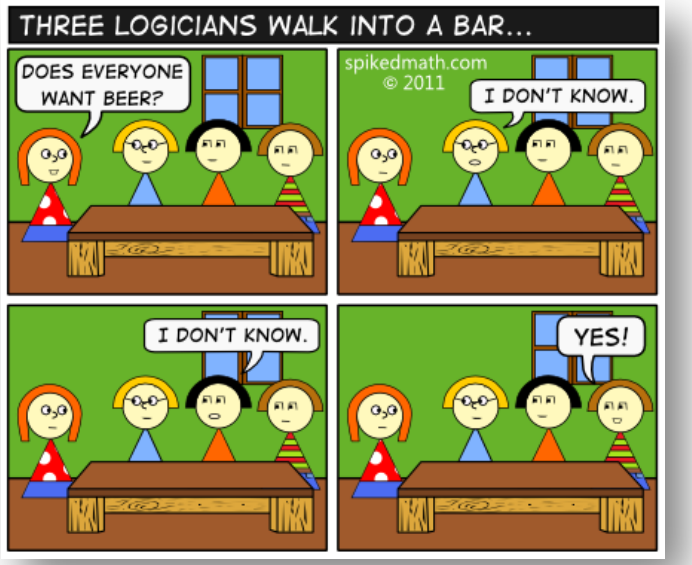
CSE 311




# Foundations of Computing I

Fall 2014

## Remember, the site is...

http://tinyurl.com/ynlecture

**Feel free to get started on the purple handout!**

# Public Service Announcement

**Do any of these sound familiar?**

"I feel like a fake"

"My classmates/professors etc. are going to find out I don't really belong here"

"Admissions made a mistake"

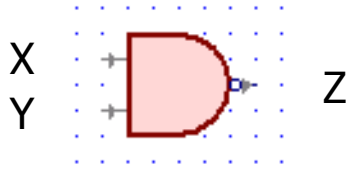
**This phenomenon is called *Imposter Syndrome***

**A lot of people have it! It's normal!**

# More Gates!

**NAND**

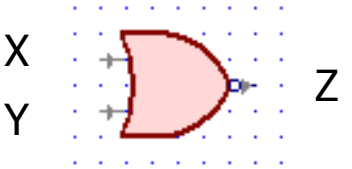
$\neg(X \wedge Y)$




| X | Y | Z |
|---|---|---|
| 0 | 0 | 1 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

**NOR**

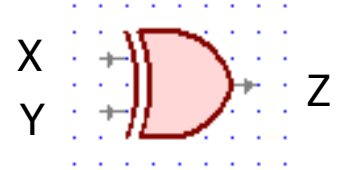
$\neg(X \vee Y)$




| X | Y | Z |
|---|---|---|
| 0 | 0 | 1 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 0 |

**XOR**

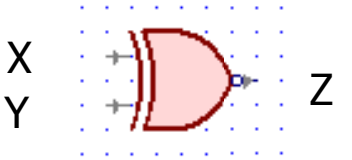
$X \oplus Y$




| 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

$p \vee \neg p$ Tautology!

$p \oplus p$ Contradiction!

$(p \rightarrow q) \wedge p$ Contingency (note: in lecture the and was an or)!

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

# Review: Logical Equivalence

*A* and *B* are *logically equivalent* if and only if
*A* ↔ *B* is a tautology

*i.e.* *A* and *B* have the same truth table

The notation *A* ≡ *B* denotes *A* and *B* are logically equivalent

**Example:** $p \equiv \neg\neg p$

| $p$ | $\neg p$ | $\neg\neg p$ | $p \leftrightarrow \neg\neg p$ |
|---|---|---|---|
| T | F | T | T |
| F | T | F | T |

# Review: De Morgan's laws

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

```
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);
}
```

This code inserts *value* into a sorted linked list.

The first if runs when…front is null or value is smaller than the first item.

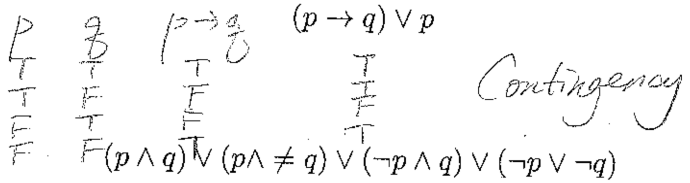
The while loop stops when…we've reached the end of the list or the next value is bigger.

# Review: 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 |

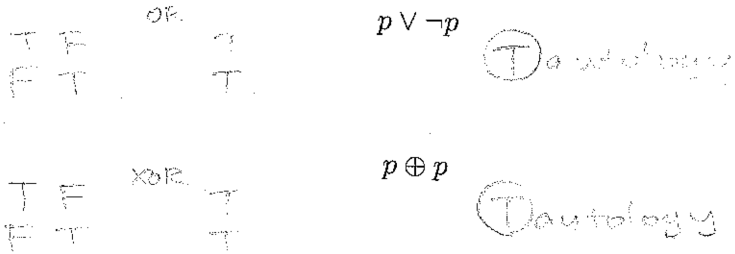
# Exercise Review




$(p \to q) \vee p$

$\neg p \vee q \vee p \equiv t \vee q \equiv q$ contingency

$p \oplus p$ Tautology




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

# Exercise Review

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

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

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

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

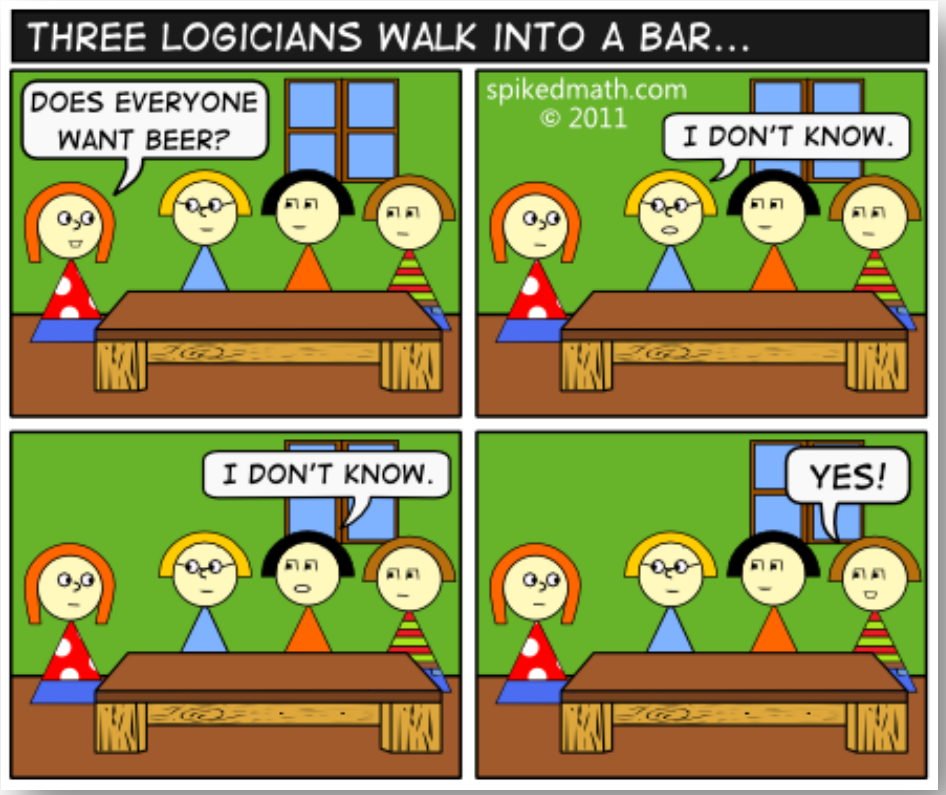
$\neg q \wedge (p \vee r)$

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

# CSE 311: Foundations of Computing

## Fall 2014

## Lecture 3: Logic and Boolean algebra



# 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**)

# 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

# Some Equivalences Related to Implication

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

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

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

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

# Some Equivalences Related to Implication

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

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

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

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

# Logical Proofs

## To show P is equivalent to Q

– Apply a series of logical equivalences to sub-expressions to convert P to Q

## To show P is a tautology

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

# Prove this is a Tautology

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

| | | |
|---|---|---|
| $(p \wedge q) \rightarrow (p \vee q) \equiv \neg(p \wedge q) \vee (p \vee q)$ | | Law of Implication |
| $\equiv (\neg p \vee \neg q) \vee (p \vee q)$ | | DeMorgan |
| $\equiv \neg p \vee (\neg q \vee p) \vee q$ | | Associativity |
| $\equiv \neg p \vee (p \vee \neg q) \vee q$ | | Commutativity |
| $\equiv (\neg p \vee p) \vee (\neg q \vee q)$ | | Associativity |
| $\equiv (\neg p \vee p) \vee (q \vee \neg q)$ | | Commutativity |
| $\equiv T \vee T$ | | Negation |
| $\equiv T$ | | Idempotency |

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

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

| | | |
|---|---|---|
| $(p \wedge (p \rightarrow q)) \rightarrow q$ | $\equiv \neg(p \wedge (p \rightarrow q)) \vee q$ | Law of Implication |
| | $\equiv (\neg p \vee \neg(p \rightarrow q)) \vee q$ | DeMorgan |
| | $\equiv (\neg p \vee \neg(\neg p \vee q)) \vee q$ | Law of Implication |
| | $\equiv (\neg p \vee (p \wedge \neg q)) \vee q$ | DeMorgan |
| | $\equiv ((\neg p \vee p) \wedge (\neg p \vee \neg q)) \vee q$ | Distributivity |
| | $\equiv ((p \vee \neg p) \wedge (\neg p \vee \neg q)) \vee q$ | Commutativity |
| | $\equiv (\mathrm{T} \wedge (\neg p \vee \neg q)) \vee q$ | Negation |
| | $\equiv ((\neg p \vee \neg q) \wedge \mathrm{T}) \vee q$ | Commutativity |
| | $\equiv (\neg p \vee \neg q) \vee q$ | Identity |
| | $\equiv \neg p \vee (\neg q \vee q)$ | Associativity |
| | $\equiv \neg p \vee (q \vee \neg q)$ | Commutativity |
| | $\equiv \neg p \vee \mathrm{T}$ | Negation |
| | $\equiv \mathrm{T}$ | Domination |

# Prove these are not equivalent

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

**When p is F, q is F, and r is F…**

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

# 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 )

# 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**

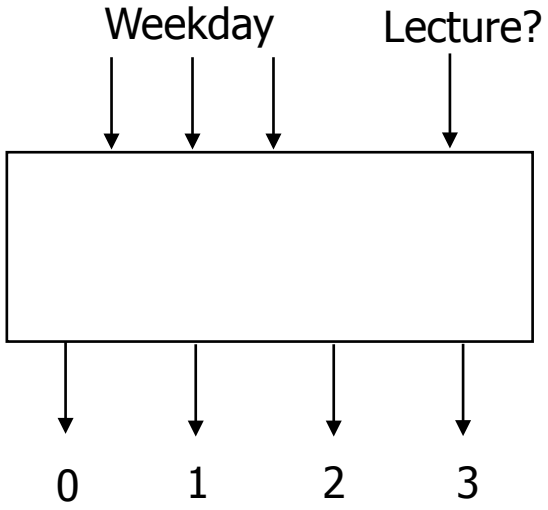
# Implementation in Software

```
public int classesLeftInMorning(weekday, lecture_flag) {
    switch (day) {
        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!

```
public int classesLeftInMorning(weekday, lecture_flag) {
    switch (day) {
        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;
   }
}
```

| 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!

| 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 | $(110)_2$ |

| Weekday | Lecture? | c0 | c1 | c2 | c3 |
|---|---|---|---|---|---|
| 000 | 0 | 0 | 1 | 0 | 0 |
| 000 | 1 | 0 | 0 | 0 | 1 |
| 001 | 0 | 0 | 1 | 0 | 0 |
| 001 | 1 | 0 | 0 | 0 | 1 |
| 010 | 0 | 0 | 1 | 0 | 0 |
| 010 | 1 | 0 | 0 | 1 | 0 |
| 011 | 0 | 0 | 1 | 0 | 0 |
| 011 | 1 | 0 | 0 | 1 | 0 |
| 100 | - | 0 | 1 | 0 | 0 |
| 101 | 0 | 1 | 0 | 0 | 0 |
| 101 | 1 | 0 | 1 | 0 | 0 |
| 110 | - | 1 | 0 | 0 | 0 |
| 111 | - | - | - | - | - |

# Truth Table to Logic (Part 1)

c3 = (DAY == SUN and LEC) or (DAY == MON and LEC)

c3 = (d2 == 0 && d1 == 0 && d0 == 0 && L == 1) ||
(d2 == 0 && d1 == 0 && d0 == 1 && L == 1)

c3 = d2'•d1'•d0'•L + d2'•d1'•d0•L

| DAY | d2 d1 d0 | L | c0 | c1 | c2 | c3 |
|---|---|---|---|---|---|---|
| SunS | 000 | 0 | 0 | 1 | 0 | 0 |
| SunL | 000 | 1 | 0 | 0 | 0 | 1 |
| MonS | 001 | 0 | 0 | 1 | 0 | 0 |
| MonL | 001 | 1 | 0 | 0 | 0 | 1 |
| TueS | 010 | 0 | 0 | 1 | 0 | 0 |
| TueL | 010 | 1 | 0 | 0 | 1 | 0 |
| WedS | 011 | 0 | 0 | 1 | 0 | 0 |
| WedL | 011 | 1 | 0 | 0 | 1 | 0 |
| Thu | 100 | - | 0 | 1 | 0 | 0 |
| FriS | 101 | 0 | 1 | 0 | 0 | 0 |
| FriL | 101 | 1 | 0 | 1 | 0 | 0 |
| Sat | 110 | - | 1 | 0 | 0 | 0 |
| - | 111 | - | - | - | - | - |

# Truth Table to Logic (Part 2)

$c3 = d2' \cdot d1' \cdot d0' \cdot L + d2' \cdot d1' \cdot d0 \cdot L$

c2 = (DAY == TUE and LEC) or
(DAY == WED and LEC)

$c2 = d2' \cdot d1 \cdot d0' \cdot L + d2' \cdot d1 \cdot d0 \cdot L$

| DAY | d2 d1 d0 | L | c0 | c1 | c2 | c3 |
|---|---|---|---|---|---|---|
| SunS | 000 | 0 | 0 | 1 | 0 | 0 |
| SunL | 000 | 1 | 0 | 0 | 0 | 1 |
| MonS | 001 | 0 | 0 | 1 | 0 | 0 |
| MonL | 001 | 1 | 0 | 0 | 0 | 1 |
| TueS | 010 | 0 | 0 | 1 | 0 | 0 |
| TueL | 010 | 1 | 0 | 0 | 1 | 0 |
| WedS | 011 | 0 | 0 | 1 | 0 | 0 |
| WedL | 011 | 1 | 0 | 0 | 1 | 0 |
| Thu | 100 | - | 0 | 1 | 0 | 0 |
| FriS | 101 | 0 | 1 | 0 | 0 | 0 |
| FriL | 101 | 1 | 0 | 1 | 0 | 0 |
| Sat | 110 | - | 1 | 0 | 0 | 0 |
| - | 111 | - | - | - | - | - |

# Truth Table to Logic (Part 3)

$c3 = d2' \cdot d1' \cdot d0' \cdot L + d2' \cdot d1' \cdot d0 \cdot L$

$c2 = d2' \cdot d1 \cdot d0' \cdot L + d2' \cdot d1 \cdot d0 \cdot L$

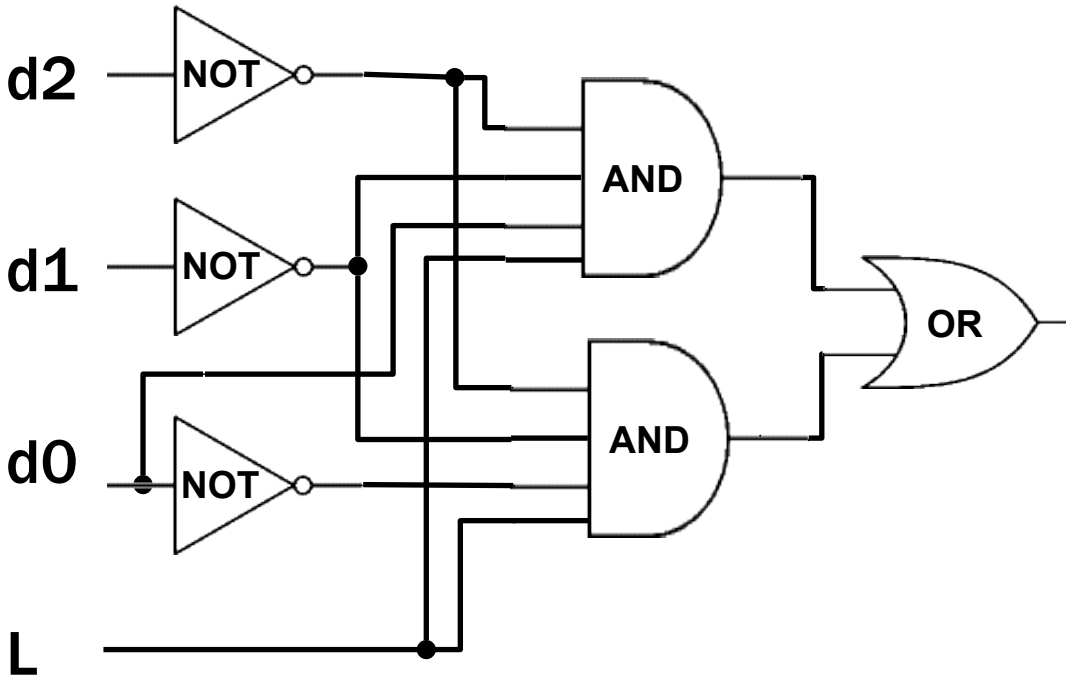
c1 = On your homework for next week!

$c0 = d2 \cdot d1' \cdot d0 \cdot L' + d2 \cdot d1 \cdot d0'$

| DAY | d2 d1 d0 | L | c0 | c1 | c2 | c3 |
|---|---|---|---|---|---|---|
| SunS | 000 | 0 | 0 | 1 | 0 | 0 |
| SunL | 000 | 1 | 0 | 0 | 0 | 1 |
| MonS | 001 | 0 | 0 | 1 | 0 | 0 |
| MonL | 001 | 1 | 0 | 0 | 0 | 1 |
| TueS | 010 | 0 | 0 | 1 | 0 | 0 |
| TueL | 010 | 1 | 0 | 0 | 1 | 0 |
| WedS | 011 | 0 | 0 | 1 | 0 | 0 |
| WedL | 011 | 1 | 0 | 0 | 1 | 0 |
| Thu | 100 | - | 0 | 1 | 0 | 0 |
| FriS | 101 | 0 | 1 | 0 | 0 | 0 |
| FriL | 101 | 1 | 0 | 1 | 0 | 0 |
| Sat | 110 | - | 1 | 0 | 0 | 0 |
| - | 111 | - | - | - | - | - |

# Logic to Gates

$c3 = d2' \cdot d1' \cdot d0' \cdot L + d2' \cdot d1' \cdot d0 \cdot L$




Multi-input AND gates

| DAY | d2 d1 d0 | L | c0 | c1 | c2 | c3 |
|---|---|---|---|---|---|---|
| SunS | 000 | 0 | 0 | 1 | 0 | 0 |
| SunL | 000 | 1 | 0 | 0 | 0 | 1 |
| MonS | 001 | 0 | 0 | 1 | 0 | 0 |
| MonL | 001 | 1 | 0 | 0 | 0 | 1 |
| TueS | 010 | 0 | 0 | 1 | 0 | 0 |
| TueL | 010 | 1 | 0 | 0 | 1 | 0 |
| WedS | 011 | 0 | 0 | 1 | 0 | 0 |
| WedL | 011 | 1 | 0 | 0 | 1 | 0 |
| Thu | 100 | - | 0 | 1 | 0 | 0 |
| FriS | 101 | 0 | 1 | 0 | 0 | 0 |
| FriL | 101 | 1 | 0 | 1 | 0 | 0 |
| Sat | 110 | - | 1 | 0 | 0 | 0 |
| - | 111 | - | - | - | - | - |

# Combinational Logic

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