Planning and Scheduling Summary

Alexander Moriarty, Alexander Hagg, *BRS University of Applied Sciences*
Planning and Scheduling summary

January 10, 2014

Planning and Scheduling mid semester summary in preparation of exam

1 Introduction

- DEFINITION: Plan
  - A collection of actions for performing some task or achieving some objective
- Conceptual model

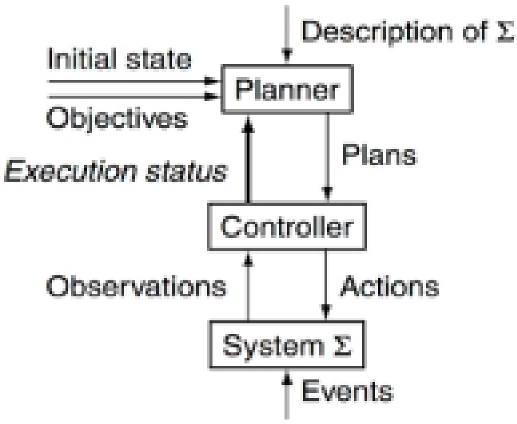


Figure 1: *Conceptual model*

- State-transition system
  $\Sigma = (S, A, E, \gamma)$
  - S: states
  - A: actions (controllable)
  - E: events (uncontrollable)
  - $\gamma : S \times (A \cup E) \mapsto 2^S$: state-transition function
- Observation function
  - $h : S \mapsto O$
- Planner
  - description of Σ, initial state $s_0 \in S$, some objective
- Objectives
  - S_g: set of goal states
  - condition over set of states followed by system
  - utility function
- Restrictive assumptions
  - A0: finite σ
  - A1: σ is fully observable (observation function is id())
  - A2: deterministic σ, action: one possible outcome
  - A3: static σ: E is empty
  - A4: attainment goals: goal state or a set of goal states
  - A5: sequential plans
  - A6: implicit time (instant transitions)
  - A7: off-line planning
- Classical planning
  - requires all eight restrictive assumptions
  - Given (Σ, s_0, S_g)
  - find a sequence of actions $< a_1, a_2, ..., a_n >$
  - that produces sequence of transitions $s_1 = \gamma(s_0, a_1), ...$
  - such that $s_n \in S_g$
- Relaxing assumptions

- A0: discrete logic (e.g. 1st order), continuous variables)
- A1: if we don't relax other restrictions, only uncertainty is about s_0
- A2: nondeterministic outcomes, seek policy/contingency plan, MDPs (probabilities)
- A1+A2: Finite POMDPs
- A0+A2: Continuous or hybrid MDPs
- A0+A1+A2: Continuous or hybrid MDPs
- A3: Other agents/dynamic environment, randomly behaving environment
- A1+A3: Imperfect-information games
- A5/A6: Temporal planning
- A0+A5+A6: Planning and resource scheduling

2 Search and Complexity

Problem Solving Agents

-

Problem Types

- Deterministic, fully observable
- Non-observable
- Nondetermin

Problem Formulation

Example Problems

Basic Search Algorithms

3 Logic and Inference

4 Representing Plans

- **Classical representation**
  - function-free FOL
  - Atom: predicate symbol and args $on(c1, c3)$, $on(c1, x)$
  - Ground expression: instantiated var $on(c1, c3)$
  - Nonground expression: at least one var $on(c1, x)$
  - Substitution $\theta = x_1 \leftarrow v_1, ...$
  - Instance of expression e: result of applying θ
- State : set of ground atoms
- Operator
  - (name, precond, effects)
  - $take(k, l, c, d, p)$
    $precond : belong(k, l)$
    $effects : holding(k, c), \neg empty(k)$
  - no need for negative effects (closed world assumption), only when they are explicitly touched (AM?)
- Action
  - ground **instance** of operator
- Notation
  - S^+, S^-, $precond^+$, ...
- Result of action
  - $\gamma(s, a) = (s \backslash effects^-(a) \cup effects^+(a)$
- Planning problem
  - given: domain (L, O) (Language, Operators)
  - $P = (\Sigma, s_0, S_g)$ (state-trans., initState, goal formula)
  - $\Sigma = (S, A, E, \gamma)$
- Plan
  - $\sigma = \langle a_1, a_2, ..., a_n \rangle$
  - is a solution for P if it is executable and achieves g
- **Set-theoretic representation**
  - classical representation + restriction to PL
  - collection of propositions (bool) instead of ground atoms
  - effects: delete, add
- exponential blow-up
- **State-Variable representation**
  - State variables: pos(x) = y (if block x is on block y)
- **Expressive power**
- linear time/space conversion between all three representations, except for expBlowup on conversion to set-theoretic

- classical rep: most popular
- set-theoretic: much more space, useful for algorithms that manipulate ground atoms directly
- state-variabl: equivalent to classical, more natural for engineers, used for non-classical planning problems (incl. functions!)

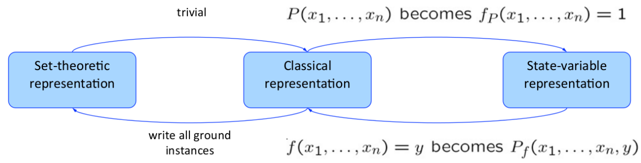
5 Complexity of Classical Planning

- **Complexity analysis**
- general idea: translate classical planning -¿ language-recognition problem and examine its complexity
- Language L, alphabet A
- recognition procedure R(x): yes iff $x \in L$, else no or fail to terminate
  - **Plan-Existence** P
  - Problem has solution
  - **Plan-Length** (P,n)
  - Problem has solution ¡= n
- Runtime, space complexity
- **Restrictions of classical planning**
- Operators are fixed or input?
- Allow infinite initial states?*
- Allow function symbols?*
- Allow negative effects?
- Allow neg. preconditions?
- Allow ¿1 preconditions?
- Operators have conditional effects?*
- *: outside CP
- **(Un)decidability**

	Decidability	
function symbols	Plan-Existence	Plan-Length
no (CP)	dec	dec
yes	semidec	dec

- **Complexity results**
- Well I guess, Plan-Existence is less complex than Plan-Length (AM?)

- **Equivalences**
- Set-theoretic and classical are basically identical
- Both: exponential blowup (input)
- Class. and state-var are basically equivalent
- (state-var:some restrictions not possible)



6 State Space Planning

- **State space vs plan space**
- State space
  - Node: state of the world
  - Plan: path through space
- Plan space
  - State: set of partially instantiated operators and some constraints
  - Impose more constraints until Plan
- **Linear search**
- Work on one goal until completely solved
- Order of problem solving is linearly-related to order in which plan actions are executed
- Maintain **goal stack**
- Implications
  - No interleaving of goal achievement
  - Efficient search if goals do "not" interact
- **Means-End analysis**
- Search relevant aspects of problem, means/operators, ends/goals
  - Start from the goal
  - find difference to start state
  - find operator that reduces this difference
- General Problem Solver (GPS)

- **Forward search**
  $s \leftarrow s_0$
  $\pi \leftarrow empty\_plan$
  loop
  if s satisfies g $\rightarrow$ return π
  $E \leftarrow$ {a ground instance of o $\in$ O, precond(a) true in s}
  if $E = \emptyset \rightarrow$ failure
  nondet choose a $\in$ E
  $s \leftarrow \gamma(s, a)$
  $\pi \leftarrow \pi, a$
  endloop

- nondet choosing -¿ (par/seq) do all actions a

- seq, we need to use backtracking (as soon as a zero set or goal state is reached)

- = sound,complete

- breadth-first, best-first
  - sound and complete
  - memory exponential in length of solution

- depth-first, greedy search
  - Worst-case mem is linear in length of solution
  - Sound, but not complete
  - but CP has only finite states $\rightarrow$ loop-checking solves completeness

- large branching factor (need good heuristic / pruning)

- **Backward search**

- Means-end analysis

- start at goal and compute inverse state transitions

- a makes at least one of g's literals true and non false

- $g' = \gamma^{-1}(g, a) = (g \backslash effects(a)) \cup precond(a)$

- take the goal state, remove effects of a and add the preconditions
  $\pi \leftarrow empty\_plan$
  loop
  if s satisfies g $\rightarrow$ return π
  $A \leftarrow \{a|a$ ground instance of o in O and $g' = \gamma^{-1}(g, a)$ defined $\}$
  if $A = \emptyset$ failure
  nondet choose $a \in A$
  $\pi \leftarrow a, \pi$
  $g \leftarrow \gamma^{-1}(g, a)$
  endloop

- Branching factor smaller, but can be big because of more actions than needed

- Problem with (x,y): you need to instantiate y with every state

- **Lifting**

- reduce branching factor by **partially instantiating** operators (=lifting)
  $\pi \leftarrow empty\_plan$
  loop
  if s_0 satisfies g $\rightarrow$ return π
  $A \leftarrow \{(o, \theta)|o$ standardization of o in O, θ is mgu for an atom of g and atom of effects+(o) and $\gamma^{-1}(\theta(g), \theta(o))$ defined
  if $A = \emptyset$ failure
  nondet choose $(o, \theta) \in A$
  $\pi \leftarrow$ concatenation of $\theta(o)$ and $\theta(\pi)$
  $g \leftarrow \gamma^{-1}(\theta(g), \theta(o))$
  endloop

- more complicated than backward-search (keep track of substitutions)

- smaller branching factor

- if sub-problems independent, all orderings need to be tried

- **STRIPS**

- Modified backward search (instead of γ^{-1}, each set of sub-goals is precond(a)

- a = executable? go forward as far as possible.

- (AM?) We need to write down all search algorithms as examples on the whiteboard and copy them here. Algorithms written out are just too unclear and direct comparison is difficult.

- STRIPS assumption -¿ solved frame problem

- introduces: difference, sub-goals

- Block stacking

- Sussman Anomaly: STRIPS cannot produce irredundant solution because it is non-interleaved.

- Register Assignment Problem (TODO!)

- **Linear Planning**

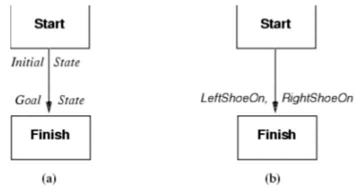
- + reduced search space (goals one at a time)
- + advantageous if goals are independent
- + sound
- - suboptimal solutions
- - incomplete

- solve -: domain-specific knowledge (fwd/bwd search)
    - ds knowledge can prune search space
    - ds specific algorithm
    - TODO solve Sussman Anomaly with ds knowledge

7 Plan Space Planning

- **Motivation**
- in SSP: try all orderings before "no solution"
- PSP: do not commit to orderings, instantiations, etc.
- **Partial Order Plan vs Total Order Plan**
- POP: "parallel" paths
- **PSP**: Backward search from g
- Each node is **partial plan**
    - { part.-instantiated operators (**steps**) }
    - Sets of constraints
- Refine until solution
- **Constraints**
    - **Precedence constraints**: a must precede b
    - **Binding constraints**: (in)equality
    - **Causal links**: use step a to establish prec p needed by c
- No more **flaws** → plan
- **Representation**
- P = ⟨ steps, orderings, bindings, causallinks ⟩

$$
\begin{aligned}
&\mathit{Plan}(\textsc{Steps}{:}\{S_1 : \mathit{Op}(\textsc{Action}{:}\mathit{Start}),\\
&\qquad S_2 : \mathit{Op}(\textsc{Action}{:}\mathit{Finish},\\
&\qquad\quad \textsc{Prec}{:}\mathit{RightShowOn} \wedge \mathit{LeftShowOn})\}\\
&\quad \textsc{Orderings}{:}\{S_1 \prec S_2\}\\
&\quad \textsc{Bindings}{:}\{\}\\
&\quad \textsc{CausalLinks}{:}\{\}\quad)
\end{aligned}
$$



8 Graph Based Planning

9 Satisfiability Based Planning

10 Hierarchical Task Network Planning

* partial ordering enables interleaving of tasks. E.g. we need a hammer and a drill and we do not care in what order