Artificial Intelligence – Programming Assignment 2

UNIZG FER, academic year 2014/15

Handed out: 15.4.2015. Due: 5.5.2015. do 23.59 sati.

Star Trek Wumpus propositional logic problem

Star Trek Fleet's USS Enteprise's captain Jean-Luc Picard has found humself in trouble once again. While Enterprise was passing near the Wumpusworld planet, a vicious creature called Wumpus the Hutt beamed Picard to the its cave on the planet. Wumpus feeds himself with living things unintentionally passing by his planet. Wumpus is, however, unable to move and it can eat only the things in its immediate proximity. Wumpus's cave contains many pits and falling into any of them takes one to a certain death in the boiling center of the Wumpusworld planet. There is also a fully functional teleporter somewhere in the cave and Picard could use it to get back to Enterprise if he could reach it without passing by Wumpus or falling into a pit on the way. Given it is very dark in the cave, Picard must rely on other senses like touch and smell in order to safely reach the teleporter.

Wumpus's cave can be represented as a 2D grid (matrix) of size N×N. Each cell in the grid is marked with exactly one of the following: *"W"* (the cell which contains Wumpus the Hutt), *"P"* (a cell containing a pit), *"T"* (a cell containing a teleporter) and *"O"* (regular cell – no Wumpus, no pit, and no teleporter). On all cells adjacent to the cell where Wumpus is (we consider only edge adjacency, not vertex adjacency Picard can smell obnoxious stench of Wumpus, whereas on all cells adjacent to any of the pits Picard can sense a hot breeze of magma rising from the pit. The cell containing a teleporter emits glow to all its adjacent cells (but the cell containing a teleporter does not glow itself). An example of the Wumpus's cave is given in Figure 1.

Your task is to help Picard find a safe way to the teleporter by applying the refutation resolution in propositional logic (finding a safe way, depending on the cave configuration, may or may not be achievable). In each step, Picard tries to infer the marks for cells adjacent to the one he is currently at by using the information collected before (e.g., marks of cells he already visited) and the information stemming from his senses, i.e., information about the stench, breeze, or glow he notices at the current cell. Sometimes Picard may not be able to infer the mark for some adjacent cell because of insufficient information. Picard moves across the cave (i.e., performs transitions from one cell to another) according to the following rules (rules are listed according to their priority):

1. If there is an adjacent cell for which Picard is able to infer the mark *"T"*, Picard moves to that cell because it contains a telerpoter and allows him to get back to Enterprise;

2. If there is an adjacent cell for which Picard is able to infer the mark *"O"* (equivalently, Picard can infer that *not "T" and not "P" and not "W"*), Picard moves to that safe cell. In case of more than one safe cell surrounding the current cell, Picard

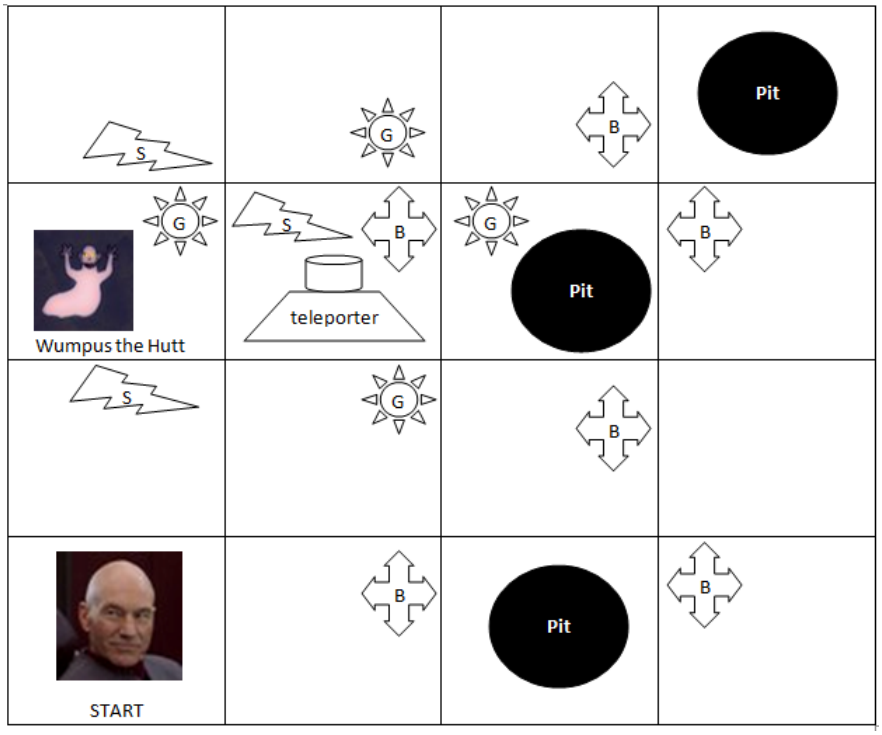

Figure 1. Wumpus's cave

chooses to move to the one with the lowest position index (assuming the position is a pair (x, y), Picard moves to the cell for which the value $10x + y$, i.e. position index, is the smallest);

3. If there is an adjacent cell for which Picard is unable to infer the mark, Picard moves to that cell. If there is more than one such cell, Picard moves to the one with the lowest position index;

4. Picard realizes that he cannot move anywhere (if none of the previous rules is satisfied, this means Picard is surrounded with cells leading to certain death – cells with pits and a cell with Wumpus), thus he stays at his current position, waiting to be rescued by Enterprise.

If his tragic destiny should let Picard to move to the cell with the Wumpus or a pit (by applying the rule 3, since he is unable to infer the mark of the cell in advance), your program should print out the message about Picard's tragic death and stop. Also, if Picard should stumble upon the cell containing the teleporter your program should print a message of his successful return to Enterprise and stop.

The module performing the refutation resolution for propositional logic should be implemented independently of its application to the Wumpusworld problem, as described beneath.

Refutation resolution

You may assume that all the formulas of prepositional logic that will be given to the refutation resolution module will already be in conjunctive normal form. Put differently,

you do not need to implement transformation of an arbitrary propositional logic formula to CNF.

Implement a theorem prover in propositional logic based on refutation resolution. Use the set-of-support strategy as the resolution strategy. The prover takes as input a set of premises $F_1, \ldots, F_n$ and a goal formula G. The function returns *true* if the goal formula is deductively follows from the premises, and *false* otherwise.
Make sure that you never resolve the same pair of clauses more than once and that you don't generate clauses that are already generated.
You should also implement a simplification strategy as follows. The strategy should eliminate all redundant and irrelevant clauses from the set of clauses after each application of the resolution rule. Removing redundant clauses is based on the absorption equivalence $F \wedge (F \vee G) \equiv F$. Assuming that clauses are represented as sets of literals, if the set of clauses contains a pair of clauses C_1 and C_2 for which $C_1 \subseteq C_2$, then clause C_2 may be removed from the set. Removing irrelevant clauses amounts to removing all clauses that are valid formulae. A clause is valid iff it contains a complementary pair of literals.

Putting Wumpusworld and refutation resolution together

Implement a wrapper function that takes as input the set of premises and the goal formula (in CNF format) and outputs whether the goal can be inferred from the premises. You'll implement calls to this wrapper function from the main program which navigates Picard from through the cave.

Each cell of the cave grid, (x, y), can be described using six propositional logic literals:

- $S_{(x,y)}$ – the stench of Wumpus is felt at the cell (x, y);
- $B_{(x,y)}$ – a breeze of a pit is felt at the cell (x, y);
- $G_{(x,y)}$ – a glow of the teleporter is seen at the cell (x, y);
- $W_{(x,y)}$ – Wumpus is found at the cell (x, y);
- $P_{(x,y)}$ – A pit is found at the cell (x, y);
- $T_{(x,y)}$ – The teleporter is found at the cell (x, y).

The following logical formulas (valid for each cell of the cave grid) stem directly from the previously described rules of the Wumpusworld:

- If Picard smells stench at a cell then one of the adjacent cells contains Wumpus, i.e., $S_{(x,y)} \rightarrow (W_{(x-1,y)} \vee W_{(x+1,y)} \vee W_{(x,y-1)} \vee W_{(x,y+1)})$;
- If Picard feels hot breeze at a cell then at least one of adjacent cells contains a pit, i.e., $B_{(x,y)} \rightarrow (P_{(x-1,y)} \vee P_{(x+1,y)} \vee P_{(x,y-1)} \vee P_{(x,y+1)})$;
- If Picard sees glow at a cell then one of the adjacent cells contain the teleporter, i.e., $G_{(x,y)} \rightarrow (T_{(x-1,y)} \vee T_{(x+1,y)} \vee T_{(x,y-1)} \vee T_{(x,y+1)})$
- If a cell contains Wumpus then no other cell contains Wumpus, i.e., $W_{(x,y)} \rightarrow (\neg W_{(x',y')})$ (for every (x', y') other than (x, y)).

You can convert the above rules to CRF manually. The above set contains only the basic rules that hold in Wumpusworld. You are free to design other formulas which hold in Wumpusworld and could potentially facilitate inference, but this is not a mandatory part of the assignment. For example, what might we infer from the fact that Picard can smell stench on two vertex-adjacent cells (knowing that Wumpus can only be found in one cell)? When generating the abovementioned formulas for different positions of the cave grid make sure to account for the dimensions of the map (x and y must bein between 1 and N).

Input and output format

In artificial intelligence terminology, this is the problem of an agent (Picard) in the agent environment (Wumpus's cave). We thus need to distinguish the representation of the environment (the cave map as given in the input file) from the representation of the agent's perception of the environment (what Picard knows of the Wumpus's cave).

The program needs to load the configuration of the cave grid from file. The grid of the cave will be given so that every line of the file describes a single cell of the grid. In each row you will first find the coordinates of the cell, (x, y), and then the values of $S_{(x,y)}$, $B_{(x,y)}$, and $G_{(x,y)}$ (i.e., information about presence of stench, breeze, and glow at the cell) – this data is part of Picard's perception of the environment (what agent knows about the environment). Finally, in each row you will be given the values for literals $W_{(x,y)}$, $P_{(x,y)}$, and $T_{(x,y)}$ (i.e., is there Wumpus, a pit, or the teleporter on the cell), but these values are not known to Picard, i.e., you **may NOT use them as facts for inference**, unless Picard inferred them beforehand – these values are part of the representation of the environment, but not of agent's perception of the environment. Put differently, you can use values $W_{(x,y)}$, $P_{(x,y)}$, and $T_{(x,y)}$ strictly to check if the program should finish due to Picard stepping on the cell with the teleporter, Wumpus, or pit (see the 3rd rule of moving above). Assume Picard is always positioned at cell $(1, 1)$.

```
(1,1) S=0 B=0 G=0 W=0 P=0 T=0
(1,2) S=1 B=1 G=0 W=0 P=0 T=0
(2,1) S=1 B=0 G=1 W=0 P=0 T=0
...
```

The format of the output is not strictly defined, but the program must output the route of Picard's movement in the cave. Also, the resolution steps for each started refutation resolutions should be printed on standard output.