Examination: Modelling of Complex Systems

23/01/2019

1 Theory Small questions (4pts)

1. Given an FO theory T and sentence φ, it holds that T logically entails φ iff $T \cup \{\neg\varphi\}$ is unsatisfiable. Prove this.

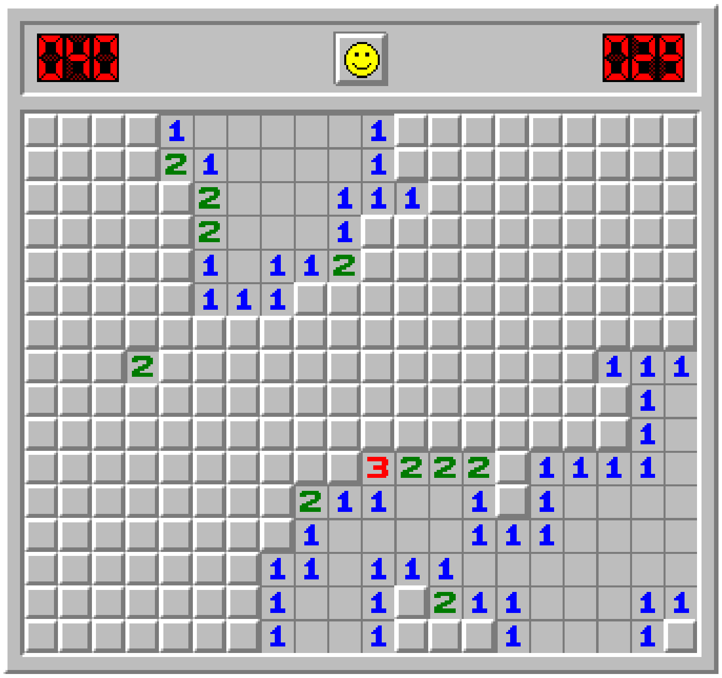
   Proof. $\Rightarrow$ Assume $T \models \varphi$. By the definition of $\models$, in every model of T, φ is true, and hence $\neg\varphi$ is false. Hence, there is no model of T and $\neg\varphi$. By the definition of unsatisfiability, $T \cup \{\neg\varphi\}$ is unsatisfiable.

   $\Leftarrow$ Assume $T \cup \{\neg\varphi\}$ is unsatisfiable. Take any model $\mathfrak{A}$ of T. It does not satisfy $\neg\varphi$. By definition of satisfaction, $\neg\varphi$ is false in $\mathfrak{A}$, and hence, φ is true in $\mathfrak{A}$. It follows that φ is true in every model of T. By definition of entailment, T entails φ.

2. For the other small questions, see the course

2 Exercise part (7+5pts)

2.1 Minesweeper (5pts)



Most of you know the minesweeper problem. In such problem, the player tries to guess the positions of B bombs on a rectangular $M \times N$ board. He clicks on a cell which then reveals either a bomb in the cell (and then the game is over) or otherwise, the number of bombs in the (at most) 8 surrounding cells. This means that at any time that the game is not lost, the board looks like in the figure. The game is won if every bomb-free cell is clicked, and no bombed cell.

We are modelling one state of the game from the point of view of the player. Given is M, N, B and a predicate that for a number of revealed positions specifies the number of bombs in adjacent positions: *ShownNrOfBombs(row,column,0..8)*.

1. Design a theory T_{mine} about the unknown predicate *BombAt(row,column)*, whose models are exactly all possible configurations of the minefield that are consistent with the information given by the data predicate *ShownNrOfBombs*. You can use FO or any extension that we have seen in the course.

2. Consider the following minesweep-player:

```
Initiate empty partial structure Minefield
While True do {
  compute the certain bomb positions in  Minefield
```

```
  compute the certainly safe positions  in  Minefield

  if there is at least one safe position
  then select a safe position; click it
  else select an unknown position;  click it.

  read input
  if Input = ‘‘Won’’
       then Write’’I won!’ ; return
  else if Input = ‘‘Boem!’’
       then  Write ‘‘Ai! I’m dead!’’ ; return
  else add Input  to MineField
       // in the latter case, Input is a set of data items about
       // bomb-free positions and the number of bombs they are
       // surrounded with

}
```

Describe how to implement step 2) of the while loop: the computation of safe positions, using inference on your theory and the current partial structure `Minefield`.

3. Now we change the perspective to the game console. Given a predicate *Click(row,column)* that contains a set of clicked positions and the predicate *BombAt(row,column)*, define the predicate *ShownNrOfBombs(row,column,nr)* which determines which positions are revealed and the number of bombs that surround them. A position is revealed if it is bomb-free and there is a path from a clicked position to it such that all positions on the path except possibly the last one are bomb-free and have zero neighboring bombs.

Solution

1. The theory:

```
{ ! r c u v: Adjacent(r,c,u,v) <- Abs(r-u)=<1 & Abs(c-v)=<1 & ~(r-u & c=v).}

#{ (r,c) : BombAt(r,c)} = B.
! r c: ShowNrOfBombs(r,c,n) => n =
          { (r1,c1) : BombAt(r1,c1) & Adjacent(r,c,r1,c1) }.
! r c:  ShowNrOfBombs(r,c,n) => ~BombAt(r,c).
```

2. The form of inference: Use maximal propagation inference.

   - Input: T, the partial structure $\mathfrak{A}$ with the current interpretation of $ShownNrOfBombs$.
   - Output: a most refined partial structure $\mathfrak{A}'$ that approximates all models that expand $\mathfrak{A}$.

Select all positions (r,c) such that $BombAt(r, c)$ is false in $\mathfrak{A}'$.

3. The definition of ShowNRfBomb is an inductive one.

```
! r c n:  ShowNrOfBombs(r,c,n) <- Clicked(r,c)  &
              n = #{ u v: Adjacent(r,c,u,v) & BombAt(u,v)}.
! r c u v n: ShowNrOfBombs(r,c,n)<- Adjacent(r,c,u,v) & ShowNrOfBombs(u,v,0) &
              n = #{ u v: Adjacent(r,c,u,v) & BombAt(u,v)}.
```