챕터2 정리

Ch2: BOOLEAN ALGEBRA

2.1 Introduction

· Boolean Algebra?

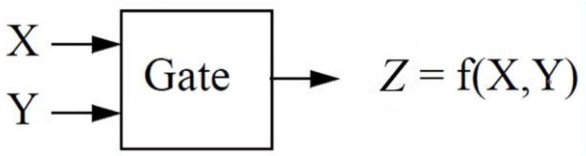
- Boolean algebra: Bool 수학자의 대수학. 컴퓨터 내부에서의 명제를 다루는 수학적 체계

이진 체계 → Switch로 표현할 수 있음 (Switching Algebra라고도 표현함)

변수: X, Y, ... → 오직 두 상태의 값만 가능(0, 1) - True(1), False(0)

· Gate

- Gate: 논리 작용을 하는 단순한 electronic circuit(혹은 system)



· Truth Table

- Truth Table(진리 표): 모든 가능한 입력 조합에 대해 출력이 어떻게 되는지 정리한 표

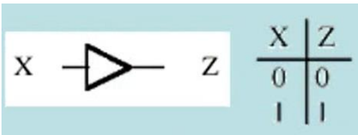
1/0을 True/False로 표현 가능하며 high/low로 표현 가능함(voltage에서)

If 1 is assigned to H and 0 to L → positive logic (보통의 경우)

If 0 is assigned to H and 1 to L → negative logic

· Standard Gates & Symbols

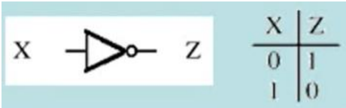
- Buffer



→ 입력의 값이 출력 값에 그대로 전달되는 게이트

입력이 하나인 기본 게이트

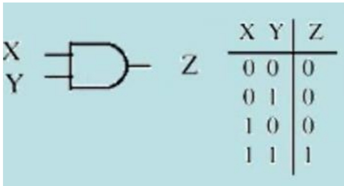
- Not (Invert or Complement)



→ 입출력이 뒤바뀌는 게이트. 그림에서 동그라미를 버블이라 함

입력이 하나인 기본 게이트

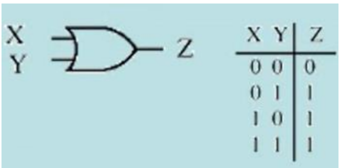
- AND



→ 둘 다 1일때만 1, 그 외에는 0을 출력하는 게이트

입력이 2개 이상이고, 출력이 하나인 게이트

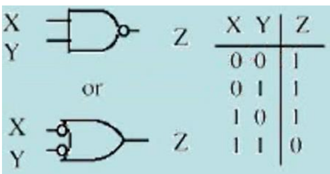
- OR



→ 입력 중 하나라도 1이면 1, 둘 다 0일 땐 0을 출력하는 게이트

입력이 2개 이상이고, 출력이 하나인 게이트

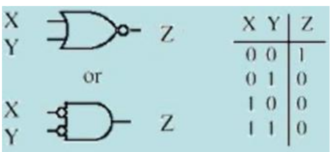
- NAND



→ NOT + AND의 조합. Bubble이 inverting 작용을 함

입력이 2개 이상이고, 출력이 하나인 게이트

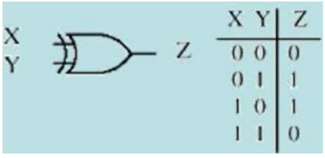
- NOR



→ NOT + OR의 조합. 1이 하나라도 있으면 0, 나머지는 1

입력이 2개 이상이고, 출력이 하나인 게이트

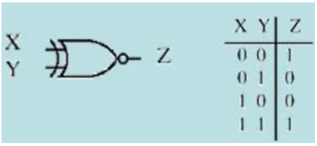
- XOR (exclusive OR)



→ EX-OR라고도 함. 입력이 같으면 0, 다르면 1

입력이 2개 이상이고, 출력이 하나인 게이트

- Equivalence



→ EX-NOR라고도 함. 입력이 같으면 1, 다르면 0

입력이 2개 이상이고, 출력이 하나인 게이트

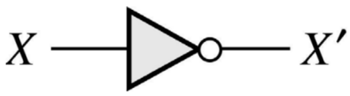
2.2 Basic Operations

· NOT(Inverter)

0' = 1 and 1' = 0

X' = 1 if X = 0 and X' = 0 if X = 1

- Gate Symbol



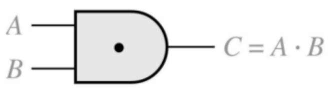
· AND

0 · 0 = 0, 0 · 1 = 0, 1 · 0 = 0, 1 · 1 = 1

- Truth Table

A B	C = A · B
0 0	0
0 1	0
1 0	0
1 1	1

- Gate Symbol



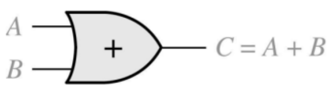
· OR

0 + 0 = 0, 0 + 1 = 1, 1 + 0 = 1, 1 + 1 = 1

- Truth Table

A B	C = A + B
0 0	0
0 1	1
1 0	1
1 1	1

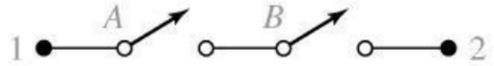
- Gate Symbol



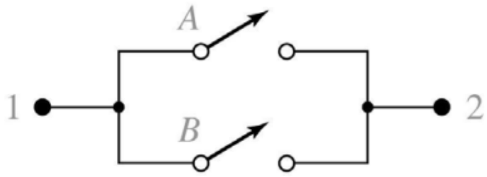
· Apply to Switch

$X = 0 \rightarrow$ switch open
$X = 1 \rightarrow$ switch closed

- AND: T = A · B



- OR: T = A + B

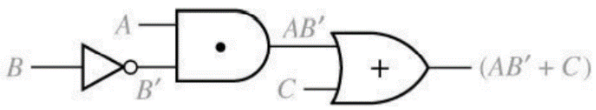


2.3 Boolean Expression and Truth Table

· Logic Expression

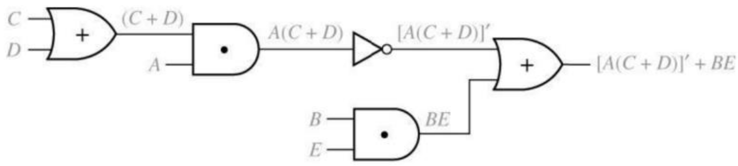
- Logic Expression: AB′ + C

- Circuit of logic gates



- Logic Expression: [A (C + D)]' + BE

- Circuit of logic gates



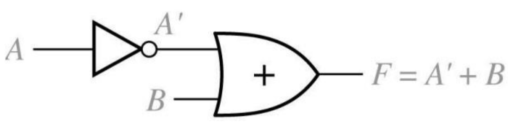
- Logic Evaluation: A = B = C = 1, D = E = 0

  [A (C + D)]' + BE = [1 (1 + 0)]' + 1 · 0 = [1(1)]' + 0 = 0 + 0 = 0

- Literal: 변수 하나 혹은 변수의 complement를 말함

  ab'c + a'b + a'bc'+ b'c' → 10 literals

· 2-Input Circuit and Truth Table



A B		A’	F = A’ + B
0 0		1	1
0 1		1	1
1 0		0	0
1 1		0	1

- Proof using Truth Table: AB′ + C = (A + C)(B′ + C)

n variable needs $\underbrace{2 \times 2 \times 2 \times \ldots}_{\text{n times}} = 2^n$ rows

TABLE 2.1

A B C	B'	AB'	AB' + C	A + C	B' + C	(A + C) (B' + C)
0 0 0	1	0	0	0	1	0
0 0 1	1	0	1	1	1	1
0 1 0	0	0	0	0	0	0
0 1 1	0	0	1	1	1	1
1 0 0	1	1	1	1	1	1
1 0 1	1	1	1	1	1	1
1 1 0	0	0	0	1	0	0
1 1 1	0	0	1	1	1	1

16/30

· 정리

- Boolean Function 세 가지 표현 방법

  1. Logical expression

  2. Truth table

  3. Logic circuit (network)

- Precedence(우선순위) in algebraic expressions: NOT AND OR except for brackets

  괄호 우선, NOT AND OR 순

2.4 Basic Theorems

· Basic Theorems

- Operations with 0, 1

X + 0 = X, X · 1 = X, X + 1 = 1, X · 0 = 0

- Idempotent Laws

X + X = X, X · X = X

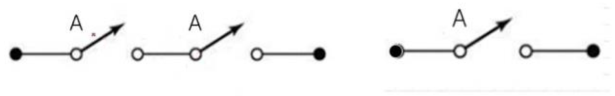
- Involution Laws

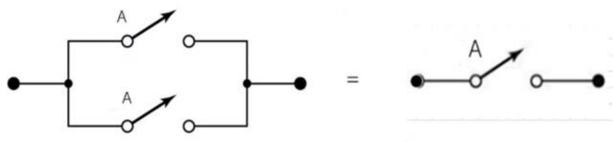
(X’)’ = X

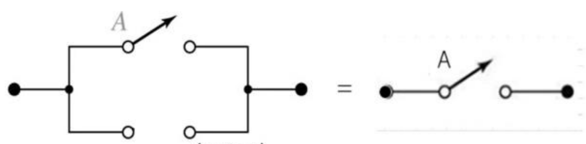
- proof

X = 0, 0 + 0' = 0 + 1 = 1, and if X = 1, 1 + 1' = 1 + 0 = 1

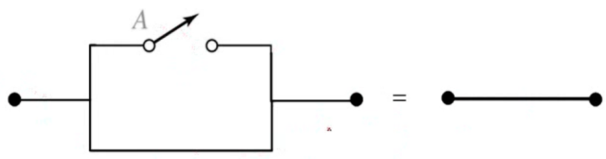
- 회로로 표현



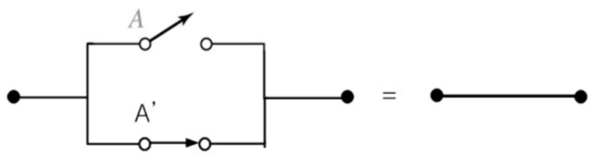




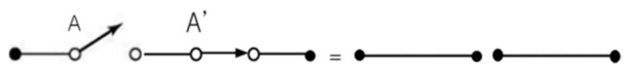
(A+0=A)



(A+1=1)



(A+A’=1)



(A A’=0)

2.5 Commutative, Associative and Distributive Laws

· 여러 법칙

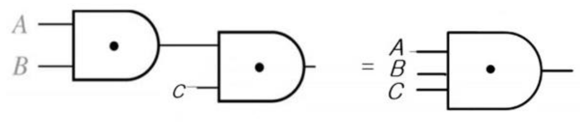
- Commutative Laws(교환 법칙): XY =YX, X + Y = Y + X

- Associative Laws(결합 법칙)

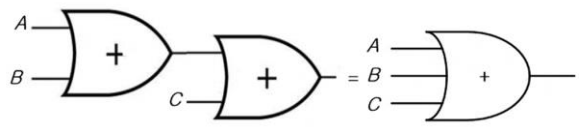
  (XY)Z = X(YZ) = XYZ

  (X + Y) + Z = X + (Y + Z) = X + Y + Z

- Gate로 설명



(AB)C=ABC



(A+B)+C=A+B+C

· 변수가 3개일 때

- AND: XYZ =1 iff X = Y = Z =1

- OR: X + Y + Z = 0 iff X = Y = Z = 0

- Distributive Laws:

  X (Y + Z) = XY + XZ

  X +YZ = (X + Y)(X + Z) → Valid only Boolean algebra, 자주 활용됨!

- Proof

$$
\begin{aligned}
(X+Y)(X+Z) &= X(X+Z)+Y(X+Z) = XX+XZ+YX+YZ \\
&= X+XZ+XY+YZ = X\cdot 1+XZ+XY+YZ \\
&= X(1+Z+Y)+YZ = X\cdot 1+YZ = X+YZ
\end{aligned}
$$

2.6 Simplification Theorem

· Useful Theorems for Simplification

XY + XY′ = X, (X + Y)(X + Y′) = X → 둘은 dual expression

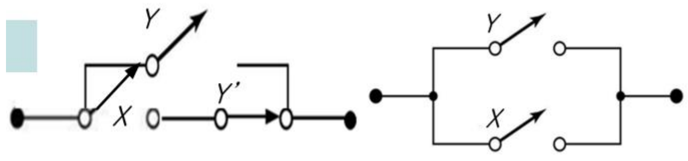
X + XY = X, X(X + Y) = X → 둘은 dual expression

(X + Y′)Y = XY, XY′ + Y = X + Y

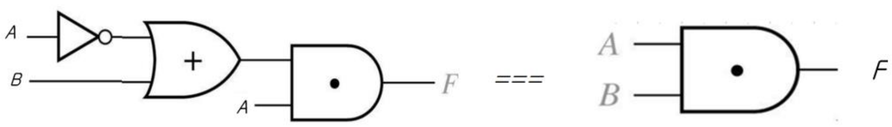
- Proof

$$X + XY = X \cdot 1 + XY = X(1+Y) = X \cdot 1 = X$$
$$X(X+Y) = XX + XY = X + XY = X$$
$$Y + XY' = (Y+X)(Y+Y') = (Y+X)1 = Y + X$$

- Proof with Switch



- Equivalent Gate Circuits: F = A(A' + B) = AB



2.7 Multiplying Out and Factoring

· Logic Expression의 두 가지 standard form: SOP와 POS

· SOP: Sum of Product(곱의 합)

- Sum of product form: AB' + CD'E + AC'E

- 이것도 해당된다

ABC' + DEFG + H

A + B'+ C + D'E

- 이건 아니다

(A + B)CD + EF

- 최대한 간단히 해야 한다

· POS: Product of Sum(합의 곱)

- Product of sum form: (A + B')(C + D' + E)(A + C' + E')

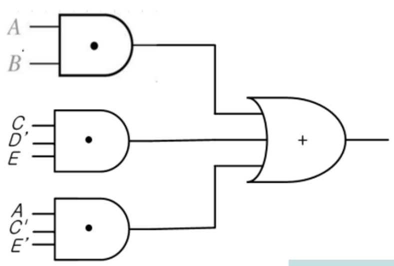
- 이것도 해당된다

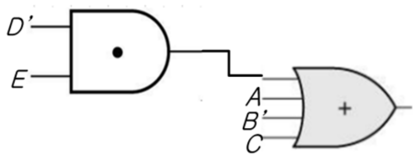
(A + B)(C + D + E)F

AB′C(D′ + E)

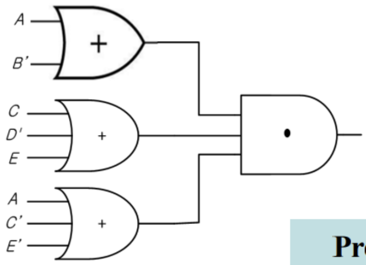
· Circuits for SOP and POS form

- Sum of product form

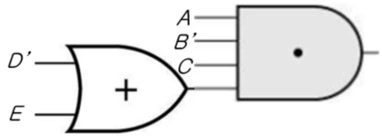




- Product of sum form



Product of sum form:



2.8 DeMorgan's Laws Problems

· DeMorgan's Laws

(X + Y)′ = X′Y′

(XY)′ = X′ + Y′

- Proof

X Y	X'Y'	X+Y	(X+Y)'	X'Y'	XY	(XY)'	X'+Y'
0 0	1 1	0	1	1	0	1	1
0 1	1 0	1	0	0	0	1	1
1 0	0 1	1	0	0	0	1	1
1 1	0 0	1	0	0	1	0	0

- DeMorgan's Laws for n variables

$$(X_1 + X_2 + X_3 + \ldots + X_n)' = X_1' X_2' X_3' \ldots X_n'$$
$$(X_1 X_2 X_3 \ldots X_n)' = X_1' + X_2' + X_3' + \ldots + X_n'$$

- Inverse of F = A′B + AB′

$$F' = (A'B + AB')' = (A'B)'(AB')' = (A + B')(A' + B)$$
$$= AA' + AB + B'A' + BB' = A'B' + AB$$

A B	A' B	A B'	F = A'B+AB'	A' B'	A B	F' = A'B' + AB
0 0	0	0	0	1	0	1
0 1	1	0	1	0	0	0
1 0	0	1	1	0	0	0
1 1	0	0	0	0	1	1

· Dual

- Dual: AND는 OR로, OR는 AND로, 0은 1로, 1은 0로 대체시킨 것

$$(XYZ\ldots)^D = X + Y + Z + \ldots \qquad (X + Y + Z + \ldots)^D = XYZ\ldots$$

→ 드모르간 법칙이랑 다르다!

→ 어떤 표현의 Dual form을 얻기 위해선 드모르간의 법칙을 이용하여 변환한 다음, 각각의 literal에 대해 다시 complementing을 취하면 됨(Inverse 해줌).

예) $(AB' + C)' = (AB')'C' = (A' + B)C', \quad \text{so} \quad (AB' + C)^D = (A + B')C$

어떤 공식이 있으면 그것의 Dual 형태도 항상 성립한다.

어떤 리터럴의 듀얼 폼은 그 자체이다. $A^D = A$