

Logic Synthesis & Verification, Fall 2014

National Taiwan University

Problem Set 3 Solution

1 [Symmetric Functions]

for relation S_1 :

- ① $\forall x_i \in X, f(\dots, x_i, \dots) = f(\dots, \bar{x}_i, \dots)$, $\therefore (x_i, x_i) \in S_1$ for all $x_i \in X \Rightarrow S_1$ is reflexive.
- ② if $(x_i, x_j) \in S_1$, then $f(\dots, x_i, x_j, \dots) = f(\dots, x_j, x_i, \dots)$
 $\therefore f(\dots, x_j, x_i, \dots) = f(\dots, x_i, x_j, \dots) \quad \therefore (x_j, x_i) \in S_1 \Rightarrow S_1$ is symmetry.
- ③ if $(x_i, x_j) \in S_1$ and $(x_j, x_k) \in S_1$, then $f(\dots, x_i, x_j, \dots) = f(\dots, x_j, x_i, \dots) \quad \text{④}$
 $f(\dots, x_j, x_k, \dots) = f(\dots, x_k, x_j, \dots) \quad \text{⑤}$
 $\therefore f(\dots, x_i, x_j, x_k, \dots) = f(\dots, x_j, x_i, x_k, \dots) = f(\dots, x_j, x_k, x_i, \dots)$
by ④
by ⑤
 $\therefore f(\dots, x_k, x_j, x_i, \dots) \quad \therefore (x_i, x_k) \in S_1 \Rightarrow S_1$ is transitive.

$\therefore S_1$ is reflexive, symmetry and transitive $\therefore S_1$ forms a equivalence relation.

for relation S_2 :

- ① $\forall x_i \in X, f(\dots, x_i, \dots)$ is not equal to $\neg f(\dots, x_i, \dots) \Rightarrow S_2$ is not reflexive.
- ② if $(x_i, x_j) \in S_2$, then $f(\dots, x_i, x_j, \dots) = \neg f(\dots, x_j, x_i, \dots) \quad \text{⑥}$
perform negation on ⑥ $\neg f(\dots, x_i, x_j, \dots) = \neg (\neg f(\dots, x_j, x_i, \dots)) = f(\dots, x_j, x_i, \dots)$
 $\therefore (x_j, x_i) \in S_2 \Rightarrow S_2$ is symmetry.
- ③ if $(x_i, x_j) \in S_2$ and $(x_j, x_k) \in S_2$, then $f(\dots, x_i, x_j, \dots) = f(\dots, x_j, x_i, \dots) \quad \text{⑦}$
 $f(\dots, x_i, x_j, x_k, \dots) = \neg f(\dots, x_j, x_i, x_k, \dots) \quad \text{⑧}$
 $f(\dots, x_i, x_j, x_k, \dots) = \neg f(\dots, x_j, x_i, x_k, \dots) = \neg (\neg f(\dots, x_j, x_k, x_i, \dots))$
 $= f(\dots, x_j, x_k, x_i, \dots) = \neg f(\dots, x_k, x_j, x_i, \dots) \Rightarrow (x_i, x_k) \in S_2 \quad \therefore S_2$ is transitive.

double negation However, S_2 doesn't form a equivalence relation since S_2 is not reflexive.

2 [Unate Functions]

2. [Unate Function]

in unate function

For an arbitrary prime implicant PI^V , we can definitely find a minterm $m \in PI$ and m is not in any other prime implicant by setting $m = \oplus$ if literal appears in PI , set this literal \ominus if literal doesn't appear in PI , set it to the opposite phase of its unate direction.
It is obvious that $m \in PI$. Then we want to prove that m is not in any other prime implicant.

proof. For each other prime implicant PI_i , we find some literals in PI_i but not in PI must conflict in m and PI_i since we set those literals to the opposite phase of their unate direction and they must be their unate direction in PI_i . We can definitely find those literals since both PI_i and PI are prime implicants and f is unate (common literals must in same unate direction, if can't find, PI can be expanded to PI_i , which is not the case.) By definition and proof above, PI must be an essential prime implicant.

Ex: $f = ab + b\bar{c} + \bar{a}\bar{c}$, for $PI = ab$, we set $m = \underbrace{ab}_{\text{opposite}} \oplus abc \in ab \quad \text{⑨}$ $ab\bar{c} \notin bc$
pos: a, b
neg: c
 $\therefore ab$ is an essential prime implicant.

3 [Generalized Cofactor]

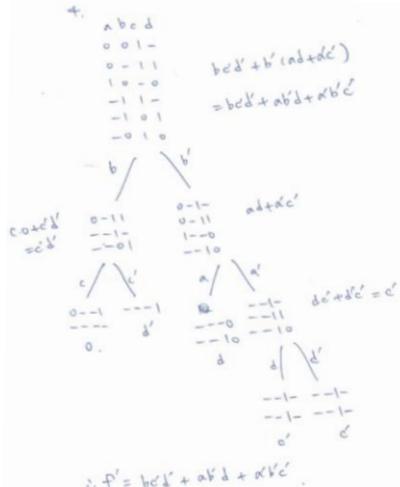
$$\begin{aligned}
 (a) \quad g \cdot co(\neg f, g) &= g \cdot (\neg f'g, g', fg)' = (\neg f'g, \emptyset, fg + g') = \neg f'g \quad (\text{'completely specified'}) \\
 \neg g \cdot \neg co(\neg f, \neg g) &= \neg g \cdot \neg (\neg f'g, g, fg') = g' (\neg f'g, g, f'g') = (fg', \emptyset, f'g' + g) = fg' \\
 g \cdot co(\neg f, g) + \neg g \cdot \neg co(\neg f, \neg g) &= \neg f'g + f'g' \neq \neg f, \quad \text{disproved}
 \end{aligned}$$

$$\begin{aligned} \text{(b)} \quad & co(co(f,g), h) = co((fg, \gamma g, (\gamma f)g), h) = (hfg, \gamma g + \gamma h, h(\gamma f)g) \\ & = (f(gh), \gamma(gh), (\gamma f)(gh)) = co(f, gh) \end{aligned}$$

$$\begin{aligned}
 (c) \quad & \omega(f, h) \cdot \omega(g, h) = (fh, h', f'h) \cdot (gh, h', g'h) = (fh \cdot gh, h' + fhh' + ghh', f'h + g'h) \\
 & = (fgh, h', f'h + g'h) = (fgh, h', (fg)h) \\
 & = \omega(fg, h) \quad \text{proved.}
 \end{aligned}$$

(d) $\omega(f,g) = \text{onset } fg \text{ offset } f'g \text{ don't-care see } g'$
 $\Rightarrow \neg co(f,g) \text{ exchange the onset and offset } f \text{ co}(f,g).$
 $\neg \omega(f,g) \text{ inset } f'g \text{ offset } fg \text{ don't-care see } g'$
 $\neg co(f,g) = (f'g, g', fg) = (f'g, g', (f')'g) = co(\neg f, g) \text{ proved}$

4 [Unate Recursive Paradigm: Complementation]



5 [Minimum Column Covering]

3. [Minimum Column Covering]
 We formulate the problem as follows: Each column is a variable (n variables). A literal becomes true iff the corresponding column is selected. Each row corresponds to a clause. A literal is added to a clause if the corresponding position is 1.

$$\begin{array}{ccccc} \text{Ex:} & p_1 & p_2 & p_3 & \\ \begin{matrix} M_1 \\ M_2 \\ M_3 \\ M_4 \end{matrix} & \begin{matrix} 1 \\ 0 \\ 1 \\ 0 \end{matrix} & \begin{matrix} 0 \\ 1 \\ 1 \\ 1 \end{matrix} & \Rightarrow (p_1 + p_3) (p_2 + p_3) (p_1 + p_2) p_2 \end{array}$$

Then, we query the solver by binary search. First, query the solver whether 1 column covering exists. To ensure this, add $2^n - C_1$ clauses ($P_1 \oplus P_2 \dots \oplus P_n$) ($P_1 \oplus P_2 \dots \oplus P_n' \oplus P_n'' \dots \oplus P_n'''$). If SAT, 1 column covering solution is found. If UNSAT, query the solver whether $\lfloor \frac{h}{2} \rfloor$ solution exists. Similarly, add $2^n - C_{\lfloor \frac{h}{2} \rfloor}$ clauses. If SAT, query $\lfloor \frac{h}{4} \rfloor$. If UNSAT, query $\lfloor \frac{h}{4} \rfloor + \lfloor \frac{h}{4} \rfloor$. Keep querying the solver until it finds the minimum # of columns. The procedure queries the solver at most $O(\lg n)$ times.

6 [Quine-McCluskey]

60 (0) 000000 D ✓	0,1 00000 - ✓	0,1,2,3 00000 -
61 (1) 000001 ✓	0,2 00000 - ✓	0,2,3,4 00000 -
62 (0) 000010 ✓	1,3 00000 - ✓	1,3,5,7 00000 -
63 (3) 000011 ✓	1,5 00000 - ✓	1,3,5,7,9 00000 -
65 (5) 000101 ✓	1,7 00000 - ✓	1,3,5,7,9,11 00000 -
63 (3) 0000101 ✓	2,3 00000 - ✓	2,3,5,7 00000 -
64 (6) 000110 D ✓	2,6 00000 - ✓	2,3,6,7 00000 -
	3,7 00000 - ✓	2,4,5,7 00000 -
65 (9) 000111 ✓	3,11 00000 - ✓	3,7,11,15 00000 -
61 (1) 001011 ✓	3,15 00000 - ✓	3,11,15,19 00000 -
63 (3) 001101 ✓	5,9 00000 - ✓	7,11,15,19 00000 -
66 (6) 000111 ✓	5,13 00000 - ✓	5,7,13,15 00000 -
	32,35 00000 - ✓	32,37,39 00000 -
63 (11) 101111 ✓	6,9 00000 - ✓	11,15,23,27 00000 -
(67) 111111 0 ✓	9,15 00000 - ✓	15,19,23,27 00000 -
(67) 1110011 D ✓	11,15 001-11 ✓	35,43,51,59 1-0011
(67) 1111111 0 ✓	13,15 0111-1 ✓	35,43,47,49 1-0011
(68) 1110111 D ✓	25,49 10-011 ✓	
	35,51 1-0011 ✓	15,49,31,63 -1111
63 (49) 1011111 ✓	15,49 0-1111 ✓	45,53,47,63 -1111
(59) 1110111 ✓	15,61 0-1111 ✓	43,49,51,63 1-1-11
(51) 0111111 D ✓	43,49 101-11 ✓	43,49,47,63 1-1-11
	43,59 1-0111 ✓	
66 (62) 111111 D ✓	51,59 11-011 ✓	
	58,59 111111 ✓	
	49,63 1-1111 ✓	
	59,63 111-11 ✓	
	31,63 -111111 D ✓	

	C ₁	C ₂	C ₃	C ₄	C ₅	C ₆	C ₇	C ₈	C ₉	C ₁₀	C ₁₁	C ₁₂	C ₁₃	C ₁₄
m ₁ a'b'c'd'e'f'	0	0	0	1	1	1	0	0	0	0	0	0	0	0
prime cube:	m ₂ a'b'c'd'e'f'	0	0	0	1	0	0	1	0	0	0	0	0	0
abcde'f'	C ₁	m ₃ a'b'c'd'e'f'	0	0	0	1	1	1	1	1	0	0	0	0
abc'd'e'f'	C ₂	m ₄ a'b'c'd'e'f'	1	0	0	0	1	0	0	0	1	0	0	0
abc'd'e'	C ₃	m ₅ a'b'c'd'e'f'	0	0	0	0	1	0	1	1	0	0	0	0
a'b'c'd'	C ₄	m ₆ a'b'c'd'e'f'	0	0	0	0	0	0	1	1	0	0	0	0
a'b'c'f	C ₅	m ₇ a'b'c'd'e'f'	0	0	0	0	0	0	0	0	1	0	0	0
b'c'd'f	C ₆	m ₈ a'b'c'd'e'f'	0	0	0	0	0	0	1	0	1	0	1	0
a'b'c'e	C ₇	m ₉ a'b'c'd'e'f'	0	0	0	0	0	1	0	0	0	0	0	0
a'b'c'f	C ₈	m ₁₀ a'b'c'd'e'f'	0	0	0	0	0	0	1	0	0	1	0	0
b'd'ef	C ₉	m ₁₁ a'b'c'd'e'f'	0	0	0	0	0	0	0	1	0	1	0	1
a'b'bf	C ₁₀	m ₁₂ a'b'c'd'e'f'	0	0	0	0	0	0	0	0	0	1	0	1
b'cef	C ₁₁	m ₁₃ a'b'c'd'e'f'	0	0	1	0	0	0	0	0	0	0	1	0
a'd'ef	C ₁₂	m ₁₄ a'b'c'd'e'f'	1	0	0	0	0	0	0	0	0	0	0	0
cdef	C ₁₃	C ₁ essential (m ₁) C ₆ essential (m ₂) C ₁₁ essential (m ₁₃)												
acef	C ₁₄	remove, covered minterms m ₁ , m ₂ , m ₄ , m ₅ , m ₇ , m ₈ , m ₁₀ , m ₁₁												

C ₃	C ₄	C ₅	C ₆	C ₇	C ₈	C ₉	C ₁₀	C ₁₁	C ₁₂	C ₁₃	C ₁₄	C ₁₁ dominates C ₈ , C ₉
m ₂	1	*	1	0	0	*	0	*	C ₁₁ dominates C ₈ , C ₉			
m ₆	0	0	*	0	1	1	0	0	C ₉ equals to C ₄			
M ₁₁	0	*	0	0	0	1	1	0				
M ₁₂	0	0	*	0	0	1	0	1				
M ₃	1	0	*	0	0	0	0	1				
	C ₇	C ₁₁	C ₁₄									
M ₂	1	0	0		C ₇ essential (m ₂)							
M ₆	0	1	0		C ₁₁ essential (m ₆)							
M ₁₁	*	1	1		C ₁₄ essential (M ₁₁)							
M ₁₂	0	1	1									
M ₃	0	0	1									
	M ₁₃	0	0	1								
												⇒ cover on f {abcde'f', b'c'd'f, a'b'df, a'b'c'e, b'cef, acef}