

Computational Boolean Algebra.

①

→ data structures + operators:

WEEK 1

Shannon Expansion Theorem

$$\left. \begin{array}{l} F(x_1, x_2, \dots, \text{set}(x_i=1), \dots, x_n) \\ F(x_1, x_2, \dots, \text{set}(x_i=0), \dots, x_n) \end{array} \right\} \begin{array}{l} \text{Shannon} \\ \text{co-factor} \end{array}$$

if $x_i=1$ \Rightarrow positive co-factor.

if $x_i=0$ \Rightarrow Negative co-factor.

* Shannon expansion:

$$F(x_1, x_2, \dots, x_i, \dots, x_n) = x_i \cdot \overset{\text{pos.}}{F(x_i=1)} + \bar{x}_i \cdot \overset{\text{neg.}}{F(x_i=0)}$$

Just like a multiplexer.

$$F(x, y, z, w) =$$

$$xy F(x=1, y=1) + x\bar{y} F(x=1, y=0) + \bar{x}y F(x=0, y=1) + \bar{x}\bar{y} F(x=0, y=0).$$

* Co-factor properties - Shannon co-factor prop.

$$F_x \oplus \bar{F}_x = (F \oplus \bar{F})x$$

* Boolean derivative: $f_x \oplus f_{\bar{x}}$

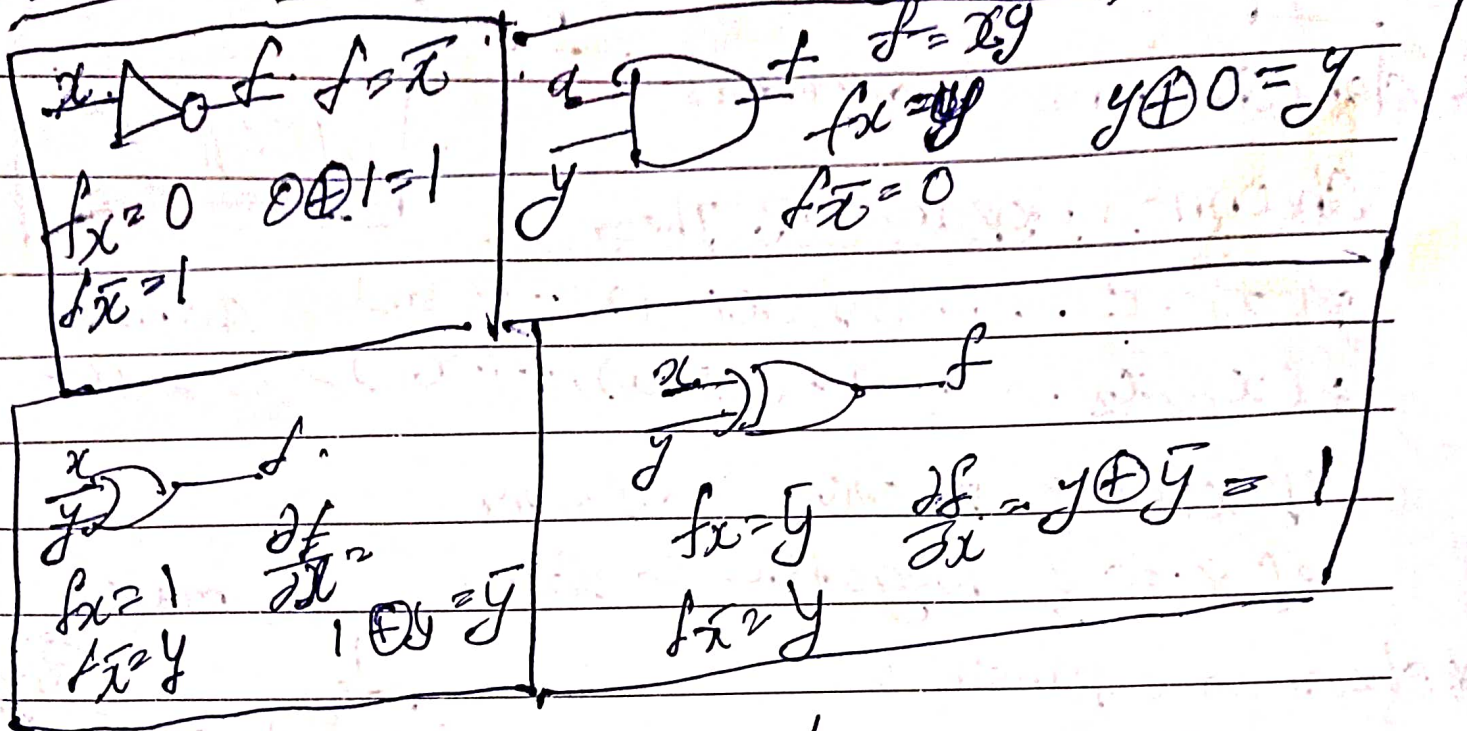
$$\frac{\partial f}{\partial x} = f_x \oplus f_{\bar{x}}$$

Boolean difference.

$$\frac{\partial f}{\partial x} = f_x \oplus f_{\bar{x}}$$

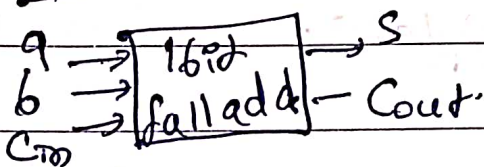
Boolean difference for some gates.

(2)



Boolean difference - example.

Example



$$S = a \oplus b \oplus C_{in}$$

$$C_{out} = ab + (a + b)C_{in}$$

$$\frac{\partial C_{out}}{\partial C_{in}} = ?$$

$$\frac{\partial C_{out}}{\partial C_{in}} = a \oplus b$$

$$C_{out}_{Cin} = ab + (a + b)$$

$$= \underline{a \oplus b} \rightarrow \text{makes sense}$$

$$C_{out}_{Cin} = ab$$

when $a \neq b$ - output changes

Quantification operators.

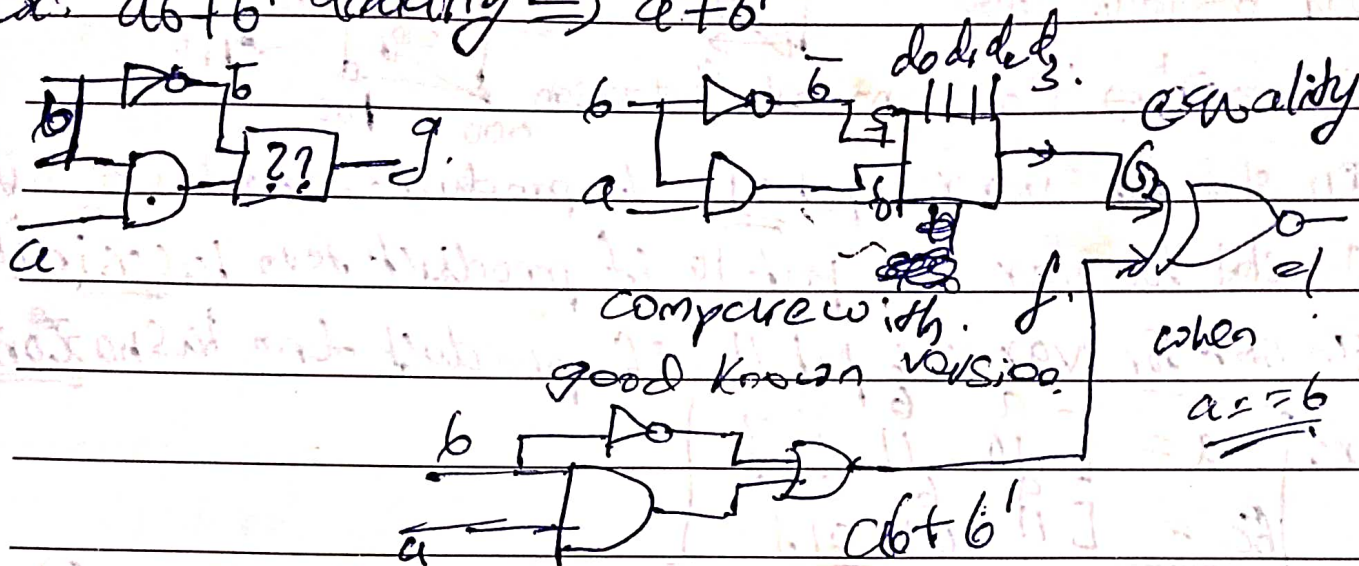
$F(x) \Rightarrow$ Universal Quantification $\Rightarrow \forall x F$ for all x
 $\neg F(x) \Rightarrow$ Existential Quantification $\Rightarrow \exists x F$ there exists x
 $F(x) \vee \neg F(x) \Rightarrow$ for all x there exists an x

Quantification: Application to logic.

(3)

Network Repair.

Ex: $ab + b'$ actually $\Rightarrow a + b'$



$$G = d_0 S_1' S_0' + d_1 S_1' S_0 + d_2 S_1 S_0' + d_3 S_1 S_0$$

$$G = d_0 \bar{a} \bar{b} \bar{b} + d_1 \bar{a} \bar{b} \bar{b} + d_2 a \bar{b} \bar{b} + d_3 a \bar{b} \bar{b}$$

$$G = d_0 \bar{a} \bar{b} + d_1 \bar{b} + d_2 a \bar{b}$$

$$f = ab + b'$$

$$G \oplus f = b f + \bar{b} \bar{f} = (a b + \bar{b}) \oplus (a b + \bar{a} \bar{b}) = a b \oplus \bar{a} \bar{b}$$

$$Z = (d_0 \bar{a} \bar{b} + d_1 \bar{b} + d_2 a \bar{b}) \oplus (a b + b')$$

$$Z_{ab'} = d_1 \quad Z_{\bar{a} \bar{b}} = d_0 \quad Z_{a \bar{b}} = d_2 \quad Z_{a b} = d_3$$

$$\{a b\} (d_0 d_1 d_2 d_3) = 0 \text{ and } \bar{a} \bar{b} = d_0 d_1 d_2 = 1$$

$$d_0 = 0$$

$$d_1 = 1$$

$$d_2 = 1$$

$$d_3 = 1$$

It can be + OR gate, expected

⊕ XOR gate, also repairs.

localize the gate

OPOX

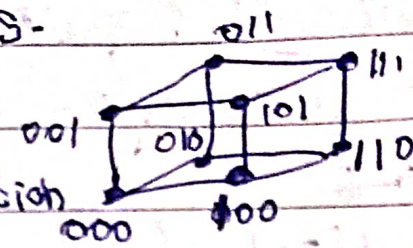
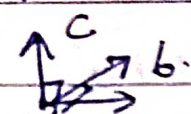
OPOX-CV

Recursive Tautology

(4)

Set of 'or'd product terms.

3 var boolean cube



cube = $2^k \Rightarrow$ Positional cube notation

In slot for var x : put 0 if product term has x in it

In slot for var x : put 1 if product term has \bar{x} in it

In slot for var x : put 1 if product term has no x or \bar{x}

ex:
$$\begin{matrix} a & b & c \\ \bar{a} = & [& 1 & 0 & 1 & 1 &] \\ bc = & [& 1 & 1 & 0 & 1 &] \end{matrix}$$

$f(a,b,c) = a + bc + ab \Rightarrow [0111], [110101], [01011]$

Result: f is a tautology if and only if $f(x=0)$ & $f(x=1)$ are both tautologies.

That is $f(x=0), f(x=1) = 1$

$$x \cdot f(x=1) + x' \cdot f(x=0) = x \cdot 1 + x' \cdot 1 = x + x' = 1$$

URP Implementation (Unique Recursive Procedure)

ex: $f = abd + bc$ $f_a a \Rightarrow 1$ $f_c c \Rightarrow 1$ (Koradyne)

abd $[0101101]$ $[1101101]$ $[0101101]$

bc $[1101101]$ $[1101101]$ $[1101101]$

~~$[1101101]$~~

Unate function

ex: $ab + ac'd + c'de \rightarrow$ unate.

ex: $xzy + x'y + xjz \rightarrow$ not unate.

PANTIN

positive unate & negative unate \rightarrow binate.

(5)

$a + b'c + ac$ UNATE:

$a + b'c + bc$ NOT UNATE

$\begin{matrix} a & b & c \\ [0 & 1 & 1] \\ [1 & 0 & 0] \\ [0 & 1 & 0] \end{matrix}$

$\begin{matrix} a & b & c \\ [0 & 1 & 1] \\ [1 & 0 & 0] \\ [1 & 0 & 1] \end{matrix}$

$\begin{matrix} [1 & 1 & 0] \\ [0 & 1 & 1] \\ [0 & 0 & 0] \end{matrix}$

$\begin{matrix} [1 & 1 & 0] \\ [0 & 1 & 1] \\ [1 & 0 & 1] \end{matrix}$

$\begin{matrix} [0 & 1 & 1] \\ [0 & 0 & 0] \end{matrix}$

$\begin{matrix} [1 & 0 & 1] \\ [1 & 0 & 0] \end{matrix}$

$\begin{matrix} [0 & 0 & 0] \\ [0 & 0 & 0] \end{matrix}$

$\begin{matrix} [1 & 0 & 0] \\ [1 & 0 & 0] \end{matrix}$

checking unate.

Unate cube list for f is tautology if it contains.

all don't care cube $[11 \dots 11] = 1$ tautology.

$f = 1$ ✓

1st rule $\boxed{f = 1}$ 2nd rule

$f = 1$ ✗

$f = 1$
 $[11 \dots 11]$ YES.

$\begin{matrix} 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 \end{matrix}$ NO
is unate but not tautology.

Another rule.

$f = 1$ YES

$\begin{matrix} 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 1 & 0 \end{matrix}$ $x + x' = 1$

Tautology checking - further:

⑥

✓ pick most product terms

✓ pick var with minimum / true var-complement
var

x y z w

01 ~~01~~ 01 01

10 ~~11~~ 01 01

10 ~~11~~ ~~11~~ 10

01 01 ~~11~~ 01

$\frac{2-2}{0}$ 0, 0 $\frac{3-1}{2}$

→ generate 4 cubes / true - imp! $< 2-2=0$.

Algorithm for tautology

tautology(f represented in cubelist) {

#check if we can terminate recursion.

if (f is unit) {

apply unit tautology determining rules
directly.

if (c == 1)

return(1).

else

return(0).

}

else if any other termination rules, else rules work? $\{ \}$
 returns the appropriate if $= 1$ or $= 0$

$\{$
 else $\{$
 # can't tell from this -- find splitting variable
 $x = \text{most - notunate variable in } f.$
 return (tautology(f_x) && tautology($f_{x'}$))
 $\}$
 $\}$

Ex: $f = ab + ac + ab'c' + a'$

	a	b	c
ab	0	1	1
ac	0	1	1
ab'c'	0	1	0
a'	1	0	1

all are
binate

$f_a \quad a \rightarrow 1$

$f_{a'} \quad a \rightarrow 0$

1	1	0	1
1	1	1	0
1	1	1	0

1	1	1
1	1	1
1	1	1

YES.

$f_{a'b} \quad a \rightarrow 0, b \rightarrow 0$
 YES

1	1	0
1	1	0
1	1	0

YES, tautology.