

Discrete Mathematics

Homework 2

Converting a formula to DNF

Shin Hong

15 Nov, 2019

Assignment Overview

2

- Write a C program that (1) converts a given propositional formula to a Disjunctive Normal Form and (2) find a solution that satisfies the formula
- Submission
 - deadline: 20 Nov (Wed), 11:59 PM
 - deliverables
 - program (C source code files, build script, README)
 - write up (1 page)
- You are not allowed to collaborate with any one in or out of the class in doing this homework
 - you can reuse & reference the PA 2 result by your team

Disjunctive Normal Form (DNF)

- A DNF formula is a conjunctive clause or disjunction of multiple conjunctive clauses
 - a literal is an atomic propositional variable or its negation
 - a conjunctive clause is a conjunction of one or multiple literals
 - a DNF is in a form of $(a_{11} \wedge a_{12} \wedge \cdots \wedge a_{1n_1}) \vee (a_{21} \wedge a_{22} \wedge \cdots \wedge a_{2n_2}) \vee \cdots (a_{m1} \wedge a_{m2} \wedge \cdots \wedge a_{mn_m})$ where a_{ij} is a literal
- Every propositional formula can be converted to an equivalent DNF formula

Requirement (1/3)

4

- Receive a propositional formula in the same grammar as PA 2
 - A propositional formula $\phi \in P$ is specified by the following rules:
 - **a** $n \in P$ for $n \in \mathbb{N}$
 - **(and** $\phi_1 \phi_2 \dots \phi_m$) for $\phi_i \in P$
 - **(or** $\phi_1 \phi_2 \dots \phi_m$) for $\phi_i \in P$
 - **(not** ϕ) for $\phi \in P$
- For a given formula, your program must print out the following two:
 1. an equivalent DNF formula
 2. satisfiability checking result
 - print out a solution (assignment) if the formula is satisfiable
 - print “UNSAT” otherwise

Requirement (2/3)

5

- Output 1. Represent a converted DNF as follows
 - each line represent a conjunctive clause
 - each line has a list of positive integers and negative integers
 - a positive integer n represents an atomic propositional variable a_n
 - a negative integer $-n$ represents the negation of an atomic propositional variable $\neg a_n$
 - print a line with “0” after the DNF representation (as a delimiter)
- Output 2. Represent a solution as a list of integers s_1, s_2, \dots, s_n where s_i is either i or $-i$
 - i means that a_i is True for the solution
 - $-i$ means that a_i is False for the solution

Requirement (3/3)

6

- **Example**

- Input

```
(or a1 (not (or (not (or a2 a3)) a4)))
```

- Output

```
1
2 -4
3 -4
0
-1 2 3 -4
```

DNF Conversion

7

1. Define a tree-like data structure to represent a propositional formula
2. Write a recursive algorithm to translate a given text to a formula structure
3. Write a recursive algorithm that converts a given formula to a Negation Normal Form (NNF)
 - a NNF has a negation only at a leaf node (i.e., as $\neg an$)
4. Write a recursive algorithm that applies the Distributive law to transform a NNF into a DNF

Program Structure

8

- Write a C program running on UNIX/LINUX
 - Each program receives input from the standard input and produces the output to the standard output
 - Tests will be conducted on Peace
- The program must be built as a single executable
- You must not use Z3 to find a solution (because you do not need to do)
- You must submit a build script and README together with source code files
 - build script: Bash script, Makefile, Ant, Maven, etc.
 - README: instruction/manual on how to build and run your program

Homework 2

Discrete Math.

2019-11-15

Evaluation Criteria

9

- Write up (60 points)
 - Description (45 points)
 - check whether you found all constraints of a solution
 - check whether each constraint is correctly represented as a logic formula
 - check whether you demonstrate the correctness of your programs in a convincing way (e.g., by tests)
 - check whether all descriptions are clear and consistent
 - Discussion (15 points)
 - detailed analysis of results, interesting observations, lessons learned, suggestions, new ideas, etc.
- Tests (40 points)
 - Run each program with several inputs to see whether the results are correct