Name : Aditi Singla
Entry No. : 2014CS50277

COL703: Logic for CS

Assignment 3 : Resolution in First Order Logic

Signature

```
signature FOL = sig
        datatype Term = CONST of string
                      | VAR of string
                      | FUNC of string * Term list
        datatype Form = TOP1
                      | BOTTOM1
                      | PRED of string * Term list
                      | NOT1 of Form
                      | AND1 of Form * Form
                      | OR1 of Form * Form
                      | IMP1 of Form * Form
                      | IFF1 of Form * Form
                      | ITE1 of Form * Form * Form
                      | FORALL of string * Form
                      | EXISTS of string * Form

        datatype Quant = F of string | E of string
        datatype Lit =  P of string * Term list | N of string * Term list
        datatype Clause = CLS of (Lit list)
        datatype Cnf    = CNF of (Clause list)

        val makePrenex : Form -> Form
        val makePCNF : Form -> Form
        val makeSCNF : Form -> Form
        val makeCNF : Form -> Cnf
        val unify : Form -> Form -> (Term * Term) list
        val resolve : Form -> bool
end
```

Explanation

- A few more datatypes have been included from Assignment 2, for easy resolution in flat list of clauses, instead of a hierarchical structure.
- **makePrenex:** Takes in a form, and returns a Prenex Normal Form(PNF), where all the quantifiers move outwards.
- **makePCNF:** Takes in a PNF and returns a Prenex Conjunctive Normal Form(PCNF), where the quantifier free formula inside is turned to a CNF.
- **makeSCNF:** Takes in a PNF/PCNF and returns a Skolem Conjunctive Normal Form(SCNF), where all the existential quantifiers are eliminated.
- **resolve:** Takes in a form, converts it into PNF, then PCNF, then SCNF, and then converts it to a CNF, which is a list of clauses, datatypes for which have been added in the signature. Then, the resultant CNF is resolved and checked for an empty clause at any step. **It returns false, if it is able to find an empty clause during resolution, else returns true.**
- **makeCNF:** It is a helper function, which takes in a SCNF, and converts it to a list of clauses. It assumes that the input is already a CNF, and simply removes all the quantifiers beforehand.
- **unify:** It is also an important function, which takes in two predicates, and returns a bool and a mgu. The bool shows if they are unifiable, and the mgu is a list of substitutions.

Resolution

The assumption is that all the clauses are horn clauses. So, it can be a program clause or a horn clause. All the clauses are then segregated into two set, program and goal. Now, we do a depth first search to find a resolution where an empty clause is returned. Each predicate of each goal clause is unified with the positive predicate (since horn, it will be 1) of each program clause. If unifiable, check resolution for the resolvent.

Problems faced: Since the problem of resolution is semi-decidable, it may go into infinite loop. For this, the depth of search has been fixed at 60, to ensure termination. But still there is non-determinism, since as we increase the depth, more is the search and some more cases might get resolved which get cut at the current depth.

References

- Slides for Logic for CS: http://www.cse.iitd.ernet.in/~sak/courses/ilcs/ilcs.pdf