Homework Set 1

Michael Anderson

January 11, 2011

CS533

Prof. Fern

1. a) Low(Monkey)
At(Monkey,A)
At(Bananas,B)
At(Box,C)

b) Go(x,y):
PRE: {At(Monkey,x)}
ADD: {At(Monkey,y)}
DEL: {At(Monkey,x)}

Push(x,l1,l2):
PRE: {Height(Monkey,Low), At(x,l1)}
ADD: {At(x,l2)}
DEL: {At(x,l1)}

ClimbUp():
PRE: {At(Box,l), At(Monkey,l), Height(Monkey,Low)}
ADD: {Height(Monkey,High)}
DEL: {Height(Monkey,Low)}

ClimbDown():
PRE: {Height(Monkey,High)}
ADD: {Height(Monkey,Low)}
DEL: {Height(Monkey,High)}

Grasp():
PRE: {Height(Monkey,High), At(Monkey,l), At(Bananas,l),
NotHolding(Bananas)}
ADD: {Holding(Bananas)}
DEL: {NotHolding(Bananas)}

Ungrasp():
PRE: {Holding(Bananas)}
ADD: {NotHolding(Bananas)}
DEL: {Holding(Bananas)}

d) Push(x,l1,l2):
PRE: {Height(Monkey,Low), At(x,l1), Weight(x,Light)}
ADD: {At(x,l2)}
DEL: {At(x,l1)}

2. Go(x,y,r):
PRE: {At(Shakey,x), In(x,r), In(y,r)}
ADD: {At(Shakey,y)}
DEL: {At(Shakey,x)}

Push(b,x,y,r):
PRE: {At(Shakey,x), At(b,x), In(x,r), In(y,r), Box(b)}
ADD: {At(Shakey,y), At(b,y)}
DEL: {At(Shakey,x), At(b,x)}

ClimbUp(x,b):
PRE: {At(Shakey,x), At(b,x), Box(b), On(Shakey,Floor)}
ADD: {On(Shakey,b)}
DEL: {On(Shakey,Floor)}

ClimbDown(b,x):
PRE: {On(Shakey,b), At(b,x), Box(b)}
ADD: {On(Shakey,Floor)}
DEL: {On(Shakey,b)}

TurnOn(s,b):
PRE: {On(Shakey,b), At(b,s), Box(b), Off(s)}
ADD: {On(s)}
DEL: {Off(s)}

TurnOff(s,b):
PRE: {On(Shakey,b), At(b,s), Box(b), On(s)}
ADD: {Off(s)}
DEL: {On(s)}

Initial State: {At(Shakey,StartLoc), At(Box1,BoxStart1),
At(Box2,BoxStart2), At(Box3,BoxStart3), At(Box4,BoxStart4),
In(Shakey,Room3), In(Box1,Room1), In(Box2,Room1), In(Box3,Room1),
In(Box4,Room1), Box(Box1), Box(Box2), Box(Box3), Box(Box4),
On(Switch1), Off(Switch2), Off(Switch3), On(Switch4)}

Plan: (Go(StartLoc,Door3,Room3), Go(Door3,Door1,Corridor),
Push(Box2,BoxStart2,Door1,Room1),
Push(Box2,Door1,Door2,Corridor))

3. Initial State: {On(A,Table), On(B,Table), On(C,A), Block(A), Block(B),
Block(C), Clear(B), Clear(C)}
Goal State: {On(A,B), On(B,C)}

Actions:

Move(b,x,y):
PRE: {On(b,x), Clear(b), Clear(y), Block(b), Block(y)}
ADD: {On(b,y), Clear(x)}
DEL: {On(b,x), Clear(y)}

MoveToTable(b,x):
PRE: {On(b,x), Clear(b), Block(b)}
ADD: {On(b,Table), Clear(x)}
DEL: {On(b,x)}

Plan: (MoveToTable(C,A), Move(B,Table,C), Move(A,Table,B))

The goal state has two parts, On(A,B) and On(B,C). Suppose a planner tries to accomplish one of the two subgoals, and then accomplish the other in succession. It can put A on B after moving C out of the way, but then it will not be able to put B on C without unburying B and

undoing the first subgoal. Similarly, it can put B on C immediately with a single move, but A will have to be unburied, undoing the second subgoal. This means that the approach of dividing a goal state into multiple subgoals that can be tackled separately is not always useful, because the process of reaching one subgoal can interfere with the process of reaching another.

4. Graphplan would have to expand 3 levels, because all 3 of the actions required to execute the plan given in (3.) are mutex.

MoveToTable(C,A) cannot be in the same layer as Move(B,Table,C). Clear(C) is a precondition of MoveToTable(C,A), but Clear(C) is deleted by Move(B,Table,C).

MoveToTable(C,A) cannot be in the same layer as Move(A,Table,B). Clear(A) is a precondition for Move(A,Table,B), while the competing precondition On(C,A) is necessary for MoveToTable(C,A).

Finally, Move(B,Table,C) cannot be in the same layer as Move(A,Table,B). Clear(B) is a precondition of Move(B,Table,C), but Clear(B) is deleted by Move(A,Table,B).

5. This is because all possible plans that span from the initial state to the final level lead to some state that is a subset of the final literals. Therefore, there can be no n-length plan leading to a state with a literal that does not appear at the nth level. Furthermore, since the set of available literals is monotonically increasing relative to the depth of the graph, no subplan of length l, where $0 \leq l \leq n$, can ever reach a literal that does not appear in the nth level of the graph.

6. a) For each positive literal L defined in p, define an additional literal NotL in p'. The initial state of p' should include the initial state of p, along with all of the Not literals whose positive counterparts do not appear in p. Because of the standard closed world assumption, this does not really alter the initial state.

Now map every action schema in p to an action schema in p', with the following alterations. Replace each negative precondition in p with the corresponding newly-minted Not literal in p'. In every instance where a literal is added as an action effect in p, delete its Not literal in p'. In every instance where a literal is deleted as an action effect in p, add its Not literal in p'. All of this produces actions in p' that are semantically equivalent to their counterparts in p, except that the closed-world assumption is rendered unneccesary and new literals are created to handle negatives, which planning software cannot and does not need to distinguish from the original literals in p.

Leave the goal unchanged, since it encodes a whole set of states that contain its propositions, instead of one individual state.

b) Make p' the same as p, with the following alterations.

Create new actions $Goal1$, $Goal2$, ... $Goaln$ in p' for each of the n standard goals in the disjunctive goal. Each new action should have as its preconditions the propositions of a standard goal. These actions should add a new literal $StandardGoalFound$. For example, the actions created for the given sample disjunctive goal would be:

Goal1:
PRE: on(a,b), on(b,c)
ADD: StandardGoalFound

Goal2:
PRE: on(c,b), on(b,a)
ADD: StandardGoalFound

Now make the goal of p' simply {StandardGoalFound}. This works because StandardGoalFound will be reachable iff one of the sets of propositions of the standard goals can be satisfied, which is what we want. A planner will be able to take the appropriate Goal action at that point and reach the goal.