CogSysI Lecture 5: Basic AI Planning

Intelligent Agents

WS 2004/2005

Part I: Acting Goal-Oriented

Basic AI Planning
AI Planning

- AI planning deals with the formalization, implementation and evaluation of algorithms for construction plans.

- Plan: sequence of actions for transforming a given state into a state which fulfills a predefined set of goals.

- Basic approach to state based planning: Strips

- Basic approach to deductive planning: Situation Calculus

- Alternative to planning: reinforcement learning
Strips


- a **linear** (and therefore incomplete) approach
- relies on the **closed-world assumption** (CWA)
- today: extensions of the Strips language and non-linear algorithms
- classical examples: moving boxes between rooms (“Strips World”), blocksworld

![Diagram of putting block A on block B]
Components of a Planning Formalism

- A language to represent states, goals, operators
- Typically, different subsets of FOL for states, goals and operators.
  States as conjunction of positive ground literals, goals might be more complex (allowing quantified variables, disjunction, etc.)
- In contrast to problem solving, operators are typically represented as schemes
- Operator semantics: achieving a state change by applying an action (state space model)
- An algorithm for constructing a plan (a search algorithm; in contrast to problem solving, typically backwards).
- Crucial difference to problem solving: no domain specific knowledge (heuristics)
Strips States

- State description as conjunction of positive ground literals, with domain objects (constants)
  \[ D = \{ A, B, C, D \} \] and predicate symbols
  \[ P = \{ \text{ontable}^1, \text{clear}^1, \text{on}^2 \} \]

- \{\text{ontable}(A), \text{ontable}(C), \text{ontable}(D), \text{clear}(A), \text{clear}(B), \text{clear}(D), \text{on}(B, C)\}

- Only conjunction \(\rightarrow\) write set of literals.
- CWA: All relations not given explicitly are assumed to be false.
- A state description corresponds to a state in a state-space model. Typically, only “relevant” aspects of a state are represented.
Strips Goals

- Goals: typically partial state descriptions, e.g., $G = \{\text{on}(B, C), \text{ontable}(C)\}$
- A state $s$ is called goal state iff

$$G \subseteq s$$
A Strips operator is described by **precondition** (PRE) and effect.

Effects are represented as **ADD** and **DEL** lists (more precisely **sets**)

In the most simple case, PRE, ADD, DEL are conjunctions of literals. Variables are assumed to be existentially quantified, literals in the precondition are positive.

**Operator:** put(?x, ?y)

**PRE:** `{ontable(?x), clear(?x), clear(?y)}`

**ADD:** `{on(?x, ?y)}`

**DEL:** `{ontable(?x), clear(?y)}`
Operator Application

The precondition of an operator is matched against the current state.

Convention: Variables with different names are instantiated with different objects.

If \( \text{PRE}_\sigma \subseteq s \), the preconditions “hold” and the operator can be applied.

Substitution \( \sigma \) is performed over the complete operator scheme, that is, the variables occurring in ADD and DEL are instantiated accordingly.

Complication: free variables in ADD/DEL (instantiate again by matching with the state)

Strips is propositional (no terms), that is, variables are always replaced by constant symbols.

Instantiated operators are called **actions**.
### Instantiated Operator

**Operator:** put(A, B)

**PRE:** \{ontable(A), clear(A), clear(B)\}

**ADD:** \{on(A, B)\}

**DEL:** \{ontable(A), clear(B)\}

\[
\sigma = \{ x \leftarrow A, y \leftarrow B \}\]
With $o$ we denote a fully instantiated operator, with 
$\text{PRE}(o)$, $\text{ADD}(o)$, $\text{DEL}(o)$, the according components of $o$.

Operator application is defined as:

$$\text{Res}(o, s) = s \setminus \text{DEL}(o) \cup \text{ADD}(o) \text{ if } \text{PRE}(o) \subseteq s$$

Predicates which only occur in $\text{PRE}$, but never in $\text{ADD/DEL}$ are called statics, they describe “constraining attributes”

Set-theoretical definition of operator application: allows for syntactic calculation of operator effects.
Arc from state $s_i$ to $s_j$ iff $s_j$ can be reached from $s_i$ by performing a single action.
Operator Application Example

\{\text{ontable}(A), \text{clear}(A), \text{clear}(B)\} \subseteq \{\text{ontable}(A), \text{ontable}(C), \text{ontable}(D), \text{clear}(A), \text{clear}(B), \text{clear}(D), \text{on}(B, C)\}
\rightarrow \text{applicable}

\{\text{ontable}(A), \text{ontable}(C), \text{ontable}(D), \text{clear}(A), \text{clear}(B), \text{clear}(D), \text{on}(B, C)\} \setminus \{\text{ontable}(A), \text{clear}(B)\}
= \{ \text{ontable}(C), \text{ontable}(D), \text{clear}(A), \text{clear}(D), \text{on}(B, C)\}

\{\text{ontable}(C), \text{ontable}(D), \text{clear}(D), \text{on}(B, C)\} \cup \{\text{on}(A, B)\}
= \{ \text{ontable}(C), \text{ontable}(D), \text{clear}(A), \text{clear}(D), \text{on}(A, B), \text{on}(B, C)\}

s_{i+1} = \text{Res}(o, s_i)
Planning Domain and Problem

**Planning Domain**: Operator Schemes (for extended formalisms additionally types, axioms, functions, ...)
General description of a domain, such as blocksworld.

**Planning Problem**: a domain, an initial state and a planning goal
The problem, not the domain, constitutes a set of domain objects!

Standardized, extended Strips language for state-based planners: **PDDL** (Planning Domain Definition Language), see, e.g.,

[http://www.dur.ac.uk/d.p.long/competition.html](http://www.dur.ac.uk/d.p.long/competition.html)

PDDL has a Lisp-based syntax. Classically state-based planning is realized in Lisp. Today most planners are in C.
Blocksworld Domain

Operators:

put(?x, ?y)
PRE: \{on\table(?x), \\text{clear}(?x), \\text{clear}(?y)\}
ADD: \{on(?x, ?y)\}
DEL: \{on\table(?x), \\text{clear}(?y)\}

put(?x, ?y)
PRE: \{on(?x, ?z), \\text{clear}(?x), \\text{clear}(?y)\}
ADD: \{on(?x, ?y), \text{clear}(?z)\}
DEL: \{on(?x, ?z), \\text{clear}(?y)\}

puttable(?x)
PRE: \{\text{clear}(?x), on(?x, ?y)\}
ADD: \{on\table(?x), \text{clear}(?y)\}
DEL: \{on(?x, ?y)\}
Blocksworld Problem

Blocksworld Domain +:

Goal: \{on(A, B), on(B, C)\}

Initial State: \{on(D, C), on(C, A), clear(D), clear(B), ontable(A), ontable(B)\}
Equality constraints and conditioned effects

(define (domain blocksworld-adl)
 (:requirements :strips :equality :conditional-effects)
 (:predicates (on ?x ?y)
             (clear ?x)) ; clear(Table) is static
 (:action puton
  :parameters (?x ?y ?z)
  :precondition (and (on ?x ?z) (clear ?x) (clear ?y)
                 (not (= ?y ?z)) (not (= ?x ?z))
                 (not (= ?x ?y)) (not (= ?x Table)))
  :effect
    (and (on ?x ?y) (not (on ?x ?z))
         (when (not (eq ?z Table)) (clear ?z))
         (when (not (eq ?y Table)) (not (clear ?y))))))
)
Planning Algorithms

- Typically: variants of depth-first search
  - Planning domains are usually too complex for applying breadth-first search
  - Breadth-first based strategies: used in universal/conformant planning, based on techniques from symbolic model-checking

- **Forward planning:** Start with the initial state, built a search tree by transforming a given state by action application, check for cycles, backtrack if you reach a dead-end, terminate if a goal state is reached or if all paths are generated (for finite domains)

- **Backward planning:** Start with the top-level goals (a partial description of the goal state), build a search tree by transforming a given state by backwards action application, ..., terminate if the a state is reached which subsumes the initial state or ...
Backward Planning

backward operator application:

\[ \text{Res}^{-1}(o, s) = s \setminus (\text{ADD}(o) \cup (\text{DEL}(o) \cup \text{PRE}(o))) \quad \text{if} \quad \text{ADD}(o) \subseteq s \]

- Also called regression planning
- Advantage: typically smaller search trees (cf. discussion in problem solving)
- Problem: inconsistent states can be produced and can e.g. be detected by including axioms (domain knowledge!)
- Graphplan strategy: build a Planning Graph by forwards search (polynomial effort) and extract the plan from the graph backwards (exponential effort, as usual for planning)
Backward Planning cont.

Inconsistency:
- to fulfill clear(y), y = B
- puttable(x)
- deletes on(x, B)

Will become inconsistent for z = B

CogSysI Lecture 5: Basic AI Planning – p. 141
Evaluating Planners

- **Termination** (critical case: no solution exists)
- **Soundness**: every plan returned is a legal sequence of actions to achieve the goal implies consistency: each intermediate state appearing in the plan is a legal state of the domain
- **Completeness**: the planner finds a solution, if one exists.
- **Optimality**: the returned plans are optimal (shortest) solutions (typically not considered)
- **Expressiveness of the planning language**
- **Efficiency**
- **AIPS planning competition**: every two years current systems compete on hard benchmark problems (logistics, freecell, lift-control, ...)

CogSysI Lecture 5: Basic AI Planning – p. 142
Planning Approaches

- **Total vs. partial order** planners: returned plan is a totally ordered sequence of actions vs. some independent actions are given in parallel; *Graphplan* planners are partial order

- **Linear vs. non-linear** planners: linear planners consider on goal at a time and work with a goal-stack, non-linear planners allow *interleaving of goals* (see below); all modern planners are non-linear
Planning Approaches cont.

- State-space vs. plan-space planners: the first partial order planners (NOAH by Sacerdoti, UCPOP by Weld) were plan-space planners, they work with a least commitment strategy.
  Plan-space: search in the space of partial plans, start with a plan which only contains initial state and top-level goals.

- Hierarchical planning (see AND-OR Trees in problem solving); not to confuse with partial order planning!

- A successful forward-planning system, which estimates heuristic values for the distance of a state from the goal from the problem definition is HSP.
Incompleteness of Linear P.

The Sussman Anomaly

Initial State

Goal: on(A, B) and on(B, C)

on(A, B)

on(B, C)

on(B, C)
Linear planning corresponds to dealing with goals organized in a stack:

\[on(A, B), on(B, C)\]

try to satisfy goal \(on(A, B)\)

solve sub-goals \([clear(A), clear(B)]\)

all sub-goals hold after \(puttable(C)\)

apply \(put(A, B)\)

goal \(on(A, B)\) is reached

try to satisfy goal \(on(B, C)\).

\(^a\)We ignore the additional subgoal \(ontable(A)\) rsp. \(on(A, z)\) here.
Interleaving of Goals

Non-linear planning allows that a sequence of planning steps dealing with one goal is interrupted to deal with another goal.

For the Sussman Anomaly, that means that after block $C$ is put on the table pursuing goal $on(A, B)$, the planner switches to the goal $on(B, C)$.

Non-linear planning corresponds to dealing with goals organized in a set.

The correct sequence of goals might not be found immediately but involve backtracking.
{on(A, B), on(B, C)}

try to satisfy goal on(A, B)
  {clear(A), clear(B), on(A, B), on(B, C)}
  clear(A) and clear(B) hold after puttable(C)

try to satisfy goal on(B, C)
apply put(B, C)

try to satisfy goal on(A, B)
apply put(A, B).
Rocket Domain

(Veloso)

- Objects: $n$ boxes, Positions (Earth, Moon), one Rocket
- Operators: load/load a box, move the Rocket (oneway: only from earth to moon, no way back!)
- Linear planning: to reach the goal, that Box1 is on the Moon, load it, shoot the Rocket, unload is, now no other Box can be transported!
Deductive Planning

- Deductive inference can be used to solve planning problems.

- Introduce a **situation variable** to store the partial plans:
  \[ s_{i+1} = \text{put}(A, B, s_i), \ldots \quad s_2 = \text{puttable}(A, s_1) \]
  \[ s = \text{put}(A, B, \text{puttable}(A, [\text{on}(A, C), \text{clear}(A)\ldots])) \]

- 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  \( \forall S[on(a, table, S) \rightarrow on(a, b, put(a, b, S)))] \equiv \) (axiom for put-operator)
    \( \neg on(a, table, S) \lor on(a, b, put(a, b, S)) \)  (clausal form)

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

1. \( \neg on(a, b, S_F) \)  (Negation of the theorem)
2. \( \neg on(a, table, S) \lor on(a, b, put(a, b, S)) \)  (A2)
3. \( \neg on(a, table, S) \)  (Resolve 1, 2)
   answer(put(a, b, S))
4. \( on(a, table, s_1) \)  (A1)
5. contradiction  (Resolve 3, 4) \( \leftrightarrow \) 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 $\iff$ 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:
  \[ \forall S[\text{on}(Y, Z, S) \rightarrow \text{on}(Y, Z, \text{put}(X, Y, S))] \] \[ \text{on}(Y, Z, \text{put}(X, Y, S)) \leftarrow \text{on}(Y, Z, S) \]

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:

\[
\text{on}(X, Y, \text{put}(X, Y, S)) \leftarrow \quad \text{clear}(X, S) \land \text{clear}(Y, S)
\]
\[
\text{clear}(Z, \text{put}(X, Y, S)) \leftarrow \quad \text{on}(X, Z, S) \land \text{clear}(X, S) \land \text{clear}(Y, S)
\]
\[
\text{clear}(Y, \text{puttable}(X, S)) \leftarrow \quad \text{on}(X, Y, S) \land \text{clear}(X, S)
\]
\[
\text{ontable}(X, \text{puttable}(X, S)) \leftarrow \quad \text{clear}(X, S)
\]

Frame Axioms:

\[
\text{clear}(X, \text{put}(X, Y, S)) \leftarrow \quad \text{clear}(X, S) \land \text{clear}(Y, S)
\]
\[
\text{clear}(Z, \text{put}(X, Y, S)) \leftarrow \quad \text{clear}(X, S) \land \text{clear}(Y, S) \land \text{clear}(Z, S)
\]
\[
\text{ontable}(Y, \text{put}(X, Y, S)) \leftarrow \quad \text{clear}(X, S) \land \text{clear}(Y, S) \land \text{ontable}(Y, S)
\]
\[
\text{ontable}(Z, \text{put}(X, Y, S)) \leftarrow \quad \text{clear}(X, S) \land \text{clear}(Y, S) \land \text{ontable}(Z, S)
\]
\[
\text{on}(Y, Z, \text{put}(X, Y, S)) \leftarrow \quad \text{clear}(X, S) \land \text{clear}(Y, S) \land \text{on}(Y, Z, S)
\]
\[
\text{on}(W, Z, \text{put}(X, Y, S)) \leftarrow \quad \text{clear}(X, S) \land \text{clear}(Y, S) \land \text{on}(W, Z, S)
\]
Frame Axioms cont.:

- \text{clear}(Z, \text{puttable}(X, S)) \iff \text{clear}(X, S) \land \text{clear}(Z, S)
- \text{ontable}(Z, \text{puttable}(X, S)) \iff \text{clear}(X, S) \land \text{ontable}(Z, S)
- \text{on}(Y, Z, \text{puttable}(X, S)) \iff \text{clear}(X, S) \land \text{on}(Y, Z, S)
- \text{clear}(Z, \text{puttable}(X, S)) \iff \text{on}(Y, X, S) \land \text{clear}(Y, S) \land \text{clear}(Z, S)
- \text{ontable}(Z, \text{puttable}(X, S)) \iff \text{on}(Y, X, S) \land \text{clear}(Y, S) \land \text{ontable}(Z, S)
- \text{on}(W, Z, \text{puttable}(X, S)) \iff \text{on}(Y, X, S) \land \text{clear}(Y, S) \land \text{on}(W, Z, S)

Facts (Initial State):

- \text{on}(d, c, s_1)
- \text{clear}(d, s_1)
- \text{ontable}(a, s_1)
- \text{on}(c, a, s_1)
- \text{clear}(b, s_1)
- \text{ontable}(b, s_1)

Theorem (Goal):

- \text{on}(a, b, S) \land \text{on}(b, c, S)
Further Topics

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

CogSysI Lecture 5: Basic AI Planning – p. 155
Question: How many AI people does it take to change a lightbulb?  
Answer: At least 67. 
6th part of the solution: **The Planning Group (4)**

- One to define STRIPS-style operators for lightbulb changing
- One to show that linear planning is not adequate
- One to show that nonlinear planning is adequate
- One to show that people don’t plan; they simply react to lightbulbs

(“Artificial Intelligence”, Rich & Knight)