Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence (IJCAI-18)

Inductive Certificates of Unsolvability for Domain-Independent Planning†

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

Abstract

If a planning system outputs a solution for a given problem, it is simple to verify that the solution is valid. However, if a planner claims that a task is unsolvable, we currently have no choice but to trust the planner blindly. We propose a sound and complete class of *certificates of unsolvability*, which can be verified efficiently by an independent program. To highlight their practical use, we show how these certificates can be generated for a wide range of state-of-the-art planning techniques with only polynomial overhead for the planner.

1 Introduction

The aim of automated planning is to find a course of actions whose application to the current situation of the world leads to a desired goal situation. Two examples are a logistics company which has packages and trucks at several distribution centers and wants to deliver all packages to their destination, or a Sokoban game where we need to push boxes to their goal locations. We focus on *classical* planning, where actions are *deterministic* and we have *complete knowledge* of the world.

Traditionally, most research in classical planning concentrated on finding *plans* (action sequences that reach the goal). Depending on the problem it might however be beneficial to instead focus on the question whether the problem is actually solvable or not. The International Planning Competition (IPC) recently encouraged further research in this direction with its first *Unsolvability IPC* in 2016, and indeed, many new techniques for detecting unsolvability emerged. [Hoffmann *et al.*, 2014; Haslum, 2016; Gnad *et al.*, 2016; Pommerening and Seipp, 2016; Seipp *et al.*, 2016; Torralba, 2016; Torralba *et al.*, 2016; Steinmetz and Hoffmann, 2017].

A question that naturally arises is whether the planners are actually correct in their assessment of which tasks are unsolvable. If a planner claims that a task is solvable, it usually provides an actual plan. This plan can easily be verified by applying the actions from the initial state and checking whether this leads to a goal state. For this purpose there already exist tools such as VAL [Howey and Long, 2003] or INVAL [Haslum, 2011].

† This is an abridged version of a paper [Eriksson *et al.*, 2017b] that won the best student paper award at ICAPS 2017.

Proving unsolvability however is not as simple. A planning task induces a directed graph, the so-called *state space*, which is exponentially larger than the task description. To prove unsolvability we need to show that this graph contains no path from the initial state to any goal state.

We propose a class of *certificates of unsolvability*, which is *complete* (in the sense that for each unsolvable planning task such a certificate exists) and can be *efficiently verified*. The core idea is based on sets of states that once entered cannot be left again. If the set contains the initial state but no goal state, the task must be unsolvable. We can efficiently generate such certificates for a wide variety of current planning techniques.

2 Planning Tasks

We consider planning tasks in propositional STRIPS representation [Fikes and Nilsson, 1971]. We do not consider action costs, which are irrelevant for the question of solvability.

A *planning task* is a tuple $\Pi = \langle V, A, I, G \rangle$, where V is a finite set of (state) *variables*, A is a finite set of *actions*, $I \subseteq V$ is the *initial state* and $G \subseteq V$ is the *goal description*.

A state $s \subseteq V$ of Π is defined by the variables that are *true* in the state. State s is a *goal state* if $G \subseteq s$. We write $S(\Pi) = 2^V$ for the set of all states of Π and $S_G(\Pi)$ for the set of goal states of Π. Each action $a \in A$ is defined by its *preconditions* $pre(a) \subseteq V$, *add effects* $add(a) \subseteq V$ and *delete effects* $del(a) \subseteq V$. It is *applicable* in state s if $pre(a) \subseteq s$, and the resulting *successor state* is $s[a] = (s \setminus del(a)) \cup add(a)$.

A *plan for* Π is a sequence of actions that are successively applicable in I and result in a goal state. Task Π is *solvable* if there exists a plan, otherwise it is *unsolvable*.

Given a state set $S \subseteq S(\Pi)$ and action a, we write $S[a] = \{s[a] \mid s \in S, a \text{ applicable in } s\}$ for the set of all successors of S via action a, and for action sets $A' \subseteq A$ we write $S[A'] = \bigcup_{a \in A'} S[a]$ for all successors of S via any action in A'.

3 Representing State Sets

Since our certificates work with potentially very large sets of states, we require a suitable and compact representation of these sets. We will use propositional formulas over the variables V from the planning task. Such a formula represents a set of states according to the following definition:

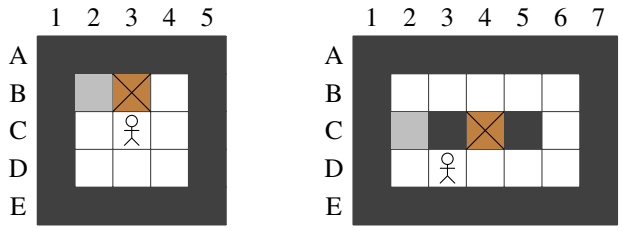

Figure 1: Two instances of the Sokoban puzzle. Dark gray fields are walls, the brown field represents the box and the gray field its goal position.

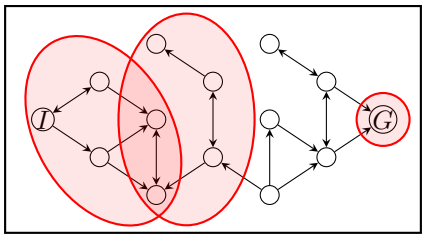

Figure 2: Examples of inductive sets in a search space. Notice that their unions and intersections are inductive as well.

Definition 1 (state set representation)**.** *Given a model $\mathcal{M}$ of a propositional formula φ, state $s_\mathcal{M}$ contains exactly the variables which are true in $\mathcal{M}$, i. e. $s_\mathcal{M} = \{v \mid \mathcal{M}(v) = \top\}$.*

Formula φ represents the set S of all states that are described by a model of φ, i. e. $S = \{s_\mathcal{M} \mid \mathcal{M} \models \varphi\}$.

While propositional formulas can in most cases compactly represent large state sets, verifying certain properties such as satisfiability is in general not efficiently possible. We thus focus on three subclasses which trade off representational power in favor of efficient operations:

- Reduced Ordered Binary Decision Diagrams (BDDs) [Bryant, 1986] represent formulas as rooted acyclic graphs.
- 2-CNF formulas are formulas in conjunctive normal form (CNF) with at most 2 literals per clause.
- Horn formulas are CNF-formulas where each clause contains at most one positive literal.

We chose BDDs since they are commonly used in planning to represent sets of states, and Horn and 2-CNF formulas since they are two of the maximal tractable classes for boolean constraint satisfaction [Schaefer, 1976].

4 Inductive Certificates

The basic idea of our certificates revolves around defining a set of states which once entered cannot be left again. Consider the left side of Figure 1, which shows a Sokoban problem where the box is facing an outer wall. While we can still move the box around (and thus reach different states), we can never reach a state where the box is not on that wall. In other words, if an action is applied to any state in this set S of states where the box is on the upper wall, the resulting state will also be contained in S. We call a set with this property an *inductive set*.

Definition 2 (inductive set)**.** *A set S of states from a planning task $\Pi = \langle V, A, I, G\rangle$ is* inductive *in Π iff $S[A] \subseteq S$.*

The empty set and the set of all states are two trivial cases of inductive sets. Another example is the set of states reachable from one specific state. Furthermore, inductive sets are closed under union and intersection. Some concrete examples can be seen in Figure 2. In the context of unsolvability, inductive sets described by a formula are also called *traps* [Lipovetzky *et al.*, 2016]

If we find an inductive set S with the additional properties that (i) it contains the initial state and (ii) it does not contain any goal state, we have a sufficient argument that the planning task is unsolvable: Since a potential solution must start in S (due to (i)) and never leaves S (because S is inductive), it must also end in S. But then it cannot reach a goal (due to (ii)) and therefore cannot be a plan. We call this form of unsolvability certificate an *inductive certificate*:

Definition 3 (inductive certificate)**.** *A set $S \subseteq S(\Pi)$ is an inductive certificate for $\Pi = \langle V, A, I, G\rangle$ if*

1. *$I \in S$ (S contains the initial state)*
2. *$S \cap S_G(\Pi) = \emptyset$ (S contains no goal state), and*
3. *$S[A] \subseteq S$ (S is inductive)*

As we argued above, if such an inductive certificate exists, the task is unsolvable. It is also easy to see that for each unsolvable planning task there exists an inductive certificate, for example the set of all states reachable from the initial state.

Corollary 1 (soundness and completeness)**.** *Inductive certificates are* sound *and* complete*, i. e. a planning task Π is unsolvable iff an inductive certificate for Π exists.*

In order for such a certificate to be verifiable in practice, we must be able to check its three properties efficiently. Property 1 ($I \in S$) corresponds to a model check, which BDDs, Horn and 2-CNF formulas all support efficiently.

Property 2 ($S \cap S_G(\Pi) = \emptyset$)) can be reformulated with logical formulas φ_S and $\bigwedge_{v \in G} v$ (representing S and $S_G(\Pi)$) as $\varphi_S \models \bigvee_{v \in G} \neg v$. This implication is a special form of entailment called *clausal entailment*, which all considered formalisms support efficiently.

Property 3 ($S[A] \subseteq S$) requires the most involved verification step. We first observe that it is sufficient to test for each $a \in A$ individually that $S[a] \subseteq S$ holds. To this end we introduce a primed version v' for each variable $v \in V$ and then define a transition formula τ_a which describes the state in which a is applied in terms of the variables from V and the successor state in terms of the primed variables: $\tau_a = \bigwedge_{v \in pre(a)} v \wedge \bigwedge_{v \in add(a)} v' \wedge \bigwedge_{v \in del(a) \setminus add(a)} \neg v' \wedge \bigwedge_{v \in V \setminus (add(a) \cup del(a))} (v \leftrightarrow v')$. If we conjoin τ_a with φ_S (which represents S), each model of the resulting formula represents a pair of states (s, s') (where s' is gained from the primed variables) with the property that $s \in S$ and s' is the successor of s with a, i. e. $s[a] = s'$. What is left to do is to check that all these successors s' are in S. We can do this by creating a formula $\varphi_{S'}$ which replaces all v with v', and then

testing if $\varphi_S \wedge \tau_a \models \varphi_{S'}$ holds. All these steps can be done efficiently for all considered formalisms.

Corollary 2 (efficient verification)**.** *Given a BDD, Horn formula or 2-CNF formula φ_S representing a state set S, testing whether S is an inductive certificate can be done in time polynomial in the representation size of φ_S and Π.*

4.1 Composite Certificates

While inductive certificates are complete, it is not always possible to describe S compactly with the considered formalisms. Thus we introduce two composite certificate types where we consider a family $\mathcal{F}$ of sets such that their union or intersection forms an inductive certificate and each set $S \in \mathcal{F}$ is represented by its own formula φ_S.

Definition 4 (disjunctive/conjunctive certificate)**.** *A family $\mathcal{F}$ of state sets is an* disjunctive certificate *if $\bigcup_{S\in\mathcal{F}} S$ is an inductive certificate for Π. It is an* conjunctive certificate *for Π if $\bigcap_{S\in\mathcal{F}} S$ is an inductive certificate for Π.*

Verifying the conditions for inductive certificates still requires considering the full union/intersection. But if a representation efficiently support this operation, there is no need to use a composite certificate. For instance, the conjunction of several Horn formulas is a Horn formula that can be build in linear time, so we could efficiently transform a conjunctive Horn certificate into a standard inductive certificate. The same is true for conjunctive 2-CNF certificates.

With a BDD representation or for disjunctive certificates, an efficient verification of the composite certificate is not possible in general (unless P = NP). However, for many applications we can use stronger properties that imply the properties of inductive certificates but only require to consider a *bounded* number of sets at once. With these stronger properties, we can also use formalisms that only efficiently support *bounded* disjunction or conjunction (such as BDDs).

Definition 5 (r-disjunctive certificate)**.** *A family $\mathcal{F}$ of state sets is an r-*disjunctive certificate *if*

1. *$I \in S$ 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 $a \in A$, there is a subfamily $\mathcal{F}' \subseteq \mathcal{F}$ with $|\mathcal{F}'| \leq r$ and $S[a] \subseteq \bigcup_{S'\in\mathcal{F}'} S'$.*

For disjunctive certificates in general, the inclusion of the initial state and the exclusion of all goal states can be checked for each set $S \in \mathcal{F}$ independently. For verifying the inductivity of $\bigcup_{S\in\mathcal{F}} S$, it is sufficient to verify for each set $S \in \mathcal{F}$ and action $a \in A$ that all successors lie in $\bigcup_{S\in\mathcal{F}} S$. For the more restrictive r-disjunctive certificates, it is sufficient to verify that they are in a union of at most r sets from $\mathcal{F}$.

In terms of the three representation formalisms, only BDDs can be used for verifying general r-disjunctive certificates. The problem with Horn and 2-CNF formulas is that they do not support the composition to a disjunction. However in the special case of 1-disjunctive certificates, they can be used since no disjunctions are required. This is for example already sufficient for tasks that are detected unsolvable by the Trapper algorithm [Lipovetzky *et al.*, 2016].

Conjunctive certificates are based on similar ideas:

Definition 6 (r-conjunctive certificate)**.** *A family $\mathcal{F}$ of state sets is an r-*conjunctive certificate *if*

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

BDDs, Horn and 2-CNF formulas are all suitable for r-conjunctive certificates, although in the case of Horn and 2-CNF formulas we could directly use inductive certificates.

5 Generating Certificates

So far, we have concentrated on the efficient *verification* of certificates, now we give an idea how they can be efficiently *generated* by current planning techniques. In this summary paper, we do not have sufficient space to go into detail and to cover many different approaches, so for an in-depth discussion, we refer the interested reader to the original paper [Eriksson *et al.*, 2017b].

Many state-of-the-art planners employ a forward-search in the induced state space, usually using a heuristic for guiding and pruning the search. We will illustrate from a high-level perspective how such a system could build a certificate.

5.1 Blind Search

We first focus on the simplest case of a so-called *blind search*, which does not prune the search but must consider all states that are reachable from the initial state.

Consider the right puzzle in Figure 1: A blind search will try to push the box to all positions it can, namely B2-B6, C4 (its start position) and D2-D6, and for each position, it considers all possible positions of the person (everywhere except at the box position). All these states are considered individually and the search can successively build up a BDD for the entire set of states. The final BDD represents an inductive certificate, because it contains the initial state, cannot be left and for an unsolvable task it does not contain a goal state.

5.2 Heuristic Search

If a heuristic is used for pruning, the search does no longer explicitly consider every reachable state, so the set of considered states alone is no longer inductive.

The search prunes state s (and all successor nodes), if the heuristic detects that s is a *dead end*, i. e., the goal is unreachable from s. In the full paper [Eriksson *et al.*, 2017b], we have shown for a wide range of heuristics how they can efficiently provide an inductive set of states that contains the dead end but no goal state for each detected dead end. These can be seen as inductive certificates for the dead ends.

From these we can build a 1-disjunctive certificate that includes the inductive sets D_i for each dead end, and for each state s' that is actually expanded by the search a set S_i containing only s'. The certificate is 1-disjunctive since for all D_i we have $D_i[A] \subseteq D_i$ and for all S_i and applicable $a \in A$ we have a single successor which must either also be expanded (and thus $S_i[a] \subseteq S_j$ for some j) or is a dead end (and thus $S_i[a] \subseteq D_j$ for some j).

As an example, consider again the right puzzle in Figure 1 and assume that a heuristic detects all states where the box is at 4B or 4D as dead ends (independent of the position of the person). Further, assume that it provides inductive sets box-at-B and box-at-D (e.g. in 2-CNF), respectively, which do not cover any goal state. As these states are pruned, the search would only explore all states where the box is at position 4C (just moving the person around). It would collect all these states as individual sets (e.g. represented by formulas like box-at-4C $\wedge$ person-at-D4). The set of all these formulas and those from the heuristic is a 1-disjunctive certificate. The initial state has been covered by the search and no formula covers a goal state. For inductivity it holds that box-at-B and box-at-D (from the heuristic) are inductive and all sets/formulas included by the search represent a single state where each applicable operator either results in another such state or in a state covered by box-at-B or by box-at-D.

6 Experiments

As a proof of concept for the practicability of our certificates we extended the Fast Downward planning system [Helmert, 2006] to generate certificates with A* search [Hart *et al.*, 1968] using the $h^{\max}$ heuristic [Bonet and Geffner, 2001], and with A* using the Merge and Shrink (M&S) heuristic [Helmert *et al.*, 2007; Helmert *et al.*, 2014]. Additionally we implemented a standalone verifier that can validate the generated certificates. Both implementations use the CUDD library [Somenzi, 2015] for representing BDDs, and are publicly available [Eriksson *et al.*, 2017a].

In order to asses the overhead of generating certificates we ran both Fast Downward configurations without and with certificate generation (which we will call FD and FD$^{\text{cert}}$). We imposed limits of 30 minutes and 2 GiB for FD and FD$^{\text{cert}}$; for the verifier we set the same memory limit but a more generous time limit of 4 hours. As benchmarks we used a collection of unsolvable planning tasks which includes the unsolvable tasks from the Unsolvability IPC.

For A* with $h^{\max}$, FD detected 212 tasks as unsolvable, and FD$^{\text{cert}}$ was able to generate certificates for 136 of them. Of these 136 certificates, the verifier could verify 123 certificates within the given limits. For A* with $h^{\text{M\&S}}$, the respective numbers are 223 for FD, 191 for FD$^{\text{cert}}$ and 155 for the verifier. As expected, all verified certificates were valid.

In order to get a better picture of the time overhead imposed on the planner, Figure 3 compares the runtime of FD and FD$^{\text{cert}}$, showing that the overhead, while noticeable, is still benign. Figure 4 plots the time required to verify a certificate against its size. Except for small certificates (where verification is dominated by parsing the task) we see that certificate size and verification time mostly correlate.

Overall the results show that our certificates are useful in practice, especially when considering that we used a proof of concept implementation that could be further optimized.

7 Conclusion

We introduced a way to formally prove unsolvability of classical planning tasks, based on inductive certificates. The presented certificates are complete, can be verified efficiently and can also be generated efficiently for a wide range of planning techniques. As a practical contribution, we implemented the generation of certificates for two configurations of a state-of-the-art planner as well as a standalone verifier for the generated certificates. Experiments show that certificates can be generated and verified with reasonable resources.

One restriction for composite certificates is that all sets must be represented in the same formalism. This limits their applicability because we can for example not use them if the search exploits multiple heuristics that require different formalisms. We addressed this issue in an extension of this line of work towards a flexible proof system [Eriksson *et al.*, 2018].

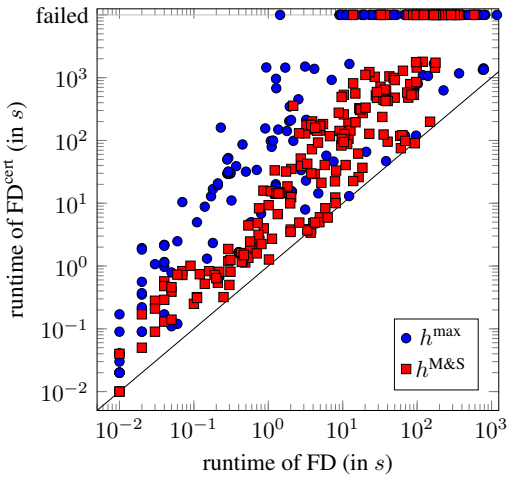

Figure 3: Runtime comparison of FD vs. FD$^{\text{cert}}$.

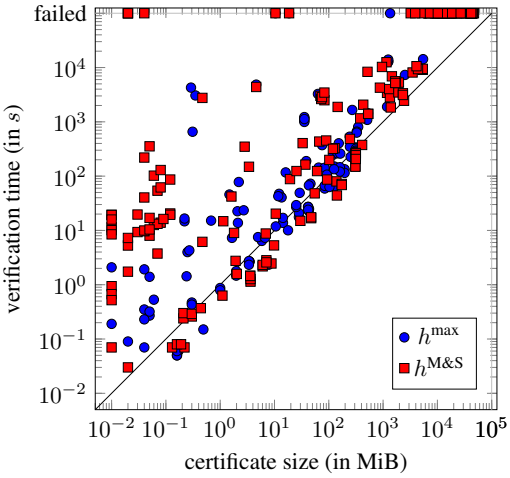

Figure 4: Verifier runtime as a function of certificate size.

Acknowledgments

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

References

[Bonet and Geffner, 2001] Blai Bonet and Héctor Geffner. Planning as heuristic search. *Artificial Intelligence*, 129(1):5–33, 2001.

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

[Eriksson *et al.*, 2017a] Salomé Eriksson, Gabriele Röger, and Malte Helmert. Certifying Fast Downward. https://doi.org/10.5281/zenodo.376651, 2017.

[Eriksson *et al.*, 2017b] Salomé Eriksson, Gabriele Röger, and Malte Helmert. Unsolvability certificates for classical planning. In *Proceedings of the Twenty-Seventh International Conference on Automated Planning and Scheduling (ICAPS 2017)*, pages 88–97. AAAI Press, 2017.

[Eriksson *et al.*, 2018] Salomé Eriksson, Gabriele Röger, and Malte Helmert. A proof system for unsolvable planning tasks. In *Proceedings of the Twenty-Eighth International Conference on Automated Planning and Scheduling (ICAPS 2018)*. AAAI Press, 2018. In Press.

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

[Gnad *et al.*, 2016] Daniel Gnad, Marcel Steinmetz, and Jörg Hoffmann. Django: Unchaining the power of red-black planning. In *Unsolvability International Planning Competition: planner abstracts*, pages 19–22, 2016.

[Hart *et al.*, 1968] Peter E. Hart, Nils J. Nilsson, and Bertram Raphael. A formal basis for the heuristic determination of minimum cost paths. *IEEE Transactions on Systems Science and Cybernetics*, 4(2):100–107, 1968.

[Haslum, 2011] Patrik Haslum. INVAL. https://github.com/patrikhaslum/INVAL, 2011. Accessed: 2018-05-25.

[Haslum, 2016] Patrik Haslum. Adapting h^{++} for proving plan non-existence. In *Unsolvability International Planning Competition: planner abstracts*, pages 1–1, 2016.

[Helmert *et al.*, 2007] Malte Helmert, Patrik Haslum, and Jörg Hoffmann. Flexible abstraction heuristics for optimal sequential planning. In Mark Boddy, Maria Fox, and Sylvie Thiébaux, editors, *Proceedings of the Seventeenth International Conference on Automated Planning and Scheduling (ICAPS 2007)*, pages 176–183. AAAI Press, 2007.

[Helmert *et al.*, 2014] Malte Helmert, Patrik Haslum, Jörg Hoffmann, and Raz Nissim. Merge-and-shrink abstraction: A method for generating lower bounds in factored state spaces. *Journal of the ACM*, 61(3):16:1–63, 2014.

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

[Hoffmann *et al.*, 2014] Jörg Hoffmann, Peter Kissmann, and Álvaro Torralba. "Distance"? Who cares? Tailoring merge-and-shrink heuristics to detect unsolvability. In Torsten Schaub, Gerhard Friedrich, and Barry O'Sullivan, editors, *Proceedings of the 21st European Conference on Artificial Intelligence (ECAI 2014)*, pages 441–446. IOS Press, 2014.

[Howey and Long, 2003] Richard Howey and Derek Long. VAL's progress: The automatic validation tool for PDDL2.1 used in the International Planning Competition. In Stefan Edelkamp and Jörg Hoffmann, editors, *Proceedings of the ICAPS 2003 Workshop on the Competition: Impact, Organisation, Evaluation, Benchmarks*, 2003.

[Lipovetzky *et al.*, 2016] Nir Lipovetzky, Christian Muise, and Hector Geffner. Traps, invariants, and dead-ends. In *Proceedings of the Twenty-Sixth International Conference on Automated Planning and Scheduling (ICAPS 2016)*, pages 211–215. AAAI Press, 2016.

[Pommerening and Seipp, 2016] Florian Pommerening and Jendrik Seipp. Fast Downward dead-end pattern database. In *Unsolvability International Planning Competition: planner abstracts*, pages 2–2, 2016.

[Schaefer, 1976] Thomas J. Schaefer. Complexity of decision problems based on finite two-person perfect-information games. In *Proceedings of the Eighth Annual ACM Symposium on Theory of Computing (STOC '76)*, pages 41–49, New York, 1976. ACM Press.

[Seipp *et al.*, 2016] Jendrik Seipp, Florian Pommerening, Silvan Sievers, Martin Wehrle, Chris Fawcett, and Yusra Alkhazraji. Fast Downward Aidos. In *Unsolvability International Planning Competition: planner abstracts*, pages 28–38, 2016.

[Somenzi, 2015] Fabio Somenzi. CUDD 3.0.0. http://vlsi.colorado.edu/~fabio/, 2015. Accessed: 2018-05-25.

[Steinmetz and Hoffmann, 2017] Marcel Steinmetz and Jörg Hoffmann. State space search nogood learning: Online refinement of critical-path dead-end detectors in planning. *Artificial Intelligence*, 245:1–37, 2017.

[Torralba *et al.*, 2016] Álvaro Torralba, Jörg Hoffmann, and Peter Kissmann. MS-Unsat and SimulationDominance: Merge-and-Shrink and dominance pruning for proving unsolvability. In *Unsolvability International Planning Competition: planner abstracts*, pages 12–15, 2016.

[Torralba, 2016] Álvaro Torralba. Sympa: Symbolic perimeter abstractions for proving unsolvability. In *Unsolvability International Planning Competition: planner abstracts*, pages 8–11, 2016.