Proceedings of the Twenty-Ninth International Conference on Automated Planning and Scheduling (ICAPS 2019)

# Symbolic Planning with Axioms

**David Speck**
University of Freiburg
speckd@cs.uni-freiburg.de

**Florian Geißer**
Research School of CS, ANU
florian.geisser@anu.edu.au

**Robert Mattmüller**
University of Freiburg
mattmuel@cs.uni-freiburg.de

**Álvaro Torralba**
Saarland University
torralba@cs.uni-saarland.de

**Abstract**

Axioms are an extension for classical planning models that allow for modeling complex preconditions and goals exponentially more compactly. Although axioms were introduced in planning more than a decade ago, modern planning techniques rarely support axioms, especially in cost-optimal planning. Symbolic search is a popular and competitive optimal planning technique based on the manipulation of sets of states. In this work, we extend symbolic search algorithms to support axioms natively. We analyze different ways of encoding derived variables and axiom rules to evaluate them in a symbolic representation. We prove that all encodings are sound and complete, and empirically show that the presented approach outperforms the previous state of the art in cost-optimal classical planning with axioms.

## Introduction

In classical planning, most planners support models that describe a state of the world in terms of Boolean or finite-domain variables, such as STRIPS (Fikes and Nilsson 1971) or SAS$^+$ (Bäckström and Nebel 1995). The aim of such planners is to find a sequence of operations that transforms a specified initial state into a desired goal. While these models are already very general, more expressive models like ADL with conditional effects (Pednault 1989) or state-dependent action costs (Geißer, Keller, and Mattmüller 2015) allow a more compact representation of a planning problem. Axioms are another example of such a model extension which allow for modeling complex preconditions and goals compactly. Planning with axioms introduces a set of derived variables whose values are not directly influenced by the actions, but are derived from the values of other variables using a set of logical axioms. Thiébaux, Hoffmann, and Nebel (2005) argue that axioms are necessary to model real-world problems in a compact and elegant way, as they allow to model complex action preconditions or goals that cannot be expressed in the original formalism without incurring a super-polynomial growth of plan length or domain description size.

Although axioms are a feature of PDDL (McDermott et al. 1998; Hoffmann and Edelkamp 2005), the common language for modeling planning tasks, modern planning techniques rarely support axioms, especially in cost-optimal planning. Most admissible heuristics commonly used in A$^*$ search, one of the most prominent approaches to cost-optimal planning, are not defined for their use with axioms. The few heuristics that support axioms are based on naive relaxations that consider axioms as zero-cost actions, which may greatly reduce the informativeness of the heuristics. One exception is the axiom-aware delete relaxation heuristic, obtained by applying a model for state constraints to planning with axioms (Ivankovic and Haslum 2015; Haslum et al. 2018). While these heuristics are often informative, they are also expensive to compute and therefore often do not pay off in terms of coverage or runtime.

In the past decade, symbolic search has proven to be a strong and competitive alternative algorithm for cost-optimal planning (Kissmann, Edelkamp, and Hoffmann 2014; Torralba et al. 2014). Here, sets of states are represented by compact data structures, e.g., by Binary Decision Diagrams (BDDs) (Bryant 1986), which make it possible to perform an exhaustive search by efficient manipulation of sets of states, often more efficiently than an explicit state search. The dominant search strategy is bidirectional blind search. While modern symbolic search planners have been generalized to support conditional effects (Kissmann, Edelkamp, and Hoffmann 2014) or state-dependent action costs (Speck, Geißer, and Mattmüller 2018a; 2018b), there is currently no planner that supports axioms.

In this paper, we analyze three different ways of encoding axioms and derived variables to evaluate them in a symbolic representation. We prove that all encodings are sound and complete and discuss their advantages and disadvantages. An empirical study on various planning domains shows that the presented encodings result in an optimal planner supporting axioms that outperforms the previous state of the art. Furthermore, in a second study we empirically show that even for classical planning domains without axioms, symbolic planning can benefit from our native axiom support, since this allows searching on reformulated planning tasks with automatically extracted axioms.

## Preliminaries

A planning task is a tuple $\Pi = (\mathcal{V}, \mathcal{D}, \mathcal{A}, \mathcal{O}, \mathcal{I}, \mathcal{G})$, where $\mathcal{V}$ is a set of (primary) state variables, each with a finite domain $D_v$, and $\mathcal{D}$ is a set of (secondary) derived variables

which can take Boolean values $\{\top, \bot\}$. A partial state is a variable assignment to variables in $\mathcal{V} \cup \mathcal{D}$. A state is a variable assignment to all variables in $\mathcal{V}$, and an extended state is a variable assignment to all variables in $\mathcal{V} \cup \mathcal{D}$. With $\mathcal{S}$ we refer to the set of all possible states defined over the primary variables $\mathcal{V}$. $\mathcal{I}$ is the initial state, and $\mathcal{G}$ is a partial state that defines all possible goal states.

$\mathcal{A}$ is a set of axioms partitioned into layers $\mathcal{A}_1 \prec \cdots \prec \mathcal{A}_k$. Each layer $\mathcal{A}_i$, $i = 1, \ldots, k$, consists of a finite number of axiom rules of the form $b \rightarrow h$, where the body $b$ is a finite conjunction of positive or negative literals over $\mathcal{V} \cup \mathcal{D}$, and the head $h$ is a variable in $\mathcal{D}$. Given an axiom (rule) $r$, we denote with $body(r)$ the body of $r$. If it is clear from the context, we sometimes abuse notation and also denote with $body(r)$ the set of variables appearing in the body of $r$. Similarly, we denote with $head(r)$ the head of $r$. The layer of an axiom is defined by the layer of its head which is determined by a partition of the set of derived variables into subsets $\mathcal{D}_1 \prec \cdots \prec \mathcal{D}_k$. We assume that this partition forms a *stratification*, i.e., that for all $i = 1, \ldots, k$, and for every $d_i \in \mathcal{D}_i$, it holds that (1) if $d_j \in \mathcal{D}_j$ appears in the body of an axiom with head $d_i$ then $j \leq i$ and (2) if $d_j \in \mathcal{D}_j$ appears negated in the body of an axiom with head $d_i$ then $j < i$. Intuitively, (1) to evaluate a derived variable, only rules in the current or previous layers have to be considered, and (2) the axioms have a negation-as-failure semantics, i.e., if a fact cannot be derived to be true, it is assumed to be false on subsequent layers. Therefore, axiom layers have to be fully evaluated before a variable from that layer can be used negatively in another axiom body, which necessitates moving to a strictly higher layer if $d_j$ is used negatively. Given a state $s \in \mathcal{S}$, the extended state $\mathcal{A}(s)$ is uniquely defined by standard stratified semantics (Apt, Blair, and Walker 1988; Thiébaux, Hoffmann, and Nebel 2005).

Algorithm 1 describes the axiom evaluation algorithm for explicit states, following the exposition of Helmert (2006). Some axiom definitions also specify a default value for derived variables, which is assumed before any rules are applied. We assume without loss of generality that the default value of all derived variables is $\bot$ (false).

**Algorithm 1:** Axiom evaluation for explicit states

**Data:** Axiom layers $\mathcal{A}_1 \prec \cdots \prec \mathcal{A}_k$, primary state $s$
**Result:** Extended state $\mathcal{A}(s)$

**foreach** *variable* $v$ **do**
  $s'(v) := \begin{cases} s(v) & \text{if } v \text{ is a primary variable} \\ \bot & \text{otherwise} \end{cases}$
**foreach** *axiom layer* $i = 1, \ldots, k$ **do**
  **while** *there exists an axiom* $b \rightarrow h \in \mathcal{A}_i$ *such that* $s' \models b$ *and* $s' \not\models h$ **do**
    choose such an axiom $b \rightarrow h$
    set $s'(h)$ to true
Return $s'$

$\mathcal{O}$ is a set of operators where each $o \in \mathcal{O}$ is a tuple $(pre_o, \mathit{eff}_o)$. The precondition $pre_o$ is a partial state and $\mathit{eff}_o$ is a set of conditional effects $(cond \rhd v = val)$ where $cond$ is a partial state, $v \in \mathcal{V}$ is a primary variable and $val$ is a value in $D_v$. For binary variables we also write $v$ for $v = 1$ and $\neg v$ for $v = 0$. An operator is applicable in a state $s$ if $pre_o \subseteq \mathcal{A}(s)$. The result of applying operator $o$ to a state $s$ is a state $s'$, where for all $v \in \mathcal{V}$ we have $s'(v) = val$ if there exists $(cond \rhd v = val) \in \mathit{eff}_o$ and $cond \subseteq \mathcal{A}(s)$, and $s'(v) = s(v)$ otherwise.[1] The objective of classical planning is to determine a sequence of operators (a plan) whose execution transforms the initial state to a goal state. Such a plan is considered to be optimal if there is no shorter plan.[2]

## Symbolic Search

Symbolic search is a state space exploration technique that uses efficient data structures to represent and manipulate sets of states (McMillan 1993). Sets of states $S \subseteq \mathcal{S}$ are represented via their characteristic function $\chi_S$, which is a Boolean function $\chi_S : \mathcal{S} \rightarrow \{\top, \bot\}$ that represents whether a given state belongs to $S$ or not. More precisely, $\chi_S(s) = \top$ (or equivalently: $s \models \chi_S$) if $s \in S$, and $\chi_S(s) = \bot$ (or equivalently: $s \not\models \chi_S$) if $s \notin S$. We also say that $\chi_S$ *characterizes* $S$. We use Binary Decision Diagrams (BDDs) that are a data-structure to represent such functions (Bryant 1986). For clarification, we sometimes annotate a set of states and its corresponding characteristic function with the set of variables over which it is defined, to explicitly distinguish state sets defined over primary variables and state sets defined over primary *and* derived variables. For example, $\chi_{S[\mathcal{V}]}$ is the characteristic function describing the set of states $S[\mathcal{V}]$, which are only defined over primary variables, while $\chi_{S[\mathcal{V},\mathcal{D}]}$ describes the set of extended states $S[\mathcal{V}, \mathcal{D}]$.

In symbolic search, operators are represented as transition relations (TRs). A TR represents a set of operators $O \subseteq \mathcal{O}$ as a BDD $T_O[\mathcal{V}, \mathcal{V}']$ that contains the set of all pairs $(s, s')$, such that $s'$ is reachable from $s$ by applying an operator $o \in O$. Given a set of states $S \subseteq \mathcal{S}$ and a TR $T$, the image operation computes the set of successor states of $S$ through $T$. The complexity of the image operation is worst-case exponential in the number of state variables, but is often more efficient than expanding the states in $S$ one by one. The reverse pre-image operation computes the set of predecessor states where $o$ can be applied to reach some state in $S$.

Symbolic forward search algorithms start from the BDD representation of the initial state, and iteratively compute the image until a BDD is found whose intersection with the goal is non-empty. Similarly, one can define backward search algorithms that start from the goal and iteratively apply pre-image operations until a BDD that contains the initial state is found.

## Symbolic Search with Axioms

Explicit search is based on the generation and expansion of single states while symbolic planning is based on the manipulation of sets of states. Thus, in explicit search after

[1] We assume that effects are well-formed, i.e., the conditions of multiple conditional effects assigning different values to the same variable can never hold in the same state.

[2] We consider unit costs without loss of generality.

each state generation the values of the derived variables can be inferred independently. The latter does not hold anymore in symbolic planning. However, the concept of an extended state $\mathcal{A}(s)$ can be generalized to a set of extended states $S_{\mathcal{A}}$ as shown in Definition 1.

**Definition 1 (Extended State Set).** Let $S[\mathcal{V}]$ be a set of states. The extended state set $S_{\mathcal{A}}[\mathcal{V}, \mathcal{D}]$ contains the extension of all states of $S[\mathcal{V}]$, i.e., $S_{\mathcal{A}}[\mathcal{V}, \mathcal{D}] = \{\mathcal{A}(s) | s \in S[\mathcal{V}]\}$.

Note, that the extension of a state set is again unique because the extension of each individual state is unique.

Symbolic search is known to be an efficient technique for exhaustive state space exploration, to enumerate all states that are reachable from a given set of states. In the following, three different ways of encoding derived variables and axiom rules for evaluation in a symbolic representation are presented.

## Encoding I: Action-Based

In the action-based encoding, the first of the three encodings proposed in this paper, we evaluate axiom rules by interpreting them as operators $\mathcal{O}_{\mathcal{A}}$ with no precondition and a conditional effect where the body of the rule is the condition and the head is the effect (note that, as opposed to original operators, the effect affects a *derived* variable instead of a primary one). Starting from a BDD $\chi_{S[\mathcal{V}]}$ representing the state set $S[\mathcal{V}]$, one can perform a breadth-first search with the operators in $\mathcal{O}_{\mathcal{A}}$ to obtain a BDD $\chi_{S_{\mathcal{A}}[\mathcal{V},\mathcal{D}]}$ which represents the corresponding extended state set $S_{\mathcal{A}}[\mathcal{V}, \mathcal{D}]$ by applying $\mathcal{O}_{\mathcal{A}}$ until a fixpoint is reached for each stratification layer. In other words, after application of operators in $\mathcal{O}$ (via transition relations), the obtained set of successor states has to be extended. A set of successor states still contains information about the derived variables of their predecessors, therefore we first quantify over all derived variables and set them to false. Afterwards, a layer-wise fixpoint computation is performed, where the result of the previous computation is the starting point for the next computation. This process is performed in ascending order of the axiom layers $\mathcal{A}_1 \prec \cdots \prec \mathcal{A}_k$, until all layers are processed. Figure 1 visualizes this action-based axiom representation.

**Example 1.** Consider a planning task with primary variables $x, y$ and derived variables $a, b, c$, together with the following set of axioms:

$$r_{11} : b \rightarrow a$$
$$r_{12} : \neg x \rightarrow b$$
$$r_{13} : y \rightarrow b$$
$$r_{21} : (\neg a \wedge \neg b) \rightarrow c,$$

where $\mathcal{A}_1 = \{r_{11}, r_{12}, r_{13}\}$ and $\mathcal{A}_2 = \{r_{21}\}$, i.e., $\mathcal{D}_1 = \{a, b\}$ and $\mathcal{D}_2 = \{c\}$.

We get the following set of operators:

$$o_{11} : \langle \top, b \rhd a \rangle$$
$$o_{12} : \langle \top, \neg x \rhd b \rangle$$
$$o_{13} : \langle \top, y \rhd b \rangle$$
$$o_{21} : \langle \top, (\neg a \wedge \neg b) \rhd c \rangle.$$

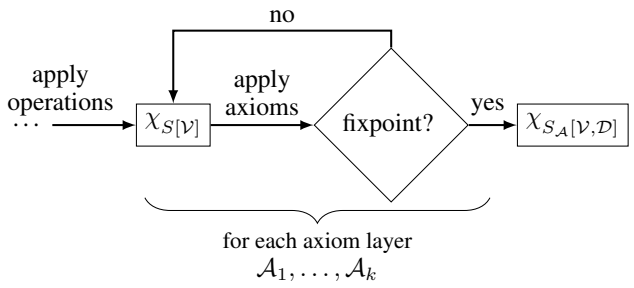


Figure 1: Visualization of symbolic forward search with the action-based encoding of axioms. The BDD $\chi_{S[\mathcal{V}]}$ representing the set of successor states is extended to $\chi_{S_{\mathcal{A}}[\mathcal{V},\mathcal{D}]}$ by applying axioms represented as operators layer by layer until a fixpoint is reached.

Let us assume that after restricting to primary variables we obtain $\chi_{S[\mathcal{V}]} = x$, i.e., our current set of states consists of the states $x \wedge y$ and $x \wedge \neg y$. Setting the derived variables to false results in $\chi_{S[\mathcal{V}]} = x \wedge \neg a \wedge \neg b \wedge \neg c$. Then, for each layer we apply the corresponding operators until a fixpoint is reached: applying $o_{11}$ and $o_{12}$ on $\chi_{S[\mathcal{V}]}$ does not change anything, application of $o_{13}$ leads to the state set $(x \wedge y \wedge \neg a \wedge b \wedge \neg c) \vee (x \wedge \neg y \wedge \neg a \wedge \neg b \wedge \neg c)$. We then can once again apply $o_{11}$ and $o_{12}$ , resulting in $(x \wedge y \wedge a \wedge b \wedge \neg c) \vee (x \wedge \neg y \wedge \neg a \wedge \neg b \wedge \neg c)$, which is a fixpoint for the current layer. We proceed with the next layer, where application of $o_{21}$ results in $\chi_{S[\mathcal{V},\mathcal{D}]} = (x \wedge y \wedge a \wedge b \wedge \neg c) \vee (x \wedge \neg y \wedge \neg a \wedge \neg b \wedge c)$. Since this is once again a fixpoint and there exist no further layers we are finished and the resulting fully evaluated extended state set $\chi_{S_{\mathcal{A}}[\mathcal{V},\mathcal{D}]}$ is computed.

One particular drawback of the action-based encoding is the fixpoint computation, which can be expensive for two reasons: first, in symbolic search we reason over multiple states at the same time, and therefore the application of an operator in $\mathcal{O}_{\mathcal{A}}$ correlates with the evaluation of the corresponding axiom over multiple states, even if this particular axiom is only relevant for a small subset of states. Second, observe that in the previous example we had to reapply operators $o_{11}$ and $o_{12}$ after application of $o_{13}$. In theory, we could have first applied $o_{13}$ before applying $o_{11}$ and $o_{12}$, to reduce the number of image computations, which usually form the main bottleneck of symbolic search. For this example it is pretty easy to see that the latter operator (i.e. axiom) ordering minimizes the number of image computations, because $o_{11}$ and $o_{12}$ are not applicable in any state contained in $\chi_{S[\mathcal{V}]}$. In general, however, one operator might initially be already applicable in a subset of states, but after application of other operators of the same layer be applicable in even more states. To reduce the number of image computations, we therefore have to come up with a good axiom operator ordering. For this, we propose the following alternative form of stratification.

**Stratification.** We analyze the dependencies of axioms to schedule the order in which they have to be applied. For

this, we distinguish two types of layers: layers where a derived variable appears both in the body and the head of a rule require a fixpoint computation, while other layers need to be evaluated only once. To determine the assignment of axioms to layers, we construct an axiom dependency graph, with a node per axiom and an edge from $r_i$ to $r_j$ if $head(r_i) \in body(r_j)$. Then, the maximal strongly connected components (Tarjan 1972) of the dependency graph are determined, resulting in an acyclic graph where each node corresponds to an SCC. We process all SCCs of the graph in topological order, always choosing an SCC where all incoming edges have already been processed. If one such SCC has more than one axiom rule, then all the rules in that SCC are assigned to the next layer. This layer has axiom rules that depend on one another, so it requires a fixpoint computation. If all SCCs have a single rule, then all those rules can be processed simultaneously. Therefore, all of them are assigned to the same layer, which needs to be applied only once.

Before we prove that the action-based encoding does indeed result in the correct fully evaluated extended state, we note that most sophisticated symbolic planners perform bidirectional search (and therefore regression), instead of forward search. While the action-based encoding is based on operators for axiom evaluation, it is an open question how to reverse the fixpoint iteration of the operators, which would be necessary for backward (and therefore bidirectional) search.

That being said, Proposition 1 states the soundness and correctness of the action-based encoding of axioms.

**Proposition 1.** *Let $S[\mathcal{V}]$ be a set of states characterized by its characteristic function $\chi_{S[\mathcal{V}]}$. Then the symbolic axiom evaluation procedure based on the action-based encoding described above, applied to $\chi_{S[\mathcal{V}]}$, results in a formula $\chi_{S_\mathcal{A}[\mathcal{V},\mathcal{D}]}$ that characterizes precisely the extended state set $S_\mathcal{A}[\mathcal{V},\mathcal{D}]$.*

*Proof sketch.* Follows immediately by construction, as the symbolic axiom evaluation procedure using the action-based encoding is essentially a symbolic implementation of the axiom evaluation algorithm for explicit states (Algorithm 1), up to the missing test whether applying a certain axiom results in a new fact or not (end of line 4 of Algorithm 1), which is not necessary in the symbolic case. □

## Representation of Derived Variables

A major drawback of the action-based encoding is that after each planning step, an expensive fixpoint iteration has to be performed. The following two axiom encodings overcome this issue, by representing each derived variable $d \in \mathcal{D}$ as a set of states over primary variables, $S_d[\mathcal{V}]$, to represent the subset of states for which $d$ holds:

**Definition 2 (Primary Representation).** Let $d \in \mathcal{V} \cup \mathcal{D}$ be a (primary or derived) variable and $\mathcal{A}$ a set of axioms. The *primary representation* of $d$ is the set of states $S_d[\mathcal{V}]$ which contains all states over $\mathcal{V}$ where $d$ is evaluated to true, i.e., $S_d[\mathcal{V}] = \{s \in \mathcal{S} | \mathcal{A}(s) \models d\}$.

Algorithm 2 shows how the primary representation in form of its characteristic function can be constructed.[3] We build the primary representations layer by layer, exploiting the property that derived variables in a layer only depend on the value of primary variables or derived variables in a previous or the current layer. At each iteration, the algorithm computes the corresponding primary representation for variables in a given layer, assuming that we have computed the primary representation $\chi_{S_d[\mathcal{V}]}$ of each derived variable $d \in \mathcal{D}$ of previous layers.

**Algorithm 2:** Construction primary representations

```
   Data: Axiom layers A_1 ≺ ··· ≺ A_k
   Data: Derived variables D_1 ≺ ··· ≺ D_k
   Result: Symb. primary representations χ_{S_d}, d ∈ D
1  foreach D_1 ≺ ··· ≺ D_k do
2      foreach d ∈ D_i do
3          χ_{S_d} ← ⋁_{r ∈ A_d^{<i}} body(r)[χ_{S_D}/D]
4      queue ← {d | d ∈ D_i}
5      while queue is not empty do
6          d ← pop(queue)
7          foreach r ∈ A_i with d ∈ body(r) do
8              χ_{S_head(r)} ← χ_{S_head(r)} ∨ body(r)[χ_{S_D}/D]
9              if χ_{S_head(r)} has changed then
10                 queue.insert(head(r))
11 return {χ_{S_d} | d ∈ D}
```

At the beginning, we collect the information of a derived variable which only depends on variables of lower levels. More precisely, we represent the set of states in which $d \in \mathcal{D}$ is true by applying all axioms in $\mathcal{A}_d^{<i}$: the set of rules that have $d$ in their head and whose bodies only contain variables of lower layers, i.e., $\mathcal{A}_d^{<i} = \{r \in \mathcal{A} \mid head(r) = d \text{ and } \forall d' \in body(r) \cap \mathcal{D} : d' \in \mathcal{D}_j \text{ for some } j < i\}$. To process an axiom $r$, we compute $body(r)[\chi_{S_\mathcal{D}}/\mathcal{D}]$, where $\varphi[\chi_{S_\mathcal{D}}/\mathcal{D}]$ stands for the result of simultaneously replacing each derived variable $d \in \mathcal{D}$ with $\chi_{S_d[\mathcal{V}]}$ in formula $\varphi$. Then, the algorithm proceeds to applying axioms that depend on variables in the same layer until a fixpoint is reached. Note that at no point, a derived variable is contained in $\chi_{S_d[\mathcal{V}]}$.

**Example 2.** Consider once again the variables and axioms of Example 1, where $\mathcal{D}_1 = \{a, b\}$ and $\mathcal{D}_2 = \{c\}$. In the following, we compute the primary representations $\chi_{S_a}, \chi_{S_b}$ and $\chi_{S_c}$ by application of Algorithm 2.

We start with $\mathcal{D}_1$ and $d = a$ (line 3). Note that $\mathcal{A}_a^{<1} = \emptyset$, since the body of $r_{11}$ contains $b$, which lies in the same layer. Therefore, $\chi_{S_a} = \bot$. We proceed with $d = b$ and have $\mathcal{A}_b^{<1} = \{r_{12}, r_{13}\}$ and therefore get $\chi_{S_b} = \neg x \vee y$. Note that both $x$ and $y$ are primary variables. We continue with initializing the queue (line 4) with $a$ and $b$. We pop $a$ from the queue, but since there is no rule in the current layer where $a$ is contained in the body (line 7), we proceed with

[3] We omit the part "$\cdot[\mathcal{V}]$" signifying the relevant variables in the pseudocode to avoid clutter. All state sets and formulas are over $\mathcal{V}$.

popping $b$. We have one rule in the current layer where $b$ occurs in the body, namely $r_{11} = b \to a$ and set (line 8) $\chi_{S_a} = \bot \vee \chi_{S_b} \equiv \neg x \vee y$. We have to insert $a$ again into the queue, but as before we can pop it from the queue and exit the loop. This concludes the iteration for $\mathcal{D}_1$ and we end up with $\chi_{S_a} = \chi_{S_b} = \neg x \vee y$.

In the next iteration, we have $d = c$ and $\mathcal{A}_c^{<2} = \{r_{21}\}$, where $r_{21} = (\neg a \wedge \neg b) \to c$. Therefore, we get

$$\begin{aligned}\chi_{S_c} &= (\neg\chi_{S_a} \wedge \neg\chi_{S_b}) \\ &= (\neg(\neg x \vee y) \wedge \neg(\neg x \vee y)) \\ &\equiv x \wedge \neg y.\end{aligned}$$

Note that since $\chi_{S_a} = \chi_{S_b}$, a BDD library may skip intermediate reformulations, since both BDDs are the same object.

We continue the algorithm, but since there is no rule where $c$ appears in the body, we are finished and end up with the primary representations $\chi_{S_a} = \chi_{S_b} = \neg x \vee y$ and $\chi_{S_c} = x \wedge \neg y$, which concludes the example.

The following Proposition 2 states that Algorithm 2 is sound and complete, i.e., that it returns a BDD $\chi_{S_d[\mathcal{V}]}$ which precisely characterizes the primary representation $S_d[\mathcal{V}]$ for each $d \in \mathcal{D}$.

**Proposition 2.** *Let $d \in \mathcal{D}$ be a derived variable, and let $\chi_{S_d[\mathcal{V}]}$ be the formula computed for $d$ by Algorithm 2. Then, for each state $s$ over primary variables $\mathcal{V}$,*

$$\mathcal{A}(s) \models d \quad \textit{if and only if} \quad s \models \chi_{S_d[\mathcal{V}]}.$$

*In other words, $\chi_{S_d[\mathcal{V}]}$ characterizes the primary representation of $d$, $S_d[\mathcal{V}]$.*

*Proof sketch.* The equivalence proof proceeds by induction over the axiom layers. We may start with the primary variables as axiom layer zero, resulting in a trivial base case. For the inductive case, we may assume that the claim already holds for all lower axiom layers. Then, the information from strictly lower axiom layers is accounted for in line 3 of Algorithm 2, so it suffices to compare the corresponding fixpoint iterations of Algorithms 1 and 2. While they may consider axioms in different orders, the fixpoints will be the same. Also notice that by substituting $\chi_{S_\mathcal{D}}$ for $\mathcal{D}$ in the bodies of applied axioms, Algorithm 2 expresses the truth conditions of all derived variables $d$ purely in terms of primary variables, and "unrolls" layered or recursive dependencies. The "in other words" part immediately follows from the previous equivalence using the definition of primary representations. □

By structural induction on formulas, we can lift the previous proposition to entire formulas as follows.

**Corollary 3.** *Let $\varphi$ be a formula over primary and derived variables. Then, for each state $s$ over primary variables $\mathcal{V}$,*

$$\mathcal{A}(s) \models \varphi \quad \textit{if and only if} \quad s \models \varphi[\chi_{S_\mathcal{D}}/\mathcal{D}]. \qquad \square$$

The primary representations can be obtained in a precomputation step preceding search and will be required for the two following axiom encodings. We note that in theory, depending on the number of axioms and derived variables and their interaction, the BDDs representing some $S_d$ can become prohibitively large, but the empirical evaluation (cf. Experiments) shows that this is usually not an issue.

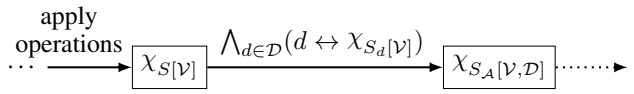


Figure 2: Visualization of symbolic forward search with variable-based encoding of axioms. A set of states represented by $\chi[\mathcal{V}]$ is extended to $\chi_\mathcal{A}[\mathcal{V},\mathcal{D}]$ by conjunctions with the primary representations of derived variables.

## Encoding II: Variable-Based

Our second proposed symbolic encoding of axioms and derived variables, the variable-based encoding, is based on the primary representation of derived variables, which allows us to extend a set of states by application of logical operations (i.e. conjunctions). Again, after we obtain a set of successor states, we restrict to primary variables, which results in the state set $S[\mathcal{V}]$. In this case it is not necessary to set the values of the derived variables to false, which was required in the action-based encoding. A derived variable $d$ is true in a state $s \in S[\mathcal{V}]$ iff $s \in S_d[\mathcal{V}]$. Therefore, $\chi_{S_\mathcal{A}[\mathcal{V},\mathcal{D}]} = \chi_{S[\mathcal{V}]} \wedge \bigwedge_{d\in\mathcal{D}}(d \leftrightarrow \chi_{S_d[\mathcal{V}]})$ represents the extended set of states. Figure 2 visualizes symbolic forward search in combination with the presented variable-based encoding of axioms. Although this encoding avoids the expensive fixpoint computation, it again remains an open question how to perform backward search, since the intermediate reasoning about the values of the derived variables still remains, albeit more efficiently.

**Example 3.** Consider the primary representations obtained in Example 2: $\chi_{S_a} = \chi_{S_b} = \neg x \vee y$ and $\chi_{S_c} = x \wedge \neg y$ and let us once again assume that $\chi_{S[\mathcal{V}]} = x$, obtained by quantifying over the derived variables. Note that we transform $\chi_{S[\mathcal{V}]}$ directly without setting the derived variables to false. With the variable-based encoding we get $\chi_{S_\mathcal{A}[\mathcal{V},\mathcal{D}]} = \chi_{S[\mathcal{V}]} \wedge (a \leftrightarrow \chi_{S_a}) \wedge (b \leftrightarrow \chi_{S_b}) \wedge (c \leftrightarrow \chi_{S_c})$ which results in $\chi_{S_\mathcal{A}[\mathcal{V},\mathcal{D}]} = (x \wedge y \wedge a \wedge b \wedge \neg c) \vee (x \wedge \neg y \wedge \neg a \wedge \neg b \wedge c)$.

Soundness and completeness then follows directly from Proposition 2. Intuitively, the following results show that the formula $\mathcal{C} = \bigwedge_{d\in\mathcal{D}}(d \leftrightarrow \chi_{S_d[\mathcal{V}]})$ characterizes the set of all extended states for which there is a state over primary variables that is extended to them by axiom evaluation. In other words, $\mathcal{C}$ can be viewed as the constraint that states are consistent with axiom evaluation.

**Proposition 4.** *Let $s'$ be a state over primary and derived variables. Then there is a state $s$ over primary variables such that $s' = \mathcal{A}(s)$, i.e., $s'$ is consistent with axiom evaluation, if and only if $s' \models \bigwedge_{d\in\mathcal{D}}(d \leftrightarrow \chi_{S_d[\mathcal{V}]})$.*

*Proof sketch.* Follows immediately from Proposition 2. □

**Corollary 5.** *Let $S[\mathcal{V}]$ be a set of states characterized by its characteristic function $\chi_{S[\mathcal{V}]}$. Then the formula $\chi_{S_\mathcal{A}[\mathcal{V},\mathcal{D}]} =$*

$\chi_{S[\mathcal{V}]} \wedge \bigwedge_{d \in \mathcal{D}} (d \leftrightarrow \chi_{S_d[\mathcal{V}]})$ *characterizes precisely the extended state set* $S_{\mathcal{A}}[\mathcal{V}, \mathcal{D}]$.

*Proof sketch.* Follows immediately from Proposition 4 and the definition of extended state sets. □

## Encoding III: Symbolic Compilation

While the previous encodings still require reasoning about derived variables during the actual search, our third proposed encoding completely forgoes derived variables, by performing search on a compilation of the original planning task. Again, we precompute the primary representations of all derived variables with Algorithm 2. The idea underlying the compilation is to replace all occurrences of derived variables in the planning task with their corresponding primary representation. In particular, derived variables in operator preconditions and the goal formula are replaced with the corresponding primary representation:[4]

**Definition 3 (Symbolic Compilation).** Let $\Pi = (\mathcal{V}, \mathcal{D}, \mathcal{A}, \mathcal{O}, \mathcal{I}, \mathcal{G})$ be a planning task (with axioms). The symbolic compilation of $\Pi$ is a new planning task (without axioms) $\Pi' = (\mathcal{V}, \emptyset, \emptyset, \mathcal{O}', \mathcal{I}, \mathcal{G}')$ where $\mathcal{O}' = \{(pre_o[\chi_{S_{\mathcal{D}}}/\mathcal{D}], \mathit{eff}_o) \mid o \in \mathcal{O}\}$ and $\mathcal{G}' = \mathcal{G}[\chi_{S_{\mathcal{D}}}/\mathcal{D}]$.

More precisely, if a BDD $\varphi$ is constructed and a derived variable $d \in \mathcal{D}$ occurs in $\varphi$, we replace $d$ with its primary representation in the form of another BDD $\chi_{S[\mathcal{V}]}$ during the construction. This makes use of the compact and efficient nature of BDDs. With the symbolic compilation we obtain a planning task without axioms and derived variables. Therefore, the symbolic compilation offers the possibility of backward search and bidirectional search, which can greatly enhance planning performance. Once again we note that replacing derived variables with their primary representation can lead to prohibitively large formulas that sometimes may not be represented compactly as BDDs.

Soundness and completeness follow directly from Corollary 3 by induction on the plan length.

**Proposition 6.** *Given a planning task* $\Pi$*, a sequence of operators* $\pi$ *is a plan for* $\Pi$ *iff* $\pi$ *is a plan for the symbolic compilation* $\Pi'$ *of* $\Pi$. □

Our symbolic compilation encoding is different from previous compilations that transform PDDL with axioms into PDDL without axioms (Gazen and Knoblock 1997; Garagnani 2000; Davidson and Garagnani 2002; Thiébaux, Hoffmann, and Nebel 2005), since at no time an explicit version of the compiled task (e.g. a new PDDL representation) is created. In the worst case our symbolic compilation also suffers from an exponential blow-up — like any compilation of axioms that preserves plan length does (Thiébaux, Hoffmann, and Nebel 2005). However, the use of a compact symbolic representation helps to alleviate this problem in practice. Another difference is that previous compilations transform derived variables into primary variables so that their value is directly set via auxiliary operators and/or conditional effects. Instead, we replace derived variables by complex formulas over the primary variables in the operator preconditions and goals, resulting in a lower number of variables and operators compared to previous compilations.

[4]For simplification, we ignore conditional effects here. In general, it is necessary to replace derived variables in the condition of each conditional effect.

# Experiments

All presented encodings of axioms are implemented in the SYMBA$^\star$ (Torralba et al. 2014) planner, which is built on top of the FAST DOWNWARD planning system (Helmert 2006).[5] We conduct two types of experiments: first, we evaluate the performance (in terms of coverage) of our symbolic planner for optimal planning on multiple domains with axioms. The second experiment compares the performance of symbolic search (with native axiom support) on different formulations of the same planning problems. More specifically, we use the work of Miura and Fukunaga (2017) to automatically extract axioms from planning problems without axioms and compare the performance of symbolic search on both versions of the same problem. For both experiments we use a time limit of 30 minutes and a 4 GB memory limit for the search. Note that we only compare the actual search and ignore the time and memory necessary for translation and preprocessing. However, planning problems which could not be translated and preprocessed in 30 minutes or 4 GB memory were omitted in the experiments.

The performance of symbolic search is heavily influenced by the exact BDD encoding that is used. Previous research on symbolic search in planning has studied in depth how to construct the BDDs that represent the actions as transition relations (Edelkamp 2001; Torralba et al. 2017), and how to order the variables (Kissmann and Edelkamp 2011; Kissmann and Hoffmann 2013). We use the default settings of SYMBA$^\star$ (Torralba et al. 2014), both for the generation of transition relations and for the order of the variables. Note that the variable order is based on the causal graph, which respects derived variables and axioms (Helmert 2006).

## Optimal Planning with Axioms

Table 1 shows the performance of symbolic uniform-cost search algorithms extended with the presented axiom encodings compared to A$^\star$ with the blind and the (naive) maximum heuristic $h^{max}$ (Ivankovic and Haslum 2015), which are currently the best search techniques for optimal planning with axioms.[6] The benchmark set consists of 20 domains from different fields, sometimes transformed to classical planning, such as verification (Ghosh, Dasgupta, and Ramesh 2015; Edelkamp 2003), multi-agent planning with beliefs (Kominis and Geffner 2015), or elevator control (Koehler and Schuster 2000).

The results of Table 1 show that the action-based encoding already performs competitively in comparison to explicit state search, although explicit A$^\star$ outperforms the symbolic approach if we do not consider the MICONIC-AXIOMS domain. Performing a symbolic fixpoint iteration

[5]Planner and benchmarks are available online: https://gkigit.informatik.uni-freiburg.de/dspeck/fd-symbolic-axioms

[6]The naive $h^{max}$ heuristic of Ivankovic and Haslum (2015) was the dominant heuristic in our experiments.

| | Algorithm | A* | | Action-Based | | Variable-Based | Sym. Compilation | | |
|---|---|---|---|---|---|---|---|---|---|
| ID | Domain (#Tasks) | blind | $h^{max}$ | $\text{fwd}_{\text{def}}$ | $\text{fwd}_{\text{scc}}$ | fwd | fwd | bwd | bid |
| 1 | BLOCKS-AXIOMS (35) | 18 | 18 | 15 | 15 | 15 | 21 | 18 | **30** |
| 2 | GRID-AXIOMS (5) | 1 | 2 | 1 | 1 | 1 | 1 | 0 | **3** |
| 3 | MICONIC-AXIOMS (150) | 60 | 60 | 127 | 120 | **150** | **150** | **150** | **150** |
| 4 | OPTICAL-TELEGRAPHS (48) | 2 | 2 | 2 | 2 | 2 | **4** | 0 | **4** |
| 5 | PSR-MIDDLE (50) | 35 | 35 | 32 | 38 | 39 | **50** | **50** | **50** |
| 6 | PSR-LARGE (50) | 14 | 14 | 13 | 15 | 15 | 24 | 23 | **25** |
| 7 | PHILOSOPHERS (48) | 5 | 5 | 9 | 10 | 9 | **12** | 4 | **12** |
| 8 | ASSEMBLY (30) | 0 | 0 | 6 | 6 | 5 | 9 | 8 | **11** |
| 9 | AIRPORT-ADL (50) | 19 | **21** | 14 | 14 | 12 | 20 | 11 | 19 |
| 10 | TRUCKS (30) | 6 | 8 | **9** | **9** | **9** | **9** | 4 | 8 |
| 11 | BLOCKER (7) | **7** | **7** | 4 | 5 | 5 | 5 | 5 | 5 |
| 12 | SOCIAL-PLANNING (2) | **2** | **2** | **2** | **2** | **2** | **2** | **2** | **2** |
| 13 | SOKOBAN-AXIOMS (25) | 19 | **20** | 7 | 7 | 7 | 18 | **20** | **20** |
| 14 | ACC-CC2 (7) | **7** | **7** | **7** | **7** | **7** | **7** | **7** | **7** |
| 15 | GRID-CC2 (13) | **8** | 7 | 0 | 0 | 0 | 0 | 0 | 0 |
| 16 | COLLAB-AND-COMM (1) | **1** | **1** | 0 | 0 | 0 | 0 | 0 | 0 |
| 17 | MUDDY-CHILDREN (1) | **1** | **1** | **1** | **1** | **1** | **1** | 0 | **1** |
| 18 | MUDDY-CHILD (1) | **1** | **1** | **1** | **1** | **1** | **1** | **1** | **1** |
| 19 | SUM (1) | **1** | **1** | **1** | **1** | **1** | **1** | 0 | **1** |
| 20 | WORD-ROOMS (2) | **2** | **2** | 0 | 0 | 0 | 0 | 0 | 0 |
| | TOTAL COV. W/O MICONIC (406) | 149 | 154 | 124 | 134 | 131 | 185 | 153 | **199** |
| | TOTAL COV. (556) | 209 | 214 | 251 | 254 | 281 | 335 | 303 | **349** |

Table 1: Coverage of the presented symbolic encodings with forward (fwd), backward (bwd) and bidirectional (bid) search and without using mutex information in comparison to explicit A* with blind and (naive) maximum heuristic $h^{max}$ (Ivankovic and Haslum 2015). The action-based encoding is benchmarked with the default stratification $\text{fwd}_{\text{def}}$ and with the strongly connected component stratification $\text{fwd}_{\text{scc}}$. Domains 1-10 are (alternative) formulations of domains with axioms from former IPCs. Domains 8-10 contain complex preconditions which are translated to axioms. Domains 11-13 originate from Ivankovic and Haslum (2015), Domains 14 and 15 originate from Ghosh, Dasgupta, and Ramesh (2015) and Domains 16-20 originate from Kominis and Geffner (2015).

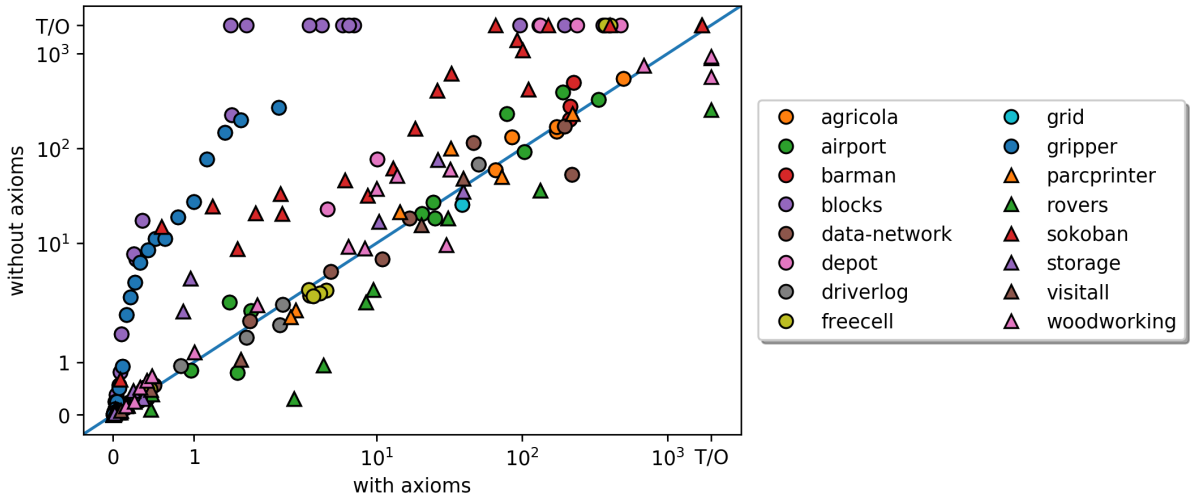


Figure 3: Runtime in seconds of the symbolic compilation (bidirectional search) without using mutex information on formulations of the same planning problems with and without $\epsilon$-axioms (Miura and Fukunaga 2017) on planning domains of IPCs 1998-2018 (optimal tracks).

after each planning step can be time consuming, and while the variable-based encoding overcomes this issue, it still has to evaluate the derived variables after each planning step. Nevertheless, the variable-based encoding performs in general superior in terms of coverage. Symbolic compilation turns out to be the dominant strategy for optimal planning with axioms. It "compiles" the reasoning over derived variables away, and in most cases, this precomputation is fast and pays off. In addition, it allows us to use bidirectional search, since the resulting representation of the planning task does not contain any derived variables. Bidirectional search outperforms all other approaches by a fair margin.

In general, computing the primary representation of each derived variable does not prove to be a limiting factor. In most solved instances the precomputation (Algorithm 2) requires approximately one second of total time. Nevertheless, there are cases where it is not possible to compute the underlying BDD due to the number of derived variables and axioms (e.g. GRID-CC2). Interestingly, for the compilation approach, the bottleneck in some domains is not the construction of the BDDs representing derived variables. In some domains (e.g. WORD-ROOMS) there are operators with a large number of conditional effects which cannot be encoded in a single TR. In PHILOSOPHERS, the substitution of derived variables in the goal formula becomes infeasible. This is related to some cases where it is known that the BDDs representing the set of goal states are exponential in the size of the task (Edelkamp and Kissmann 2008). These cases, unlike in classical planning tasks where the goal is a simple conjunction of facts, may require a more advanced representation of the goal facts, e.g., based on a conjunctive BDD partitioning. Nevertheless, these BDDs are usually either constructed in a few seconds or not at all, so one could use a portfolio approach where an explicit approach is used whenever computing the symbolic representation is unfeasible.

## Different Problem Representations

Miura and Fukunaga (2017) presented a method for automated extraction of axioms from a classical planning problem. This method can be used to create an equivalent planning problem with axioms that is potentially more compact and therefore easier to handle for a planner. Satisficing planners showed that reformulating the planning problem with axioms can improve performance. The question remains whether this also applies to optimal planning. Table 2 shows the coverage and Figure 3 shows the runtime of the symbolic compilation on formulations of the same planning problems with and without axioms. The formulation without axioms corresponds to the standard IPC formulations, while the formulation with axioms contains $\epsilon$-axioms based on the work of Miura and Fukunaga (2017). We have included domains from the optimal tracks of IPCs 1998-2018 in which $\epsilon$-axioms are found and extracted. The coverage and runtime comparison show that the actual search can benefit from the compact encoding with axioms if no mutex information is used. Note that mutexes are used for automated extraction of axioms, therefore we have included the default configuration of SYMBA$^\star$, which performs a bidirectional uniform cost search and uses mutexes to prune the search space (Torralba and Alcázar 2013; Alcázar and Torralba 2015). Interestingly, this configuration solves 50 problems more on the axiom-free problem benchmark set. However, it should be mentioned that mutex detection is more sophisticated for planning without axioms. We leave the generalization of mutex detection for axioms as future work. Nevertheless, these results suggest that symbolic planning may benefit from the support of axioms even for planning problems which do not natively contain axioms.

| Algorithm | SYMBD | | **Sym. Compilation** |
|---|---|---|---|
| Domain | no axioms no mutexes | no axioms mutexes | axioms no mutexes |
| AGRICOLA (20) | 5 | +9 | +1 |
| AIRPORT (50) | 18 | +6 | +0 |
| BARMAN (14) | 3 | +3 | +0 |
| BLOCKS (35) | 19 | +13 | +11 |
| DATA-NETWORK (20) | 13 | +0 | +0 |
| DEPOT (22) | 4 | +3 | +3 |
| DRIVERLOG (20) | 12 | +1 | +0 |
| FREECELL (60) | 10 | +6 | +2 |
| GRID (5) | 2 | +1 | +0 |
| GRIPPER (20) | 20 | +0 | +0 |
| PARCPRINTER (20) | 8 | +7 | +0 |
| PARKING (20) | 0 | +3 | +0 |
| ROVERS (40) | 14 | +0 | −1 |
| SOKOBAN (30) | 19 | +8 | +5 |
| STORAGE (30) | 14 | +0 | +0 |
| TIDYBOT (20) | 0 | +5 | +0 |
| VISITALL (20) | 7 | −1 | −1 |
| WOODWORKING (20) | 16 | +4 | −2 |
| TOTAL COV. (478) | 184 | +68 | +18 |

Table 2: Coverage of the symbolic compilation (bidirectional search) without using mutex information on formulations of the same planning problems with and without $\epsilon$-axioms (Miura and Fukunaga 2017) on domains of IPCs (optimal tracks). SYMBD (Torralba et al. 2014) indicates the bidirectional blind search configuration of the SYMBA$^\star$ planner with mutexes and $h^2$ invariant computation (Torralba and Alcázar 2013; Alcázar and Torralba 2015).

# Conclusion

In this work, we extended symbolic planning to natively handle planning problems with axioms. Overall, we provided three different symbolic encodings of axioms and proved that all are sound and complete. The experiments empirically showed that the presented approach outperforms the previous state of the art in cost-optimal classical planning with axioms. Furthermore, we showed that symbolic planning can benefit from native axiom support, which allows searching on reformulated planning tasks with automatically extracted axioms based on the work of Miura and Fukunaga (2017). For future work, we plan a generalization of mutex detection algorithms (Bonet and Geffner 2001; Torralba and Alcázar 2013; Alcázar and Torralba 2015) to planning tasks with axioms, which promises a further improvement of symbolic planning.

**Acknowledgments.** This work was supported by the German National Science Foundation (DFG) as part of the project EPSDAC (MA 7790/1-1) and the Research Unit FOR 1513 (HYBRIS). The FAI group of Saarland University has received support by DFG grant 389792660 as part of TRR 248 (see https://perspicuous-computing.science). We thank Patrik Haslum and the anonymous reviewers for their comments and suggestions.

## References

Alcázar, V., and Torralba, Á. 2015. A reminder about the importance of computing and exploiting invariants in planning. In *Proc. ICAPS 2015*, 2–6.

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

Bäckström, C., and Nebel, B. 1995. Complexity results for $SAS^+$ planning. *Computational Intelligence* 11(4):625–655.

Bonet, B., and Geffner, H. 2001. Planning as heuristic search. *Artificial Intelligence* 129(1):5–33.

Bryant, R. E. 1986. Graph-based algorithms for boolean function manipulation. *IEEE Transactions on Computers* 35(8):677–691.

Davidson, M., and Garagnani, M. 2002. Pre-processing planning domains containing language axioms. In *Proc. UK PlanSIG workshop*, 23–34.

Edelkamp, S., and Kissmann, P. 2008. Limits and possibilities of BDDs in state space search. In *Proc. AAAI 2008*, 1452–1453.

Edelkamp, S. 2001. Planning with pattern databases. In *Proc. ECP 2001*, 84–90.

Edelkamp, S. 2003. Limits and possibilities of PDDL for model checking software. In *ICAPS 2003 Workshop on the Competition*.

Fikes, R. E., and Nilsson, N. J. 1971. STRIPS: A new approach to the application of theorem proving to problem solving. *Artificial Intelligence* 2:189–208.

Garagnani, M. 2000. A correct algorithm for efficient planning with preprocessed domain axioms. In *Proc. of ES2000 International Conference on Knowledge Based Systems and Applied Artificial Intelligence*. 363–374.

Gazen, B. C., and Knoblock, C. A. 1997. Combining the expressivity of UCPOP with the efficiency of graphplan. In *Proc. of ECP'97*, 221–233.

Geißer, F.; Keller, T.; and Mattmüller, R. 2015. Delete relaxations for planning with state-dependent action costs. In *Proc. IJCAI 2015*, 1573–1579.

Ghosh, K.; Dasgupta, P.; and Ramesh, S. 2015. Automated planning as an early verification tool for distributed control. *Journal of Automated Reasoning* 54(1):31–68.

Haslum, P.; Ivankovic, F.; Ramirez, M.; Gordon, D.; Thiébaux, S.; Shivashankar, V.; and Nau, D. S. 2018. Extending classical planning with state constraints: Heuristics and search for optimal planning. *Journal of Artificial Intelligence Research* 62:373–431.

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

Hoffmann, J., and Edelkamp, S. 2005. The deterministic part of IPC-4: An overview. *Journal of Artificial Intelligence Research* 24:519–579.

Ivankovic, F., and Haslum, P. 2015. Optimal planning with axioms. In *Proc. IJCAI 2015*, 1580–1586.

Kissmann, P., and Edelkamp, S. 2011. Improving cost-optimal domain-independent symbolic planning. In *Proc. AAAI 2011*, 992–997.

Kissmann, P., and Hoffmann, J. 2013. What's in it for my BDD? On causal graphs and variable orders in planning. In *Proc. ICAPS 2013*, 327–331.

Kissmann, P.; Edelkamp, S.; and Hoffmann, J. 2014. Gamer and dynamic-gamer – symbolic search at ipc 2014. In *IPC-8 planner abstracts*, 77–84.

Koehler, J., and Schuster, K. 2000. Elevator control as a planning problem. In *Proc. AIPS 2000*, 331–338.

Kominis, F., and Geffner, H. 2015. Beliefs in multiagent planning: From one agent to many. In *Proc. ICAPS 2015*, 147–155.

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

McMillan, K. L. 1993. *Symbolic Model Checking*. Kluwer Academic Publishers.

Miura, S., and Fukunaga, A. 2017. Automatic extraction of axioms for planning. In *Proc. ICAPS 2017*, 218–227.

Pednault, E. P. D. 1989. ADL: Exploring the middle ground between STRIPS and the situation calculus. In *Proc. KR 1989*, 324–332.

Speck, D.; Geißer, F.; and Mattmüller, R. 2018a. Symbolic planning with edge-valued multi-valued decision diagrams. In *Proc. ICAPS 2018*.

Speck, D.; Geißer, F.; and Mattmüller, R. 2018b. SYMPLE: Symbolic Planning based on EVMDDs. In *IPC-9 planner abstracts*, 82–85.

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(1–2):38–69.

Torralba, Á., and Alcázar, V. 2013. Constrained symbolic search: On mutexes, BDD minimization and more. In *Proc. SoCS 2013*, 175–183.

Torralba, Á.; Alcázar, V.; Borrajo, D.; Kissmann, P.; and Edelkamp, S. 2014. SymBA*: A symbolic bidirectional A* planner. In *IPC-8 planner abstracts*, 105–109.

Torralba, Á.; Alcázar, V.; Kissmann, P.; and Edelkamp, S. 2017. Efficient symbolic search for cost-optimal planning. *Artificial Intelligence* 242:52–79.