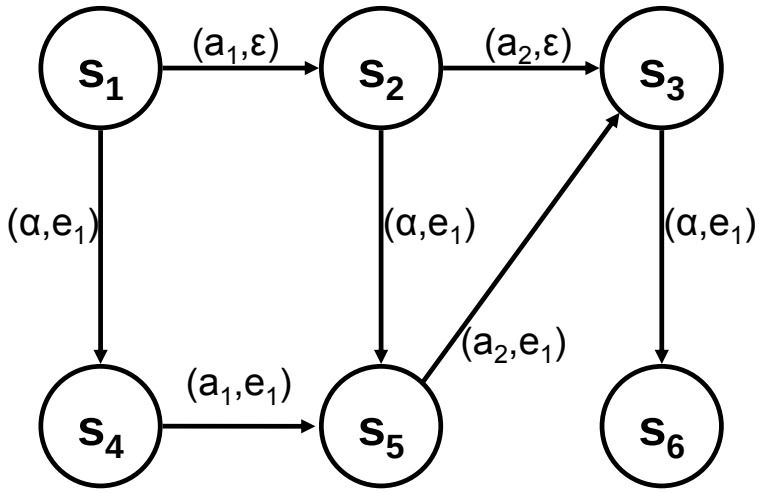
Module Title: Automated Planning
Exam Diet (Dec/April/Aug): December 2004
Brief notes on answers:

1. (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 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_2	$\{\}$	$\{s_3\}$	$\{s_5\}$	$\{\}$	$\{\}$
s_3	$\{\}$	$\{\}$	$\{s_6\}$	$\{\}$	$\{\}$
s_4	$\{\}$	$\{\}$	$\{\}$	$\{s_5\}$	$\{\}$
s_5	$\{\}$	$\{\}$	$\{\}$	$\{\}$	$\{s_3\}$
s_6	$\{\}$	$\{\}$	$\{\}$	$\{\}$	$\{\}$

(b) **Situation Calculus**: In the situation calculus a planning problem is represented as a set of sentences in first-order logic. Show how the initial state given in the planning problem in appendix 1 can be represented in the situation calculus. *[2 marks]*

Answer: initial state s_i: $P(A, s_i) \wedge P(B, s_i) \wedge Q(A, A, s_i) \wedge Q(B, B, s_i)$

(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 are refined until a solution plan is found. Partial plans that are not solutions may contain threats. Construct an example that explains the concept of a threat using the operators given in the planning domain defined in appendix 1. *[5 marks]*

Answer:

- partially instantiated actions in the plan:
  - `(op1 ?x1 ?y1 C)`
  - `(op2 ?x2 C ?z2)`
  - `(op2 ?x3 ?y3 C)`
- with `(op1 ?x1 ?y1 C)` before `(op2 ?x2 C ?z2)`
- and causal link `(op1 ?x1 ?y1 C)` $\overset{P(C)}{\rightarrow}$ `(op2 ?x2 C ?z2)`

Then the effect `(not (P C))` of `(op2 ?x3 ?y3 C)` constitutes a threat to the causal link. That is, if this operator was executed between the two connected by the causal link, it would undo the condition protected by the causal link.

(e) **Representation**: The STRIPS representation, the propositional representation, and the state-variable representation can be shown to be equally expressive. This is done by providing methods that can translate back and forth between the different representations. Translate the following operator in the state-variable representation into the STRIPS representation.

- name: $svop(x, y, z)$
- precondition: $Sv1(x, y) = z$
- effects: $Sv1(x, y) \leftarrow y, Sv2(x) \leftarrow z$

[3 marks]

Answer:

- name: $svop(x, y, z, v)$
- preconditions: $Sv1(x, y, z), Sv2(x, v)$
- effects: $Sv1(x, y, y), \neg Sv1(x, y, z), Sv2(x, z), \neg Sv2(x, v)$

(f) **Graphplan**: The planning graph developed by the Graphplan planner quickly grows from layer to layer until the proposition layers contain all those propositions that are eventually achievable. However, propositions that occur in the

same layer may not be achievable simultaneously. More specifically, two mutually exclusive propositions cannot be achieved in the same proposition layer. Define this so-called mutex relation between propositions. [*3 marks*]

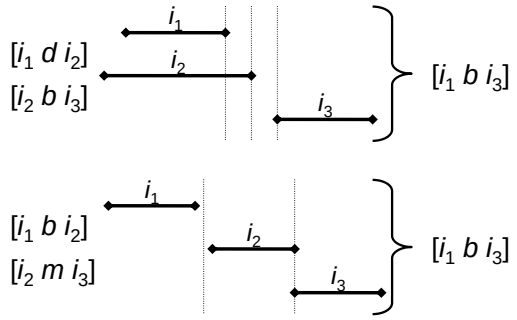
Answer:
Two propositions p and q in the jth proposition layer P_j are mutex (mutually exclusive) if:

- every action in the preceding action layer A_j that has p as a positive effect (incl. no-op actions) is mutex with every action in A_j that has q as a positive effect, and
- there is no single action in A_j that has both, p and q, as positive effects.

(g) **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 $(b \bullet m)$? 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)
- $i_1(b \bullet m)i_3$ can be simplified to $i_1 b i_3$ (m = meets)



2. (a) **AI Planning**: Planning is an area that has been researched in Artificial Intelligence for a long time now. What do we mean by "AI Planning"? What is being studied in this field? [*2 marks*]

Answer:

- "planning":
  - explicit deliberation process that chooses and organizes actions by anticipating their outcomes (reasoning about actions)
  - aims at achieving some pre-stated objectives
- "AI planning":
  - computational study of this deliberation process

(b) **Situation Calculus**: In the situation calculus actions are denoted by function terms. The definition of these functions is given by a set of axioms, namely the applicability axioms, the effect axioms, and the frame axioms. Define these axioms for the action `op1` defined as part of the domain in appendix 1. [*6 marks*]

Answer:

- applicability:
  $\forall x, y, z, s : applicable(op1(x, y, z), s) \leftrightarrow P(x, s) \wedge Q(x, y, s)$
- effect axiom:
  $\forall x, y, z, s : applicable(op1(x, y, z), s) \rightarrow P(z, result(op1(x, y, z), s))$
- effect axiom:
  $\forall x, y, z, s : applicable(op1(x, y, z), s) \rightarrow Q(y, z, result(op1(x, y, z), s))$
- frame axiom:
  $\forall x, x', y, z, s : P(x', s) \rightarrow P(x', result(op1(x, y, z), s))$
- frame axiom:
  $\forall x, x', y, y', z, s : Q(x', y', s) \wedge (x \neq x' \vee y \neq y') \rightarrow Q(x', y', result(op1(x, y, z), s))$

(c) **State-Space Search**: In the STRIPS representation a world state s is represented by a set of ground atoms. When is an action a applicable in s? Define the state transition function γ for such an action. [*3 marks*]

Answer:

- Let L be a set of literals.
  - L^+ is the set of atoms that are positive literals in L and
  - L^- is the set of all atoms whose negations are in L.
- Let a be an action and s a state. Then a is applicable in s iff:
  - $precond^+(a) \subseteq s$; and
  - $precond^-(a) \cap s = \emptyset$.
- The state transition function γ for an applicable action a in state s is defined as:
  - $\gamma(s, a) = (s effects^-(a)) \cup effects^+(a)$

(d) **Plan-Space Search**: An alternative to state-space search is plan-space search in which nodes in the search space are partial plans and arcs correspond to plan

refinements. In plan-space search, what does the initial search state, i.e. the root node of the search tree, look like for a planning problem $P = (O, s_i, g)$? [*3 marks*]

Answer:

- represent initial state s_i and goal g as dummy actions
  - action *init*: no preconditions, initial state s_i as effects
  - action *goal*: goal conditions g as preconditions, no effects
- initial search state: empty plan $\pi_0 = (\{init, goal\}, (init \prec goal), \{\}, \{\})$:
  - two dummy actions *init* and *goal*;
  - one ordering constraint: *init* before *goal*;
  - no variable bindings; and
  - no causal links.

(e) **Task Networks**: When planning with task networks, the aim is not to achieve some goal state, but to accomplish a given set of tasks. Describe, in pseudo code, the ground partial-order forward decomposition algorithm (Ground-PFD) that takes an initial state s, an initial task network $w = (U, E)$, a set of primitive operators O, and a set of methods M. [*7 marks*]

function Ground-PFD(s, w, O, M)
    if $w.U = \emptyset$ return $\langle\rangle$
    $task \leftarrow \{t \in U | t$ has no predecessors in $w.E\}$.chooseOne()
    if $task$.isPrimitive() then
        $actions \leftarrow \{(a, \sigma) | a = \sigma(task)$ and a applicable in $s\}$
        if $actions$.isEmpty() then return failure
        $(a, \sigma) \leftarrow actions$.chooseOne()
        $plan \leftarrow$ Ground-PFD($\gamma(s, a), \sigma(w - \{task\}), O, M$)
        if $plan =$ failure then return failure
        else return $\langle a \rangle \bullet plan$
    else
        $methods \leftarrow \{(m, \sigma) | \sigma(m)$ is relevant for $task$ and m is applicable in $s\}$
        if $methods$.isEmpty() then return failure
        $(m, \sigma) \leftarrow methods$.chooseOne()
        return Ground-PFD($s, \delta(w, task, m, \sigma), O, M$)

(f) **SAT Planning**: One approach to solving a bounded planning problem is to transform it into a satisfiability problem and use a SAT solver to find a plan. How are the actions in a planning domain encoded in this approach? When the SAT solver has finished, how can we extract the solution plan from the SAT solver's output? [*4 marks*]

Answer: Let A be the set of action propositions and $a \in A$.

- encoding actions:
  for $0 \leq i \leq n-1 : a_i \Rightarrow (\bigwedge_{f \in precond(a)} f_i \wedge \bigwedge_{f \in effects^+(a)} f_{i+1} \wedge \bigwedge_{f \in effects^-(a)} \neg f_{i+1})$
  where n is the size of the bounded problem
- if the SAT solver finds a_i to be true then a is the ith action in the plan

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 the state-transition system Σ is *finite*. 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:
Finite Σ: system Σ has a finite set of states.

- Must be relaxed if: need to describe actions that construct or bring new objects into the world; or need to handle numerical state variables.
- Issues: decidability and termination of planning.

(b) **Situation Calculus**: The first approach to planning was to use theorem provers to solve planning problems. For this, planning problems were written as first-order theories in the situation calculus. However, this gave rise to the frame problem, i.e. the need to explicitly represent frame axioms in the theory. What are the three principal ways in which the frame problem as it appears in the situation calculus has been addressed in AI planning? [*3 marks*]

Answer:

- use a different style of representation in first-order logic (same formalism)
- use a different logical formalism, e.g. non-monotonic logic
- write a procedure that generates the right conclusions and forget about the frame problem

(c) **State-Space Search**: Consider the planning domain and problem defined in appendix 1. The first step in the ground backward state-space search algorithm computes the set of relevant actions. What are the relevant actions for the given goal? [*7 marks*]

Answer:

- The actions that can achieve the first goal `(P B)` are of the form `(op1 ?x ?y B)`, which have as negative effect `(not (Q ?x ?y))`, which may interfere with the other goal `(not (Q B C))`. Hence, the (fully ground) actions that are relevant for the first goal are:
  - `(op1 A A B)`
  - `(op1 A B B)`
  - `(op1 A C B)`
  - `(op1 B A B)`
  - `(op1 C A B)`
  - `(op1 C B B)`
  - `(op1 C C B)`

  `(op1 B C B)` would interfere with the second goal and thus, is not relevant. `(op1 B B B)` has inconsistent effects. `op2` is not relevant for `(P B)` as it cannot achieve this goal.

- The actions that can achieve the second goal `(Q B C)` are of the form `(op1 ?x B C)` or `(op2 ?x B C)`. None of the negative effects of these actions (`(not (Q ?x B))` and `(not (P C))` respectively) interfere with other goals. Hence, the (fully ground) actions that are relevant for the second goal are:
  - `(op1 A B C)`
  - `(op1 B B C)`
  - `(op1 C B C)`
  - `(op2 A B C)`
  - `(op2 B B C)`
  - `(op2 C B C)`

(d) **Plan-Space Search**: The basic PSP procedure works by refining a partial plan π while maintaining the consistency of the ordering and variable binding constraints. The plan π is returned as a solution when it has no more flaws. What types of flaw are there? [*2 marks*]

Answer: A flaw in a plan $\pi = (A, \prec, B, L)$ is either:

- an unsatisfied sub-goal, i.e. a precondition of an action in A without a causal link that supports it; or
- a threat, i.e. an action that may interfere with a causal link.

(e) **Task Networks**: Another search space is searched by STN planners. Here, a node in the search space is a task network. A possible plan refinement operator selects a task t in the network and chooses an applicable and relevant method m to decompose t. Give definitions for:

- applicability of a method instance m,
- relevance of a method instance m, and
- the decomposition function δ.

[*4 marks*]

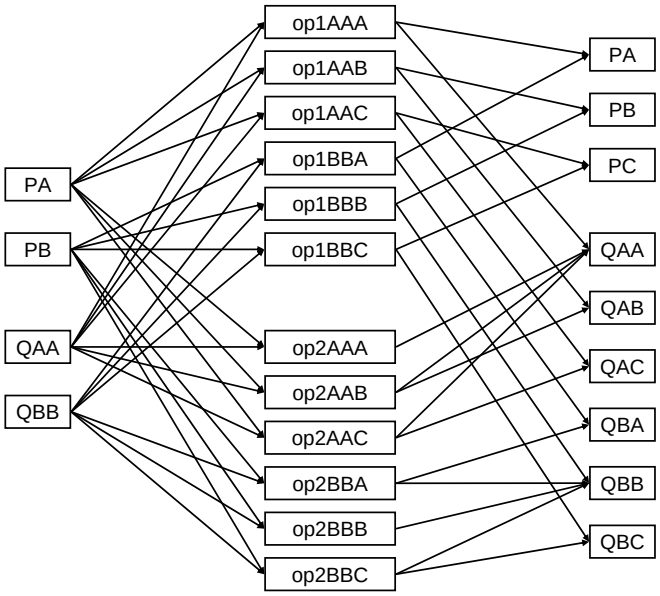
Answer:

- A method instance m is applicable in a state s if
  - $precond^+(m) \subseteq s$ and $precond^-(m) \cap s = \emptyset$.
- A method instance m is relevant for a task t if
  - there is a substitution σ such that $\sigma(t) = task(m)$.
- The decomposition of a task t by a relevant method m under σ is
  - $\delta(t, m, \sigma) = \sigma(network(m))$ or
  - $\delta(t, m, \sigma) = \sigma(\langle subtasks(m) \rangle)$ if m is totally ordered.

(f) **Graphplan**: The planning problem in appendix 1 can be translated into a propositional problem by grounding it, i.e. by replacing all variables with all possible combinations of objects defined in the problem. This results in 12 proposition symbols for states (`PA`, `PB`, `PC`, `QAA`, `QAB`, ..., `QCB`, `QCC`) and 54 possible actions (`op1AAA`, `op1AAB`, ..., `op2CCB`, `op2CCC`). Which symbols will be found in the first three layers of the planning graph (P_0, A_1, and P_1)? Give an example

of a mutex pair of actions in action layer A_1. [*6 marks*]

Answer: The following figure shows part of the planning graph containing P_0, A_1, and P_1:



No `no-op` operations are shown in A_1. Arcs from P_0 to A_1 are precondition links and arcs from A_1 to P_1 are positive effect links. No negative effect links are shown. In action layer A_1 `op1AAA` and `op2AAA` represent an example of a mutex pair of actions. These actions are dependent because the former has `PA` as a positive effect whereas the latter has `PA` as a negative effect.

Appendix 1

The following is definition of a planning domain and a statement of a planning problem in PDDL:

```
(define (domain pq-domain)
 (:requirements :strips)
 (:predicates
   (P ?x)
   (Q ?x ?y))

 (:action op1
     :parameters (?x ?y ?z)
     :precondition (and (P ?x) (Q ?x ?y))
     :effect (and  (P ?z) (not (Q ?x ?y)) (Q ?y ?z)) )

 (:action op2
     :parameters (?x ?y ?z)
     :precondition (and (P ?y) (Q ?x ?y))
     :effect (and (not (P ?z)) (Q ?x ?x) (Q ?y ?z)) )
)


(define (problem problem1)
  (:domain pq-domain)

  (:objects
   A B C)

  (:init
   (P A)
   (P B)

   (Q A A)
   (Q B B))

  (:goal
    (and (P B) (Q B C)))
)
```