Planning as Satisfiability

Ron Parr
CPS 570

Planning vs. Satisfiability

- Planning is PSPACE-hard
- Satisfiability is NP-complete
- PSPACE-hard is (almost certainly) harder

Q: Use satisfiability to solve planning?
A: Bounded length planning is in NP

Propositionalizing (Grounding Out)

- Planning domains typically expressed in relational (first order/predicate) form
- SAT is propositional

What do we do?
- Introduce time indices
- Create propositions for every substitution
- PDDL actions -> huge number of propositions

- Can automated, but human expertise helps if you can change representation to reduce number of arguments

Action Representation

- Actions models look like this (simplified):
  - If preconditions for action_i are met at time step j and action_i is taken at time step j, then the effects of action_i at time step j+1 are true

- Convert this to a disjunction
  - Recall p\rightarrow q = (\neg q \lor p)
  - Apply DeMorgan's law to convert conjunctions to disjunctions in preconditions
  - Split up conjunctions in effects
What does a plan look like?

- Plan will be a satisfying assignment to a (huge) SAT instance
- By checking which action variables are true at any time, we can construct a sequence of actions to achieve the goal

Anomalous Models

- With situation calculus/planning as theorem proving, we ask the theorem prover if a situation exists where goal is achieved
  - Situations are created by stringing together actions
  - Goal must be necessarily true final situation for theorem prover to succeed
- Satisfiability asks if single assignment of truth values to variables satisfies a logical expression
  - Need to avoid degenerate solutions explicitly
  - False preconditions can make expression true

Tricks

- Require actions to imply their effects as well (iff)
- Require one action to be taken at every step
- Reduce number of arguments (save memory)
- Add additional axioms to reduce search space

Why does reducing the number of arguments help so much?

- $O(n^3)$ vs. $O(n^2)$
- In general if you have $n$ objects and your predicates take $k$ arguments you create $O(n^k)$ propositions for each predicate!
### Intrinsic Hardness

- Are all planning problems hard?
- In fact, no single problem instance is hard—only sets of problems can be hard or not
- Some sets of planning problems may be easy, e.g., non-optimal blocks world planning
- Claim: Earlier planning competitions had many problem classes that were in P
- Intrinsically hard means problem class is NP-hard
- Claim: These are more “real world”

### What is “Real World”?

- A1: I don’t know
- A2: Whatever somebody is willing to pay “real money” for
  - Similar to problems people solve?
  - Similar to problems people can’t solve easily?
- A3: Not classical planning problems: Planning competition has evolved to include problems where actions have non-deterministic outcomes

### Why didn’t SATPlan happen earlier?

- Does CPU power alone explain this?
- GSAT hadn’t been invented yet
- Memory was expensive
- Memory bandwidth not very high
- People hadn’t worked out tricks to reduce memory consumption

### Other examples of xformations to SAT?

- Often- tried, sometimes with success
  (caveat: RP is not expert in this)
- SATPlan = most notable, high profile successes
- Other fields have different canonical problems
- Operations research uses integer programs as their basic language for NP-complete problems