20875 Software Engineering – Assignment 1

Due Sunday, October 1st 2023, 23:59

The assignment consists in implementing a program that:
1. reads a Boolean formula in conjunctive normal form (CNF), and
2. prints its truth table.

Grading will be partially automated, and you are asked to follow precisely the instructions given in this document. The assignment is to be uploaded as a single archive file (either `.zip` or `.tgz`) on BlackBoard by the end of October 1st.

Rules. This is an individual assignment. You are allowed to talk about the assignment with your classmates. However, you must write all your code on your own. As a consequence, you will be able to explain and modify any part of your code upon request.

Input. The input is a Boolean formula f in the "DIMACS" CNF format specified here:
`https://www.poirrier.ca/courses/softeng/hw01/cnf.pdf`
Your program must be able to correctly parse any file that conforms to this specification.

Output. If the input file is in the correct format, your program outputs the truth table, and nothing else. The truth table must be in the following format:

```
<value of variables>,<value of formula>
<value of variables>,<value of formula>
<value of variables>,<value of formula>
...
```

Here, `<value of variables>` is a contiguous series of zeros and ones (not separated by anything), corresponding to the values of variables $x_n x_{n-1} \dots x_2 x_1$ (in that order), and `<value of formula>` is a single 0 or 1, the value of $f(x)$.

Moreover, the rows of the truth table must be in lexicographic order, i.e., they must start with all zeros, then $000\dots0001$, etc. For example, the truth table of $f(x_1, x_2, x_3) = x_1 \vee x_2$ would be printed as:

```
000,0
001,1
010,1
011,1
100,0
101,1
110,1
111,1
```

In the above example, the first column corresponds to x_3, the second to x_2 and the third to x_1. The last column gives $f(x_1, x_2, x_3)$.

Invocation. This assignment may be completed in Python or in C. The program will be invoked with the following commands:

Python: `python3 truthtable.py <table> formula.cnf`

C: `./truthtable <table> formula.cnf`

The `<table>` argument is either "`all`", "`ones`" or "`zeros`". If it is "`all`", your program displays the whole truth table. If it is "`ones`", it only displays the rows for which $f(x) = 1$. If it is "`zeros`", it only displays the rows for which $f(x) = 0$.

In the above example with $f(x) = x_1 \vee x_2$, the command `python3 truthtable.py zeros f.cnf` yields:

```
000,0
100,0
```

Compilation. If a `Makefile` is present in the submitted archive, the code will be compiled with the command:

```
make
```

Otherwise, it will be compiled with the command `clang -Wall -O3 -o truthtable *.c`

Grading.

- [4 marks] The program accepts all files that conform to the specification and correctly parses them.
- [1 mark] The program correctly rejects malformed files (no crash or uncaught exception/error).
- [3 marks] The program prints the correct output as specified in this document.
- [1 mark] The program supports CNF formulas of arbitrary size, in particular it can process formulas with at least 10,000 variables without running out of memory.
- [1 mark] The program can print the one entries of the truth table (`<table>` is "`ones`") for formulas with up to 24 variables (of sizes similar to `ag24_00`, ..., `ag24_15`, for which the "`ones`" table has 0-200 entries) under a minute. Bonus point for under a second.

Full example. A CNF formula for `XOR` is: `x1 xor x2 = (x or y) and ((not x) or (not y))`. A corresponding DIMACS CNF representation could be:

```
c This is a CNF file representing the formula x1 XOR x2
c
p cnf 2 2
1 2 0
-1 -2 0
```

If we store it in a file called `xor.cnf`, the full truth table could be obtained with the command:

Python: `python3 truthtable.py all xor.cnf`

C: `./truthtable all xor.cnf`

and the correct output would be:

```
00,0
01,1
10,1
11,0
```