Foundations of Computing I

Fall 2014
Remember, the site is...

http://tinyurl.com/ynlecture

Get started on the green handout!
Negations of Quantifiers

- $\forall x \text{PurpleFruit}(x)$
  - “All fruits are purple”
- What is $\neg \forall x \text{PurpleFruit}(x)$?
  - “Not all fruits are purple”

- How about $\exists x \text{PurpleFruit}(x)$?
  - “There is a purple fruit”
  - If it’s the negation, all situations should be covered by a statement and its negation
  - Consider the domain \{Orange\}: Neither statement is true!
  - No!

- How about $\exists x \neg \text{PurpleFruit}(x)$?
  - “There is a fruit that isn’t purple”
  - Yes!
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”
Scope of Quantifiers

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

\[ \forall x (\exists y (P(x, y) \rightarrow \forall x Q(y, x))) \]
scope of quantifiers

\[ \exists x \ (P(x) \land Q(x)) \quad \text{vs.} \quad \exists x \ P(x) \land \exists x \ Q(x) \]

This one asserts \( P \) and \( Q \) of the same \( x \).

This one asserts \( P \) and \( Q \) of potentially different \( x \)’s.
Quantifiers are Like Code

\[ \forall x \ (\exists y \ (P(x, y) \rightarrow \forall x \ Q(y, x))) \]

```java
public boolean blue() {
    for (T x : DOMAIN) {
        if (!green(x)) {
            return false;
        }
    }
    return true;
}

public boolean red(T z, T y) {
    for (T x : DOMAIN) {
        if (!Q(y, x)) {
            return false;
        }
    }
    return true;
}

public boolean green(T x) {
    for (T y : DOMAIN) {
        if (!P(x, y) || red(x, y)) {
            return true;
        }
    }
    return false;
}
```

Notice that we renamed \( x \) in `red`, because we define another \( x \) inside.

We recommend that you NOT re-use the same variable like this.
If the tortoise walks at a rate of one node per step, and the hare walks at a rate of two nodes per step, then the distance between them increases by one node per step.

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

If the tortoise is on node \(x\), and the hare is on node \(2x\), then the distance between them increases by one node per step.

\[(\forall x \,(\text{OnNode(Tortoise, } x) \land \text{OnNode(Hare, } 2x))) \rightarrow p\]
Nested Quantifiers

• **Bound variable names don’t matter**
  \[ \forall x \exists y P(x, y) \equiv \forall a \exists b P(a, b) \]

• **Positions of quantifiers can sometimes change**
  \[ \forall x (Q(x) \land \exists y P(x, y)) \equiv \forall x \exists y (Q(x) \land P(x, y)) \]

• **But: order is important...**
Predicate with Two Variables

$P(x, y)$
## Quantification with Two Variables

<table>
<thead>
<tr>
<th>expression</th>
<th>when true</th>
<th>when false</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\forall x \forall y \ P(x, y)$</td>
<td>Every pair is true.</td>
<td>At least one pair is false.</td>
</tr>
<tr>
<td>$\exists x \exists y \ P(x, y)$</td>
<td>At least one pair is true.</td>
<td>All pairs are false.</td>
</tr>
<tr>
<td>$\forall x \exists y \ P(x, y)$</td>
<td>We can find a specific $y$ for each $x$. $(x_1, y_1), (x_2, y_2), (x_3, y_3)$</td>
<td>Some $x$ doesn’t have a corresponding $y$.</td>
</tr>
<tr>
<td>$\exists y \forall x \ P(x, y)$</td>
<td>We can find ONE $y$ that works no matter what $x$ is. $(x_1, y), (x_2, y), (x_3, y)$</td>
<td>For any candidate $y$, there is an $x$ that it doesn’t work for.</td>
</tr>
</tbody>
</table>
Logical Inference

• So far we’ve considered:
  – How to understand and express things using propositional and predicate logic
  – How to compute using Boolean (propositional) logic
  – How to show that different ways of expressing or computing them are equivalent to each other

• Logic also has methods that let us infer implied properties from ones that we know
  – Equivalence is a small part of this
Applications of Logical Inference

- **Software Engineering**
  - Express desired properties of program as set of logical constraints
  - Use inference rules to show that program implies that those constraints are satisfied

- **Artificial Intelligence**
  - Automated reasoning

- **Algorithm design and analysis**
  - e.g., Correctness, Loop invariants.

- **Logic Programming, e.g. Prolog**
  - Express desired outcome as set of constraints
  - Automatically apply logic inference to derive solution
Proofs

- Start with hypotheses and facts
- Use rules of inference to extend set of facts
- Result is proved when it is included in the set
An inference rule: *Modus Ponens*

- If p and p → q are both true then q must be true

- Write this rule as \( p, \ p \rightarrow q \) \[\therefore q\]

- Given:
  - If it is Monday then you have a 311 class today.
  - It is Monday.

- Therefore, by modus ponens:
  - You have a 311 class today.
Proofs

Show that \( r \) follows from \( p, p \rightarrow q, \) and \( q \rightarrow r \)

1. \( p \) given
2. \( p \rightarrow q \) given
3. \( q \rightarrow r \) given
4. \( q \) modus ponens from 1 and 2
5. \( r \) modus ponens from 3 and 4
Proofs can use equivalences too

Show that $\neg p$ follows from $p \rightarrow q$ and $\neg q$

1. $p \rightarrow q$  given
2. $\neg q$  given
3. $\neg q \rightarrow \neg p$  contrapositive of 1
4. $\neg p$  modus ponens from 2 and 3
Inference Rules

• Each inference rule is written as:
  ...which means that if both A and B are true then you can infer C and you can infer D.
  – For rule to be correct \((A \land B) \rightarrow C\) and \((A \land B) \rightarrow D\) must be a tautologies

• Sometimes rules don’t need anything to start with. These rules are called axioms:
  – e.g. Excluded Middle Axiom
    \[
    \therefore p \lor \neg p
    \]
Simple Propositional Inference Rules

Excluded middle plus two inference rules per binary connective, one to eliminate it and one to introduce it

\[
\begin{align*}
\text{p} \land \text{q} & \quad \therefore \quad \text{p}, \text{q} \\
\text{p} \lor \text{q}, \neg \text{p} & \quad \therefore \quad \text{q} \\
\text{p}, \text{p} \rightarrow \text{q} & \quad \therefore \quad \text{q} \\
\text{p} \Rightarrow \text{q} & \quad \therefore \quad \text{p} \rightarrow \text{q} \\
\text{p} & \quad \therefore \quad \text{p} \lor \text{q}, \text{q} \lor \text{p}
\end{align*}
\]

Direct Proof Rule

Not like other rules
Important: Applications of inference rules

• You can use equivalences to make substitutions of any sub-formula.

• Inference rules only can be applied to whole formulas (not correct otherwise).

  e.g. 1. \( p \rightarrow q \)  
  
  2. \( (p \lor r) \rightarrow q \)  
  intro \( \lor \) from 1.

  Does not follow! e.g. \( p=F, q=F, r=T \)