CS 2710/ISSP 2160
Homework 3 Answer key
(Paper Problems)

Problem 2: FOL Inference: Forward and Backward Chaining (15 pts)

(a) Knowledge base:

1. $Programmer(x) \wedge Emulator(y) \wedge People(z) \wedge Provide(x, y, z) \implies Criminal(x)$
2. $Use(Friends, x) \wedge Runs(x, GameXgames) \implies Provide(SuperProgrammer, x, Friends)$
3. $Software(x) \wedge Runs(x, GameXgames) \implies Emulator(x)$
4. $Programmer(SuperProgrammer)$
5. $People(Friends)$
6. $Software(Emulator1)$
7. $Use(Friends, Emulator1)$
8. $Runs(Emulator1, GameXgames)$

(b) Forward chaining:

Step 1: Rules 2 and 3 can fire, but rule 2 fires first since it has a lower rule number. (Facts 8 and 7 match the premise of rule 2 with Emulator1 substituted for x). Firing rule 2 adds the fact $Provide(SuperProgrammer, Emulator1, Friends)$ to the knowledge base. Rule 2 will not fire again with these bindings.

Step 2: Rule 3 fires since facts 6 and 8 match its premise with Emulator1 substituted for x, adding the fact $Emulator(Emulator1)$ to the knowledge base.

Step 3: Rule 1 fires since facts 4 and 5 and the newly generated facts $Provide(SuperProgrammer, Emulator1, Friends)$ and $Emulator(Emulator1)$ match this premise with SuperProgrammer substituted for x, Emulator1 for y, and Friends for z, adding the fact $Criminal(SuperProgrammer)$ to the knowledge base.

(c) Backward chaining:
We want to prove $Criminal(SuperProgrammer)$. It is not in the knowledge base. It matches

the conclusion of only rule 1 with SuperProgrammer substituted for x. Thus, we need to prove the premise of rule 1, namely $Programmer(SuperProgrammer)$, $Emulator(y)$, $People(z)$, and $Provide(SuperProgrammer, y, z)$, for some objects y and z. We consider these in turn.

$Programmer(SuperProgrammer)$ is proven since it is in the knowledge base.

$Emulator(y)$ does not match facts in the knowledge base but it matches the conclusion of rule 3 with y equal to x. Thus, we need to prove the premise of rule 3, namely $Software(y)$ and $Runs(y, GameXgames)$. $Software(y)$ is proven with Emulator1 substituted for y since $Software(Emulator1)$ is in the knowledge base. We then need to prove $Runs(Emulator1, GameXgames)$. This is proven since it is in the knowledge base as well.

$People(z)$ is proven with Friends substituted for z since $People(Friends)$ is in the knowledge base.

Finally, $Provide(SuperProgrammer, y, z)$ now is $Provide(SuperProgrammer, Emulator1, Friends)$, which is not in the knowledge base but matches the conclusion of rule 2 with Emulator1 substituted for x. Thus, we need to prove the premise of rule 2, namely $Use(Friends, Emulator1)$ and $Runs(Emulator1, GameXgames)$, which are both in the knowledge base.

At this point, the proof of $Criminal(SuperProgrammer)$ succeeds.

Problem 3: FOL Inference: Resolution (15 pts)

The relevant definitions for the family are:

$$Rel(Parent, x, y) \wedge Male(x) \iff Rel(Father, x, y) \tag{1}$$
$$Rel(Son, x, y) \iff Rel(Parent, y, x) \wedge Male(x) \tag{2}$$
$$Rel(Sibling, x, y) \iff \neg(x = y) \wedge \exists p\ Rel(Parent, p, x) \wedge Rel(Parent, p, y) \tag{3}$$
$$Rel(Father, x_1, y) \wedge Rel(Father, x_2, y) \implies x_1 = x_2 \tag{4}$$

The facts of the case are:

$$\neg Rel(Sibling, Me, x) \tag{5}$$
$$Male(MrX) \tag{6}$$
$$Rel(Father, FatherX, MrX) \tag{7}$$
$$Rel(Father, FatherMe, Me) \tag{8}$$
$$Rel(Son, FatherX, FatherMe) \tag{9}$$

We want to be able to show that Me us the only son of my father, and therefore that Me is father of MrX, who is make, and therefore that "that man" is my son. The query we want us therefore:

$$Rel(r, MrX, y) \tag{10}$$

We want to be able to get back the answer $\{r/Son, y/Me\}$. Translating 1–10 into CNF (and negating (10)) we get:

(1a) $\neg Rel(Parent, x, y) \vee \neg Male(x) \vee Rel(Father, x, y)$

(1b) $\neg Rel(Father, x, y) \vee Male(x)$

(1c) $\neg Rel(Father, x, y) \vee Rel(Parent, x, y)$

(2a) $\neg Rel(Son, x, y) \vee Rel(Parent, y, x)$

(2b) $\neg Rel(Son, x, y) \vee Male(x)$

(2c) $\neg Rel(Parent, y, x) \vee \neg Male(x) \vee Rel(Son, x, y)$

(3a) $\neg Rel(Sibling, x, y) \vee \neg(x = y)$

(3b) $\neg Rel(Sibling, x, y) \vee Rel(Parent, P(x, y), x)$

(3c) $\neg Rel(Sibling, x, y) \vee Rel(Parent, P(x, y), y)$

(3d) $\neg Rel(Parent, P(x, y), x) \vee \neg Rel(Parent, P(x, y), y) \vee (x = y) \vee Rel(Sibling, x, y)$

(4) $\neg Rel(Father, x_1, y) \vee \neg Rel(Father, x_2, y) \vee (x_1 = x_2)$

(10) $\neg Rel(r, MrX, y)$

The following lists the steps of the proof, with the resolvents of each step in braces on the left:

(11):{1c,8} $Rel(Parent, FatherMe, Me)$

(12):{2a,9} $Rel(Parent, FatherMe, FatherX)$

(13):{3d,11} $\neg Rel(Parent, FatherMe, y) \vee (Me = y) \vee Rel(Sibling, Me, y)$

(14):{5,13} $\neg Rel(Parent, FatherMe, y) \vee (Me = y)$

(15):{12,13} $Me = FatherX \vee Rel(Sibling, Me, FatherX)$

(16):{5,15} $Me = FatherX$

(17):{7,16} $Rel(Father, Me, MrX)$

(18):{1c,17} $Rel(Parent, Me, MrX)$

(19):{2c,18} $\neg Male(MrX) \vee Rel(Son, MrX, Me)$

(20):{6,19} $Rel(Son, MrX, Me)$

(21):{10,20} $FALSE\{r/Son, y/Me\}$

Problem 4: Planning: Partial Order (15 pts)

Initial State: A B C **Goal:** C D E

Operator	Precondition	Add	Delete
O1	A	F	C
O2	A	D	B
O3	B	C	
O4	F	DEG	

(a) Graphplan (Fig. 1)

- Two actions instances at level i are mutex if either
  - Inconsistent effects: the effect of one action is the negation of the other action effect, or
  - Interference: one action deletes the precondition of the other, or
  - Competing needs: the actions have preconditions that are mutually exclusive at level $i-1$
- Two propositions at level i are mutex if all ways of achieving the propositions (i.e., actions at level $i-1$) are pairwise mutex.

Level 0 begins with the initial state of ABC

Level 1 consists of the possible actions that have preconditions satisfied from level 0. All arcs drawn between levels indicate mutex relations. Action O1 and O3 are mutually exclusive (mutex) due to inconsistent effects. O1 and the maintenance action of C are also mutex for the same reason. O2 is mutex O3 by interference and mutex the maintenance action of B by inconsistent effects.

Level 2 consists of the possible effects from the actions in level 1. A and B are possible due to the maintenance actions. C is possible from a maintenance action and action O3. D is possible from action O2, and F is possible from action O1. B D and C F are mutex by inconsistent support.

Level 3 contains all the actions and mutexes from level 1 with one additional action and two additional mutexes. Action O4 is now possible due to the addition of F at level 2. Action O4 is mutex the maintenance action of C by competing needs. Action O3 is mutex the maintenance action of D by competing needs.

Level 4 consists of all the possible effects from the actions in level 3. A B C D F are possible from maintenance actions. C from action O3. D from action O2 or 03. F from action O1. E and G from action O4.

Graphplan proceeds by expanding the planning graph to level 2. It then checks to see if effects present at this level can satisfy the goal requirements. In this case C and D are present but E is absent. Graphplan then proceed to expand the graph to level 4. Level 4 contains effects that satisfy the goal conditions, so graphplan begins to search backward to find a valid plan. We discuss a successful backward search. An unsuccessful search would

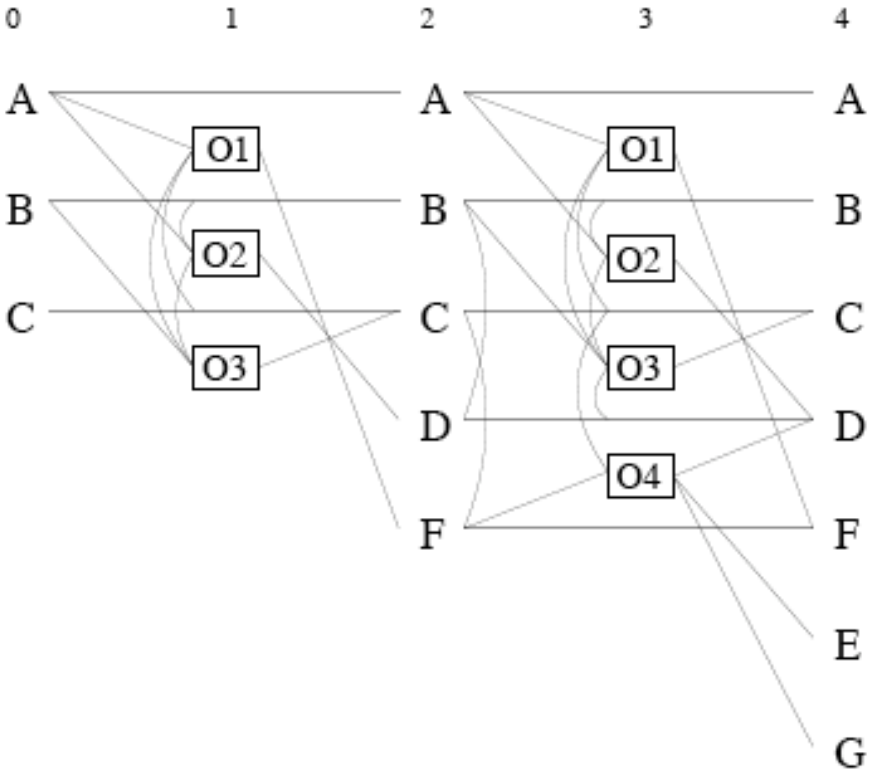

Figure 1: Planning graph: The graph has been expanded two time steps which is sufficient to find a valid plan.

result in dead ends with no actions to choose within the mutex constraints which would then backtrack the search and continue. At level 3 graphplan chooses O3 to satisfy C and O4 to satisfy D and E, this is valid as there is no mutex between O3 and O4. The preconditions for level 3, B and F, are now the new goals to achieve. At level 1 graphplan chooses the maintenance action of B to satisfy B and action O1 to satisfy F. The preconditions from level 1 are satisfied from the initial state so the plan is completed: O1 at time 1, O3 and O4 at time 2.

(b) Regression planner
The following diagrams (Fig. 2) will show one way that a partial order planner would find a solution the planning problem. It only displays a perfect selection sequence, but a real planner may choose incorrectly resulting in backtracking. Each square is an operator. The letters above the squares are preconditions and the letters below are effects. Single width arrow lines are ordering constraints, and double width arrow lines are causal links. Notice that not all ordering constraints are shown in the graph as causal links are assumed to be accompanied by a similar ordering constraint and ordering constraints that can be inferred from those shown are not displayed. For example every operator has an ordering constraint

of $Start \prec Operator \prec Goal$ but these are not displayed if they can be inferred.

Step 0 The plan is created with only the start and goal states.

Step 1 The planner selects for a subgoal the precondition D from the goal state. It then chooses the O4 operator to satisfy this subgoal. It places an ordering constrain $Start \prec O4 \prec Goal$, a causal link $O4 \stackrel{D}{\rightarrow}$ Goal and ordering constrain $O4 \prec Goal$ (from now on we omit the mention of the ordering constraint that accompanies the causal link – it is now implied).

Step 2 The planner selects for a subgoal the precondition E from the goal state. It then chooses the O4 operator to satisfy this subgoal. It adds a causal link $O4 \stackrel{E}{\rightarrow} Goal$.

Step 3 The planner selects for a subgoal the precondition F from operator O4. It then chooses the O1 operator to satisfy this subgoal. It adds an ordering constraint $Start \prec O1 \prec Goal$ and a causal link $O1 \stackrel{F}{\rightarrow} O4$.

Step 4 The planner selects for a subgoal the precondition A from operator O1. It then chooses the start state to satisfy this subgoal. It adds a causal link $Start \stackrel{A}{\rightarrow} O1$.

Step 5 The planner selects for a subgoal the precondition C from operator goal state. It then chooses the O3 operator to satisfy this subgoal. It places an ordering constrain $Start \prec O3 \prec Goal$ and a causal link $O3 \stackrel{C}{\rightarrow} Goal$.

Step 6 There is a threat between O3 and O1. The planner chooses to promote O1 by adding an ordering constraint $O1 \prec O3$.

Step 7 The planner selects for a subgoal the precondition B from operator O3. It then chooses the start state to satisfy this subgoal. It adds a causal link $Start \stackrel{B}{\rightarrow} O3$.

Finished The planner has now reached a solution. O1 followed by O3 and O4 together or in any order.

Problem 5: Planning: STRIPS (15 pts)

Actions: $Action(\text{ACTION} : Go(x, y), \text{PRECOND} : At(Shakey, x) \wedge In(x, r) \wedge In(y, r), \text{EFFECT} : At(Shakey, y) \wedge \neg(At(Shakey, x)))$
$Action(\text{ACTION} : Push(b, x, y), \text{PRECOND} : At(Shakey, x) \wedge Pushable(b), \text{EFFECT} : At(b, y) \wedge At(Shakey, y) \wedge \neg At(b, x) \wedge \neg At(Shakey, x))$
$Action(\text{ACTION} : ClimbUp(b), \text{PRECOND} : At(Shakey, x) \wedge At(b, x) \wedge Climbable(b), \text{EFFECT} : On(Shakey, b) \wedge \not On(Shakey, Floor))$
$Action(\text{ACTION} : ClimbDown(b), \text{PRECOND} : On(Shakey, b), \text{EFFECT} : On(Shakey, Floor) \wedge \not On(Shakey, b))$
$Action(\text{ACTION} : TurnOn(l), \text{PRECOND} : On(Shakey, b) \wedge At(Shakey, x) \wedge At(l, x), \text{EFFECT} : TurnedOn(l))$
$Action(\text{ACTION} : TurnOff(l), \text{PRECOND} : On(Shakey, b) \wedge At(Shakey, x) \wedge At(l, x), \text{EFFECT} : \neg TurnedOn(l))$

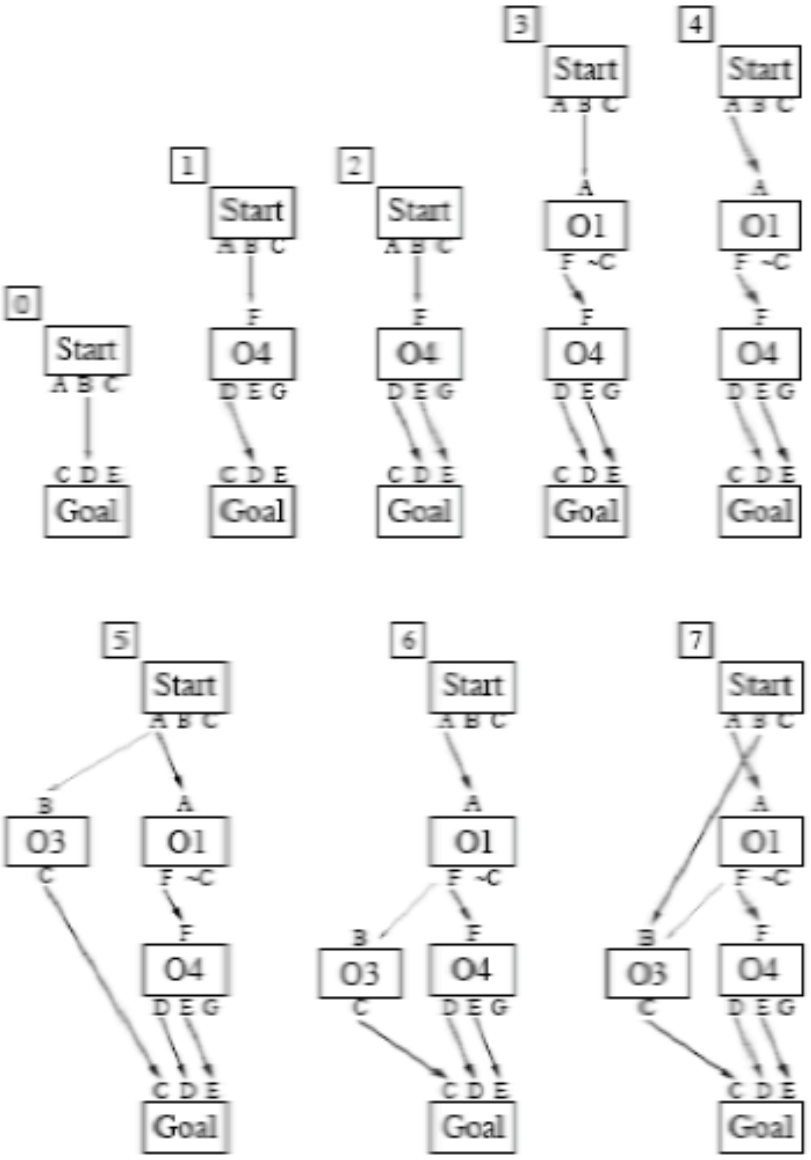

Figure 2: Steps taken by a partial order planner with perfect selection.

Initial State: $In(Switch_1, Room_1) \wedge In(Door_1, Room_1) \wedge In(Door_1, Corridor)$
$In(Switch_1, Room_2) \wedge In(Door_2, Room_2) \wedge In(Door_2, Corridor)$
$In(Switch_1, Room_3) \wedge In(Door_3, Room_3) \wedge In(Door_3, Corridor)$
$In(Switch_1, Room_4) \wedge In(Door_4, Room_4) \wedge In(Door_4, Corridor)$
$In(Shakey, Room_3) \wedge At(Shakey, X_S)$
$In(Box_1, Room_1) \wedge In(Box_2, Room_1) \wedge In(Box_3, Room_1) \wedge In(Box_4, Room_1)$
$Climbable(Box_1) \wedge Climbable(Box_2) \wedge Climbable(Box_3) \wedge Climbable(Box_4)$
$Pushable(Box_1) \wedge Pushable(Box_2) \wedge Pushable(Box_3) \wedge Pushable(Box_4)$
$At(Box_1, X_1) \wedge At(Box_2, X_2) \wedge At(Box_3, X_3) \wedge At(Box_4, X_4)$
$TurnedOn(Switch_1) \wedge TurnedOn(Switch_4)$.

Plan: $Go(X_S, Door_3)$
$Go(Door_3, Door_1)$
$Go(Door_1, X_2)$
$Push(Box_2, X_2, Door_1)$
$Push(Box_2, Door_1, Door_2)$
$Push(Box_2, Door_2, Switch_2)$