

CS 541 Midterm

Please turn in your answers at the end of class. Be sure to put your name at the top of your work. All questions carry equal weight.

1. Describe the three methods that a partial-order planner such as UCPOP would use to resolve a threat.

Answer:

Assume a causal link (o1 p o2) and an operator o3 with effect $\neg p$
In order to prevent the threat a partial order planner can do:

1. Promotion: move o3 before o1 (i.e. insert ordering $o3 < o1$)
2. Demotion: move o3 after o2 (i.e. insert ordering $o2 < o3$)

Assume operator o4 with conditional effect (when q $\neg p$)

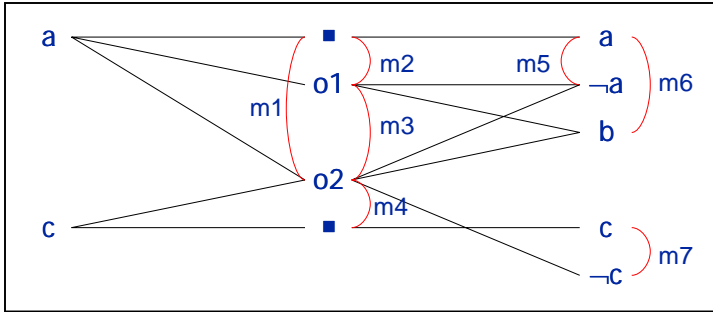
3. Confrontation: make sure that the antecedent (q) of the conditional effect is false so that the effect ($\neg p$) cannot occur, i.e., insert the open condition ($\neg p$, o4) into the agenda.

2. Assume you have the following 3 operators:

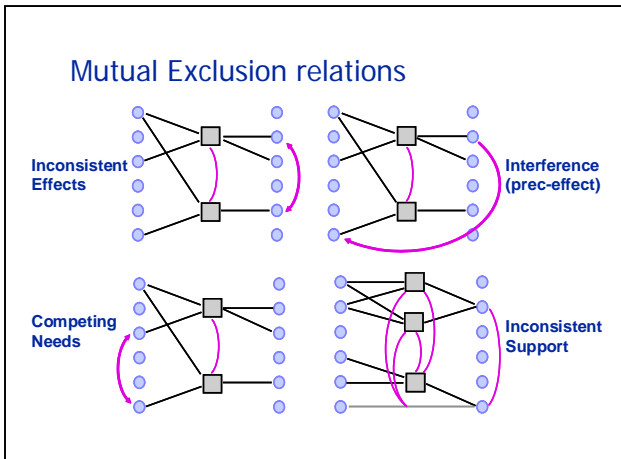
- o1: pre: a
eff: $\neg a \wedge b$
i.e., o1 has proposition a as precondition, and its effects are to add b and delete a.
- o2: pre: $a \wedge c$
eff: $\neg a \wedge b \wedge \neg c$
- o3: pre: $b \wedge c$
eff: $\neg c \wedge d$

- 2.1. Show the first three layers (proposition, action, proposition) of the planning graph starting from an initial state where only a and c are true. Include all the mutual exclusion relations (mutex) and justify each of them with a brief explanation.

Answer:

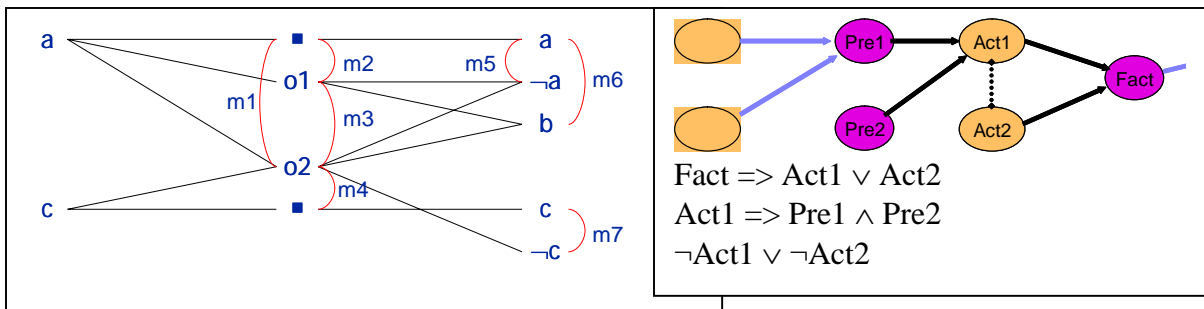


- m1: inconsistent effects (a), interference prec-effect (a)
- m2: inconsistent effects (a), interference prec-effect (a)
- m3: interference prec-effect (a)
- m4: inconsistent effects (c), interference prec-effect (c)
- m5: logically inconsistent (a)
- m6: inconsistent support, persist-a, only way of producing a, is mutex with both o1 and o2, only way to produce b.
- m7: logically inconsistent (c)



2.2. Write the propositional formula based on this planning graph that a planner based on satisfiability, like Blackbox, would use to solve the planning problem.

Answer:



IS: $a_0 \wedge c_0$

Mutex:

Effect \Rightarrow Act1 \vee Act2 :

$a_2 \rightarrow \text{persist_a_1}$
 $\neg a_2 \rightarrow o1_1 \vee o2_1$
 $b_2 \rightarrow o1_1 \vee o2_1$
 $c_2 \rightarrow \text{persist_c_1}$
 $\neg c_2 \rightarrow o2_1$

m1: $\neg \text{persist_a_1} \vee \neg o2_1$
m2: $\neg \text{persist_a_1} \vee \neg o1_1$
m3: $\neg o1_1 \vee \neg o2_1$
m4: $\neg o2_1 \vee \neg \text{persist_c_1}$
m5: $\neg a_2 \vee \neg \neg a_2 = \text{True}$
m6: $\neg a_2 \vee \neg b_2$
m7: $\neg c_2 \vee \neg \neg c_2 = \text{True}$

Act1 \Rightarrow Pre1 \wedge Pre2 :

$\text{persist_a_1} \rightarrow a_0$
 $o1_1 \rightarrow a_0$
 $o2_1 \rightarrow a_0 \wedge c_0$
 $\text{persist_c_1} \rightarrow c_0$

3. When planning is used to create a workflow in the Pegasus system, what different kinds of information are used in the preconditions and effects of the operators? What other kinds of information could be included?

Student answer:

Pegasus uses data dependencies, input requirements derived from outputs, and task constraints, e.g. the component must be run on an MPI machine. The host is identified and the goals are desired data products. Represents applying a given component at a particular location with fixed parameters, input and output.

Other information can be included, e.g. time for completing the task, expected run time, probability of failure, expected number of retries, computational cost, use of 'expensive' resources, conformance to policies.

4. How does TLPlan's use of control knowledge differ from that of earlier planning systems like Prodigy? How TLPlan's approach made efficient to use?

Answer:

TLPlan uses temporal logic expressions that test conditions over plan prefixes, while Prodigy uses control rules that apply to specific choice points in the search. Prodigy rules require the user to understand the planning algorithm, while TLPlan expressions can be thought of as independent of the planning algorithm.

TLPlan expressions would need to be re-evaluated for each new prefix in a naive implementation, essentially matching the same earlier plan prefix an exponential number

of times. Using progression, the logical requirement on the remaining part of the prefix is propagated, which only has to be checked once, and only on the next step.

5. How are mutex sets used to improve search in HSPr? How do they differ from mutexes in GraphPlan?

Student answer:

Mutex searches are used in HSPr to avoid regressing to impossible goal states, and prune them from the search tree. Mutexes in HSP are structural mutexes that describe physical impossibilities within the system. Mutexes in GraphPlan are time-dependent mutexes that show which actions and predicates exclude each other at each block of time.

6. Can distributed, coordinated planning occur without task sharing or result sharing?
7. Contrast web service representations and planning representations.