CogSysI Lecture 7: Different Approaches To AI Planning

Intelligent Agents

WS 2006/2007

Part II: Problem Solving and Planning

II.7 Different Approaches To AI Planning

Deductive Planning

- Deductive inference can be used to solve planning problems.
- Introduce a situation variable to store the partial plans:
  $s_{i+1} = \text{put}(A, B, s_i)$, ... $s_2 = puttable(A, s_1)$
  $s = \text{put}(A, B, \text{puttable}(A, [\text{on}(A, C), \text{clear}(A)...]))$
- Situation calculus: Introduced by McCarthy (1963) and used for plan construction by resolution by Green (1969)
- In general: extensions of FOL (action languages)
- Proof logically, that a set of goals follows from an initial state given operator definitions (axioms)
- Perform the proof in a constructive way (plan is constructed as a byproduct of the proof)

Situation Calculus

A1 *on(a, table,* s_1*)* (literal of the initial state)

A2 (axiom for put-operator)

$\forall\, S[\text{on(a, table, S)} \rightarrow \text{on(a, b, put(a, b, S))}] \equiv$

$\neg$*on(a, table, S)* $\vee$ *on(a, b, put(a, b, S))* *(clausal form)*

Proof the goal predicate *on(a, b,* S_F*)*

1. $\neg$*on(a, b,* S_F*)* (Negation of the theorem)
2. $\neg$ *on(a, table, S)* $\vee$ *on(a, b, put(a, b, S))* (A2)
3. $\neg$ *on(a, table, S)* (Resolve 1, 2) $\hookrightarrow$ answer(put(a, b, S))
4. *on(a, table,* s_1*)* (A1)
5. contradiction (Resolve 3, 4) $\hookrightarrow$ answer(put(a, b, s_1))

s_2 = *on(a, table,* s_1*)* with *on(a, b,* s_2*)* exists and s_2 can be reached by putting a on b in situation s_1.

Frame Problem Revisited

- No closed world assumption $\hookrightarrow$ full expressive power of FOL
- Problem: additionally to axioms describing the effects of actions, frame axioms become necessary
- Frame axioms are necessary to allow proofing conjunctions of goal literals.
- Example for a frame axiom:
  $\forall\, S[\mathsf{on}(Y, Z, S) \rightarrow \mathsf{on}(Y, Z, \mathsf{put}(X, Y, S))]$
  `on(Y, Z, put(X, Y, S)) ← on(Y, Z, S)`
  After a block X was put on a block Y, it still holds that Y is lying on a block Z, if this did hold before the action was performed.

Blocksworld in Prolog

Effect Axioms:

on(X, Y, put(X, Y, S)) $\leftarrow$	clear(X, S) $\wedge$ clear(Y, S)
clear(Z, put(X, Y, S)) $\leftarrow$	on(X, Z, S) $\wedge$ clear(X, S) $\wedge$ clear(Y, S)
clear(Y, puttable(X, S)) $\leftarrow$	on(X, Y, S) $\wedge$ clear(X, S)
ontable(X, puttable(X, S)) $\leftarrow$	clear(X, S)

Frame Axioms:

clear(X, put(X, Y, S)) $\leftarrow$	clear(X, S) $\wedge$ clear(Y, S)
clear(Z, put(X, Y, S)) $\leftarrow$	clear(X,S) $\wedge$ clear(Y, S) $\wedge$ clear(Z, S)
ontable(Y, put(X, Y, S)) $\leftarrow$	clear(X, S) $\wedge$ clear(Y, S) $\wedge$ ontable(Y, S)
ontable(Z, put(X, Y, S)) $\leftarrow$	clear(X, S) $\wedge$ clear(Y, S) $\wedge$ ontable(Z, S)
on(Y, Z, put(X, Y, S)) $\leftarrow$	clear(X, S) $\wedge$ clear(Y, S) $\wedge$ on(Y, Z, S)
on(W, Z, put(X, Y, S)) $\leftarrow$	clear(X, S) $\wedge$ clear(Y, S) $\wedge$ on(W, Z, S)

Blocksworld in Prolog cont.

Frame Axioms cont.:

clear(Z, puttable(X, S)) $\leftarrow$ clear(X, S) $\wedge$ clear(Z, S)

ontable(Z, puttable(X, S)) $\leftarrow$ clear(X, S) $\wedge$ ontable(Z, S)

on(Y, Z, puttable(X, S)) $\leftarrow$ clear(X, S) $\wedge$ on(Y, Z, S)

clear(Z, puttable(X, S)) $\leftarrow$ on(Y, X, S) $\wedge$ clear(Y, S) $\wedge$ clear(Z, S)

ontable(Z, puttable(X, S)) $\leftarrow$ on(Y, X, S) $\wedge$ clear(Y, S) $\wedge$ ontable(Z, S)

on(W, Z, puttable(X, S)) $\leftarrow$ on(Y, X, S) $\wedge$ clear(Y, S) $\wedge$ on(W, Z, S)

Facts (Initial State):

on(d, c, s_1) on(c, a, s_1)

clear(d, s_1) clear(b, s_1)

ontable(a, s_1) ontable(b, s_1)

Theorem (Goal):

on(a, b, S) $\wedge$ on(b, c, S)

AI Planning: Current Approaches

Domain-independent planning in deterministic domains

- 70ies: STRIPS and deductive planning
- 80ies: partial order planning (e.g. UCPOP)
- 90ies: extensions of STRIPS language (PDDL) and new, efficient algorithms – Graphplan, SATPlan

Graphplan: Basic Ideas

- Make search for a plan more efficient by first constructing a planning graph from which the valid plan can be extracted
- A planning graph is a directed, levelled graph, that is, nodes can be partitioned into disjoint sets $L_1, L_2, \ldots, L_n$ such that the edges connect only nodes in adjacent levels
- Two kinds of nodes
  - Starting with level 0, nodes at even levels represent propositions true at time t_i
  - Nodes at odd levels represent possible actions at time t

Graphplan: Basic Ideas cont.

- Four kinds of edges
  - Precondition edges: from propositions to actions
  - Add edges: from actions to propositions
  - Del edges: from actions to propositions
  - No-op edges: from propositions to propositions

Rocket Example

move(?r ?f ?t)
PRE: (at ?r ?f), (?f $\neq$?t), (has-fuel ?r)
ADD: (at ?r ?t)
DEL: (at ?r ?f), (has-fuel ?r)

unload(?r ?p ?c)
PRE: (cargo ?c) (at ?r ?p) (in ?c ?r)
ADD: (at ?c ?p)
DEL: (in ?c ?p)

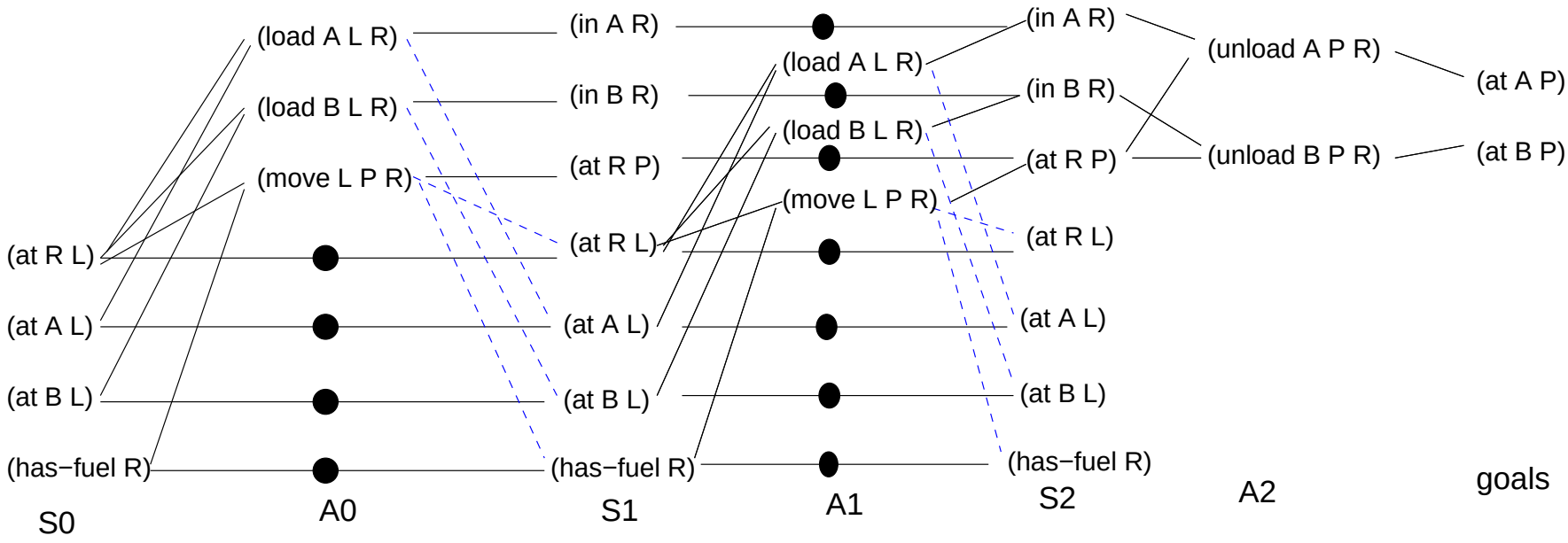
load(?r ?p ?c)
PRE: (cargo ?c) (at ?r ?p) (at ?c ?p)
ADD: (in ?c ?r)
DEL: (at ?c ?p)

Example Problem:
Initial State: {(at R L), (at A L), (at B L), (has-fuel R), (cargo A), (cargo B)} Goal: {(at A P), (at B P)}

Construct Planning Graph

- Start with initial state
- Introduce all propositions as nodes on level S_0
- Construct the next levels:
  - Find all actions whose preconditions are contained in level S_i and introduce them as nodes in level A_i
  - Introduce all propositions of the ADD and DEL list in level S_{i+1}
  - Copy all propositions from level S_i to level S_{i+1}

Rocket Planning Graph



Remarks on Planning Graphs

- can be constructed in polynomial time
  since planning is NP-hard (PSPACE complete), plan extraction is of exponential effort!
- based on propositional logic
  may be hard to transform a problem given in PDDL into propositional form!
  (all possible instantiation of actions, which instances are legal?)
- represent a partially-ordered plan (independent actions are given on the same level and can be performed in arbitrary order)
  actions are independent, if one does not delete a precondition or an add-effect of the other
- to facilitate plan extraction, mutual exclusive nodes can be identified

Mutex Relations

Two actions at one level are mutex, if

- either one deletes a precondition or add-effect of the other (interference)
- their preconditions are mutex

Two propositions are mutex, if
actions adding them are mutex.
Mutex Relations of the Rocket Problem:

- (move L P R) and (load A L R) are mutex because the precondition for “load” is deleted by “move”

Mutex Relations are a heuristic:

- Not all mutex relations can be found by this rules
- More complex mutex relations (between triples etc.) are not checked (too expensive)

Extraction of a Valid Plan

by backwards-search (goal regression)

- Level-by-level to exploit the mutex constraints
- Given a set of goals at level i, find a set of actions (including no-ops) at the preceding level which have these goals as add-effects
- The preconditions of these actions form the subgoals for the next regression step
- if a goal set at a level is unsolvable, backtrack and select different actions
- continue until success or prove that the original goals are not solvable at this level ($\hookrightarrow$ expand planning graph to a next level containing all goals)

Termination

- Termination on unsolvable problems is tricky, but the Graphplan algorithm is sound and complete for STRIPS-style operators where only a finite set of propositions can be generated.

Planning in Real-World Domains

- Incomplete Information
  - Conformant planning: Creat plans that work for all cases
  - Conditional planning: sense world during execution and decide which branch of the plan to follow
- Incorrect Information
  - Execution monitoring: check for unsatisfied preconditions
  - Re-planning
- Continuous planning: create new goals during acting in real time
- Multiagent planning

Including Knowledge

Using knowledge about the structure of the domain

- Hierarchical Planning (decomposition rules)
  cf. problem solving with AND-OR trees
- Domain axioms
- Domain specific search strategies

$\hookrightarrow$ larger plans become feasible (necessary for many real world problems, e.g. Mars Mobile)
Alternative to knowledge engineering: Learning of planning strategies!

Further Topics

- Interleaving plan construction and plan execution
- Plan revision
- Planning with temporal/resource constraints
- Non-deterministic planning
- ...