Exercisesheet No.2

Alexander Diete Magnus Müller Martin Pfannemüller

October 26, 2014

Ex.1

$$(((P \vee Q) \Rightarrow R) \wedge (R \vee (P \wedge \neg Q))) \wedge \neg R$$

Translate the implication to an or-clause:

$$((\neg(P \vee Q) \vee R) \wedge (R \vee (P \wedge \neg Q)))\neg R$$

De Morgan:

$$(((\neg P \wedge \neg Q) \vee R) \wedge (R \vee (P \wedge \neg Q))) \wedge \neg R$$

Distributivity:

$$((\neg P \vee R) \wedge (\neg Q \vee R) \wedge (R \vee P) \wedge (R \vee \neg Q)) \wedge \neg R$$

Distributivity (inverse):

$$(R \vee (\neg P \wedge \neg Q \wedge P \wedge \neg Q)) \wedge \neg R$$

Complements over P $((\neg P \wedge \neg Q \wedge P \wedge \neg Q)$ =false):

$$R \wedge \neg R$$

We are ending up with a contradiction.

Ex.2

$Init(Room(Room1) \wedge Room(Room2) \wedge Room(Room3) \wedge Room(Room4) \wedge Room(Corridor) \wedge Switch(s1) \wedge Switch(s2) \wedge Switch(s3) \wedge Switch(s4) \wedge Box(b1) \wedge Box(b2) \wedge Box(b3) \wedge Box(b4) \wedge Door(Door1) \wedge Door(Door2) \wedge Door(Door3) \wedge Door(Door4) \wedge At(Shakey, Floor) \wedge In(Shakey, Room3) \wedge$

$TurnedOn(s4) \wedge TurnedOff(s3) \wedge TurnedOff(s2) \wedge TurnedOn(s1) \wedge In(b1, Room1) \wedge In(b2, Room1) \wedge In(b3, Room1) \wedge In(b4, Room1) \wedge At(s1, Room1) \wedge At(s2, Room2) \wedge At(s3, Room3) \wedge At(s4, Room4) \wedge In(Door1, Room1) \wedge In(Door1, Corridor) \wedge In(Door2, Room2) \wedge In(Door2, Corridor) \wedge In(Door3, Room3) \wedge In(Door3, Corridor) \wedge In(Door4, Room4) \wedge In(Door4, Corridor))$

Action(Go(x,y,r)),
  PRECOND: $At(Shakey, x) \wedge In(x, r) \wedge In(y, r)$
  EFFECT: $At(y, Shaky) \wedge \neg At(x, Shaky)$

Action(Push(b,x,y,r)),
  PRECOND: $At(b, x) \wedge In(x, r) \wedge In(y, r) \wedge In(Shakey, r) \wedge \neg At(Shakey, x) \wedge Box(b)$
  EFFECT: $At(b, y) \wedge \neg At(b, x)$

Action(ClimbUp(x,b)),
  PRECOND: $In(b, r) \wedge In(x, r) \wedge At(Shakey, x) \wedge \neg At(b, x) \wedge On(Shakey, Floor)$
  EFFECT: $\neg On(Shakey, Floor) \wedge On(Shakey, b) \neg At(Shakey, x)$

Action(ClimbDown(b,x)),
  PRECOND: $In(x, r) \wedge In(b, r) \wedge \neg At(b, x) \wedge On(Shakey, b)$
  EFFECT: $\neg On(Shakey, Floor) \wedge On(Shakey, b) \neg At(Shakey, x)$

Action(TurnOn(s,b)),
  PRECOND: $On(Shakey, b) \wedge \neg On(Shakey, Floot) \wedge At(b, s) \wedge At(Shakey, s)$
  EFFECT: $TurnedOn(s)$

Action(TurnOff(s,b)),
  PRECOND: $On(Shakey, b) \wedge \neg On(Shakey, Floot) \wedge At(b, s) \wedge At(Shakey, s)$
  EFFECT: $TurnedOff(s)$

```
Plan:
Go(X,Door3,Room3)
Go(Door3,Door1,Corridor)
Go(Door1,Box2,Room1)
Push(Box2,Box2,Door1,Room1)
Push(Box2,Door1,Door2,Corridor)
```

Ex.3

See figure 1.

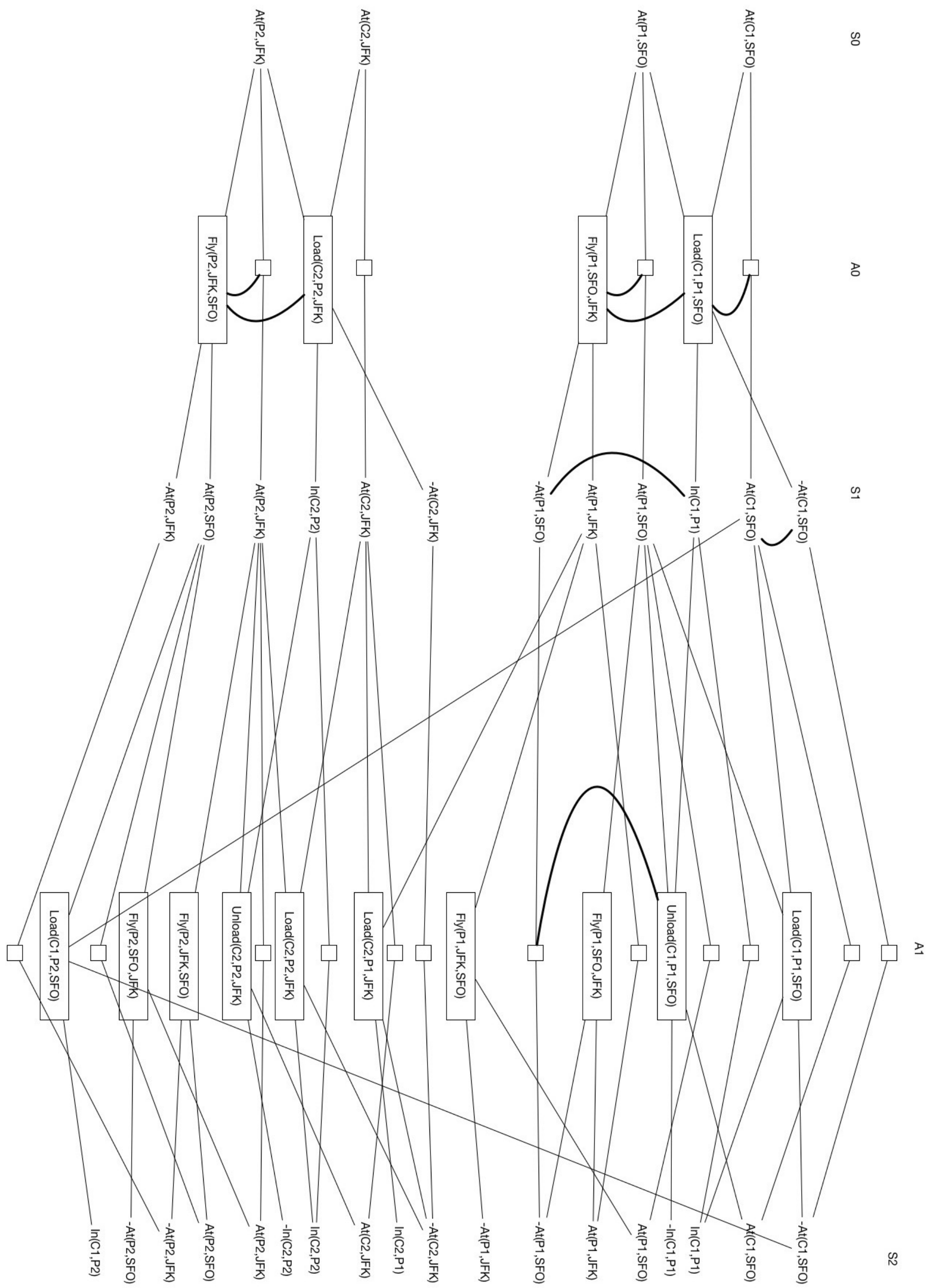
S0
A0
S1
A1
S2
At(C1,SFO)
At(P1,SFO)
At(C2,JFK)
At(P2,JFK)
Load(C1,P1,SFO)
Fly(P1,SFO,JFK)
Load(C2,P2,JFK)
Fly(P2,JFK,SFO)
-At(C1,SFO)
At(C1,SFO)
In(C1,P1)
At(P1,SFO)
At(P1,JFK)
-At(P1,SFO)
-At(C2,JFK)
At(C2,JFK)
In(C2,P2)
At(P2,JFK)
At(P2,SFO)
-At(P2,JFK)
Load(C1,P1,SFO)
Unload(C1,P1,SFO)
Fly(P1,SFO,JFK)
Fly(P1,JFK,SFO)
Load(C2,P1,JFK)
Load(C2,P2,JFK)
Unload(C2,P2,JFK)
Fly(P2,JFK,SFO)
Fly(P2,SFO,JFK)
Load(C1,P2,SFO)
-At(C1,SFO)
At(C1,SFO)
In(C1,P1)
-In(C1,P1)
At(P1,SFO)
At(P1,JFK)
-At(P1,SFO)
-At(P1,JFK)
-At(C2,JFK)
In(C2,P1)
At(C2,JFK)
In(C2,P2)
-In(C2,P2)
At(P2,JFK)
At(P2,SFO)
-At(P2,JFK)
-At(P2,SFO)
In(C1,P2)

Figure 1: Ex.3

Ex.4

```
Primitive actions where t is truck and l is load:
Forward(t);
TurnLeft(t);
TurnRight(t);
Load(l,t)
Unload(l,t)

We have the following high level actions in the grid map with
x and y as start and a and b as destination:
Move(t, x, y);
Transport(l, t, x, y, a, b);

Refinements:
Transport(l,t,x,y,a,b)
    PRECOND: Truck(t) AND Load(l) AND At(l,x,y)
    STEPS: Move(t,x,y), Load(l,t), Move(t,a,b), Unload(l,t)

Move(t,x,y)
    PRECOND: Truck(t) AND At(t,x,y)
    STEPS:

Move(t,x,y)
    PRECOND: Truck(t)
    STEPS: Forward(t)

Move(t,x,y)
    PRECOND: Truck(t)
    STEPS: TurnLeft(t)

Move(t,x,y)
    PRECOND: Truck(t)
    STEPS: TurnRight(t)
```

Ex.5

We need an action which has an effect that is dependant on the evaluation of a condition (like in if-statements from programming languages).

```
Move(b,x,y)
```

```
    PRECOND: On(b,c) AND Clear(b) AND Clear(y)
    EFFECTS: if y!=Table
                Then On(b,y) AND Clear(x) AND ¬On(b,x) AND ¬Clear(y)
             else
                On(b,y) AND Clear(x) AND ¬On(b,x)
```

Ex.6

a)

```
Drink(p)
    PRECOND: Patient(p)
    EFFECTS: ¬Dehydrated(p)

Medicate(p)
    PRECOND: Patient(p) AND Disease(D)
    EFFECTS: if(has(p,D))
                then Cured(p)
             else
                SideEffect(p)
```

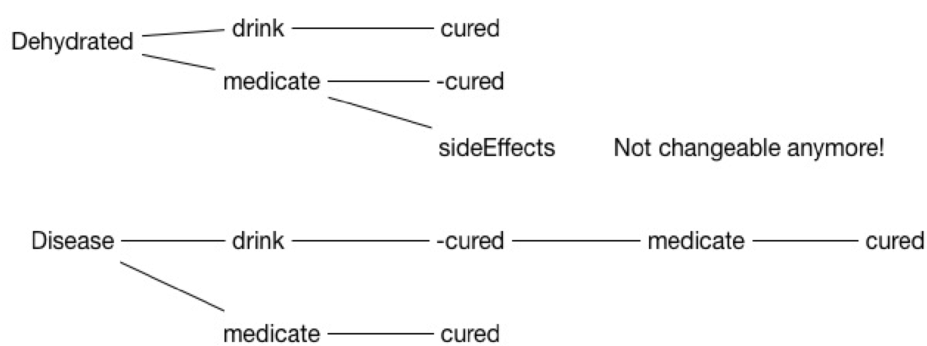

Figure 2: Since we cannot remove the side effects, we do not continue the top path

b)

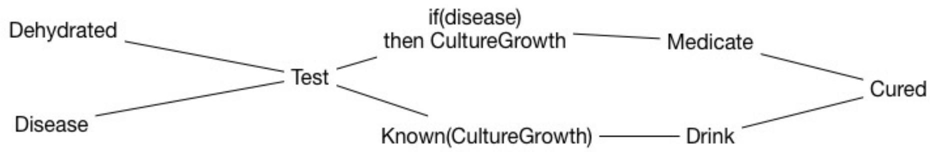

Figure 3: Conditional plan that solves the problem

Ex.7

In order so to solve this exercise we have to first transform the PDDL in a Form, that can be processed by a SATPlaner. This is described in the Artificial Intelligence Book, Chapter 10.4.1. Hence we transform goal and initial state, the successor state axiom, precondition and actions exclusion axioms into propositional logic.

Init:

$$CapOn^0 \land \neg SimIn^0 \tag{1}$$

Goal:

$$CapOn^t \land SimIn^t \tag{2}$$

Successor state axiom:

$$CapOn^{t+1} \Leftrightarrow PutCapOn^t \lor (CapOn^t \land \neg RemoveCap^t) \tag{3}$$
$$\neg CapOn^{t+1} \Leftrightarrow RemoveCap^t \lor (\neg CapOn^t \land \neg PutCapOn^t) \tag{4}$$
$$SimIn^{t+1} \Leftrightarrow InsertSim^t \lor (SimIn^t) \tag{5}$$

Preconditions:

$$PutCapOn^t \Rightarrow \neg CapOn^t \tag{6}$$
$$RemoveCap^t \Rightarrow CapOn^t \tag{7}$$
$$InsertSim^t \Rightarrow \neg SimIn^t \land \neg CapOn^t \tag{8}$$

ActionsExclusion:

$$PutCapOn^t \Rightarrow \neg(RemoveCap^t \lor InsertSim^t) \tag{9}$$
$$RemoveCap^t \Rightarrow \neg(PutCapOn^t \lor InsertSim^t) \tag{10}$$
$$InsertSim^t \Rightarrow \neg(RemoveCap^t \lor PutOnCap^t) \tag{11}$$

These rules have to be converted to CNF in order to be processable by a SATPlaner. As the goal and initial state already fulfill that form, they do not have to be transformed.We used an online tool (Wolfram Alpha) to speed up the process.

Successor State:
CapOn:
$(\neg CapOn^{t+1} \vee PutCapOn^{t} \vee CapOn^{t}) \wedge (\neg CapOn^{t+1} \vee PutCapOn^{t} \vee \neg RemoveCap^{t}) \wedge (CapOn^{t+1} \vee \neg PutCapOn^{t}) \wedge (CapOn^{t+1} \vee \neg CapOn^{t} \vee RemoveCap^{t})$

Not CapOn:
$(\neg CapOn^{t+1} \vee \neg RemoveCap^{t}) \wedge (\neg CapOn^{t+1} \vee CapOn^{t} \vee PutCapOn^{t}) \wedge (CapOn^{t+1} \vee RemoveCap^{t} \vee \neg CapOn^{t}) \wedge (CapOn^{t+1} \vee RemoveCap^{t} \vee \neg PutCapOn^{t})$

InsertSim:
$(\neg SimIn^{t+1} \vee InsertSim^{t} \vee SimIn^{t}) \wedge (SimIn^{t+1} \vee \neg InsertSim^{t}) \wedge (SimIn^{t+1} \vee \neg SimIn^{t})$

Preconditions:
PutCapOn:
$(\neg PutCapOn^{t} \vee \neg CapOn^{t})$

RemoveCap:
$(\neg RemoveCap^{t} \vee CapOn^{t})$

InsertSim:
$(\neg InsertSim^{t} \vee \neg SimIn^{t}) \wedge (\neg InsertSim^{t} \vee \neg CapOn^{t})$

Action Exlusion:
PutCapOn:
$(\neg PutCapOn^{t} \vee \neg RemoveCap^{t}) \wedge (\neg PutCapOn^{t} \vee \neg InsertSim^{t})$

RemoveCap:
$(\neg RemoveCap^{t} \vee \neg PutCapOn^{t}) \wedge (\neg RemoveCap^{t} \vee \neg InsertSim^{t})$

InsertSim:
$(\neg InsertSim^{t} \vee \neg RemoveCap^{t}) \wedge (\neg InsertSim^{t} \vee \neg PutCapOn^{t})$

The next thing to to is replace the time-variables with concrete variables. In order to do this we have to have an estimation of the length of the plan. With heuristics from the lecture (i.e. level-sum) we can assume a plan of length three. Also at the same time, we replaced the variable-names with numbers to fit the DIMACS format. We came up with this mapping:

```
CapOn0 1
CapOn1 2
CapOn2 3
CapOn3 4
SimIn0 5
SimIn1 6
SimIn2 7
SimIn3 8
PutCapOn0 9
PutCapOn1 10
PutCapOn2 11
RemoveCap0 12
RemoveCap1 13
RemoveCap2 14
InsertSim0 15
InsertSim1 16
InsertSim2 17
```

The last task is to write down all formulas in cnf form using the mapped values and solving them with a SATPLaner. We did just that (manually keeping the number of formulas down, by excluding redundant formulas) and got a result plan. The planer returned a set of literals that satisfies all the clauses.

Clauses in DIMACS:

```
p cnf 17 55
1 0
-5 0
4 0
8 0
-2 9 1 0
-3 10 2 0
-4 11 3 0
-2 9 -12 0
-3 10 -13 0
-4 11 -14 0
2 -9 0
3 -10 0
4 -11 0
-2 -12 1 0
-3 -13 2 0
-4 -14 3 0
-2 -12 0
-3 -13 0
-4 -14 0
2 12 -1 0
3 13 -2 0
4 14 -3 0
2 12 -9 0
3 13 -10 0
4 13 -11 0
-6 15 5 0
-7 16 6 0
-8 17 7 0
6 -15 0
7 -16 0
8 -17 0
6 -5 0
7 -6 0
8 -7 0
-9 -1 0
-10 -2 0
-11 -3 0
```

```
-12 1 0
-13 2 0
-14 3 0
-15 -5 0
-16 -6 0
-17 -7 0
-15 -1 0
-16 -2 0
-17 -3 0
-9 -12 0
-10 -13 0
-11 -14 0
-9 -15 0
-10 -16 0
-11 -17 0
-12 -15 0
-13 -16 0
-14 -17 0
```

Result:

```
This is MiniSat 2.0 beta
============================[ Problem Statistics ]=============================
| |
| Number of variables: 17 |
| Number of clauses: 55 |
| Parsing time: 0.00 s |
============================[ Search Statistics ]==============================
| Conflicts | ORIGINAL | LEARNT | Progress |
| | Vars Clauses Literals | Limit Clauses Lit/Cl | |
===============================================================================
| 0 | 9 24 55 | 8 0 nan | 0.000 % |
===============================================================================
Verified 24 original clauses.
restarts : 1
conflicts : 0 (0 /sec)
decisions : 3 (0.00 % random) (3 /sec)
propagations : 17 (17 /sec)
conflict literals : 0 ( nan % deleted)
CPU time : 1 s

SATISFIABLE
v 1 -2 -3 4 -5 -6 7 8 -9 -10 11 12 -13 -14 -15 16 -17 0
```

So the plan generated suggests to: first remove the cap (12), then insert the sim (16) and then close the cap (11).