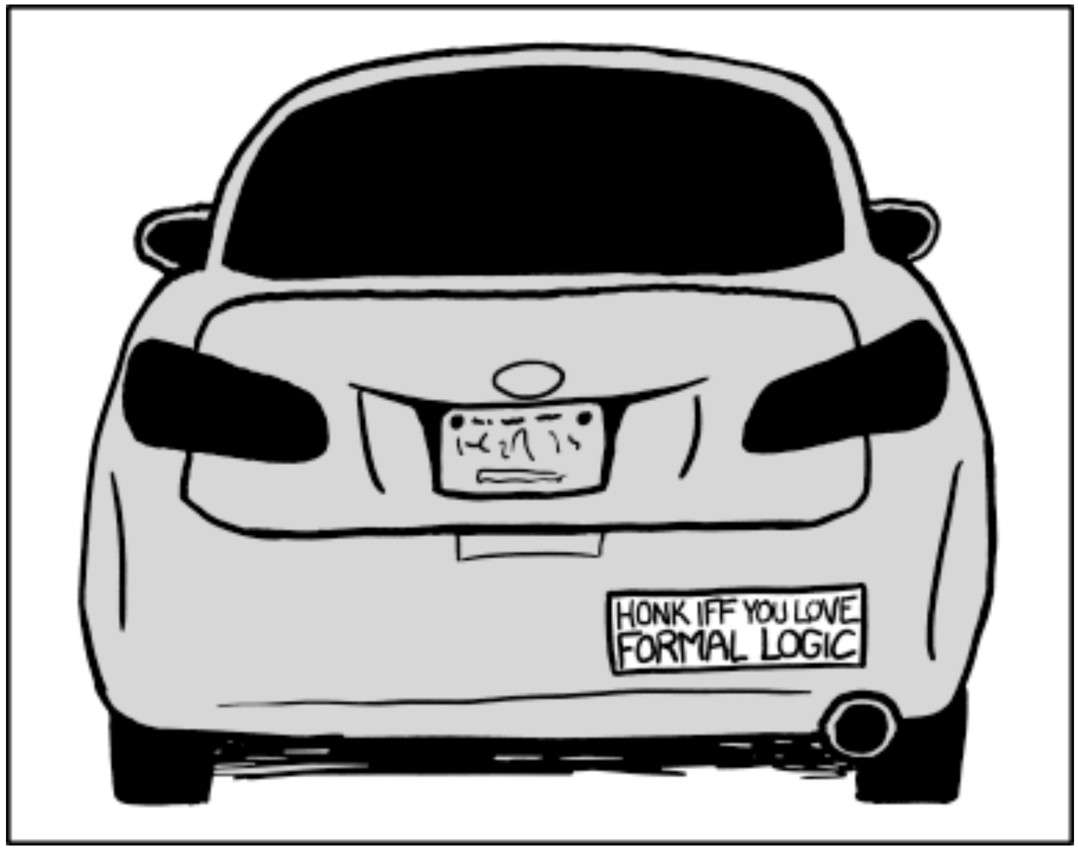
CSE 311: Foundations of Computing I

Topic 1: Propositional Logic



What is logic?

Logic is a language, like English or Java

Why learn another language?

We know English and Java already?

Why not use English?

– Turn right here…

Does “right” mean the direction or now?

– We saw her duck

Does “duck” mean the animal or crouch down?

– Buffalo buffalo Buffalo buffalo buffalo buffalo Buffalo buffalo

This means “Bison from Buffalo, that bison from Buffalo bully, themselves bully bison from Buffalo.

Natural languages can be unclear or imprecise

Why learn a new language?

We need a language of reasoning to

- state sentences more precisely
- state sentences more concisely
- understand sentences more quickly

Formal logic has these properties

What is logic and why do we need it?

Logic is a language, like English or Java, with its own

- words and rules for combining words into sentences (syntax)
- ways to assign meaning to words and sentences (semantics)

Propositions: building blocks of logic

A ***proposition*** is a statement that

- is "well-formed"
- is either true or false

Propositions: building blocks of logic

A ***proposition*** is a statement that

- is "well-formed"
- is either true or false

All cats are mammals

true

All mammals are cats

false

Are These Propositions?

$2 + 2 = 5$

This is a proposition. It's okay for propositions to be false.

$x + 2 = 5389$, where x is my PIN number

This is a proposition. We don't need to know what x is.

Akjsdf!

Not a proposition because it's gibberish.

Who are you?

This is a question which means it doesn't have a truth value.

Every positive even integer can be written as the sum of two primes.

This is a proposition. We don't know if it's true or false, but we know it's one of them!

Propositions

We need a way of talking about *arbitrary* ideas…

Propositional Variables: $p, q, r, s, \ldots$

Truth Values:

- **T** for **true**
- **F** for **false**

Familiar from Java

Java `boolean` represents a truth value

– constants `true` and `false`

– variables hold *unknown* values

Operators calculate new values from given ones

– unary: not (`!`)

– binary: and (&&), or (`||`)

Logical Connectives

Negation (not)	$\neg p$
Conjunction (and)	$p \wedge q$
Disjunction (or)	$p \vee q$
Exclusive Or	$p \oplus q$
Implication	$p \longrightarrow r$
Biconditional	$p \longleftrightarrow r$

Some Truth Tables

p	$\neg p$
T	
F	

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

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

p	q	$p \oplus q$
T	T	
T	F	
F	T	
F	F	

Some Truth Tables

p	$\neg p$
T	F
F	T

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

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

p	q	$p \oplus q$
T	T	F
T	F	T
F	T	T
F	F	F

Logic forces us to distinguish $\vee$ from $\oplus$

Implication

"If it's raining, then I have my umbrella"

p	r	$p \rightarrow r$
T	T	T
T	F	F
F	T	T
F	F	T

It's useful to think of implications as promises. That is "Did I lie?"

	It's raining	It's not raining
I have my umbrella		
I do not have my umbrella		

Implication

"If it's raining, then I have my umbrella"

p	r	$p \rightarrow r$
T	T	T
T	F	F
F	T	T
F	F	T

It's useful to think of implications as promises. That is "Did I lie?"

	It's raining	It's not raining
I have my umbrella	No	No
I do not have my umbrella	**Yes**	No

The only lie is when:

(a) It's raining AND

(b) I don't have my umbrella

Implication

"If the Seahawks won, then I was at the game."

p	*r*	$p \rightarrow r$
T	T	T
T	F	F
F	T	T
F	F	T

What's the one scenario where I lied?

	I was at the game	I wasn't at the game
Seahawks won		
Seahawks lost		

Implication

"If the Seahawks won, then I was at the game."

p	r	$p \rightarrow r$
T	T	T
T	F	F
F	T	T
F	F	T

What's the one scenario where I lied?

	I was at the game	I wasn't at the game
Seahawks won	Ok	**I lied**
Seahawks lost	Ok	Ok

Implication

"If it's raining, then I have my umbrella"

p	r	$p \to r$
T	T	T
T	F	F
F	T	T
F	F	T

Are these true?

$2 + 2 = 4 \to$ *earth is a planet*

The fact that these are unrelated doesn't make the statement false! "2 + 2 = 4" is true; "earth is a planet" is true. T→T is true. So, the statement is true.

$2 + 2 = 5 \to$ *26 is prime*

Again, these statements may or may not be related. "2 + 2 = 5" is false; so, the implication is true. (Whether 26 is prime or not is irrelevant).

Implication is not a causal relationship!

$p \to r$

---

(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"

In English, the "if" can be written at the end of the sentence rather than at the beginning of the sentence (followed by a ",").

$p \rightarrow r$

(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 implications in opposite directions:

(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.

$p \to r$

p	r	$p \to r$
T	T	T
T	F	F
F	T	T
F	F	T

Implication:

– *p* implies *r*

– whenever *p* is true *r* must be true

– if *p* then *r*

– *r* if *p*

– *p* only if *r*

– *p* is sufficient for *r*

– *r is necessary for p*

Biconditional: $p \leftrightarrow r$

- *p* if and only if *r*
- *p* “iff” *r*
  - *p and r have the same value truth value*

p	r	$p \leftrightarrow r$
T	**T**	**T**
T	**F**	**F**
F	**T**	**F**
F	**F**	**T**

A Compound Proposition (Practical Example)

"Show the notification to the user if its their second login or they've used it for two weeks and haven't tried the feature X unless they did use the feature Y."

Not at all clear what exactly this means!

Can use logic to understand exactly when to show it

A Compound Proposition (Silly Example)

"Garfield has black stripes if he is an orange cat and likes lasagna, and he is an orange cat or does not like lasagna"

We'd like to *understand* what this proposition means.

A Compound Proposition

"Garfield has black stripes if he is an orange cat and likes lasagna, and he is an orange cat or does not like lasagna"

We'd like to *understand* what this proposition means.

First find the simplest (**atomic) propositions**:

q "Garfield has black stripes"

r "Garfield is an orange cat"

s "Garfield likes lasagna"

(q if (r and s)) and (r or (not s))

Logical Connectives

Negation (not)	$\neg p$
Conjunction (and)	$p \land q$
Disjunction (or)	$p \lor q$
Exclusive Or	$p \oplus q$
Implication	$p \longrightarrow r$
Biconditional	$p \longleftrightarrow r$

q	"Garfield has black stripes"
r	"Garfield is an orange cat"
s	"Garfield likes lasagna"

"Garfield has black stripes if he is an orange cat and likes lasagna, and he is an orange cat or does not like lasagna"

↓

(q if (r and s)) and (r or (not s))

Logical Connectives

Negation (not)	$\neg p$
Conjunction (and)	$p \wedge q$
Disjunction (or)	$p \vee q$
Exclusive Or	$p \oplus q$
Implication	$p \longrightarrow r$
Biconditional	$p \leftrightarrow r$

q "Garfield has black stripes"
r "Garfield is an orange cat"
s "Garfield likes lasagna"

"Garfield has black stripes if he is an orange cat and likes lasagna, and he is an orange cat or does not like lasagna"

↓

(q if (r and s)) and (r or (not s))

↓

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

Analyzing the Garfield Sentence with a Truth Table

q	r	s	$((r \land s) \to q) \land (r \lor \neg s)$
F	F	F	
F	F	T	
F	T	F	
F	T	T	
T	F	F	
T	F	T	
T	T	F	
T	T	T	

Analyzing the Garfield Sentence with a Truth Table

q	r	s	$r \vee \neg s$	$(r \wedge s) \rightarrow q$	$((r \wedge s) \rightarrow q) \wedge (r \vee \neg s)$
F	F	F			
F	F	T			
F	T	F			
F	T	T			
T	F	F			
T	F	T			
T	T	F			
T	T	T			

Analyzing the Garfield Sentence with a Truth Table

q	r	s	$\neg s$	$r \vee \neg s$	$r \wedge s$	$(r \wedge s) \rightarrow q$	$((r \wedge s) \rightarrow q) \wedge (r \vee \neg s)$
F	F	F					
F	F	T					
F	T	F					
F	T	T					
T	F	F					
T	F	T					
T	T	F					
T	T	T					

Analyzing the Garfield Sentence with a Truth Table

q	r	s	$\neg s$	$r \vee \neg s$	$r \wedge s$	$(r \wedge s) \rightarrow q$	$((r \wedge s) \rightarrow q) \wedge (r \vee \neg s)$
F	F	F	T	T	F	T	T
F	F	T	F	F	F	T	F
F	T	F	T	T	F	T	T
F	T	T	F	T	T	F	F
T	F	F	T	T	F	T	T
T	F	T	F	F	F	T	F
T	T	F	T	T	F	T	T
T	T	T	F	T	T	T	T

Understanding Garfield Claim

"Garfield has black stripes if he is an orange cat and likes lasagna, and he is an orange cat or does not like lasagna"

Black Stripes	Orange	Likes Lasagna	Claim
F	F	F	**T**
F	F	T	F
F	T	F	**T**
F	T	T	F
T	F	F	**T**
T	F	T	F
T	T	F	**T**
T	T	T	**T**

Propositional Logic makes clear exactly what is being claimed.

Formalization

Most problems come to us in English

– can be hard to understand (easy to *misunderstand*)

First step is to "formalize"

– translate into a precise, mathematical statement

Then, we can apply our full tool set tools…

Converse, Contrapositive

Implication:

$p \rightarrow r$

Converse:

$r \rightarrow p$

Contrapositive:

$\neg r \rightarrow \neg p$

Inverse:

$\neg p \rightarrow \neg r$

Consider

p: 6 is divisible by 2

r: 6 is divisible by 4

$p \rightarrow r$	
$r \rightarrow p$	
$\neg r \rightarrow \neg p$	
$\neg p \rightarrow \neg r$	

Converse, Contrapositive

Implication:

$p \rightarrow r$

Converse:

$r \rightarrow p$

Contrapositive:

$\neg r \rightarrow \neg p$

Inverse:

$\neg p \rightarrow \neg r$

<u>Consider</u>

p: 6 is divisible by 2

r: 6 is divisible by 4

$p \rightarrow r$	F
$r \rightarrow p$	T
$\neg r \rightarrow \neg p$	F
$\neg p \rightarrow \neg r$	T

Converse, Contrapositive

Implication:

$p \to r$

Contrapositive:

$\neg r \to \neg p$

Converse:

$r \to p$

Inverse:

$\neg p \to \neg r$

How do these relate to each other?

p	r	$p \to r$	$r \to p$	$\neg p$	$\neg r$	$\neg p \to \neg r$	$\neg r \to \neg p$
T	T						
T	F						
F	T						
F	F						

Converse, Contrapositive

Implication:

$p \rightarrow r$

Contrapositive:

$\neg r \rightarrow \neg p$

Converse:

$r \rightarrow p$

Inverse:

$\neg p \rightarrow \neg r$

An implication and its contrapositive
have the same truth value!

p	r	$p \rightarrow r$	$r \rightarrow p$	$\neg p$	$\neg r$	$\neg p \rightarrow \neg r$	$\neg r \rightarrow \neg p$
T	T	T	T	F	F	T	T
T	F	F	T	F	T	T	F
F	T	T	F	T	F	F	T
F	F	T	T	T	T	T	T

Converse, Contrapositive

Implication:

$p \to r$

Contrapositive:

$\neg r \to \neg p$

Converse:

$r \to p$

Inverse:

$\neg p \to \neg r$

An implication and its inverse
do not have the same truth value!

p	r	$p \to r$	$r \to p$	$\neg p$	$\neg r$	$\neg p \to \neg r$	$\neg r \to \neg p$

Assuming they are the same is called the

"Fallacy of the Inverse"

Application: Digital Circuits

Computing With Logic

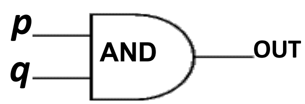
- T corresponds to 1 or "high" voltage
- F corresponds to 0 or "low" voltage

Gates

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

Last class: AND, OR, NOT Gates

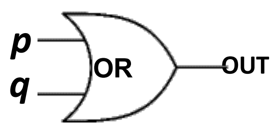
AND Gate



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

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

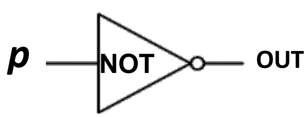
OR Gate



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

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

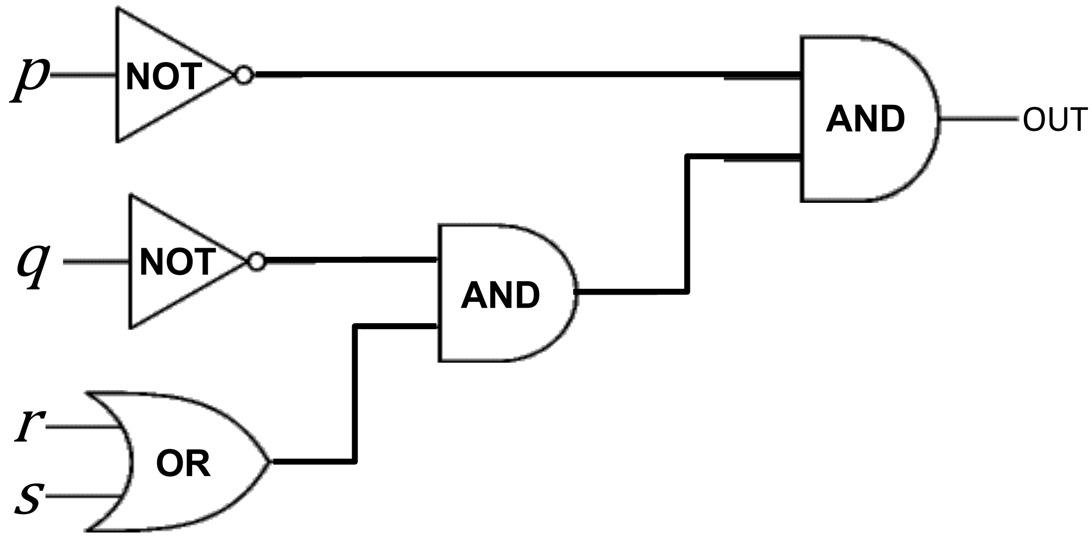
NOT Gate



p	OUT
1	0
0	1

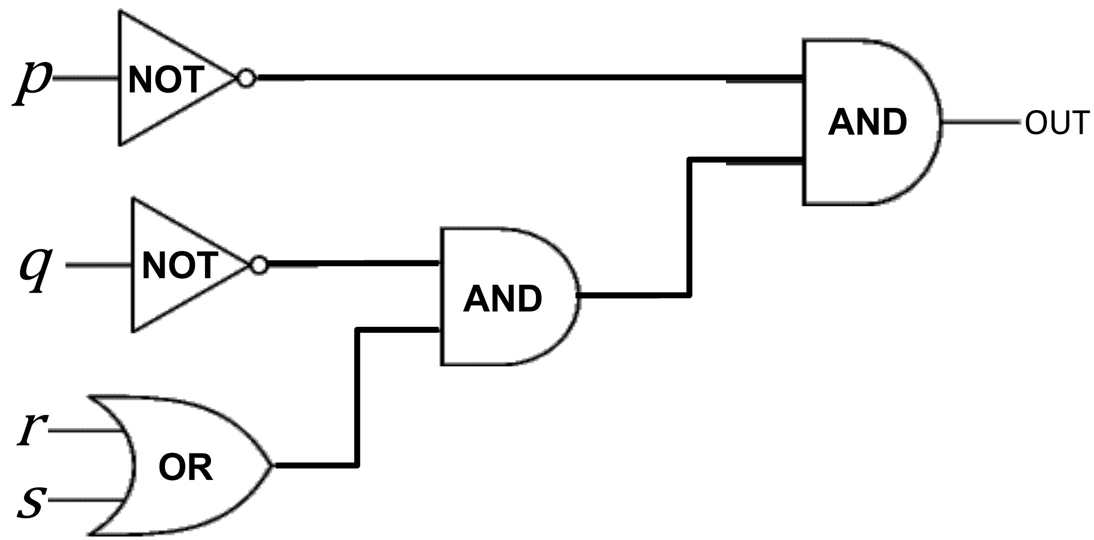
p	$\neg p$
T	F
F	T

Combinational Logic Circuits



Values get sent along wires connecting gates

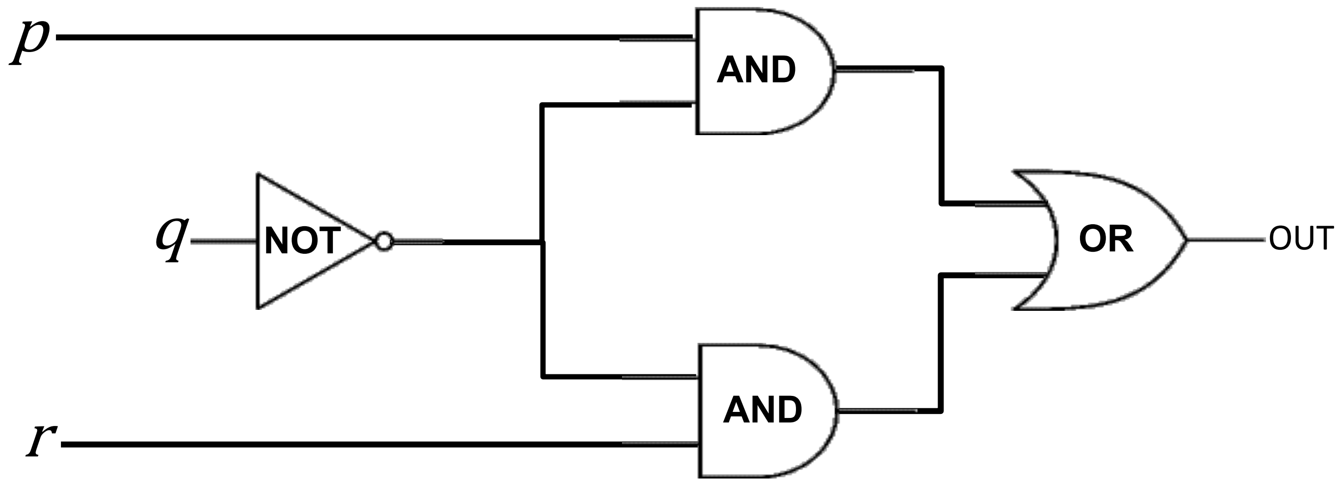
Combinational Logic Circuits



Values get sent along wires connecting gates

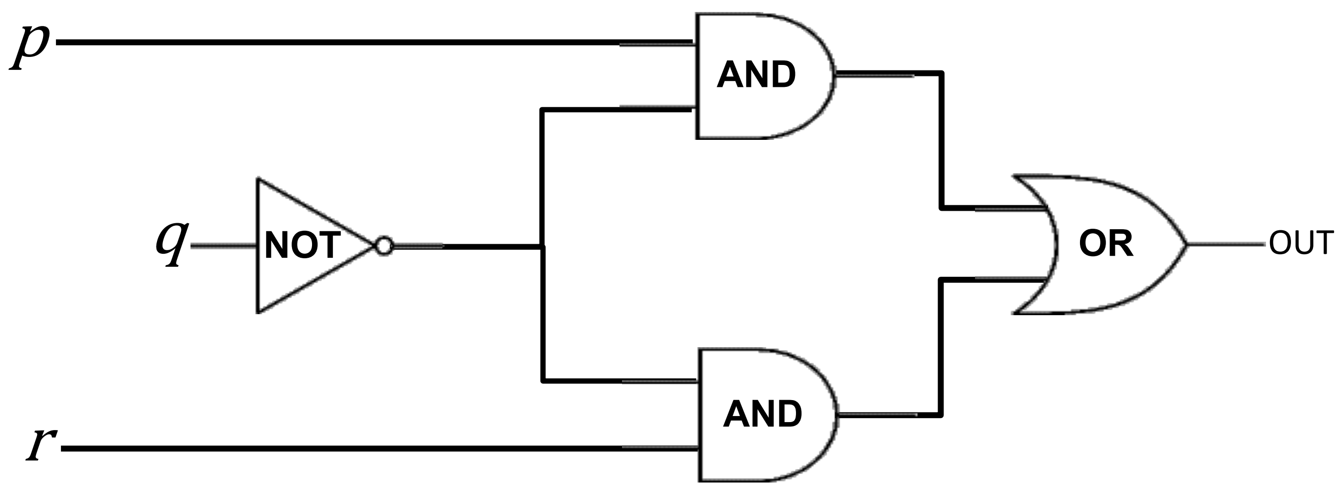
$$\neg p \wedge (\neg q \wedge (r \vee s))$$

Combinational Logic Circuits



Wires can send one value to multiple gates!

Combinational Logic Circuits



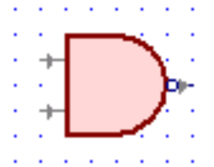
Wires can send one value to multiple gates!

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

Other Useful Gates

NAND

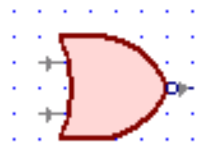
$\neg(p \wedge q)$



p	q	out
0	0	1
0	1	1
1	0	1
1	1	0

NOR

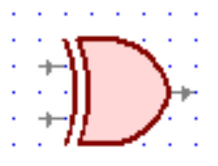
$\neg(p \vee q)$



p	q	out
0	0	1
0	1	0
1	0	0
1	1	0

XOR

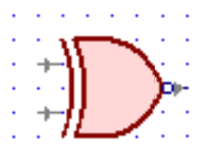
$p \oplus q$



p	q	out
0	0	0
0	1	1
1	0	1
1	1	0

XNOR

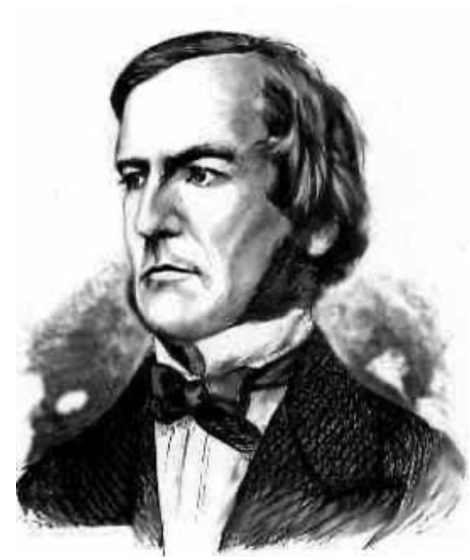
$p \leftrightarrow q$



p	q	out
0	0	1
0	1	0
1	0	0
1	1	1

Boolean Algebra

- **Usual notation used in circuit design**
- **Boolean algebra**
  - a set of elements B containing {0, 1}
  - binary operations { + , • }
  - and a unary operation { a' } or { $\bar{a}$ }



Write these in Boolean Algebra:

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

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

Boolean Algebra

- Usual notation used in circuit design

- Boolean algebra
  - a set of elements B containing {0, 1}
  - binary operations { + , • }
  - and a unary operation { a' } or { $\bar{a}$ }

Write these in Boolean Algebra:

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

$p'q'(r+s)$ $pq' + q'r$

Warm-up Exercise

- Create a Boolean Algebra expression for C below in terms of the variables $\boldsymbol{a}$ and $\boldsymbol{b}$

$\boldsymbol{a}$	$\boldsymbol{b}$	$\boldsymbol{C(a,b)}$
1	1	0
1	0	1
0	1	1
0	0	0

Note that we have only +, * (not XOR)

$$C = (a+b)(ab)'$$

How do we do this with many variables?
Would be nice to have a mechanical way to do it! (More later...)

Warm-up Exercise

- Create a Boolean Algebra expression for C below in terms of the variables $\boldsymbol{a}$ and $\boldsymbol{b}$

$$C = (a + b)(ab)'$$

- Draw this as a circuit (using AND, OR, NOT)

Correspondence between logic (propositions) and computation (circuits). This isn't the last time we will see such connections…

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

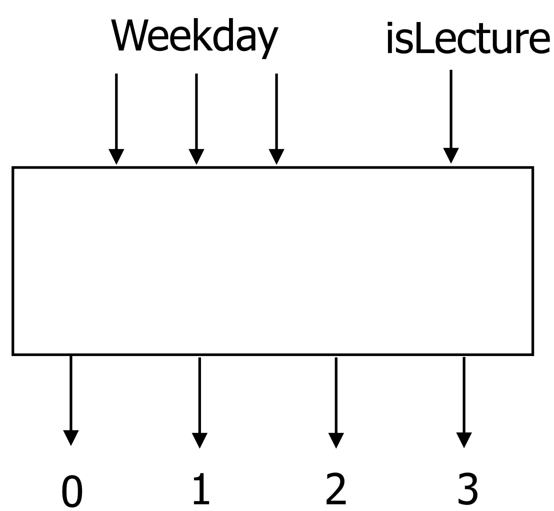
Implementation in Software

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

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

Weekday		isLecture	c_0	c_1	c_2	c_3
SUN	000	0				
SUN	000	1				
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	-				

Converting to a Truth Table!

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

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

Truth Table to Logic (Part 1)

	$d_2d_1d_0$	L	c_0	c_1	c_2	c_3
SUN	000	0	0	1	0	0
SUN	000	1	0	0	0	1
MON	001	0	0	1	0	0
MON	001	1	0	0	0	1
TUE	010	0	0	1	0	0
TUE	010	1	0	0	1	0
WED	011	0	0	1	0	0
WED	011	1	0	0	1	0
THU	100	-	0	1	0	0
FRI	101	0	1	0	0	0
FRI	101	1	0	1	0	0
SAT	110	-	1	0	0	0
-	111	-	1	0	0	0

Let's begin by finding an expression for c_3. To do this, we look at the rows where c_3 = 1 (true).

Truth Table to Logic (Part 1)

	$d_2d_1d_0$	L	c_0	c_1	c_2	c_3
SUN	000	0	0	1	0	0
SUN	000	1	0	0	0	1
MON	001	0	0	1	0	0
MON	001	1	0	0	0	1
TUE	010	0	0	1	0	0
TUE	010	1	0	0	1	0
WED	011	0	0	1	0	0
WED	011	1	0	0	1	0
THU	100	-	0	1	0	0
FRI	101	0	1	0	0	0
FRI	101	1	0	1	0	0
SAT	110	-	1	0	0	0
-	111	-	1	0	0	0

DAY == SUN && L == 1

DAY == MON && L == 1

Truth Table to Logic (Part 1)

	$d_2d_1d_0$	L	c_0	c_1	c_2	c_3
SUN	000	0	0	1	0	0
SUN	000	1	0	0	0	1
MON	001	0	0	1	0	0
MON	001	1	0	0	0	1
TUE	010	0	0	1	0	0
TUE	010	1	0	0	1	0
WED	011	0	0	1	0	0
WED	011	1	0	0	1	0
THU	100	-	0	1	0	0
FRI	101	0	1	0	0	0
FRI	101	1	0	1	0	0
SAT	110	-	1	0	0	0
-	111	-	1	0	0	0

$d_2d_1d_0$ == 000 && L == 1

$d_2d_1d_0$ == 001 && L == 1

Substituting DAY for the binary representation.

Truth Table to Logic (Part 1)

	$d_2d_1d_0$	L	c_0	c_1	c_2	c_3
SUN	000	0	0	1	0	0
SUN	000	1	0	0	0	1
MON	001	0	0	1	0	0
MON	001	1	0	0	0	1
TUE	010	0	0	1	0	0
TUE	010	1	0	0	1	0
WED	011	0	0	1	0	0
WED	011	1	0	0	1	0
THU	100	-	0	1	0	0
FRI	101	0	1	0	0	0
FRI	101	1	0	1	0	0
SAT	110	-	1	0	0	0
-	111	-	1	0	0	0

SUN, L = 1 row → **d_2 == 0 && d_1 == 0 && d_0 == 0 && L == 1**

MON, L = 1 row → **d_2 == 0 && d_1 == 0 && d_0 == 1 && L == 1**

Splitting up the bits of the day; so, we can write a formula.

Truth Table to Logic (Part 1)

	$d_2d_1d_0$	L	c_0	c_1	c_2	c_3
SUN	000	0	0	1	0	0
SUN	000	1	0	0	0	1
MON	001	0	0	1	0	0
MON	001	1	0	0	0	1
TUE	010	0	0	1	0	0
TUE	010	1	0	0	1	0
WED	011	0	0	1	0	0
WED	011	1	0	0	1	0
THU	100	-	0	1	0	0
FRI	101	0	1	0	0	0
FRI	101	1	0	1	0	0
SAT	110	-	1	0	0	0
-	111	-	1	0	0	0

$d_2' \cdot d_1' \cdot d_0' \cdot L$

$d_2' \cdot d_1' \cdot d_0 \cdot L$

Replacing with Boolean Algebra...

Truth Table to Logic (Part 1)

	$d_2d_1d_0$	L	c_0	c_1	c_2	c_3
SUN	000	0	0	1	0	0
SUN	000	1	0	0	0	1
MON	001	0	0	1	0	0
MON	001	1	0	0	0	1
TUE	010	0	0	1	0	0
TUE	010	1	0	0	1	0
WED	011	0	0	1	0	0
WED	011	1	0	0	1	0
THU	100	-	0	1	0	0
FRI	101	0	1	0	0	0
FRI	101	1	0	1	0	0
SAT	110	-	1	0	0	0
-	111	-	1	0	0	0

SUN, 000, L=1 → $d_2' \cdot d_1' \cdot d_0' \cdot L$

MON, 001, L=1 → $d_2' \cdot d_1' \cdot d_0 \cdot L$

How do we combine them?

Truth Table to Logic (Part 1)

	$d_2d_1d_0$	L	c_0	c_1	c_2	c_3
SUN	000	0	0	1	0	0
SUN	000	1	0	0	0	1
MON	001	0	0	1	0	0
MON	001	1	0	0	0	1
TUE	010	0	0	1	0	0
TUE	010	1	0	0	1	0
WED	011	0	0	1	0	0
WED	011	1	0	0	1	0
THU	100	-	0	1	0	0
FRI	101	0	1	0	0	0
FRI	101	1	0	1	0	0
SAT	110	-	1	0	0	0
-	111	-	1	0	0	0

$d_2' \cdot d_1' \cdot d_0' \cdot L$

$d_2' \cdot d_1' \cdot d_0 \cdot L$

Either situation causes c_3 to be true. So, we “or” them.

$$c_3 = d_2' \cdot d_1' \cdot d_0' \cdot L + d_2' \cdot d_1' \cdot d_0 \cdot L$$

Truth Table to Logic (Part 2)

	$d_2d_1d_0$	L	c_0	c_1	c_2	c_3
SUN	000	0	0	1	0	0
SUN	000	1	0	0	0	1
MON	001	0	0	1	0	0
MON	001	1	0	0	0	1
TUE	010	0	0	1	0	0
TUE	010	1	0	0	1	0
WED	011	0	0	1	0	0
WED	011	1	0	0	1	0
THU	100	-	0	1	0	0
FRI	101	0	1	0	0	0
FRI	101	1	0	1	0	0
SAT	110	-	1	0	0	0
-	111	-	1	0	0	0

$$c_3 = d_2' \cdot d_1' \cdot d_0' \cdot L + d_2' \cdot d_1' \cdot d_0 \cdot L$$

Now, we do c_2.

Truth Table to Logic (Part 3)

	$d_2d_1d_0$	L	c_0	c_1	c_2	c_3
SUN	000	0	0	1	0	0
SUN	000	1	0	0	0	1
MON	001	0	0	1	0	0
MON	001	1	0	0	0	1
TUE	010	0	0	1	0	0
TUE	010	1	0	0	1	0
WED	011	0	0	1	0	0
WED	011	1	0	0	1	0
THU	100	-	0	1	0	0
FRI	101	0	1	0	0	0
FRI	101	1	0	1	0	0
SAT	110	-	1	0	0	0
-	111	-	1	0	0	0

Now, we do **c_1**:

$$c_3 = d_2' \cdot d_1' \cdot d_0' \cdot L + d_2' \cdot d_1' \cdot d_0 \cdot L$$

$$c_2 = d_2' \cdot d_1 \cdot d_0' \cdot L + d_2' \cdot d_1 \cdot d_0 \cdot L$$

Truth Table to Logic (Part 3)

	$d_2d_1d_0$	L	c_0	c_1	c_2	c_3
SUN	000	0	0	1	0	0
SUN	000	1	0	0	0	1
MON	001	0	0	1	0	0
MON	001	1	0	0	0	1
TUE	010	0	0	1	0	0
TUE	010	1	0	0	1	0
WED	011	0	0	1	0	0
WED	011	1	0	0	1	0
THU	100	-	0	1	0	0
FRI	101	0	1	0	0	0
FRI	101	1	0	1	0	0
SAT	110	-	1	0	0	0
-	111	-	1	0	0	0

Now, we do c_1:

- SUN 000, L = 0 → $d_2' \cdot d_1' \cdot d_0' \cdot L'$
- MON 001, L = 0 → $d_2' \cdot d_1' \cdot d_0 \cdot L'$
- TUE 010, L = 0 → $d_2' \cdot d_1 \cdot d_0' \cdot L'$
- WED 011, L = 0 → $d_2' \cdot d_1 \cdot d_0 \cdot L'$
- THU 100, L = - → ???
- FRI 101, L = 1 → $d_2 \cdot d_1' \cdot d_0 \cdot L$

$c_3 = d_2' \cdot d_1' \cdot d_0' \cdot L + d_2' \cdot d_1' \cdot d_0 \cdot L$

$c_2 = d_2' \cdot d_1 \cdot d_0' \cdot L + d_2' \cdot d_1 \cdot d_0 \cdot L$

Truth Table to Logic (Part 3)

	$d_2d_1d_0$	L	c_0	c_1	c_2	c_3
SUN	000	0	0	1	0	0
SUN	000	1	0	0	0	1
MON	001	0	0	1	0	0
MON	001	1	0	0	0	1
TUE	010	0	0	1	0	0
TUE	010	1	0	0	1	0
WED	011	0	0	1	0	0
WED	011	1	0	0	1	0
THU	100	-	0	1	0	0
FRI	101	0	1	0	0	0
FRI	101	1	0	1	0	0
SAT	110	-	1	0	0	0
-	111	-	1	0	0	0

Now, we do c_1:

$d_2' \cdot d_1' \cdot d_0' \cdot L'$

$d_2' \cdot d_1' \cdot d_0 \cdot L'$

$d_2' \cdot d_1 \cdot d_0' \cdot L'$

$d_2' \cdot d_1 \cdot d_0 \cdot L'$

$d_2 \cdot d_1' \cdot d_0'$

$d_2 \cdot d_1' \cdot d_0 \cdot L$

No matter what L is, we always say it's 1. So, we don't need L in the expression.

$$c_3 = d_2' \cdot d_1' \cdot d_0' \cdot L + d_2' \cdot d_1' \cdot d_0 \cdot L$$

$$c_2 = d_2' \cdot d_1 \cdot d_0' \cdot L + d_2' \cdot d_1 \cdot d_0 \cdot L$$

Truth Table to Logic (Part 3)

	$d_2d_1d_0$	L	c_0	c_1	c_2	c_3	
SUN	000	0	0	1	0	0	→ $d_2' \cdot d_1' \cdot d_0' \cdot L'$
SUN	000	1	0	0	0	1	
MON	001	0	0	1	0	0	→ $d_2' \cdot d_1' \cdot d_0 \cdot L'$
MON	001	1	0	0	0	1	
TUE	010	0	0	1	0	0	→ $d_2' \cdot d_1 \cdot d_0' \cdot L'$
TUE	010	1	0	0	1	0	
WED	011	0	0	1	0	0	→ $d_2' \cdot d_1 \cdot d_0 \cdot L'$
WED	011	1	0	0	1	0	
THU	100	-	0	1	0	0	→ $d_2 \cdot d_1' \cdot d_0'$
FRI	101	0	1	0	0	0	
FRI	101	1	0	1	0	0	→ $d_2 \cdot d_1' \cdot d_0 \cdot L$
SAT	110	-	1	0	0	0	
-	111	-	1	0	0	0	

Now, we do c_1:

$$c_3 = d_2' \cdot d_1' \cdot d_0' \cdot L + d_2' \cdot d_1' \cdot d_0 \cdot L$$

$$c_2 = d_2' \cdot d_1 \cdot d_0' \cdot L + d_2' \cdot d_1 \cdot d_0 \cdot L$$

$$c_1 = d_2' \cdot d_1' \cdot d_0' \cdot L' + d_2' \cdot d_1' \cdot d_0 \cdot L' + d_2' \cdot d_1 \cdot d_0' \cdot L' + d_2' \cdot d_1 \cdot d_0 \cdot L' + d_2 \cdot d_1' \cdot d_0' + d_2 \cdot d_1' \cdot d_0 \cdot L$$

Truth Table to Logic (Part 4)

	$d_2d_1d_0$	L	c_0	c_1	c_2	c_3
SUN	000	0	0	1	0	0
SUN	000	1	0	0	0	1
MON	001	0	0	1	0	0
MON	001	1	0	0	0	1
TUE	010	0	0	1	0	0
TUE	010	1	0	0	1	0
WED	011	0	0	1	0	0
WED	011	1	0	0	1	0
THU	100	-	0	1	0	0
FRI	101	0	1	0	0	0
FRI	101	1	0	1	0	0
SAT	110	-	1	0	0	0
-	111	-	1	0	0	0

$$c_1 = d_2' \cdot d_1' \cdot d_0' \cdot L' + d_2' \cdot d_1' \cdot d_0 \cdot L' + d_2' \cdot d_1 \cdot d_0' \cdot L' + d_2' \cdot d_1 \cdot d_0 \cdot L' + d_2 \cdot d_1' \cdot d_0' + d_2 \cdot d_1' \cdot d_0 \cdot L$$

$$c_2 = d_2' \cdot d_1 \cdot d_0' \cdot L + d_2' \cdot d_1 \cdot d_0 \cdot L$$

$$c_3 = d_2' \cdot d_1' \cdot d_0' \cdot L + d_2' \cdot d_1' \cdot d_0 \cdot L$$

Finally, we do c_0:

FRI 101, L = 0 → $d_2 \cdot d_1' \cdot d_0 \cdot L'$

SAT 110 → $d_2 \cdot d_1 \cdot d_0'$

111 → $d_2 \cdot d_1 \cdot d_0$

Truth Table to Logic (Part 4)

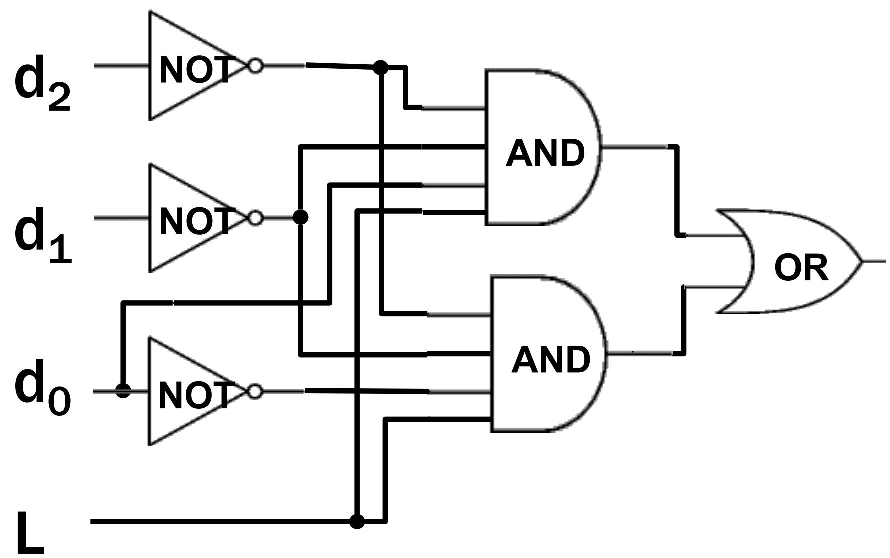
$c_0 = d_2 \cdot d_1' \cdot d_0 \cdot L' + d_2 \cdot d_1 \cdot d_0' + d_2 \cdot d_1 \cdot d_0$

$c_1 = d_2' \cdot d_1' \cdot d_0' \cdot L' + d_2' \cdot d_1' \cdot d_0 \cdot L' + d_2' \cdot d_1 \cdot d_0' \cdot L' + d_2' \cdot d_1 \cdot d_0 \cdot L' + d_2 \cdot d_1' \cdot d_0' + d_2 \cdot d_1' \cdot d_0 \cdot L$

$c_2 = d_2' \cdot d_1 \cdot d_0' \cdot L + d_2' \cdot d_1 \cdot d_0 \cdot L$

$c_3 = d_2' \cdot d_1' \cdot d_0' \cdot L + d_2' \cdot d_1' \cdot d_0 \cdot L$

Here's c_3 as a circuit:



Important Corollary of this Construction

$\neg$, $\wedge$, $\vee$ can implement any Boolean function!

no need for XOR, XNOR, etc.

Why? Because this construction only uses $\neg$, $\wedge$, $\vee$

works for any boolean function

General Computing from Other Gates

A theoretical example…

Canonical Forms

- Truth table is the **unique signature** of a 0/1 function

- The same truth table can have many circuit realizations
  - many ways to compute the same thing

- Can we choose a circuit so same table → same circuit?

- Yes: Canonical forms
  - standard forms as a Boolean expression (circuit)
  - we all produce the same expression

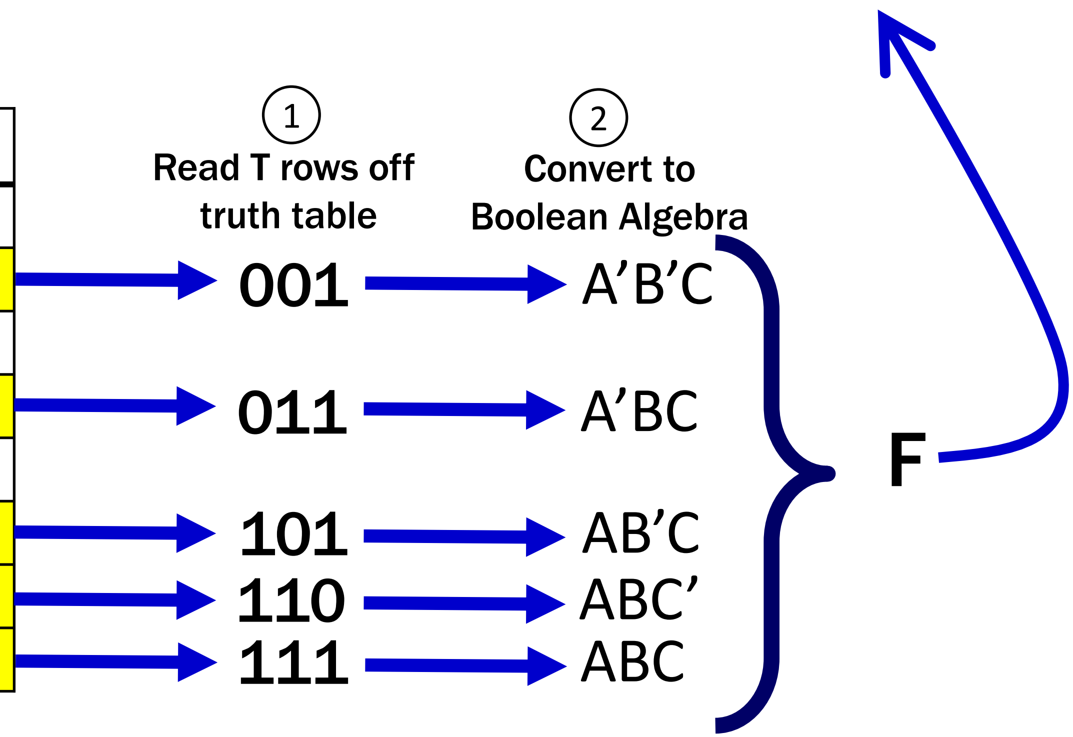
Sum-of-Products Canonical Form

- AKA **Disjunctive Normal Form (DNF)**
- AKA **Minterm Expansion**

③ **Add the minterms together**

$$F = A'B'C + A'BC + AB'C + ABC' + ABC'$$

A	B	C	F
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1



Sum-of-Products Canonical Form

Product term (or minterm)

- ANDed product of literals – input combination for which output is true
- each variable appears exactly once, true or inverted (but not both)

A	B	C	minterms
0	0	0	A′B′C′
0	0	1	A′B′C
0	1	0	A′BC′
0	1	1	A′BC
1	0	0	AB′C′
1	0	1	AB′C
1	1	0	ABC′
1	1	1	ABC

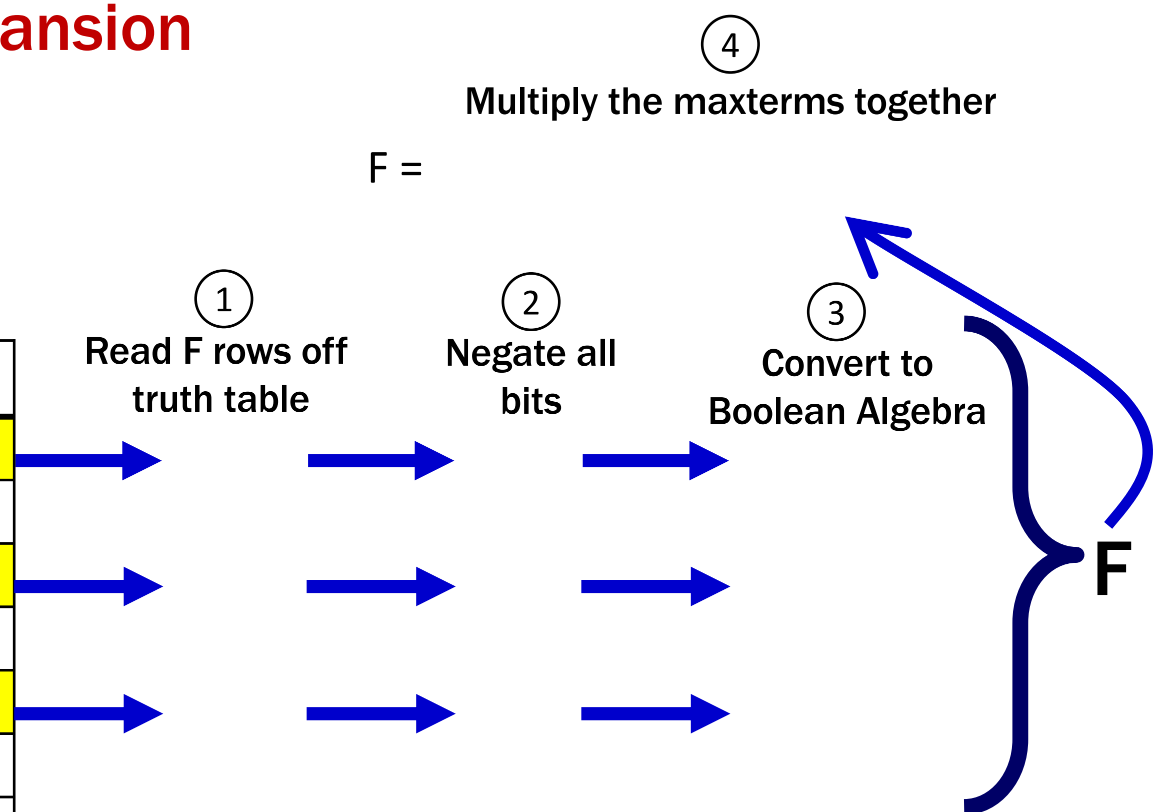
F in canonical form:

F(A, B, C) = A′B′C + A′BC + AB′C + ABC′ + ABC

Product-of-Sums Canonical Form

- AKA **Conjunctive Normal Form (CNF)**
- AKA **Maxterm Expansion**

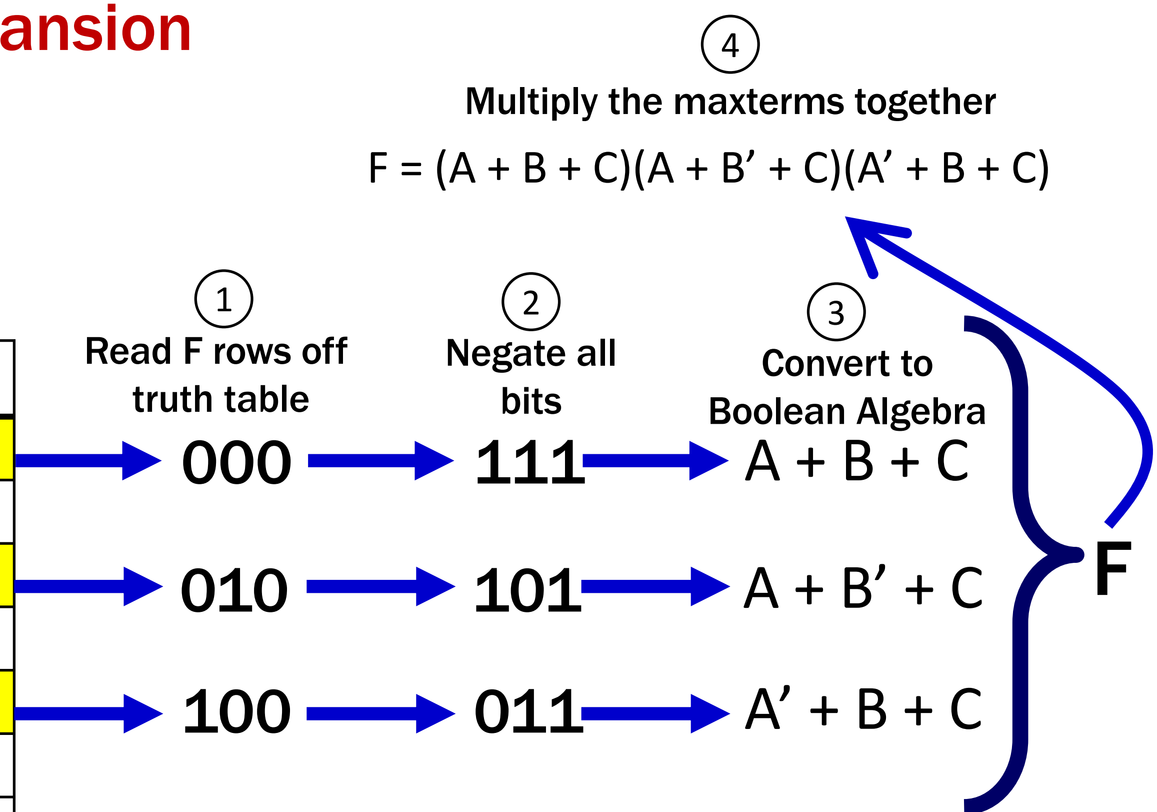
A	B	C	F
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1



Product-of-Sums Canonical Form

- AKA Conjunctive Normal Form (CNF)
- AKA Maxterm Expansion

A	B	C	F
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1



Product-of-Sums: Why does this procedure work?

Useful Facts:

- **We know (F')' = F**
- **We know how to get a minterm expansion for F'**

A	B	C	F
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

$$F' = A'B'C' + A'BC' + AB'C'$$

Product-of-Sums: Why does this procedure work?

Useful Facts:

- We know (F')' = F
- We know how to get a **minterm** expansion for F'

A	B	C	F
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

$$F' = A'B'C' + A'BC' + AB'C'$$

Taking the complement of both sides...

$$(F')' = (A'B'C' + A'BC' + AB'C')'$$

And using DeMorgan/Comp....

$$F = (A'B'C')' \ (A'BC')' \ (AB'C')'$$

$$F = (A + B + C)(A + B' + C)(A' + B + C)$$

Product-of-Sums Canonical Form

Sum term (or maxterm)

- ORed sum of literals – input combination for which output is false
- each variable appears exactly once, true or inverted (but not both)

A	B	C	maxterms
0	0	0	A+B+C
0	0	1	A+B+C′
0	1	0	A+B′+C
0	1	1	A+B′+C′
1	0	0	A′+B+C
1	0	1	A′+B+C′
1	1	0	A′+B′+C
1	1	1	A′+B′+C′

F in canonical form:

F(A, B, C) = (A + B + C) (A + B′ + C) (A′ + B + C)

Mapping Truth Tables to Logic Gates

Given a truth table:

1. Write the output in a table
2. Write the Boolean expression
3. Draw as gates
4. Map to available gates

A	B	C	F
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	1

This will give us *some* circuit.
But is it the <u>best</u> circuit?