**Homework #1** Due Today at 11:59pm

Your Gradescope account is created by your UW/CSE email address

Homework #2 will be posted today and it is due next Friday

**TA Office Hours**

<table>
<thead>
<tr>
<th>TA</th>
<th>Office hours</th>
<th>Room</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sam Castle</td>
<td>Wed, 12:00-1:00</td>
<td>CSE 021</td>
</tr>
<tr>
<td>Jiechen Chen</td>
<td>Tue, 4:00-5:00</td>
<td>CSE 218</td>
</tr>
<tr>
<td>Rebecca Leslie</td>
<td>Mon, 8:30-9:30</td>
<td>CSE 218</td>
</tr>
<tr>
<td>Evan McCarty</td>
<td>Tue, 11:30-12:30</td>
<td>CSE 220</td>
</tr>
<tr>
<td>Tim Oleskiw</td>
<td>Tue, 3:00-4:00</td>
<td>CSE 218</td>
</tr>
<tr>
<td>Spencer Peters</td>
<td>Tue, 1:00-2:00</td>
<td>CSE 218</td>
</tr>
<tr>
<td>Robert Weber</td>
<td>Wed, 3:30-4:30</td>
<td>CSE 678 (except Oct 21st at CSE 110)</td>
</tr>
<tr>
<td>Ian Zhu</td>
<td>Thu, 4:30-5:30</td>
<td>CSE 021</td>
</tr>
</tbody>
</table>
a 2-bit ripple-carry adder

1-Bit Adder

A
B
Cin
Sum

A
B
0
Cin
Sum

A
B
Cin
Sum

A
B
0
Cin
Sum

A
B
Cin
Sum
Fall 2015
Lecture 5: Canonical forms and predicate logic
mapping truth tables to logic gates

Given a truth table:

1. Write the Boolean expression
2. Minimize the Boolean expression
3. Draw as gates
4. Map to available gates

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>F</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

\[ F = A'B + AC \]

\[ F = A'BC' + A'BC + AB'C + ABC \]

\[ = A'B(C + C) + AC(B + B) \]

\[ = A'B + AC \]
• Truth table is the unique signature of a Boolean function

• The same truth table can have many gate realizations
  – we’ve seen this already
  – depends on how good we are at Boolean simplification

• **Canonical forms**
  – standard forms for a Boolean expression
  – we all come up with the same expression
• also known as **Disjunctive Normal Form (DNF)**
• also known as **minterm expansion**
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)

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
<th>minterms</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>A’B’C’</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>A’B’C</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>A’BC’</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>A’BC</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>AB’C’</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>AB’C</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>ABC’</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>ABC</td>
</tr>
</tbody>
</table>

F in canonical form:
F(A, B, C) = A’B’C + A’BC + AB’C + ABC’ + ABC

canonical form ≠ minimal form
F(A, B, C) = A’B’C + A’BC + AB’C + ABC + ABC’
= (A’B’ + A’B + AB’ + AB)C + ABC’
= ((A’ + A)(B’ + B))C + ABC’
= C + ABC’
= ABC’ + C
= AB + C
• Also known as **Conjunctive Normal Form (CNF)**
• Also known as **maxterm expansion**

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
<th>F</th>
<th>F'</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

\[
F = \begin{align*}
000 & \quad 010 & \quad 100 \\
(A + B + C) & \quad (A + B' + C) & \quad (A' + B + C)
\end{align*}
\]
Complement of function in sum-of-products form:

- \( F' = A'B'C' + A'BC' + AB'C' \)

Complement again and apply de Morgan’s and get the product-of-sums form:

- \((F')' = (A'B'C' + A'BC' + AB'C')'\)
- \(F = (A + B + C) (A + B' + C) (A' + B + C)\)
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)

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
<th>maxterms</th>
<th>F in canonical form:</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>A+B+C</td>
<td>F(A, B, C) = (A + B + C) (A + B’ + C) (A’ + B + C)</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>A+B+C’</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>A+B’+C</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>A+B’+C’</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>A’+B+C</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>A’+B+C’</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>A’+B’+C</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>A’+B’+C’</td>
<td></td>
</tr>
</tbody>
</table>

F in canonical form:
F(A, B, C) = (A + B + C) (A + B’ + C) (A’ + B + C)

canonical form ≠ minimal form

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

= (A + B + C) (A + B’ + C)

(A + B + C) (A’ + B + C)

= (A + C) (B + C)
• **Propositional Logic**
  – If Pikachu doesn’t wear pants, then he flies on Bieber’s jet unless Taylor is feeling lonely.

• **Predicate Logic**
  – If \( x, y, \) and \( z \) are positive integers, then \( x^3 + y^3 \neq z^3 \).
Predicate or Propositional Function

– A function that returns a truth value, e.g.,

“x is a cat”
“x is prime”
“student x has taken course y”
“x > y”
“x + y = z” or Sum(x, y, z)
“5 < x”

Predicates will have variables or constants as arguments.
We must specify a “domain of discourse”, which is the possible things we’re talking about.

“x is a cat”
  (e.g., mammals)

“x is prime”
  (e.g., positive whole numbers)

student x has taken course y”
  (e.g., students and courses)
\( \forall x \ P(x) \)

- P(x) is true for every x in the domain
- read as “for all x, P of x”

\( \exists x \ P(x) \)

- There is an x in the domain for which P(x) is true
- read as “there exists x, P of x”
statements with quantifiers

- $\exists x \text{ Even}(x)$
- $\forall x \text{ Odd}(x)$
- $\forall x (\text{Even}(x) \lor \text{Odd}(x))$
- $\exists x (\text{Even}(x) \land \text{Odd}(x))$
- $\forall x \text{ Greater}(x+1, x)$
- $\exists x (\text{Even}(x) \land \text{Prime}(x))$

Domain: Positive Integers

- Even(x)
- Odd(x)
- Prime(x)
- Greater(x, y) (or “x>y”)
- Equal(x, y) (or “x=y”)
- Sum(x, y, z) (or “z=x+y”)
• $\forall x \exists y \text{ Greater } (y, x)$  \(T\)

• $\forall x \exists y \text{ Greater } (x, y)$  \(F\) 
  \(x = 1\)

• $\forall x \exists y (\text{Greater}(y, x) \land \text{Prime}(y))$  \(T\)
  \(x = 20\)  \(y = 23\)

• $\forall x (\text{Prime}(x) \rightarrow (\text{Equal}(x, 2) \lor \text{Odd}(x)))$  \(T\)

• $\exists x \exists y (\text{Sum}(x, 2, y) \land \text{Prime}(x) \land \text{Prime}(y))$  \(T\)
  \(x = 3\)  \(y = 5\)
statements with quantifiers

- $\forall x \exists y \text{ Greater } (y, x)$
  - Prev: True
  - Now: False
  - Domain: All integers

- $\forall x \exists y \text{ Greater } (x, y)$
  - Prev: False
  - Now: True
  - Domain: Positive integers

Domain of quantifiers is important!
English to predicate logic

- “Red cats like tofu”

\[ \forall x \ (\text{Cat}(x) \land \text{Red}(x)) \rightarrow \text{LikesTofu}(x) \]  

Domain: mammals

- “Some red cats don’t like tofu”

\[ \exists x \ (\text{Cat}(x) \land \text{Red}(x)) \land \neg \text{LikesTofu}(x) \]
• not every positive integer is prime
  \[ \neg \forall x \text{ prime}(x) \]

• some positive integer is not prime
  \[ \exists x \neg \text{prime}(x) \]

• prime numbers do not exist
  \[ \neg \exists x \text{ prime}(x) \]

• every positive integer is not prime
  \[ \forall x \neg \text{prime}(x) \]
\( \forall x \text{PurpleFruit}(x) \)

Every fruit is purple

Which one is equal to \( \neg \forall x \text{PurpleFruit}(x) \)?

- \( \exists x \text{PurpleFruit}(x) \)?
  
  There is a purple fruit

- \( \exists x \neg \text{PurpleFruit}(x) \)?
  
  There is at least one fruit which is not purple.
de Morgan’s laws for quantifiers

\[ \neg \forall x \ P(x) \equiv \exists x \ \neg P(x) \]
\[ \neg \exists x \ P(x) \equiv \forall x \ \neg P(x) \]
de Morgan’s laws for quantifiers

\[
\neg \forall x \ P(x) \equiv \exists x \ \neg P(x) \\
\neg \exists x \ P(x) \equiv \forall x \ \neg P(x)
\]

“There is no largest integer.”

\[
\neg \exists \ x \ \forall \ y \ (x \geq y)
\equiv \forall \ x \ \neg \forall \ y \ (x \geq y)
\equiv \forall \ x \ \exists \ y \ \neg (x \geq y)
\equiv \forall \ x \ \exists \ y \ (y > x)
\]

“For every integer there is a larger integer.”
example: \[ \text{Notlargest}(x) \equiv \exists \ y \ \text{Greater} \ (y, x) \equiv \exists \ z \ \text{Greater} \ (z, x) \]

truth value:
- doesn’t depend on y or z “bound variables”
- does depend on x “free variable”

quantifiers only act on free variables of the formula they quantify

\[ \forall \ x \ (\exists \ y \ (P(x, y) \to \forall \ x \ Q(y, x))) \]
\( \exists x \ (P(x) \land Q(x)) \) \hspace{1cm} \text{vs.} \hspace{1cm} \exists x \ P(x) \land \exists x \ Q(x) \)