CSC 481 - Resolution in Wumpus World

**Exercise 1: Safe cells in Wumpus world (Propositional Logic)**

Consider the Wumpus world below. The full rules of the game can be seen at <https://en.wikipedia.org/wiki/Hunt_the_Wumpus>, but we’re using a simplified version with no arrows and no bats.

| 3. Not yet explored |  |  |
| --- | --- | --- |
| 2. Reek, no breeze | 5. Not yet explored |  |
| 1. No reek, no breeze | 4. No reek, breeze | 6. Not yet explored |

We are using the following notation:

* means that the **Wumpus** is in cell
* means that a **Hole** is in cell
* means that cell **reeks**. All cells adjacent to the Wumpus reek (and these are the only cells that reek).
* means that cell **has a breeze**. All cells adjacent to a hole have a breeze (and these are the only cells with a breeze).
* means that cell is **safe**, defined as having no Wumpus or hole

In class, we expressed these statements in Propositional Logic for some cells of interest, and converted all the resulting sentences to CNF, which can be seen in cells 1..25 of the knowledge base (KB) below. We were also interested in the query “is cell 5 safe?”, and added the negation of “cell 5 is safe” to the KB.

**Task:**

* Use resolution to prove that cell 5 is safe, showing your steps in table 1 below
  + In other words, show that the KB with added is inconsistent
* Use the “Premises” column to record which two clauses were resolved together in that resolution step. For example, you could resolve clauses 7 () and 21 () to get , and then write (7,21) in the premises column.

Table 1: Use this table for your derivation of “cell 5 is safe”

| **ID** | **Clause** | **Premises** | **Notes** |
| --- | --- | --- | --- |
| 1 |  |  | Equivalent to ) by definition of implication |
| 2 |  |  | One of three clauses derived from . Clause 2 is equivalent to |
| 3 |  |  | Equivalent to |
| 4 |  |  | Equivalent to |
| 5 |  |  | Clauses 5 to 8 are analogous to 1 to 4, but regarding cell 4 and its neighbors |
| 6 |  |  |  |
| 7 |  |  |  |
| 8 |  |  |  |
| 9 |  |  | Clauses 9 to 16 are analogous to 1 to 4, but regarding holes and breezes |
| 10 |  |  |  |
| 11 |  |  |  |
| 12 |  |  |  |
| 13 |  |  |  |
| 14 |  |  |  |
| 15 |  |  |  |
| 16 |  |  |  |

Table 1 (cont)

| ID | Clause | Premises | Notes |
| --- | --- | --- | --- |
| 17 |  |  |  |
| 18 |  |  |  |
| 19 |  |  |  |
| 20 |  |  |  |
| 21 |  |  |  |
| 22 |  |  |  |
| 23 |  |  | Part of the definition of what “safe square” means. CNF form of  (Or perhaps more simply, of its contraposition ) |
| 24 |  |  | One of the clauses resulting from converting to CNF. Equivalent to |
| 25 |  |  | The other clause resulting from converting to CNF. Equivalent to |
| 26 |  |  | Our negated query |
|  |  |  | Start your proof! |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

Table 1 (cont again)

| ID | Clause | Premises | Notes |
| --- | --- | --- | --- |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

**Exercise 2: Safe paths with First-Order Logic**

Consider the alternate knowledge base shown below in Table 2, this time using First-Order Logic, and the problem of finding a “safe” path between two cells:

* Clauses 1-16 provide basic facts about the domain, such as which cells are known to be safe (using the Safe predicate) and which cells are neighbors to each other.
* Clauses 17 and 18 provide a recursive definition of a safe path:
  + Base case: a path from a safe cell to itself is always safe
  + Recursive case: if the starting cell is safe and a neighbor of that cell has a safe path to the final cell , then there exists a safe path from to .
  + Finally, clause 19 is the negation of our query: we are trying to prove that there is a safe path from cell 1 to 5, so we add to the KB

Perform resolution in FOL to prove that this KB is inconsistent. At each step, provide state which cells were resolved together and which substitutions (if any) were used. For example, if you wanted to resolve clauses 1 and 17, you’d get and you’d write (1,17, x/1) in the Premises column.

Table 2- Safe paths with first-order logic

| **ID** | **Clause** | **Premises/notes** |
| --- | --- | --- |
| 1 |  | Safe cells |
| 2 |  |  |
| 3 |  |  |
| 4 |  |  |
| 5 |  | List of neighbors |
| 6 |  |  |
| 7 |  |  |
| 8 |  |  |
| 9 |  |  |
| 10 |  |  |
| 11 |  |  |
| 12 |  |  |
| 13 |  |  |
| 14 |  |  |
| 15 |  |  |
| 16 |  |  |

Note that instead of redundantly listing the symmetric statement of each neighbor relation (Clauses 6,8,10,12,14,16), we could have added a general rule that the relation is symmetric: . I chose to list them explicitly to avoid potentially adding more resolution steps.

Table 2 (cont)

| ID | Clause | Premises |
| --- | --- | --- |
| 17 |  | Base case |
| 18 |  | Recursive definition |
| 19 |  | Negated query |
|  |  | Start your proof! |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |