

First Order Logic and Backward Chaining Homework

T. Nathan Mundhenk and Laurent Itti
University of Southern California
Department of Computer Science

Copyright (c) 2006 T. Nathan Mundhenk, Laurent Itti

Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.2 or any later version published by the Free Software Foundation; with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts. A copy of the license is included in the section entitled "GNU Free Documentation License".

CS561 Homework 4

Due: By class Thursday, April 6, 2006

Guidelines:

This assignment has a written and programming component. You can see how much each part of the assignment is worth by the percentage next to it. For the written part, please turn in typewritten answers. You are not English majors, however blatant spelling and grammatical errors may cost you points, so be sure and run the spell check at least once. Answers on the written part will require justification and a simple yes/no answer will almost certainly garner you little or no points. In general, it is safer to write longer answers than shorter ones, but stay focused as a long but off-topic answer will not work either. This way, we can discern your train of thoughts better and grant partial credit for a wrong answer, if you were on the right track. Graphs must be neat and tidy. Hand-drawn graphs are OK, but computer-drawn graphs (e.g., made with Adobe Illustrator or Microsoft Power Point) are preferred. If you draw a graph by hand, be sure and use a straight edge (ruler) so it looks neat.

For the programming part, you will be provided sample inputs and at least one sample output. Since each homework is checked via an automated Perl script, your output should match the example format *exactly*. Failure to do so will most certainly cost some points. Since the output format is simple and there is an example on the web, this should not be a problem. Additionally, if your code generates a large number of warnings during compilation, you may lose points, so try and eliminate compile-time warnings. Additionally, your code should be well documented. If something goes wrong during compile and grading, if the fix proves easy, the amount of points lost will be far less. As such, documentation makes fixing easier, so it is to your advantage to do so.

You will be provided with a stub Perl script called “stubby.pl”, which works like the grading Perl script. You can run this script to make sure that your project will work properly. Thus, it is expected that your project will run through the grading Perl script without problems. Pedantically, we assert it will cost you points if your code does not run on the Perl script correctly. You will be able to tell if your output is correct if stubby shows each of the lines from your program output is exactly the same as from the comparison file which shows what output you should be getting.

To run stubby, copy it to your code directory along with the input, output and comparison files. Be sure to be in your code directory then type “perl stubby.pl”. The script is loaded with all sorts of output and feedback, so you should be able to see what it is doing if for some reason it isn't working for you. If you don't understand the script and want to know more about Perl, go to <http://www.perl.org/>.

Small amounts of extra credit (not more than 10% total) are given for all sorts of things. So long as you meet the base homework requirements anything creative, fun, interesting or outright cool

will most likely earn you extra credit. What is cool and earns you extra credit is somewhat subjective and bound to the whim of the grader.

Handing in the Assignment:

Since the class is very large it is important to hand in your homework as directed. *The homework is due before class on the due date shown.* There are two parts of the homework, which you must hand in. These are:

- (A) Your code tar/gzipped in **one** file. Do not send the binaries. - This should just include your uncompiled code and a readme file called readme.txt. When I run tar/gzip to uncompress your file, it should uncompress into its own directory. To keep things uniform, do not use bzip2. I should then be able to run my stubby Perl script and grade your code. See below on how to tar/gzip your code.
- (B) Your written part on paper. *Iff* you are a DEN student, you may submit your written part in the DEN digital drop-box (remember your cover sheet). Otherwise, you must submit a paper copy either in class or in a drop-box in front of my office in HNB (NOTE: the HNB building closes at 5:00pm). Late submissions will be noted. Be sure your homework is stapled together and is generally tidy. Electronic submissions of the written part from non-DEN students will not be accepted.

Be sure that your name is *clearly visible* on all material you hand in. To hand in your code you can compress it with tar/gzip with a command like:

```
“tar -czvf my.name.hw4.code.tgz my.name.hw4.code”
```

You might also try:

```
“tar -cvf my.name.hw4.code.tar my.name.hw4.code” then type “gzip my.name.hw4.code.tar”
```

This will compress the contents of the directory named “my.name.hw4.code” into a single file my.name.hw4.code.tgz. If you need more info on this, try “man tar”.

Electronic Submission of code:

You need to submit the assignment electronically via the DEN digital drop-box. Do this by accessing the DEN web page and then the DEN digital drop-box. Be sure to name your code with a name such as:

```
my.name.hw4.code.tgz
```

Be sure and substitute “my.name” with your own name. The reason for naming your submission is to make it much easier to match up projects with the students who submit them. Otherwise this task becomes difficult for the grader. When you have uploaded your code, be sure and click submit.

DEN students using electronic submission of the written part:

For DEN students handing in the written part electronically, if you use Word files, please compress the file using .zip or .rar.. Don't worry about compressing Acrobat documents since they are already compressed. Thus, I should see two files if you are doing an all-electronic submission, one with a name like bob.bobbinopolis.hw4.written.zip and the other with a name like bob.bobbinopolis.hw4.code.tgz. If you send me a file with a name like hw4.zip, I will put a hex on your credit cards. Of course this does not apply to your .cpp/.h files which should have names that work with the stubby Perl script. Also, be sure your name is on the documents you hand in since we print them up to grade them and it can be hard keeping documents matched. Please make sure all images and material for the written part are collated into one single file. Thus, *do not* send a word document and 30 image files, be sure and place them inside the word document itself.

Additionally, DEN students should remember to include a *cover sheet* with their submissions.

What exactly is Stubby?

This is very important. We use a Perl script to automate the grading process. Stubby is a Perl script we give to you so that you can check and make sure that your project conforms to specifications. That is, you can use Stubby to make sure that your assignment when handed in will run with our grading script. Thus, we have our own grading script *like* stubby, which we use to grade your project. By checking to make sure your project works with stubby, you make sure that your project will work with our automated grading script.

It is important to note that we will not use your version of stubby. We have our own script. As such, if you edit stubby to work with your code and not the other way around, this will not be very helpful. Additionally, since there are so many projects to grade, it is imperative that your program work with the grading Perl script. *If your program does not work with the grading Perl script you will lose points.*

Question 1 (4%)

You are a Vulcan space explorer when you set foot on planet Talerad VI. You are greeted by the local primitives known as the Moog. They are a rather uncivilized and barbaric race who seem to go to war over the most trivial matters. However, you notice that the Moog remind you very much of the descriptions of the Vulcan people before the time of Surak. As such you decide that you should teach the Moog the basic ideas of logic, in hope that they too will learn how to live in peace and prosperity with one-another in a logical society.

You are teaching a group of elders from local tribes the nature of logic. However it is very difficult, the Moog are still very much primitive and given to emotional outbursts. One of the local elders, Trolog, begins to become impatient with your explanations of logic and jumps up and smashes his axe into a log in front of you. He then screams “This logic is very difficult, you say so many things, but yet I do not understand. For instance, I do not have any notion of *what it means for a logical inference procedure to be sound*”. To this you reply:

(A) (2 points)

Trolog seems to be satisfied with you explanation and settles back to sit with the other elders. At this point a voice comes from the back, it is Org the Wise. He is the oldest of all the elders and is so feared and respected, that he never needs to speak in a loud tone. He says,” Trolog’s quandary is shared by us all, this idea of soundness however seems to have been stated in a manner that we can understand, yet I do not understand *what you mean when you say that a logical inference procedure is complete*.” This is indeed a good question to which you state:

(B) (2 points)

The Moog seem for the time being satisfied with your lessons when it comes time for you to leave Talerad. However, while seeing you off, Grom the Terrible steps on Lothar of the Hill People’s foot setting off a new wave of war between the local tribes. Oh well you think, such is the work in vein of a Vulcan space explorer.

Question 2 (15%)

You are employed as an editor at Random House. Your boss has recently stopped taking his medication and has demanded that you translate the following Dr. Seuss verses into **first order logic**. He has also demanded that you wear a tin-foil hat while doing your work. However, you will try and take each request one at a time.

Translate each sentence into first order logic (3 points each)

- (1) When the Star-Belly Sneetches had frankfurter roasts or picnics or parties or marshmallow toasts, they never invited the Plain-Belly Sneetches.
- (2) I'm a North-Going Zax, and I always go north.
- (3) I do not like them, Sam-I-am. I do not like green eggs and ham.
- (4) It's a troublesome world. All the people who're in it are troubled with troubles almost every minute.
- (5) I will eat them in a house, and I will eat them with a mouse, and I will eat them here and there, I will eat them ANYWHERE.

Question 3 (Coding 61%, Related Questions 20%):

Dr. Ernest Beakerman has been experimenting with gene knockout mice for the last 10 years. Recently he has had a major breakthrough with his KM-52a strain. It turns out that the mice are hyper intelligent and possess a keen grasp of logic far superior to any humans. Since mice by their nature are quite friendly the KM-52a mice have employed themselves to bettering humanity. For instance, Algernon II is now advising the president on foreign matters and has brought the Iraqi crisis to a graceful ending and stopped the spread of AIDS in Africa.

The problem with KM-52a is that like all other mice, they live at most 4 years. As a result, new generations must be constantly trained to replace the current mice, which die off very quickly. Additionally, they must be trained with great speed so that they can spend as much time as possible aiding humanity into a new era of prosperity.

Dr. Beakerman has employed you in his lab to facilitate the training of KM-52a mice. Each mouse will sit in front of a computer practicing logic exercises. It will be presented with several propositional logic statements in **Horn clauses**. It will read them and then type in a logical query, which the statements should be able to prove. Your job is to write a program that will check each conclusion a mouse makes and give it immediate feedback as to whether the query can be proven. Thus, the mice will learn very quickly via feedback a strong understanding of logic.

To help you develop your system, Dr Beakerman has provided you with several sample knowledge bases the mice have produced. The knowledge bases contain Horn clauses with the following defined operators:

NOT X	~X
X AND Y	X && Y
X IMPLIES Y	X => Y

You will use **backward-chaining** (see Textbook P. 217 2nd ed) to solve this problem.

Input: You will be given text files that contain samples from several mice. The first line indicates the query. The second line contains an integer n specifying the number of clauses in the text file. Each line after will contain a horn clause, which is written as an implication whose premise is a conjunction of positive literals and whose conclusions is a single positive literal. It has the form:

$$p_1 \wedge p_2 \wedge \dots \wedge p_n \Rightarrow Q$$

Output: If you can prove the query, then the answer will be **YES**, otherwise, **NO** (with NO meaning that you either disapprove the query, or the knowledge base is incomplete to determine the query).

A sample input file will look like:

```
A_Witch
```

7

```
Made_Of_Wood => A_Witch
Weights_Same_As_Duck && Floats_On_Water => Made_Of_Wood
Turned_Me_Into_A_Newt && Weights_Same_As_Duck => Floats_On_Water
Has_A_Wart && Made_Of_Wood => Weights_Same_As_Duck
Has_A_Wart && Turned_Me_Into_A_Newt => Weights_Same_As_Duck
Has_A_Wart
Turned_Me_Into_A_Newt
```

To which your program will output: **YES**

Once you have completed this project Dr. Beakerman will use it to expedite the training of KM-52a mice and thus usher in a new era of peace and prosperity. After that he will collect his Nobel prize and from then on forget to ever mention your name when talking about how to train KM-52a mice.

NOTE: *Do not* use AIMA 1st edition for any part of this problem since it has mistakes involving first order logic and proofs. Some additional help on chaining and proofs may *also* be found in ch. 9 AIMA 2nd ed.

Questions (5 points each):

- A. For the above sample input, please draw a proof tree of backward chaining.
- B. For the above sample input, please draw a diagram of a resolution proof.
- C. Describe the complexity of the backward-chaining algorithm.
- D. What if the problem did not contain Horn Clauses? How might your solution be different?