CompSci 102
Discrete Math for Computer Science

Announcements
• Read for next time Chap. 1.1-1.3
• Recitations start Friday, Aug 30
• Add yourself to CompSci 230 on Piazza

<table>
<thead>
<tr>
<th>$p$</th>
<th>$q$</th>
<th>$p \land q$</th>
</tr>
</thead>
<tbody>
<tr>
<td>F</td>
<td>F</td>
<td>F</td>
</tr>
<tr>
<td>F</td>
<td>T</td>
<td>F</td>
</tr>
<tr>
<td>T</td>
<td>F</td>
<td>F</td>
</tr>
<tr>
<td>T</td>
<td>T</td>
<td>T</td>
</tr>
</tbody>
</table>

August 29, 2013

Prof. Rodger

Last Time
• The Pancake Problem
  – Showed 4 flips necessary to flip this stack
  – $P_n =$ number of flips required to sort a worst case stack of $n$ pancakes
  – Lower and upper bounds
    • Bring to Top Method gave us an upper bound.
    • Break Apart Method gave us a lower bound.
    • $n \leq P_n \leq 2n - 3$
    • Want the largest lower bound and the smallest upperbound
  – $P_5 = 5$. Will explore more in recitation

Logic King of them All
1. If I'm doing better than you, you're a noob.
2. If you're doing better than me, you have no life.
3. If everyone is doing better than me, I have lag.

Logic Problem

- On one side of the river you have:
  - You, goat, head of cabbage, wolf
- You can’t leave the wolf with the goat, and you can’t leave the goat with the cabbage.
- You have a boat that can only hold two of you.
- How do you get everyone across the river?

Xkcd boat logic

Logic

- Rules of logic specify meaning of mathematical statements
- How do you understand:
  \[ \forall n > 0 \, \sum_{i=1}^{n} i = \frac{n(n+1)}{2} \]
- Applications in CS
  - Designing computers
  - Designing programming languages
  - Correctness of programs
  - Many areas such as artificial intelligence
How old …..

• Aristotle developed propositional logic over 2000 years ago….

• George Boole wrote “The Mathematical Analysis of Logic” in 1848

Proposition

• A **proposition** is a sentence that declares a fact that is true or false

• A **theorem** is a proposition that is guaranteed by a proof

Examples of Propositions

• Which are propositions? What is their value?
  1. Duke won the NCAA men’s basketball title in 2010.
  2. $3x > 2$
  3. Clean up after yourself.
  4. Durham is the capital of NC.
  5. Pepsi was invented in New Bern NC in 1898.
  6. $8 + 3 = 11$

A Proof Example

• **Theorem:** *(Pythagorean Theorem of Euclidean geometry)* For any real numbers $a$, $b$, and $c$, if $a$ and $b$ are the base-length and height of a right triangle, and $c$ is the length of its hypotenuse, then $a^2 + b^2 = c^2$.

• **Proof?**
### Proof of Pythagorean Theorem

- **Proof.** Consider the below diagram:

![Diagram](image_url)

- Exterior square area = $c^2$, the sum of the following regions:
  - The area of the 4 triangles = $4(\frac{1}{2}ab) = 2ab$
  - The area of the small interior square = $(b-a)^2 = b^2 - 2ab + a^2$.
- Thus, $c^2 = 2ab + (b^2 - 2ab + a^2) = a^2 + b^2$. ■

### Operators / Connectives

An operator or connective combines one or more operand expressions into a larger expression. (E.g., “+” in numeric exprs.)

- **Unary** operators take 1 operand (e.g., $-3$);
- **binary** operators take 2 operands (e.g., $3 \times 4$).
- **Propositional** or **Boolean** operators operate on propositions (or their truth values) instead of on numbers.

### Some Popular Boolean Operators

<table>
<thead>
<tr>
<th>Formal Name</th>
<th>Nickname</th>
<th>Arity</th>
<th>Symbol</th>
</tr>
</thead>
<tbody>
<tr>
<td>Negation operator</td>
<td>NOT</td>
<td>Unary</td>
<td>¬</td>
</tr>
<tr>
<td>Conjunction operator</td>
<td>AND</td>
<td>Binary</td>
<td>∧</td>
</tr>
<tr>
<td>Disjunction operator</td>
<td>OR</td>
<td>Binary</td>
<td>∨</td>
</tr>
<tr>
<td>Exclusive-OR operator</td>
<td>XOR</td>
<td>Binary</td>
<td>⊕</td>
</tr>
<tr>
<td>Implication operator</td>
<td>IMPLIES</td>
<td>Binary</td>
<td>→</td>
</tr>
<tr>
<td>Biconditional operator</td>
<td>IFF</td>
<td>Binary</td>
<td>↔</td>
</tr>
</tbody>
</table>
### The Negation Operator

The unary *negation operator* “¬” (NOT) transforms a prop. into its logical *negation*.

*E.g.* If \( p = \text{“I have brown hair.”} \)

then \( \neg p = \text{“I do not have brown hair.”} \)

The *truth table* for NOT:

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

\( T \) : \( \equiv \) True; \( F \) : \( \equiv \) False

“\( :\)” means “is defined as”

### The Conjunction Operator

The binary *conjunction operator* “\( \land \)” (AND) combines two propositions to form their logical *conjunction*.

*E.g.* If \( p=\text{“I will have salad for lunch.”} \) and \( q=\text{“I will have steak for dinner.”}, \) then

\( p \land q=\text{“I will have salad for lunch and I will have steak for dinner.”} \)

Remember: “\( \land \)” points up like an “A”, and it means “\( \land \text{AND} \)”

### Conjunction Truth Table

- A conjunction of \( n \) propositions will have how many rows in its truth table?
- Note: \( \neg \) and \( \land \) operations together are sufficient to express *any* Boolean truth table!

<table>
<thead>
<tr>
<th>( p \land q )</th>
<th>( p )</th>
<th>( q )</th>
</tr>
</thead>
<tbody>
<tr>
<td>F</td>
<td>F</td>
<td>F</td>
</tr>
<tr>
<td>T</td>
<td>F</td>
<td>F</td>
</tr>
<tr>
<td>T</td>
<td>T</td>
<td>T</td>
</tr>
</tbody>
</table>

### The Disjunction Operator

The binary *disjunction operator* “\( \lor \)” (OR) combines two propositions to form their logical *disjunction*.

\( p=\text{“My car has a bad engine.”} \) \nm\( q=\text{“My car has a bad carburetor.”} \)

\( p \lor q=\text{“Either my car has a bad engine, or my car has a bad carburetor.”} \)

Meaning is like “\( \land \text{AND} \)” in English.
• Note that $p \lor q$ means that $p$ is true, or $q$ is true, or both are true!

<table>
<thead>
<tr>
<th>$p$</th>
<th>$q$</th>
<th>$p \lor q$</th>
</tr>
</thead>
<tbody>
<tr>
<td>F</td>
<td>F</td>
<td>F</td>
</tr>
<tr>
<td>F</td>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>T</td>
<td>F</td>
<td>T</td>
</tr>
<tr>
<td>T</td>
<td>T</td>
<td>T</td>
</tr>
</tbody>
</table>

Note difference from AND

• So, this operation is also called inclusive or, because it includes the possibility that both $p$ and $q$ are true.

• “¬” and “∨” together are also universal.

Nested Propositional Expressions

• Use parentheses to group sub-expressions: “I just saw my old friend, and either he’s grown or I’ve shrunk.” = $f \land (g \lor s)$

  $-(f \land g) \lor s$ would mean something different
  $f \land g \lor s$ would be ambiguous

• By convention, “¬” takes precedence over both “∧” and “∨”.

  $-s \land f$ means $(s) \land f$, not $(s \land f)$

A Simple Exercise

Let $p$ = “It rained last night”,
$q$ = “The sprinklers came on last night,”
$r$ = “The lawn was wet this morning.”

Translate each of the following into English:

$\neg p$ = “It didn’t rain last night.”
$r \land \neg p$ = “The lawn was wet this morning, and it didn’t rain last night.”
$\neg r \lor p \lor q$ = “Either the lawn wasn’t wet this morning, or it rained last night, or the sprinklers came on last night.”
The *Exclusive Or Operator*

The binary *exclusive-or operator* “⊕” (XOR) combines two propositions to form their logical “exclusive or” (exjunction?).

\[ p = \text{“I will earn an A in this course,”} \]
\[ q = \text{“I will drop this course,”} \]
\[ p \oplus q = \]

---

**Exclusive-Or Truth Table**

- Note that \( p \oplus q \) means that \( p \) is true, or \( q \) is true, but **not both**!
- This operation is called *exclusive or*, because it excludes the possibility that both \( p \) and \( q \) are true.
- “¬” and “⊕” together are **not** universal.

<table>
<thead>
<tr>
<th>( p )</th>
<th>( q )</th>
<th>( p \oplus q )</th>
</tr>
</thead>
<tbody>
<tr>
<td>F</td>
<td>F</td>
<td>F</td>
</tr>
<tr>
<td>F</td>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>T</td>
<td>F</td>
<td>T</td>
</tr>
<tr>
<td>T</td>
<td>T</td>
<td>F</td>
</tr>
</tbody>
</table>

---

**Exclusive-Or Truth Table**

- Note that \( p \oplus q \) means that \( p \) is true, or \( q \) is true, but **not both**!
- This operation is called *exclusive or*, because it excludes the possibility that both \( p \) and \( q \) are true.
- “¬” and “⊕” together are **not** universal.
Natural Language is Ambiguous

Note that English “or” can be ambiguous regarding the “both” case!

<table>
<thead>
<tr>
<th>p</th>
<th>q</th>
<th>p &quot;or&quot; q</th>
</tr>
</thead>
<tbody>
<tr>
<td>F</td>
<td>F</td>
<td>F</td>
</tr>
<tr>
<td>F</td>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>T</td>
<td>F</td>
<td>T</td>
</tr>
<tr>
<td>T</td>
<td>T</td>
<td>?</td>
</tr>
</tbody>
</table>

“Pat is a singer or Pat is a writer.”
“Pat is a man or Pat is a woman.”
Need context to disambiguate the meaning!
For this class, assume “or” means inclusive.

The Implication Operator

The implication $p \rightarrow q$ states that $p$ implies $q$.

I.e., if $p$ is true, then $q$ is true; but if $p$ is not true, then $q$ could be either true or false.

E.g., let $p =$ “You study hard.”
$q =$ “You will get a good grade.”
$p \rightarrow q =$ “If you study hard, then you will get a good grade.” (else, it could go either way)

Implication Truth Table

- $p \rightarrow q$ is false only when $p$ is true but $q$ is not true.
- $p \rightarrow q$ does not say that $p$ causes $q$!
- $p \rightarrow q$ does not require that $p$ or $q$ are ever true!
- E.g. “(1=0) \rightarrow pigs can fly” is
Implication Truth Table

- \( p \rightarrow q \) is **false** only when \( p \) is true but \( q \) is **not** true.
- \( p \rightarrow q \) does **not** say that \( p \) **causes** \( q \).
- \( p \rightarrow q \) does **not** require that \( p \) or \( q \) are **ever true**!
- E.g. \( (1=0) \rightarrow \) pigs can fly” is **TRUE**!

<table>
<thead>
<tr>
<th></th>
<th></th>
<th>( p \rightarrow q )</th>
</tr>
</thead>
<tbody>
<tr>
<td>F</td>
<td>F</td>
<td>T</td>
</tr>
<tr>
<td>F</td>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>T</td>
<td>F</td>
<td>F</td>
</tr>
<tr>
<td>T</td>
<td>T</td>
<td>T</td>
</tr>
</tbody>
</table>

The only False case!

Examples of Implications

- “If this lecture ever ends, then the sun will rise tomorrow.” **True or False?**
- “If Tuesday is a day of the week, then I am a penguin.” **True or False?**
- “If 1+1=6, then Obama is president.” **True or False?**
- “If the moon is made of green cheese, then I am richer than Bill Gates.” **True or False?**

Why does this seem wrong?

- Consider a sentence like,
  - “If I wear a red shirt tomorrow, then I will win the lottery!”
- In logic, we consider the sentence **True** so long as either I don’t wear a red shirt, or I win the lottery.
- But, in normal English conversation, if I were to make this claim, you would think that I was lying.
  - Why this discrepancy between logic & language?

Resolving the Discrepancy

- In English, a sentence “if \( p \) then \( q \)” usually really implicitly means something like,
  - “In all possible situations, if \( p \) then \( q \).”
    - That is, “For \( p \) to be true and \( q \) false is impossible.”
    - Or, “I guarantee that no matter what, if \( p \), then \( q \).”
- This can be expressed in **predicate logic** as:
  - “For all situations \( s \), if \( p \) is true in situation \( s \), then \( q \) is also true in situation \( s \)”
  - Formally, we could write: \( \forall s, P(s) \rightarrow Q(s) \)
- That sentence is logically **False** in our example, because for me to wear a red shirt and for me to not win the lottery is a **possible** (even if not actual) situation.
  - Natural language and logic then agree with each other.
English Phrases Meaning $p \rightarrow q$

- “$p$ implies $q$”
- “if $p$, then $q$”
- “$p$, $q$”
- “when $p$, $q$”
- “whenever $p$, $q$”
- “$q$ if $p$”
- “$q$ when $p$”
- “$q$ whenever $p$”

Converse, Inverse, Contrapositive

Some terminology, for an implication $p \rightarrow q$:

- Its converse is: $q \rightarrow p$.
- Its inverse is: $\neg p \rightarrow \neg q$.
- Its contrapositive: $\neg q \rightarrow \neg p$.

One of these three has the same meaning (same truth table) as $p \rightarrow q$. Can you figure out which?

How do we know for sure?

Proving the equivalence of $p \rightarrow q$ and $\neg q \rightarrow \neg p$ using truth tables:

<table>
<thead>
<tr>
<th>$p$</th>
<th>$q$</th>
<th>$\neg q$</th>
<th>$\neg p$</th>
<th>$p \rightarrow q$</th>
<th>$\neg q \rightarrow \neg p$</th>
</tr>
</thead>
<tbody>
<tr>
<td>F</td>
<td>F</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>F</td>
<td>T</td>
<td>F</td>
<td>T</td>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>T</td>
<td>F</td>
<td>T</td>
<td>F</td>
<td>F</td>
<td>F</td>
</tr>
<tr>
<td>T</td>
<td>T</td>
<td>F</td>
<td>T</td>
<td>T</td>
<td>T</td>
</tr>
</tbody>
</table>
The biconditional operator

The biconditional \( p \iff q \) states that \( p \) is true if and only if (IFF) \( q \) is true.
When we say \( P \) if and only if \( q \), we are saying that \( P \) says the same thing as \( Q \).
Examples?
Truth table?

Biconditional Truth Table

\[
\begin{array}{c|cc}
 p & q & p \iff q \\
\hline
 F & F & T \\
 F & T & F \\
 T & F & F \\
 T & T & T \\
\end{array}
\]

- \( p \iff q \) means that \( p \) and \( q \) have the same truth value.
- Note this truth table is the exact opposite of \( \oplus \)'s!
  Thus, \( p \iff q \) means \( \neg(p \oplus q) \)
- \( p \iff q \) does not imply that \( p \) and \( q \) are true, or that either of them causes the other, or that they have a common cause.

Boolean Operations Summary

- We have seen 1 unary operator and 5 binary operators. Their truth tables are below.

<table>
<thead>
<tr>
<th>( p )</th>
<th>( q )</th>
<th>( \neg p )</th>
<th>( p \land q )</th>
<th>( p \lor q )</th>
<th>( p \oplus q )</th>
<th>( p \rightarrow q )</th>
<th>( p \iff q )</th>
</tr>
</thead>
<tbody>
<tr>
<td>F</td>
<td>F</td>
<td>T</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>F</td>
<td>T</td>
<td>T</td>
<td>F</td>
<td>T</td>
<td>T</td>
<td>F</td>
<td>T</td>
</tr>
<tr>
<td>T</td>
<td>F</td>
<td>F</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>F</td>
<td>F</td>
</tr>
<tr>
<td>T</td>
<td>T</td>
<td>F</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>F</td>
<td>T</td>
</tr>
</tbody>
</table>

Some Alternative Notations

<table>
<thead>
<tr>
<th>Name:</th>
<th>not</th>
<th>and</th>
<th>or</th>
<th>xor</th>
<th>implies</th>
<th>iff</th>
</tr>
</thead>
<tbody>
<tr>
<td>Propositional logic:</td>
<td>( \neg )</td>
<td>( \land )</td>
<td>( \lor )</td>
<td>( \oplus )</td>
<td>( \rightarrow )</td>
<td>( \iff )</td>
</tr>
<tr>
<td>Boolean algebra:</td>
<td>( \overline{p} )</td>
<td>( pq )</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C/C++/Java (wordwise):</td>
<td>!</td>
<td>&amp; &amp;</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C/C++/Java (bitwise):</td>
<td>( \sim )</td>
<td>&amp;</td>
<td></td>
<td>^</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Logic gates:</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>