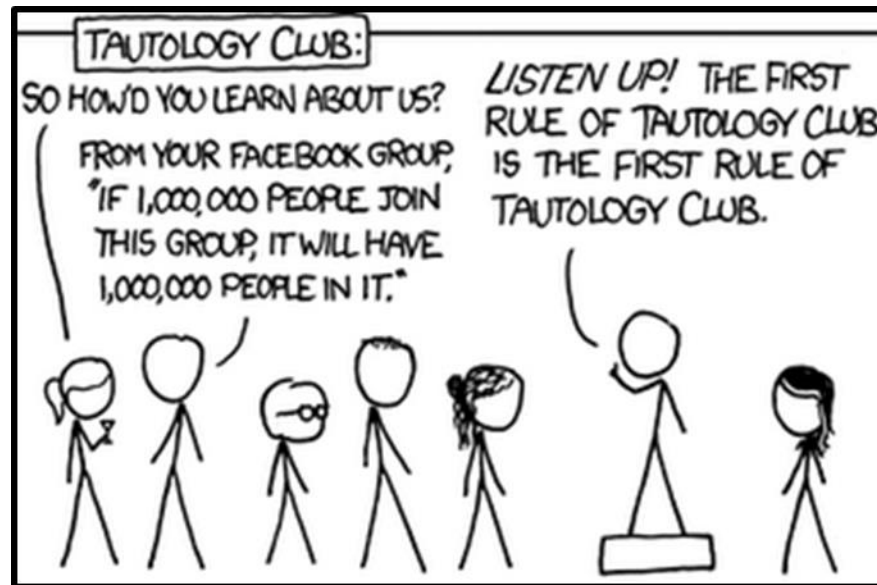


# CSE 311: Foundations of Computing

---

## Lecture 3: Digital Circuits



# Review: Propositional Logic

---

## Propositions

- atomic propositions are T/F-valued variables
- combined using logical connectives (not, and, or, etc.)
- can be described by a truth table
  - shows the truth value of the proposition in each combination of truth values of the atomic propositions

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

## Logical equivalence

- used to simplify logical expressions

## First application

- Simplifying English sentences

# Truth Table to show tautology

---

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

$$(p \wedge r) \rightarrow (r \vee p) \equiv \mathbf{T}$$

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

# Logical Proofs of Equivalence

---

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

Use a series of equivalences like so:

$$\begin{aligned}(p \wedge r) \rightarrow (r \vee p) &\equiv \neg(p \wedge r) \vee (r \vee p) \\ &\equiv (\neg p \vee \neg r) \vee (r \vee p) \\ &\equiv \neg p \vee (\neg r \vee (r \vee p)) \\ &\equiv \neg p \vee ((\neg r \vee r) \vee p) \\ &\equiv \neg p \vee (p \vee (\neg r \vee r)) \\ &\equiv (\neg p \vee p) \vee (\neg r \vee r) \\ &\equiv (p \vee \neg p) \vee (r \vee \neg r) \\ &\equiv \mathbf{T} \vee \mathbf{T} \\ &\equiv \mathbf{T}\end{aligned}$$

Law of Implication

De Morgan

Associative

Associative

Commutative

Associative

Commutative (twice)

Negation (twice)

Domination/Identity

## Identity

- $p \wedge \mathbf{T} \equiv p$
- $p \vee \mathbf{F} \equiv p$

## Domination

- $p \vee \mathbf{T} \equiv \mathbf{T}$
- $p \wedge \mathbf{F} \equiv \mathbf{F}$

## Idempotent

- $p \vee p \equiv p$
- $p \wedge p \equiv p$

## Commutative

- $p \vee q \equiv q \vee p$
- $p \wedge q \equiv q \wedge p$

# Logical Proofs of Equivalence/Tautology

---

- Not smaller than truth tables when there are only a few propositional variables...
- ...but usually ***much shorter*** than truth table proofs when there are many propositional variables
- A big advantage will be that we can extend them to a more in-depth understanding of logic for which truth tables don't apply.

# Another key 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)

# Circuits: AND, OR, NOT Gates

---

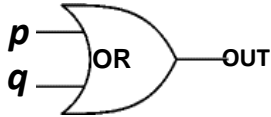
## AND Gate



$p$	$q$	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

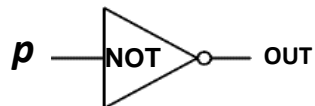
## OR Gate



$p$	$q$	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

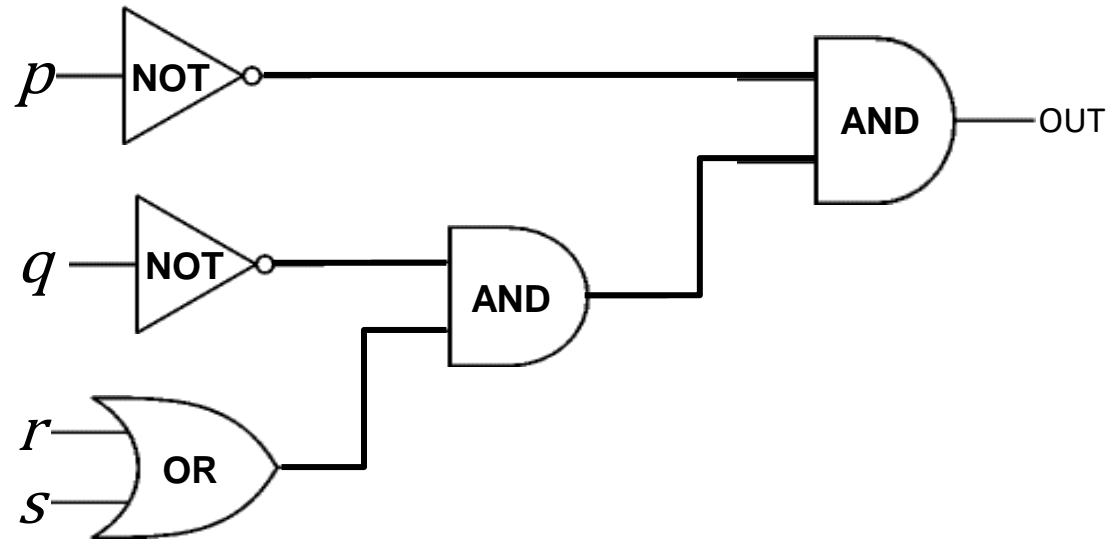


$p$	OUT
1	0
0	1

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

# Combinational Logic Circuits

---

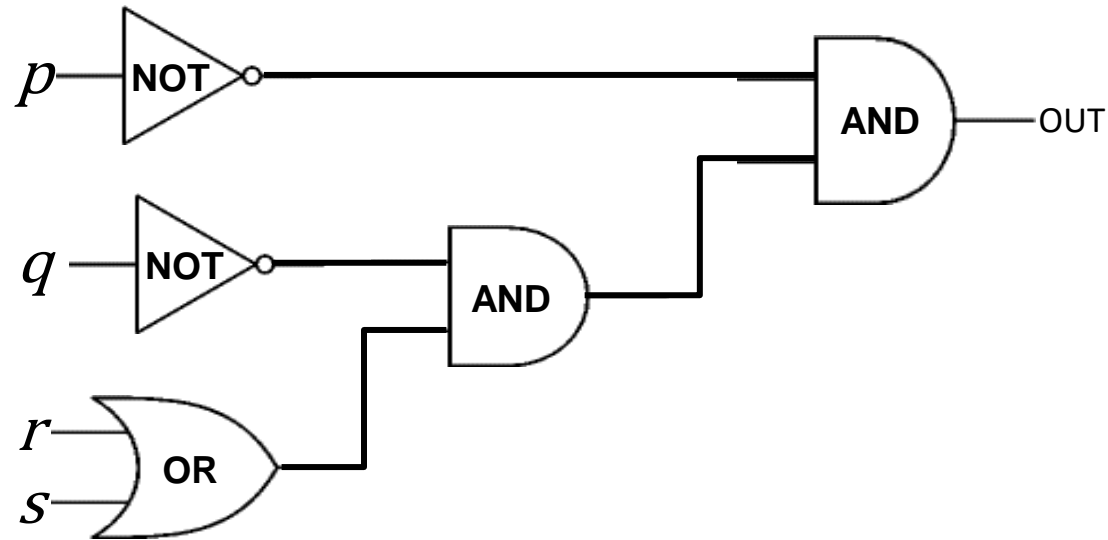


**Values get sent along wires connecting gates**



# Combinational Logic Circuits

---

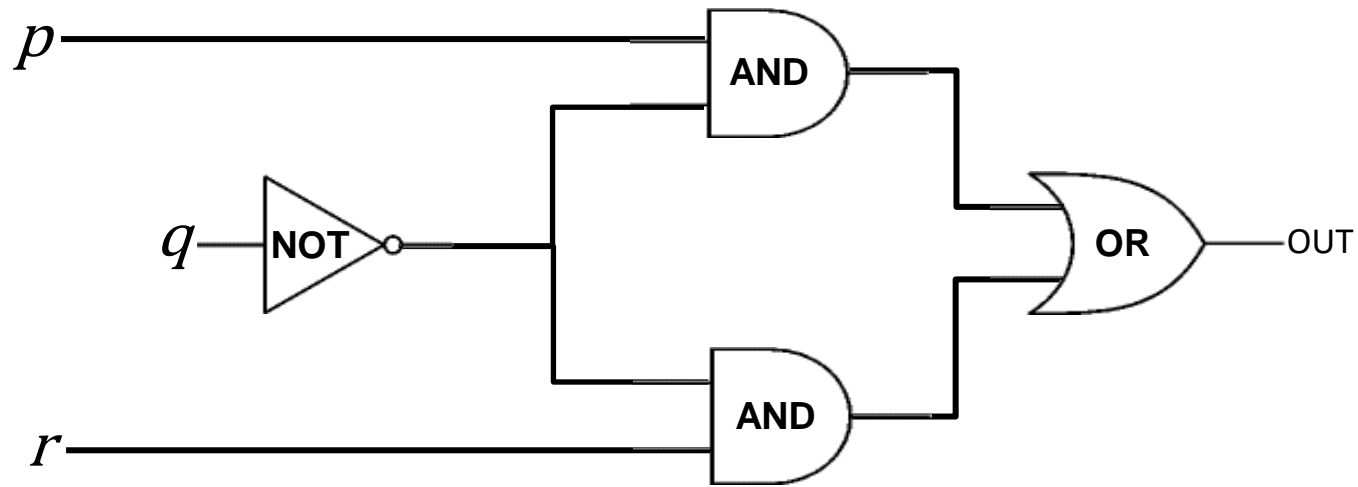


**Values get sent along wires connecting gates**

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

# Combinational Logic Circuits

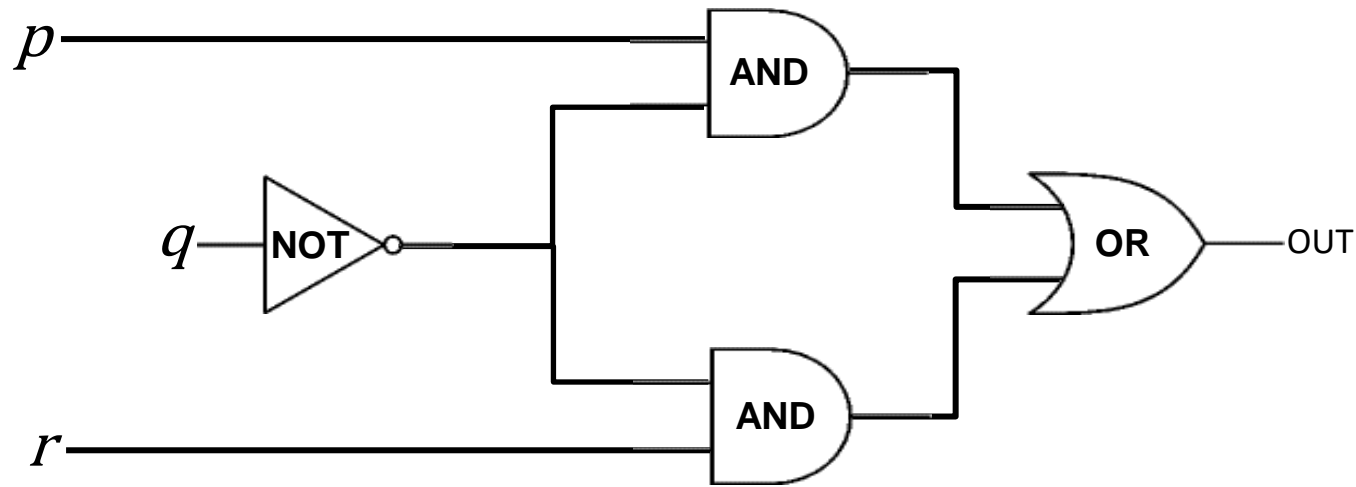
---



**Wires can send one value to multiple gates!**

# Combinational Logic Circuits

---



**Wires can send one value to multiple gates!**

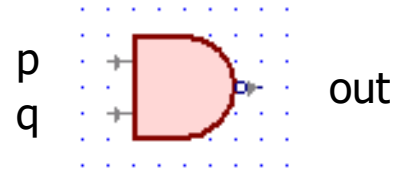
$$(p \wedge \neg q) \vee (\neg q \wedge r)$$

# Other Useful Gates

---

## NAND

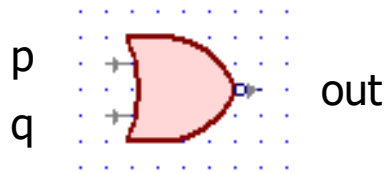
$$\neg(p \wedge q)$$



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

## NOR

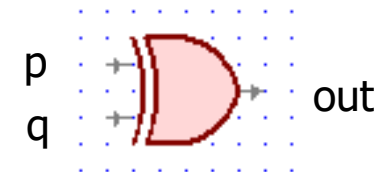
$$\neg(p \vee q)$$



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

## XOR

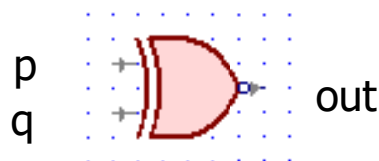
$$p \oplus q$$



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 Logic

---

## Combinational Logic

– output =  $F(\text{input})$

## Sequential Logic

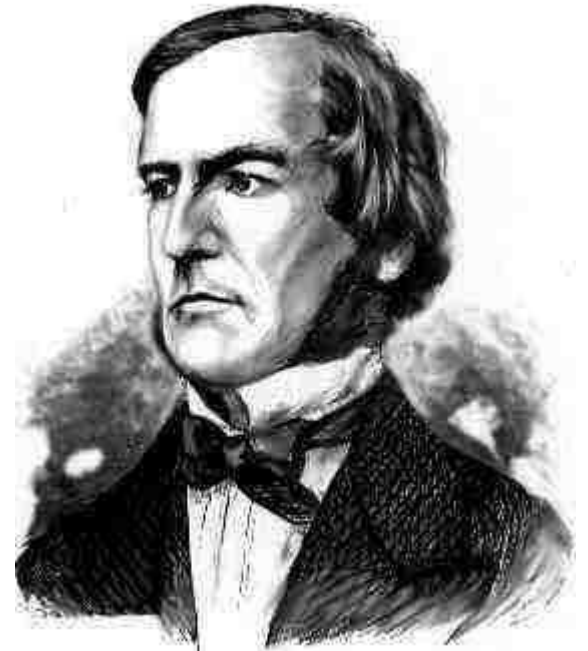
- $\text{output}_t = F(\text{output}_{t-1}, \text{input}_t)$ 
  - output dependent on history
  - concept of a time step (clock,  $t$ )
- Covered in CSE 369

# Boolean Logic

---

## Combinational Logic

– output =  $F(\text{input})$



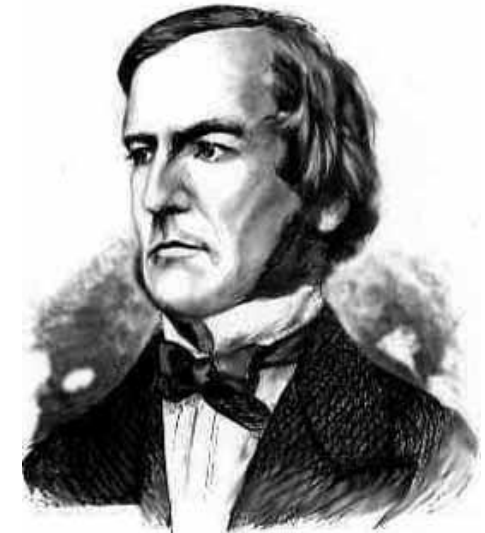
## Boolean Algebra: another notation for logic consisting of...

- a set of elements  $B = \{0, 1\}$
- binary operations  $\{ + , \cdot \}$  (OR, AND)
- and a unary operation  $\{ ' \}$  (NOT)

# Boolean Algebra

---

- Usual notation used in circuit design
- Boolean algebra
  - a set of elements  $B$  containing  $\{0, 1\}$
  - binary operations  $\{ + , \cdot \}$
  - and a unary operation  $\{ ' \}$
  - such that the following axioms hold:



For any  $a, b, c$  in  $B$ :

1. closure:

$$a + b \text{ is in } B$$

$$a \cdot b \text{ is in } B$$

2. commutativity:

$$a + b = b + a$$

$$a \cdot b = b \cdot a$$

3. associativity:

$$a + (b + c) = (a + b) + c$$

$$a \cdot (b \cdot c) = (a \cdot b) \cdot c$$

4. distributivity:

$$a + (b \cdot c) = (a + b) \cdot (a + c)$$

$$a \cdot (b + c) = (a \cdot b) + (a \cdot c)$$

5. identity:

$$a + 0 = a$$

$$a \cdot 1 = a$$

6. complementarity:

$$a + a' = 1$$

$$a \cdot a' = 0$$

7. null:

$$a + 1 = 1$$

$$a \cdot 0 = 0$$

8. idempotency:

$$a + a = a$$

$$a \cdot a = a$$

9. involution:

$$(a')' = a$$

# Proving Theorems

---

## Using truth table:

For example, de Morgan's Law:

$(X + Y)' = X' \cdot Y'$   
NOR is equivalent to AND  
with inputs complemented

X	Y	X'	Y'	$(X + Y)'$	$X' \cdot Y'$
0	0	1	1	1	1
0	1	1	0	0	0
1	0	0	1	0	0
1	1	0	0	0	0

$(X \cdot Y)' = X' + Y'$   
NAND is equivalent to OR  
with inputs complemented

X	Y	X'	Y'	$(X \cdot Y)'$	$X' + Y'$
0	0	1	1	1	1
0	1	1	0	1	1
1	0	0	1	1	1
1	1	0	0	0	0

More generally  $(a + b + c + \dots)' = a' \cdot b' \cdot c' \cdot \dots$

$(a \cdot b \cdot c \cdot \dots)' = a' + b' + c' + \dots$



# Proving Theorems

---

- 2. commutativity:
- 3. associativity:
- 4. distributivity:
- 5. identity:
- 6. complementarity:
- 7. null:
- 8. idempotency:
- 9. involution:

$$\begin{aligned}a + b &= b + a \\a + (b + c) &= (a + b) + c \\a + (b \cdot c) &= (a + b) \cdot (a + c) \\a + 0 &= a \\a + a' &= 1 \\a + 1 &= 1 \\a + a &= a \\(a')' &= a\end{aligned}$$

$$\begin{aligned}a \cdot b &= b \cdot a \\a \cdot (b \cdot c) &= (a \cdot b) \cdot c \\a \cdot (b + c) &= (a \cdot b) + (a \cdot c) \\a \cdot 1 &= a \\a \cdot a' &= 0 \\a \cdot 0 &= 0 \\a \cdot a &= a\end{aligned}$$

## Using the laws of Boolean Algebra:

prove the "Uniting theorem":

$$X \cdot Y + X \cdot Y' = X$$

$$X \cdot Y + X \cdot Y' =$$

prove the "Absorption theorem":

$$X + X \cdot Y = X$$

$$X + X \cdot Y =$$

# Proving Theorems

---

2. commutativity:
3. associativity:
4. distributivity:
5. identity:
6. complementarity:
7. null:
8. idempotency:
9. involution:

$$\begin{aligned}a + b &= b + a \\a + (b + c) &= (a + b) + c \\a + (b \cdot c) &= (a + b) \cdot (a + c) \\a + 0 &= a \\a + a' &= 1 \\a + 1 &= 1 \\a + a &= a \\(a')' &= a\end{aligned}$$

$$\begin{aligned}a \cdot b &= b \cdot a \\a \cdot (b \cdot c) &= (a \cdot b) \cdot c \\a \cdot (b + c) &= (a \cdot b) + (a \cdot c) \\a \cdot 1 &= a \\a \cdot a' &= 0 \\a \cdot 0 &= 0 \\a \cdot a &= a\end{aligned}$$

## Using the laws of Boolean Algebra:

prove the "Uniting theorem":

$$X \cdot Y + X \cdot Y' = X$$

distributivity

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

complementarity

$$= X \cdot 1$$

identity

$$= X$$

prove the "Absorption theorem":

$$X + X \cdot Y = X$$

identity

$$X + X \cdot Y = X \cdot 1 + X \cdot Y$$

distributivity

$$= X \cdot (1 + Y)$$

commutativity

$$= X \cdot (Y + 1)$$

null

$$= X \cdot 1$$

identity

$$= X$$

# 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

---

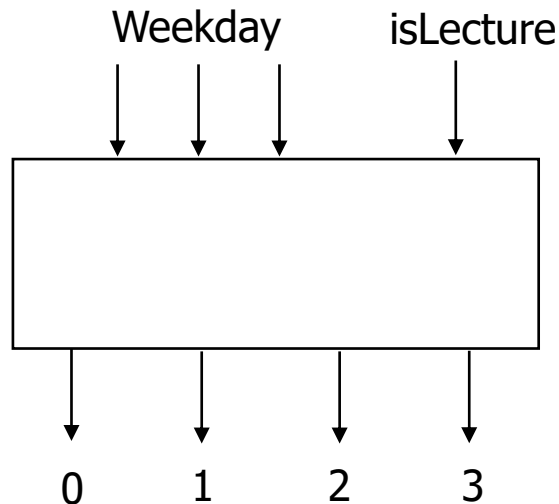
```
public int classesLeftInMorning(int weekday, boolean isLecture) {
    switch (weekday) {
        case SUNDAY:
        case MONDAY:
            return isLecture ? 3 : 1;
        case TUESDAY:
        case WEDNESDAY:
            return isLecture ? 2 : 1;
        case THURSDAY:
            return isLecture ? 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 <sub>0</sub>	c <sub>1</sub>	c <sub>2</sub>	c <sub>3</sub>
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 <sub>0</sub>	c <sub>1</sub>	c <sub>2</sub>	c <sub>3</sub>
SUN	000	0	1	0	0
SUN	000	1	0	0	1
MON	001	0	1	0	0
MON	001	1	0	0	1
TUE	010	0	1	0	0
TUE	010	1	0	1	0
WED	011	0	1	0	0
WED	011	1	0	1	0
THU	100	-	0	1	0
FRI	101	0	1	0	0
FRI	101	1	0	1	0
SAT	110	-	1	0	0
-	111	-	1	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

$d_2 == 0 \ \&\& \ d_1 == 0 \ \&\& \ d_0 == 0 \ \&\& \ L == 1$

$d_2 == 0 \ \&\& \ d_1 == 0 \ \&\& \ d_0 == 1 \ \&\& \ L == 1$

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



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



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

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)

Now, we do  $c_1$ :

	$d_2d_1d_0$	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

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

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

# 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	???
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  $c_1$ :

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

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

# 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  $c_1$ :

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  $c_1$ :

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

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

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

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

# 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  $c_0$ :

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

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

$$d_2 \cdot d_1 \cdot d_0$$

# Truth Table to Logic (Part 4)

---

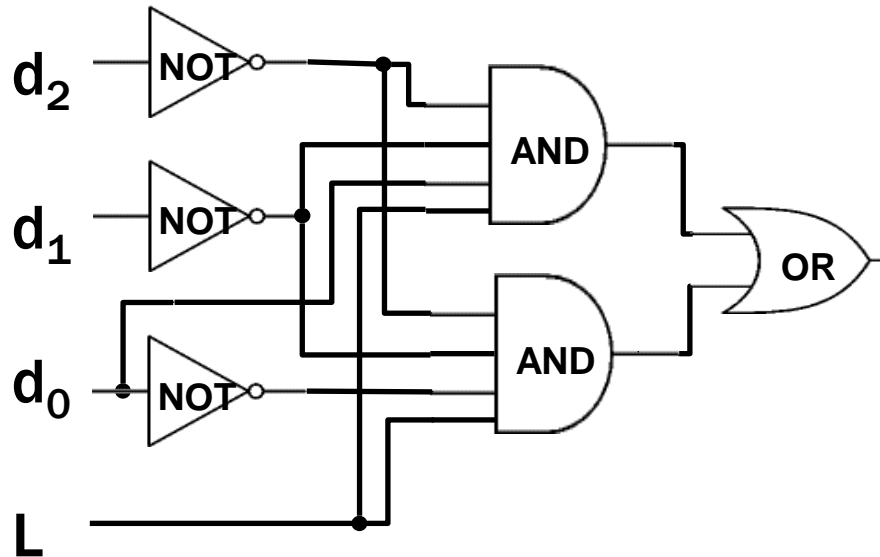
$$c_0 = d_2 \cdot d_1' \cdot d_0 \cdot L' + d_2 \cdot d_1 \cdot d_0' + d_2 \cdot d_1 \cdot d_0$$

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

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

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

Here's  $c_3$  as a circuit:



# Simplifying using Boolean Algebra

---

$$\begin{aligned}c3 &= d2' \cdot d1' \cdot d0' \cdot L + d2' \cdot d1' \cdot d0 \cdot L \\ &= d2' \cdot d1' \cdot (d0' + d0) \cdot L \\ &= d2' \cdot d1' \cdot 1 \cdot L \\ &= d2' \cdot d1' \cdot L\end{aligned}$$

