Implementation of Minesniffer in Java and Prolog

Problem Description

Minesniffer game contains a rectangular grid where each cell is one of a mine or an integer value or empty containing no hint or containing a X. If a cell has value X, then it may or may not have a mine. All other cells can not contain a mine. If the value is an integer, it represents the number of adjacent cells that contain a mine. The value can range from 0 to 8. With this setup, we can use inference to determine if a node contains a mine.

Inference in Propositional Logic using Resolution

If a cell has an integer value k, then any k of its neighbours have a mine in their cells and the rest dont. M_i is the sentence that represents cell i having a mine. Consider a node at cell P which has m neighbours in the grid. Consider a single combination of k neighbours of a given cell. The fact that the cell has value k indicates that this could be a valid combination where all neighbours in the set have mines, while all neighbours not in the set do not have mines at the positions. This can be represented by a suitable conjunction easily. The correct combination can be any one of the m choose k combinations, this can be represented by a disjunction of individual conjunctions. This forms a propositional sentence with respect a single cell P. We can form all such sentences with special cases for some values. If the value of a cell Q is NH, then we only know that the cell does not have a mine so the only sentence formed is NOT M_Q. If a node contains X, the other cases do not apply and we would like to check if the node has a mine through inference.

The sentences for each cell can be combined by conjunction in the knowledge base. Using any proof technique we can infer sentences in Propositional Logic. Our Java code makes use of ttEntails method of ttEntailment for Propositional Logic. This involves conversion of the sentence to Conjunctive Normal Form. We know that if there are n atomic premises in a sentence, the equivalent sentence in CNF has 2^n atomic premises. Computing the CNF is essential for resolution, thus for even small dimensions of the grid, the method takes too long to compute CNF, thus it may not provide an answer for inference within acceptable time limits.

Inference Using First Order Logic in XSB

XSB is a Prolog like declarative programming language. The facts are represented by predicates and the rules are written as a special form of horn clauses. Prolog systems use a goal based reasoning approach (backward chaining), which attempts to attain a goal by the process of attaining a sequence of sub goals.

The rules for inferring the position of a mine in first order logic makes use of the below ideas.

1. If a cell has value k and only k of its neighbouring cells have value X, then all of these k neighbouring nodes have a mine.
2. If a cell has value k and all k mines have been located, then all other neighbouring cells that have a value X, do not have mines.
3. If a cell has value k and m mines, (m<k) have been located and n neighbouring cells with value X do not have neighbours. and there are p neighbouring cells with value X, but where the position of mines have not been established, and k = m+p , then the remaining p cells also have mines at their positions.

By repeated application of the above three rules, all cells that can be inferred without any ambiguity can be identified. This process can be continued till a fixed point is reached in the last step, after which the process of finding mines in the minesniffer grid terminates.

Even after a fixed point has been reached there are some sentences that may not be entailed from the knowledge base. Consider the grid configuration given below.

```
2,2
1,1
X,X
```

For this input, the sentences indicating the occurrence of mines at (2,1) and (2,2) cannot be entailed from the knowledge base. Such configurations do not have solutions from some cells, which can be solved by neither First Order nor Propositional logic.

Sample Run:

Case 1: Using Propositional Logic

```
3,5

1,2,2,1,NH

1,X,X,1,NH

1,2,2,1,NH


P22 – True
P23 – True
P11 – False
```

Case 2: Using First Order Logic

```
8,8

x,x,nh,nh,1,x,1,nh

1,2,3,2,2,1,1,nh

1,x,x,x,2,2,2,1

1,2,3,2,2,x,x,1

nh,nh,nh,nh,1,2,3,2

1,1,1,nh,nh,1,2,x

1,x,1,1,1,2,x,2

1,1,1,1,x,2,nh,nh
```

mine(1,6,x) – True
mine(3,2,x) – True
mine(3,3,x) – True
mine(3,4,x) – True
mine(4,6,x) – True
mine(4,7,x) – True
mine(6,8,x) – True
mine(7,2,x) – True
mine(7,7,x) – True
mine(8,5,x) – True
mine(1,1,x) – False
mine(1,2,x) – False