ECE1733 Topics in Switching Theory Assignment #3

DPLL SAT Solver

Name: Shaoyu Chen

ID: 1002831091

Name: Zhehui Zhou

ID: 997716761

**Introduction**

The program reads a BLIF file that contains a CNF function, and determine its satisfiability by using the DPLL algorithm. The program is written in C++.

**Instructions**

Execution Instruction (Windows)

1. Unzip the **1773\_A3.zip** file, you can find a folder called **DPLL\_SAT\_solver**. Go to the sub-folder called **DPLL** and then **Release** which contains the executable file called **DPLL.exe**.
2. Open the windows command shell **cmd.exe**, and cd to the folder contains the **DPLL.exe** executable file described in step 1.
3. Type: DPLL.exe <path to the BLIF test file>, you can find some test files in the **test\_nodes** folder within the **1773\_A3** folder.
4. The program should start, and shows the major steps it used to find the satisfiability of the CNF function.

Compile Instruction (Visual Studio 2015)

1. In the Solution Explorer, right click on the **DPLL** project and select **property**.
2. Change the **Configuration** to **Release**, **Platform** to **win32**.
3. Under Configuration Properties, go to C/C++ and click Precompiled Headers. Change the option on **Precompiled Header** to **Not Using Precompiled Headers**.
4. Under Configuration Properties, go to C/C++ and click Preprocessor. Add the following to the **Preprocessor Definitions**: **\_CRT\_SECURE\_NO\_WARNINGS**
5. **Build** the solution under the **Release** and **x86** configuration.

The program is also available on GitHub: <https://github.com/marmot1234/DPLL_SAT_solver>

**Execution Report**

CNF1.blif: (a+b) (a+b’) (a’+c) (a’+c’)

CNF2.blif: (x1’+x2) (x1’+x2+x3’) (x1’+x3+x4’) (x1+x4)

CNF3.blif: (a+b’+d) (a+b’+e) (b’+d’+c’) (a+b+c+d) (a+b+c+d’) (a+b+c’+e) (a+b+c’+e’)

CNF4.blif: (x1’+x2’+x4’) (x1+x2’+x3’) (x2’+x3+x4’) (x1+x2+x3+x4) (x1’+x2’ +x4) (x1’+x2+x3)

CNF5.blif: (x1’+x2+x3) (x1+x3+x4) (x1+x3+x4’) (x1+x3’+x4) (x1+x3’+x4’) (x2’+x3’+x4) (x1’+x2+x3’) (x1’+x2’+x3)

|  |  |  |
| --- | --- | --- |
| Test Case | Satisfiability | Solution |
| CNF1 | UNSAT | N/A |
| CNF2 | SAT | 1 0 0 0 |
| CNF3 | SAT | 1 0 1 1 1 |
| CNF4 | SAT | 1 0 1 1 |
| CNF5 | SAT | 1 1 1 1 |

Please see the Appendix for the detailed information printed on the screen when the program is executed.

**Algorithm and Discussion**

The program implements the DPLL algorithm recursively:

Φ is a CNF function contains a set of input variables and a set of clauses

Unit Clause is a clause that only contain one variable

Pure Literal is a variable that only has one polarity in all clauses

function DPLL(Φ)

if Φ has no clause left

return true;

if Φ has no input variable left

return false;

for every unit clause l in Φ

assign l a proper value and remove l from input variables set in Φ;

removes clauses that contains l and evaluated to be true in Φ;

for every literal l that occurs pure in Φ

assign l a proper value and remove l from input variables set in Φ;

removes clauses that contains l and evaluated to be true in Φ;

if Φ has no clause left

return true;

if Φ has no input variable left

return false;

let l = the first variable in input variable set in Φ;

let Φ1 = Φ;

assign l to be **true** and remove l from input variables set in Φ1;

removes clauses that contains l and evaluated to be true in Φ1;

if DPLL(Φ1) == true

return true;

otherwise

let Φ2 = Φ;

assign l to be **false** and remove l from input variables in Φ2;

removes clauses that contains l and evaluated to be true in Φ2;

return DPLL(Φ2);

end

This program prints every major step it took during the execution to the screen, which makes the debugging process easier, and you can clearly see the deducing process in action.

To improve the performance of the program, it is possible to combine the process of finding unit clause and pure literal so that the number of loops to iterate the clauses is reduced. For a very large CNF function, the recursive method may crush due to occupying too much memory, so modifying the method from recursive to iterative could be a good idea. Also some article suggested to remove the process of finding pure literal because it is too expensive. Another improvement could be introducing a method to smartly choose branching variable, right now the program just choose the first variable in the input variable set.

**Appendix**

Detailed Execution Report, I used a Git BASH Shell to execute the program.

CNF1.blif:

$ ./DPLL.exe ../../test\_nodes/CNF1.blif

DPLL SAT SOLVER START!

READING FILE ../../test\_nodes/CNF1.blif

a b c

1 1 -

1 0 -

0 - 1

0 - 0

a Branch 1 --- Deduce Variable: a (1)

b c

- 1

- 0

Unit Clause Found with Variable: c --- Deduce Variable: c (1)

b

-

Pure Literal Found: b --- Deduce Variable: b (0)

Back Track

a Branch 0 --- Deduce Variable: a (0)

b c

1 -

0 -

Unit Clause Found with Variable: b --- Deduce Variable: b (1)

c

-

Pure Literal Found: c --- Deduce Variable: c (0)

Back Track

The CNF function is Un-Satisfiable

CNF2.blif:

$ ./DPLL.exe ../../test\_nodes/CNF2.blif

DPLL SAT SOLVER START!

READING FILE ../../test\_nodes/CNF2.blif

x1 x2 x3 x4

0 0 - -

0 1 0 -

0 - 1 0

1 - - 1

x1 Branch 1 --- Deduce Variable: x1 (1)

x2 x3 x4

0 - -

1 0 -

- 1 0

Unit Clause Found with Variable: x2 --- Deduce Variable: x2 (0)

x3 x4

0 -

1 0

Unit Clause Found with Variable: x3 --- Deduce Variable: x3 (0)

x4

0

Unit Clause Found with Variable: x4 --- Deduce Variable: x4 (0)

The CNF function is Satisfiable with Solution:

x1(1) x2(0) x3(0) x4(0)

CNF3.blif:

$ ./DPLL.exe ../../test\_nodes/CNF3.blif

DPLL SAT SOLVER START!

READING FILE ../../test\_nodes/CNF3.blif

a b c d e

1 0 - 1 -

1 0 - - 1

- 0 - 0 0

1 1 1 1 -

1 1 1 0 -

1 1 0 - 1

1 1 0 - 0

Pure Literal Found: a --- Deduce Variable: a (1)

b c d e

0 - 0 0

Pure Literal Found: b --- Deduce Variable: b (0)

c d e

The CNF function is Satisfiable with Solution:

a(1) b(0) c(1) d(1) e(1)

CNF4.blif:

$ ./DPLL.exe ../../test\_nodes/CNF4.blif

DPLL SAT SOLVER START!

READING FILE ../../test\_nodes/CNF4.blif

x1 x2 x3 x4

0 0 - 0

1 0 0 -

- 0 1 0

1 1 1 1

0 0 - 1

0 1 1 -

x1 Branch 1 --- Deduce Variable: x1 (1)

x2 x3 x4

0 - 0

0 1 0

0 - 1

1 1 -

Pure Literal Found: x3 --- Deduce Variable: x3 (1)

x2 x4

0 0

0 1

Pure Literal Found: x2 --- Deduce Variable: x2 (0)

x4

The CNF function is Satisfiable with Solution:

x1(1) x2(0) x3(1) x4(1)

CNF5.blif:

$ ./DPLL.exe ../../test\_nodes/CNF5.blif

DPLL SAT SOLVER START!

READING FILE ../../test\_nodes/CNF5.blif

a b c d

0 1 1 -

1 - 1 1

1 - 1 0

1 - 0 1

1 - 0 0

- 0 0 1

0 1 0 -

0 0 1 -

a Branch 1 --- Deduce Variable: a (1)

b c d

1 1 -

0 0 1

1 0 -

0 1 -

Pure Literal Found: d --- Deduce Variable: d (1)

b c

1 1

1 0

0 1

b Branch 1 --- Deduce Variable: b (1)

c

1

Unit Clause Found with Variable: c --- Deduce Variable: c (1)

The CNF function is Satisfiable with Solution:

a(1) b(1) c(1) d(1)