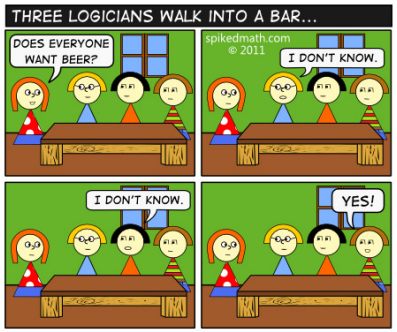
CSE 311 Foundations of Computing I

Spring 2013, Lecture 3

Propositional Logic, Boolean Logic/Boolean Algebra



Administrative

- Course web: http://www.cs.washington.edu/311
  - Homework, Lecture slides, Office Hours ...
- Homework:
  - Due Wednesday at the start of class

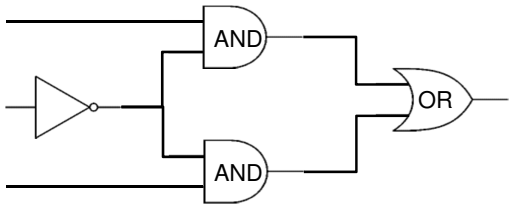
Announcements

- Reading assignments
  - Propositional Logic
    - 1.1 -1.3 7th Edition
    - 1.1 -1.2 6th Edition
  - Boolean Algebra
    - 12.1 – 12.3 7th Edition
    - 11.1 – 11.3 6th Edition

So far

- Propositional/Boolean logic
  - Basic logical connectives
    - If pigs can whistle, then horses can fly
  - Basic circuits
  - Tautologies, equivalences (in progress)

Combinational Logic Circuits

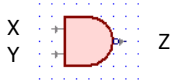


Design a 3 input circuit to compute the majority of 3. Output 1 if at least two inputs are 1, output 0 otherwise

What about a majority of 5 circuit?

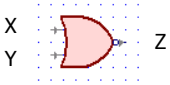
Other gates (more later)

- NAND



X	Y	Z
0	0	1
0	1	1
1	0	1
1	1	0

- NOR

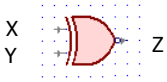


X	Y	Z
0	0	1
0	1	0
1	0	0
1	1	0

- XOR
  $X \oplus Y$

X	Y	Z
0	0	0
0	1	1
1	0	1
1	1	0

- XNOR
  $X \leftrightarrow Y$, $X = Y$



X	Y	Z
0	0	1
0	1	0
1	0	0
1	1	1

Review

Logical equivalence

- Terminology: A compound proposition is a
  - *Tautology* if it is always true
  - *Contradiction* if it is always false
  - *Contingency* if it can be either true or false

$p \vee \neg p$

$p \oplus p$

$(p \rightarrow q) \wedge p$

$(p \wedge q) \vee (p \wedge \neg q) \vee (\neg p \wedge q) \vee (\neg p \wedge \neg q)$

Review

Logical Equivalence

- p and q are *logically equivalent* iff $p \leftrightarrow q$ is a tautology
  - *i.e.* p and q have the same truth table
- The notation $p \equiv q$ denotes p and q are logically equivalent
- Example: $p \equiv \neg\neg p$

p	$\neg p$	$\neg\neg p$	$p \leftrightarrow \neg\neg p$

Review

De Morgan's Laws

- $\neg (p \wedge q) \equiv \neg p \vee \neg q$
- $\neg (p \vee q) \equiv \neg p \wedge \neg q$

- What are the negations of:
  - The Yankees and the Phillies will play in the World Series
  - It will rain today or it will snow on New Year's Day

Review

De Morgan's Laws

Example: $\neg (p \wedge q) \equiv (\neg p \vee \neg q)$

p	q	$\neg p$	$\neg q$	$\neg p \vee \neg q$	$p \wedge q$	$\neg(p \wedge q)$	$\neg(p \wedge q) \leftrightarrow (\neg p \vee \neg q)$
T	T						
T	F						
F	T						
F	F						

Law of Implication

Example: $(p \rightarrow q) \equiv (\neg p \vee q)$

p	q	$p \rightarrow q$	$\neg p$	$\neg p \vee q$	$(p \rightarrow q) \leftrightarrow (\neg p \vee q)$

Understanding connectives

- Reflect basic rules of reasoning and logic
- Allow manipulation of logical formulas
  - Simplification
  - Testing for equivalence
- Applications
  - Query optimization
  - Search optimization and caching
  - Artificial Intelligence
  - Program verification

Properties of logical connectives

- Identity
- Domination
- Idempotent
- Commutative
- Associative
- Distributive
- Absorption
- Negation
- DeMorgan's Laws
- Double Negation

Textbook, 1.3 7th Edition/1.2 6th Edition, Table 6

Equivalences relating to implication

- $p \rightarrow q \equiv \neg p \vee q$
- $p \rightarrow q \equiv \neg q \rightarrow \neg p$
- $p \vee q \equiv \neg p \rightarrow q$
- $p \wedge q \equiv \neg (p \rightarrow \neg q)$
- $p \leftrightarrow q \equiv (p \rightarrow q) \wedge (q \rightarrow p)$
- $p \leftrightarrow q \equiv \neg p \leftrightarrow \neg q$
- $p \leftrightarrow q \equiv (p \wedge q) \vee (\neg p \wedge \neg q)$
- $\neg (p \leftrightarrow q) \equiv p \leftrightarrow \neg q$

Logical Proofs

- To show P is equivalent to Q
  - Apply a series of logical equivalences to subexpressions to convert P to Q
- To show P is a tautology
  - Apply a series of logical equivalences to subexpressions to convert P to **T**

Show $(p \wedge q) \rightarrow (p \vee q)$ is a tautology

Show $(p \rightarrow q) \rightarrow r$ and $r \rightarrow (q \rightarrow p)$ are not equivalent

Boolean Logic

Applications of Propositional Logic for Circuits

Boolean logic

- Combinational logic
  - $output_t = F(input_t)$
- Sequential logic
  - $output_t = F(output_{t-1}, input_t)$
    - output dependent on history
    - concept of a time step (clock)
- An algebraic structure consists of
  - a set of elements B = {0, 1}
  - binary operations { + , • } (OR, AND)
  - and a unary operation { ' } (NOT)

A quick combinational logic example

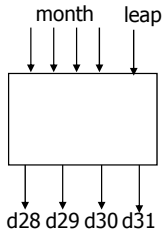
- Calendar subsystem: number of days in a month (to control watch display)
  - used in controlling the display of a wrist-watch LCD screen
  - inputs: month, leap year flag
  - outputs: number of days

Implementation in software

```
integer number_of_days ( month, leap_year_flag) {
   switch (month) {
      case 1: return (31);
      case 2: if (leap_year_flag == 1) then
       return (29) else return (28);
      case 3: return (31);
      ...
      case 12: return (31);
      default: return (0);
   }
}
```


Implementation as a combinational digital system

- Encoding:
  - how many bits for each input/output?
  - binary number for month
  - four wires for 28, 29, 30, and 31



month	leap	d28	d29	d30	d31
0000	–	–	–	–	–
0001	–	0	0	0	1
0010	0	1	0	0	0
0010	1	0	1	0	0
0011	–	0	0	0	1
0100	–	0	0	1	0
0101	–	0	0	0	1
0110	–	0	0	1	0
0111	–	0	0	0	1
1000	–	0	0	0	1
1001	–	0	0	1	0
1010	–	0	0	0	1
1011	–	0	0	1	0
1100	–	0	0	0	1
1101	–	–	–	–	–
1110	–	–	–	–	–
1111	–	–	–	–	–

Combinational example (cont.)

- Truth-table to logic to switches to gates
  - d28 = “1 when month=0010 and leap=0”
  - d28 = m8'•m4'•m2•m1'•leap'

  - d31 = “1 when month=0001 or month=0011 or ... month=1100”
  - d31 = (m8'•m4'•m2'•m1) + (m8'•m4'•m2•m1) + ... (m8•m4•m2'•m1')
  - d31 = can we simplify more?

month	leap	d28	d29	d30	d31
0000	–	–	–	–	–
0001	–	0	0	0	1
0010	0	1	0	0	0
0010	1	0	1	0	0
0011	–	0	0	0	1
0100	–	0	0	1	0
...					
1100	–	0	0	0	1
1101	–	–	–	–	–
111–	–	–	–	–	–

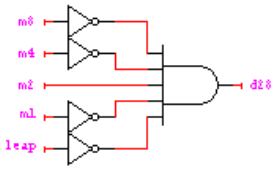
Combinational example (cont.)

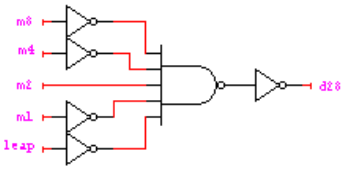
d28 = m8'•m4'•m2•m1'•leap'

d29 = m8'•m4'•m2•m1'•leap

d30 = (m8'•m4•m2'•m1') + (m8'•m4•m2•m1') + (m8•m4'•m2'•m1) + (m8•m4'•m2•m1)
= (m8'•m4•m1') + (m8•m4'•m1)

d31 = (m8'•m4'•m2'•m1) + (m8'•m4'•m2•m1) + (m8'•m4•m2'•m1) + (m8'•m4•m2•m1) + (m8•m4'•m2'•m1') + (m8•m4'•m2•m1') + (m8•m4•m2'•m1')



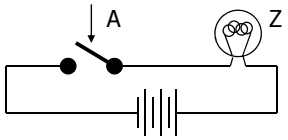


Combinational logic

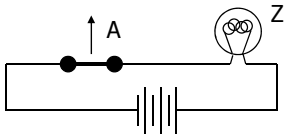
- Switches
- Basic logic and truth tables
- Logic functions
- Boolean algebra
- Proofs by re-writing and by perfect induction

Switches: basic element of physical implementations

- Implementing a simple circuit (arrow shows action if wire changes to "1"):



close switch (if A is "1" or asserted) and turn on light bulb (Z)



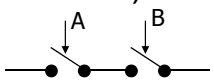
open switch (if A is "0" or unasserted) and turn off light bulb (Z)

Z ≡ A

Switches (cont.)

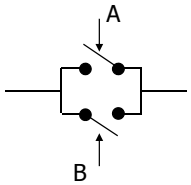
- Compose switches into more complex ones (Boolean functions):

AND



Z ≡ A <u>and</u> B

OR

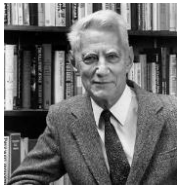


Z ≡ A <u>or</u> B

Transistor networks

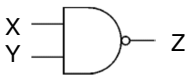
- Modern digital systems are designed in CMOS technology
  - MOS stands for Metal-Oxide on Semiconductor
  - C is for complementary because there are both normally-open and normally-closed switches
- MOS transistors act as voltage-controlled switches
  - similar, though easier to work with than relays.

Multi-input logic gates

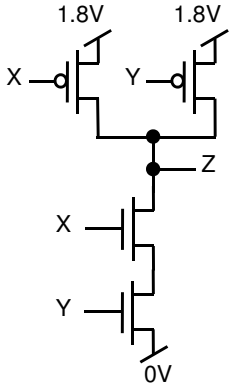


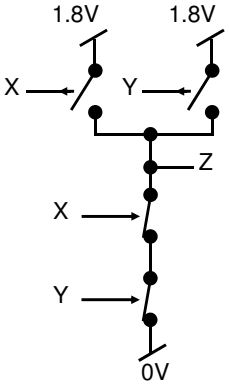
Claude Shannon – 1938

- CMOS logic gates are inverting
  - Easy to implement NAND, NOR, NOT while AND, OR, and Buffer are harder



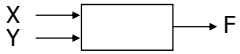
X	Y	Z
0	0	1
0	1	1
1	0	1
1	1	0





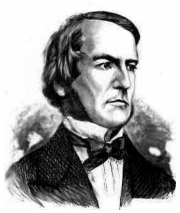
Possible logic functions of two variables

- There are 16 possible functions of 2 input variables:
  - in general, there are 2**(2**n) functions of n inputs



X	Y	16 possible functions (F_0–F_{15})															
0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
0	1	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
1	0	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
1	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
		0	X and Y	X	Y	X xor Y		X or Y		X nor Y not (X or Y)		X = Y	not Y	not X		X nand Y not (X and Y)	1

Boolean algebra



George Boole – 1854

- An algebraic structure consisting of
  - a set of elements B
  - binary operations { + , • }
  - and a unary operation { ' }
  - such that the following axioms hold:

1. the set B contains at least two elements: a, b		
2. closure:	a + b is in B	a • b is in B
3. commutativity:	a + b = b + a	a • b = b • a
4. associativity:	a + (b + c) = (a + b) + c	a • (b • c) = (a • b) • c
5. identity:	a + 0 = a	a • 1 = a
6. distributivity:	a + (b • c) = (a + b) • (a + c)	a • (b + c) = (a • b) + (a • c)
7. complementarity:	a + a' = 1	a • a' = 0

Logic functions and Boolean algebra

Any logic function that can be expressed as a truth table can be written as an expression in Boolean algebra using the operators: ', +, and •

X, Y are Boolean algebra variables

X	Y	X • Y
0	0	0
0	1	0
1	0	0
1	1	1

X	Y	X'	X' • Y
0	0	1	0
0	1	1	1
1	0	0	0
1	1	0	0

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

(X • Y) + (X' • Y') ➔ X = Y

Boolean expression that is true when the variables X and Y have the same value and false, otherwise

Axioms and theorems of Boolean algebra

identity		
	1. $X + 0 = X$	1D. $X \bullet 1 = X$
null		
	2. $X + 1 = 1$	2D. $X \bullet 0 = 0$
idempotency:		
	3. $X + X = X$	3D. $X \bullet X = X$
involution:		
	4. $(X')' = X$	
complementarity:		
	5. $X + X' = 1$	5D. $X \bullet X' = 0$
commutatively:		
	6. $X + Y = Y + X$	6D. $X \bullet Y = Y \bullet X$
associativity:		
	7. $(X + Y) + Z = X + (Y + Z)$	7D. $(X \bullet Y) \bullet Z = X \bullet (Y \bullet Z)$
distributivity:		
	8. $X \bullet (Y + Z) = (X \bullet Y) + (X \bullet Z)$	8D. $X + (Y \bullet Z) = (X + Y) \bullet (X + Z)$

Axioms and theorems of Boolean algebra (cont.)

uniting:		
	9. $X \bullet Y + X \bullet Y' = X$	9D. $(X + Y) \bullet (X + Y') = X$
absorption:		
	10. $X + X \bullet Y = X$	10D. $X \bullet (X + Y) = X$
	11. $(X + Y') \bullet Y = X \bullet Y$	11D. $(X \bullet Y') + Y = X + Y$
factoring:		
	12. $(X + Y) \bullet (X' + Z) = X \bullet Z + X' \bullet Y$	12D. $X \bullet Y + X' \bullet Z = (X + Z) \bullet (X' + Y)$
consensus:		
	13. $(X \bullet Y) + (Y \bullet Z) + (X' \bullet Z) = X \bullet Y + X' \bullet Z$	13D. $(X + Y) \bullet (Y + Z) \bullet (X' + Z) = (X + Y) \bullet (X' + Z)$
de Morgan's:		
	14. $(X + Y + \ldots)' = X' \bullet Y' \bullet \ldots$	14D. $(X \bullet Y \bullet \ldots)' = X' + Y' + \ldots$

Proving theorems (rewriting)

- Using the laws of Boolean algebra:
  - e.g., prove the theorem: $X \bullet Y + X \bullet Y' = X$

distributivity (8)	$X \bullet Y + X \bullet Y' = X \bullet (Y + Y')$
complementarity (5)	$= X \bullet (1)$
identity (1D)	$= X$

e.g., prove the theorem: $X + X \bullet Y = X$

identity (1D)	$X + X \bullet Y = X \bullet 1 + X \bullet Y$
distributivity (8)	$= X \bullet (1 + Y)$
identity (2)	$= X \bullet (1)$
identity (1D)	$= X$

Proving theorems (truth table)

- Using complete truth table:
  - e.g., de Morgan's:

$(X + Y)' = X' \bullet Y'$

NOR is equivalent to AND with inputs complemented

X	Y	X′	Y′	(X + Y)′	X′ • Y′
0	0	1	1		
0	1	1	0		
1	0	0	1		
1	1	0	0		

$(X \bullet Y)' = X' + Y'$

NAND is equivalent to OR with inputs complemented

X	Y	X′	Y′	(X • Y)′	X′ + Y′
0	0	1	1		
0	1	1	0		
1	0	0	1		
1	1	0	0		