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 obvious to even the most primitive of humans..."
or "as everyone learned in high school..."

you can basically just ignore me
more gates

NAND
\[ \neg (X \land Y) \]

```
+---+---+---+
| X | Y | Z |
+---+---+---+
| 0 | 0 | 1 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |
```

NOR
\[ \neg (X \lor Y) \]

```
+---+---+---+
| 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 |
```
**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

\[ p \lor \neg p \quad \text{Tautology!} \]

\[ p \oplus p \quad \text{Contradiction!} \]

\[ (p \rightarrow q) \land p \quad \text{Contingency!} \]

\[ (p \land q) \lor (p \land \neg q) \lor (\neg p \land q) \lor (\neg p \land \neg q) \quad \text{Tautology!} \]
**logical equivalence**

*A* and *B* are *logically equivalent* if and only if

\[
A \iff B \text{ is a tautology}
\]

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

The notation \( A \equiv B \) denotes *A* and *B* are logically equivalent.

**Example:** \( p \equiv \neg \neg p \)

<table>
<thead>
<tr>
<th>( p )</th>
<th>( \neg p )</th>
<th>( \neg \neg p )</th>
<th>( p \iff \neg \neg p )</th>
</tr>
</thead>
<tbody>
<tr>
<td>T</td>
<td>F</td>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>F</td>
<td>T</td>
<td>F</td>
<td>T</td>
</tr>
</tbody>
</table>
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

\[(p \rightarrow q) \equiv (\neg p \lor q)\]

<table>
<thead>
<tr>
<th>(p)</th>
<th>(q)</th>
<th>(p \rightarrow q)</th>
<th>(\neg p)</th>
<th>(\neg p \lor q)</th>
<th>((p \rightarrow q) \leftrightarrow (\neg p \lor q))</th>
</tr>
</thead>
<tbody>
<tr>
<td>T</td>
<td>T</td>
<td>T</td>
<td>F</td>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>T</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>T</td>
</tr>
<tr>
<td>F</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>F</td>
<td>F</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
</tr>
</tbody>
</table>
Lecture 3: Logic and Boolean algebra
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

- \( x + y = y + x \)  
  \(- p \lor q \equiv q \lor p \)
  \(- p \land q \equiv q \land p \)

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

- \( (x + y) + z = x + (y + z) \)  
  \(- (p \lor q) \lor r \equiv p \lor (q \lor r) \)
  \(- (p \land q) \land r \equiv p \land (q \land r) \)

(commutativity)

(distributivity)

(associativity)
properties of logical connectives

- **Identity**
  - \( p \land T \equiv p \)
  - \( p \lor F \equiv p \)

- **Domination**
  - \( p \lor T \equiv T \)
  - \( p \land F \equiv F \)

- **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**
  - \( p \lor \neg p \equiv T \)
  - \( p \land \neg p \equiv F \)

You will always get this list.
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

“woah.”

"CS is META"
equivalences related to implication

\[ p \to q \equiv \neg p \lor q \]

\[ p \to q \equiv \neg q \to \neg p \]

\[ p \leftrightarrow q \equiv (p \to q) \land (q \to p) \]

\[ p \leftrightarrow q \equiv \neg p \leftrightarrow \neg q \]
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

\[(p \land q) \rightarrow (p \lor q)\]

\[\equiv \neg (p \land q) \lor (p \lor q)\]

\[\equiv (\neg p \lor \neg q) \lor (p \lor q)\]

\[\equiv (\neg p \lor \neg q) \lor (\neg q \lor q)\]

\[\equiv \top \lor (\neg q \lor q)\]

\[\equiv \top\]
prove this is a tautology

\[(p \land (p \to q)) \to q\]

\[
\equiv \neg (p \land (p \to q)) \to q
\]

\[
\equiv \neg p \lor \neg (p \to q) \lor q
\]

\[
\equiv \neg p \lor \neg \neg (p \lor q) \lor q
\]

\[
\equiv \neg p \lor (p \lor q) \lor q
\]

\[
\equiv \neg p \lor (p \lor q) \lor q
\]

\[
\equiv \neg p \lor (p \lor q) \lor q
\]
prove these are equivalent

\[(p \rightarrow q) \rightarrow r\]

\[p \rightarrow (q \rightarrow r)\]

No!

(next slide)
prove these are **not** equivalent

\[(p \rightarrow q) \rightarrow r\]  \hspace{1cm}  \[p \rightarrow (q \rightarrow r)\]

Consider 
\[p = q = r = F\]

\[F \rightarrow T\]

\[\text{different truth values}\]
**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** consisting of...
- a set of elements B = \{0, 1\}
- binary operations \{ +, \cdot \} (OR, AND)
- and a unary operation \{'\} (NOT)
Sessons 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
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

```java
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;
    }
}
```

<table>
<thead>
<tr>
<th>Weekday</th>
<th>Number</th>
<th>Binary</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sunday</td>
<td>0</td>
<td>(000)₂</td>
</tr>
<tr>
<td>Monday</td>
<td>1</td>
<td>(001)₂</td>
</tr>
<tr>
<td>Tuesday</td>
<td>2</td>
<td>(010)₂</td>
</tr>
<tr>
<td>Wednesday</td>
<td>3</td>
<td>(011)₂</td>
</tr>
<tr>
<td>Thursday</td>
<td>4</td>
<td>(100)₂</td>
</tr>
<tr>
<td>Friday</td>
<td>5</td>
<td>(101)₂</td>
</tr>
<tr>
<td>Saturday</td>
<td>6</td>
<td>(110)₂</td>
</tr>
</tbody>
</table>
## Converting to a Truth Table

<table>
<thead>
<tr>
<th>Weekday</th>
<th>Number</th>
<th>Binary</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sunday</td>
<td>0</td>
<td>(000)$_2$</td>
</tr>
<tr>
<td>Monday</td>
<td>1</td>
<td>(001)$_2$</td>
</tr>
<tr>
<td>Tuesday</td>
<td>2</td>
<td>(010)$_2$</td>
</tr>
<tr>
<td>Wednesday</td>
<td>3</td>
<td>(011)$_2$</td>
</tr>
<tr>
<td>Thursday</td>
<td>4</td>
<td>(100)$_2$</td>
</tr>
<tr>
<td>Friday</td>
<td>5</td>
<td>(101)$_2$</td>
</tr>
<tr>
<td>Saturday</td>
<td>6</td>
<td>(110)$_2$</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Weekday</th>
<th>Lecture?</th>
<th>c0</th>
<th>c1</th>
<th>c2</th>
<th>c3</th>
</tr>
</thead>
<tbody>
<tr>
<td>000</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>000</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>001</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>001</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>010</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>010</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>011</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>011</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>100</td>
<td>-</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>101</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>101</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>110</td>
<td>-</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>111</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>
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'\cdot d1'\cdot d0'\cdot L + d2'\cdot d1'\cdot d0\cdot L
\[ c_3 = d_2' \cdot d_1' \cdot d_0' \cdot L + d_2' \cdot d_1' \cdot d_0 \cdot L \]

\[ c_2 = (\text{DAY} == \text{TUE} \text{ and LEC}) \text{ or } (\text{DAY} == \text{WED and LEC}) \]

\[ c_2 = d_2' \cdot d_1 \cdot d_0' \cdot L + d_2' \cdot d_1 \cdot d_0 \cdot L \]

<table>
<thead>
<tr>
<th>DAY</th>
<th>d_2d_1d_0</th>
<th>L</th>
<th>c_0</th>
<th>c_1</th>
<th>c_2</th>
<th>c_3</th>
</tr>
</thead>
<tbody>
<tr>
<td>SunS</td>
<td>000</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>SunL</td>
<td>000</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>MonS</td>
<td>001</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>MonL</td>
<td>001</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>TueS</td>
<td>010</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>TueL</td>
<td>010</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>WedS</td>
<td>011</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>WedL</td>
<td>011</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>Thu</td>
<td>100</td>
<td>-</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>FriS</td>
<td>101</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>FriL</td>
<td>101</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Sat</td>
<td>110</td>
<td>-</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>-</td>
<td>111</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>
### Truth Table ⇒ Logic (Part Three)

<table>
<thead>
<tr>
<th>DAY</th>
<th>d2d1d0</th>
<th>L</th>
<th>c0</th>
<th>c1</th>
<th>c2</th>
<th>c3</th>
</tr>
</thead>
<tbody>
<tr>
<td>SunS</td>
<td>000</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>SunL</td>
<td>000</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>MonS</td>
<td>001</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>MonL</td>
<td>001</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>TueS</td>
<td>010</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>TueL</td>
<td>010</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>WedS</td>
<td>011</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>WedL</td>
<td>011</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>Thu</td>
<td>100</td>
<td>-</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>FriS</td>
<td>101</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>FriL</td>
<td>101</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Sat</td>
<td>110</td>
<td>-</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>111</td>
<td>-</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

\[
c_3 = d_2' \cdot d_1' \cdot d_0' \cdot L + d_2' \cdot d_1' \cdot d_0 \cdot L
\]

\[
c_2 = d_2' \cdot d_1 \cdot d_0' \cdot L + d_2' \cdot d_1 \cdot d_0 \cdot L
\]

\[
c_1 = \text{[you do this one]}
\]

\[
c_0 = d_2 \cdot d_1' \cdot d_0 \cdot L' + d_2 \cdot d_1 \cdot d_0'
\]
c3 = d2'•d1'•d0'•L + d2'•d1'•d0•L

(multiple input AND gates)