Certified Unsolvability in Classical Planning

1. Theoretical Foundations

Salomé Eriksson Gabriele Röger Malte Helmert

University of Basel, Switzerland

ICAPS 2020

Introduction

Goal

learn about:

- different levels of correctness guarantees
- unsolvability certificates for classical planning
- how to make your planner certifying

Target Audience

- familiar with classical planning
- optional: planner developer

About us

Salomé

Gabi

Malte

Certifying Algorithms

Motivation

- ISP wants to build antenna towers for 5G.
- antenna supplier:
  - "You need at least x towers"
  - shows calculations with a tool, if using less than x towers tool says "unsolvable"
- ISP's options:
  - blindly trust the tool
  - demand some form of correctness guarantee for their tool

Levels of Verification

How can we verify the correctness of an algorithm?

1. theoretical: correctness proof in papers
2. implementation:

verification method	verified inputs
unit test	some predesignated inputs
certifying algorithms	each input when it occurs
theorem provers	all possible inputs

- Unit tests are easy to do, but it is also easy to miss bugs.
- Theorem provers are very expensive but offer highest guarantee (tiny core which is checked very carefully).
- Certifying algorithms strike a balance between trust and effort.

Certifying Algorithms [McConnell et al. 2011]

"A certifying algorithm is an algorithm that produces, with each output, a certificate or witness (easy-to-verify proof) that the particular output has not been compromised by a bug."

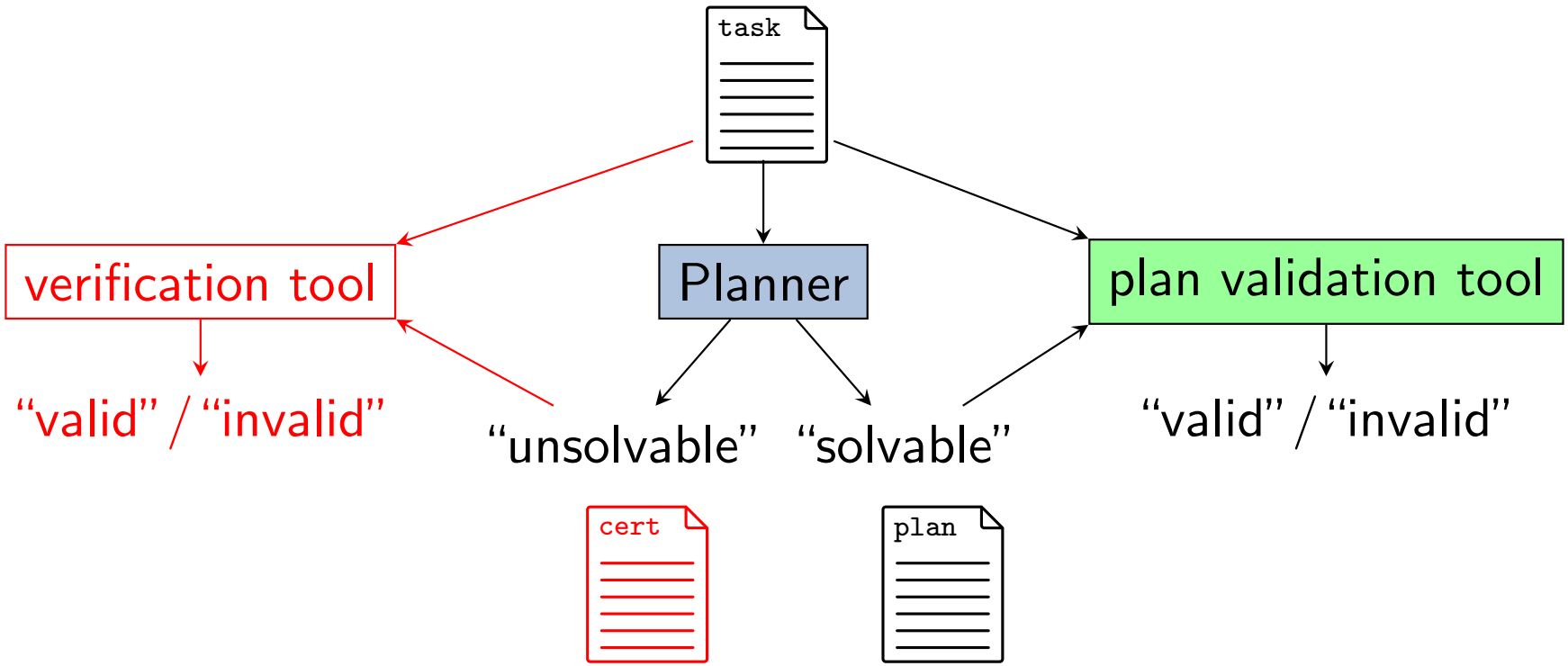
$\rightarrow$ algorithm might still contain bug, but current output is correct

Example

Is CNF formula φ satisifiable?

- yes $\rightarrow$ provide satisfying assignment $\mathcal{I}$
- no $\rightarrow$ provide UNSAT certificate (resolution, DRAT proof...)

Certifying Algorithms in Planning



Guiding Properties

Soundness & Completeness

We can create a certificate for task Π iff Π is unsolvable.

Efficient Generation

Certificate creation incurs only polynomial overhead to the planner.

Efficient Verification

Certificate verification is at most polynomial in its size.

Generality

A wide variety of planning techniques can produce a certificate.

Quis custodiet ipsos custodes?

Certificate must be verified by a verifier. But what if the verifier has bugs?

- Verify the verifier!
- Verify the verifier-verifier!
- . . . ?
- At some point we need to trust something.

Good news: Verifiers are often simpler than the original algorithm.
→ Verify the verifier with theorem provers.

Example: A Formally Verified Validator for Classical Planning Problems and Solutions [Abdulaziz & Lammich, 2018]

Proof Systems

Natural Deduction

- Proof systems are built on axioms and inference rules.
  - axioms: tautology ($A \vee \neg A$)
  - inference rules: conclusion based on premises (if $A \wedge B$ then A)
- Hilbert-style systems try to express as much as possible in axioms.
- Natural deduction in contrast focuses on inference rules.
  - first proposed by Gerhard Genzen [Genzen 1935]
  - should more closely reflect our natural way of reasoning

The proof system presented here uses the natural deduction style.

Inference Rules

Inference rule

An inference rule **I** takes premises $A_1, \ldots, A_n$ and concludes B:

$$\frac{A_1 \qquad \ldots \qquad A_n}{B}\ \mathbf{I}$$

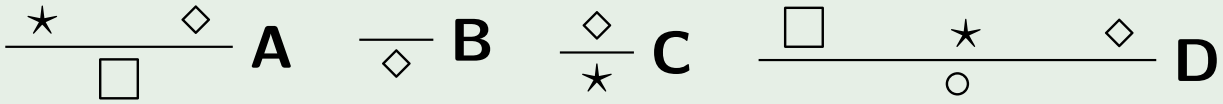
- Rules use placeholder variables and are universally true for all instantiations.
- The correctness of rules can be shown in two ways:
  - within the proof system using existing rules, or
  - outside of the proof system.

  $\rightarrow$ Once proven correct, we can use rules purely syntactically.
- Axioms are rules with no premises.
- Rules can also use and discard assumptions: For example, if under assumption A we can prove B, we have shown $A \rightarrow B$.

Example Proof System

Example

Inference Rules:



Show ○:

#	statement	justification

Example Proof System

Example

Inference Rules:

Show $\circ$:

#	statement	justification
(1)	$\diamond$	from **B**

Example Proof System

Example

Inference Rules:

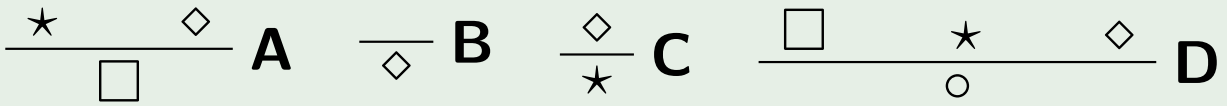
Show $\circ$:

#	statement	justification
(1)	$\diamond$	from **B**
(2)	$\star$	from **C** with (1)

Example Proof System

Example

Inference Rules:



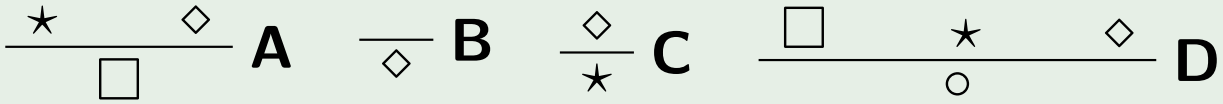
Show $\circ$:

#	statement	justification
(1)	$\diamond$	from **B**
(2)	$\star$	from **C** with (1)
(3)	$\square$	from **A** with (2) and (1)

Example Proof System

Example

Inference Rules:



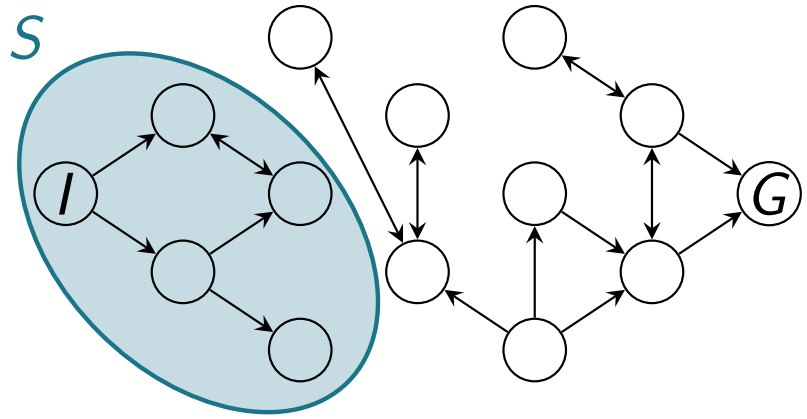
Show $\circ$:

#	statement	justification
(1)	$\diamond$	from **B**
(2)	$\star$	from **C** with (1)
(3)	$\square$	from **A** with (2) and (1)
(4)	$\circ$	from **D** with (3), (2) and (1)

Unsolvability Proof System

First Certificate Attempt

How can we show that a planning task is unsolvable?

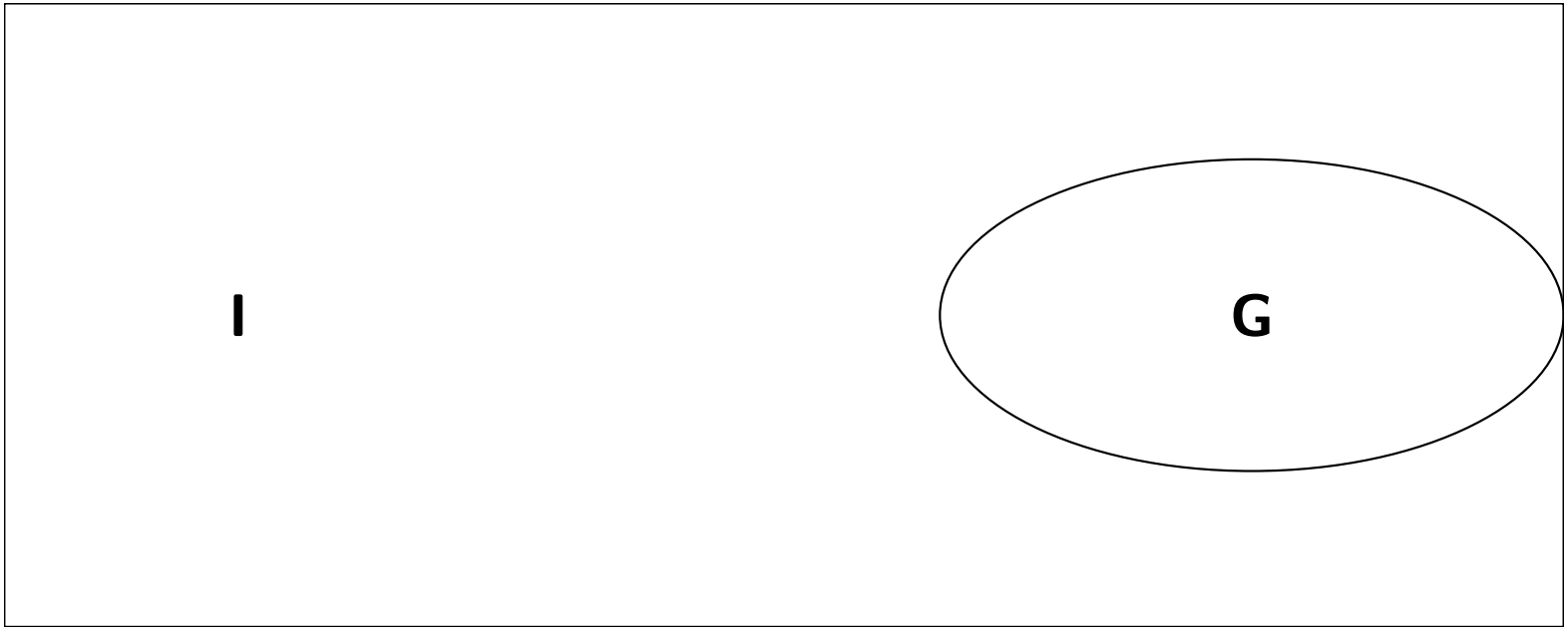


Inductive Certificate [E et al 2017]

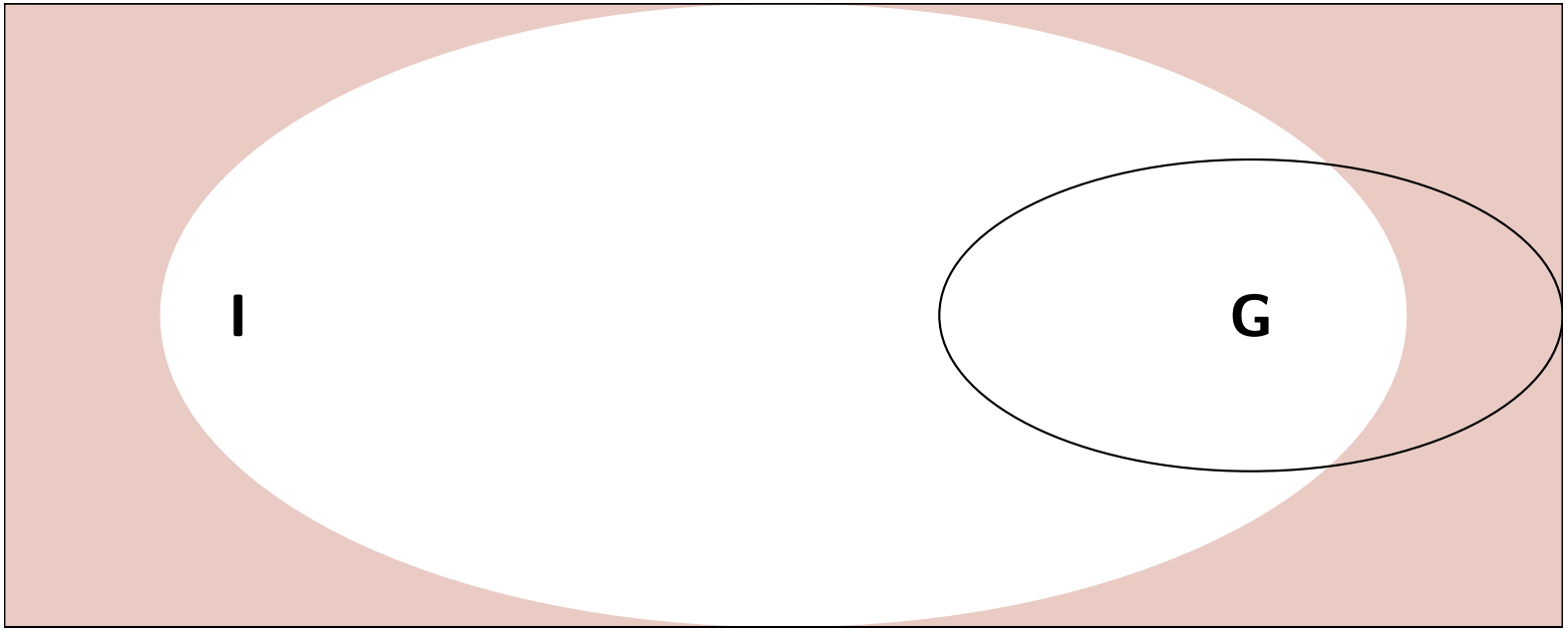
An inductive certificate for a STRIPS planning task $\Pi = \langle \mathbf{V}, \mathbf{A}, \mathbf{I}, \mathbf{G} \rangle$ is a set of states S with the following properties:

- $\mathbf{I} \in S$
- S contains no goal state
- $S[\mathbf{A}] \subseteq S$, where $S[\mathbf{A}] = \{s' \mid s[a] = s' \text{ for some } a \in \mathbf{A}\}$

How Planners Show Unsolvability

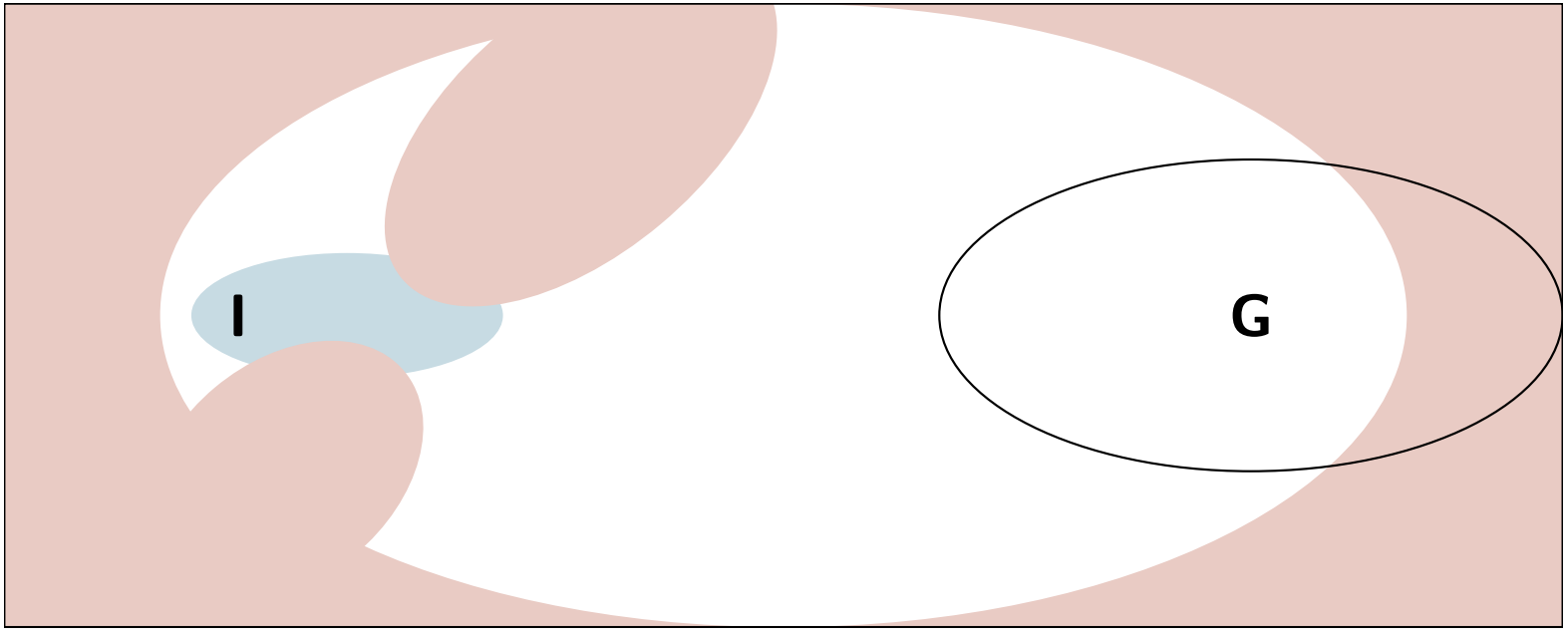


How Planners Show Unsolvability



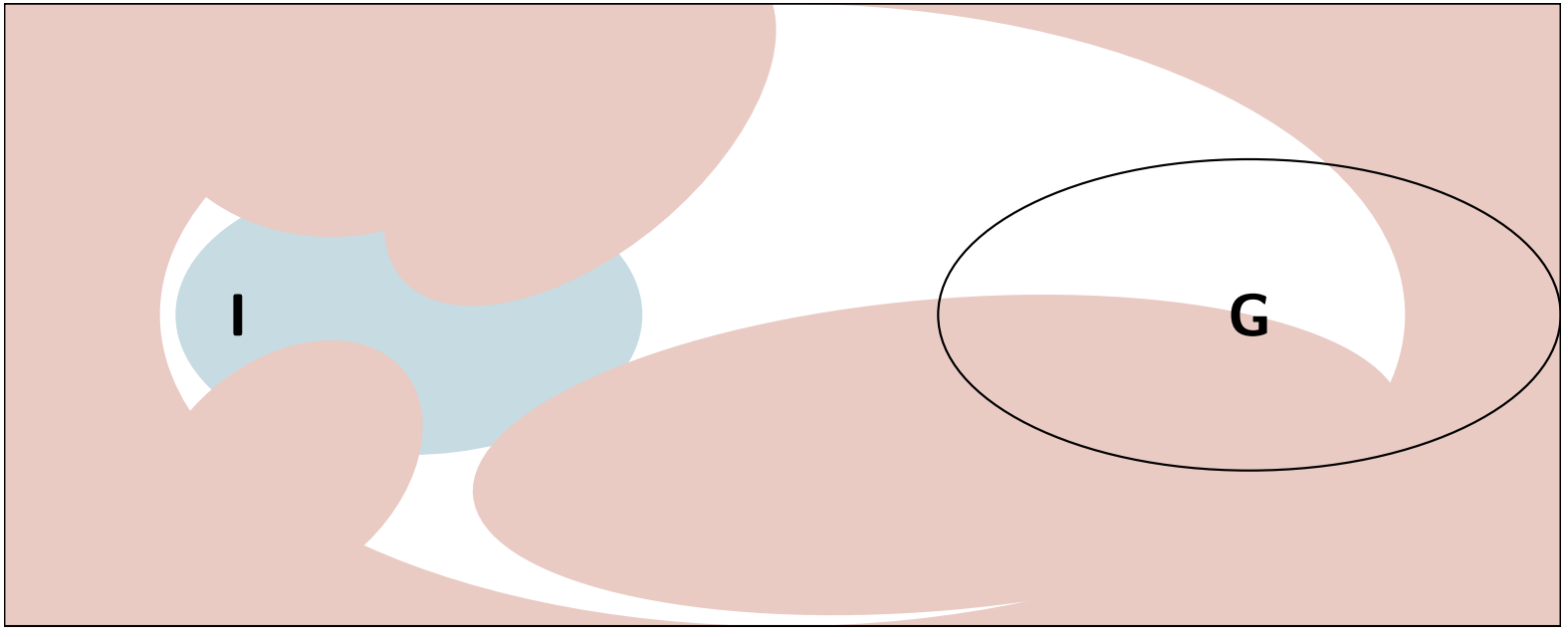
preprocessing

How Planners Show Unsolvability



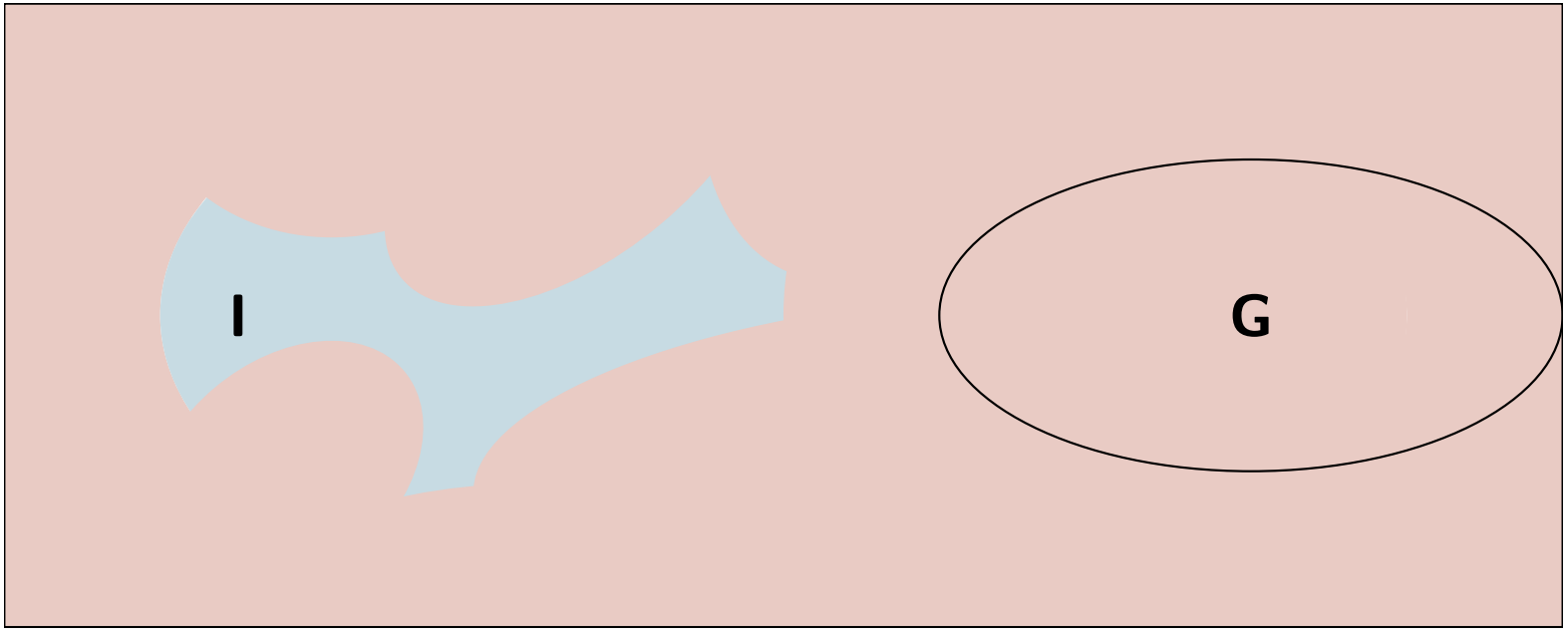
heuristic search with dead-end pruning

How Planners Show Unsolvability



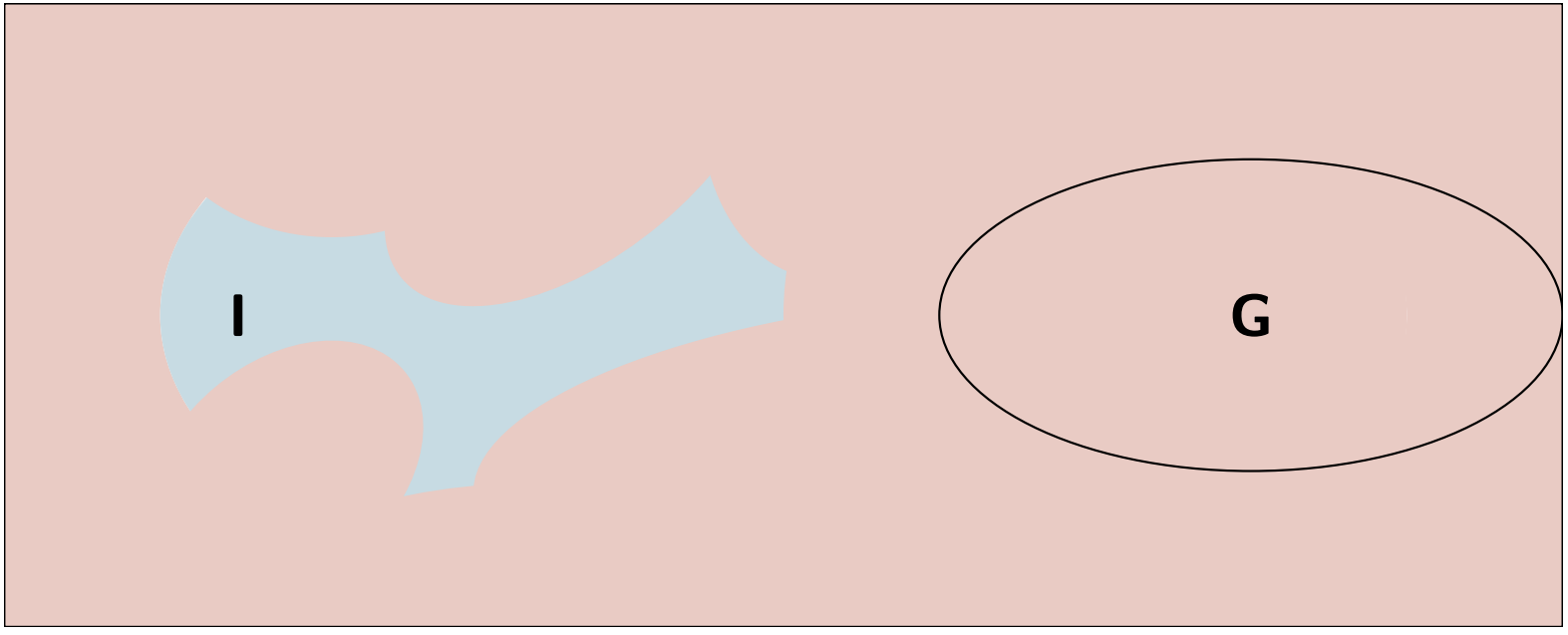
heuristic search with dead-end pruning

How Planners Show Unsolvability



heuristic search with dead-end pruning

How Planners Show Unsolvability



- Uninteresting search space areas get pruned incrementally
- Later pruning steps can use knowledge from previous ones.
- Distilling these steps into a singular argument is difficult.

→ Proof systems can capture this type of incremental reasoning.

Proof System Objects

- state sets S represented as
  - BDD
  - (dual)-Horn formula
  - 2CNF formula
  - explicit enumeration
  - ...
- action sets A represented as ID enumeration

Types of Knowledge

Dead State

A state s is dead if no plan traverses s, i.e. there is no plan $\pi = \langle a_1, \ldots, a_n \rangle$ and $1 \leq i \leq n$ with $s = \mathbf{I}[a_1] \ldots [a_i]$.

$\rightarrow$ captures idea of pruned states (in both directions)

statements in the proof system:

- S dead (all $s \in S$ dead)
- $E \sqsubseteq E'$ (where E and E' are sets of states or actions)
- unsolvable

Inference Rules - Showing Deadness

Empty set **D**ead

$$\frac{}{\emptyset \text{ dead}}\ \textbf{ED}$$

Union **D**ead

$$\frac{S \text{ dead} \qquad S' \text{ dead}}{S \cup S' \text{ dead}}\ \textbf{UD}$$

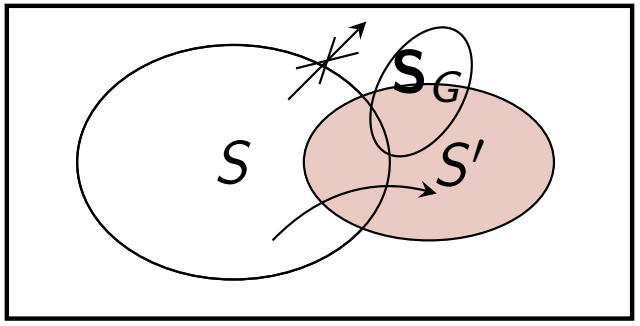
Subset **D**ead

$$\frac{S' \text{ dead} \qquad S \sqsubseteq S'}{S \text{ dead}}\ \textbf{SD}$$

Inference Rules - Showing Deadness

Progression
Goal

$$\frac{S[\mathbf{A}] \sqsubseteq S \cup S' \qquad S' \text{ dead} \qquad S \cap \mathbf{S}_G \text{ dead}}{S \text{ dead}}\ \mathbf{PG}$$



Inference Rules - Showing Deadness

Progression Goal

$$\frac{S[\mathbf{A}] \sqsubseteq S \cup S' \qquad S' \text{ dead} \qquad S \cap \mathbf{S}_G \text{ dead}}{S \text{ dead}}\ \mathbf{PG}$$

Progression Initial

$$\frac{S[\mathbf{A}] \sqsubseteq S \cup S' \qquad S' \text{ dead} \qquad \{\mathbf{I}\} \sqsubseteq S}{\overline{S} \text{ dead}}\ \mathbf{PI}$$

Regression Goal

$$\frac{[\mathbf{A}]S \sqsubseteq S \cup S' \qquad S' \text{ dead} \qquad \overline{S} \cap \mathbf{S}_G \text{ dead}}{\overline{S} \text{ dead}}\ \mathbf{RG}$$

Regression Initial

$$\frac{[\mathbf{A}]S \sqsubseteq S \cup S' \qquad S' \text{ dead} \qquad \{\mathbf{I}\} \sqsubseteq \overline{S}}{S \text{ dead}}\ \mathbf{RI}$$

Inference Rules - Showing Unsolvability

Conclusion **I**nitial

$$\frac{\{\mathbf{I}\} \text{ dead}}{\text{unsolvable}}\ \mathbf{CI}$$

Conclusion **G**oal

$$\frac{\mathbf{S}_G \text{ dead}}{\text{unsolvable}}\ \mathbf{CG}$$

Inference Rules - Set Theory

Union Left

$$\frac{}{E \sqsubseteq (E' \cup E)}\ \mathbf{UL}$$

Intersection Right

$$\frac{}{(E \cap E') \sqsubseteq E}\ \mathbf{IR}$$

Subset Union

$$\frac{E \sqsubseteq E'' \qquad E' \sqsubseteq E''}{(E \cup E') \sqsubseteq E''}\ \mathbf{SU}$$

Subset Intersection

$$\frac{E \sqsubseteq E' \qquad E \sqsubseteq E''}{E \sqsubseteq (E' \cap E'')}\ \mathbf{SI}$$

Subset Transitivity

$$\frac{E \sqsubseteq E' \qquad E' \sqsubseteq E''}{E \sqsubseteq E''}\ \mathbf{ST}$$

...

Inference Rules - Progression and Regression

Action **U**nion

$$\frac{S[A] \sqsubseteq S' \qquad S[A'] \sqsubseteq S'}{S[A \cup A'] \sqsubseteq S'}\ \mathbf{AU}$$

Progression **T**ransitivity

$$\frac{S[A] \sqsubseteq S'' \qquad S' \sqsubseteq S}{S'[A] \sqsubseteq S''}\ \mathbf{PT}$$

Progression to **R**egression

$$\frac{S[A] \sqsubseteq S'}{[A]\overline{S'} \sqsubseteq \overline{S}}\ \mathbf{PR}$$

...

Is this enough?

How can we show $S[A] \subseteq S$ or similar statements?

- depends on planning task and contents of S
- requires semantic analysis
- set theory rules only syntactical

$\rightarrow$ new source of information: basic statements

Basic Statements

B1 $\bigcap L_{\mathbf{R}} \subseteq \bigcup L'_{\mathbf{R}}$

B2 $(\bigcap X_{\mathbf{R}})[A] \cap \bigcap L_{\mathbf{R}} \subseteq \bigcup L'_{\mathbf{R}}$

B3 $[A](\bigcap X_{\mathbf{R}}) \cap \bigcap L_{\mathbf{R}} \subseteq \bigcup L'_{\mathbf{R}}$

B4 $L_{\mathbf{R}} \subseteq L'_{\mathbf{R}'}$

B5 $A \subseteq A'$

X_R	state set variable (represented by formalism **R**)
L_R	state set literal (either X_R or $\overline{X_R}$)
A	action set

- In **B1-B3** all sets must be represented by the same formalism.
- Unions and intersections are bounded.
- We only support pro-/regression for set variables.
- **B4** enables us to mix formalisms.

Soundness and Completeness

Given a STRIPS planning tasks $\Pi = \langle \mathbf{V}, \mathbf{A}, \mathbf{I}, \mathbf{G} \rangle$, there is an proof in the proof system for Π iff Π is unsolvable.

Proof

Soundness: This follows from the correctness of the inference rules and basic statements.

Completeness: Consider $\mathcal{R}^\Pi$, the set of states reachable from $\mathbf{I}$.

#	statement	justification
(1)	$\emptyset$ dead	**ED**
(2)	$\mathcal{R}^\Pi[\mathbf{A}] \sqsubseteq \mathcal{R}^\Pi \cup \emptyset$	**B2**
(3)	$\mathcal{R}^\Pi \cap \mathbf{S}_G \sqsubseteq \emptyset$	**B1**
(4)	$\mathcal{R}^\Pi \cap \mathbf{S}_G$ dead	**SD** with (1) and (3)
(5)	$\mathcal{R}^\Pi$ dead	**PG** with (2), (1) and (4)
(6)	$\{\mathbf{I}\} \sqsubseteq \mathcal{R}^\Pi$	**B1**
(7)	$\{\mathbf{I}\}$ dead	**SD** with (5) and (6)
(8)	unsolvable	**CI** with (7)

Efficient Verification

Verifying Statements

The verifier needs to verify each step of the proof.

- inference rules
  - universally true
  - check if rule is applied corretly (syntax)

  $\rightarrow$ easy to verify
- basic statements
  - sets must be interpreted (semantic)

  $\rightarrow$ depends on set representation formalism

Formalisms & Operations

How can we analyze whether basic statements can be verified efficiently?

1. check each formalism separately
2. check what operations a formalism **R** must support
   - **SE** (sentential entailment): Given **R**-formulas φ and ψ, test whether $\varphi \models \psi$.
   - $\wedge$**BC** (bounded conjunction): Given **R**-formulas φ and ψ, construct an **R**-formula representing $\varphi \wedge \psi$.
   - **toCNF** (transform to CNF): Given **R**-formula φ, construct a CNF formula that is equivalent to φ.
   - . . . [Darwiche & Marquis 2002]

Efficient Verification of **B1**

To verify $\bigcap X_{\mathbf{R}} \subseteq \bigcup X'_{\mathbf{R}}$ efficiently **R** must efficiently support:

	$\lvert\bigcap X_{\mathbf{R}}\rvert = 0$	$\lvert\bigcap X_{\mathbf{R}}\rvert = 1$	$\lvert\bigcap X_{\mathbf{R}}\rvert > 1$
$\lvert\bigcup X'_{\mathbf{R}}\rvert = 0$		**CO**	**CO**, $\wedge$**BC** **toDNF**
$\lvert\bigcup X'_{\mathbf{R}}\rvert = 1$	**VA**	**SE**	**SE**, $\wedge$**BC** **toDNF**, **IM**
$\lvert\bigcup X'_{\mathbf{R}}\rvert > 1$	**VA**, $\vee$**BC** **toCNF**	**SE**, $\vee$**BC** **toCNF**, **CE**	**SE**, $\wedge$**BC**, $\vee$**BC** **toDNF**, **IM**, $\vee$**BC** **toCNF**, **CE**, $\wedge$**BC**

- multiple rows indicate different possible options
- for **B1**: move negated literals to the "correct" side

Efficient Verification of **B1** - Example

	$\lvert\bigcap X_{\mathbf{R}}\rvert = 0$	$\lvert\bigcap X_{\mathbf{R}}\rvert = 1$	$\lvert\bigcap X_{\mathbf{R}}\rvert > 1$
$\lvert\bigcup X'_{\mathbf{R}}\rvert = 0$		**CO**	**CO**, $\wedge$**BC** **toDNF**
$\lvert\bigcup X'_{\mathbf{R}}\rvert = 1$	**VA**	**SE**	**SE**, $\wedge$**BC** **toDNF**, **IM**
$\lvert\bigcup X'_{\mathbf{R}}\rvert > 1$	**VA**, $\vee$**BC** **toCNF**	**SE**, $\vee$**BC** **toCNF**, **CE**	**SE**, $\wedge$**BC**, $\vee$**BC** **toDNF**, **IM**, $\vee$**BC** **toCNF**, **CE**, $\wedge$**BC**

Example

For sets S_1 to S_5 (all represented with **R**), the statement $S_1 \cap \overline{S_2} \subseteq S_3 \cup S_4 \cup S_5$ can be verified efficiently iff **R** supports

- **SE** (sentential entailment) and $\vee$**BC**(bounded disjunction), or
- **toCNF** (transform to CNF) and **CE** (clausal entailment).

Concrete Formalisms

What do BDDs, (dual-)Horn formulas, 2CNF formulas and explicit enumeration support?

- Basic statements **B1-B3** are fully supported by all formalisms.
- Basic statement **B4** between those formalisms is supported in most cases with the following exceptions:
  - $\varphi_{\mathbf{R}} \models \neg\psi'_{\mathbf{R}}$ where $\mathbf{R}$ and $\mathbf{R}'$ are a combination of BDD, (dual-)Horn and 2CNF
  - $\varphi_{\mathbf{(dual-)Horn/2CNF}} \models \psi_{\mathbf{BDD}}$