Unsolvability Certificates for Classical Planning

Salomé Eriksson and **Gabriele Röger** and **Malte Helmert**
University of Basel, Switzerland
{salome.eriksson,gabriele.roeger,malte.helmert}@unibas.ch

Abstract

The plans that planning systems generate for solvable planning tasks are routinely verified by independent validation tools. For unsolvable planning tasks, no such validation capabilities currently exist. We describe a family of *certificates of unsolvability* for classical planning tasks that can be efficiently verified and are sufficiently general for a wide range of planning approaches including heuristic search with delete relaxation, critical-path, pattern database and linear merge-and-shrink heuristics, symbolic search with binary decision diagrams, and the Trapper algorithm for detecting dead ends. We also augmented a classical planning system with the ability to emit certificates of unsolvability and implemented a planner-independent certificate validation tool. Experiments show that the overhead for producing such certificates is tolerable and that their validation is practically feasible.

Introduction

The aim of classical planning is to find a plan for a given planning task or to prove that the task is unsolvable. While more emphasis has traditionally been placed on finding plans, the problem of proving unsolvability of planning tasks has recently attracted much interest (e. g., Bäckström, Jonsson, and Ståhlberg 2013; Hoffmann, Kissmann, and Torralba 2014; Lipovetzky, Muise, and Geffner 2016).

The more intelligent problem-solving algorithms become part of our everyday lives, the more important it is that we can trust their computations. When a planning system claims to have found a plan for a planning task or claims that it is unsolvable, this claim can be wrong due to software bugs, hardware faults or even malicious reasons. It is thus prudent to perform an independent verification of such claims. For solvable planning tasks, validation tools such as VAL (Howey and Long 2003) and INVAL[1] are commonly used for this purpose.

For unsolvable planning tasks, no such validation capabilities currently exist. For example, the recently held first International Planning Competition for proving unsolvability (*Unsolvability IPC*) included a rule to disqualify planning systems from a particular planning domain when it falsely reports a planning task as unsolvable to discourage the participating planning systems from "guessing".

Verifying the correctness of the output of an algorithm is a need that arises in many areas of computer science. A *certifying algorithm* (McConnell et al. 2011) produces, along with its regular output, a *certificate* (or *witness*) that proves the correctness of the output. The validity of a certificate can be checked by a *verifier*. To be practically useful, certificates should satisfy four criteria:

- *completeness:* There should be a certificate for every unsolvable planning task.
- *efficient generation:* Transforming a non-certifying algorithm into a certifying one should be possible with reasonable (polynomial, ideally linear) overhead.
- *efficient verification:* Verifying the correctness of a certificate should be possible with reasonable effort (e.g., polynomial in the size of the certificate).
- *generality:* A single class of certificates (and hence verifier) should be useful for a wide range of algorithms.

For example, for the unsatisfiability problem of propositional logic formulas in conjunctive normal form (UNSAT), *resolution proofs* are a form of certificate that can be reasonably efficiently supported by the majority of state-of-the-art UNSAT algorithms (Beame, Kautz, and Sabharwal 2004) and are one of a small number of certificate types permitted in the *Certified UNSAT* tracks of recent SAT competitions.[2]

In this paper, we introduce a family of certificates that can be used to prove the unsolvability of classical planning tasks. The central notion underlying these certificates are *inductive sets* of states that separate the initial state of a planning task from its goal states. Using the established approach of representing sets of states over propositional state variables as logical formulas or circuits, we utilize known tractability results for knowledge representation formalisms in propositional logic (e. g., Dowling and Gallier 1984; Darwiche and Marquis 2002) to define classes of unsolvability certificates that can be efficiently verified. We then show that a number of non-certifying planning approaches in the literature based on heuristic or symbolic search can be augmented to produce certificates with polynomial overhead. Finally, we describe a prototype implementation of a

[1]`http://users.cecs.anu.edu.au/~patrik/`

[2]`http://satcompetition.org/`

certifying planning algorithm and of a stand-alone verifier for unsolvability certificates and experimentally show that certificate verification, though computationally expensive, is practically feasible.

Planning Tasks

We consider planning tasks in propositional STRIPS representation (Fikes and Nilsson 1971). We do not consider action costs, which do not matter for unsolvable tasks. Planning tasks in SAS$^+$ representation (Bäckström and Nebel 1995) can easily be converted to STRIPS.

A *STRIPS planning task* is a tuple $\Pi = \langle V, A, I, G \rangle$, where V is a finite set of (state) *variables* or *atoms*, A is a finite set of *actions* or *operators*, $I \subseteq V$ is the *initial state* and $G \subseteq V$ is the *goal*. We write $\|\Pi\|$ for the size of a compact description of Π.

A state $s \subseteq V$ of Π is defined by the atoms that are *true* in this state. A state s is a *goal state* if $G \subseteq s$. We write $S(\Pi) = 2^V$ for the set of states of Π and $S_G(\Pi)$ for the set of goal states of Π.

Each operator $o \in A$ is defined by its *preconditions* $\mathit{pre}(o) \subseteq V$, *add effects* $\mathit{add}(o) \subseteq V$ and *delete effects* $\mathit{del}(o) \subseteq V$. It is *applicable* in state s if $\mathit{pre}(o) \subseteq s$, and the resulting *successor state* is $s[o] = (s \setminus \mathit{del}(o)) \cup \mathit{add}(o)$.

An *s-plan* for a state s is a sequence of operators that are successively applicable in s and result in a goal state. An I-plan is also called a *plan for* Π. A state s is *unsolvable* if no s-plan exists, and Π is unsolvable if no I-plan exists.

Given a state set $S \subseteq S(\Pi)$ and operator $o \in A$, we write $S[o] = \{s[o] \mid s \in S, o \text{ applicable in } s\}$ for the set of all successors of S via operator o, and for operator sets $O \subseteq A$ we write $S[O] = \bigcup_{o \in O} S[o]$ for all successors of S via any operator in O. The successor set is *monotonic* in S and O, i.e., $S[O] \subseteq S'[O']$ for all $S \subseteq S'$ and $O \subseteq O'$.

Sets of States as Logical Formulas

A critical aspect of the unsolvability certificates we consider is the compact representation of sets of states. Following ideas used in planning as satisfiability (Kautz and Selman 1992) and planning as symbolic search (Edelkamp and Helmert 2001), we use logical formulas to represent sets of states. Specifically, consider a propositional formula φ whose propositional variables correspond to the state variables V of a planning task. We say that a state s *satisfies* φ if the interpretation where exactly the variables in s are true satisfies φ. The set of all such states is denoted by $\mathit{states}(\varphi)$.

Representing Logical Formulas

More generally, instead of logical formulas we can use any formalism $\mathbf{R}$ for representing sets of truth assignments for a set of propositional logic variables $\mathbf{V}$. Examples of such formalisms include the class of formulas over $\mathbf{V}$ in conjunctive normal form (CNF) or the set of reduced ordered binary decision diagrams (ROBDDs, in the following BDDs for short) for a given variable order on $\mathbf{V}$ (Bryant 1986). Because variable orders are important for BDDs, we assume that $\mathbf{V}$ is totally ordered and write $\prec$ for this variable order.

An $\mathbf{R}$*-formula* φ is a particular instance of formalism $\mathbf{R}$, such as a particular CNF formula or a particular BDD. We use φ to refer both to the represented formula and its representation and write $\|\varphi\|$ for its representation size. For example, if φ is represented as a BDD, $\|\varphi\|$ is the number of BDD nodes. We write $\mathit{vars}(\varphi)$ for the propositional variables used in formula φ. In general, $\mathit{vars}(\varphi)$ may include variables that do not directly correspond to state variables of the planning task, in which case $\mathit{states}(\varphi)$ is not defined.

Following Darwiche and Marquis (2002), we can view $\mathbf{R}$-formulas as abstract data structures where different formalisms $\mathbf{R}$ differ in the operations they support efficiently. For example, testing satisfiability of an $\mathbf{R}$-formula is efficient for BDDs but NP-complete for general CNF. Rather than committing to a specific representation for certificates, we prove results that hold for any formalism $\mathbf{R}$ that efficiently supports certain operations on $\mathbf{R}$-formulas. The following list describes the operations used in this paper, some of which are taken from Darwiche and Marquis (2002):

MO (model testing) Given $\mathbf{R}$-formula φ and truth assignment I, test if $I \models \varphi$.

CO (consistency) Given $\mathbf{R}$-formula φ, test if φ is satisfiable.

CE (clausal entailment) Given $\mathbf{R}$-formula φ and clause γ, test if $\varphi \models \gamma$.

SE (sentential entailment) Given $\mathbf{R}$-formulas φ and ψ, test if $\varphi \models \psi$.

$\wedge$BC (bounded conjunction) Given $\mathbf{R}$-formulas φ and ψ, construct an $\mathbf{R}$-formula representing $\varphi \wedge \psi$.

$\wedge$C (general conjunction) Given $\mathbf{R}$-formulas $\varphi_1, \dots, \varphi_n$, construct an $\mathbf{R}$-formula representing $\varphi_1 \wedge \dots \wedge \varphi_n$.

$\neg$C (negation) Given $\mathbf{R}$-formula φ, construct an $\mathbf{R}$-formula representing $\neg\varphi$.

CL (conjunction of literals) Given a conjunction φ of literals, construct an $\mathbf{R}$-formula representing φ.

RN (renaming) Given $\mathbf{R}$-formula φ and an injective variable renaming $r : \mathit{vars}(\varphi) \to \mathbf{V}$, construct an $\mathbf{R}$-formula representing $\varphi[r]$, i.e., φ with each variable X replaced by $r(X)$.

RN$_\prec$ (renaming consistent with order) Same as **RN**, but r must be consistent with the variable order in the sense that if $V, V' \in \mathit{vars}(\varphi)$ with $V \prec V'$, then $r(V) \prec r(V')$.

BI (biimplication pairs) Given a disjoint set S of pairs of propositional variables, construct an $\mathbf{R}$-formula representing $\bigwedge_{\{v,v'\} \in S} (v \leftrightarrow v')$.

BI$_\prec$ (biimplication pairs consistent with order) Same as **BI**, but S must be consistent with $\prec$ in the sense that if $\{V, V'\} \in S$ with $V \prec V'$, then no other pair in S may contain a variable V'' with $V \prec V'' \prec V'$.

We say that a formalism $\mathbf{R}$ *efficiently supports* one of these operations if the operation can be performed in polynomial time in the representation size of the inputs. Our main focus is on three formalisms: Horn formulas (CNF formulas with at most one positive literal per clause), 2-CNF formulas (CNF formulas with at most two literals per clause, also called Krom formulas), and BDDs.

BDDs are very common for representing sets of states, e. g. in symbolic search. Horn formulas and 2-CNF formulas are interesting due to their role as maximal tractable classes in Schaefer's dichotomy (1978) for Boolean constraint satisfaction, and we will later see certificates that can be compactly represented in these formalisms but not as BDDs.

Horn and 2-CNF formulas efficiently support all listed operations except $\neg$**C**. BDDs efficiently support all listed operations except $\wedge$**C**, **RN** and **BI**. In particular they do efficiently support $\wedge$**BC**, **RN**$_{\prec}$ and **BI**$_{\prec}$, the weaker forms of the three unsupported operations.

Inductive Certificates

We are now ready to define unsolvability certificates for planning. The key concept is the notion of *inductive sets*, i.e., sets of states that are closed under operator application.

Definition 1 (inductive set)**.** *A set $S \subseteq S(\Pi)$ of states of a STRIPS planning task $\Pi = \langle V, A, I, G\rangle$ is* inductive in Π *if $S[A] \subseteq S$, i.e., all operator applications in a state in S lead to a state in S.*

The key property of inductive sets is that they are impossible to leave: once a state inside S is reached, all further operator applications stay within S. For this reason, Lipovetzky et al. (2016) call formulas that describe inductive sets *traps*.

If an inductive set contains no goal state, then there is no plan for any of its states. Therefore, we can use inductive sets as a basis for unsolvability certificates.

Definition 2 (inductive certificate)**.** *Given a task $\Pi = \langle V, A, I, G\rangle$, an* inductive certificate *for a state $s \in S(\Pi)$ is given by a set $S \subseteq S(\Pi)$ of states such that*

1. *$s \in S$,*
2. *$S \cap S_G(\Pi) = \emptyset$, and*
3. *S is inductive in Π.*

An inductive certificate for the initial state I is also called an inductive certificate for Π.

For now, we focus on the mathematical properties of inductive certificates, leaving the question of how to compactly represent and efficiently verify them for the next section. It is easy to see that an inductive certificate for a state exists iff there is no solution from that state.

Proposition 1 (soundness and completeness)**.** *Let $\Pi = \langle V, A, I, G\rangle$ be a STRIPS planning task. There is an inductive certificate for state $s \in S(\Pi)$ iff s is unsolvable. There is an inductive certificate for Π iff Π is unsolvable.*

To see that inductive certificates always exist for unsolvable states s, observe that the set of states reachable from s satisfies properties 1–3 of Definition 2 if s is unsolvable. To see that no inductive certificate exists if s is solvable, note that an inductive set S for s must include s (by property 1) and hence all states reachable from s (by property 3). Hence, if there is an s-plan, S must violate property 2.

Before we move on to more complex certificates, we observe a structural property of inductive certificates.

Theorem 1. *Inductive certificates are closed under union and intersection.*

Proof. Let S and S' be certificates for state s. From $s \in S$ and $s \in S'$ we get $s \in S \cup S'$ and $s \in S \cap S'$ (property 1).

As there is no goal state in S or in S', there is no goal state in $S \cup S'$ or $S \cap S'$ (property 2).

For the inductivity of the union, consider a state $s' \in S \cup S'$ and assume w.l.o.g. that $s' \in S$. We conclude from the inductivity of S that $s'[A] \subseteq S \subseteq S \cup S'$. For the inductivity of the intersection, consider $s' \in S \cap S'$ and $o \in A$ such that o is applicable in s'. Since $s' \in (S \cap S') \subseteq S$ and S is inductive, we have $s'[o] \in S$. With the same argument, we have $s'[o] \in S'$ and hence $s'[o] \in S \cap S'$. This shows property 3. $\square$

Inductive certificates are general in the sense that they can be generated for every unsolvable state. However, actually constructing and verifying an inductive certificate as a single monolithic set can be computationally challenging. We therefore introduce two "factored" representations of inductive certificates, which will be useful for efficient certificate generation and verification.

Definition 3 (disjunctive/conjunctive certificate)**.** *A family $\mathcal{F} \subseteq 2^{S(\Pi)}$ of state sets of task Π is called a* disjunctive certificate *for state s of Π if $\bigcup_{S' \in \mathcal{F}} S'$ is an inductive certificate for s. It is called a* conjunctive certificate *for s if $\bigcap_{S' \in \mathcal{F}} S'$ is an inductive certificate for s.*

Disjunctive and conjunctive certificates allow combining different sources of information without having to explicitly compute the union/intersection of the component sets. However, verifying the conditions for inductive certificates in general requires considering the full union/intersection, negating this advantage. For this reason, we now introduce a more restrictive form of disjunctive and conjunctive certificates which allows verifying the certificate properties in a factored way.

These certificates are parameterized by a number r which determines how many component sets must be considered at the same time. We will later show that with suitable representations, such certificates can be verified in polynomial time for any fixed r.

Definition 4 (r-disjunctive certificate)**.** *For $r \in \mathbb{N}_0$, a family $\mathcal{F} \subseteq 2^{S(\Pi)}$ of state sets of task $\Pi = \langle V, A, I, G\rangle$ is called an* r-disjunctive certificate *for state s if:*

1. *$s \in S'$ for some $S' \in \mathcal{F}$,*
2. *$S' \cap S_G(\Pi) = \emptyset$ for all $S' \in \mathcal{F}$, and*
3. *for all $S' \in \mathcal{F}$ and all $o \in A$, there is a subfamily $\mathcal{F}' \subseteq \mathcal{F}$ with $|\mathcal{F}'| \leq r$ such that $S'[o] \subseteq \bigcup_{S'' \in \mathcal{F}'} S''$.*

We call the third property of the definition *disjunctive r-inductivity*. Essentially, it means that to verify the inductivity property for a component set, only r subsets need to be considered at the same time. We now show that this is a sufficient condition for inductivity.

Theorem 2. *All r-disjunctive certificates for a state s are disjunctive certificates for s.*

Proof. Let $\mathcal{F}$ be an r-disjunctive certificate for s, and let $S = \bigcup_{S' \in \mathcal{F}} S'$. We show that each of the three properties of Def. 4 (for $\mathcal{F}$) implies the corresponding property of Def. 2 (for S). Properties 1 and 2 of the two definitions are equivalent by simple set arithmetic.

For property 3, consider any state $s' \in S$ and any operator o applicable in s'. Because $S = \bigcup_{S' \in \mathcal{F}} S'$, there exists a set $S' \in \mathcal{F}$ with $s' \in S'$. From Def. 4, there must be a subfamily $\mathcal{F}' \subseteq \mathcal{F}$ such that $s'[o] \in S'[o] \subseteq \bigcup_{S'' \in \mathcal{F}'} S'' \subseteq \bigcup_{S' \in \mathcal{F}} S' = S$, and hence S is inductive. □

We can use the same idea to define *r-conjunctive certificates* as a variant of conjunctive certificates that allows factored verification. Here, the bound r is also applied to restrict the complexity of verifying that the inductive set contains no goal states.

Definition 5 (r-conjunctive certificate). *For $r \in \mathbb{N}_0$, a family $\mathcal{F} \subseteq 2^{S(\Pi)}$ of state sets of task $\Pi = \langle V, A, I, G \rangle$ is called an* r-conjunctive certificate *for state s if:*

1. *$s \in S'$ for all $S' \in \mathcal{F}$,*
2. *there is a subfamily $\mathcal{F}' \subseteq \mathcal{F}$ with $|\mathcal{F}'| \leq r$ such that $(\bigcap_{S'' \in \mathcal{F}'} S'') \cap S_G(\Pi) = \emptyset$, and*
3. *for all $S' \in \mathcal{F}$ and all $o \in A$, there is a subfamily $\mathcal{F}' \subseteq \mathcal{F}$ with $|\mathcal{F}'| \leq r$ such that $(\bigcap_{S'' \in \mathcal{F}'} S'')[o] \subseteq S'$.*

We call the third property of this definition *conjunctive r-inductivity*. As in the disjunctive case, we can show that r-conjunctive certificates are conjunctive certificates.

Theorem 3. *All r-conjunctive certificates for a state s are conjunctive certificates for s.*

Proof. Let $\mathcal{F}$ be an r-conjunctive certificate for s, and let $S = \bigcap_{S' \in \mathcal{F}} S'$. We show that each of the three properties of Def. 5 (for $\mathcal{F}$) implies the corresponding property of Def. 2 (for S). Properties 1 are equivalent, and it is easy to verify that property 2 of Def. 5 implies property 2 of Def. 2.

For property 3, consider any state $s' \in S$, any set $S' \in \mathcal{F}$ and any operator o applicable in s'. Because $S = \bigcap_{S' \in \mathcal{F}} S'$, s' is contained in all sets of the form $\bigcap_{S'' \in \mathcal{F}'} S''$ with $\mathcal{F}' \subseteq \mathcal{F}$. From Def. 5, there is a subfamily $\mathcal{F}'$ with $s'[o] \in \bigcap_{S'' \in \mathcal{F}'} S''[o] \subseteq S'$. Therefore, $s'[o]$ is contained in *all* sets $S' \in \mathcal{F}$ and hence also in their intersection, S. □

Efficient Certificate Verification

An inductive certificate for a state s is always an over-approximation of all states reachable from s. Since the number of reachable states can be huge, we require compact representations of such certificates. In the following, we identify representations that permit polynomial verification of the certificates. For this purpose, we make the representation formalism $\mathbf{R}$ explicit.

Definition 6 (inductive $\mathbf{R}$-certificate). *Let $\Pi = \langle V, A, I, G \rangle$ be a STRIPS planning task. An* inductive $\mathbf{R}$-certificate *for state $s \in S(\Pi)$ is an $\mathbf{R}$-formula φ with vars$(\varphi) \subseteq V$ such that states(φ) is an inductive certificate for s.*

For the composite certificates, the $\mathbf{R}$-representation represents each component as an $\mathbf{R}$-formula.

Definition 7 (composite $\mathbf{R}$-certificates). *Let $\Pi = \langle V, A, I, G \rangle$ be a STRIPS planning task. A* disjunctive/conjunctive/r-disjunctive/r-conjunctive $\mathbf{R}$-certificate *for state $s \in S(\Pi)$ is a set Φ of $\mathbf{R}$-formulas with vars$(\varphi) \subseteq V$ for all $\varphi \in \Phi$ such that $\{$states$(\varphi) \mid \varphi \in \Phi\}$ is a disjunctive/conjuctive/r-disjunctive/r-conjunctive certificate for s.*

The following theorems establish a set of requirements on the representation formalism that allow $\mathbf{R}$-certificates to be verified efficiently.

Theorem 4. *Inductive $\mathbf{R}$-certificates φ for state s of task Π can be verified in polynomial time in $\|\varphi\|$ and $\|\Pi\|$ if $\mathbf{R}$ efficiently supports **MO**, **CE**, **SE**, $\wedge$**BC**, **CL**, **RN**$_\prec$ and **BI**$_\prec$.*

Proof. To verify that φ is an inductive $\mathbf{R}$-certificate for state s of task $\Pi = \langle V, A, I, G \rangle$, we must verify that it satisfies the three requirements of Definition 2, where $S =$ *states*(φ).

Requirement 1 ($s \in$ *states*(φ)) can be verified using **MO** by testing $I \models \varphi$ with the truth assignment I defined as $I(v) = \mathbf{T}$ for $v \in s$ and $I(v) = \mathbf{F}$ otherwise.

Requirement 2 (*states*$(\varphi) \cap S_G(\Pi) = \emptyset$) can be verified as $\varphi \models \bigvee_{v \in G} \neg v$ using **CE**.

For requirement 3 (*states*(φ) is inductive in Π), we can verify for each operator $o \in A$ individually that *states*$(\varphi)[o] \subseteq$ *states*(φ). For this purpose we need a fresh auxiliary propositional variable $v' \in \mathbf{V}$ for every state variable $v \in V$. We write V' for these new ("primed") variables. The original ("unprimed") variables V are used to describe a state in which o is applied, and the primed variables V' describe the successor state. We ensure that each primed variable is adjacent to its unprimed variable in the variable order: if $v_1 \prec \cdots \prec v_n$ is the order on V, then $v_1 \prec v_1' \prec \cdots \prec v_n \prec v_n'$ is the order on $V \cup V'$.

We then proceed as follows:

1. Build the transition $\mathbf{R}$-formula τ_o over $V \cup V'$ that represents $\bigwedge_{v \in pre(o)} v \wedge \bigwedge_{v \in add(o)} v' \wedge \bigwedge_{v \in del(o) \setminus add(o)} \neg v' \wedge \bigwedge_{v \in V \setminus (add(o) \cup del(o))} (v \leftrightarrow v')$. This can be done in polynomial time using **CL**, **BI**$_\prec$ and $\wedge$**BC**. This $\mathbf{R}$-formula represents all state pairs $\langle s, s' \rangle$ with $s' = s[o]$.
2. Build the $\mathbf{R}$-formula $\varphi \wedge \tau_o$. This can be done in polynomial time using $\wedge$**BC**. This $\mathbf{R}$-formula represents all pairs $\langle s, s' \rangle$ with $s' = s[o]$ and $s \in$ *states*(φ).
3. Test if $\varphi \wedge \tau_o \models \varphi[V \to V']$, where $\varphi[V \to V']$ is φ with each unprimed variable renamed to the corresponding primed variable. This can be done in polynomial time using **RN**$_\prec$ and **SE**. The test succeeds iff $s' \in$ *states*(φ) for all states s' with $s' = s[o]$ and $s \in$ *states*(φ). In other words, it succeeds iff φ satisfies the inductivity property for operator o.

In summary, all tests can be performed in polynomial time and succeed iff φ is an inductive $\mathbf{R}$-certificate. □

We remark that BDDs, Horn formulas and 2-CNF formulas satisfy all requirements of the theorem and are hence all suitable representations for inductive $\mathbf{R}$-certificates.

An analogous result using the same requirements on $\mathbf{R}$ can be proved for r-conjunctive $\mathbf{R}$-certificates. The representation size of a composite certificate Φ is defined as $\|\Phi\| = \sum_{\varphi \in \Phi} \|\varphi\|$.

Theorem 5. *For fixed r, r-conjunctive $\mathbf{R}$-certificates Φ for state s of task Π can be verified in polynomial time in $\|\Phi\|$ and $\|\Pi\|$ if $\mathbf{R}$ efficiently supports **MO**, **CE**, **SE**, $\wedge$**BC**, **CL**, **RN**$_\prec$ and **BI**$_\prec$.*

Proof. Let Φ be an r-conjunctive $\mathbf{R}$-certificate for state s of task $\Pi = \langle V, A, I, G\rangle$. We must verify the three requirements of Definition 5.

For requirement 1 (initial state), we must verify that $s \in$ *states*(φ') for all $\varphi' \in \Phi$. This can be done in polynomial time using **MO**, iterating over all $\varphi' \in \Phi$.

For requirement 2 (non-inclusion of goal states), we must test if there exists some $\Phi' \subseteq \Phi$ with $|\Phi'| \leq r$ such that $\bigwedge_{\varphi'' \in \Phi'} \varphi'' \models \bigvee_{v \in G} \neg v$. While building $\bigwedge_{\varphi'' \in \Phi'} \varphi''$ can be exponential in r, for fixed r the runtime is polynomial using $\wedge$**BC** and **CE**. The number of candidate subsets Φ' is $O(|\Phi|^r)$.

For requirement 3 (conjunctive r-inductivity), we must verify for all $\varphi' \in \Phi$ and all $o \in A$ that there exists some $\Phi' \subseteq \Phi$ with $|\Phi'| \leq r$ s.t. $(\bigcap_{\varphi'' \in \Phi'} \mathit{states}(\varphi''))[o] \subseteq \mathit{states}(\varphi')$. We test this separately for each φ', o and Φ', amounting to a polynomial number of tests.

Each test can be performed on $\mathbf{R}$-formulas as $(\bigwedge_{\varphi'' \in \Phi'} \varphi'') \wedge \tau_o \models \varphi'[V \to V']$, where the transition $\mathbf{R}$-formula τ_o is computed in polynomial time using **CL**, **BI**$_\prec$ and $\wedge$**BC** as in Theorem 4, and the conjunctions, renaming and entailment test can be performed in polynomial time using $\wedge$**BC**, **RN**$_\prec$ and **SE**. $\square$

The same proof idea can be used for conjunctive certificates (without the parameter r) if $\mathbf{R}$ efficiently supports unbounded conjunction ($\wedge$**C**). However, in this case conjunctive certificates are not actually necessary because they can be efficiently converted to (monolithic) inductive $\mathbf{R}$-certificates. We remark that Horn and 2-CNF formulas efficiently support unbounded conjunction, but BDDs do not.

To complete our results on efficient certificate verification, we now consider (r-) disjunctive certificates. In this case, the requirements on $\mathbf{R}$ are slightly different, requiring $\neg$**C** and **CO** instead of **SE**.

Theorem 6. *For fixed r, r-disjunctive $\mathbf{R}$-certificates Φ for state s of task Π can be verified in polynomial time in $\|\Phi\|$ and $\|\Pi\|$ if $\mathbf{R}$ efficiently supports **MO**, **CO**, **CE**, $\wedge$**BC**, $\neg$**C**, **CL**, **RN**$_\prec$ and **BI**$_\prec$.*

Proof. Let Φ be an r-disjunctive $\mathbf{R}$-certificate for state s of task $\Pi = \langle V, A, I, G\rangle$. We must verify the three requirements of Definition 4.

For requirement 1 (initial state), we must verify that $s \in$ *states*(φ') for some $\varphi' \in \Phi$. This can be done in polynomial time using **MO**, iterating over all $\varphi' \in \Phi$.

For requirement 2 (non-inclusion of goal states), we must verify that *states*$(\varphi') \cap S_G(\Pi) = \emptyset$ for all $\varphi' \in \Phi$. Each test can be performed in polynomial time using **CE**, as in Theorem 4.

For requirement 3 (disjunctive r-inductivity), we test *states*$(\varphi')[o] \subseteq \bigcup_{\varphi'' \in \Phi'} \mathit{states}(\varphi'')$ for all $\varphi' \in \Phi$, $o \in A$ and $\Phi' \subseteq \Phi$ with $|\Phi'| \leq r$ and combine the results appropriately. This is a polynomial number of tests.

Each test can be performed following the same ideas as in Theorem 4. We first construct τ_o (using **CL**, **BI**$_\prec$ and $\wedge$**BC**) and then test $\varphi' \wedge \tau_o \models (\bigvee_{\varphi'' \in \Phi'} \varphi'')[V \to V']$. This is equivalent to testing if $\varphi' \wedge \tau_o \wedge (\bigwedge_{\varphi'' \in \Phi'} \neg\varphi'')[V \to V']$ is unsatisfiable, which can be done in polynomial time using $\wedge$**BC**, $\neg$**C**, **RN**$_\prec$ and **CO**. $\square$

Similar to the conjunctive case, the result can be generalized to general disjunctive certificates (with no bound on r) by requiring $\wedge$**C** instead of $\wedge$**BC**, but this is somewhat pointless because in this case $\wedge$**C** and $\neg$**C** can be used to efficiently perform unbounded disjunction (using De Morgan's laws) and hence convert the disjunctive certificate into a monolithic inductive certificate.

While BDDs satisfy all requirements of Theorem 6, Horn formulas and 2-CNF formulas do not, as they lack support for $\neg$**C**. However, for the special case of $r = 1$, which the following section shows to be relevant for heuristic search and the Trapper algorithm, we can use **SE** instead of $\neg$**C**.

Theorem 7. *1-disjunctive $\mathbf{R}$-certificates Φ for state s of task Π can be verified in polynomial time in $\|\Phi\|$ and $\|\Pi\|$ if $\mathbf{R}$ efficiently supports **MO**, **CO**, **CE**, **SE**, $\wedge$**BC**, **CL**, **RN**$_\prec$ and **BI**$_\prec$.*

Proof. The proof is identical to the one for Theorem 6 except that the test $\varphi' \wedge \tau_o \models (\bigvee_{\varphi'' \in \Phi'} \varphi'')[V \to V']$ is handled differently. For 1-disjunctive certificates we have $|\Phi'| \leq 1$, and hence the disjunction is either empty or a singleton. If $\Phi' = \emptyset$, we can use **CO** to test whether $\varphi' \wedge \tau_o$ is unsatisfiable (= entails the empty disjunction), and if $\Phi' = \{\varphi''\}$, we can use **SE** to test whether $\varphi' \wedge \tau_o \models \varphi''[V \to V']$. $\square$

We conclude this section by summarizing the results for the three main representation formalisms we consider:

- Inductive certificates can be verified efficiently when represented as BDDs, Horn formulas or 2-CNF formulas.
- r-conjunctive certificates can be verified efficiently when represented as BDDs, Horn formulas or 2-CNF formulas. Horn formulas and 2-CNF formulas also support efficient verification of unrestricted conjunctive certificates.
- r-disjunctive certificates can be verified efficiently when represented as BDDs, and in the case $r = 1$ also when represented as Horn formulas or 2-CNF formulas.

Certifying Planning Algorithms

To demonstrate the practical utility of the family of unsolvability certificates we introduced, we now show how a range of common planning algorithms can be converted into certifying algorithms. Unless noted otherwise, the overhead for producing a certificate is bounded by a multiplicative constant and hence does not change the big-O complexity of the algorithm.

Blind Search (Explicit and Symbolic)

Arguably the simplest planning algorithm is blind forward search. To prove a task unsolvable, an explicit-state search must explore all states S_{reach} reachable from the initial state and show that they include no goal state. The set S_{reach} can be efficiently converted into a BDD,[3] yielding an inductive BDD certificate.

Symbolic blind search algorithms are among the strongest current planning algorithms (e. g., Torralba 2015). A symbolic *progression* search proving unsolvability produces a BDD for the *closed set* that, upon termination, contains all reachable states and is an inductive BDD certificate.

Similarly, a symbolic *regression* search proving unsolvability computes a BDD representing all states from which a goal state can be reached and shows that it does not include the initial state. The BDD for the complement set, which can be efficiently computed, is an inductive BDD certificate.

A symbolic *bidirectional* search proving unsolvability terminates either when it has proved that the goal is unreachable in the forward direction or that the initial state is unreachable in the backward direction. In the first case, it can use the same certificate as a forward search, and in the second case the same certificate as a backward search.

Infinite Heuristic Estimates

Distance heuristics h are useful for proving a task unsolvable if $h(s) = \infty$, i.e., if they can show that there is no solution from a given state s. We only consider heuristics that are *safe*, i.e., that only assign infinite values to unsolvable states. In the following, we show how to compute certificates for states s with $h(s) = \infty$ for a range of common heuristics for classical planning.

Merge-and-Shrink and PDB Heuristics For every consistent heuristic, the set of states with $h(s) = \infty$ is an inductive set, and for every admissible heuristic, this set contains no goal state. The cascading tables representation of a merge-and-shrink (M&S) heuristic (Helmert, Haslum, and Hoffmann 2007; Helmert et al. 2014) with a *linear* merge strategy can be transformed into an algebraic decision diagram (Bahar et al. 1993) representing the same function in linear time (Torralba 2015). From this decision diagram, a BDD for all states with infinite heuristic estimate can be extracted in linear time. For "pure" M&S heuristics as described in the literature, this results in an inductive BDD certificate for *all* states pruned by the heuristic.[4]

[3]We can construct a binary decision tree for the set of states in polynomial time as follows: convert the set into a trie by treating each state as a string. Then mark the labels of the trie with 1 and add a leaf labeled with 0 to each unary node as the second child. Applying the standard Shannon and isomorphism reductions to this decision diagram yields a BDD.

[4]Practical implementations of M&S heuristics often deviate from theory by assigning $h(s) = \infty$ to states that are known to be unreachable from the initial state, even though the goal might be reachable from s. Such M&S heuristics are inadmissible, but still usable for optimal search. They require more advanced techniques for generating certificates.

While the limitation to linear merge strategies is a restriction in general, the existing work on tailoring M&S to proving unsolvability (Hoffmann, Kissmann, and Torralba 2014) only uses linear merge strategies.

The M&S result directly carries over to pattern database heuristics (Edelkamp 2001), which are a special case of M&S with linear merge strategies.

Delete Relaxation Delete relaxation heuristics like h^+, h^{max}, h^{add}, h^{FF} and $h^{\text{LM-Cut}}$ (e. g., Bonet and Geffner 2001; Hoffmann and Nebel 2001) all assign an infinite heuristic value to the same set of states, so it suffices to discuss one of them. Helmert and Domshlak (2009) describe a compilation from h^{max} to linear M&S heuristics which implies that the result for M&S carries over to h^{max}: if $h^{\text{max}}(s) = \infty$, there exists an inductive BDD certificate for s. However, as the compilation is state-dependent, different states may require different certificates.

By following their construction, we can see that the full expressive power of BDDs is not needed for delete relaxation heuristics. For a given state s, let $R^+_{\text{unreach}}(s)$ denote the set of atoms that are not reachable from s in the delete relaxation. $R^+_{\text{unreach}}(s)$ can be computed in $O(\|\Pi\|)$. It is easy to see that $h^{\text{max}}(s) = \infty$ iff $R^+_{\text{unreach}}(s)$ contains a goal atom and that the formula $\bigwedge_{v \in R^+_{\text{unreach}}(s)} \neg v$ describes an inductive certificate for s in this case. This formula can be compactly represented as a BDD with an arbitrary variable order, as a Horn formula, or as a 2-CNF formula.

Critical Path Heuristics For critical path heuristics, we show how to efficiently compute unsolvability certificates for the h^m heuristic family (Haslum and Geffner 2000). We conjecture that similar results can be shown for generalizations of h^m based on the Π^C compilation (Keyder, Hoffmann, and Haslum 2014).

We base the treatment of h^m on the Π^m compilation (Haslum 2009). For a given task Π and natural number $m > 0$, Π^m is a delete-free planning task with $h^m_\Pi(s) = h^{\text{max}}_{\Pi^m}(s^m)$ for all states s, where s^m is a state in Π^m that corresponds to state s in Π. (Subscripts denote in which task a heuristic is evaluated.) In particular, the correspondence between h^m_Π and $h^{\text{max}}_{\Pi^m}$ implies that $h^m(s) = \infty$ iff s^m is unsolvable in the delete-free task Π^m.

The key idea of the Π^m compilation is to explicitly represent conjunctions $c = \{v_1, \dots, v_k\} \subseteq V$ of up to m atoms by new atoms π_c such that whenever π_c is unreachable in Π^m, the conjunction $v_1 \wedge \dots \wedge v_k$ is unreachable in Π. Let Π^m_{unreach} denote the set of unreachable atoms in Π^m. (Computing this set is a side effect of computing h^m.) Using the result for delete relaxation above, it follows that the formula $\varphi = \bigwedge_{\{v_1,\dots,v_k\} \in \Pi^m_{\text{unreach}}} \neg(v_1 \wedge \dots \wedge v_k)$ describes an inductive certificate for s when $h^m(s) = \infty$.

We can write φ in clause form as $\bigwedge_{\{v_1,\dots,v_k\} \in \Pi^m_{\text{unreach}}} (\neg v_1 \vee \dots \vee \neg v_k)$, which shows that inductive Horn formula certificates for states with $h^m(s) = \infty$ can be efficiently generated. In the common case $m \leq 2$, φ is also a 2-CNF formula.

However, even in the limited case of $m = 2$, φ is in general not compactly representable as a BDD (cf. Edelkamp and Kissmann 2011), so this result does not directly result in efficiently computable inductive BDD certificates for h^m. However, if we consider each clause of φ as a separate formula, every individual clause can be efficiently converted into a BDD, and hence this collection of BDDs forms a *conjunctive* BDD certificate for s. Moreover, it can be shown that this certificate is actually 1-conjunctive and hence can be efficiently verified.[5]

Landmarks The landmark heuristics described in the literature can all be understood as landmarks of the delete relaxation or of the Π^m compilation (e. g., Keyder, Richter, and Helmert 2010; Bonet and Helmert 2010; Bonet and Castillo 2011), and hence states with infinite landmark heuristic estimates are covered by the previous results.

Heuristic Forward State-Space Search

Many classical planning systems employ heuristic search algorithms such as A*, IDA* or greedy best-first search searching forward in the state space of the planning task. In the context of proving unsolvability, heuristics are useful for proving individual states unsolvable.

An unsolvability certificate for a heuristic search algorithm must establish that none of the states expanded by such a search algorithm are goal states and that all their successors have been expanded or proven unsolvable by a heuristic. In this case, a 1-disjunctive certificate for the overall algorithm can be efficiently computed if we can compute inductive certificates for each state shown unsolvable by a heuristic.

Theorem 8. *Let $\Pi = \langle V, A, I, G\rangle$ be a STRIPS planning task, let $S_\infty \subseteq S(\Pi)$ be a set of unsolvable states, and let $\mathcal{F}_\infty$ be a family of inductive certificates such that $S_\infty \subseteq \bigcup_{S' \in \mathcal{F}_\infty} S'$. (In other words, each state in S_∞ is included in some inductive certificate in $\mathcal{F}_\infty$.)*

Moreover, let $S_{\text{exp}} \subseteq S(\Pi)$ be a set of states such that $I \in S_{\text{exp}} \cup S_\infty$, S_{exp} contains no goal state, and $S_{\text{exp}}[A] \subseteq S_{\text{exp}} \cup S_\infty$. Then $\mathcal{F} = \{\{s\} \mid s \in S_{\text{exp}}\} \cup \mathcal{F}_\infty$ is a 1-disjunctive certificate for Π.

Proof. We check the three requirements for 1-disjunctive certificates in Def. 4. Property 1 (inclusion of the initial state) is trivial. Property 2 (non-inclusion of goal states) holds because S_{exp} contains no goal states and all $S' \in \mathcal{F}_\infty$ are inductive certificates and therefore do not contain any goal states either. It remains to check property 3 (disjunctive 1-inductivity) for all component certificates $S' \in \mathcal{F}$ and all operators $o \in A$.

Case 1: $S' = \{s\}$ for some $s \in S_{\text{exp}}$. Then $S'[o]$ is either empty (if o is inapplicable in s) and the condition holds trivially, or $S'[o] = \{s[o]\}$, and we can set $\mathcal{F}'$ in Def. 4 to include any set in $\mathcal{F}$ that includes $s[o]$. Such a set must exist because $S_{\text{exp}}[A] \subseteq S_{\text{exp}} \cup S_\infty$ and $S_\infty \subseteq \bigcup_{S' \in \mathcal{F}_\infty} S'$.

Case 2: $S' \in \mathcal{F}_\infty$. Then we can set $\mathcal{F}' = \{S'\}$ in Def. 4 because S' is an inductive certificate. □

To generate a 1-disjunctive $\mathbf{R}$-certificate for a heuristic forward search algorithm, we apply the theorem with S_{exp} as the set of expanded states and S_∞ as the set of states pruned due to infinite heuristic estimates. We need inductive $\mathbf{R}$-certificates for the states in S_∞ as components. To generate the component certificates for singleton state sets, it suffices that $\mathbf{R}$ efficiently support $\mathbf{CL}$, which all representation formalisms we consider do.

Trapper

The *Trapper* algorithm (Lipovetzky, Muise, and Geffner 2016) computes a formula φ_{trap} such that all states satisfying φ_{trap} only have states satisfying φ_{trap} as successors and all goal states satisfying φ_{trap} violate a mutex, where mutexes are computed with the h^2 algorithm. Lipovetzky et al. (2016) show that all states satisfying φ_{trap} that can be reached by a forward search algorithm are unsolvable.

By itself, *states*(φ_{trap}) is an inductive set but not an inductive certificate because it may include goal states. Hence, it is not possible to use *states*(φ_{trap}) as an unsolvability certificate on its own. This makes sense because the soundness of the algorithm relies on the soundness of the mutexes used, and hence a certifying algorithm must certify the soundness of the mutexes as well.

For a planning task $\Pi = \langle V, A, I, G\rangle$, let $M \subseteq 2^V$ be the set of unordered pairs determined as mutually exclusive by the h^2 algorithm, i.e., $\{v_1, v_2\} \in M$ iff $h^2(\{v_1, v_2\}) = \infty$. Then the *consistent* states of Π given M (the states not violating a mutex in M) can be described by the formula $\varphi_{\text{cons}} = \bigwedge_{\{v_1,v_2\} \in M} (\neg v_1 \vee \neg v_2)$. It is easy to see that φ_{cons} describes an inductive set, i.e., all successors of consistent states are consistent.

Let $\varphi_{\text{prune}} = \varphi_{\text{trap}} \wedge \varphi_{\text{cons}}$. Then *states*$(\varphi_{\text{prune}})$ contains all states pruned by the Trapper algorithm and is an inductive certificate for all its states. S is closed under operator application because φ_{trap} and φ_{cons} are individually closed under operator application, and S does not contain a goal state because all goal states satisfying φ_{trap} must violate a mutex and hence do not satisfy φ_{cons}.

It remains to show that S can be compactly described using a tractable representation formalism $\mathbf{R}$. The trap formula φ_{trap} is a disjunction of conjunctions of atoms, i.e., $\varphi_{\text{trap}} = \bigvee_{1 \le i \le n} \bigwedge_{v \in c_i} v$ for certain variable subsets (conjunctions) $c_1, \dots, c_n \subseteq V$. Hence $\varphi_{\text{prune}} = \varphi_{\text{trap}} \wedge \varphi_{\text{cons}} = (\bigvee_{1 \le i \le n} \bigwedge_{v \in c_i} v) \wedge \varphi_{\text{cons}}$, which is equivalent to $\bigvee_{1 \le i \le n}((\bigwedge_{v \in c_i} v) \wedge \varphi_{\text{cons}})$, which we can write as $\bigvee_{1 \le i \le n} \psi_i$ with $\psi_i = (\bigwedge_{v \in c_i} v) \wedge \varphi_{\text{cons}}$. Each ψ_i is a CNF formula consisting of unit clauses from c_i and negative binary clauses from φ_{cons}, and hence ψ_i is a Horn formula and a 2-CNF formula. Therefore, $\Phi = \{\psi_1, \dots, \psi_n\}$ is a disjunctive Horn certificate and a disjunctive 2-CNF certificate

[5]For space reasons, we do not prove the 1-conjunctiveness result, which would require elaborating the details of the Π^m compilation. The key idea is that for every original operator o and every conjunction c represented in Π^m that o might achieve, there exists a unique minimal (w.r.t. its preconditions) operator $\tilde{o}$ based on o achieving c in Π^m. If c is unreachable in Π^m, one of the preconditions of $\tilde{o}$ must be unreachable, and such a precondition can then be used to define the family $\mathcal{F}'$ of size at most 1 needed in property 3 of Definition 5.

for all states pruned by Trapper.[6]

Arbitrary disjunctive Horn or 2-CNF certificates are not efficiently verifiable (unless P = NP), but it turns out that Φ is actually 1-disjunctive. To see this, we must verify that for every component certificate $\psi_i \in \Phi$ and every operator $o \in A$, there exists a component certificate $\psi_j \in \Phi$ such that applying operator o in a state satisfying ψ_i is guaranteed to lead to a state satisfying ψ_j.

A closer investigation of the Trapper algorithm shows that this is indeed the case. Trapper is based on *marking* certain conjunctions of variables, and the conjunctions $c_1, \dots, c_n$ defining φ_{trap} are the ones that remain *unmarked* after executing the algorithm. Unmarked conjunctions c_i have the property that each operator $o \in A$ is either inapplicable in all states satisfying ψ_i (in which case disjunctive 1-inductivity holds trivially), or there exists an unmarked *o-child* c_j of c_i, i.e., a conjunction satisfying $c_j \subseteq ((c_i \cup \mathit{pre}(o)) \setminus \mathit{del}(o)) \cup \mathit{add}(o)$. In this case, every o-successor of a state satisfying c_i satisfies c_j, and hence every o-successor of a state satisfying ψ_i satisfies ψ_j.

Proof-of-Concept Implementation

To experimentally test inductive unsolvability certificates, we augmented the implementations of A*, $h^{\max}$ and $h^{\text{M\&S}}$ (linear merge strategies, no pruning of unreachable abstract states) in Fast Downward (Helmert 2006) to produce BDD-based unsolvability certificates. We refer to the original Fast Downward as FD and the certifying version as FD$^{\text{cert}}$.

We also implemented a certificate verification algorithm supporting regular, r-conjunctive and r-disjunctive BDD certificates. The verifier is a stand-alone implementation (in C++) not related to an existing planning system. Both FD$^{\text{cert}}$ and the verifier use the CUDD library[7] to implement BDD operations and are publicly available.[8]

We tested FD$^{\text{cert}}$ and the verifier with these two heuristics (using default settings for $h^{\text{M\&S}}$) on the unsolvable tasks from the Unsolvability IPC 2016, on the unsolvable resource-constrained benchmarks used by Steinmetz and Hoffmann (2016) and on the unsolvable benchmarks by Hoffmann, Kissmann, and Torralba (2014).[9] We set limits of 30 minutes and 2 GiB for FD and FD$^{\text{cert}}$ and a more generous limit of 4 hours and 2 GiB for the verifier. A data set with detailed results is publicly available.[10]

Table 1 shows how many tasks in each domain could be solved with and without certificate generation and how many of the generated certificates could be verified within the resource bounds. For A* with $h^{\max}$, certificates could be produced for 64% of the successful runs, and in 90% of the

	$h^{\max}$			$h^{\text{M\&S}}$		
	FD	FD$^{\text{cert}}$	Ver.	FD	FD$^{\text{cert}}$	Ver.
3unsat (30)	15	10	10	15	10	10
bag-barman (20)	8	8	3	12	8	4
bag-gripper (25)	3	3	1	3	3	1
bag-transport (29)	6	6	4	7	6	4
bottleneck (25)	20	15	15	10	8	7
cave-diving (25)	7	5	5	6	6	5
chessboard-pebbling (23)	5	3	3	5	5	4
diagnosis (18)	5	4	4	3	1	1
document-transfer (20)	7	5	5	12	9	4
mystery (9)	2	0	0	2	2	1
nomystery (150+24)	54	20	16	46	38	27
over-rovers (150+20)	14	8	8	24	22	19
over-tpp (25+30)	22	10	10	34	29	25
pegsol (24)	24	20	20	24	24	24
pegsol-row5 (15)	5	4	4	5	5	4
sliding-tiles (20)	10	10	10	10	10	10
tetris (20)	5	5	5	5	5	5
total (702)	212	136	123	223	191	155

Table 1: Completed tasks by domain (FD, FD$^{\text{cert}}$, verifier).

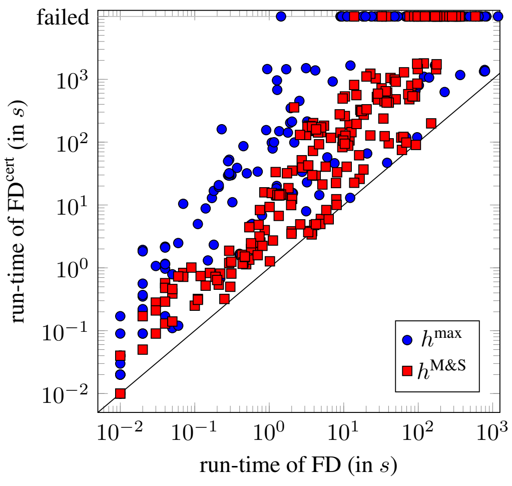

Figure 1: Run-time comparison of FD vs. FD$^{\text{cert}}$.

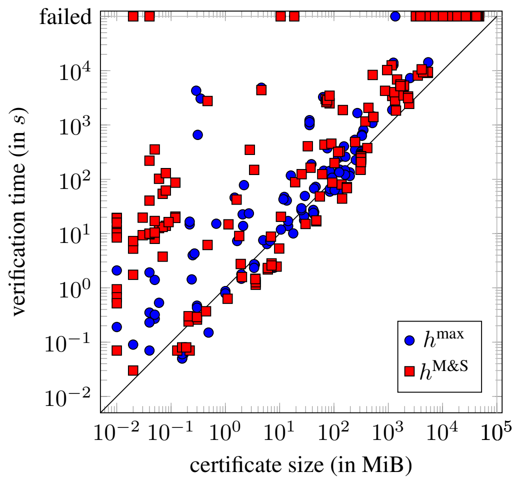

Figure 2: Verifier run-time as a function of certificate size.

[6]Unlike the previous constructions, constructing this certificate takes quadratic rather than linear time because the mutex information is replicated in each component formula ψ_i. A linear representation is possible if formulas are represented as circuits, i.e., if common subformulas may be reused.

[7]http://vlsi.colorado.edu/~fabio/

[8]http://doi.org/10.5281/zenodo.376651

[9]For the latter two benchmark collections, see https://fai.cs.uni-saarland.de/software.html.

[10]http://doi.org/10.5281/zenodo.376652

cases where a certificate could be generated, it could be verified within the given resource bounds. For A* with $h^{\text{M\&S}}$, the corresponding numbers are 86% and 81%. We consider these numbers encouraging given that (to the best of our knowledge) this is the first attempt at implementing a certifying planning algorithm, and neither the certifying algorithm nor the verifier are highly optimized at this stage.

For $h^{\text{M\&S}}$, 30.5% of the cases where certificate verification failed were due to the memory limit and 69.5% of the cases were due to the time limit. For $h^{\max}$, all certificate verification failures were due to the time limit. However, as CUDD seems to slow down to preserve memory when nearing memory exhaustion, the distinction can be fuzzy.

Figures 1 and 2 provide a more detailed look at this data. Figure 1 shows that in most cases, but not always, the overhead of FD$^{\text{cert}}$ compared to FD is benign. Figure 2 shows the time needed to verify a certificate as a function of its size (which is itself at most linear in the run-time of FD$^{\text{cert}}$). Apart from very small certificates, where verification time is dominated by processing the task representation, the scaling behavior appears to be not much worse than linear.

Conclusion

We described unsolvability certificates for classical planning based on inductive sets. By using suitable representations together with disjunctive and conjunctive factoring, these certificates are sufficiently general to cover many planning algorithms from the literature. A proof-of-concept implementation shows that the approach is practically feasible.

In future work, we would like to extend the approach to further planning techniques. For example, pruning techniques based on partial-order reduction are good candidates for certification because several flawed algorithms have been proposed in the literature (cf. Wehrle and Helmert 2012).

Acknowledgments

This work was supported by the European Research Council as part of the project “State Space Exploration: Principles, Algorithms and Applications”.

References

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

Bäckström, C.; Jonsson, P.; and Ståhlberg, S. 2013. Fast detection of unsolvable planning instances using local consistency. In Helmert, M., and Röger, G., eds., *Proceedings of the Sixth Annual Symposium on Combinatorial Search (SoCS 2013)*, 29–37. AAAI Press.

Bahar, R. I.; Frohm, E. A.; Gaona, C. M.; Hachtel, G. D.; Macii, E.; Pardo, A.; and Somenzi, F. 1993. Algebraic decision diagrams and their applications. In Lightner, M. R., and Jess, J. A. G., eds., *Proceedings of the 1993 IEEE/ACM International Conference on Computer-Aided Design (ICCAD 1993)*, 188–191.

Beame, P.; Kautz, H. A.; and Sabharwal, A. 2004. Towards understanding and harnessing the potential of clause learning. *Journal of Artificial Intelligence Research* 22:319–351.

Bonet, B., and Castillo, J. 2011. A complete algorithm for generating landmarks. In Bacchus, F.; Domshlak, C.; Edelkamp, S.; and Helmert, M., eds., *Proceedings of the Twenty-First International Conference on Automated Planning and Scheduling (ICAPS 2011)*, 315–318. AAAI Press.

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

Bonet, B., and Helmert, M. 2010. Strengthening landmark heuristics via hitting sets. In Coelho, H.; Studer, R.; and Wooldridge, M., eds., *Proceedings of the 19th European Conference on Artificial Intelligence (ECAI 2010)*, 329–334. IOS Press.

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

Darwiche, A., and Marquis, P. 2002. A knowledge compilation map. *Journal of Artificial Intelligence Research* 17:229–264.

Dowling, W. F., and Gallier, J. H. 1984. Linear-time algorithms for testing the satisfiability of propositional Horn formulae. *Journal of Logic Programming* 1(3):367–383.

Edelkamp, S., and Helmert, M. 2001. The model checking integrated planning system (MIPS). *AI Magazine* 22(3):67–71.

Edelkamp, S., and Kissmann, P. 2011. On the complexity of BDDs for state space search: A case study in Connect Four. In Burgard, W., and Roth, D., eds., *Proceedings of the Twenty-Fifth AAAI Conference on Artificial Intelligence (AAAI 2011)*, 18–23. AAAI Press.

Edelkamp, S. 2001. Planning with pattern databases. In Cesta, A., and Borrajo, D., eds., *Proceedings of the Sixth European Conference on Planning (ECP 2001)*, 84–90. AAAI Press.

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.

Haslum, P., and Geffner, H. 2000. Admissible heuristics for optimal planning. In Chien, S.; Kambhampati, S.; and Knoblock, C. A., eds., *Proceedings of the Fifth International Conference on Artificial Intelligence Planning and Scheduling (AIPS 2000)*, 140–149. AAAI Press.

Haslum, P. 2009. $h^m(P) = h^1(P^m)$: Alternative characterisations of the generalisation from $h^{\max}$ to h^m. In Gerevini, A.; Howe, A.; Cesta, A.; and Refanidis, I., eds., *Proceedings of the Nineteenth International Conference on Automated Planning and Scheduling (ICAPS 2009)*, 354–357. AAAI Press.

Helmert, M., and Domshlak, C. 2009. Landmarks, critical paths and abstractions: What's the difference anyway? In Gerevini, A.; Howe, A.; Cesta, A.; and Refanidis, I., eds., *Proceedings of the Nineteenth International Conference on Automated Planning and Scheduling (ICAPS 2009)*, 162–169. AAAI Press.

Helmert, M.; Haslum, P.; Hoffmann, J.; and Nissim, R. 2014. Merge-and-shrink abstraction: A method for generating lower bounds in factored state spaces. *Journal of the ACM* 61(3):16:1–63.

Helmert, M.; Haslum, P.; and Hoffmann, J. 2007. Flexible abstraction heuristics for optimal sequential planning. In Boddy, M.; Fox, M.; and Thiébaux, S., eds., *Proceedings of the Seventeenth International Conference on Automated Planning and Scheduling (ICAPS 2007)*, 176–183. AAAI Press.

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

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

Hoffmann, J.; Kissmann, P.; and Torralba, Á. 2014. "Distance"? Who cares? Tailoring merge-and-shrink heuristics to detect unsolvability. In Schaub, T.; Friedrich, G.; and O'Sullivan, B., eds., *Proceedings of the 21st European Conference on Artificial Intelligence (ECAI 2014)*, 441–446. IOS Press.

Howey, R., and Long, D. 2003. VAL's progress: The automatic validation tool for PDDL2.1 used in the International Planning Competition. In Edelkamp, S., and Hoffmann, J., eds., *Proceedings of the ICAPS 2003 Workshop on the Competition: Impact, Organisation, Evaluation, Benchmarks*.

Kautz, H., and Selman, B. 1992. Planning as satisfiability. In Neumann, B., ed., *Proceedings of the 10th European Conference on Artificial Intelligence (ECAI 1992)*, 359–363. John Wiley and Sons.

Keyder, E.; Hoffmann, J.; and Haslum, P. 2014. Improving delete relaxation heuristics through explicitly represented conjunctions. *Journal of Artificial Intelligence Research* 50:487–533.

Keyder, E.; Richter, S.; and Helmert, M. 2010. Sound and complete landmarks for and/or graphs. In Coelho, H.; Studer, R.; and Wooldridge, M., eds., *Proceedings of the 19th European Conference on Artificial Intelligence (ECAI 2010)*, 335–340. IOS Press.

Lipovetzky, N.; Muise, C.; and Geffner, H. 2016. Traps, invariants, and dead-ends. In *Proceedings of the Twenty-Sixth International Conference on Automated Planning and Scheduling (ICAPS 2016)*, 211–215. AAAI Press.

McConnell, R. M.; Mehlhorn, K.; Näher, S.; and Schweitzer, P. 2011. Certifying algorithms. *Computer Science Review* 5(2):119–162.

Schaefer, T. J. 1978. The complexity of satisfiability problems. In *Proceedings of the Tenth Annual ACM Symposium on Theory of Computing (STOC '78)*, 216–226. New York: ACM Press.

Steinmetz, M., and Hoffmann, J. 2016. Towards clause-learning state space search: Learning to recognize dead-ends. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence (AAAI 2016)*, 760–768. AAAI Press.

Torralba, Á. 2015. *Symbolic Search and Abstraction Heuristics for Cost-Optimal Planning*. Ph.D. Dissertation, Universidad Carlos III de Madrid.

Wehrle, M., and Helmert, M. 2012. About partial order reduction in planning and computer aided verification. In McCluskey, L.; Williams, B.; Silva, J. R.; and Bonet, B., eds., *Proceedings of the Twenty-Second International Conference on Automated Planning and Scheduling (ICAPS 2012)*, 297–305. AAAI Press.