COL703: Logic for Computer Science
I semester 2021-22

Assignment: Explainable First-order Tableau

Problem Statement : In this assignment, you have to write a program which given a finite (possibly empty) list of closed formulae $\Phi = \{\phi_1, \ldots, \phi_n\}$ and another closed formula ϕ_{n+1}, constructs an analytic tableau to check if ϕ_{n+1} is a logical consequence of Φ. You have to construct the First Order Tableau using unification.

If ψ is not a logical consequence of Φ then your program should run forever (or until you get a segmentation fault or it is interrupted).

If $\Phi = \{\phi_1, \ldots, \phi_n\} \models \psi$, then a finite closed tableau should be the output.

Input : The input is a file `arg.sml` containing an argument as defined int eh signature given below.

Output : The output in case the argument is correct, is a file `tableau.dot` in dot format. The first-order tableau is "explainable" if it also has the dotted blue arrows and the red links as in the Hypernotes along with the unification remarks.

```
signature FOL =
  sig
      datatype term = VAR   of string
                    | FUN   of string * term list
                    | CONST of string (* for generated constants only *)
      datatype Pred = FF (* special constant for closing a tableau path *)
                    | ATOM  of string * term list
                    | NOT   of Pred
                    | AND   of Pred * Pred
                    | OR    of Pred * Pred
                    | COND  of Pred * Pred
                    | BIC   of Pred * Pred
                    | ITE   of Pred * Pred * Pred
                    | ALL   of term * Pred
                    | EX    of term * Pred
      datatype Argument = HENCE Pred list * Pred
      fun mktableau: Pred list * Pred -> unit (* outputs file "tableau.dot" in dot format *)

      exception NotVAR (* Binding term in a quantified formula is not a variable *)
      exception NotWFT (* term is not well-formed *)
      exception NotWFP (* predicate is not well-formed *)
      exception NotWFA (* argument is not well-formed *)
      exception NotClosed (* a formula is not closed *)
  end
```

1. Read the comments accompanying the declarations in the signature carefully.
2. The predicate constructor `FF` signifies the end of a closed path in the tableau.
3. The first occurrence of a function/predicate symbol determines its arity. A term/predicate is not well-formed if it has different arities in different occurrences in the argument.
4. A constant a is written as `FUN (‘‘a’’, [])` if it is part of the input argument. However you may have to generate new constants c which are then written as `CONST ‘‘c’’`.

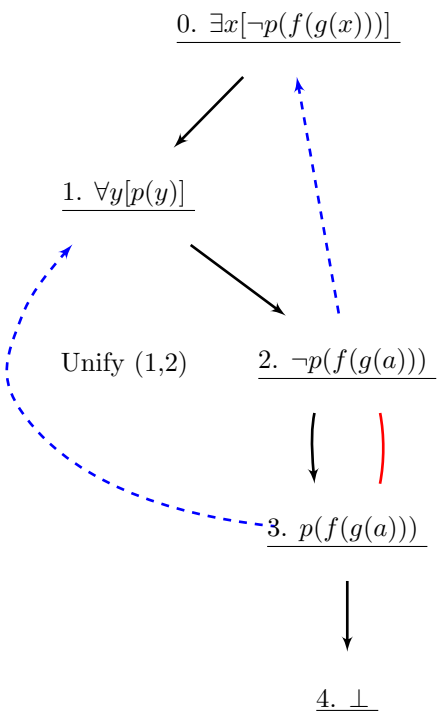
5. An atomic proposition (parameterless) p is written `ATOM (‘‘p’’, [])`.
6. The consructors `ALL` and `EX` stand for $\forall$ and $\exists$ respectively. Obviously all well-formed quantified formula should be of the form `ALL (VAR ‘‘string’’, phi)` or `EX (VAR ‘‘string’’, phi)` where `phi` denotes a predicate.

Sample dot file source code (without colours and fonts). By running the following commands

```
$ /usr/bin/dot2tex sample.dot > sample.dot.tex
$ pdflatex sample.dot.tex
```

we obtain a file `sample.dot.pdf` which displays the full tableau (*Try it!*).

```
digraph{
        nodesep = 0.5;
        ranksep = 0.35;
        node [shape=plaintext];
        0 [texlbl="\underline{0. $\exists x[\neg p(f(g(x)))]$  }"];
        1 [texlbl="\underline{1. $\forall y[p(y)]$  }"];
        2 [texlbl="\underline{2. $\neg p(f(g(a)))$  }"];
        3 [texlbl="\underline{3. $p(f(g(a)))$ }"];
        4 [texlbl="\underline{4. $\bot$ }"];
        subgraph dir
        {
                0 -> 1;
                1 -> 2;
                2 -> 3;
                3 -> 4;
        }
        subgraph ancestor
        {
                edge [dir=back, color=blue, style=dashed]
                0->2;
                1->3 [label="Unify (1,2)"];
        }
        subgraph undir
        {
                edge [dir=none, color=red]
                2 -> 3
        }
}
```


0. $\exists x[\neg p(f(g(x)))]$
1. $\forall y[p(y)]$
Unify (1,2)
2. $\neg p(f(g(a)))$
3. $p(f(g(a)))$
4. $\perp$