**Term Project #1 Resolution Proof System**

**Due Date: 05/18/2015 11:59PM**

**Introduction**

Design a proof system that applies resolution to prove (or disprove) a simple query based on a given knowledge base (KB) consisting of first-order-logic (FOL) sentences. Your proof system is expected to have the following capabilities.

1. To update the internal KB by reading a text file that contains FOL sentences.
2. To prove (or disprove) a given query provided as a text file.
3. To report the series of resolution steps before reaching the conclusion. Note that the report should include the conversion from FOL sentences to clausal forms, and the unification of variables with constants if they are required of the proof.

**Syntax (Grammar)**

To simplify the project (to ease your burden), the proof system deals with simple syntax, which involves only basic symbols, rather than complex natural languages.

1. Variables are denoted by capital letters, e.g. X, Y, Z, P, Q, R.
2. Constants are delimited by double quotation marks, e.g. “a”, “b”, “John”, “mary”.
3. Predicates are denoted by a word (letter), e.g. father, followed by a pair of parentheses within which the variables (parameters) are placed, e.g. P(X), father(Y, Z).
4. Quantifiers:

Exist

Forall

1. Connectives:

& AND (ascii code: 38)

| OR (ascii code: 124)

! NOT (ascii code: 33)

=> implies (ascii code: 61 62)

<=> mutual implication (ascii code: 60 61 62)

1. [ ] bracket

**Example:**

A text file to build your KB.

e.g. KB\_file(FOL): **StarWar.KB**

Variables: X,Y

Constants: "ObiWan","AnakinSkywalker"

Predicates: WantToKill(v1,v2),Jedi(v1),Sith(v1),CanUseForce(v1),StudentOf(v1,v2)

FOL start

Forall X Forall Y [WantToKill(X,Y) & Jedi(Y) => ! Jedi(X)]

Forall X [CanUseForce(x) <=> Jedi(X) | Sith(X)]

Forall X Forall Y [StudentOf(X,Y) & CanUseForce(Y) => CanUseForce(X)]

StudentOf("AnakinSkywalker","ObiWan")

WantToKill("AnakinSkywalker","ObiWan")

Jedi(“ObiWan”)

FOL end

A text file to form the queries.

e.g. Query\_file: **StarWar.Query**

Sith("AnakinSkywalker")

The attachment contains some examples KB and Query file for development.

**What to turn in**

A XXX.zip file containing (XXX is your student ID)

1. An electronic report file

Prove/disprove **StarWar** example (You can manually prove it if your program do not work). The file should contain (a) conversion from FOL sentences to clausal forms, (b) the unification of variables with constants if they are required of the proof, (c) proof procedure. An example report is at the end of this file.

1. Source code in **ANSI-C (or C++)**

Make sure your code can be compiled, and executed; otherwise, you will NOT get any credit for this project. By “compiled” and “executed”, we mean that TAs can compile your code, and get it to run, i.e. at least your code can read the input and proceed without crash (no matter whether your final answer is correct or not). The output of your program should be the same as (a), (b), (c) above.

**Grading Policy**

1. Correct proof: (a) correct conclusion, (b) correct conversion, and (c) correct unification, (d) proof procedure.
2. Better performance: fewer resolution steps, less running time, etc.
3. You will get no point if you plagiarize any existing source codes from the Internet or your classmates.

**Additional information**

1. There may exist some complicated FOL in KB and Query, like:

*Forall X [[Forall Y [Animal(Y) => Loves(X,Y)]] => [Exist Y [Loves(Y,X)]]]*

Your parser should be smart enough to handle it. Please refer to the example in **Curiosity2.KB and Curiosity2.Query**.

1. There may exist the situation that you need to do Occur-Check (p.327 of Book), otherwise, you will be trapped in loop when you do unification, you can refer to the example in **OccurCheck.KB and OccurCheck.Query**

Format of Report and Program

Compare.Result

conversion start

KB

{

Forall X Forall Y [chick(X) & crocodile(Y) => smaller(X,Y)]

to

C1 ! chick(X) | ! crocodile(Y) | smaller(X,Y)

Forall X Forall Y [crocodile(X) & elephant(Y) => smaller(X,Y)]

to

C2 ! crocodile(X) | ! elephant(Y) | smaller(X,Y)

Forall X Forall Y Forall Z [smaller(X,Y) & smaller(Y,Z) => smaller(X,Z)]

to

C3 ! smaller(X,Y) | ! smaller(Y,Z) | smaller(X,Z)

Exist X [elephant(X)]

to

C4 elephant($v1)

}

Negated Query

{

! [Exist X Exist Y Exist Z [chick(X) & crocodile(Y) & elephant(Z) => smaller(X,Z)]]

to

C5 chick(X)

C6 crocodile(Y)

C7 elephant(Z)

C8 ! smaller(X,Z)

}

conversion end

proof & unification start

C1+C5+C6 with unifiers{C1.X/C5.X, C1.Y/C6.Y}

C9 Smaller(X,Y)

C8+C9 with unifiers{C8.X/C9.X, C8.Z/C9.Y}

C10 NULL

proof & unification end

PROVE/DISPROVE:

PROVE