

CSCI 561
Foundations of Artificial Intelligence
Spring 2010

Project 2: Logical Agents

Due: 4:59 p.m., April 19, 2010

1 Introduction

This project gives you an opportunity to code two propositional inference algorithms, the DPLL backward-chaining algorithm and WalkSAT. Both algorithms are used to check whether a given knowledge base entails a query. This project should be done alone.

2 Project Description

This project consists of two parts. The first part requires that you code the DPLL backward-chaining algorithm and WalkSAT. The second part requires that you answer the questions in the “Questions” section.

2.1 Coding

You will be given input files that contain samples of propositional logic statements, where $\sim X$ corresponds to $\neg X$, $X \ \&\& \ Y$ corresponds to $X \wedge Y$, and $X \Rightarrow Y$ corresponds to $X \Rightarrow Y$. The first line of the input file indicates the query. The second line contains an integer n specifying the number of clauses in the text file. Each line after that contains a Horn clause, which is written as an implication whose premise is a conjunction of positive literals and whose conclusion is a single positive literal. You are to insert the negation

of the query into the knowledge base and convert all Horn clauses in the knowledge base into conjunctive normal form.

Then, you are to implement two propositional inference algorithms. The first is the DPLL backward-chaining algorithm described in Figure 7.16 of the textbook, and the second is WalkSAT described in Figure 7.17 of the textbook. The algorithms are worth 45 points each. Both algorithms take as input a set of clauses in conjunctive normal form and checks if there exists a model that satisfies all clauses. The query is entailed by the knowledge base *if and only if* no such model exists.

We provide ten input files, which are named from `example1.txt` to `example10.txt`. Make sure that your DPLL backward-chaining algorithm and WalkSAT can be executed with the following flags:

```
backwardchaining -i inputfile and
walkSAT -i inputfile -f maxflips
```

respectively, where `inputfile` indicates the name of the input file and `maxflips` indicates the maximum number of flips WalkSAT should make before terminating.

Your DPLL backward-chaining program should output the symbols and truth values found by the `FIND-PURE-SYMBOL` and `FIND-UNIT-CLAUSE` functions, if there are any, and whether the query is entailed by the knowledge base. For example, executing `backwardchaining -i example10.txt` should output

```
Found pure symbol:  I_Watch_Too_Much_X_Files:  T
Found pure symbol:  Taking_My_Haldol:  T
Found unit clause:  Elvis_is_Alive:  F
Found unit clause:  I_Find_Tinfoil:  T
Found unit clause:  Transmitters_Implanted_in_Brain:  T
Found unit clause:  UFOs_Kidnap_Me:  T
Found unit clause:  The_FBI_is_Bored:  T
Found unit clause:  The_FBI_Taps_My_Phone:  T
Found unit clause:  I_Freak:  T
Found unit clause:  Wear_Tinfoil_Hat:  F
```

```
Query is entailed by the KB: YES
```

Your WalkSAT program should output the initial and final truth values of all the symbols and whether the query is entailed by the knowledge base. For example, executing `walkSAT -i example2.txt -f 100` should output

```
Initial truth values:
Youd_Like_to_be_a_Pepper_too:  F
Im_a_Pepper:  F
Shes_a_Pepper:  F
Were_all_Peppers:  F
I_Drink_Dr_Pepper:  F
Im_Proud:  F
Hes_a_Pepper:  F
```

```
Final truth values:
Youd_Like_to_be_a_Pepper_too:  F
Im_a_Pepper:  F
Shes_a_Pepper:  T
Were_all_Peppers:  F
I_Drink_Dr_Pepper:  T
Im_Proud:  F
Hes_a_Pepper:  T
```

Query is entailed by the KB: No

2.2 Questions

2.2.1 Question 1 [6 points]

WalkSAT needs to initialize the values of all symbols at the start of the algorithm. The choice of the initial values can determine how quickly WalkSAT converges to a solution, if one exists. Describe two good heuristic functions that can be used to initialize these initial values. Indicate which heuristic function was implemented in your submitted version of WalkSAT.

2.2.2 Question 2 [2 points]

Is it possible for the DPLL backward-chaining algorithm to find a model that satisfies all clauses while WalkSAT does not? Explain.

2.2.3 Question 3 [2 points]

Is it possible for WalkSAT to find a model that satisfies all clauses while the DPLL backward-chaining algorithm does not? Explain.

3 Submission Instructions

3.1 Code

Your source code must be written in C/C++ and be well commented. Make sure that your source code can be compiled and executed on `aludra.usc.edu`. Please provide a README file with instructions to run your programs and a Makefile to compile where possible.

3.2 Report

Your report should contain the answers to the questions in Section 2.2 and must be submitted in PDF format only. Please include your full name, USC login name and USC ID on the front page of this report. Name the report as `Firstname_Lastname_report2.pdf`.

Use the following steps to submit your project:

Step 1: Place all the required files in a single directory.

Step 2: `tar` and `gzip` these files listed into an archive called:

`Firstname_Lastname_project2.tar.gz`.

Step 3: Submit the project in the following manner (all in one line): `submit -user csci561b -tag project2 Firstname_Lastname_project2.tar.gz`.

The following message will appear if the submission is successful:

Submitting file ‘‘Firstname_Lastname_Project1.tar.gz’’ to

```
‘‘csci561b’’ ... SUCCEEDED.
```

Important Note: We will be enforcing strict submission guidelines for this project. We will not accept zipped files, email submissions and files in sub-folders etc. A submission can be updated simply by resubmitting as long as it is on or before the due date. Projects handed in one day late will result in a 25% reduction in the total score, two days late will yield a 50% reduction, and no credit will be give for three or more days late.

4 Grading Policy

The grading policy follows the guidelines below:

Code [90 points]

- Your source code must be written in C/C++.
- Your source code must compile and execute on `aludra.usc.edu`.
- A README containing instructions to run your programs must be provided.
- A Makefile must be provided to compile the source code.
- The source code must be properly commented.
- The DPLL backward-chaining algorithm and WalkSAT works correctly and follows all our requirements.

Report [10 points]

- Your report must be in PDF format.
- The front page of your report must contain your full name, USC login name and USC ID.
- Your report must be named according to our naming convention.

5 Acknowledgment

We extended this project from the first order logic and backward chaining homework by T. Nathan Mundhenk and Laurent Itti, University of Southern California, under the GNU Free Documentation License.

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”.