Principles of AI Planning
Summer 2004

Institut für Informatik
Universität Freiburg
Rintanen

Exercise 2

To be returned on Monday, May 10, 2004

Assignment 2.1

Prove the following equivalences on effects:

$$\begin{aligned} c \rhd (e_1 \wedge e_2) &\equiv (c \rhd e_1) \wedge (c \rhd e_2) \\ c \rhd (c' \rhd e) &\equiv (c \wedge c') \rhd e \\ (c \rhd e) \wedge (c' \rhd e) &\equiv (c \vee c') \rhd e \end{aligned}$$

Note: Two effects e and e' are equivalent iff their application in identical states results in identical states, i.e. iff $[e]_s = [e']_s$ for all states s.

Assignment 2.2

Consider the following situation: Romeo and Juliet are at home.

$$I(p) = 1 \text{ iff } p \in \{\text{romeo-at-home}, \text{juliet-at-home}\}$$

Juliet wants to go dancing, but Romeo wants to stay at home.

$$G = \text{juliet-dancing} \wedge \text{romeo-at-home}$$

Since this is a real couple, Romeo can't just say that he doesn't want to go dancing – if Juliet goes dancing and he is at home, he has to join her. This is modelled by the following operator:

$$\begin{aligned} \text{go-dance} = \langle\ & \text{juliet-at-home}, \\ & \text{juliet-dancing} \wedge \neg\text{juliet-at-home} \wedge \\ & (\text{romeo-at-home} \rhd (\text{romeo-dancing} \wedge \neg\text{romeo-at-home}))\rangle \end{aligned}$$

Of course, Romeo can always pretend he has work to do:

$$\text{go-work} = \langle \text{romeo-at-home}, \text{romeo-at-work} \wedge \neg\text{romeo-at-home} \rangle$$

Since he would not want to stay at work forever, we must also model the inverse operator:

$$\text{go-home} = \langle \text{romeo-at-work}, \text{romeo-at-home} \wedge \neg\text{romeo-at-work} \rangle$$

We thus obtain the planning problem

$$\begin{aligned}\langle\quad & \{\text{romeo-at-home}, \text{romeo-dancing}, \text{romeo-at-work},\\ & \ \text{juliet-at-home}, \text{juliet-dancing}\},\\ & I, \{\text{go-dance}, \text{go-work}, \text{go-home}\}, G\rangle\end{aligned}$$

Solve this problem with a breadth-first search using the *regression* method. Submit the search tree that you obtain and record the solution plan. At every node of the search tree, simplify the state formula as much as possible and do not expand the node further if that formula is unsatifiable or identical to a previously encountered node. At the solution layer, you only need to include the node that represents a set of states containing the initial state – you can assume that the breadth first search algorithm is lucky.