COMP 509
2022 Course Project

Interim Report Due Date: midnight, November 13, 2022
Final Report Due Date: midnight, December 2, 2022, but there will be no penalty for submitting by midnight, December 13, 2022.

The goal of this project is to investigate the performance of the DPLL Procedure for propositional CNF satisfiability. (Formulas should be represented in DIMACS format; see `http://www.cs.rice.edu/~vardi/comp409/satformat.pdf`.) Your solver should be able so solve arbitrary CNF formulas in this format. The output of the solver should be a satisfying truth assignment or the string "UNSAT".

Note: You are allowed to search on the WWW and read papers about satisfiability solving, but you **must not** use code written by others.

The DPLL Procedure

The DPLL procedure consists of two rules:

- Unit-preference rule; select a unit clause c, assign to the proposition of c a truth value that satisfies c, simplify the input formula and call DPLL recursively.
- Splitting rule: select a proposition p, assign to it a truth value, simplify the input formula and call DPLL recursively. If this call fails to satisfy the input formula, assign another truth value to p, simplify the input formula and call DPLL recursively.

Note that this description leaves you with important implementation choices: how to choose propositions in the unit-preference rule and in the splitting rule and how to choose a truth value in the splitting rule. These choices can have a significant impact on the running time of DPLL. Note that even though DPLL is described as a recursive procedure, you need not implement it as such; you can gain significant efficiency by implementing it as an iterative procedure.

You should choose a heuristic for choosing propositions and truth values. To evaluate the effectiveness of your heuristic you should compare them to the *random-choice heuristic*, in which all choices are resolved randomly with a uniform distribution. You should also compare *your heuristic* to the *2-clause heuristic*, in which we choose propositions with maximum occurrences in 2-clauses, i.e., clauses with two literals, and break ties randomly. I expect you to find a heuristic that would be better than the 2-clause heuristic. Make sure that all implementations agree on the answer (satisfiable or unsatisfiable) for all each input formula.

You will evaluate your solver by using it to solve a puzzle and measuring its performance on random formulas.

1 The Puzzle

Supposedly, Albert Einstein wrote this riddle, and said 98% of the people could not solve it.

There are five houses in five different colors. In each house lives a man with a different nationality. The five owners drink a certain type of beverage, smoke a certain brand of cigar, and keep a certain pet. No owners have the same pet, smoke the same brand of cigar or drink the same beverage. The question is: "Who owns the fish?"

Hints:

- The Brit lives in the red house.
- The Swede keeps dogs as pets.
- The Dane drinks tea.
- The green house is on the left of the white house.
- The green house's owner drinks coffee.
- The person who smokes Pall Mall rears birds.
- The owner of the yellow house smokes Dunhill.
- The man living in the center house drinks milk.
- The Norwegian lives in the first house.
- The man who smokes Blends lives next to the one who keeps cats.
- The man who keeps the horse lives next to the man who smokes Dunhill.
- The owner who smokes Bluemasters drinks beer.
- The German smokes Prince.
- The Norwegian lives next to the blue house.
- The man who smokes Blends has a neighbor who drinks water.

In order to solve the puzzle, you have to make some assumptions:

1. The owner is the resident of each house.
2. One of the residents owns the fish.
3. The term neighbor in the last hint refers only to a directly adjacent neighbor.
4. The houses are on the same side of the street.
5. They are next to each other, and are ordered from left to right as you face them rather than standing in front of a house facing the street (i.e. facing the same direction as the house).

You need to solve this puzzle by expressing it first as a CNF formula such that the satisfying assignment of the formula correspond to possible solutions of the puzzle.

Interim Report: Your interim report should include your encoding of the problem as a CNF formula and your solution of the puzzle.

2 The Random Model

You will evaluate the performance of DPLL on random formulas generated according to the fixed-clause-length model. In this model there are three parameters: the number N of variables, the number K of distinct literals per clause, and the number L of clauses. To keep things simple, in this project we will take K to be three; that is, we will focus on the 3-SAT problem. Since the goal is to see how DPLL performs on *large* problems, you should aim at handling values of N that are at least 150 (though you may not be able to handle so many variables with the random-choice heuristics). If your implementation of DPLL is good, you should be able to handle even larger values of N. The larger N is, the clearer the picture that will emerge from your experiments, and the better your analysis can be.

For a given N and L, an instance of random 3-SAT is produced by randomly generating L clauses of length three. Each clause is produced by randomly choosing a set of three distinct propositions from the set of N propositions and negating each one with probability 0.5. You should generate a set of formulas in advance, and then evaluate the performance of different implementations on this set of formulas.

3 Final Report

You will measure the performace of DPLL by counting both total compute time as well as the total number of splitting-rule applications required to find whether the input formula is satisfiable or not.

In your experiments on the random model you will investigate the dependence of the performance of DPLL on the ratio L/N for a fixed N (N should be as large as you can handle); vary this ratio at increments of 0.2 from 3 to 6. For each ratio, run 100 experiments and report their median (the mean can be influenced heavily by "outliers"). (It is ok to abort experiments that run too long to affect the median.) You should also report on the probability that an input formula is satisfiable. You should compute the probabilities for at least two values of N separated by at least 25 (e.g., N=100 and N=150) and draw conclusions from the difference in the probabilities.

Your final report should include the source code for your implementation of DPLL, an explanation of the code (i.e., a clear verbal explanation of your implementation choices), the numerical results and plots for the following:

- Compute time and number of DPLL calls on random formulas as a function of L/N.
- Probability of satisfiability of random formulas as a function of L/N.
- Ratio of the performance of your heuristic to that of the random-choice heuristic on random formulas as a function of L/N.
- Ratio of the performance of your heuristic to that of the 2-clause heuristic on random formulas as a function of L/N.

You should conclude your report with a verbal summary and an analysis of your findings. Your analysis can be intuitive, but you need to come up with an explanation of your findings. YOUR REPORT SHOULD END WITH A PARAGRAPH EXPLAINING WHAT YOU LEARNED FROM THIS PROJECT.

4 Other Comments

Unlike written assignents in this course, this project should be done individually. You are allowed to discuss your implementation strategy with your peers, but you should write the code and conduct the experiments on your own. Similarly, you should conduct the analysis on your own. You may read the literature on SAT solving, but you may not download code.

The project will be graded according to quality of your implementation of DPLL, the presentation of the results, and the quality of summary and analysis.