CSE 311

Foundations of Computing I

Boy It's Hot In Here!

- Yep. The room doesn't have enough seats.
- Yep. The room is boiling hot.
- I tried to get a new room. There wasn't one ☹

Collaboration Policy

- **There are two types of HW questions:**
  - **Written:**

    You may work with other students, but you must write your work up individually.

  - **Online:**

    You **may not discuss these with anyone other than course staff!** You will have multiple attempts though!

$p \rightarrow q$

(1) "I have collected all 151 Pokémon if I am a Pokémon master"

(2) "I have collected all 151 Pokémon only if I am a Pokémon master"

These sentences are opposites of each other:

(1) "Pokémon masters have all 151 Pokémon"

(2) "People who have 151 Pokémon are Pokémon masters"

So, the implications are:

(1) If I am a Pokémon master, then I have collected all 151 Pokémon.

(2) If I have collected all 151 Pokémon, then I am a Pokémon master.

$p \rightarrow q$

p	q	$p \rightarrow q$
T	T	T
T	F	F
F	T	T
F	F	T

Implication:

- *p* implies *q*
- whenever *p* is true *q* must be true
- if *p* then *q*
- *q* if *p*
- *p* is sufficient for *q*
- *p* only if *q*

A Note On Formality

```
Console.WriteLine("Hello World!");
```

vs.

```
System.out.println("Hello World!");
```

It's clear what both of these mean, but the Java compiler will only accept one and the C# compiler will accept the other. **Neither one of them is WRONG, it's just a context change.**

Why are we talking about this? We're dealing with a formal language here:

$$p \rightarrow q \text{ vs. } p \Rightarrow q$$

Our formal language uses the former.

You may not use the latter.

Biconditional: $p \leftrightarrow q$

- *p* iff *q*
- *p* is equivalent to *q*
- *p* implies *q* and *q* implies *p*

p	q	$p \leftrightarrow q$
T	**T**	**T**
T	**F**	**F**
F	**T**	**F**
F	**F**	**T**

Converse, Contrapositive, Inverse

Implication:

$p \rightarrow q$

Converse:

$q \rightarrow p$

Contrapositive:

$\neg q \rightarrow \neg p$

Inverse:

$\neg p \rightarrow \neg q$

How do these relate to each other?

Consider

p: x is divisible by 2

q: x is divisible by 4

	Divisible By 2	Not Divisible By 2
Divisible By 4	4	Nothing Here!
Not Divisible By 4	2	3

Converse, Contrapositive, Inverse

Implication:

$p \rightarrow q$

Converse:

$q \rightarrow p$

Contrapositive:

$\neg q \rightarrow \neg p$

Inverse:

$\neg p \rightarrow \neg q$

Consider

***p*: *x* is divisible by 2**

***q*: *x* is divisible by 4**

$p \rightarrow q$	F
$q \rightarrow p$	T
$\neg q \rightarrow \neg p$	F
$\neg p \rightarrow \neg q$	T

	Divisible By 2	Not Divisible By 2
Divisible By 4	4	Nothing Here!
Not Divisible By 4	2	3

An implication and it's contrapositive have the same truth value!

Back to Roger's Sentence

"Roger is an orange elephant who has toenails if he has tusks, and has toenails, tusks, or both."

↓

RElephant ∧ (RToenails **if** RTusks) ∧ (RToenails ∨ RTusks ∨ (RToenails ∧ RTusks))

Define shorthand …

p : RElephant
q : RTusks
r : RToenails

↓

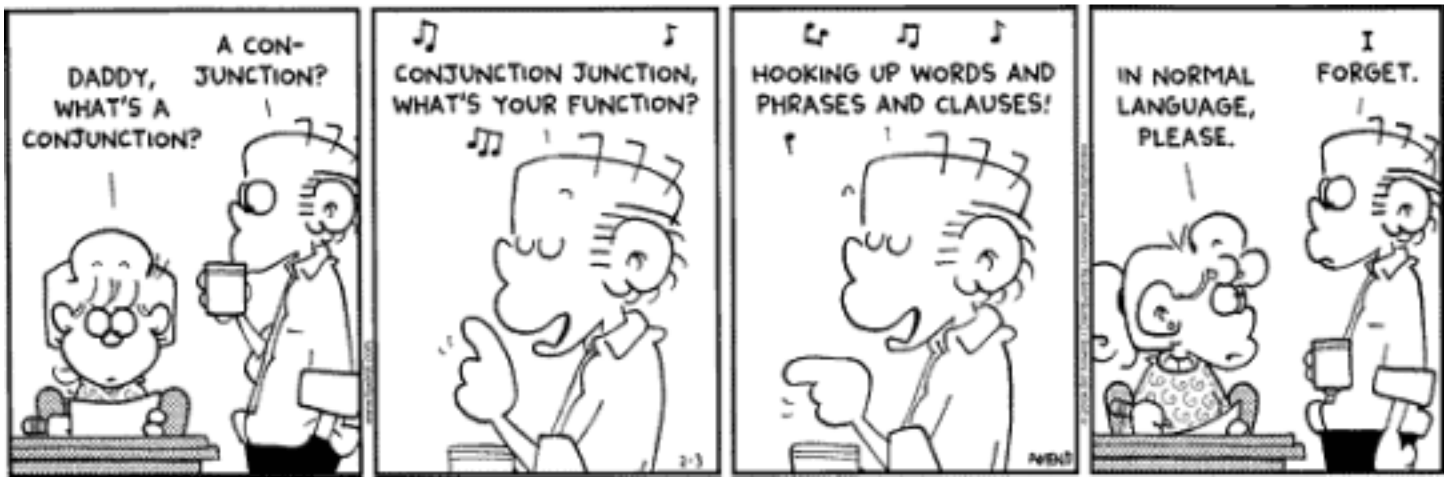
$$(p \land (q \rightarrow r) \land (r \lor q \lor (r \land q))$$

Roger's Sentence with a Truth Table

p	q	r	$q \longrightarrow r$	$p \wedge (q \longrightarrow r)$	$r \vee q$	$r \wedge q$	$(r \vee q) \vee (r \wedge q)$	$p \wedge (q \longrightarrow r) \wedge (r \vee q) \vee (r \wedge q)$
T	T	T	T	T	T	T	T	T
T	T	F	F	F	T	F	T	F
T	F	T	T	T	T	F	T	T
T	F	F	T	T	F	F	F	F
F	T	T	T	F	T	T	T	F
F	T	F	F	F	T	F	T	F
F	F	T	T	F	T	F	T	F
F	F	F	T	F	F	F	F	F

CSE 311: Foundations of Computing

Lecture 2: Logical Equivalence & Digital Circuits



Tautologies!

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

$p \vee \neg p$

This is a tautology. It's called the "law of the excluded middle.
If p is true, then $p \vee \neg p$ is true. If p is false, then $p \vee \neg p$ is true.

$p \oplus p$

This is a contradiction. It's always false no matter what truth value p takes on.

$(p \rightarrow q) \wedge p$

This is a contingency. When p=T, q=T, $(T \rightarrow T) \wedge T$ is true.
When p=T, q=F, $(T \rightarrow F) \wedge T$ is false.

Logical Equivalence

A = B means A and B are identical "strings":

- $p \wedge q = p \wedge q$
  These are equal, because they are character-for-character identical.

- $p \wedge q \neq q \wedge p$
  These are NOT equal, because they are different sequences of characters. They "mean" the same thing though.

A ≡ B means A and B have identical truth values:

- $p \wedge q \equiv p \wedge q$
  Two formulas that are **equal** also are equivalent.

- $p \wedge q \equiv q \wedge p$
  These two formulas have the same truth table!

- $p \wedge q \not\equiv q \vee p$
  When p=T and q=F: $T \wedge F$ is false, but $F \vee T$ is true!

$A \leftrightarrow B$ vs. $A \equiv B$

$A \equiv B$ is an **assertion over all possible truth values** that A and B always have the same truth values.

$A \leftrightarrow B$ is a **proposition** which depends on hat may be true or false depending on the truth values of the variables in A and B.

$A \equiv B$ and $(A \leftrightarrow B) \equiv \mathbf{T}$ have the same meaning.

De Morgan's Laws

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

Negate the statement:

"My code compiles or there is a bug."

To negate the statement, ask "when is the original statement false".

It's false when not(my code compiles) AND not(there is a bug).

Translating back into English, we get:

My code doesn't compile and there is not a bug.

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	F	F	F	T	F	T
T	F	F	T	T	F	T	T
F	T	T	F	T	F	T	T
F	F	T	T	T	F	T	T

De Morgan's Laws

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

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

```
if (!(front != null && value > front.data))
    front = new ListNode(value, front);
else {
    ListNode current = front;
    while (current.next != null && current.next.data < value))
        current = current.next;
    current.next = new ListNode(value, current.next);
}
```

De Morgan's Laws

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

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

```
!(front != null && value > front.data)
```

$$\equiv$$

```
front == null || value <= front.data
```

You've been using these for a while!

Law of Implication

$$p \rightarrow q \equiv \neg p \vee q$$

p	q	$p \rightarrow q$	$\neg p$	$\neg p \vee q$	$p \rightarrow q \leftrightarrow \neg p \vee q$
T	T	T	F	T	T
T	F	F	F	F	T
F	T	T	T	T	T
F	F	T	T	T	T

Some Equivalences Related to Implication

$$p \rightarrow q \equiv \neg p \vee q$$

$$p \rightarrow q \equiv \neg q \rightarrow \neg p$$

$$p \leftrightarrow q \equiv (p \rightarrow q) \wedge (q \rightarrow p)$$

$$p \leftrightarrow q \equiv \neg p \leftrightarrow \neg q$$

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**
  - $p \vee \neg p \equiv \mathrm{T}$
  - $p \wedge \neg p \equiv \mathrm{F}$

Digital Circuits

Computing With Logic

- T corresponds to 1 or “high” voltage
- F corresponds to 0 or “low” voltage

Gates

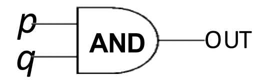
- Take inputs and produce outputs (functions)
- Several kinds of gates
- Correspond to propositional connectives (most of them)

And Gate

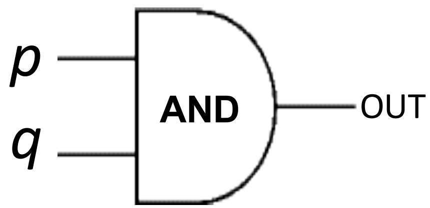
AND Connective vs. AND Gate

$p \wedge q$

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



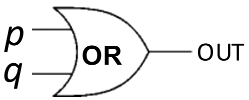
"block looks like D of AND"

Or Gate

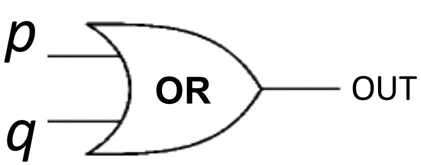
OR Connective vs. OR Gate

$p \vee q$

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



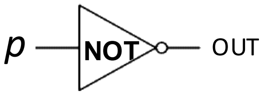
"arrowhead block looks like V"

Not Gates

NOT Connective vs. NOT Gate

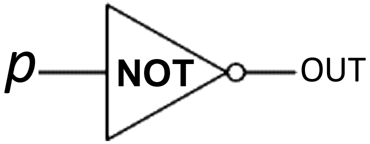
$\neg p$

p	$\neg$ *p*
T	F
F	T



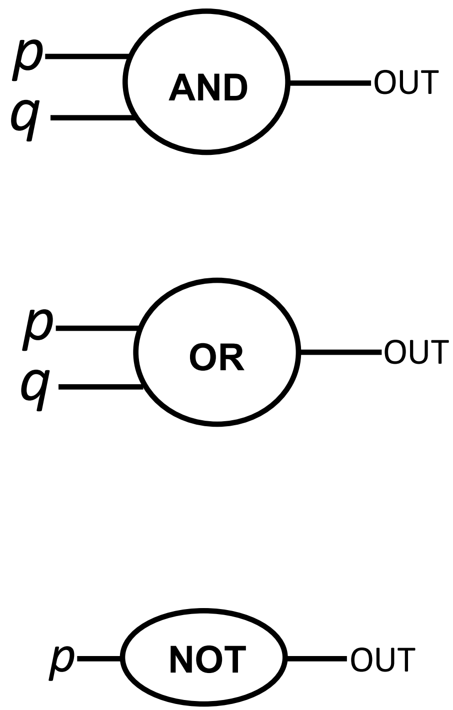
Also called *inverter*

p	OUT
1	0
0	1

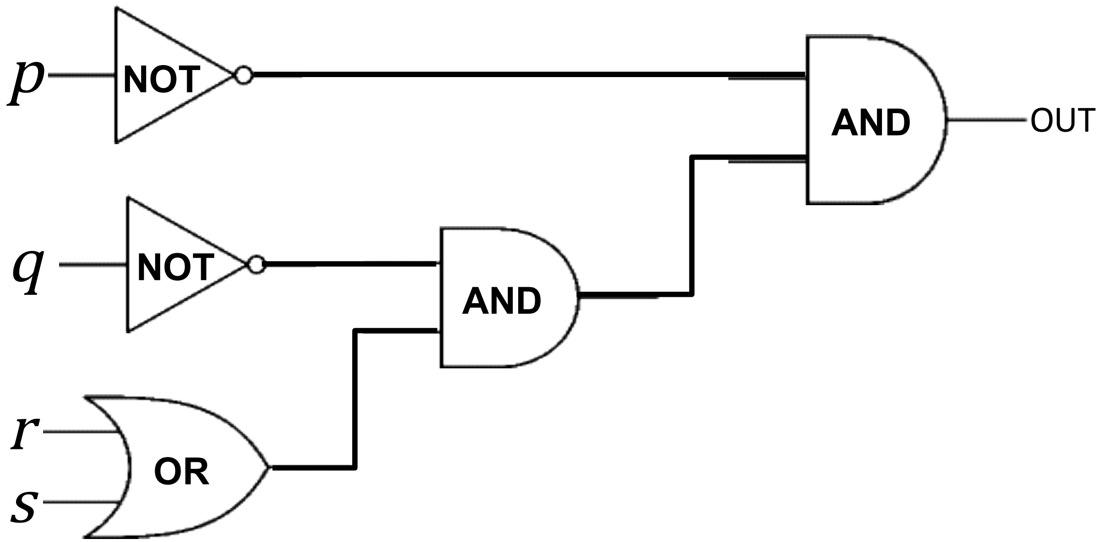


Blobs are Okay!

You may write gates using blobs instead of shapes!



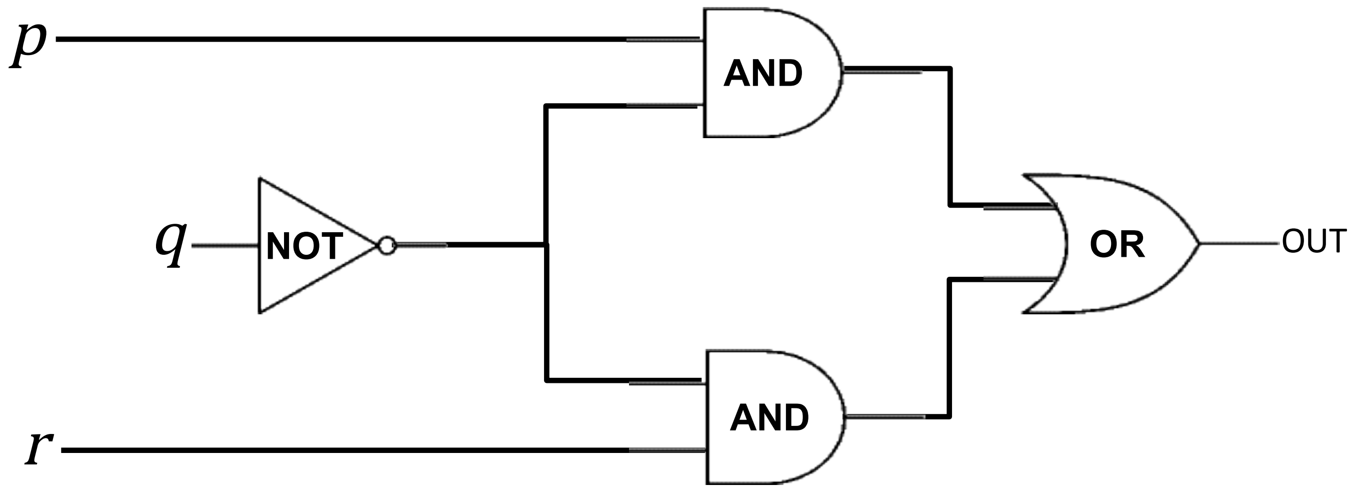
Combinational Logic Circuits



Values get sent along wires connecting gates

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

Combinational Logic Circuits



Wires can send one value to multiple gates!

$$(p \land \neg q) \lor (\neg q \land r)$$

Computing Equivalence

Describe an algorithm for computing if two logical expressions/circuits are equivalent.

What is the run time of the algorithm?

Compute the entire truth table for both of them!

There are 2^n entries in the column for *n* variables.

Some Familiar Properties of Arithmetic

- $x + y = y + x$ **(Commutativity)**
- $x \cdot (y + z) = x \cdot y + x \cdot z$ **(Distributivity)**
- $(x + y) + z = x + (y + z)$ **(Associativity)**

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