Complexity of domain-independent planning

José Luis Ambite
Decidability

Decision problem: a problem with a yes/no answer e.g. “is N prime?”

- **Decidable:** if there is a program (i.e. a Turing Machine) that takes any instance and correctly halts with answer “yes” or “no”.

- **Semi-decidable:** if program halts with correct answer in one of the cases (either “yes” or “no”) but not in the other case (goes on forever)

- **Undecidable:** There is no algorithm to solve the problem. Ex: Halting Problem.
Undecidability (Intuition)

There are more problems than solutions!!!

- Turing Machine
  - Can be encoded as an integer
    => Countably Many ($\mathbb{N}$)

- Problem
  - Mapping from inputs ($\mathbb{N}$) to outputs ($\mathbb{N}$)
    => Uncountably Many ($\mathbb{R} = 2^{\mathbb{N}}$)
Planning Decision Problems

- Plan Existence (PLANSAT):
  - Given a planning problem instance $P = (I, O, G)$,
  - Is there a plan that achieves goals $G$ from initial state $I$ using operators from $O$?

- Plan Length (PLANMIN):
  - Given a planning problem instance $P = (I, O, G)$ and an integer $k$ (encoded in binary),
  - Is there a plan that achieves goals $G$ from initial state $I$ using less than $k$ operators from $O$?
## Decidability results from [Erol et al 94]

<table>
<thead>
<tr>
<th>Allow function symbols?</th>
<th>Allow infinitely many constant symbols?(\alpha)</th>
<th>infinite initial states?(\alpha)</th>
<th>Allow delete lists and/or negated preconditions?</th>
<th>PLAN EXISTENCE (telling if a plan exists)</th>
</tr>
</thead>
<tbody>
<tr>
<td>yes</td>
<td>xxx/no</td>
<td>xxx/no</td>
<td>yes/no/no/(\beta)</td>
<td>semidecidable</td>
</tr>
<tr>
<td>no</td>
<td>no</td>
<td>yes/(\gamma)</td>
<td>no</td>
<td>decidable</td>
</tr>
<tr>
<td>no</td>
<td>yes</td>
<td>no</td>
<td>yes/(\delta)</td>
<td>semidecidable</td>
</tr>
<tr>
<td>yes</td>
<td>no</td>
<td>no</td>
<td>yes/(\delta)</td>
<td>decidable</td>
</tr>
</tbody>
</table>

\(\alpha\), \(\beta\), \(\gamma\), \(\delta\)
Decidability results from [Erol et al. 94]

- Exploits relationship between planning and logic programming.
- Can transform a planning problem without delete lists or negative preconditions to a logic program (and vice versa) in polynomial time:
  - R1: \( a \leftarrow b_1 \land b_2 \land b_3 \)
  - Op_R1: \([\text{pre: \{b1, b2, b3\} add: \{a\} del: \{\}}]\)
- function symbols => undecidable
  - unless have acyclicity and boundedness conditions.
- No function symbols and finite initial state => decidable
Worst-case Complexity of Problems

- If a problem is decidable, we might ask how many resources a program requires to compute the answer (in the worst case).

- We measure the resources a program takes in terms of *time* or *space* (memory), as a function of the size of the input.

- If a problem is known to be in some complexity class, then we know there is a program that solves it using resources bound by that class.
Complexity Classes

- A problem is in P: if ∃ program to decide it taking polynomial time in the size of the input.
- A problem is in NP: if ∃ nondeterministic program that solves it in polynomial time.
  - program makes polynomially-many guesses to find the correct answer (solution check also P). Ex: SAT.
- A problem is NP-Complete if any problem in NP can be reduced to it. Ex: SAT
- PSPACE: polynomial space. Ex: QSAT
- EXP, EXPSPACE: exponential time, space
- NEXP: nondeterministic exponential time, etc.
Hierarchy of Complexity Classes

Undecidable

Decidable

EXPSPACE

NEXP

EXP

PSPACE

NP

P

PSPACE \subset EXPSPACE

P \subset EXP

PSPACE = NPSPACE

P \subset NP \subset PSPACE

P =? NP
States, operators, plans.
How many, how big?

Assume no function symbols, finite states, n objects, m predicates with arity r, and o operators (with s variables max each):

- Possible atoms: $p = m \times n^r$
  - $\Rightarrow$ Each state requires exponential space
- Possible states = Powerset{$p$} = $2^p$
  - $\Rightarrow$ State space is double exponential
- Possible ground operators = $o \times n^s$
- In general plans will be bounded by the number of states. (Why?)
Complexity bounds for decidable domain-independent planning

- **With no restrictions:** EXPSPACE
  - Search through all states
  - Each state consumes exponential space
- **No delete lists:** NEXP
  - Operators only need to appear once
  - Choose among exponentially-many operators
- **No negative preconds and no deletes:** EXP
  - Plans for different subgoals won’t negatively interfere with each other => order does not matter (no choose)
Propositional Planning

- Propositions = 0-ary predicates
- State has $p$ propositions (polynomial)
- Possible States = Powerset{$p$} = $2^p$ (single! exponential)
- Number of Operators is also polynomial

=> Reduced complexity:
  - General case: from EXPSPACE to PSPACE
  - No deletes: from NEXP to NP
  - No deletes and no negative preconds: from EXP to P

If you know the operators in advance, this in effect bounds the arity of predicates and operators, with the same result
Propositional PLANSAT

[Bylander94]
Propositional PLANMIN

- If PLANSAT was PSPACE(NP)-complete, PLANMIN is also PSPACE(NP)-complete
What does all this mean?

- Domain-independent planning in general is very hard: PSPACE, NP, ...
- Even for very restricted cases:
  - 2 positive preconds, 2 effects (PSPACE)
  - 1 precond, 1 positive effect (NP)
    ... in the worst case ...
- What about the average case, structured domains, real-world problem distributions?
  => Heuristics, reuse solutions, learning