Logic Synthesis & Verification, Fall 2024

National Taiwan University

Problem Set 2

Due on 2024/10/18 23:59 on NTU Cool.

1 [Cofactor]

(a) Let $f = f(x_1, \ldots, x_n)$ be an n-variable Boolean function and $v \in \{x_k, \neg x_k\}$ be the literal corresponding to the k^{th} variable. By definition, $\neg f$ is defined by $(\neg f)^1 = f^0$ and $(\neg f)^0 = f^1$. In other words, $(\neg f)(m_1, \ldots, m_n) = (f(m_1, \ldots, m_n))'$ for any input assignment $m_1 \ldots m_n$, where $m_i \in \{0, 1\}$. Similarly, $(\neg(f_v))(m_1, \ldots, m_n) = (f_v(m_1, \ldots, m_n))'$ for any input assignment $m_1 \ldots m_n$. Now consider an arbitrary input assignment $m_1 \ldots m_n$. Then we have

$$\begin{aligned}((\neg f)_v)(m_1, \ldots, m_n) &= (\neg f)(m_1, \ldots, m_{k-1}, V(v), m_{k+1}, \ldots, m_n) \\ &= (f(m_1, \ldots, m_{k-1}, V(v), m_{k+1}, \ldots, m_n))' \\ &= (f_v(m_1, \ldots, m_n))' \\ &= (\neg(f_v))(m_1, \ldots, m_n),\end{aligned}$$

where

$$V(v) = \begin{cases} 0, & \text{if } v = \neg x_k \\ 1, & \text{otherwise} \end{cases}.$$

Since $(\neg f)_v$ and $\neg(f_v)$ have the same values for all input assignments, they have the same onset and offset. Therefore, $(\neg f)_v = \neg(f_v)$.

(b) Let $f = f(x_1, \ldots, x_n)$ and $g = g(x_1, \ldots, x_n)$ be two n-variable Boolean functions and $v \in \{x_k, \neg x_k\}$ be the literal corresponding to the k^{th} variable. Then $f \oplus g$ is defined by $(f \oplus g)^1 = (f^0 \cap g^1) \cup (f^1 \cap g^0)$ and $(f \oplus g)^0 = (f^0 \cap g^0) \cup (f^1 \cap g^1)$. In other words, $(f \oplus g)(m_1, \ldots, m_n) = (f'(m_1, \ldots, m_n)g(m_1, \ldots, m_n)) + (f(m_1, \ldots, m_n)g'(m_1, \ldots, m_n))$ for any input assignments $m_1, \ldots, m_n$, where $m_i \in \{x_i, \neg x_i\}$. Similarly, $(f_v \oplus g_v)(m_1, \ldots, m_n) = (f'_v(m_1, \ldots, m_n)g_v(m_1, \ldots, m_n)) + (f_v(m_1, \ldots, m_n)g'_v(m_1, \ldots, m_n))$ for any input assignments $m_1, \ldots, m_n$, where $m_i \in \{x_i, \neg x_i\}$. Now consider an arbitrary input assignment $m_1, \ldots, m_n$. Then we have

$$\begin{aligned}((f \oplus g)_v)(m_1, \ldots, m_n) &= (f \oplus g)(m_1, \ldots, m_{k-1}, V(v), m_{k+1}, m_n) \\ &= (f'(m_1, \ldots, m_{k-1}, V(v), m_{k+1}, m_n)g(m_1, \ldots, m_{k-1}, V(v), m_{k+1}, m_n)) \\ &+ (f(m_1, \ldots, m_{k-1}, V(v), m_{k+1}, m_n)g'(m_1, \ldots, m_{k-1}, V(v), m_{k+1}, m_n)) \\ &= (f'_v(m_1, \ldots, m_n)g_v(m_1, \ldots, m_n)) + (f_v(m_1, \ldots, m_n)g'_v(m_1, \ldots, m_n)) \\ &= (f_v \oplus g_v)(m_1, \ldots, m_n)\end{aligned}$$

where

$$V(v) = \begin{cases} 0, & \text{if } v = \neg x_k \\ 1, & \text{otherwise} \end{cases}.$$

Since $(f \oplus g)_v$ and $(f_v) \oplus (g_v)$ have the same values for all input assignments, they have the same onset and offset. Therefore, $(f \oplus g)_v = (f_v) \oplus (g_v)$.

2 [Quantification]

(a)

$$\begin{aligned}
&F_1 \to F_2 \\
&F_2 \to F_1 \\
&F_3 \to F_1, F_2, F_4, F_5, F_6, F_7, F_8 \\
&F_4 \to F_1, F_2, F_3, F_5, F_6, F_7, F_8 \\
&F_5 \to F_1, F_2, F_6, F_8 \\
&F_6 \to F_1, F_2, F_8 \\
&F_7 \to F_1, F_2 \\
&F_8 \to F_1, F_2, F_6
\end{aligned}$$

(b) False. Here is a counter-example. Let $f(x, y) = x, g(x, y) = \neg x$

$$\begin{aligned}
\forall x.(f(x, y) \vee g(x, y)) &= \forall x.(x \vee \neg x) \\
&= \forall x.(\underline{1}) \\
&= \underline{1}
\end{aligned}$$

while

$$\begin{aligned}
(\forall x.f(x, y)) \vee (\forall x.g(x, y)) &= (\forall x.(x)) \vee (\forall x.(\neg x)) \\
&= (\underline{0}) \vee (\underline{0}) \\
&= \underline{0}
\end{aligned}$$

(c) True. The proof is as follows.

$$\begin{aligned}
\exists x.(f(x, y) \vee g(x, y)) &= (f(0, y) \vee g(0, y)) \vee (f(1, y) \vee g(1, y)) \\
&= (f(0, y) \vee f(1, y)) \vee (g(0, y) \vee g(1, y)) \\
&= (\exists x.f(x, y)) \vee (\exists x.g(x, y))
\end{aligned}$$

(d) True. The proof is as follows.

$$\begin{aligned}
\forall x.(f(x, y) \vee g(y)) &= (f(0, y) \vee g(y)) \wedge (f(1, y) \vee g(y)) \\
&= (f(0, y) \wedge f(1, y)) \vee g(y) \\
&= (\forall x.f(x, y)) \vee g(y)
\end{aligned}$$

(e) True.

$$\begin{aligned}\exists x.(f(x,y) \to g(x,y)) &= (\neg f(0,y) \vee g(0,y)) \vee (\neg f(1,y) \vee g(1,y)) \\ &= (\neg f(0,y) \vee \neg f(1,y)) \vee (g(0,y) \vee g(1,y)) \\ &= \neg(f(0,y) \wedge f(1,y)) \vee (g(0,y) \vee g(1,y)) \\ &= (\forall x.f(x,y)) \to (\exists x.g(x,y))\end{aligned}$$

3 [BDD and ITE]

(a) see the following figure.

(b) see the following figure.

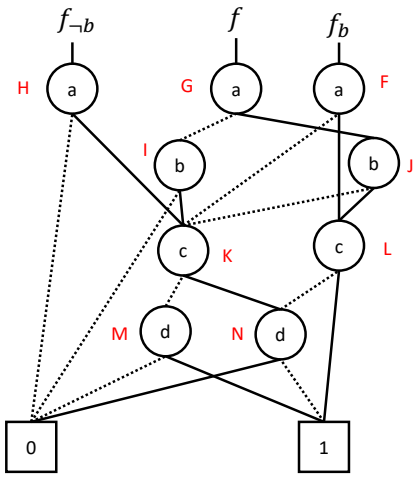


Fig. 1. Solution for Problem 5.

(c) $\forall b.f.$

$$
\begin{aligned}
\forall b.f. &= f_b \cdot f_{\neg b} \\
&= \mathrm{ITE}(f_b, f_{\neg b}, 0) \\
&= \mathrm{ITE}(F, H, 0) \\
&= \mathrm{ITE}(a, \mathrm{ITE}(L, K, 0), \mathrm{ITE}(K, 0, 0)) \\
&= \mathrm{ITE}(a, \mathrm{ITE}(L, K, 0), 0) \\
&= \mathrm{ITE}(a, \mathrm{ITE}(c, \mathrm{ITE}(1, N, 0), \mathrm{ITE}(N, M, 0)), 0) \\
&= \mathrm{ITE}(a, \mathrm{ITE}(c, N, \mathrm{ITE}(N, M, 0)), 0) \\
&= \mathrm{ITE}(a, \mathrm{ITE}(c, N, \mathrm{ITE}(d, \mathrm{ITE}(0, 1, 0), \mathrm{ITE}(1, 0, 0))), 0) \\
&= \mathrm{ITE}(a, \mathrm{ITE}(c, N, \mathrm{ITE}(d, 0, 0)), 0) \\
&= \mathrm{ITE}(a, \mathrm{ITE}(c, N, 0), 0)
\end{aligned}
$$

4 [BDD for Counting]

Algorithm 1: ROBDD Onset Counting

Input : G: an ROBDD;
n: the number of variables
Output: $G.const_1.count$: The number of onset minterms of G

```
for each node v do
    v.count ← 0
end
G.root.count ← 2^n
for each node v in the top-down order do
    v.left.count ← v.left.count + v.count/2
    v.right.count ← v.right.count + v.count/2
end
return G.const_1.count
```

5 [ZDD]

(a) For each node n, let Boolean variables n_0, n_1, n_2 represents coloring node n with color $0, 1, 2$ respectively. A valid solution should have exactly one of these three variables assigned to true. In fig. 2, we can see that (a, e) must

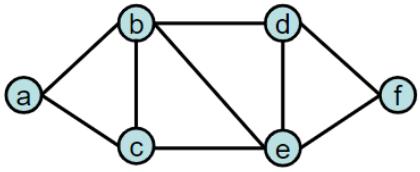


Fig. 2. Graph for coloring

be colored the same, so do (b, f) and (c, d). Therefore, with the following variable order

$$a_0, a_1, a_2, e_0, e_1, e_2, b_0, b_1, b_2, f_0, f_1, f_2, c_0, c_1, c_2, d_0, d_1, d_2,$$

we can derive the ZDD shown in fig. 3

(b) To compute the number of paths from the root node to the constant one node, we can simply enumerate nodes from the bottom level (leaf nodes) to

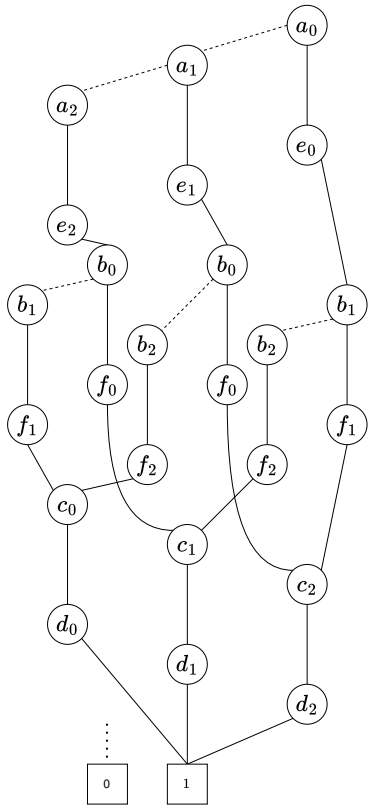


Fig. 3. ZDD for problem 5(a).

the top level (root nodes) and compute the number of paths from each node to the constant one node. For each node n, the number of paths from n to the constant one node $P(n)$ is

$$P(n) = \begin{cases} P(n.then) + P(n.else)\,, \text{if } n \text{ is not a leaf node} \\ 1\,, \text{if } n \text{ is the constant one node} \\ 0\,, \text{if } n \text{ is the constant zero node} \end{cases}$$

where $n.then$ and $n.else$ are the children of n. The algorithm is linear time because each node is visited once.

(c) The count are shown next to each node in fig. 4. The number of solutions, i.e. the number of path from the root node to the constant 1 node, is 6.

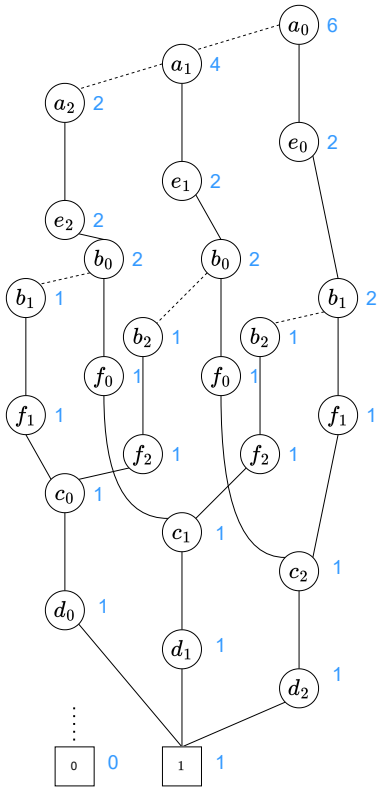
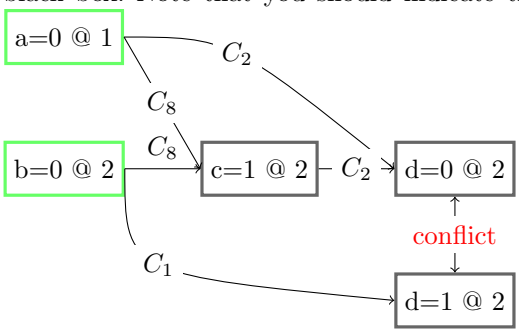


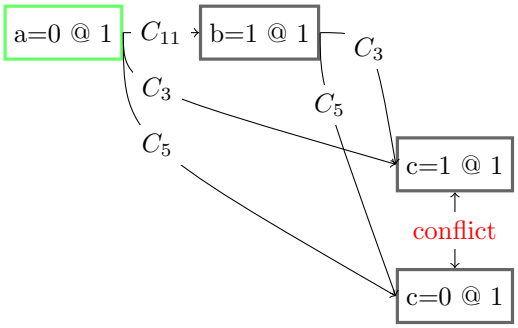
Fig. 4. Result of problem 5(c).

6 [SAT Solving]

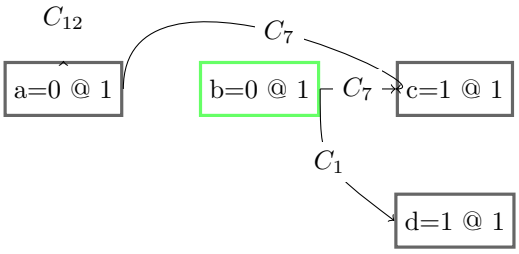
(a) Decision nodes are marked green, while implications are presented in simple black box. Note that you should indicate the decision level of each node.



All UIP clauses: (a + b) Learnt clause C_{11} = (a + b)



All UIP clauses: (a) Learned clause C_{12} = (a)



Satisfying assignment: (a, b, c, d) = (1, 0, 1, 1)

(b) 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** $(C_i^* + C_j^*)$. The resolution is often denoted as

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

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.
Clause C_{11}:

$$C_1 = (b + d),\ C_2 = (a + b + c' + d'),\ C_8 = (a + b + c)$$

Resolution

$$\frac{\dfrac{C_1 \qquad C_2}{(a + b + c')} \qquad C_8}{(a + b)}$$

Clause C_{12}:

$$C_3 = (a + b' + c), C_5 = (a + b' + c'), C_{11} = (a + b)$$

Resolution

$$\frac{\dfrac{C_3 \qquad C_5}{(a + b')} \qquad C_{11}}{(a)}$$

7 [SAT Solving]

(a) We introduce variables $P_{i,j}$ for $1 \leq i \leq m$, $1 \leq j \leq n$. Let $P_{i,j} = 1$ if and only if the pigeon i is in the hole j. Then the CNF contains two parts:

$$\bigwedge_{i=1}^{m} \bigvee_{j=1}^{n} P_{i,j} \tag{1}$$

poses the constraint that each pigeon must be in some hole;

$$\bigwedge_{j=1}^{n} \bigwedge_{i=1}^{m-1} \bigwedge_{k=i+1}^{m} (\neg P_{i,j} \vee \neg P_{k,j}) \tag{2}$$

poses the constraint that each hole contains at most one pigeon. The formula size is $m + 0.5nm(m-1)$ (in terms of number of clauses).
Some may additionally pose the constraint that one pigeon can only be in at most one hole:

$$\bigwedge_{i=1}^{m} \bigwedge_{j=1}^{n-1} \bigwedge_{k=j+1}^{n} (\neg P_{i,j} \vee \neg P_{i,k}) \tag{3}$$

Then the formula size would become $m + 0.5nm(m-1) + 0.5mn(n-1)$ (in terms of number of clauses).

(b) Yes, the solver is expected to be scalable on this problem because the CNF for the case $n = m + 1$ is satisfiable. The solver can evoke some implications to lead to a satisfying assignment.

(c) No, the solver is not expected to be scalable on this problem because the CNF for the case $n = m - 1$ is unsatisfiable, and the number of backtracks in solver grows exponentially.

Note: The scalability trend may not seem clear for $m = 4, 5, 6$. However, if you experiment with values from $m = 7$ to $m = 11$, you will find that when $m = n - 1$, the required number of decisions is approximately $O(m^2)$. In contrast, the required number of decisions grows exponentially when $m = n + 1$.