CSCI 360 – Project #1

Theoretical Part: First-Order Logic Resolution - 3 Points

We thank Mike Genesereth for providing the questions below. Each question is worth 1.5 points.

1. Don has been murdered, and Alice, Bob, and Carl are suspects. Alice says she did not do it. She says that Bob was the victim's friend but that Carl hated the victim. Bob says he was out of town the day of the murder, and besides he didn't even know the guy. Carl says he is innocent and he saw Alice and Bob with the victim just before the murder. Assuming that everyone – except possibly for the murderer – is telling the truth, model this problem in First-Order Logic and use resolution to solve the crime.

   Hint: You should model the following general facts while solving the crime:

   - Anyone who was with Don on the day of the murder must have been in town.
   - A person knows who their friends are.
   - A person who likes someone must know them.

2. If a course is easy, some students are happy. If a course has a final, no students are happy. Model this problem in First-Order Logic and use resolution to show that, if a course has a final, the course is not easy.

Programming Part: Clue - 7 Points

In this project, we will model the boardgame Clue[1] by using propositional logic and an off-the-shelf solver (namely, zChaff) to reason about it. You are required to complete Section 8 (pages 15-22) of `http://modelai.gettysburg.edu/2011/clue/clue.pdf`. We suggest that you read through Sections 1-7, as they will help you to understand the game's rules and modeling language.

The original project is written for Java, but we have converted their provided code into C++. For this project, you will implement three methods in the given "ClueReasoner.cpp":

- `AddInitialClauses`

[1] `http://en.wikipedia.org/wiki/Cluedo`

- `Hand`
- `Suggest`

Note that "ClueReasoner.cpp" also contains the function `Accuse`, which is included in the original project. However, since the project does not test for the correct implementation of `Accuse`, you do not need to implement it.

To get a perfect grade, your output should be identical to:

```
        sc      mu      wh      gr      pe      pl      cf
mu      n       n       n       n       n       Y       n
pl      n       Y       n       n       n       n       n
gr      n       n       Y       n       n       n       n
pe      n       n       n       n       n       n       Y
sc      n       Y       n       n       n       n       n
wh      Y       n       n       n       n       n       n
kn      n       n       Y       n       n       n       n
ca      n       n       n       Y       n       n       n
re      n       n       n       n       Y       n       n
ro      n       n       Y       n       n       n       n
pi      n       n       n       n       n       n       Y
wr      n       n       n       Y       n       n       n
ha      n       n       n       n       Y       n       n
lo      n       Y       n       n       n       n       n
di      n       n       n       n       Y       n       n
ki      n       n       n       Y       n       n       n
ba      n       n       n       n       n       Y       n
co      n       n       n       n       n       Y       n
bi      n       n       n       n       n       n       Y
li      Y       n       n       n       n       n       n
st      Y       n       n       n       n       n       n
```

In this output, the rows correspond to the cards and the columns correspond to the players (or the case file). A `Y` means that the card is in the hand of the player (or in the case file), an `n` means that the card is not in the hand of the player (or in the case file), and a `-` means that it is unknown whether the card is in the hand of the player (or in the case file). For instance, a `Y` in row `kn` and column `wh` means that the card `kn` is in the hand of player `wh`.

You should not modify any of the source files except for "ClueReasoner.cpp". In addition to the output of your code, we will also check the clauses generated by your code. Therefore, even if your code's output exactly matches the table above, you might still lose points if your code has unnecessary/missing clauses. On the other hand, small mistakes in your code can significantly affect the correctness of the table generated by your code and, by checking the clauses generated by your code, we can give you some partial credit.

Provided Code and the Programming Environment

We have tested and verified that the provided code (and a solution to the project that extends the provided code) compiles and runs on `aludra.usc.edu`. Running `make` should compile and run the code, which should generate an output like the one provided above (with `Y`'s and `n`'s replaced by `-`'s, meaning that the corresponding proposition is not guaranteed to be true or false). As you extend the provided code by implementing some of the functions, you should notice some `Y`'s and `n`'s appearing in the output of the program, meaning that you are closer to solving the case.

You are free to develop your code on any platform that you choose. We will test your code on a Linux machine that supports C++11.

Installing zChaff

The project folder contains an executable for zChaff, called `zchaff`, that was compiled on `aludra.usc.edu`. Before you can run `zchaff`, you will need to set its access permission as an executable, using the following command:

```
chmod 754 zchaff
```

If you want to develop your code in another environment, you can download zChaff from `http://www.princeton.edu/~chaff/zchaff/zchaff.2008.10.12.zip`. After untaring it, you need to run `make` to create the executable "zchaff", and move the executable to your project folder, which contains the source code and the makefile for the project.

You can call the `TestSatSolver()` function from "main.cpp" to check if your code interfaces correctly with zChaff.

Note that, in Ubuntu releases starting with 14.04, you might need to add

```
#include <cstring> to sat_solver.cpp
```

and

```
#include <stdlib.h> to zchaff_dbase.cpp
```

Submission

You should submit a PDF file named 'Part1.pdf' for your solution to the theoretical part (scanning hand-written answers are okay as well, as long as the handwriting is legible and you are submitting a PDF file) and your modified "ClueReasoner.cpp" file for the programming part through the blackboard system by Wednesday, Sep. 21, 11:59pm. If you have any questions about the project, you can post them on piazza (you can find the link on the course wiki) or come to the TAs' office hours.