Module Title: PLAN
Exam Diet (Dec/April/Aug): Dec 2007
Brief notes on answers:

1. (a) **AI Planning**: Domain-specific planning uses specific representations and techniques adapted to each specific problem. Briefly describe some of the problems with domain-specific planning that justify the study of domain-independent planning. [*2 marks*]

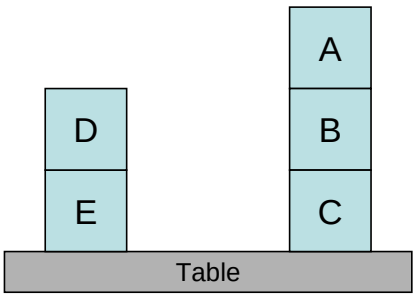
Answer:

- commonalities to all forms of planning are not addressed
- more costly to address each domain anew
- no good for study and design of autonomous intelligent machines

(b) **Situation Calculus**: A problem that was encountered by planners early on is the so-called frame problem. Consider a theory in the situation calculus describing the Blocks World in which there is just a single action *move* defined by the applicability axiom Δ_a and effect axioms Δ_e:

$$applicable(move(x,y,z),s) \leftrightarrow clear(x,s) \wedge clear(z,s) \wedge on(x,y,s)$$
$$applicable(move(x,y,z),s) \rightarrow on(x,z,result(move(x,y,z),s))$$
$$applicable(move(x,y,z),s) \rightarrow clear(y,result(move(x,y,z),s))$$

Note that there are no frame axioms Δ_f in this theory. Suppose the initial state depicted below is described by some formula Σ_{s_i}.



Give a logical formula F that does not follow from this theory due to the lack of frame axioms, i.e. give F such that:

$$\Sigma_{s_i} \wedge \Delta_a \wedge \Delta_e \not\models F, \text{ but}$$
$$\Sigma_{s_i} \wedge \Delta_a \wedge \Delta_e \wedge \Delta_f \models F.$$

[*2 marks*]

Answer:
For example: $on(B,C,result(move(A,B,D),s_i))$, i.e. B is still on C after moving A from B onto D.

(c) **State-Space Search**: One of the earliest AI planners was the STRIPS planner. While the planning algorithm turned out to be incomplete, the STRIPS representation for planning operators and actions is still used. Formally define STRIPS planning operators and actions. [*4 marks*]

Answer:

- A planning operator in a STRIPS planning domain is a triple $o = (name(o), precond(o), effects(o))$ where:
  - the name of the operator $name(o)$ is a syntactic expression of the form $n(x_1, \ldots, x_k)$ where n is a (unique) symbol and $x_1, \ldots, x_k$ are all the variables that appear in o, and
  - the preconditions $precond(o)$ and the effects $effects(o)$ of the operator are sets of literals.
- An action in a STRIPS planning domain is a ground instance of a planning operator.

(d) **Plan-Space Search**: In plan-space search the nodes in the search space are partial plans that the planner has to refine into a solution plan. Formally define the components that make up a partial plan. [*4 marks*]

Answer:

A partial plan is a tuple $\pi = (A, \prec, B, L)$, where:

- $A = \{a_1, \ldots, a_k\}$ is a set of partially instantiated planning operators;
- $\prec$ is a set of ordering constraints on A of the form $(a_i \prec a_j)$;
- B is a set of binding constraints on the variables of actions in A of the form $x = y$, $x \neq y$, or $x \in D_x$;
- L is a set of causal links of the form $\langle a_i \xrightarrow{p} a_j \rangle$ such that:
  - a_i and a_j are actions in A;
  - the constraint $(a_i \prec a_j)$ is in $\prec$;
  - proposition p is an effect of a_i and a precondition of a_j; and
  - the binding constraints for variables in a_i and a_j appearing in p are in B.

(e) **Graphplan**: The planning graph developed by Graphplan consists of action and proposition layers. Under what condition is a set of independent actions π (in one action layer) applicable to a state s? What is the result, $\gamma(s, \pi)$, of applying this set of independent actions? [*4 marks*]

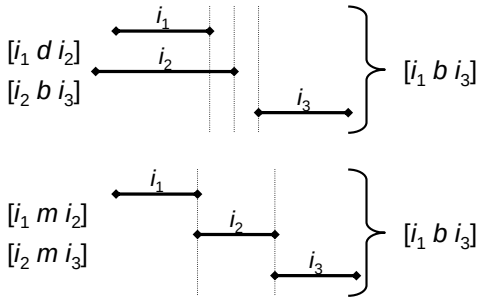
Answer:

- A set π of independent actions is applicable in a state s if and only if $\bigcup_{a \in \pi} \text{precond}(a) \subseteq s$.
- The result of applying the set π in s is defined as: $\gamma(s, \pi) = (s - \text{effects}^-(\pi)) \cup \text{effects}^+(\pi)$, where:
  - $\text{effects}^+ = \bigcup_{a \in \pi} \text{effects}^+(a)$, and
  - $\text{effects}^- = \bigcup_{a \in \pi} \text{effects}^-(a)$.

(f) **Temporal Planning**: In the Interval Algebra there are 13 different primitive relations that can relate two intervals. These can be combined using the composition operator ($\bullet$). For example, $(b \bullet b)$ can be simplified to b. What relations are expressed by $(d \bullet b)$ and $(m \bullet m)$? Explain your answers. [*4 marks*]

Answer:

- $i_1(d \bullet b)i_3$ can be simplified to $i_1 b i_3$ (d = during, b = before)
- $i_1(m \bullet m)i_3$ can be simplified to $i_1 b i_3$ (m = meets)

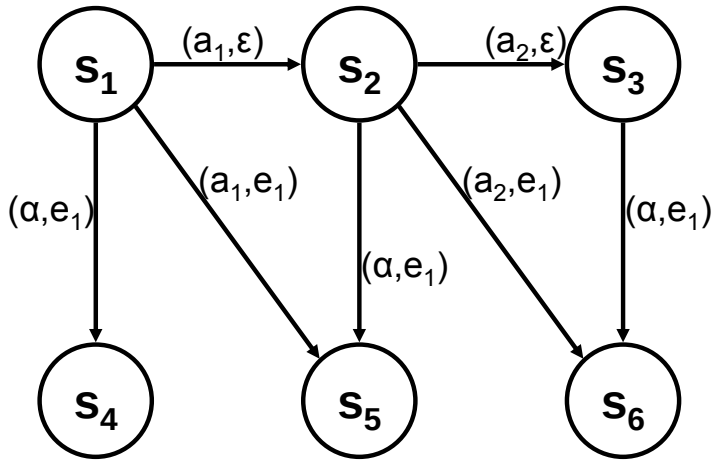


(g) **General**: The Situation Calculus was the first approach to AI planning. Discuss the fundamental differences between this approach and other approaches attempted since. [*5 marks*]

Answer:

- Situation Calculus uses existing algorithms (theorem prover) to solve planning problems, only needs to solve the problem of how to represent world states and actions in logic; other approaches develop specific algorithms
- Situation Calculus has to address the frame problem explicitly because of its choice of representation; other approaches address it implicitly by building the solution into the algorithm
- first-order logic is more expressive than the representations used by other approaches

2. (a) **AI Planning**: The following graph represents a state-transition system $\Sigma = (S, A, E, \gamma)$ where α denotes no action (no-op) and ε denotes no event taking place. Define the 4 components of this state-transition system.



[*4 marks*]

Answer:

- $S = \{s_1, s_2, s_3, s_4, s_5, s_6\}$
- $A = \{a_1, a_2\}$
- $E = \{e_1\}$
-

γ	(a_1, ε)	(a_2, ε)	(α, e_1)	(a_1, e_1)	(a_2, e_1)
s_1	$\{s_2\}$	$\{\}$	$\{s_4\}$	$\{s_5\}$	$\{\}$
s_2	$\{\}$	$\{s_3\}$	$\{s_5\}$	$\{\}$	$\{s_6\}$
s_3	$\{\}$	$\{\}$	$\{s_6\}$	$\{\}$	$\{\}$
s_4	$\{\}$	$\{\}$	$\{\}$	$\{\}$	$\{\}$
s_5	$\{\}$	$\{\}$	$\{\}$	$\{\}$	$\{\}$
s_6	$\{\}$	$\{\}$	$\{\}$	$\{\}$	$\{\}$

(b) **Situation Calculus**: A problem that was encountered by planners early on is the so-called frame problem. Consider a theory in the situation calculus describing the Blocks World in which there is just a single action *move* defined by the applicability axiom Δ_a and effect axioms Δ_e:

$$applicable(move(x, y, z), s) \leftrightarrow clear(x, s) \wedge clear(z, s) \wedge on(x, y, s)$$
$$applicable(move(x, y, z), s) \rightarrow on(x, z, result(move(x, y, z), s))$$
$$applicable(move(x, y, z), s) \rightarrow clear(y, result(move(x, y, z), s))$$

What frame axioms Δ_f need to be added to this theory to allow a theorem prover to draw all the desired conclusions? [*3 marks*]

Answer:

Δ_f:

- $\forall v, w, x, y, z, s : on(v, w, s) \wedge v \neq x \rightarrow on(v, w, result(move(x, y, z), s))$
- $\forall v, x, y, z, s : clear(v, s) \wedge v \neq z \rightarrow clear(v, result(move(x, y, z), s))$

(c) **State-Space Search**: The planning problem can be seen as a search problem. In the state-space search approach, what is the search space? What do the nodes in this search space represent? What do the arcs represent? What does a path

in this search space correspond to? [*2 marks*]
Answer:

- the search space is a subset of state space
- the nodes correspond to world states
- the arcs correspond to state transitions
- a path in the search space corresponds to a plan

(d) **Plan-Space Search**: In plan-space search the nodes in the search space are partial plans which contain explicit causal links between the different actions in the plan. A potential flaw in a partial plan could be that effect q of action a_t threatens a causal link $\langle a_i \xrightarrow{p} a_j \rangle$. How can such a flaw be resolved? Under what conditions will the different resolvers be applicable? [*5 marks*]
Answer:

- order action before threatened link:
  - if $(a_t = a_i)$ or $(a_j \prec a_t)$ then not a resolver
  - otherwise: adding $(a_t \prec a_i)$ is a resolver
- order threatened link before action:
  - if $(a_t = a_i)$ or $(a_t \prec a_i)$ then not a resolver
  - otherwise: adding $(a_j \prec a_t)$ is a resolver
- extend variable bindings such that unification fails:
  - for every variable v in p or q
    if $v \neq \sigma(v)$ is consistent with B then
    adding $v \neq \sigma(v)$ is a resolver

(e) **Hierarchical Planning**: Consider an STN planning domain that contains the following three methods:

- take-and-put$(c, k, l, p_o, p_d, x_o, x_d)$
  - task: move-topmost(p_o, p_d)
  - preconditions: top(c, p_o), on(c, x_o), attached(p_o, l), belong(k, l), attached(p_d, l), top(x_d, p_d)
  - subtasks: $\langle$take(k, l, c, x_o, p_o),put$(k, l, c, x_d, p_d)\rangle$
- recursive-move(p_o, p_d, c, x_o)
  - task: move-stack(p_o, p_d)
  - preconditions: top(c, p_o), on(c, x_o)
  - subtasks: $\langle$move-topmost(p_o, p_d), move-stack$(p_o, p_d)\rangle$
- no-move(p_o, p_d)
  - task: move-stack(p_o, p_d)
  - preconditions: top$(\texttt{pallet}, p_o)$
  - subtasks: $\langle\rangle$

Let the initial state s_i be as follows: belong(`crane`,`loc`), attached(`p1`,`loc`), attached(`p2`,`loc`), attached(`p3`,`loc`), top(`c1`,`p1`), top(`pallet`,`p2`), top(`pallet`,`p3`), on(`c1`,`c2`), on(`c2`,`c3`), on(`c3`,`pallet`). Finally, let the current task be:

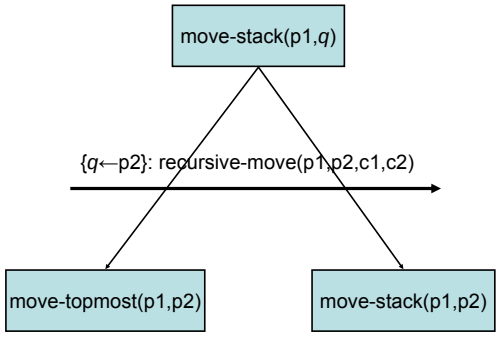
move-stack$(\texttt{p1}, q)$

where q is a variable. Which of the above methods are applicable and which are relevant here? Justify your answer. Show how an applicable and relevant method can be used to decompose the above task. [*7 marks*]

Answer:

- take-and-put(c1,crane,loc,p1, q,c2, x_d):
  - applicable because $\text{precond}^+(m) \subseteq s_i$ and $\text{precond}^-(m) \cap s_i = \emptyset$
  - not relevant because move-topmost(p1, q) $\neq$ move-stack(p1, q)
- recursive-move(p1, q,c1,c2)
  - applicable because $\text{precond}^+(m) \subseteq s_i$ and $\text{precond}^-(m) \cap s_i = \emptyset$
  - relevant because $\text{task}(m) =$ move-stack(p1, q)
- no-move(p2, p_d) and no-move(p3, p_d)
  - applicable because $\text{precond}^+(m) \subseteq s_i$ and $\text{precond}^-(m) \cap s_i = \emptyset$
  - not relevant because move-stack(p2, p_d) $\neq$ move-stack(p1, q) and move-stack(p3, p_d) $\neq$ move-stack(p1, q)

Decomposition:



(f) **Temporal Planning**: Temporal Databases can be augmented with domain axioms to express facts that are always true. Write domain axioms to express the following facts:

- no object can be in two places at the same time
- at any given time, each location can be occupied by only one robot

[*4 marks*]

Answer:

- no object can be in two places at the same time:
  $\{\text{at}(o,l)@[t_b,t_e[,\text{at}(o,l')@[t'_b,t'_e[\} \rightarrow (l = l') \vee (t_e \leq t'_b) \vee (t'_e \leq t_b)$
- every location can be occupied by one robot only:
  $\{\text{at}(r,l)@[t_b,t_e[,\text{at}(r',l)@[t'_b,t'_e[\} \rightarrow (r = r') \vee (t_e \leq t'_b) \vee (t'_e \leq t_b)$

3. (a) **AI Planning**: A number of restricting assumptions are often made in AI planning to simplify the problem. One of these assumptions is that an objective is given as a *restricted goal*. What does this assumption mean? In what type of domain do we want to drop this assumption? What issues will arise as a result? [*3 marks*]

Answer:

Restricted goals: the planner handles only goals that are given as an explicit goal state s_g or set of goal states S_g.

- Must be relaxed if: need to handle constraints on states and plans, utility functions, or tasks.
- Issues: representation and reasoning over constraints, utility, and tasks.

(b) **Situation Calculus**: Consider a theory in the situation calculus describing the Blocks World in which there is just a single action *move* defined by the applicability axiom Δ_a and effect axioms Δ_e:

$$\begin{gathered} applicable(move(x,y,z),s) \leftrightarrow clear(x,s) \wedge clear(z,s) \wedge on(x,y,s) \\ applicable(move(x,y,z),s) \rightarrow on(x,z,result(move(x,y,z),s)) \\ applicable(move(x,y,z),s) \rightarrow clear(y,result(move(x,y,z),s)) \end{gathered}$$

Now suppose we want to extend this domain with a new fluent $colour(v,w)$ to represent that the colour of block v is w, and a new action $paint(x,y)$ representing the action of painting block x in colour y. How many additional frame axioms will be necessary as a result of this extension? Justify your answer. [*3 marks*]

Answer:

Four new frame axioms are required. In general, we need (nr. of actions) times (nr. of fluents) frame axioms. Thus, the original domain required two frame axioms, and the extended domain requires six frame axioms, i.e. four more.

(c) **State-Space Search**: To show that an algorithm is sound and complete, it is necessary to formally define the problem that the algorithm solves. Give the formal definition of the planning problem for the STRIPS representation. [*3 marks*]

Answer:

A STRIPS planning problem is a triple $\mathcal{P} = (\Sigma, s_i, g)$ where:

- $\Sigma = (S, A, \gamma)$ is a STRIPS planning domain on some first-order language $\mathcal{L}$
- $s_i \in S$ is the initial state
- g is a set of ground literals describing the goal such that the set of goal states is: $S_g = \{s \in S | s \ satisfies \ g\}$

(d) **Plan-Space Search**: In plan-space search the nodes in the search space are partial plans which contain explicit causal links between the different actions in the plan. Thus, planners that perform a plan-space search must find the new threats in a partial plan when the plan is refined. For which types of refinement do the threats need to be detected? For each of these describe in pseudo-code how the detection is performed. [*6 marks*]

Answer:

- in the initial plan π_0: no threats
- when adding an action a_{new} to $\pi = (A, \prec, B, L)$:

for every causal link $\langle a_i \xrightarrow{p} a_j \rangle \in L$
if $(a_{new} \prec a_i)$ or $(a_j \prec a_{new})$ then next link
else for every effect q of a_{new}
if $(\exists \sigma : \sigma(p) = \sigma(\neg q))$ then q of a_{new} threatens $\langle a_i \xrightarrow{p} a_j \rangle$

- when adding a causal link $\langle a_i \xrightarrow{p} a_j \rangle$ to $\pi = (A, \prec, B, L)$:

for every action $a_{old} \in A$
if $(a_{old} \prec a_i)$ or $(a_j = a_{old})$ or $(a_j \prec a_{old})$ then next action
else for every effect q of a_{old}
if $(\exists \sigma : \sigma(p) = \sigma(\neg q))$ then q of a_{old} threatens $\langle a_i \xrightarrow{p} a_j \rangle$

(e) **Hierarchical Planning**: Consider the following hierarchical task network $w = (U, C)$, where:

- $U = \{t_1 = \text{move-stack}(\texttt{p1}, q)\}$ and
- $C = \emptyset$

Give the result of applying the decomposition function δ to t_1, using the method recursive-move(p_o, p_d, c, x_o) defined as follows:

- task: move-stack(p_o, p_d)
- network:
  - subtasks: $\{t_1 =$move-topmost$(p_o, p_d), t_2 =$move-stack$(p_o, p_d)\}$
  - constraints: $\{t_1 \prec t_2,$before$(\{t_1\},$top$(c, p_o)),$before$(\{t_1\},$on$(c, x_o))\}$

Give the result of using the method take-and-put$(c, k, l, p_o, p_d, x_o, x_d)$ to refine a task in the result of the previous step, i.e. in:
$\delta(w, t_1,$recursive-move$(p_o, p_d, c, x_o), \{p_o \leftarrow \texttt{p1}, p_d \leftarrow q\})$
where take-and-put$(c, k, l, p_o, p_d, x_o, x_d)$ is given as:

- task: move-topmost(p_o, p_d)
- network:
  - subtasks: $\{t_1 =$take$(k, l, c, x_o, p_o), t_2 =put(k, l, c, x_d, p_d)\}$
  - constraints: $\{t_1 \prec t_2,$ before$(\{t_1\},$top$(c, p_o)),$ before$(\{t_1\},$on$(c, x_o)),$ before$(\{t_1\},$attached$(p_o, l)),$ before$(\{t_1\},$belong$(k, l)),$ before$(\{t_2\},$attached$(p_d, l)),$ before$(\{t_2\},$top$(x_d, p_d))\}$

[*6 marks*]

Answer:
$\delta(w, t_1,$recursive-move$(p_o, p_d, c, x_o), \{p_o \leftarrow \texttt{p1}, p_d \leftarrow q\}) = w' = (U', C')$ with:

- $U' = \{t_2 = \text{move-topmost}(\texttt{p1}, q), t_3 = \text{move-stack}(\texttt{p1}, q)\}$ and
- $C' = \{t_2 \prec t_3,$before$(\{t_2\},$top$(c,\texttt{p1})),$before$(t_2,$on$(c, xo))\})$

$\delta(w', t_2,$take-and-put$(c, k, l, p_o, p_d, x_o, x_d), \{p_o \leftarrow \texttt{p1}, p_d \leftarrow q\}) = (U'', C'')$ with:

- $U'' = \{t_3 = \text{move-stack}(\texttt{p1}, q), t_4 =$take$(k, l, c, x_o,\texttt{p1}), t_5 =put(k, l, c, x_d, q)\}$ and
- $C'' = \{t_4 \prec t_3, t_5 \prec t_3,$ before$(\{t_4, t_5\},$top$(c,\texttt{p1})),$ before$(\{t_4, t_5\},$on$(c, x_o))\} \cup \{t_4 \prec t_5,$ before$(\{t_4\},$top$(c,\texttt{p1})),$ before$(\{t_4\},$on$(c, x_o)),$ before$(\{t_4\},$attached$(\texttt{p1}, l)),$ before$(\{t_4\},$belong$(k, l)),$ before$(\{t_5\},$attached$(q, l)),$ before$(\{t_5\},$top$(x_d, q))\}$

(f) **Graphplan**: The planning graph developed by Graphplan can be described as an AND/OR-graph. Which nodes are OR-nodes and which nodes are AND-nodes? What nodes do the k-connectors from the AND-nodes connect to? Why does Graphplan not simply follow the (provably optimal) AO* algorithm to search this AND/OR-graph? [*4 marks*]

Answer:

- OR-nodes:
  - nodes in proposition layers
  - links to actions that support the propositions
- AND-nodes:
  - nodes in action layers
  - k-connectors to all preconditions of the action

AO* is not the best algorithm to search this AND/OR-graph because it does not exploit the layered structure of the planning graph.