

# Homework #5: Chapters 11-12

1.

a)

$At(P1,LAX) \wedge At(P2,JFK) \wedge At(C1,LAX) \wedge In(C2,P1)$	
Load(C1,P1,LAX)	$At(P1,LAX) \wedge At(P2,JFK) \wedge In(C1,P1) \wedge In(C2,P1)$
Unload(C2,P1,LAX)	$At(P1,LAX) \wedge At(P2,JFK) \wedge At(C1,LAX) \wedge At(C2,LAX)$
Fly(P1,LAX,LAX)	$At(P2,JFK) \wedge At(C1,LAX) \wedge In(C2,P1) \wedge At(P1,LAX)$
Fly(P1,LAX,JFK)	$At(P2,JFK) \wedge At(C1,LAX) \wedge In(C2,P1) \wedge At(P1,JFK)$
Fly(P1,LAX,ORD)	$At(P2,JFK) \wedge At(C1,LAX) \wedge In(C2,P1) \wedge At(P1,ORD)$
Fly(P2,JFK,LAX)	$At(P1,LAX) \wedge At(C1,LAX) \wedge In(C2,P1) \wedge At(P2,LAX)$
Fly(P2,JFK,JFK)	$At(P1,LAX) \wedge At(C1,LAX) \wedge In(C2,P1) \wedge At(P2,JFK)$
Fly(P2,JFK,ORD)	$At(P1,LAX) \wedge At(C1,LAX) \wedge In(C2,P1) \wedge At(P2,PRD)$

b)

$At(P1,JFK) \wedge At(P2,ORD) \wedge At(C1,JFK) \wedge In(C2,P2)$	
Load(C2,P2,a)	$At(P1,JFK) \wedge At(P2,ORD) \wedge At(C1,JFK) \wedge At(C2,a) \wedge At(P2,a)$
Unload(C1,p,JFK)	$At(P1,JFK) \wedge At(P2,ORD) \wedge In(C2,P2) \wedge In(C1,p) \wedge At(p,JFK)$
Fly(P1,a,JFK), $a \neq JFK$	$At(P2,ORD) \wedge At(C1,JFK) \wedge In(C2,P2) \wedge At(P1,a)$
Fly(P2,a,ORD), $a \neq ORD$	$At(P1,JFK) \wedge At(C1,JFK) \wedge In(C2,P2) \wedge At(P2,a)$

Notes:

- it is acceptable to use only constants if all possible legal combinations are enumerated. That is,  $Load(C2,P1,ORD)$ ,  $Load(C2,P1,JFK)$ ,  $Load(C2,P1,LAX)$ ,  $Unload(C1,P1,JFK)$ ,  $Unload(C1,P2,JFK)$ ,  $Fly(P1, ORD,JFK)$ ,  $Fly(P1,LAX,JFK)$ ,  $Fly(P2,JFK,ORD)$ , and  $Fly(P2,LAX,ORD)$ .
- Technically, we can't simply have  $Load(C2,P2,ORD)$  or  $Unload(C1,P1,JFK)$  because the planner doesn't know that  $P2$  cannot be at  $ORD$  and another airport at the same time..

2.

- a) The minimal set is: Clean(x), Empty(x), Swept(x), and AppliedOn(x,y).

Optionally, additional predicates for Dirty(x) and Full(x) may be used, but then the actions must be sure to remove Clean/Dirty when adding the other and remove Empty/Full when adding the other. You may also define predicates for each type of item in the kitchen, but this will require more complicated actions in part (b).

- b) Note: Here, we assume that Stove, Floor, Cleaner, Garbage, Sink, Counter, and Fridge are constants:

Action(CleanStove(), PRECOND:  $\emptyset$ , EFFECT: Clean(Stove)  $\wedge$   $\neg$ Clean(Floor))  
Action(ApplyOvenCleaner(), PRECOND:  $\emptyset$ , EFFECT: AppliedOn(Cleaner,Oven))  
Action(WipeOvenCleaner(), PRECOND: AppliedOn(Cleaner,Oven), EFFECT:  
     $\neg$ AppliedOn(Cleaner,Oven)  $\wedge$  Clean(Oven))  
Action(SweepFloor(), PRECOND: Empty(Garbage), EFFECT: Swept(Floor))  
Action(WashFloor(), PRECOND: Swept(Floor), EFFECT:  $\neg$ Clean(Sink)  $\wedge$  Clean(Floor))  
Action(TakeOutGarbage(), PRECOND:  $\emptyset$ , EFFECT: Empty(Garbage))  
Action(CleanFridge(), PRECOND:  $\emptyset$ , EFFECT:  $\neg$ Clean(Counter)  $\wedge$   $\neg$ Empty(Garbage)  
     $\wedge$   $\neg$ Clean(Floor)  $\wedge$  Clean(Fridge))  
Action(WashCounter(), PRECOND:  $\emptyset$ , EFFECT:  $\neg$ Clean(Sink)  $\wedge$  Clean(Counter))  
Action(CleanSink(), PRECOND:  $\emptyset$ , EFFECT: Clean(Sink))

- c) There are many possible initial states, depending on the extent that the kitchen needs cleaning. For the minimal set of predicates, the dirtiest kitchen is represented by the empty set. The goal state, is one in which all items are clean (whether or not the garbage is empty is optional).

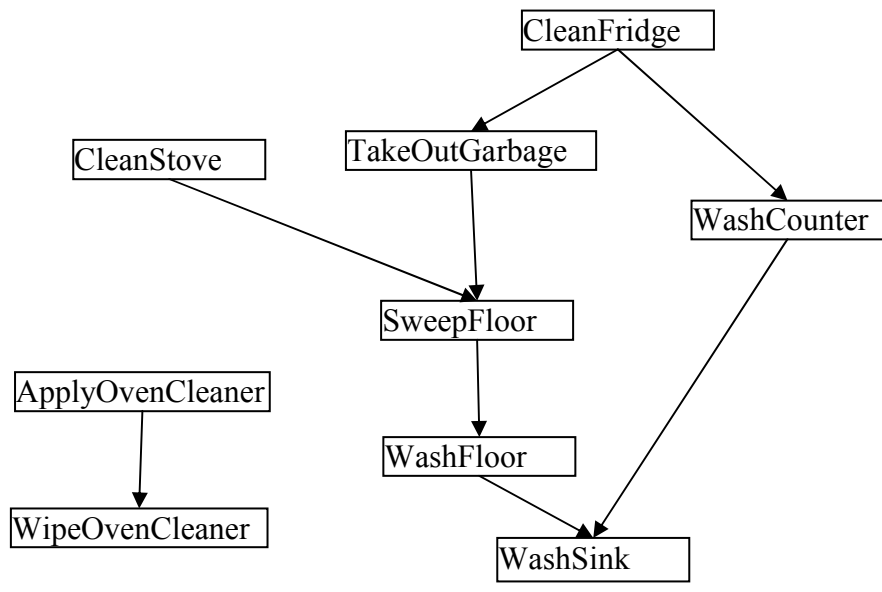
**Initial State:**  $\emptyset$

**Goal State:** Clean(Stove)  $\wedge$  Clean(Oven)  $\wedge$  Clean(Floor)  $\wedge$  Clean(Fridge)  $\wedge$   
Clean(Counter)  $\wedge$  Clean(Sink) [ $\wedge$  Empty(Garbage)]

- d) There are a number of possible orderings. Here's one:

ApplyOvenCleaner()  
WipeOvenCleaner()  
CleanStove()  
CleanFridge()  
TakeOurGarbage()  
SweepFloor()  
WashFloor()  
WashCounter()  
CleanSink()

On the next page is a diagram that shows the dependencies for the various actions. Any plan that does not repeat some steps must obey the partial order shown.

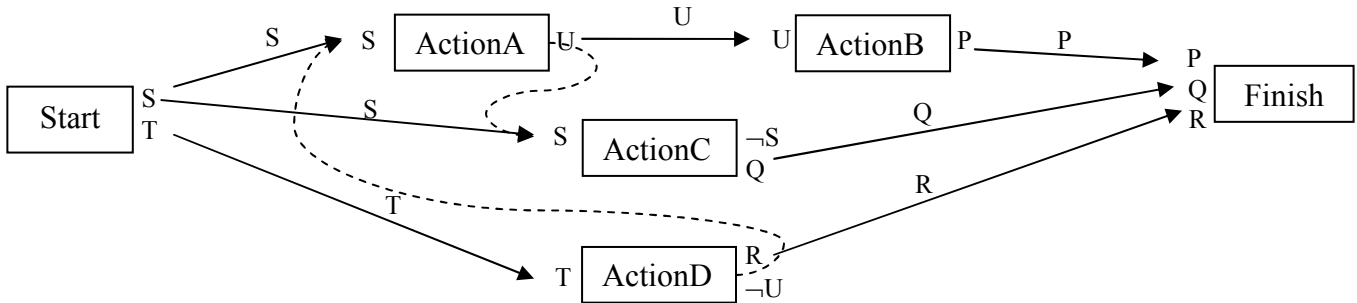


3.

Conflicts:

- ActionC cannot come between Start and ActionA.
- ActionD cannot come between ActionA and ActionB.

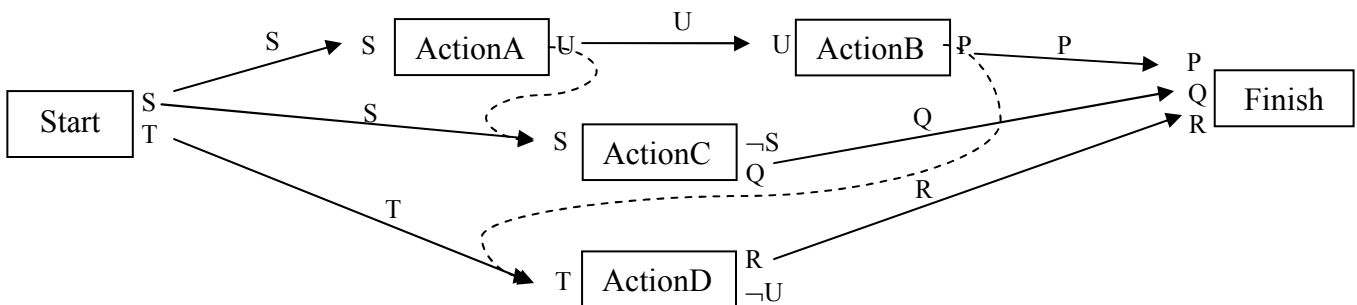
Resolution 1:



Linearizations of Resolution 1: (ActionA abbreviated as A, and so on...)

- D, A, B, C
- D, A, C, B

Resolution 2:



Linearizations of Resolution 2: (ActionA abbreviated as A, and so on...)

- A, B, D, C
- A, B, C, D
- A, C, B, D

4. There are many possible answers to this question. The examples from the book and class (which should not be repeated) are:

Bounded indeterminacy:

- flipping a coin – it will either come up heads or tails

Unbounded indeterminacy:

- driving, economic planning, military strategy (however, a sufficiently detailed example is acceptable, since the book does not go into detail).

Sample acceptable answers are:

**Bounded indeterminacy:**

- a single die roll – there are 6 possible results (1-6)
- choice to hit or stay in Black Jack. Although there are 52 cards the player could receive, these can be divided into 11 equivalence classes (ace, digits 2-10, and face cards) in terms of impact on the game.

**Unbounded indeterminacy:**

- a real robot's "turn 90 degrees" action – The robot might turn 89 degrees, or 91 degrees, the motor might fail altogether, the robot might get stuck against an obstacle, it might lose traction on the floor, a person might be blocking it, etc. We could continue to list reasons why the turn was not precise or didn't occur at all.
- predicting the weather – Although, we could discretize the possible events (e.g., rain, sun, snow, cloud), a good prediction should involve temperature and amount of precipitation as well (which are continuous values). Also, there many factors (pressure, temperature, moisture conditions at various locations, geographic effects, etc.) that determine the weather, and many of these may be unknown. Note, this is indeterminacy regarding the state of the world, as opposed to the effects of actions.
- developing a business plan – there are simply too many economic, market and production factors to consider all possible eventualities. Although one can plan for the most likely or most serious contingencies, there will always be unexpected circumstances that must be dealt with when they arise.