Logic Synthesis & Verification, Fall 2023

National Taiwan University

Problem Set 2

Due on 2023/10/15 by 23:59

1 [Cofactor]

(10%) Given two Boolean functions f and g and a Boolean variable v, prove the following equalities.

(a) (5%) $(\neg f)_v = \neg(f_v)$, and
(b) (5%) $(f \rightarrow g)_v = (f_v) \rightarrow (g_v)$.

2 [Quantification]

(20%)

(a) (8%) Consider the following 8 quantified Boolean formulas

$$
\begin{aligned}
F_1 &: \exists x, \exists y. f(x, y, z), \\
F_2 &: \neg(\exists y, \exists x. \neg f(x, y, z)), \\
F_3 &: \exists x, \forall y. f(x, y, z), \\
F_4 &: \neg(\exists y, \forall x. \neg f(x, y, z)), \\
F_5 &: \forall y, \exists x. f(x, y, z), \\
F_6 &: \forall x, \exists y. f(x, y, z), \\
F_7 &: \forall x, \forall y. f(x, y, z), \\
F_8 &: \forall y, \forall x. f(x, y, z).
\end{aligned}
$$

List the set of implications $F_i \rightarrow F_j$ for $i, j = 1, \ldots, 8$ and $i \neq j$.

(b) (3%) Prove or disprove

$$\exists x.(f(x, y) \vee g(x, y)) = (\exists x. f(x, y) \vee \exists x. g(x, y)).$$

(c) (3%) Prove or disprove

$$\exists x.(f(x, y) \vee g(x, y)) = \exists x. f(x, y) \vee \forall x. g(x, y).$$

(d) (3%) Prove or disprove

$$\forall x.(f(x, y) \vee g(y)) = (\forall x. f(x, y)) \vee g(y).$$

(e) (3%) Prove or disprove

$$\exists x.(f(x, y) \rightarrow g(x, y)) = (\exists x. f(x, y)) \rightarrow (\exists x. g(x, y)).$$

3 [BDD and ITE]

(15%) Let $f = (a \oplus b \oplus c) \wedge (b \oplus c \oplus d)$.

(a) (5%) Draw the ROBDD of f with variable ordering $a < b < c < d$ (with a on top).
(b) (5%) Draw the ROBDDs of f_b and $f_{\neg b}$ as shared ROBDDs along with that of f.
(c) (5%) Apply the ITE operation on the above ROBDDs to compute $\frac{\partial f}{\partial b}$.

4 [BDD Onset Counting]

(10%) Design a linear-time algorithm that counts the number of onset minterms of a given ROBDD.

5 [ZDD]

(10%) Consider the graph of Figure 1.

(a) (5%) Construct a ZDD that represents the set of simple paths (no vertex reappearance) over the edges connecting the leftmost vertex to the rightmost vertex. (E.g., 26548 is a simple path, but not 265479.)
(b) (5%) Construct a ZDD that represents the set of Hamiltonian paths (all vertex visited exactly once) over the edges connecting the leftmost vertex to the rightmost vertex.

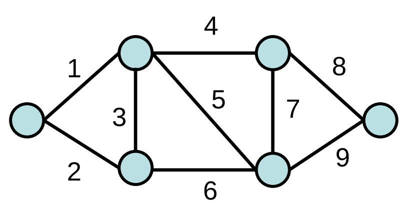

Fig. 1. Graph for path enumeration.

6 [SAT Solving]

(15%) Consider SAT solving the CNF formula consisting of the following ten clauses

$$C_1 = (a + b + c), C_2 = (a + b + c' + d'), C_3 = (a + b' + c),$$
$$C_4 = (a + b' + c'), C_5 = (a + c' + d), C_6 = (a' + b + c), C_7 = (a' + b' + d),$$
$$C_8 = (a' + b' + c' + d'), C_9 = (b + d), C_{10} = (b' + c + d').$$

(a) (7%) Apply implication and conflict-based learning to solve the above CNF formula. Assume that the decision order follows a, b, c, and then d; assume each variable is assigned 0 first and then 1; assume the implications are prioritize by the clause index in case there would be multiple ways of leading to a conflict. Whenever a conflict occurs, draw the implication graph and enumerate all possible learned clauses under the Unique Implication Point (UIP) principle. (In your implication graphs, annotate each vertex with "`variable = value@decision_level`", e.g., "$b = 0@2$", and annotate each edge with the clause that implication happens.) If there are multiple UIP learned clauses for a conflict, pick the one with the UIP closest to the conflict vertex in the implication graph.

(b) (8%) The **resolution** between two clauses $C_i = (C_i^* + x)$ and $C_j = (C_j^* + x')$ (where C_i^* and C_j^* are sub-clauses of C_i and C_j, respectively) is the process of generating their **resolvent** $(C_1^* + C_j^*)$. The resolution is often denoted as

$$\frac{(C_i^* + x) \qquad (C_j^* + x')}{(C_1^* + C_j^*)}$$

A fact is that a learned clause in SAT solving can be derived by a few resolution steps. Show how that the learned clauses of (a) can be obtained by resolution with respect to their implication graphs.

7 [SAT Solving]

(20%)

(a) (8%) Write a CNF formula to encode the Pigeon-Hole Principle for m pigeons and n holes, denoted PHP_n^m, such that every hole lives at most one pigeon and every pigeon lives in some hole. The formula must reflect the fact that a satisfying assignment to the formula corresponds to a legitimate pigeon-hole assignment. What is the size of the formula in terms of m and n?

(b) (6%) Use MiniSAT (http://minisat.se/) to solve the pigeon-hole problem for $m = n = 4, 5, 6$. (Note that the formulas should be in the DIMACS format http://www.satcompetition.org/2009/format-benchmarks2009.html.)
Print out the MiniSAT statistics. Do you expect the solver is scalable on this problem? Why or why not?

(c) (6%) Use MiniSAT (http://minisat.se/) to solve the pigeon-hole problem for $m = n + 1 = 4, 5, 6$. (Note that the formulas should be in the DIMACS format
http://www.satcompetition.org/2009/format-benchmarks2009.html.)
Print out the MiniSAT statistics. Do you expect the solver is scalable on this problem? Why or why not?