value can be determined from other fluents. Our IP model has the following binary variables:

- y_f=1 if and only if a fluent f need not be covered.
- x_m=1 if and only if a mutex m is selected to be a variable.

The objective function is $min((|E| + 1) \cdot \sum_{m \in M} x_m + \sum_{f \in F} y_f)$ to minimize the number of primary variables, breaking ties in favor of less inessential variables. Note that E, M, and F denote a set of ϵ-groups, mutex groups and fluents respectively. There are 2 constraints:

- $y_f + \sum_{m \in M} a_{fm} x_m \geq 1$ for every fluent f where a_{fm} represents if a fluent f is contained in a mutex m. (a fluent have to be covered by a mutex or chosen to be left uncovered).
- $\sum_{f \in e} y_f \leq 1$ for every ϵ-group e (at most one fluent from every ϵ-group can be uncovered). Note that for a fluent f not in any ϵ-group, y_f=0.

Each uncovered fluent corresponds to a secondary variable u in the resulting SAS+ problem, whose value is determined by an ϵ-axiom [$u \leftarrow$ not x, not v,...,not z] where x,v and z are the other fluents in the same ϵ-group as u.

Combining τ-axioms and ϵ-axioms Extracting ϵ-axioms can sometimes enable τ-axioms extraction. The τ-extraction algorithms in Sec. 3 can fail to extract τ-axioms due to "superfluous" fluents in the domain. For example, consider the standard SAS+ formulation of Sokoban, with two actions,

move(from,to): pre(move) = [at-player = from] $\wedge$ clear(to), eff(move) = [at-player = to] $\wedge$ $\neg$**clear(to)** $\wedge$ **clear(from)**, and;

push(from,to,dest) : pre(push) = [at-player = from] $\wedge$ [at-stone = to] $\wedge$ clear(dest), eff(push) = [at-player = to] $\wedge$ [at-stone = dest] $\wedge$ **clear(from)** $\wedge$ $\neg$ **clear(dest)**.

In this formulation, the dominance relationship does not hold between the move/push operators because clear(to)$\in$ eff(move) (the place the player moved from will be vacant after the move). Although all of the other effects of move are in the preconditions of push, clear(to) $\notin$ pre(push), preventing the dominance condition (Def. 5) from being satisfied, so the partition into τ/observable operators fails, preventing axiom extraction. If we could somehow eliminate clear(to) from eff(move), we could establish that push dominates move. Indeed, clear(to) is inessential, since clear(to) is determined entirely by the locations of the player and stones (clear(to) $\leftarrow$ not at-player = loc, not at-stone1 = loc, ..., not at-stonen = loc). Running ϵ-axiom extraction before τ-extraction eliminates clear(to), allowing τ-extraction to succeed.

Previously, for τ-axiom extraction, we assumed the original SAS+ problem had U=ϕ and A=ϕ. To apply τ-axiom extraction to a problem with ϵ-axioms, we must deal with introduced secondary variables U and axioms A. For every $u \in U$ and internal state r, we introduce a secondary variable u^r, which represents the value of u in internal state r. For every axiom with u as its head, we make a copy of it for every u^r. A reformulated operator $o' \in \overline{\mathcal{T}}$ has u^r in pre(o') instead of u.

5 Decoding

A plan p'=$o'_1, ..., o'_m$ for the encoded problem with τ-axioms Π' is not necessarily a plan to the original problem Π. Theorem 3 guarantees we can always decode p' back to p. We now show exactly how to do this. From the initial state of Π, we try to apply operators in p' in order. Whenever the current state does not satisfy pre(o'_i), we solve a subproblem Π_{sub} with pre(o'_i) as its goal, operators restricted to τ-operators. A plan for Π_{sub} is then inserted before o'_i. Unfortunately, since Π_{sub} is a SAS+ problem itself, finding a plan for it is PSPACE-complete (Bäckström and Nebel 1995).

However, empirically, Π_{sub} is trivially solved for all our IPC benchmarks (Table 2). Our current implementation solves the axiom decoding subproblems using Fast Downward (Helmert 2006), using A* with the blind heuristic. Note that by turning τ-operators into axioms, we lose their information about cost. Thus, decoding an optimal plan for Π' does not mean we will have an optimal plan for Π'.

For a problem only with ϵ-axioms, we do not need to perform decoding at all since the possible transitions do not change from the original problem.

6 Evaluation of Axiom Extraction

We applied the τ-axiom and ϵ-axiom extraction algorithms to all IPC domains (IPC1998-IPC2014), *excluding* domains containing conditional effects, and domains with preexisting derived predicates. Out of duplicate sets of domains (e.g., pegsol-[opt11/sat11], pegsol-[opt08/sat08]), we only include one (in this case, pegsol-opt11). This results in a set of 1442 instances from 44 domains on which we executed axiom extraction (30min., 2GB /instance). Table 1 compares: (1) Default - standard Fast Downward translator (Helmert 2006), (2) Graph-based τ-axiom extraction, (3) Variable-based τ-axiom extraction, (4) ϵ-axiom extraction, and (5) ϵ-axiom followed by Variable-based τ-axiom extraction.

Overall, τ-axioms or ϵ-axioms were found in 22/44 domains, so these axioms are relevant to a substantial fraction of IPC domains. Although variable-based τ-axiom extraction has worst-case exponential runtime complexity, it is significantly faster than the polynomial-time graph-based algorithm in practice, and the graph-based algorithm fails on many instances due to memory exhaustion. On instances where both methods finished within the resource limit, they both found the same number of τ-operators. ϵ-axiom extraction, which includes a significantly modified PDDL to SAS+ translator, is slightly slower than the default translator.

Although τ-axiom extraction by itself failed on Sokoban (for the reason explained in Section 4), combination with ϵ extraction enables τ-axiom extractions. In the grid domain, where the player walks around a maze to retrieve the key to the goal, we identified the player movements as τ-operators and armempty as an inessential variable – unlike in Sokoban, these τ-axioms could be found without extracting ϵ-axioms. miconic is a elevator domain where we must transport a set of passengers from their start floors to their destination floors

Domain (30min,2GB)	Default		Graph-based τ			Var-based τ			ϵ-Axiom			ϵ-Axiom + Var-based τ		
	done	time	done	time	# τ	done	time	# τ	done	time	# ϵ	done	time	# τ
Domains with τ-axioms only														
miconic(150)	150	0.26	150	1.12	184455/150	150	0.77	184455/150	150	150	0/0	150	0.87	184455/150
pegsol-opt11(20)	20	0.17	20	0.31	660/20	20	0.28	660/20	20	20	0/0	20	0.33	660/20
satellite(36)	34	0.34	18	5.01	45/1	33	0.58	45/1	34	34	0/0	33	0.69	45/1
scanalyzer-opt11(20)	19	0.44	9	5.60	252/1	19	0.69	252/1	19	19	0/0	19	0.87	252/1
tpp(30)	30	0.37	18	3.86	8/4	30	0.56	8/4	30	30	0/0	30	0.68	8/4
Domains with ϵ-axioms only														
barman-opt14(14)	14	0.32	14	0.95	0/0	14	0.49	0/0	14	14	116/14	14	0.57	0/0
blocks(35)	35	0.17	35	0.32	0/0	35	0.27	0/0	35	35	372/35	35	0.33	0/0
depot(22)	22	0.56	19	4.29	0/0	22	0.81	0/0	22	22	824/22	22	1.06	0/0
driverlog(20)	20	0.24	16	1.97	0/0	20	0.38	0/0	20	20	67/20	20	0.45	0/0
freecell(80)	80	0.94	17	12.74	0/0	80	1.38	0/0	80	80	320/80	80	1.67	0/0
mystery(30)	30	0.42	20	4.59	0/0	30	0.65	0/0	30	30	22/15	30	0.77	0/0
parcprinter-opt11(20)	20	0.20	20	0.37	0/0	20	0.31	0/0	20	20	20/20	20	0.37	0/0
parking-opt14(20)	20	0.83	4	12.27	0/0	20	1.31	0/0	20	20	500/20	20	1.58	0/0
rovers(40)	40	0.46	23	3.95	0/0	40	0.65	0/0	40	40	333/40	40	0.78	0/0
storage(30)	30	0.30	20	1.83	0/0	30	0.47	0/0	30	30	810/30	30	0.56	0/0
tidybot-opt14(20)	20	None	0	None	0/0	20	None	0/0	20	20	20/20	20	None	0/0
woodworking-opt11(20)	20	0.39	20	1.27	0/0	20	0.57	0/0	20	20	359/20	20	0.67	0/0
Domains with both ϵ-axioms and τ axioms														
airport(50)	50	2.53	41	7.21	23/6	50	3.02	23/6	50	50	8275/50	50	10.71	23/6
grid(5)	5	0.63	2	8.34	200/2	5	1.08	880/5	5	5	5/5	5	1.36	880/5
gripper(20)	20	0.15	20	0.28	40/20	20	0.25	40/20	20	20	40/20	20	0.29	40/20
sokoban-opt08(30)	30	0.28	30	0.62	0/0	30	0.40	0/0	30	30	1208/30	30	1.42	3176/30
visitall-opt14(20)	20	0.20	20	0.52	80/20	20	0.32	80/20	20	20	20/20	20	0.51	80/20
all(1442)	1439	0.46	1105	2.65	185763	1438	0.71	186443	1439	1439	13311	1438	1.16	189619

Table 1: Axiom extraction results: for each method, done = number of instances where axiom extraction algorithms completed within resource limit (30min, 2GB), time = average runtime of axiom extraction algorithm(s) among instances where all the configurations completed within resource limit, # τ = sum of τ-operators found / number of instances with τ-operators, # ϵ = number of ϵ-axioms found / number of instances with ϵ-axioms. 44 Domains, 1442 instances. Axioms were found in 22/44 domains. Due to space, details are not shown for 22/44 domains where axioms were not found: childsnack-opt14(20), elevators-opt11(20), floortile-opt14(20), ged-opt14(20), hiking-opt14(20), logistics00(28), logistics98(35), maintenance-opt14-adl(5), movie(30), mprime(35), nomystery-opt11(20), openstacks-opt14(20), pathways(30), pipesworld-notankage(50), pipesworld-tankage(50), psr-small(50), schedule(150), tetris-opt14(17), thoughtful-sat14-strips(20), transport-opt14(20), trucks(30), zenotravel(20).

using 1 elevator. Operators for moving the elevator up/down were identified as τ-operators.

There are "incidental" τ-axioms which exist in some instances but don't reflect the structure of the domain, e.g., in tpp, where trucks move around to buy goods, the easiest instances contain only one truck, and our algorithm identified operators driving trucks as τ-axioms since all other operators (buying and loading goods) specify the truck's location. This no longer holds, however, when instances contain $\geq$2 trucks, because when truck2 loads goods, the location of truck1 is indeterminate.

7 Improving Planner Performance Using Extracted Axioms

The main contributions of this paper are the novel methods for extracting τ-operators and ϵ-axioms described in the previous sections. However, having extracted the axioms, a natural question is whether the axioms can be successfully exploited to improve planner performance. Thus, in this section, we experimentally evaluate the effect of integrating τ-operators and ϵ-axioms into several classes of planners.

We evaluate all 22 IPC benchmark domains (732 instances total) for which either τ-operators, ϵ-axioms, or both were extracted in *at least one instance* in the experiment in Section 6. All experiments are performed on a Xeon E5-2670 v3, 2.3GHz. and the runtime limits include all steps of problem solving, including axiom extraction, encoding, decoding, translation/parsing, and search.

7.1 Model-Based Planners

We evaluate the effect of adding axioms to *satisficing*, model-based planners. Although the fastest model-based planners are SAT-based (Rintanen, Heljanko, and Niemelä 2006), we focus on two model-based approaches with standard solvers that handle numbers, i.e., Integer Programming (IP) and Answer Set Programming (ASP), because one of the primary advantages of a model-based planner is the ability to add new constraints objectives to the cost function, facilitating extensions to the classical planning framework, (c.f., (Srivastava et al. 2007)). A standard approach to model-based planning is to translate a planning problem instance into a k-step model, where a feasible solution to the k-step model corresponds to a solution to the original planning problem with $n<k$ "steps". A standard, *sequential search strategy* first generates a 1-step model (e.g., SAT/IP model), and attempts to solve it. If it has a solution, then the system terminates. Otherwise, a 2-step model is attempted, and so on (Kautz and Selman 1992).

$\forall$ **vs. sequential (seq-) semantics and model selection** Adding axioms changes the semantics of a "step" in the k-step model. As noted by Dimopoulos et al (1997) and Rintanen et al (2006), most model-based planners, including the

base IPlan/ASPlan planners we evaluate below, use $\forall$ semantics , where each "step" in a k-step model consists of a set of actions which are independent of each other and can be executed in parallel. In contrast, *sequential* semantics (seq-semantics) adds exactly 1 action at each step in the iterative, sequential search strategy. In general, $\forall$ semantics is faster than seq-semantics, since $\forall$ semantics significantly decreases the # of iterations of the sequential search strategy.

For the axiom-reformulated models, we add a constraint which restricts the number of actions executed at each step to 1, imposing seq-semantics. This is because a single operator can have far-reaching effects on derived variables, and establishing independence with respect to all derived variables affected by multiple operators is nontrivial (future work). seq-semantics can degrade the performance of axiom-enhanced planners compared to base planners using $\forall$ semantics because seq-semantics usually requires more iterations than $\forall$ semantics. However, with axioms, a single action can trigger many axioms (implied actions), so it is possible that solving axiom-reformulated models can require significantly fewer steps than a model without axioms. Thus, there is a trade-off between loss of explicit parallel execution semantics ($\forall$ vs. seq) and the implied parallel execution semantics due to axioms. In the experiments below, the base planners (IPlan, ASPlan) use $\forall$-semantics.

On instances where axioms are not extracted, it is better to use $\forall$ semantics. Thus, the IP/ASP models below with a "$^+$" postfix in the configuration name use a *$\forall$/seq model selection policy* : First, execute τ-axiom extraction. If τ-axioms are discovered, then use the axiom enhanced, seq-semantics model, otherwise, use the standard base model (with $\forall$ semantics), e.g., IPlan$(\tau,\epsilon)^+$ first runs axiom extraction and if τ-axioms are found, then it uses the axiom-reformulated seq-semantic IPlan(τ,ϵ) model, otherwise, it uses the standard $\forall$-semantic IPlan model.

The model selection policy can make mistakes, resulting in poor performance compared to the "correct" model selection for each problem. Thus, we also evaluate an *ideal model selection policy* which selects the best model out of: (1) the base planner ($\forall$ semantics), (2) τ-enhanced (seq-semantics), (3) ϵ-enhanced (seq), (4) τ,ϵ-enhanced (seq). It retrospectively selects the best result out of the 4 models, representing an upper bound achievable using perfect model selection.

Integer Programming Our baseline IP-based planner, which we call IPlan (Miura and Fukunaga 2017), is based on the state change variable model, where the variables represent changes in state values (Vossen et al. 1999). The state change variable model was the basis of Optiplan (van den Briel and Kambhampati 2005), which incorporated variable elimination based on a relaxed planning graph. IPlan improves upon Optiplan by adding some additional, straightforward mutex constraints.

Axioms can be integrated into an IP-based model, as follows: Given an axiom of the form $a \leftarrow b_1, ..., b_m, \text{not}c_1, ..., \text{not}c_m$, the corresponding normal logic program (NLP) P_t for time step t, is the rule:

$$x^{\text{sat}}_{a,t} \leftarrow x^{\text{sat}}_{b_1,t}, ..., x^{\text{sat}}_{b_n,t}, \text{not}x^{\text{sat}}_{c_1,t}, ..., \text{not}x^{\text{sat}}_{c_m,t} \quad (1)$$

where $x^{\text{sat}}_{f,t}$ is an auxiliary boolean variable which denotes whether f is true at step t. The models for P_t correspond to the truth values for the derived variables. Each NLP P_t is then translated to a integer program (IP) using the method by Liu, Janhunen, and Niemeiä (2012), and these linear constraints are added to the IPlan model.

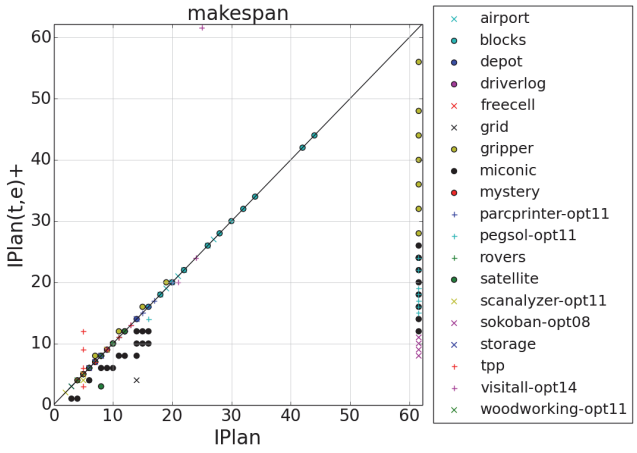

Figure 2: Makespan (n) comparison per instance for IPlan vs. IPlan$(\tau,\epsilon)^+$.

We compared the following: (1) SCV (a reimplementation of Optiplan (van den Briel and Kambhampati 2005)) [1], (2) IPlan (IPlan with $\forall$ semantics), (3) IPlanS (IPlan with seq-semantics), (4) IPlan(τ) (τ-axiom extraction only), (5) IPlan$(\tau)^+$ (τ-axiom extraction + $\forall$/seq-model selection), (6) IPlan(ϵ) (ϵ-axiom extraction only), (7) IPlan(τ,ϵ) (both ϵ and τ-axiom extraction), (8) IPlan$(\tau,\epsilon)^+$ (IPlan(τ,ϵ) + $\forall$/seq model selection), and (9) IPlan$(\tau,\epsilon)^*$ (ideal model selection). The IP models are solved using Gurobi Optimizer 6.5.0 (single-threaded, 5min., 2GB RAM per problem.

Table 2 shows that IPlan$(\tau,\epsilon)^+$ achieves significantly higher coverage than the baseline models (SCV, IPlan, IPlanS). Switching from $\forall$ semantics (IPlan) to seq-semantics (IPlanS) significantly degrades performance, but the performance boost due to axioms more than compensates for this loss. Improvements due to axioms are particularly apparent on gripper, miconic, pegsol and Sokoban. Although most of the improvements are due to τ-axioms, the utility of ϵ-axioms is shown by the improved coverage of IPlan(ϵ) on blocks compared to the baseline. There is strong synergy between ϵ-extraction and τ-extraction (Sec. 4), and as a result, IPlan(τ,ϵ) significantly improves coverage on gripper and Sokoban compared to IPlan(ϵ) and IPlan(τ). The coverage gap between IPlan$(\tau,\epsilon)^+$ and IPlan$(\tau,\epsilon)^*$ is only 6, so the $\forall$/seq-model selection is quite successful for IPlan.

As shown in Fig. 2 and Table 2, the value for n (the step at which the solution was found) is usually smaller with axioms than without. On the other hand, since IPlan with axioms is constrained to use sequential semantics (1 action/layer), while IPlan uses $\forall$ semantics (multiple inde-

[1] We could not use the original Optiplan code because it depends on an outdated ILOG Concert API which is no longer supported in recent CPLEX releases.

Domain	SCV		Iplan		IplanS		Iplan(ϵ)		Iplan(τ)			Iplan(τ,ϵ)			Iplan(τ)$^+$			Iplan(τ,ϵ)$^+$			(τ,ϵ)*
	#	n	#	n	#	n	#	n	#	n	d	#	n	d	#	n	d	#	n	d	#
Domains with τ-axioms only																					
miconic(150)	28	9.04	29	9.04	25	10.44	25	10.44	**65**	5.80	1.28	**65**	5.80	1.28	**65**	5.80	1.28	**65**	5.80	1.28	65
pegsol-opt11(20)	1	16.00	1	16.00	2	16.00	2	16.00	**17**	14.00	1.11	**17**	14.00	1.12	16	14.00	1.11	**17**	14.00	1.18	17
satellite(36)	**8**	8.67	**8**	8.67	3	11.00	3	11.00	3	9.00	0.27	3	9.00	0.26	**8**	7.00	0.24	**8**	7.00	0.26	8
scanalyzer-opt11(20)	**11**	4.33	**11**	4.33	4	7.67	3	7.67	3	7.33	0.22	3	7.33	0.22	**11**	4.00	0.21	**11**	4.00	0.22	11
tpp(30)	10	5.40	**12**	5.40	5	11.40	5	11.40	5	9.80	0.40	5	9.80	0.39	**12**	7.40	0.38	**12**	7.40	0.40	12
Domains with ϵ axioms only																					
barman-opt14(14)	**0**	None	**0**	None	**0**	None	**0**	None	**0**	None	None	**0**	None	None	**0**	None	None	**0**	None	None	0
blocks(35)	16	15.75	28	15.75	28	15.75	**32**	15.75	28	15.75	None	**32**	15.75	None	28	15.75	None	28	15.75	None	32
depot(22)	7	6.50	**11**	6.50	2	12.50	2	12.50	2	12.50	None	2	12.50	None	**11**	6.50	None	**11**	6.50	None	11
driverlog(20)	**11**	6.00	**11**	6.00	4	10.75	4	10.75	4	10.75	None	4	10.75	None	**11**	6.00	None	**11**	6.00	None	11
freecell(80)	**18**	6.00	**18**	6.00	7	9.14	9	9.14	9	9.14	None	10	9.14	None	**18**	6.00	None	**18**	6.00	None	18
mystery(30)	13	5.09	**16**	5.09	14	5.45	13	5.45	12	5.45	None	12	5.45	None	**16**	5.09	None	**16**	5.09	None	16
parcprinter-opt11(20)	**20**	11.18	**20**	11.18	11	27.64	11	27.64	12	27.64	None	11	27.64	None	**20**	11.18	None	**20**	11.18	None	20
parking-opt14(20)	**0**	None	**0**	None	**0**	None	**0**	None	**0**	None	None	**0**	None	None	**0**	None	None	**0**	None	None	0
rovers(40)	18	5.00	**21**	5.00	4	9.25	4	9.25	4	9.25	None	4	9.25	None	**21**	5.00	None	**21**	5.00	None	21
storage(30)	**9**	6.14	**9**	6.14	7	6.71	7	6.71	7	6.71	None	7	6.71	None	**9**	6.14	None	**9**	6.14	None	9
tidybot-opt14(20)	**0**	None	**0**	None	**0**	None	**0**	None	**0**	None	None	**0**	None	None	**0**	None	None	**0**	None	None	0
woodworking-opt11(20)	**20**	3.10	**20**	3.10	10	14.30	10	14.30	10	14.30	None	10	14.30	None	**20**	3.10	None	**20**	3.10	None	20
Domains with both ϵ axioms and τ axioms																					
airport(50)	13	15.14	**15**	15.14	7	16.29	7	16.29	7	16.29	0.05	7	16.29	0.06	**15**	15.14	0.05	**15**	15.14	0.06	15
grid(5)	0	None	**1**	None	**1**	None	**1**	None	**1**	None	0.55	**1**	None	0.56	**1**	None	0.54	**1**	None	0.56	1
gripper(20)	4	9.00	4	9.00	2	14.00	2	14.00	8	10.00	1.37	**12**	10.00	2.01	8	10.00	1.29	**12**	10.00	1.99	12
sokoban-opt08(30)	0	None	0	None	0	None	1	None	0	None	None	**5**	None	1.00	0	None	None	**5**	None	1.00	5
visitall-opt14(20)	3	22.50	3	22.50	**4**	22.50	**4**	22.50	3	22.00	0.11	2	22.00	0.08	3	22.00	0.12	2	22.00	0.09	4
all(732)	210	9.01	238	9.01	140	12.93	145	12.93	200	11.75	1.09	212	11.75	1.18	293	8.36	1.09	302	8.36	1.19	308

Table 2: IPlan Results: for each method, #= number of instances solved within resource limit (5min, 2GB), n = avg. makespan (the first step the solution was found) for instances solved by all configurations, decode = avg. time needed for decoding

pendent actions per step), it is possible for IPlan to solve problems in fewer steps than IPlan+axioms (e.g., in the gripper domain, where n is larger for IPlan with axioms among problems solved by both IPlan with/without axioms).

Answer Set Programming We use ASPlan (Miura and Fukunaga 2017) which uses a SAS+ adopted encoding of (Gebser, Kaufmann, and Schaub 2012).

The extracted axioms are directly expressible in ASP. We use `clingo4.5.4`, a state-of-the-art ASP solver (Potassco 2016) to solve the ASP models. We compared (1) ASPlan (base), (2) ASPlanS, (3) ASPlan(τ), (4) ASPlan(τ,ϵ), (5)ASPlan(ϵ), (6) ASPlan$(\tau,\epsilon)^+$, and (7) ASPlan$(\tau,\epsilon)^*$. These configurations are analogous to the IPlan configurations evaluated above.

As shown in Table 3, ASPlan$(\tau,\epsilon)^+$ solved more problems than ASPlan on the grid, miconic, pegsol, Sokoban domains. In addition, ASPlan(ϵ) solved more problems than both ASPlan and ASPlan$(\tau,\epsilon)^+$ on blocks, freecell, mystery, visitall. ASPlan$(\tau,\epsilon)^*$ has a coverage of 263, significantly higher than all other configurations. This indicates that improving the model selection heuristic can result in better usage of axiom extraction, and is a direction for future work.

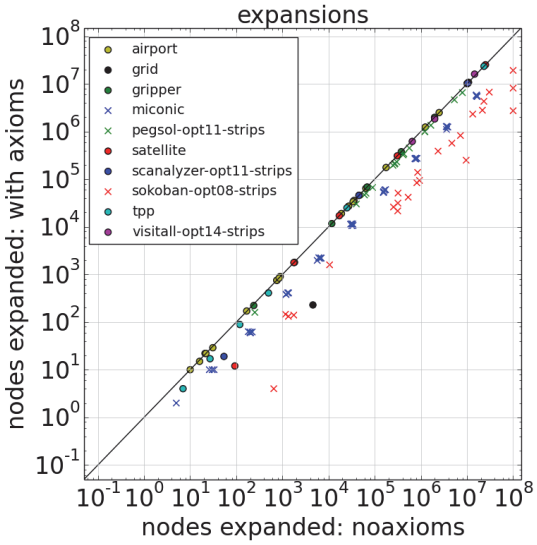

Figure 3: Number of nodes expanded by Fast Downward A* (blind heuristic) with and without extracted axioms.

7.2 Forward Search Planning

We ran Fast Downward (Helmert 2006) planner (30 min., 2GB RAM per instance) using A* search with a blind heuristic with and without axioms on the domains with τ-axioms. As shown in Figure 3, the number of nodes expanded for sokoban-opt08, miconic, and grid are smaller with extracted axioms than the number of nodes expanded without axioms, i.e., *search efficiency improved with axioms*. However, so far, we have not found a forward search planner which handles derived predicates and outperforms the baseline search with respect to *runtime* when run on our τ+ϵ axiom-reformulated problems. Due to nontrivial interactions between axioms and heuristics, developing a forward search-based planner which runs faster using an axiom-reformulated domain is a direction for future work.[2]

[2]Thiébaux, Hoffmann, and Nebel (2005) compare the performance of directly supporting derived predicates vs. using compiled axioms (which adds an exponential number of variables to the domain or results in a significantly longer plan); this is different from comparing a standard domain vs the axiom-reformulated version of that domain. Ivankovic and Haslum (2015b) evaluates the # of instances solved vs. the # of search steps.

Domain (5min, 2GB)	ASPlan		ASPlanS		ASPlan(ϵ)		ASPlan(τ, ϵ)			ASPlan$(\tau, \epsilon)^+$			$(\tau, \epsilon)^*$
	#	n	#	n	#	n	#	n	decode	#	n	decode	#
Domains with both ϵ axioms and τ axioms													
miconic(150)	35	9.90	29	11.69	29	11.69	**40**	6.66	0.68	**40**	6.66	0.69	40
pegsol-opt11(20)	3	18.50	3	18.50	3	18.50	**4**	14.50	0.88	**4**	14.50	0.88	4
satellite(36)	**11**	9.00	4	12.50	4	12.50	4	11.00	0.25	**11**	7.75	0.26	11
scanalyzer-opt11(20)	**5**	5.50	3	7.50	3	7.50	3	7.00	0.55	**5**	5.00	0.55	5
tpp(30)	15	5.40	5	11.40	5	11.40	5	9.80	0.28	**16**	7.40	0.28	16
Domains with ϵ-axioms only													
barman-opt14(14)	**0**	None	**0**	None	**0**	None	**0**	None	None	**0**	None	None	0
blocks(35)	18	16.78	18	16.78	29	16.78	**30**	16.78	None	18	16.78	None	30
depot(22)	**9**	6.50	2	12.50	2	12.50	2	12.50	None	**9**	6.50	None	9
driverlog(20)	**14**	6.86	7	13.71	7	13.71	7	13.71	None	**14**	6.86	None	14
freecell(80)	1	5.00	**7**	8.00	**7**	8.00	**7**	8.00	None	1	5.00	None	7
mystery(30)	9	5.11	**16**	5.56	**16**	5.56	**16**	5.56	None	9	5.11	None	16
parcprinter-opt11(20)	**20**	9.50	4	17.25	4	17.25	4	17.25	None	**20**	9.50	None	20
parking-opt14(20)	**0**	None	**0**	None	**0**	None	**0**	None	None	**0**	None	None	0
rovers(40)	**23**	5.00	4	9.25	4	9.25	4	9.25	None	**23**	5.00	None	23
storage(30)	**14**	7.60	10	8.80	11	8.80	11	8.80	None	**14**	7.60	None	14
tidybot-opt14(20)	**0**	None	**0**	None	**0**	None	**0**	None	None	**0**	None	None	0
woodworking-opt11(20)	**20**	3.00	3	11.50	2	11.50	2	11.50	None	**20**	3.00	None	20
Domains with both ϵ-axioms and τ-axioms													
airport(50)	**18**	17.09	11	24.73	11	24.73	11	24.73	0.06	**18**	17.09	0.06	18
grid(5)	0	None	2	None	2	None	**3**	None	1.06	**3**	None	1.05	3
gripper(20)	**3**	9.00	2	14.00	2	14.00	**3**	10.00	0.59	**3**	10.00	0.57	3
sokoban-opt08(30)	3	45.00	4	45.00	4	45.00	**6**	10.00	1.25	**6**	10.00	1.22	6
visitall-opt14(20)	**4**	23.33	**4**	23.33	**4**	23.33	3	22.67	0.10	3	22.67	0.10	4
sum(732)	225	11.55	138	14.42	149	14.42	165	12.01	0.65	237	9.83	0.65	263

Table 3: ASPlan Results: for each method, #= number of instances solved within resource limit (5min, 2GB), n = avg. makespan (the first step the solution was found) for instances solved by all configurations, decode = avg. time needed for decoding

7.3 Other Planners

There are few existing satisficing that fully handle axioms as defined in Def. 1. Although our definition of axioms in Def. 1 is a general one used by other work (Helmert 2006; Ivankovic and Haslum 2015b), (e.g., Fast Downward supports our definition), The version of PDDL used in IPC-4 only supported a more limited form which does not allow negated derived predicates to appear in other axioms. Thus planners such as FF-x (Thiébaux, Hoffmann, and Nebel 2005) and LPG-td (Gerevini, Saetti, and Serina 2011) from that time cannot handle ϵ-axioms, and as a result, some τ-axioms can not be extracted (Sec. 4).

For example, LPG-td is a local-search based planner which handles derived predicates (Gerevini, Saetti, and Serina 2011). Due to the limitations explained above, the ϵ-axioms can not be extracted, and τ-axioms can only be partially extracted. Nevertheless, we evaluated the performance of LPG-td on the (partial) τ-axiom-reformulated domains vs the performance of LPG-td on the original domains. This usually results in slightly worse performance than the base planner, which is to be expected. Despite this, LPG-td coverage increases from 3 to 20 on pegsol-sat11, and 4 to 5 on tpp, so partial τ-extraction is useful on some domains. FF-X is a version of Fast Forward (Hoffmann and Nebel 2001) which supports axioms. We also tried evaluating FF-X, with partial τ-extraction, but did not obtain any positive results.

8 Conclusions

We proposed novel methods for automatically identifying and extracting two classes of axioms, τ-axioms and ϵ-axioms, from standard SAS+ models without explicit axioms. While previous work focused on axioms as a domain modeling tool which improves expressivity , our work shows that starting from a standard domain model which does not include axioms, we can successfully extract derived predicates in a standard domain without axioms and then exploit this to improve search performance. The domains on which our system improved planner performance were not originally designed using derived predicates and axioms, showing that natural, PDDL models sometimes contain derived variables, and our automated reformulation results in a "better" model than the original, human-designed models.

References

Alcázar, V., and Torralba, Á. 2015. A reminder about the importance of computing and exploiting invariants in planning. In *Proceedings of the Twenty-Fifth International Conference on Automated Planning and Scheduling*, 2–6.

Apt, K. R.; Blair, H. A.; and Walker, A. 1988. Towards a theory of declarative knowledge. *Foundations of Deductive Databases and Logic Programming* 89–148.

Bäckström, C., and Nebel, B. 1995. Complexity results for SAS+ planning. *Computational Intelligence* 11:625–656.

Barrett, A.; Christianson, D.; Friedman, M.; Kwok, C.;

Golden, K.; Penberthy, S.; Sun, Y.; and Weld, D. 1995. UCPOP users manual. Technical Report TR93-09-06d, University of Washington, CS Department.

Coles, A., and Smith, A. 2007. Marvin: A heuristic search planner with online macro-action learning. *Journal of Artificial Intelligence Research* 28:119–156.

Dimopoulos, Y.; Nebel, B.; and Koehler, J. 1997. Encoding planning problems in nonmonotonic logic programs. In *Proceedings of the 4th European Conference on Planning: Recent Advances in AI Planning*, 169–181.

Edelkamp, S., and Helmert, M. 2000. Exhibiting knowledge in planning problems to minimize state encoding length. In *Proceedings of 5th European Conference on Planning: Recent Advances in AI Planning*, 135–147.

Edelkamp, S., and Hoffmann, J. 2004. PDDL2.2: The language for the classical part of the 4th International Planning competition. Technical Report Technical Report 195, Albert-Ludwigs-Universitt Freiburg, Institut fr Informatik, 2004.

Gebser, M.; Kaufmann, R.; and Schaub, T. 2012. Gearing up for effective asp planning. In *Correct Reasoning*. Springer. 296–310.

Gelfond, M., and Kahl, Y. 2014. *Knowledge Representation, Reasoning, and the Design of Intelligent Agents: The Answer-Set Programming Approach*. Cambridge University Press.

Gerevini, A.; 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.

Haslum, P. 2007. Reducing accidental complexity in planning problems. In *Proceedings of the 20th international joint conference on Artifical intelligence*, 1898–1903.

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(5):503–535.

Hoffmann, J., and Nebel, B. 2001. The ff planning system: Fast plan generation through heuristic search. *Journal of Artificial Intelligence Research* 14:253–302.

Ivankovic, F., and Haslum, P. 2015a. Code supplement for "optimal planning with axioms. http://users.cecs.anu.edu.au/~patrik/tmp/fd-axiom-aware.tar.gz.

Ivankovic, F., and Haslum, P. 2015b. Optimal planning with axioms. In *Proceedings of the 24th International Conference on Artificial Intelligence*, 1580–1586.

Kautz, H., and Selman, B. 1992. Planning as satisfiability. In *Proceedings of the 10th European Conference on Artificial Intelligence*, 359–363.

Liu, G.; Janhunen, T.; and Niemeiä, I. 2012. Answer set programming via mixed integer programming. In *Proceedings of the Thirteenth International Conference on Principles of Knowledge Representation and Reasoning*, 32–42.

Manna, Z., and Waldinger, R. J. 1987. How to clear a block: A theory of plans. *Journal of Automated Reasoning* 3(4):343–377.

Miura, S., and Fukunaga, A. 2017. Axioms in model-based planners. https://arxiv.org/abs/1703.03916.

Potassco. 2016. The University of Potsdam Answer Set Programming collection (Potassco). http://potassco.sourceforge.net/.

Rintanen, J.; Heljanko, K.; and Niemelä, I. 2006. Planning as satisfiability: parallel plans and algorithms for plan search. *Artificial Intelligence* 170(12-13):1031–1080.

Srivastava, B.; Nguyen, T. A.; Gerevini, A.; Kambhampati, S.; Do, M. B.; and Serina, I. 2007. Domain independent approaches for finding diverse plans. In *Proceedings of the 20th International Joint Conference on Artificial Intelligence*, 2016–2022.

Tarjan, R. 1972. Depth-first search and linear graph algorithms. *SIAM Journal on Computing* 1(2):146–160.

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

van den Briel, M. H. L., and Kambhampati, S. 2005. Optiplan: Unifying ip-based and graph-based planning. *Journal of Aritificial Intelligence* 24(1):919–931.

Vossen, T.; Ball, M. O.; Lotem, A.; and Nau, D. S. 1999. On the use of integer programming models in AI planning. In *Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence*, 304–309.