Adam Blank Spring 2017

CSE 311

Foundations of Computing I

Digital Circuits

Computing With Logic

- **T corresponds to 1** or "high" voltage
- **F corresponds to 0** or "low" voltage

Gates

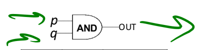
- Take inputs and produce outputs (functions)
- Several kinds of gates
- Correspond to propositional connectives (most of them)

And Gate

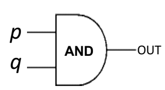
AND Connective vs. **AND Gate**

$p \wedge q$

p	q	$p \wedge q$
T	T	T
T	F	F
F	T	F
F	F	F



p	q	OUT
1	1	1
1	0	0
0	1	0
0	0	0



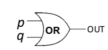
"block looks like D of AND"

Or Gate

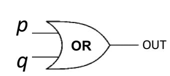
OR Connective vs. **OR Gate**

$p \vee q$

p	q	$p \vee q$
T	T	T
T	F	T
F	T	T
F	F	F



p	q	OUT
1	1	1
1	0	1
0	1	1
0	0	0



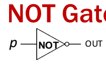
"arrowhead block looks like V"

Not Gates

NOT Connective vs. **NOT Gate**

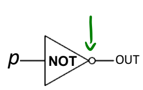
$\neg p$

p	$\neg p$
T	F
F	T



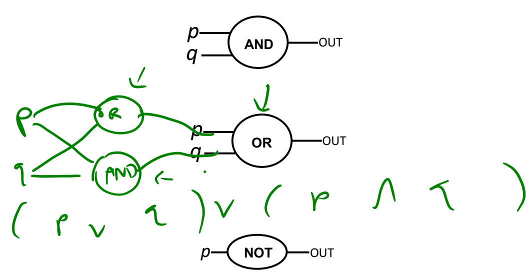
Also called *inverter*

p	OUT
1	0
0	1

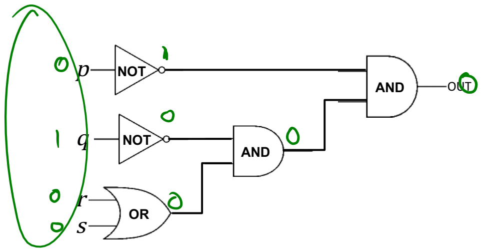


Blobs are Okay!

You may write gates using blobs instead of shapes!

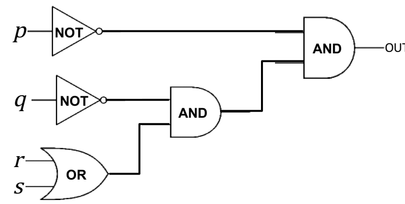


Combinational Logic Circuits



Values get sent along wires connecting gates

Combinational Logic Circuits

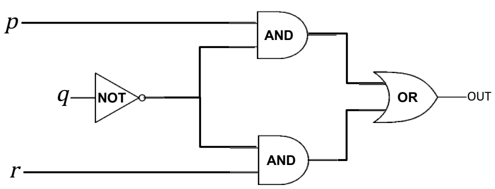


Values get sent along wires connecting gates

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

F T F F → F

Combinational Logic Circuits



Wires can send one value to multiple gates!

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

Administrivia

- Extra Credit: Included post-grades-calculation
- Tokens: Redos for **WRITTEN** questions
- Lying TAs approach!
- Come to my office hours ☹

Some Familiar Properties of Arithmetic

What are there logical versions of these rules?

- $x + y = y + x$ (Commutativity)
  $x \vee y \equiv y \vee x$
  $x \wedge y \equiv y \wedge x$
  $\wedge \quad \vee \quad ||$
- $x \cdot (y + z) = x \cdot y + x \cdot z$ (Distributivity)
  $x + (y \cdot z) \overset{?}{\equiv} (x + y)(x + z)$
- $(x + y) + z = x + (y + z)$ (Associativity)

Some Familiar Properties of Arithmetic

What are there logical versions of these rules?

- $x + y = y + x$ (Commutativity)
  - $p \vee q \equiv q \vee p$
  - $p \wedge q \equiv q \wedge p$
- $x \cdot (y + z) = x \cdot y + x \cdot z$ (Distributivity)
  - $p \wedge (q \vee r) \equiv (p \wedge q) \vee (p \wedge r)$
  - $p \vee (q \wedge r) \equiv (p \vee q) \wedge (p \vee r)$
- $(x + y) + z = x + (y + z)$ (Associativity)
  - $(p \vee q) \vee r \equiv p \vee (q \vee r)$
  - $(p \wedge q) \wedge r \equiv p \wedge (q \wedge r)$

Properties of Logical Connectives

We will always give you this list!

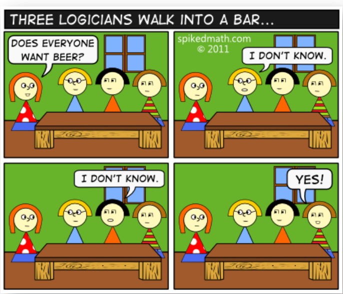
- Identity
  - $p \wedge \mathrm{T} \equiv p$
  - $p \vee \mathrm{F} \equiv p$
- Domination
  - $p \vee \mathrm{T} \equiv \mathrm{T}$
  - $p \wedge \mathrm{F} \equiv \mathrm{F}$
- Idempotent
  - $p \vee p \equiv p$
  - $p \wedge p \equiv p$
- Commutative
  - $p \vee q \equiv q \vee p$
  - $p \wedge q \equiv q \wedge p$
- Associative
  - $(p \vee q) \vee r \equiv p \vee (q \vee r)$
  - $(p \wedge q) \wedge r \equiv p \wedge (q \wedge r)$
- Distributive
  - $p \wedge (q \vee r) \equiv (p \wedge q) \vee (p \wedge r)$
  - $p \vee (q \wedge r) \equiv (p \vee q) \wedge (p \vee r)$
- Absorption
  - $p \vee (p \wedge q) \equiv p$
  - $p \wedge (p \vee q) \equiv p$
- Negation
  - $p \vee \neg p \equiv \mathrm{T}$
  - $p \wedge \neg p \equiv \mathrm{F}$

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

CSE 311: Foundations of Computing

Lecture 3: More Equivalence & Boolean Algebra



Computing Equivalence

Given two propositions, can we write an algorithm to determine if they are equivalent?

Yes! Generate the truth tables for both propositions and check if they are the same for every entry.

What is the runtime of our algorithm?

Every atomic proposition has two possibilities (T, F). If there are n atomic propositions, there are 2^n rows in the truth table.

Logical Proofs

To show A is equivalent to B:

Apply a series of logical equivalences to sub-expressions to convert A to B

idemp: $p \equiv p \vee p$

Example:

Let A be "$p \vee (p \vee p)$", and B be "p".

Our general proof looks like:

$p \vee (p \vee p) \equiv ($ $p \vee p$ $)$ (by idemp.)

$\equiv p$ (By idmp.)

Logical Proofs

To show A is equivalent to B:

Apply a series of logical equivalences to sub-expressions to convert A to B

Example:

Let A be "$p \vee (p \vee p)$", and B be "p".

Our general proof looks like:

$p \vee (p \vee p) \equiv ($ $p \vee p$ $)$ By Idempotency

$\equiv p$ By Idempotency

Logical Proofs

To show A is a Tautology:

Apply a series of logical equivalences to sub-expressions to convert P to **T**.

Example:

Let A be "$\neg p \vee (p \vee p)$"

Our general proof looks like:

$\neg p \vee (p \vee p) \equiv (\ \neg p \vee p\ )$ By idemp.

$\equiv$ **T** By neg

Logical Proofs

To show A is a Tautology:

Apply a series of logical equivalences to sub-expressions to convert P to **T**.

Example:

Let A be "$\neg p \vee (p \vee p)$".

Our general proof looks like:

$\neg p \vee (p \vee p) \equiv (\ \neg p \vee p\ )$ By Idempotency

$\equiv$ **T** By Negation

Prove this is a Tautology: Option 1

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

Make a Truth Table and show:

$(p \wedge q) \rightarrow (p \vee q) \equiv$ **T**

p	q	$p \wedge q$	$p \vee q$	$(p \wedge q) \rightarrow (p \vee q)$
T	T	T	T	T
T	F	F	T	T
F	T	F	T	T
F	F	F	F	T

Prove this is a Tautology: Option 2

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

Use a series of equivalences like so:

$(p \wedge q) \rightarrow (p \vee q) \equiv \neg(p \wedge q) \vee (p \vee q)$ By Law of Implication

$\equiv (\neg p \vee \neg q) \vee (p \vee q)$ By DeMorgan's Laws

$\equiv \neg p \vee (\neg q \vee (p \vee q))$ By Associativity

$\equiv \neg p \vee (\neg q \vee (q \vee p))$ By Commutativity

$\equiv \neg p \vee ((\neg q \vee q) \vee p)$ By Associativity

$\equiv \neg p \vee ((q \vee \neg q) \vee p)$ By Commutativity

$\equiv \neg p \vee (\mathbf{T} \vee p)$ By Negation

$\equiv \neg p \vee (p \vee \mathbf{T})$ By Commutativity

$\equiv \neg p \vee \mathbf{T}$ By Domination

$\equiv \mathbf{T}$ By Domination

Prove these propositions are equivalent

Prove: $p \wedge (p \rightarrow q) \equiv p \wedge q$

$p \wedge (p \rightarrow q) \equiv p \wedge (\neg p \vee q)$ By Law of Implication

$\equiv (p \wedge \neg p) \vee (p \wedge q)$ By Distributivity

$\equiv \mathbf{F} \vee (p \wedge q)$ By Negation

$\equiv (p \wedge q) \vee \mathbf{F}$ By Commutativity

$\equiv p \wedge q$ By Identity

Prove these are not equivalent

F

T → F

(F → T) → F

$(p \rightarrow q) \rightarrow r$

T

F

F → (T → F)

$p \rightarrow (q \rightarrow r)$

Consider: p is F, q is T, and r is F...

Prove these are not equivalent

$(p \rightarrow q) \rightarrow r$ $\quad\quad$ $p \rightarrow (q \rightarrow r)$

Consider: p is F, q is F, and r is F…

$(F \rightarrow F) \rightarrow F \equiv T \rightarrow F$
$\equiv F$

$F \rightarrow (F \rightarrow F) \equiv F \rightarrow F$
$\equiv T$

Boolean Logic

Combinational Logic
– output = F(input)

Sequential Logic
– $output_t = F(output_{t-1}, input_t)$
  • output dependent on history
  • concept of a time step (clock, t)

Boolean Algebra consists of…
– a set of elements B = {0, 1}
– binary operations { + , • } (OR, AND)
– and a unary operation { ' } (NOT)

$p \vee q \rightarrow p + q$

$p \wedge q \rightarrow p \cdot q$

$\neg p \rightarrow p'$

Combinational Logic

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

A Combinational Logic Example

Sessions of Class:

We would like to compute the number of lectures or quiz sections remaining *at the start* of a given day of the week.

– **Inputs:** Day of the Week, Lecture/Section flag
– **Output:** Number of sessions left

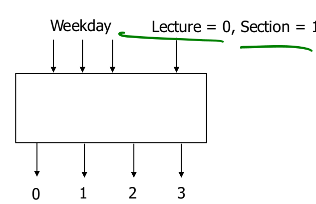
Examples: Input: (Wednesday, Lecture) Output: **2**
Input: (Monday, Section) Output: **1**

Implementation in Software

```
public int classesLeftInMorning(weekday, lecture_flag) {
    switch (weekday) {
        case SUNDAY:
        case MONDAY:
            return lecture_flag ? 3 : 1;
        case TUESDAY:
        case WEDNESDAY:
            return lecture_flag ? 2 : 1;
        case THURSDAY:
            return lecture_flag ? 1 : 1;
        case FRIDAY:
            return lecture_flag ? 1 : 0;
        case SATURDAY:
            return lecture_flag ? 0 : 0;
    }
}
```

Implementation with Combinational Logic

Encoding:
– How many bits for each input/output?
– Binary number for weekday
– One bit for each possible output

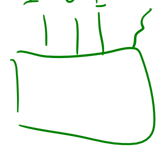


Defining Our Inputs!

Weekday Input:

- Binary number for weekday
- Sunday = 0, Monday = 1, …
- We care about these in binary:

1 0 1 0



Weekday	Number	Binary
Sunday	0	$(000)_2$
Monday	1	$(001)_2$
Tuesday	2	$(010)_2$
Wednesday	3	$(011)_2$
Thursday	4	$(100)_2$
Friday	5	$(101)_2$
Saturday	6	$(110)_2$

Converting to a Truth Table!

```
case SUNDAY or MONDAY:
   return lecture_flag ? 3 : 1;
case TUESDAY or WEDNESDAY:
   return lecture_flag ? 2 : 1;
case THURSDAY:
   return lecture_flag ? 1 : 1;
case FRIDAY:
   return lecture_flag ? 1 : 0;
case SATURDAY:
   return lecture_flag ? 0 : 0;
```

Weekday		Lecture=0	c_0	c_1	c_2	c_3
SUN	000	0	0	0	0	1
SUN	000	1	0	1	0	0
MON	001	0				
MON	001	1				
TUE	010	0				
TUE	010	1				
WED	011	0				
WED	011	1				
THU	100	-				
FRI	101	0				
FRI	101	1				
SAT	110	-				
-	111	-				