Programming Assignment 3

CMPE 250, Data Structures and Algorithms, Fall 2014

Instructor: A. T. Cemgil
TA's: Barış Kurt, Atakan Arıkan

Due: 14 December 2014, 23:59

2-Satisfiability Problem

2-satisfiability is the problem of determining whether a collection of binary variables with constraints on pairs of variables can be assigned values satisfying all the constraints. A 2-SAT problem may be described using a Boolean expression with a special restricted form: a conjunction (AND) of disjunctions (ORs), where each OR operation has two arguments that may either be variables or the negations of variables. For example, the expression :

$$(x_1 \vee x_2) \wedge (x_2 \vee \neg x_3) \wedge (\neg x_2 \vee \neg x_4) \wedge (x_2 \vee x_4) \wedge (x_1 \vee x_4) \tag{1}$$

is satisfiable when we assign the following values to the variables:

$$x_1 = T, \quad x_2 = T, \quad x_3 = F, \quad x_4 = F$$

However, the expression

$$(x_1 \vee x_2) \wedge (\neg x_1 \vee x_2) \wedge (x_1 \vee \neg x_2) \wedge (\neg x_1 \vee \neg x_2) \tag{2}$$

is not satisfiable since we cannot find any value assignments for x_1 and x_2 that will make the expression true.

One possible solution to this problem is to search for an assignment by brute force: trying all possible assignments until we find an assignment which satisfies the expression. Obviously, this is not an acceptable solution since we have to check N^2 different assignments for N variables.

A clever solution to this problem, which uses strongly connected components from the graph theory, has been introduced by Aspvall, Plass & Tarjan in 1979.

Strongly Connected Components Solution

The algorithm starts by representing the boolean expression by a directed graph, called the implication graph. Then, the strongly connected components in the graph is found by a linear algorithm. Finally, the true/false values are assigned to the variables if possible.

Part 1: The Implication Graph

In the first step, the graph is created as follows.

1. For each variable x_i we introduce two vertices x_i and $\neg x_i$.
2. For each clause $(x_i \vee x_j)$, we add edges $\neg x_i \to x_j$ and $\neg x_j \to x_i$.

The implication graph for the boolean expression (1) is given in Figure 1.

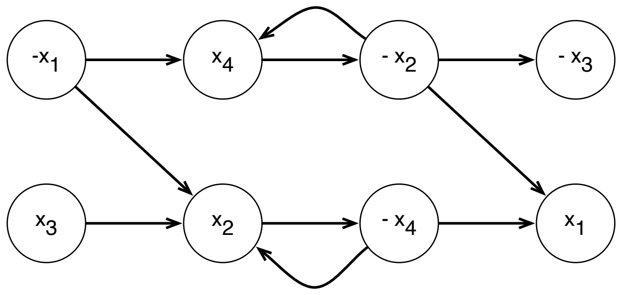

Figure 1: Implication graph for expression (1).

The following logic rule about implications, is the key for defining this graph.

$$(x_1 \vee x_2) \equiv (\neg x_1 \Rightarrow x_2) \equiv (\neg x_2 \Rightarrow x_1)$$

In the following steps, we are going to assign true/false labels to the vertices. In order to satisfy the boolean expression, the following rules must be satisfied:

- **Rule 1:** Each x_i and $\neg x_i$ must have complementary truth values.
- **Rule 2:** If there exist and edge $x_i \to x_j$, we can never assign $x_i = T$ and $x_j = F$.

Part 2: Strongly Connected Components

In the second step, we find the strongly connected components of the implication graph. A strongly connected component in a directed graph is a set of vertices such that within this set every vertex is reachable form every other vertex. A linear time algorithm to find the strongly connected components is the Tarjan's Algorithm. The graph in Figure 2 show the strongly connected component of our implication graph.

Part 3: Assignment of Truth Values

The boolean expression is satisfiable if and only if there is no vertex x_i and $\neg x_i$ in the same strongly connected component. Otherwise rules 1 and 2 cannot be satisfied. The steps in the final part is as follows:

1. Check whether there are two vertices x_i and $\neg x_i$ in any strongly connected component. If you find such vertices, output -1 and quit.
2. If the expression is satisfiable, you will see that the graph is skew-symmetric. For each component $S = \{x_i, x_j, \ldots\}$, there is another component $\neg S = \{\neg x_i, \neg x_j, \ldots\}$. Process the component in their topological order: if S is not assigned, assign $S = F$ and $\neg S = T$.

For this part, you don't need to topologically sort the components, because the Tarjan's algorithm in part 2 already generates the strongly connected components in the reverse topological order.

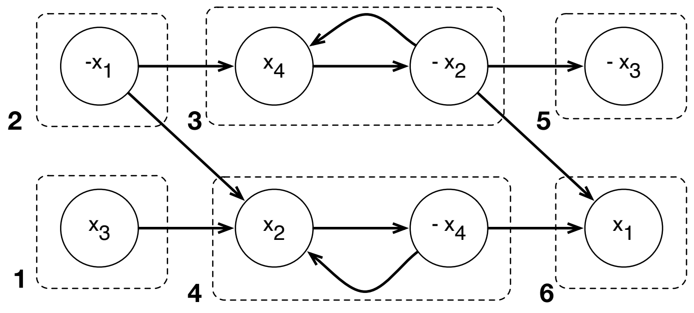

Figure 2: Strongly connected components of the implication graph. The numbers show the topological ordering of the components.

To Do:

In the assignment, you are going to implement the strongly connected components solution, and output the result after each part. You can run your program from the command line with one of the test cases with command:

```
./project3 testcases/inputX.txt
```

Your program should output 3 files in the *working directory*.

```
inputX_part1.txt
inputX_part2.txt
inputX_part3.txt.
```

The *Main.cc* file is written for you and you cannot change it, since it generates the file names for you. You are supposed implement your solution in the *SatSolver* class.

The boolean expression will be defined in an input file. The file for defining the expression (1) contains:

```
4 6
1 2
2 3
-2 -4
2 4
1 4
```

The first line defines the number of variables (4) and number of clauses (6). Each following line defines a single clause.

Part 1 (30 points)

You are supposed to output the implication graph that results from the expression. Output the adjacency list of the graph line by line to the output file. The adjacency list for the graph in Figure 1 will be like this:

```
-4 2 1
-3
-2 1 -3 4
-1 2 4
1
2 -4
3 2
4 -2
```

The order of lines and the order child vertices in each line is not important, since the list will be checked algorithmically.

Part 2 (40 points)

Output the strongly connected components to the output file. Each component will be written in a single line. The output file for the strongly connected components in Figure 2 will be:

```
-1
4 -2
-3
3
2 -4
1
```

The order of the components and vertices are not important.

Part 3 (30 points)

Output the truth values as numbers (1=TRUE, 0=FALSE) to the third output file. The assignments will be written in a single line. If the boolean expression is not satisfiable, output a single 1. The output file for the assignments for expression (1) will be:

```
1 1 0 0
```

The order of the assignments must be x_1 x_2 $x_3, \ldots$.

Submission Details

You are supposed to use the Git system provided to you for all projects. No other type of submission will be accepted. Also pay attention to the following points:

- All source codes are checked automatically for similarity with other submissions and exercises from previous years. Make sure you write and submit your own code.

- You are expected to use C++ as powerful, steady and flexible as possible. Use mechanisms that affects these issues positively.
- Make sure you document your code with necessary inline comments, and use meaningful variable names. Do not over-comment, or make your variable names unnecessarily long.