slightly more stuff

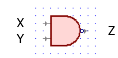
Homework #1: Posted now! Due next Friday **before** class.
Gradescope!

If I say "as you probably know…"
or "as you can clearly see…"
or "as was obious to even the most primitive of humans…"
or "as everyone learned in high school…"

you can basically just ignore me

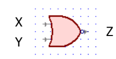
more gates

NAND
$\neg(X \wedge Y)$



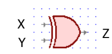
X	Y	Z
0	0	1
0	1	1
1	0	1
1	1	0

NOR
$\neg(X \vee Y)$



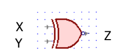
X	Y	Z
0	0	1
0	1	0
1	0	0
1	1	0

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!

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

logical equivalence

A and *B* are *logically equivalent* if and only if
A $\leftrightarrow$ *B* is a tautology
i.e. *A* and *B* have the same truth table

The notation $A \equiv B$ 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

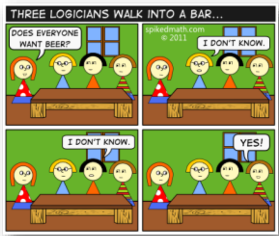
$$(p \rightarrow q) \equiv (\neg p \vee q)$$

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

cse 311: foundations of computing

Spring 2015

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)
  - $p \lor q \equiv q \lor p$
  - $p \land q \equiv q \land p$
- $x \cdot (y + z) = x \cdot y + x \cdot z$ (distributivity)
  - $p \land (q \lor r) \equiv (p \land q) \lor (p \land r)$
  - $p \lor (q \land r) \equiv (p \lor q) \land (p \lor r)$
- $(x + y) + z = x + (y + z)$ (associativity)
  - $(p \lor q) \lor r \equiv p \lor (q \lor r)$
  - $(p \land q) \land r \equiv p \land (q \land r)$

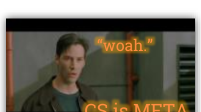
properties of logical connectives

You will always get this list.

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

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 / machine learning
  - Program verification



equivalences related to implication

$$p \rightarrow q \equiv \neg p \lor q$$
$$p \rightarrow q \equiv \neg q \rightarrow \neg p$$
$$p \leftrightarrow q \equiv (p \rightarrow q) \land (q \rightarrow p)$$
$$p \leftrightarrow q \equiv \neg p \leftrightarrow \neg q$$

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

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

prove this is a tautology

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

prove these are **not** equivalent

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

Boolean logic

Combinational Logic
- output = F(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)

George "homeopathy" Boole

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

a combinatorial 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

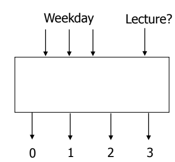
implementation in software

```
public int classesLeft (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 classesLeft (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 ⇒ logic (part one)

DAY	d2d1d0	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	-	-	-	-	-

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

truth table ⇒ logic (part two)

DAY	d2d1d0	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	-	-	-	-	-

c3 = d2'•d1'•d0'•L + d2'•d1'•d0•L

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

c2 = d2'•d1•d0'•L + d2'•d1•d0•L

truth table ⇒ logic (part three)

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

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

c1 =

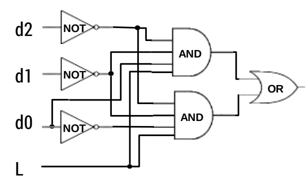
[you do this one]

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

DAY	d2d1d0	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 ⇒ gates

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



(multiple input AND gates)

[LEVEL UP]

DAY	d2d1d0	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	-	-	-	-	-