Boolean algebra

- Introduced in 1854 by George Boole. Shown in 1938 to be useful for manipulating Boolean logic functions by C. E. Shannon.
- Postulates and theorems for Boolean algebra are useful to simply logic equations, demonstrate equivalence of expressions, etc.

Postulates

- We have a set of elements **B** and two binary operators $+$ and $\bullet$ that satisfy the following postulates:
  1. Closure with respect to: (a) $+$ and (b) $\bullet$
  2. Identify elements with respect to: (a) $+$, designated by 0 and (b) $\bullet$, designated by 1
  3. Commutative with respect to: (a) $+$ and (b) $\bullet$
  4. Distributive for: (a) $\bullet$ over $+$ and (b) $+$ over $\bullet$
  5. For each element $x \in B$, $\exists !x \in B$ such that (a) $x+!x = 1$ and (b) $x \bullet !x = 0$
  6. There exists at least two elements $x, y \in B$ such that $x \neq y$.
- Axioms are truths and do not require proof.
- *Our definitions of* **AND**, **OR** *and* **NOT** *satisfy the axioms.*

Postulates and theorems

Postulate 2	(a)	$x + 0 = x$	(b)	$x \bullet 1 = x$	identity
Postulate 3	(a)	$x + y = y + x$	(b)	$x \bullet y = y \bullet x$	commutative
Postulate 4	(a)	$x \bullet (y + z) = x \bullet y + x \bullet z$	(b)	$x + (y \bullet z) = (x + y)(z + z)$	distributive
Postulate 5	(a)	$x + !x = 1$	(b)	$x \bullet !x = 0$	
Theorem 1	(a)	$x + x = x$	(b)	$x \bullet x = x$	
Theorem 2	(a)	$x + 1 = 1$	(b)	$x \bullet 0 = 0$	
Theorem 3		$(x')' = x$			involution
Theorem 4	(a)	$x + (y + z) = (x + y) + z$	(b)	$x \bullet (y \bullet z) = (x \bullet y) \bullet z$	associative
Theorem 5	(a)	$(x + z)' = x' \bullet y'$	(b)	$(x \bullet y)' = x' + y'$	DeMorgan
Theorem 6	(a)	$x + x \bullet y = x$	(b)	$x \bullet (x + y) = x$	absorption

- Theorems must be proven (from postulates and/or other theorems). Note that truth tables can be used for proofs.
- Note *duality* if we interchange $+$ with $\bullet$ and 0 with 1.

Proofs

- Prove Theorem 1b ($x \bullet x = x$):

$$\begin{aligned} x &= x \bullet 1 && \text{P2b} \\ &= x \bullet (x + !x) && \text{P5a} \\ &= x \bullet x + x \bullet !x && \text{P4a} \\ &= x \bullet x + 0 && \text{P5b} \\ &= x \bullet x && \text{P2a} \end{aligned}$$

- With truth tables...

x	x	$x \bullet x$
0	0	0
1	1	1

  - Comparing column 1 and 3 shows $x = x \bullet x$.

Simplification

- Simplification means to find a "simpler" expression for a function f.
- Example... Find a simpler expression for $f = ab + cd + a+!(!(cd) + a)$.

$$
\begin{aligned}
f &= ab + cd + a+!(!(cd) + a) \\
f &= ab + a + cd + (cd)!a \\
f &= a(1 + b) + cd(1+!a) \\
f &= a + cd
\end{aligned}
$$

- Note that when simplifying expressions with Boolean algebra, it might be hard to know that you have the absolute smallest expression! Oh well...

Circuit cost

- We previously used Boolean algebra to obtain a simpler expression for a logic function f.
  - We can define the *cost* of a function (or circuit). There can be many different ways to define cost depending on our overall objective.
- For now (unless otherwise stated), let us define the cost of a circuit as follows:
  1. Inverters of input variables are free (no cost);
  2. Every logic gate costs 1 (so more gates are bad);
  3. Every logic gate input costs 1 (so larger gates are bad).
- Cost defined in this way tends to result in circuits that require less overall *area* (less and smaller gates).

Circuit cost

- The cost of $f = a + cd + ab + !(!(cd) + a)$...

$$f = a + \underbrace{\underbrace{cd}_{1+2=3} + \underbrace{ab}_{1+2=3} + \underbrace{!}_{1}\overbrace{(\underbrace{!}_{1}\underbrace{(cd)}_{1+2=3} + a)}^{1+2=3}}_{1+4=5}$$

  So the total cost is **19** (3, 2-input AND, 1, 2-input OR, 1, 4-input OR and 3 non-trivial inverters).

- The cost of $f = cd + a$...

$$f = \underbrace{\overbrace{cd}^{1+2=3} + a}_{1+2=3}$$

  So the total cost is **6** (1, 2-input AND, 1, 2-input OR).

- Since this was our previous simplification example, we can see that Boolean algebra has allowed us to reduce the cost of implementing f (reduced the total number of gates and the size of the gates).

Postive and negative literals

- Let x be a binary variable. Depending on the situation, we might write x (variable **not** complemented) or $!x$ (variable complemented).
  - The uncomplemented version of x is called the "positive literal" of variable x.
  - The complemented version of x is called the "negative literal" of variable x.
- Anyway, I might occasionally use the terminology "literal" and know you know what I mean.