Automatic Extraction of Axioms for Planning

Submission 2332

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 an IP-based planner.

1 Introduction

Planning is the problem of finding a sequence of actions which satisfy a set of goals, given an initial state and a set of operators, where each operator has a set of preconditions and effects. Currently, in the most commonly studied classical planning models, all changes to the world are the direct effects of some operator. 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*, which encode logical rules defining how the derived predicates follow from basic variables. Many planners through the years have supported various forms of derived predicates since relatively early systems (Manna and Waldinger 1987; Barrett et al. 1995). The widely used PDDL modeling language has supported axioms which specify derived predicates as a logic program with negation-as-failure semantics since version 2.2 (Edelkamp and Hoffmann. 2004)

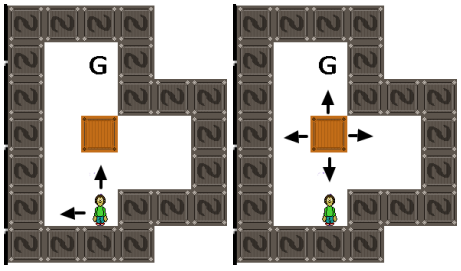
Previous work on derived predicates and axioms for planning has focused on the advantages of expressivity (compactness) of domain modeling using axioms, (Thiébaux, Hoffmann, and Nebel 2005), as well as search algorithms which are aware of axioms (Ivankovic and Haslum 2015b). Previous work has shown that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

For example, consider the well-known 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. A push operator lets the player push a box in one direction, while a move operator 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 refomulated 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(loc) ← reachable(from), clear(loc)

Intuitively, the first axiom means that the current location of the player is reachable. The second axiom means a location next to a reachable location is also reachable. Figure1 illustrates the search space with and without axioms. Note that in the domain with axioms, a solution is a sequence consisting solely of push operators. The move operators are implied by the push operators, i.e., the player jumps into position to push a box and the axioms allows us to assume there was some way to get to that position.



(a) Sokoban (b) Sokoban with Axioms

Figure 1: The search space of the Sokoban domain with and without axioms

While previous work focused on axioms as expressive domain modelling constructs used by human domain modellers, 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-enhanced 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 reformulated domain without the move operator. After finding a plan which achieves the goals in this reformulated state space (i.e. push(4-3,3-3),push(3-3,2-3)), we then decode the plan to generate a valid plan for the original domain (move(5-3,4-3),push(4-3,3-3),push(3-3,2-3)).

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 tau operators, which are "auxiliary" operators. Given a list observable operators to execute, a set of tau 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 tau 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 the fluents must be true at all times. We generate axioms which describe the truth value of one member of such a "exactly-1" group in terms of the other members of the group. The procedure for converting solutions to the reformulated problems into solutions for the original domains (decoding) is described in Sec. 5.

We show that using our methods, axioms can be extracted from a number of standard IPC benchmark problems (Sec. 6. We then show that the axioms extracted from IPC standard benchmarks can then be exploited to speed up the search for a satisficing plan. We embed the axioms as new constraints in a model-based planner, and show that that this results in significantly higher coverage on several standard IPC domains (Sec. 7). We also show that the reformulated problems can reduce search spaces for A* forward search based planning.

2 Review of SAS+ and Axioms

We adopt the definition of axiom-enhanced SAS+ by Ivankovic and Haslum (2015b).

Definition 1. A SAS+ proplem 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). 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 of the 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 stratisfiable 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. In practice, the SAS+ representation of a problem is usually obtained by translation from the PDDL modeling language, using an algorithm such as (Helmert 2009).

3 tau-Axiom Extraction: Representing Internal Transitions as Axioms

We extract operators whose effects become irrelevant given some other operator, and express this structure as axioms. We call these operators tau operators after tau transition in labeled transition systems (c.f, Fernandez and Mounier 1991).

Definition 2. Let $T(s)$ be a set of states reachable from state s using only the operators in $T{\subset}O$, and let $\overline{T}{=}O \setminus T$. We call a set of operators T the *tau operators*, if and only if for every state s, pair of states $s', s''{\in}T(s)$ ($s'{\neq}s''$) , operator $o{\in}\overline{T}$ applicable in both s' and s'', $o(s'){=}o(s'')$. We call $\overline{T}$ the *observable operators*.

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 move operators only change at-player, the effects of move operators are irrelevant once one of the push operators applied specifies the value for at-player. Thus, in this example, the move and push operators are the tau and observable operators respectively.

As in this example, we can use the multi-valued semantics of a SAS+ problem to identify tau operators using the notion

of dominance.

Definition 3. 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).

Theorem 1. Given a set of operators T, if every operator in T is dominated by every operator in $\overline{T}$, then T is the set of tau operators. [*Proof:* See Supplement Sec. 1]

We denote the set of all variables appearing in the effects of all of the tau operators as V_{tau}. For the Sokoban example, V_{tau} consists of one variable at-player. Note that there can be multiple partitions for a given set of operators O that satisfies the difinition of tau operators. For now we are interested in finding any one of such partition.

Graph-Based tau-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 4. 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 T and $\overline{T}$ such that there is an edge from every vertex in $\overline{T}$ to every vertex in T, then the operators corresponding to T and $\overline{T}$ are tau 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 $\overline{G}$. This is similar to checking whether a given undirected graph can be partitioned into a biclique (Fleischner et al. 2009). The strongly connected components (SCCs) of G can be computed in $\mathcal{O}(|V| + |E|)$ time using Tarjan's algorithm (Tarjan 1972). Let $\overline{T}$ be the vertices in a sink (with no out-going edges) SCC of the complement graph $\overline{G}$. Then there is an edge from every vertex in $\overline{T}$ to every vertex in T on the graph G. This is because there is no edge from a vertex in $\overline{T}$ to a vertex in T on $\overline{G}$. If a partition exists, then the tau and observable operators have been found. T found this way is maximal w.r.t. set inclusion. Suppose we add $o{\in}\overline{T}$ to T. Since $\overline{T}$ is a strongly connected component, there is some $o'{\in}\overline{T}$ that has an edge to o on the complement graph $\overline{G}$, 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 tau-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 a greedy algorithm that keeps track of all the candidates for V_{tau}. The initial set of candidates consists of prevar(o) for every operator o. As we iterate through every operator o, we check if for a 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 tau operaotr 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 tau operator. Although we may have exponential number of candidates (all the subsets of the initial candidates) in the worst case, this algorithm usually outperforms the graph-based algorithm as we will show later.

3.1 Encoding

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

- $V'{=}V$, $I'{=}I$.
- For $v_1, ..., v_n{\in}V_{\text{tau}}$, $x_1{\in}D(v_1), ..., x_n{\in}D(v_n)$, we introduce a secondary variable $a_{\{v_1=x_1,...,v_n=x_n\}}{\in}U'$, which corresponds to a (internal) state reachable using tau operators. $[a_{\{v_1=x_1,...,v_n=x_n\}}{\leftarrow}v_1{=}x_1, ..., v_n{=}x_n]{\in}A'$. For Sokoban, this introduces an axiom $[a_{\text{at-player=loc}}{\leftarrow}$ at-player=loc].
- For $[v_i{=}x]{\in}G$, if $v_i{\notin}V_{\text{tau}}$, $[v_i{=}x]{\in}G'$. Otherwise, we introduce a secondary variable $[g{=}1]{\in}G'$, which represents that all of the goal conditions on $v_i{\in}V_{\text{tau}}$ are true. $[g{\leftarrow}a_{\{v_1=x_1,...,v_n=x_n\}}]{\in}A'$ for all the internal states that satisfy the goal conditions.
- For $o{\in}\overline{T}$, a reformulated operator $o'{\in}O'$ such that:
  - $\text{pre}(o')$ consists of $\{v_i{=}x|v_i{\in}V_{\text{tau}}, [v_i{=}x]{\in}\text{pre}(o)\}$ and $a_{\{v_1=x_1,...,v_n=x_n\}}$ where $\{v_1{=}x_1, ..., v_n{=}x_n\}$ are the preconditions on V_{tau}. In Sokoban, $a_{\text{at-player=from}}$ replaces [at-player=from] in pre(push).
  - $\text{eff}(o'){=}\{v_i{=}x|v_i{\in}V_{\text{tau}}, [v_i{=}x]{\in}\text{eff}(o)\} \cup \{v_i{=}x|[v_i{=}x]{\in}\text{pre}(o), v_i{\notin}\text{effvar}(o')\}$.
- For $o{\in}T$ and $a_{\{v_1=x_1,...,v_n=x_n\}}$, if o is applicable in a state corresponding to $a_{\{v_1=x_1,...,v_n=x_n\}}$, $[a_{\{v_1=x'_1,...,v_n=x'_n\}}{\leftarrow}a_{\{v_1=x_1,...,v_n=x_n\}}, v_{n+1}{=}x_{n+1}, ..., v_m{=}x_m]{\in}A'$, where $v_{n+1}, ..., v_m$ are preconditions not in V_{tau}, and $\{v_1{=}x'_1, ..., v_n{=}x'_n\}$ is the result of applying o. For Sokoban, this introduces the axiom $[a_{\text{at-player=to}}{\leftarrow}a_{\text{at-player=from}}, \text{clear(to)}]$
  This encoding has the following properties.

Lemma 1. For every state $A(s)$ in Π', $a_{\{v_1=x_1,...,v_n=x_n\}}{=}1$ if and only if $\{v_1{=}x_1, ..., v_n{=}x_n\}$ is achievable from s in the original problem Π using only tau operators. [*Proof:* See Supplement Sec. 1]

Theorem 2. Π' has a plan if and only if Π has a plan. [*Proof:* See Supplement Sec. 1]

4 Exactly-1 Group Axioms

Our next method for extracting a second class of axioms is motivated by the fact the algorithms described above (Sec. 3) can fail to extract tau-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)**.

The dominance relationship does not hold between the move/push operators in the formulation because clear(loc)$\in$ 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(loc) $\notin$ pre(push), preventing the dominance condition (Definition 3) from being satisfied, so the partition into tau/observable operators fails, preventing axiom extraction.

If we could somehow eliminate clear(loc) from eff(move), we could establish that push dominates move. Intuitively, clear(loc) is redundunt, since clear(loc) is determined entirely by the locations of the player and stones, which are the more "essential" features of the domain. Below, we show how to convert "inessential" variables into secondary variables, extracting them into another type of axiom, and eliminating them from the effects of actions.

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 (set of fluents of which at most one is true in every state). 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, the algorithm greedily selects the mutex+none group with the largest cardinality until all fluents are covered. On Sokoban, this greedy strategy chooses the locations of the player and stones as variables, leaving clear(loc) fluents uncovered. Since there is no way of covering clear(loc) other than with {at-player=loc, stone1=loc, ..., stonen=loc, clear(loc)}, the resulting SAS+ problem has a binary variable for each clear(loc) fluent.

We now propose a new translation algorithm which transforms "inessential" variables such as clear(loc) into secondary variables whose values are determined by a new type of axiom, *exactly-1 axioms*.

Definition 5. A set of fluents g is an *exactly-1 group* iff exactly one member of g is true in every reachable state.

Every SAS+ variable is an exactly-1 group, since variables have exactly one value at a time, e.g., in Sokoban, variables for the locations for the player and stones capture the fact that they must be at exactly one location at time. However, some exactly-1 groups are not captured by variables. Each location can be occupied by at most one object, so g={at-player=loc,stone1=loc,...,stonen=loc,clear(loc) } is an exactly-1 group which does not correspond to a SAS+ variable. Since exactly one of g holds, and the fluents other than clear(loc) are represented by other variables, we can derive an axiom [clear(loc) $\leftarrow$ not at-player=loc, not stone1=loc,...,not stonen=loc] which determines when loc is vacant.

Unlike in a mutex group, at least one of the fluents in an exactly-1 group must be true. We find exactly-1 groups by iterating through all of the mutex groups found by the PDDL-to-SAS+ translator (Helmert 2009) and filtering them according to the folowing criterion. A mutex group g is an exactly-1 group if: (1) At least 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, $|\text{del}(o,g)| \leq |\text{add}(o,g)|$, 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, instead of generating SAS+ variables by greedily covering the fluents using mutex+none groups as (Helmert 2009), we use the following algorithm to seek a SAS+ problem with the number of primary variables as small as possible, while representing inessential variables with axiom.

Instead of covering the fluents with mutex+none groups using a greedy algorithm, we cover the fluents with exactly-1 groups, using an integer program (IP). The key idea is that we can leave one fluent from every exactly-1 group uncovered, because its value can be determined from other fluents. Our IP model has the following binary variables:

- y_i=1 if and only if i-th fluent need not be covered.
- x_j=1 if and only if j-th mutex is selected to be a variable.

The objective function is $min \sum x_i$ to minimize the number of primary variables. There are 2 constraints:

- $y_i + \sum a_{ij}x_j \geq 1$ for every fluent where a_{ij} represents if i-th (a fluent have to be covered by a mutex or chosen to be left uncovered). fluent is contained in a j-th mutex.
- $\sum y_i \leq 1$ for every fluent in an exactly-1 group (at most one fluent from every exactly-1 group can be uncovered).

Each uncovered fluent corresponds to a secondary variable u in the resulting SAS+ problem, whose value is determined by an axiom [$u\leftarrow$ not x, not v,...,not z] where x,v and z are the other fluents in the same exactly-1 group as u.

To apply tau axiom extraction to a problem Π with exactly-1 axioms, we need to deal with introduced secondary variables U and axioms A. For every $u \in U$ and internal state $\{v_1{=}x_1, ..., v_n{=}x_n\}$, we introduce a secondary variable $u_{\{v_1=x_1,...,v_n=x_n\}} \in U'$, which represents the value of u in an internal state $\{v_1{=}x_1, ..., v_n{=}x_n\}$. For every axiom with u as its head, we make a copy of it for every $u_{\{v_1=x_1,...,v_n=x_n\}}$. A reformulated operator $o' \in \overline{T}$ has $u_{\{v_1=x_1,...,v_n=x_n\}}$ in pre(o') instead of u.

5 Decoding

A plan $p'{=}o'_1, ..., o'_m$ to the encoded problem Π' is not necessarily a plan to the original problem Π. We showed we can always decode a plan p' back to p but not exactly how. p' is decoded back to p as follows. 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 tau 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, as our empirical results show, Π_{sub} is trivially solved for all our IPC benchmarks (Supplement, Table S1).

In our current implementation, the axiom decoding subproblems are solved using Fast Downward (Helmert 2006), using A* (Hart, Nilsson, and Raphael 1968) with the blind heuristic (h=0 for a goal state, otherwise, h is the cost of the least expensive operator).

Note that by turning tau 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' unless all tau operators have 0 cost.

Domain	Default		Graph-based Tau			Var-based Tau			Exactly-1			Exactly-1 + Var-based Tau			
	done	time	done	time	# tau	done	time	# tau	done	time	# exactly-1	done	time	# exactly-1	# tau
Domains with tau axioms only															
miconic(150)	150	0.33	150	3.00	184455	150	2.70	184455	150	0.37	0	150	2.74	0	184455
pegsol-opt11(20)	20	0.23	20	0.38	660	20	0.36	660	20	0.26	0	20	0.38	0	660
satellite(36)	34	0.45	17	4.68	45	33	0.64	45	34	0.51	0	33	0.71	0	45
Domains with exactly-1 axioms only															
barman-opt14(14)	14	0.39	14	0.90	0	14	0.52	0	14	0.42	116	14	0.60	116	0
blocks(35)	35	0.22	35	0.38	0	35	0.34	0	35	0.26	372	35	0.38	372	0
depot(22)	22	0.63	19	3.38	0	22	0.83	0	22	0.72	824	22	1.18	824	0
driverlog(20)	20	0.30	16	1.57	0	20	0.43	0	20	0.33	67	20	0.48	67	0
freecell(80)	80	1.01	17	9.18	0	80	1.29	0	80	1.17	320	80	1.48	320	0
mystery(30)	30	0.48	20	3.49	0	30	0.67	0	30	0.55	22	30	0.74	22	0
parcprinter-opt11(20)	20	0.28	20	0.43	0	20	0.39	0	20	0.31	20	20	0.41	20	0
parking-opt14(20)	20	0.97	4	9.11	0	20	1.30	0	20	1.03	500	20	1.67	500	0
storage(30)	30	0.38	20	1.60	0	30	0.52	0	30	0.41	810	30	0.63	810	0
tidybot-opt14(20)	20	None	0	None	0	20	None	0	20	None	20	20	None	20	0
woodworking-opt11(20)	20	0.48	20	1.18	0	20	0.65	0	20	0.52	359	20	0.70	359	0
Domains with both exactly-1 axioms and tau axioms															
airport(50)	50	2.57	41	6.30	23	50	2.93	23	50	8.94	8275	50	13.22	8275	23
grid(5)	5	0.69	2	7.08	200	5	1.56	880	5	0.86	5	5	2.29	5	880
gripper(20)	20	0.23	20	0.37	40	20	0.35	40	20	0.24	40	20	0.36	40	40
scanalyzer-opt11(20)	19	0.55	9	4.44	252	19	0.81	252	19	0.63	0	19	0.85	0	252
sokoban-opt08(30)	30	0.33	30	0.58	0	30	0.46	0	30	0.41	1208	28	53.64	1208	2590
visitall-opt11(20)	20	0.22	20	0.36	76	20	0.32	76	20	0.24	20	20	0.40	20	76
all(2240)	2231	0.54	1686	2.51	188273	2230	0.92	188977	2231	0.82	23717	2221	5.11	23717	197967

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), # tau = sum of tau operators found, # exactly-1 = number of exactly-1 axioms found. 2240 instances (Details not shown for domains where axioms weren't found).

6 Evaluation of Axiom Extraction

We applied the algorithms for extracting tau-axioms and exactly-1-axioms to a set of 2240 standard International Planning Competition (IPC) instances (5min runtime, 2GB RAM per instance). Domains containing conditional effects are excluded. Table 1 compares the following: (1) Default - standard Fast Downward translator (Helmert 2006), (2) Graph-based tau axiom extraction, (3) Variable-based tau axiom extraction, (4) Exactly-1 axiom extraction, and (5) Exactly-1 axiom followed by Variable-based tau axiom extraction.

Although the variable-based tau 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, the graph-based and variable-based tau axiom extraction algorithms found the same number of tau operators.

Exactly-1 axiom extraction, which includes a significantly modified PDDL to SAS+ translator, is somewhat slower than the default translator.

Although tau axiom extraction by itself failed on Sokoban (for the reason explained in Sectoin 4), combining exactly-1 extraction and variable-based tau enables extraction of the tau axioms in sokoban. In addition to Sokoban, we successfully found tau axioms in domains such as grid and miconic. In the grid domain, where the player walks around a maze to retrieve the key to the goal, we identified the player movements as tau operators just as in Sokoban. miconic is a elevator domain where each passenger has a start floor and a destination floor, and we must transport the passengers to their destination using a single elevator. Operators for moving up/down the elevator were identified as tau operators.

7 Improving Planer Performance Using Extracted Axioms

Extracted Axioms for Model-based Planning We investigate our reformulation approach by integrating the axioms extracted using the techniques proposed above into an integer programming (IP) based planner.

Our IP-based planner 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. Our baseline IP-based planner improves upon Optiplan, by some additional, straightforward mutex constraints on the values of SAS+ variables. Below, we refer to this baseline IP-based planner as IPlan.

Axioms can be integrated into an IP-based model (or, similarly, a constraint programming 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:

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

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 Niemelä (2012), and these linear constraints are added to the IPlan model.

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 Optiplan and IPlan) 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 therefore be executed in parallel.

For the models with axioms, we add a constraint which restricts the number of actions executed at each step to 1 (sequential 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 not easy (restoring some step-parallelism by analyzing the interactions between operators is a direction for future work). However, in an axiom-enhanced encoding, a single action can trigger many axioms (implied actions), so it is very possible that solutions to axiom-enhanced models can require fewer steps than a model without axioms. Thus, there is a tradeoff between loss of explicit parallel execution semantics ($\forall$ vs. sequential) and the implied parallel execution semantics due to axioms.

We compared the following: (1) Optiplan (a reimplementation of (van den Briel and Kambhampati 2005)), (2) IPlan (IPlan with $\forall$ semantics), (3) IPlanS (IPlan with sequential semantics), (4) IPlan+T (tau axioms only), (5) IPlan+T+ (tau axiom + model selection), (6) IPlan+E (exactly-1 axioms only), (7) IPlan+ET (both exactly-1 and tau axioms), and (8) IPlan+ET+ (IPlan+ET + model selection). The *model selection* policy used in IPlan+T+ and IPlan+ET+ chooses the IPlan+T and IPlan+ET model, respectively, if tau axioms are discovered; otherwise, the standard IPlan model (with $\forall$ semantics, without axioms) is used. While Optiplan, IPlan and IPlanS use the relaxed planning graph based variable elimination method of Optiplan, the current implementations of IPlan+E and IPlan+ET do not – adapting planning-graph based variable reductions compatible with axioms is a direction for future work.

Domain (5min, 2GB)	Optiplan	IPlan	IPlanS	IPlan+T	IPlan+T+	IPlan+E	IPlan+ET	IPlan+ET+
Domains with tau axioms only								
miconic(150)	28	29	25	**70**	**70**	25	**70**	**70**
pegsol-opt11(20)	1	1	2	**17**	**17**	2	**17**	**17**
satellite(36)	**8**	**8**	3	3	**8**	3	3	**8**
scanalyzer-opt11(20)	**11**	**11**	4	4	**11**	3	3	**11**
Domains with exactly-1 axioms only								
barman-opt14(14)	0	0	0	0	0	0	0	0
blocks(35)	16	29	29	29	29	**32**	**32**	29
depot(22)	7	**11**	2	2	**11**	2	2	**11**
driverlog(20)	**11**	**11**	4	4	**11**	4	4	**11**
freecell(80)	**18**	**18**	7	7	**18**	9	9	**18**
mystery(30)	13	**16**	14	14	**16**	13	13	**16**
parcprinter-opt11(20)	**20**	**20**	12	12	**20**	12	12	**20**
parking-opt14(20)	0	0	0	0	0	0	0	0
storage(30)	**9**	**9**	7	7	**9**	7	7	**9**
tidybot-opt14(20)	0	0	0	0	0	0	0	0
woodworking-opt11(20)	**20**	**20**	10	10	**20**	10	10	**20**
Domains with both exactly-1 axioms and tau axioms								
airport(50)	13	**16**	7	7	**16**	7	7	**16**
grid(5)	1	1	1	1	1	1	1	1
gripper(20)	4	4	2	8	8	2	**13**	**13**
sokoban-opt08(30)	0	0	0	0	0	1	**5**	**5**
visitall-opt11(20)	8	8	**10**	9	9	9	8	8
all(662)	188	212	139	204	274	142	216	**283**

Table 2: Coverage (# solved) for IP-based Planners.

The IP models generated are solved using Gurobi Optimizer 6.5.0, with 5 minute runtime (single-threaded) and 2GB of RAM per problem. The runtime includes all phases of IPlan with axioms, including the PDDL to SAS+ conversion, axiom extraction, solving the model, and axiom decoding.

The results are shown in Table 2. IPlan+ET+ achieves significantly higher coverage than the baseline models (Optiplan, IPlan, IPlanS). Switching form $\forall$ semantics (IPlan) to sequential semantics (IPlanS) significantly degrades performance, but the performance boost due to axioms more than compensates for the loss due to sequential semantics. Improvements due to axioms are particularly apparent on gripper, miconic, pegsol and Sokoban. IPlan+E performs poorly compared to IPlan and is only marginally better than IPlanS, showing that most of the power of IPlan+ET+ and IPLan+ET comes from the tau axioms. However, recall from Sec. 4 that exactly-1 axiom extraction is necessary to enable tau axiom extraction in domains such as Sokoban. Also, some domains directly benefit from exactly-1 axioms (blocks)

Table S1 (Supplement) shows that 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 per layer, as explained above), while IPlan uses $\forall$ semantics (multiple independent 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).

Extracted Axioms for Forward Search Planning In addition to IP-based planning, we applied our reformulation approach (exactly-1 axioms + tau axioms) to A* searchin Fast Downward (Helmert 2006) (5 min., 2GB RAM per instance). However, to guarantee that A* returns an optimal solution, all tau operator must be 0 cost (see Sec. 5), so application of our approach to A* on the IPC domains is limited to Sokoban and pegsol. To assess the effect of axioms on the size of the state space, we used the blind heuristic.

On Sokoban, A* (without axioms) solved 20 instances (85,807,984 node expansions until last jump), while A*+axioms solved 22 instances (11,759,970 nodes). On pegsol, A* solved 17 instances (15,392,253 nodes), while A*+axioms solved 17 instances (11,929,920 nodes). Thus, automatically extracted axioms resulted in reduction of the state space in A*.

8 Conclusions

We proposed novel methods for automatically identifying and extracting two classes of axioms, tau axioms and exactly-1 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 "discover" 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 contains derived variables, and our automated reformulation results in a "better" model than the original, human-designed models.

References

Apt, K. R.; Blair, H. A.; and Walker, A. 1988. Towards a theory of declarative knowledge.

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.

Dimopoulos, Y.; Nebel, B.; and Koehler, J. 1997. Encoding planning problems in nonmonotonic logic programs. In *Recent Advances in AI Planning, 4th European Conference on Planning, ECP'97, Toulouse, France, September 24-26, 1997, Proceedings*, 169–181.

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.

Fernandez, J.-C., and Mounier, L. 1991. on the fly verification of behavioural equivalences and preorders. In *International Conference on Computer Aided Verification*, 181–191. Springer.

Fleischner, H.; Mujuni, E.; Paulusma, D.; and Szeider, S. 2009. Covering graphs with few complete bipartite subgraphs. *Theoretical Computer Science* 410(21):2045–2053.

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

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

Helmert, M. 2006. The Fast Downward planning system. *J. Artif. Intell. Research* 26:191–246.

Helmert, M. 2009. Concise finite-domain representations for PDDL planning tasks. *Artificial Intelligence* 173(5):503–535.

Ivankovic, F., and Haslum, P. 2015a. Code supplement for "optimal planning with axioms". IJCAI2015.

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

Liu, G.; Janhunen, T.; and Niemelä, I. 2012. Answer set programming via mixed integer programming. *Proc. Int. Conf. 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. *J. Autom. Reasoning* 3(4):343–377.

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

Tarjan, R. 1972. Depth-first search and linear graph algorithms. *SIAM J. 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. *J. Artific. Intell. Research*.

Vossen, T.; Ball, M. O.; Lotem, A.; and Nau, D. 1999. On the use of integer programming models in AI planning. In *Proc. IJCAI*.