Foundations of Artificial Intelligence

Prof. Dr. B. Nebel, Prof. Dr. W. Burgard
B. Frank, A. Karwath, G. Röger
Summer Term 2009

University of Freiburg
Department of Computer Science

Exercise Sheet 7

Due: Tuesday, June 23, 2009

Exercise 7.1 (Situation Calculus and Golog)

Action languages are used to program a system's (eg. a robot's) behaviour by describing it in a high-level manner. Golog is one such language that is (in its theoretical aspects) based on the situation calculus. The implementation of Golog is based on Prolog (hence the name).
IndiGolog is a variant of Golog that allows sensing and exogenous events and which is executed online (the actions of the system are not precomputed before the execution starts).
You find a tarball containing IndiGolog on the webpage. Since there is only a Linux implementation available, you will need a Linux installation or to use one of the computers in the pool. In the pool all necessary additional software is already installed. On your personal computer you might have to install additional software (SWI-Prolog, consult the exercise webpage).
On the exercise webpage you can also find some information about how to execute the Wumpus world example contained in the tarball.

(a) Execute the Wumpus example. Afterwards, have a look at the file `wumpus.pl` (you need not to understand everything) and find out what the following parts mean:

- `poss(moveFwd, neg(inTheEdge(locRobot,dirRobot))).`
- `causes(turn, dirRobot, Y, rotateRight(dirRobot,Y)).`
- `interrupt(isGold(locRobot)=true, [pickGold])`

Probably you can find the correct answer by a closer look and some thinking. If not, a search in the following paper should help:

`http://www.dis.uniroma1.it/~degiacom/papers/2009/IndiGologChapter09.pdf`.

(b) When observing the Action history and Sensing results in the applet, you notice that the robot still performs `smell` actions, also if it already has shot the Wumpus. Modify the procedure `mainControl(4)` in such a way that the robot no longer performs these unnecessary actions. (Add an `if` statement that test whether the Wumpus is still *alive*. For the else part you need an empty action which you can simulate using `?(true)`.)

(c) Up to now, the robot can only turn clock-wise (using the `turn` action). There are already stubs for new actions `turnRight` and `turnLeft`, but these are currently not executable (and not fully implemented). Make these actions executable and the original `turn` action non-executable by modifying their preconditions. Add code that specifies the effect of these new actions. For this purpose, you will need a new predicate `rotateLeft` analogous to `rotateRight` for action `turnLeft`. Up to now, the `turn` action was used by the procedures `move(D)` and `shoot(D)`. Procedure `move(D)` must be read as follows: Search for an action sequence that executes `turn` at most 4 times to make `dirRobot=D` true. After that, perform action `moveFwd`.

Your task in this exercise is to modify these procedures so that they use the new actions instead of `turn`. Do this in such a way that the robot performs as few turns as possible (within these procedures).

To solve this, use `ndet(actions1,actions2)` which allows the search for the action sequence to choose whether to perform `actions1` or `actions2`. Don't use the test action (the one starting with the questionmark) inside `ndet`.

Test your implementation by executing the program.

Your submission for parts (b) and (c) (together) should contain all changes to file `wumpus.pl` and the *Action history and Sensing results* of a run of the final program version.

Exercise 7.2 (Allen's Interval Calculus)

(a) In general, the composition of two binary relations R and S (over X) is defined as

$$R \circ S = \{(x, z) \mid \exists y \in X \text{ such that } (x, y) \in R \text{ and } (y, z) \in S\}.$$

Allen's interval calculus is *closed under composition* which means that every composition of Allen relations (also for unions of the 13 base relations) can be represented as union of base relations. For example, $f \circ s = d$ because for arbitrary intervals A, B and C with AfB and BsC it must hold that AdC. Note that in general the composition of two base relations needs not to result in a single base relation, as you can see from the example $f^{-1} \circ d = (o, d, s)$. Determine the following compositions:

(1) $o \circ m$

(2) $m \circ f$

(3) $(o, m) \circ f$

(b) The composition is also used for the constraint propagation technique. Use this technique to make the following constraint network 3-consistent.

Hint: If there is no directed edge from one interval to another one, this implicitly implies that the all-relation (the union of all 13 base relations) holds for these intervals. You may use in your solution that $(IRJ, ISJ)^{-1} = (IR^{-1}J, IS^{-1}J)$ (for intervals I, J and R, S base relations of Allen's interval calculus).

$$\begin{tikzcd}
\text{A} \arrow[r, "{(<,=,>)}"] \arrow[d, "{(o,d^{-1},f^{-1})}"'] \arrow[dr, "m"] & \text{B} \\
\text{C} \arrow[r, "{(o,s,d)}"'] & \text{D} \arrow[u, "{(<,s^{-1})}"']
\end{tikzcd}$$

Exercise 7.3 (Decision Trees)

(a) Specify decision trees representing the following Boolean functions:

(1) A XOR B

(2) $(A \land B) \lor (C \land D)$

(b) Here we will practice the basic information-theoretical concepts used to build decision trees. Consider the following set of training examples:

a_1	a_2	Classification
T	T	+
T	T	+
T	F	-
F	F	+
F	T	-
F	T	+

What is the information content of this collection of training examples with respect to the target function *Classification*? What is the information gain of a_2 relative to these training examples?

The exercise sheets may and should be handed in and be worked on in groups of three (3) students. Please fill the cover sheet[1] and attach it to your solution.

[1]`http://www.informatik.uni-freiburg.de/~ki/teaching/ss09/gki/coverSheet-english.pdf`