Write-up on Project II

Working Content

I finish this project independently. The working content includes:

1. Truth-table enumeration algorithm.
2. Resolution algorithm
3. Sentence parser function which read various textual representations of propositional logic sentences and create the internal representations.
4. Test tt-entails and resolution algorithm with first four test questions:
    a. Modus Ponens
    b. Wumpus world (simple)
    c. Horn clauses (Russel & Norvig)
    d. Liars and truth tellers
5. Write-up on this project.

TT-ENTAILS: Inference by enumeration in propositional logic

A knowledge base KB entails a statement alpha if and only if in every world where KB is true, alpha is also true. The term ‘world’ is an assignment of Boolean values to all symbols. In truth-table enumeration algorithm, a world is a row in the truth table.

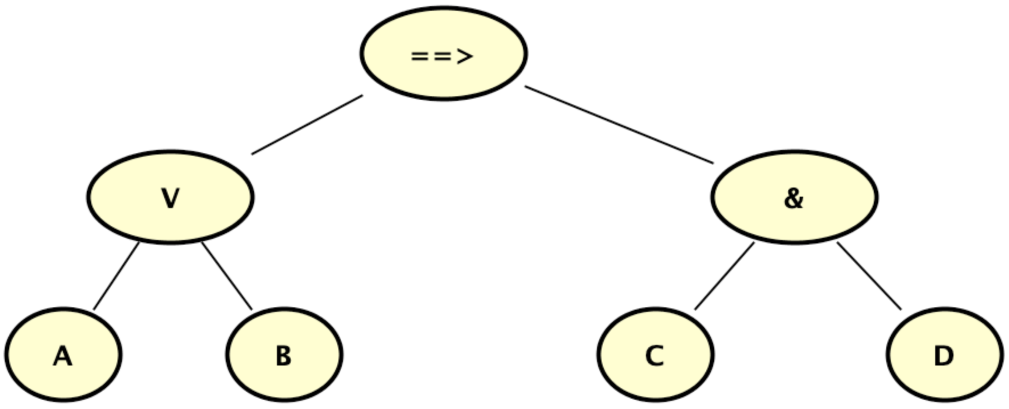
Truth-table enumeration algorithm is a smoking gun algorithm. The basic steps that a smoking gun algorithm does are as follows:

1. Ask the algorithm a question.
2. The algorithm does a loop and search for a smoking gun.
3. If it finds a smoking gun, it returns one answer.
4. If it finds no smoking gun, it returns another answer.

In truth-table enumeration algorithm, the question we ask it is that whether KB entails alpha. A smoking gun the algorithm is searching for is a row in the truth table where KB is true and alpha is false. If it finds a smoking gun, it will return false which means KB does not entail alpha. If it finds no smoking gun, it returns true which means KB entails alpha.

Within a propositional logic sentence, there are two things here: propositional symbols and connectives. We can imagine a sentence as a tree, in which connective and symbols are all nodes of the tree. A node could represent either connective or symbol. If a node stands for symbol, its connective attribute is null. If a node stands for connective, its symbol attribute is null and there is a list which contains the partitions of the sentence which are connected by this connective. It’s obvious that this is a recursive process. In this project, I adopt tree as the data structure to represent a propositional logic statement.

For example, we can represent a propositional logic sentence: (A V B) ==> (C & D) by constructing a tree as follow:



The root node of this tree is a node which has a connective attribute ‘==>’ and two children. One child has a connective attribute ‘V’ and two children A and B, the other child has a connective attribute ‘&’ and two children C and D. In this case, we can find that symbol node is always leaf node and root node is always connective node. Specifically, root node’s connective is the lowest priority level connective of the sentence except for the content within brackets given the priority of connectives: () > ~ > & > | > ==> > <==>.

We know that KB is a set of statements and alpha is a single statement, therefore it is reasonable that KB is the same data type as alpha. At this point, we have established what our data structure is. Next step is extract all symbols appearance in sentence. At above example, we get a list of strings contains A, B, C, and D.

The job of TT-entails is to go through the truth table to find a smoking gun (any row where KB is true and alpha is false). If it finds one, it returns true. Otherwise, it returns true. The actual work is done by TT-check-all method.

The first things that TT-check-all does is calls itself to build a truth table. Each time it calls itself, a symbol is removed from symbol list and is assigned a value in model (Symbol list contains all symbols appearance in sentence). The model is empty initially and is extended at each recursive call. When all symbols are removed from symbol list, TT-check-all begins checking whether KB is true in current model. If KB is true, then check whether alpha is true in current model. If both are true, it means that KB could entails alpha.

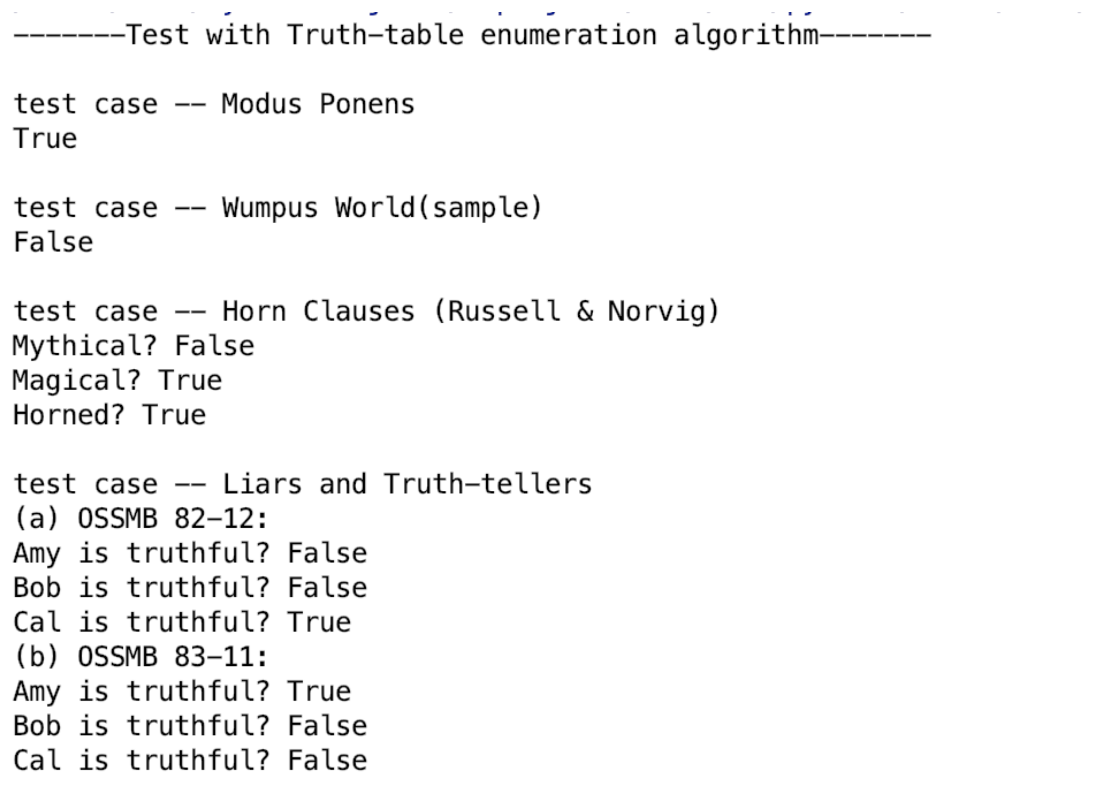
A resolution-based theorem prover

The resolution rule applies only to clauses (that is, disjunctions of literals), so it would seem to be relevant only to knowledge bases and queries consisting of clauses (AIMA p253).

The resolution algorithm takes as input a set C of clauses (KB & ~alpha) and returns true if and only if C is satisfiable. It does this by performing a sequence of resolution steps, where each step consists of identifying a pair (c1, c2) where c2, c2 ∈C of the form c1 = P | X, and c2 = Q | ~x, and then adding to C the

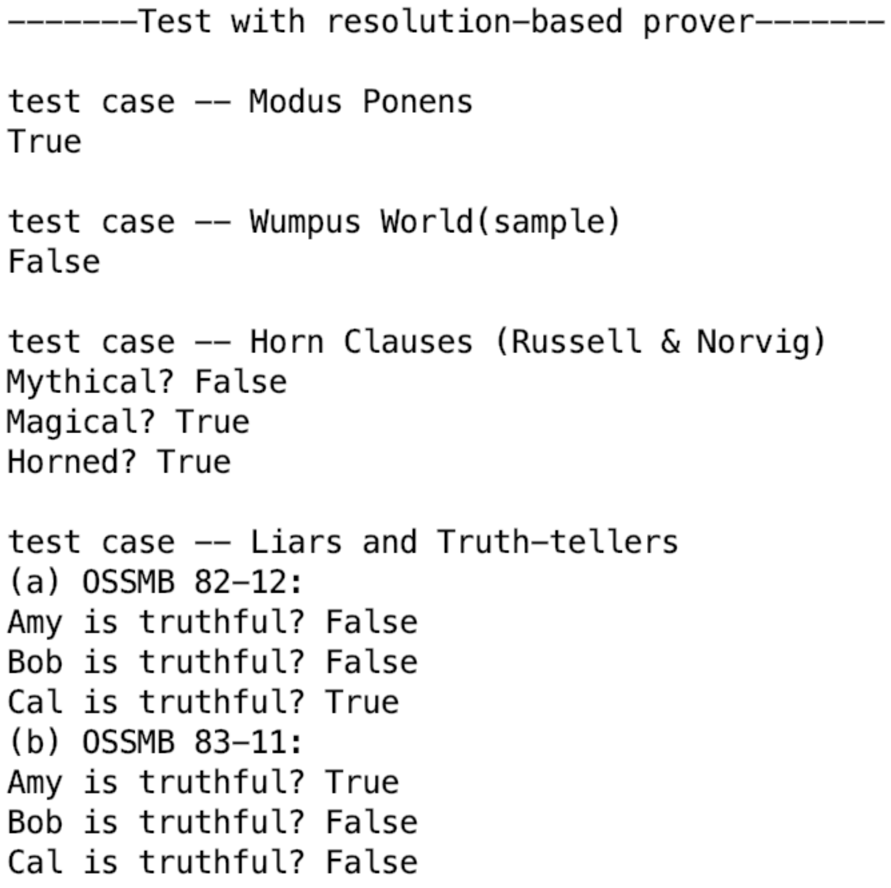
resolvent clauses c = P | Q.
There are two possibilities for how the resolution algorithm terminates:
In the first case, there is a step for which c1 = x and c2 = ~x, in which case P and Q are empty, and hence the resolvent P|Q is the empty clauses. Such a step is called a refutation step, and the algorithm is said to have performed a refutation proof. In this case, KB does not entail alpha.
In the second case, for each possible pair of clauses c1 = P | x, and c2 = Q | ~x, it is the case that P | Q is subsumed by a member of C, in which case no new resolvents can be added to C. in this case, KB entails alpha.

The screenshots of program as follow:
(1) Test with Truth-table enumeration.



```
-------Test with Truth-table enumeration algorithm-------

test case -- Modus Ponens
True

test case -- Wumpus World(sample)
False

test case -- Horn Clauses (Russell & Norvig)
Mythical? False
Magical? True
Horned? True

test case -- Liars and Truth-tellers
(a) OSSMB 82-12:
Amy is truthful? False
Bob is truthful? False
Cal is truthful? True
(b) OSSMB 83-11:
Amy is truthful? True
Bob is truthful? False
Cal is truthful? False
```

(2) Test with resolution-based prover.



```
-------Test with resolution-based prover-------

test case -- Modus Ponens
True

test case -- Wumpus World(sample)
False

test case -- Horn Clauses (Russell & Norvig)
Mythical? False
Magical? True
Horned? True

test case -- Liars and Truth-tellers
(a) OSSMB 82-12:
Amy is truthful? False
Bob is truthful? False
Cal is truthful? True
(b) OSSMB 83-11:
Amy is truthful? True
Bob is truthful? False
Cal is truthful? False
```