Predicate logic, logical inference

**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 \( \text{Sum}(x, y, z) \)

Predicates will have variables or constants as arguments.

**Review: Quantifiers**

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

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

• ∃ x Even(x)

• ∀ x Odd(x)

• ∀ x (Even(x) ∨ Odd(x))

• ∃ x (Even(x) ∧ Odd(x))

• ∀ x Greater(x+1, x)

• ∃ x (Even(x) ∧ Prime(x))

Domain: Positive Integers

review: statements with quantifiers

• ∀ x ∃ y Greater(y, x)

• ∀ 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))

Domain: Positive Integers

review: statements with quantifiers

• “There is an odd prime”

• “If x is greater than two, x is not an even prime”

• ∀x∀y∀z ((Sum(x, y, z) ∧ Odd(x) ∧ Odd(y))→ Even(z))

• “There exists an odd integer that is the sum of two primes”

Domian: Positive Integers

review: English to predicate logic

“Red cats like tofu”

Domain: Integers

Cat(x) Red(x) LikesTofu(x)
**Goldbach’s Conjecture**

“Every even integer greater than two can be expressed as the sum of two primes”

**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

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

**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></td>
<td></td>
</tr>
<tr>
<td>( \exists x \exists y P(x, y) )</td>
<td></td>
<td></td>
</tr>
<tr>
<td>( \forall x \exists y P(x, y) )</td>
<td></td>
<td></td>
</tr>
<tr>
<td>( \exists y \forall x P(x, y) )</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Negations of quantifiers**

- Not every positive integer is prime
- Some positive integer is not prime
- Prime numbers do not exist
- Every positive integer is not prime

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

“For every integer there is a larger integer”

**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 \[
\frac{p, p \rightarrow q}{\therefore q}
\]

- 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:
  \[
  \frac{A, B}{\therefore C,D}
  \]
  ...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 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

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

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 \quad \text{given}

  2. (p \lor r) \rightarrow q \quad \text{intro } \lor \text{ from 1.}

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

direct proof of an implication

- p \Rightarrow q denotes a proof of q given p as an assumption

- The direct proof rule:
  
  If you have such a proof then you can conclude that p \rightarrow q is true

Example:

\begin{align*}
1. p & \quad \text{assumption} \\
2. p \lor q & \quad \text{intro for } \lor \text{ from 1} \\
3. p \rightarrow (p \lor q) & \quad \text{direct proof rule}
\end{align*}