Logic Synthesis & Verification, Fall 2024

National Taiwan University

Problem Set 2

Due on 2024/10/18 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%) $(f \oplus g)_v = (f_v) \oplus (g_v)$.

2 [Quantification]

(20%)

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

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

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

(b) (3%) Prove or disprove

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

(c) (3%) Prove or disprove

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

(d) (3%) Prove or disprove

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

(e) (3%) Prove or disprove

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

3 [BDD and ITE]

(15%) Let $f = ab(c + \neg d) + (a\neg b + \neg ab)(\neg cd + c\neg d)$.

(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 $\forall b.f$.

4 [BDD for Counting]

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

5 [ZDD]

(20%) Consider coloring the graph of Figure 1 with 3 colors such that no two vertices receive the same color if they are connected by an edge.

(a) (5%) Construct a ZDD that represents the set of all possible 3-coloring solutions.
(b) (10%) Develop a linear-time algorithm that counts the number of subsets in a set represented by a ZDD.
(c) (5%) Apply the algorithm of (b) on the ZDD of (a) to calculate the number of 3-coloring solutions of the given graph.

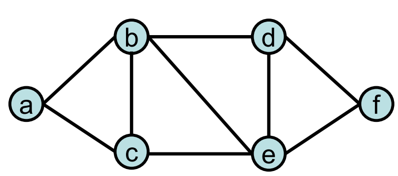

Fig. 1. Graph for coloring.

6 [SAT Solving]

(10%) Consider SAT solving the CNF formula consisting of the following ten clauses

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

(a) (5%) 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) (5%) 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]

(15%)

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

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