HOMEWORK 3 SOLUTIONS

1. Planning States

S0 in Situation Calculus	**S0 in Strips**
NON-FLUENTS	**NON-FLUENTS**
ROOM(Rm1)	ROOM(Rm1)
ROOM (Rm2)	ROOM (Rm2)
ROOM (Rm3)	ROOM (Rm3)
ROOM (Rm4)	ROOM (Rm4)
ROOM (Cor)	ROOM (Cor)
ROBOT(Shky)	ROBOT(Shky)
BOX(Bx1)	BOX(Bx1)
BOX(Bx2)	BOX(Bx2)
BOX(Bx3)	BOX(Bx3)
BOX(Bx4)	BOX(Bx4)
DOOR(Dr1)	DOOR(Dr1)
DOOR(Dr2)	DOOR(Dr2)
DOOR(Dr3)	DOOR(Dr3)
DOOR(Dr4)	DOOR(Dr4)
CONNECTS(Dr1,Rm1,Cor)	CONNECTS(Dr1,Rm1,Cor)
CONNECTS(Dr1,Cor,Rm1)	CONNECTS(Dr1,Cor,Rm1)
CONNECTS(Dr2,Rm2,Cor)	CONNECTS(Dr2,Rm2,Cor)
CONNECTS(Dr2,Cor,Rm2)	CONNECTS(Dr2,Cor,Rm2)
CONNECTS(Dr3,Rm3,Cor)	CONNECTS(Dr3,Rm3,Cor)
CONNECTS(Dr3,Cor,Rm3)	CONNECTS(Dr3,Cor,Rm3)
CONNECTS(Dr4,Rm4,Cor)	CONNECTS(Dr4,Rm4,Cor)
CONNECTS(Dr4,Cor,Rm4)	CONNECTS(Dr4,Cor,Rm4)
DIFF(Rm1,Rm2)	DIFF(Rm1,Rm2)
DIFF(Rm1,Rm3)	DIFF(Rm1,Rm3)
DIFF(Rm1,Rm4)	DIFF(Rm1,Rm4)
DIFF(Rm2,Rm1)	DIFF(Rm2,Rm1)
DIFF(Rm2,Rm3)	DIFF(Rm2,Rm3)
DIFF(Rm2,Rm4)	DIFF(Rm2,Rm4)
DIFF(Rm3,Rm1)	DIFF(Rm3,Rm1)
DIFF(Rm3,Rm2)	DIFF(Rm3,Rm2)
DIFF(Rm3,Rm4)	DIFF(Rm3,Rm4)
DIFF(Rm4,Rm1)	DIFF(Rm4,Rm1)
DIFF(Rm4,Rm2)	DIFF(Rm4,Rm2)
DIFF(Rm4,Rm3)	DIFF(Rm4,Rm3)
DIFF(Bx1,Bx2)	DIFF(Bx1,Bx2)
DIFF(Bx1,Bx3)	DIFF(Bx1,Bx3)
DIFF(Bx1,Bx4)	DIFF(Bx1,Bx4)

DIFF(Bx2,Bx1)	DIFF(Bx2,Bx1)
DIFF(Bx2,Bx3)	DIFF(Bx2,Bx3)
DIFF(Bx2,Bx4)	DIFF(Bx2,Bx4)
DIFF(Bx3,Bx1)	DIFF(Bx3,Bx1)
DIFF(Bx3,Bx2)	DIFF(Bx3,Bx2)
DIFF(Bx3,Bx4)	DIFF(Bx3,Bx4)
DIFF(Bx4,Bx1)	DIFF(Bx4,Bx1)
DIFF(Bx4,Bx2)	DIFF(Bx4,Bx2)
DIFF(Bx4,Bx3)	DIFF(Bx4,Bx3)
FLUENTS	**FLUENTS**
IN_ROOM(Shky,Rm3,S0)	IN_ROOM(Shky,Rm3)
IN_ROOM(Bx1,Rm1,S0)	IN_ROOM(Bx1,Rm1)
IN_ROOM(Bx2,Rm1,S0)	IN_ROOM(Bx2,Rm1)
IN_ROOM(Bx3,Rm1,S0)	IN_ROOM(Bx3,Rm1)
IN_ROOM(Bx4,Rm1,S0)	IN_ROOM(Bx4,Rm1)
¬ IS_CARRYING(Shky,Bx1,S0)	NOT_CARRYING(Shky,Bx1)
¬ IS_CARRYING(Shky,Bx2,S0)	NOT_CARRYING(Shky,Bx2)
¬ IS_CARRYING(Shky,Bx3,S0)	NOT_CARRYING(Shky,Bx3)
¬ IS_CARRYING(Shky,Bx4,S0)	NOT_CARRYING(Shky,Bx4)

Only things that are allowed to change are fluents.

We need a new relation DIFF(x,y) [meaning x and y denote different objects]. Note that DIFF must be asserted both directions. Also we need CONNECTS both ways through the doors. Many other things are DIFF but we need not assert them since they are not needed by our operators. We need some way of keeping track of how many things Shakey can pick up and what moves and doesn't when Shakey goes to a new room. In Strips we need a new relation NOT_CARRYING(x,y) [meaning that box y is not being carried by x]. This is needed because the preconditions can only test positive literals. IS_CARRYING and NOT_CARRYING are treated as pairs. Whenever IS_CARRYING is added we delete NOT_CARRYING that box. Whenever NOT_CARRYING is added we delete IS_CARRYING that box. etc. This is not needed in situation calculus (or PDDL) because we can use negation.

2. Planning Actions

1) In situation calculus we can manage with three operators: one for PICKUP, one for PUTDOWN and one for MOVE. The operators are somewhat involved with internal conditions or disjunctions.

2) Strips is more complicated because of the confining syntax of precondition, add and delete lists of positive literals. Consider MOVE. When Shakey goes to a new room, all the boxes he is carrying go to the new room as well. We need a separate MOVE operator for every number of boxes being carried: one for carrying no boxes, one for a single box, and one for carrying two boxes. For PICKUP we need

two operators (the same as for MOVE except we do not want preconditions to be satisfied when two boxes are already carried). For PUTDOWN we also need two operators (the same for MOVE except we do not need the case of nothing being carried). So in all 3 + 2 + 2 = 7 operators.

3) PDDL in this domain provides all of the expressiveness that we used in the situation calculus treatment so it also suffices to use three operators. Internally, most PDDL implementations would expend these three to the 7 of Strips. Incidentally, these would be further multiplied by SAT and GRAPH planners. Propositionalization would introduce more propositional versions (specialized to a particular context – for each room, each door, etc.) as the contexts were considered to be potentially relevant.

4a) We give a situation calculus change axiom for a robot moving through a door.

$\forall r, \forall d, \forall m1, \forall m2, \forall s$ [ROBOT(r) $\wedge$ ROOM(m1) $\wedge$ ROOM(m2) $\wedge$ CONNECTS(d,m1,m2) $\wedge$
DIFF(m1,m2) $\wedge$ IN_ROOM(r,m1,s)] $\Rightarrow$
[IN_ROOM(r,m2,Result(MOVE(r,d),s)) $\wedge$ $\forall$b {(BOX(b) $\wedge$ IS_CARRYING(r,b,s))
$\Rightarrow$ IN_ROOM(b,m2,Result(MOVE(r,d),s))}]

4b) We give a situation calculus frame axiom for boxes not carried to stay where they are in a MOVE.

$\forall r, \forall b, \forall d, \forall m1, \forall m2, \forall m3, \forall s$ [ROBOT(r) $\wedge$ ROOM(m1) $\wedge$ ROOM(m2) $\wedge$ CONNECTS(d,m1,m2) $\wedge$
DIFF(m1,m2) $\wedge$ IN_ROOM(r,m1,s) $\wedge$ BOX(b) $\wedge$ IN_ROOM(b,m3,s) $\wedge$ $\neg$IS_CARRYING(r,b,s)]
$\Rightarrow$ IN_ROOM(b,m3,Result(MOVE(r,d),s))

5) We give a Strips operator for a robot moving while carrying a single box:

MOVE(?r,?d): /* meaning robot ?r moves through door ?d */
Preconditions: ROBOT(?r), ROOM(?m1), ROOM(?m2), CONNECTS(?d,?m1,?m2), DIFF(?m1,?m2),
BOX(?b1), BOX(?b2), BOX(?b3), BOX(?b4),
DIFF(?b1,?b2), DIFF(?b1,?b3), DIFF(?b1,?b4), DIFF(?b2,?b3), DIFF(?b2,?b4),
DIFF(?b3,?b4),
IS_CARRYING(?r,?b1), NOT_CARRYING(?r,?b2), NOT_CARRYING(?r,?b3),
NOT_CARRYING(?r,?b4),
IN_ROOM(?r,?m1)
Effects: $\neg$IN_ROOM(?r,?m1), $\neg$IN_ROOM(?b1,?m1),
IN_ROOM(?r,?m2), IN_ROOM(?b1,?m2)

We could make these more complicated or simplify them a bit. For example, we do not need to satisfy ROOM or DIFF for m1 and m2 as CONNECTS subsumes these. But the operator correctness is clearer as it is.

3. Incompleteness of linear planning

1) Attacking ON(A,B) first will result in a subgoal of clearing B (which initially supports C). C will be moved somewhere (say to the table) and with B now clear, A will be moved on top of B. But B is still on the table so we are no closer to the goal. A must be taken off of B so that B can be placed onto C.

2) Attacking ON(B,C) first is easy since B and C are both clear we simply move B to C. Unfortunately, they are both now occluding A. So again we must undo what we just did, clearing C so we can clear A so we can move it to B.

3) The obvious plan: MoveToTable(C,A,Table), MoveToBlock(B,Table,C), MoveToBlock(A,Table,B) cannot be found by a linear planner due to the interaction of subgoals. Completeness requires separate operator choice and operator scheduling decisions.