Lecture 6: More Predicate Logic

THREE LOGICIANS WALK INTO A BAR...

DOES EVERYONE WANT BEER?

I DON'T KNOW.

I DON'T KNOW.

YES!
Predicate

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

Cat(x) ::= “x is a cat”
Prime(x) ::= “x is prime”
HasTaken(x, y) ::= “student x has taken course y”
LessThan(x, y) ::= “x < y”
Sum(x, y, z) ::= “x + y = z”
GreaterThan5(x) ::= “x > 5”
HasNChars(s, n) ::= “string s has length n”

Predicates can have varying numbers of arguments and input types.
For ease of use, we define one “type”/“domain” that we work over. This set of objects is called the “domain of discourse”.

For each of the following, what might the domain be?
(1) “x is a cat”, “x barks”, “x ruined my couch”

(2) “x is prime”, “x = 0”, “x < 0”, “x is a power of two”

(3) “x is a pre-req for z”
Domain of Discourse

For ease of use, we define one “type”/“domain” that we work over. This set of objects is called the “domain of discourse”.

For each of the following, what might the domain be?

(1) “\(x\) is a cat”, “\(x\) barks”, “\(x\) ruined my couch”

“mammals” or “sentient beings” or “cats and dogs” or …

(2) “\(x\) is prime”, “\(x = 0\)”, “\(x < 0\)”, “\(x\) is a power of two”

“numbers” or “integers” or “integers greater than 5” or …

(3) “\(x\) is a pre-req for \(z\)”

“courses”
Last Class: Quantifiers

We use *quantifiers* to talk about collections of objects.

\[ \forall x \ P(x) \]

*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*”
Determine the truth values of each of these statements:

\[ \exists x \text{ Even}(x) \]

\[ \forall x \text{ Odd}(x) \]

\[ \forall x (\text{Even}(x) \lor \text{Odd}(x)) \]

\[ \exists x (\text{Even}(x) \land \text{Odd}(x)) \]

\[ \forall x \text{ Greater}(x+1, x) \]

\[ \exists x (\text{Even}(x) \land \text{Prime}(x)) \]
Statements with Quantifiers

<table>
<thead>
<tr>
<th>Domain of Discourse</th>
<th>Predicate Definitions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Positive Integers</td>
<td>Even(x) ::= “x is even”</td>
</tr>
<tr>
<td></td>
<td>Greater(x, y) ::= “x &gt; y”</td>
</tr>
<tr>
<td></td>
<td>Odd(x) ::= “x is odd”</td>
</tr>
<tr>
<td></td>
<td>Equal(x, y) ::= “x = y”</td>
</tr>
<tr>
<td></td>
<td>Prime(x) ::= “x is prime”</td>
</tr>
<tr>
<td></td>
<td>Sum(x, y, z) ::= “x + y = z”</td>
</tr>
</tbody>
</table>

Determine the truth values of each of these statements:

\[\exists x \text{ Even}(x)\]  \[T\]  e.g. 2, 4, 6, ...

\[\forall x \text{ Odd}(x)\]  \[F\]  e.g. 2, 4, 6, ...

\[\forall x (\text{Even}(x) \lor \text{Odd}(x))\]  \[T\]  every integer is either even or odd

\[\exists x (\text{Even}(x) \land \text{Odd}(x))\]  \[F\]  no integer is both even and odd

\[\forall x \text{ Greater}(x+1, x)\]  \[T\]  adding 1 makes a bigger number

\[\exists x (\text{Even}(x) \land \text{Prime}(x))\]  \[T\]  Even(2) is true and Prime(2) is true
Translate the following statements to English

∀x ∃y Greater(y, x)

∀x ∃y Greater(x, y)

∀x ∃y (Greater(y, x) ∧ Prime(y))

∀x (Prime(x) → (Equal(x, 2) ∨ Odd(x)))

∃x ∃y (Sum(x, 2, y) ∧ Prime(x) ∧ Prime(y))
Translate the following statements to English

\( \forall x \ \exists y \ \text{Greater}(y, x) \)

For every positive integer \( x \), there is a positive integer \( y \), such that \( y > x \).

\( \forall x \ \exists y \ \text{Greater}(x, y) \)

For every positive integer \( x \), there is a positive integer \( y \), such that \( x > y \).

\( \forall x \ \exists y (\text{Greater}(y, x) \land \text{Prime}(y)) \)

For every positive integer \( x \), there is a pos. int. \( y \) such that \( y > x \) and \( y \) is prime.

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

For each positive integer \( x \), if \( x \) is prime, then \( x = 2 \) or \( x \) is odd.

\( \exists x \ \exists y (\text{Sum}(x, 2, y) \land \text{Prime}(x) \land \text{Prime}(y)) \)

There exist positive integers \( x \) and \( y \) such that \( x + 2 = y \) and \( x \) and \( y \) are prime.
Translate the following statements to English

\( \forall x \, \exists y \, \text{Greater}(y, x) \)

There is no greatest integer.

\( \forall x \, \exists y \, \text{Greater}(x, y) \)

There is no least integer.

\( \forall x \, \exists y \, (\text{Greater}(y, x) \land \text{Prime}(y)) \)

For every positive integer there is a larger number that is prime.

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

Every prime number is either 2 or odd.

\( \exists x \, \exists y \, (\text{Sum}(x, 2, y) \land \text{Prime}(x) \land \text{Prime}(y)) \)

There exist prime numbers that differ by two.”
English to Predicate Logic

<table>
<thead>
<tr>
<th>Domain of Discourse</th>
<th>Predicate Definitions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mammals</td>
<td>Cat(x) ::= “x is a cat”</td>
</tr>
<tr>
<td></td>
<td>Red(x) ::= “x is red”</td>
</tr>
<tr>
<td></td>
<td>LikesTofu(x) ::= “x likes tofu”</td>
</tr>
</tbody>
</table>

“Red cats like tofu”

“Some red cats don’t like tofu”
“Red cats like tofu”

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

“Some red cats don’t like tofu”

\[ \exists y \ ((\text{Red}(y) \land \text{Cat}(y)) \land \neg \text{LikesTofu}(y)) \]
English to Predicate Logic

**Domain of Discourse**

| Mammals |

**Predicate Definitions**

| Cat(x) ::= “x is a cat”     |
| Red(x) ::= “x is red”       |
| LikesTofu(x) ::= “x likes tofu” |

When putting two predicates together like this, we use an “and”.

“**Red cats** like tofu”

When there’s no leading quantification, it means “for all”.

“**Some red cats** don’t like tofu”

When restricting to a smaller domain in a “for all” we use implication.

When restricting to a smaller domain in an “exists” we use and.

“Some” means “there exists”.

When putting two predicates together like this, we use an “and”. 
Negations of Quantifiers

Predicate Definitions
PurpleFruit(x) ::= “x is a purple fruit”

(*) $\forall x \text{PurpleFruit}(x)$ (“All fruits are purple”)

What is the negation of (*)?
(a) “there exists a purple fruit”
(b) “there exists a non-purple fruit”
(c) “all fruits are not purple”

Try your intuition! Which one “feels” right?

Key Idea: In every domain, exactly one of a statement and its negation should be true.
Negations of Quantifiers

Predicate Definitions

\[ \text{PurpleFruit}(x) ::= \text{“}x \text{ is a purple fruit}\text{”} \]

\((*)\) \(\forall x \ \text{PurpleFruit}(x)\) (“All fruits are purple”)

What is the negation of \((*)\)?

(a) “there exists a purple fruit”
(b) “there exists a non-purple fruit”
(c) “all fruits are not purple”

Key Idea: In every domain, exactly one of a statement and its negation should be true.

<table>
<thead>
<tr>
<th>Domain of Discourse</th>
<th>Domain of Discourse</th>
<th>Domain of Discourse</th>
</tr>
</thead>
<tbody>
<tr>
<td>{plum}</td>
<td>{apple}</td>
<td>{plum, apple}</td>
</tr>
<tr>
<td>((*)), (a)</td>
<td>(b), (c)</td>
<td>(a), (b)</td>
</tr>
</tbody>
</table>
Negations of Quantifiers

Predicate Definitions

PurpleFruit(x) ::= “x is a purple fruit”

\( \forall x \text{ PurpleFruit}(x) \) (“All fruits are purple”)

What is the negation of (*)?
(a) “there exists a purple fruit”
(b) “there exists a non-purple fruit”
(c) “all fruits are not purple”

Key Idea: In every domain, exactly one of a statement and its negation should be true.

<table>
<thead>
<tr>
<th>Domain of Discourse</th>
<th>Domain of Discourse</th>
<th>Domain of Discourse</th>
</tr>
</thead>
<tbody>
<tr>
<td>{plum}</td>
<td>{apple}</td>
<td>{plum, apple}</td>
</tr>
</tbody>
</table>

(*) \( \forall x \text{ PurpleFruit}(x) \)

(b), (c) \( \forall x \text{ PurpleFruit}(x) \)

(a), (b) \( \forall x \text{ PurpleFruit}(x) \)

The only choice that ensures exactly one of the statement and its negation is (b).
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”
Scope of Quantifiers

$\exists x \ (P(x) \land Q(x)) \quad \text{vs.} \quad \exists x \ P(x) \land \exists x \ Q(x)$
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.
Scope of Quantifiers

**Example:**

\[ \neg \text{NotLargest}(x) \equiv \exists \ y \ \text{Greater} \ (y, x) \]
\[ \equiv \exists \ z \ \text{Greater} \ (z, x) \]

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

\[ \forall \ x \ (\exists \ y \ (P(x,y) \rightarrow \forall \ x \ Q(y, x))) \]
This isn’t “wrong”, it’s just horrible style. Don’t confuse your reader by using the same variable multiple times...there are a lot of letters...
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...**
Quantifier Order Can Matter

“There is a number greater than or equal to all numbers.”

\[ \exists x \, \forall y \, \text{GreaterEq}(x, y) \]

“Every number has a number greater than or equal to it.”

\[ \forall y \, \exists x \, \text{GreaterEq}(x, y) \]

The purple statement requires an entire row to be true. The red statement requires one entry in each column to be true.
# 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 $p \rightarrow q$ 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.
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. 
5.
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. \( q \) MP: 1, 2
5. \( r \) MP: 3, 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: 1
4. $\neg p$               MP: 2, 3
Inference Rules

• Each inference rule is written as:
  \[ \frac{\text{A, B}}{\therefore \text{C, D}} \]
  ...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 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

\[ p \land q \quad \text{\vdash} \quad p, q \]
\[ p, q \quad \text{\vdash} \quad p \land q \]
\[ p \lor q, \neg p \quad \text{\vdash} \quad p \lor q, q \lor p \]
\[ p \Rightarrow q \quad \text{\vdash} \quad p \lor q, q \lor p \]
\[ p, p \rightarrow q \quad \text{\vdash} \quad q \]
\[ p \Rightarrow q \quad \text{\vdash} \quad p \rightarrow q \]

Direct Proof Rule
Not like other rules
Show that $r$ follows from $p$, $p \rightarrow q$ and $(p \land q) \rightarrow r$

How To Start:
We have givens, find the ones that go together and use them. Now, treat new things as givens, and repeat.

$p, p \rightarrow q$
\[\therefore q\]

$p \land q$
\[\therefore p, q\]

$p, q$
\[\therefore p \land q\]
Show that \( r \) follows from \( p, p \to q \), and \( p \land q \to r \)

1. \( p \)  Given
2. \( p \to q \)  Given
3. \( q \)  MP: 1, 2
4. \( p \land q \)  Intro \( \land \): 1, 3
5. \( p \land q \to r \)  Given
6. \( r \)  MP: 4, 5
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. \( p \rightarrow q \) given

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

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