1 Syntax of Predicate Logic
2 Predicate Logic as a Formal Language
3 Semantics of Predicate Logic
1 Syntax of Predicate Logic
   - Need for Richer Language
   - Predicates
   - Variables
   - Functions

2 Predicate Logic as a Formal Language

3 Semantics of Predicate Logic
More Declarative Sentences

- Propositional logic can easily handle simple declarative statements such as:

  Example
  Student Peter Lim enrolled in CS3234.

- Propositional logic can also handle combinations of such statements such as:

  Example
  Student Peter Lim enrolled in Tutorial 1, and student Julie Bradshaw is enrolled in Tutorial 2.

  But: How about statements with “there exists...” or “every...” or “among...”?
What is needed?

Example

Every student is younger than some instructor.

What is this statement about?

- Being a student
- Being an instructor
- Being younger than somebody else

These are properties of elements of a set of objects.

We express them in predicate logic using predicates.
Predicates

Example

*Every* student is younger than *some* instructor.

- $S(andy)$ could denote that Andy is a student.
- $l(paul)$ could denote that Paul is an instructor.
- $Y(andy, paul)$ could denote that Andy is younger than Paul.
The Need for Variables

Example

*Every* student is younger than *some* instructor.

We use the predicate $S$ to denote student-hood. How do we express “*every student*”?

We need *variables* that can stand for constant values, and a *quantifier* symbol that denotes “*every*”. 
The Need for Variables

Example

*Every* student is younger than *some* instructor.

Using variables and quantifiers, we can write:

\[ \forall x (S(x) \rightarrow (\exists y (I(y) \land Y(x, y)))) \].

Literally: For every \( x \), if \( x \) is a student, then there is some \( y \) such that \( y \) is an instructor and \( x \) is younger than \( y \).
Another Example

English
Not all birds can fly.

Predicates

\[ B(x): \text{x is a bird} \]
\[ F(x): \text{x can fly} \]

The sentence in predicate logic

\[ \neg (\forall x (B(x) \rightarrow F(x))) \]
A Third Example

English
Every girl is younger than her mother.

Predicates

\[
G(x): \text{x is a girl} \\
M(x, y): \text{x is y's mother} \\
Y(x, y): \text{x is younger than y}
\]

The sentence in predicate logic

\[
\forall x \forall y (G(x) \land M(y, x) \rightarrow Y(x, y))
\]
A “Mother” Function

The sentence in predicate logic

$$\forall x \forall y (G(x) \land M(y, x) \to Y(x, y))$$

Note that $y$ is only introduced to denote the mother of $x$.

If everyone has exactly one mother, the predicate $M(y, x)$ is a function, when read from right to left.

We introduce a function symbol $m$ that can be applied to variables and constants as in

$$\forall x (G(x) \to Y(x, m(x)))$$
A Drastic Example

English
Andy and Paul have the same maternal grandmother.

The sentence in predicate logic without functions

$$\forall x \forall y \forall u \forall v (M(x, y) \land M(y, andy) \land M(u, v) \land M(v, paul) \rightarrow x = u)$$

The same sentence in predicate logic with functions

$$m(m(andy)) = m(m(paul))$$
Syntax: We formalize the language of predicate logic, including scoping and substitution.

Semantics: We describe models in which predicates, functions, and formulas have meaning.

Proof theory: We extend natural deduction from propositional to predicate logic (next week)

Further topics: Soundness/completeness, undecidability, incompleteness results, compactness results
1. Syntax of Predicate Logic

2. Predicate Logic as a Formal Language
   - Predicate and Functions Symbols
   - Terms
   - Formulas
   - Variable Binding and Substitution

3. Semantics of Predicate Logic
Predicate Vocabulary

At any point in time, we want to describe the features of a particular “world”, using predicates, functions, and constants. Thus, we introduce for this world:

- a set of predicate symbols $\mathcal{P}$
- a set of function symbols $\mathcal{F}$
Arity of Functions and Predicates

Every function symbol in $\mathcal{F}$ and predicate symbol in $\mathcal{P}$ comes with a fixed arity, denoting the number of arguments the symbol can take.

Special case: Nullary Functions
Function symbols with arity 0 are called *constants*.

Special case: Nullary Predicates
Predicate symbols with arity 0 denotes predicates that do not depend on any arguments. They correspond to propositional atoms.
Terms

\[ t ::= x \mid c \mid f(t, \ldots, t) \]

where

- \(x\) ranges over a given set of variables \(\mathcal{V}\),
- \(c\) ranges over nullary function symbols in \(\mathcal{F}\), and
- \(f\) ranges over function symbols in \(\mathcal{F}\) with arity \(n > 0\).
Examples of Terms

If $n$ is nullary, $f$ is unary, and $g$ is binary, then examples of terms are:

- $g(f(n), n)$
- $f(g(n, f(n)))$
More Examples of Terms

If 0, 1, 2 are nullary (constants), s is unary, and +, − and * are binary, then

\[ *(-2, +(s(x), y)), x) \]

is a term.

Occasionally, we allow ourselves to use infix notation for function symbols as in

\[ (2 - (s(x) + y)) * x \]
Formulas

\[ \phi ::= P(t, \ldots, t) \mid (\neg \phi) \mid (\phi \land \phi) \mid (\phi \lor \phi) \mid (\phi \rightarrow \phi) \mid (\forall x \phi) \mid (\exists x \phi) \]

where
- \( P \in \mathcal{P} \) is a predicate symbol of arity \( n \geq 0 \),
- \( t \) are terms over \( \mathcal{F} \) and \( \mathcal{V} \), and
- \( x \) are variables in \( \mathcal{V} \).
Conventions

Just like for propositional logic, we introduce convenient conventions to reduce the number of parentheses:

- \( \neg, \forall x \) and \( \exists x \) bind most tightly;
- then \( \land \) and \( \lor \);
- then \( \rightarrow \), which is right-associative.
Parse Trees

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

has parse tree

```
        \forall x
           /
          /
         /
        \land
           /
          /
         /
        \rightarrow
           /
            /
           /
          S
            /
           /
          /
         P Q
            /
           /
          /
         /
        x y
            /
           /
          /
         /
        X X
```
Another Example

Every son of my father is my brother.

Predicates

\[ S(x, y): x \text{ is a son of } y \]
\[ B(x, y): x \text{ is a brother of } y \]

Functions

\[ m: \text{ constant for “me”} \]
\[ f(x): \text{ father of } x \]

The sentence in predicate logic

\[ \forall x(S(x, f(m)) \rightarrow B(x, m)) \]

Does this formula hold?
Equality as Predicate

Equality is a common predicate, usually used in infix notation.

\[ = \in \mathcal{P} \]

Example

Instead of the formula

\[ = (f(x), g(x)) \]

we usually write the formula

\[ f(x) = g(x) \]
Free and Bound Variables

Consider the formula

$$\forall x((P(x) \rightarrow Q(x)) \land S(x, y))$$

What is the relationship between variable “binder” $x$ and occurrences of $x$?

\[
\begin{array}{c}
\forall x \\
\land \\
\rightarrow \\
S \\
P \\
Q \\
x \\
y \\
X \\
X
\end{array}
\]
Consider the formula

$$(\forall x (P(x) \land Q(x))) \rightarrow (\neg P(x) \lor Q(y))$$

Which variable occurrences are free; which are bound?
Substitution

Variables are *placeholders*. Replacing them by terms is called *substitution*.

**Definition**

Given a variable $x$, a term $t$ and a formula $\phi$, we define $[x \Rightarrow t]\phi$ to be the formula obtained by replacing each free occurrence of variable $x$ in $\phi$ with $t$.

**Example**

$$[x \Rightarrow f(x, y)]((\forall x (P(x) \land Q(x))) \rightarrow (\neg P(x) \lor Q(y)))$$

$$= \forall x (P(x) \land Q(x))) \rightarrow (\neg P(f(x, y)) \lor Q(y))$$
Example as Parse Tree

\[ [x \Rightarrow f(x, y)]((\forall x (P(x) \land Q(x))) \rightarrow (\neg P(x) \lor Q(y))) \]

\[ = (\forall x (P(x) \land Q(x))) \rightarrow (\neg P(f(x, y)) \lor Q(y)) \]
Example as Parse Tree

\[ \forall x \left( P(x) \land \neg Q(x) \right) \lor \left( \exists y \, P(y) \land f(x, y) \right) \]
Capturing in $[x \Rightarrow t] \phi$

Problem

$t$ contains variable $y$ and $x$ occurs under the scope of $\forall y$ in $\phi$

Example

$[x \Rightarrow f(y, y)](S(x) \land \forall y(P(x) \rightarrow Q(y))))$
Avoiding Capturing

Definition
Given a term \( t \), a variable \( x \) and a formula \( \phi \), we say that \( t \) is free for \( x \) in \( \phi \) if no free \( x \) leaf in \( \phi \) occurs in the scope of \( \forall y \) or \( \exists y \) for any variable \( y \) occurring in \( t \).

Free-ness as precondition
In order to compute \( [x \Rightarrow t] \phi \), we demand that \( t \) is free for \( x \) in \( \phi \).

What if not?
Rename the bound variable!
Example of Renaming

\[ [x \Rightarrow f(y, y)](S(x) \land \forall y(P(x) \rightarrow Q(y))) \]

\[ \Downarrow \]

\[ [x \Rightarrow f(y, y)](S(x) \land \forall z(P(x) \rightarrow Q(z))) \]

\[ \Downarrow \]

\[ S(f(y, y)) \land \forall z(P(f(y, y)) \rightarrow Q(z)) \]
Syntax of Predicate Logic

Predicate Logic as a Formal Language

Semantics of Predicate Logic

Models

Equality

Free Variables

Satisfaction and Entailment
Models

Definition
Let \( \mathcal{F} \) contain function symbols and \( \mathcal{P} \) contain predicate symbols. A model \( \mathcal{M} \) for \( (\mathcal{F}, \mathcal{P}) \) consists of:

1. A non-empty set \( A \), the *universe*;
2. for each nullary function symbol \( f \in \mathcal{F} \) a concrete element \( f^\mathcal{M} \in A \);
3. for each \( f \in \mathcal{F} \) with arity \( n > 0 \), a concrete function \( f^\mathcal{M} : A^n \to A \);
4. for each \( P \in \mathcal{P} \) with arity \( n > 0 \), a function \( P^\mathcal{M} : U^n \to \{F, T\} \).
5. for each \( P \in \mathcal{P} \) with arity \( n = 0 \), a value from \( \{F, T\} \).
Example

Let $\mathcal{F} = \{e, \cdot\}$ and $\mathcal{P} = \{\leq\}$. Let model $\mathcal{M}$ for $(\mathcal{F}, \mathcal{P})$ be defined as follows:

1. Let $A$ be the set of binary strings over the alphabet $\{0, 1\}$;
2. let $e^\mathcal{M} = \epsilon$, the empty string;
3. let $\cdot^\mathcal{M}$ be defined such that $s_1 \cdot^\mathcal{M} s_2$ is the concatenation of the strings $s_1$ and $s_2$; and
4. let $\leq^\mathcal{M}$ be defined such that $s_1 \leq^\mathcal{M} s_2$ iff $s_1$ is a prefix of $s_2$. 
Example (continued)

1. Let \( A \) be the set of binary strings over the alphabet \( \{0, 1\} \);
2. let \( e^M = \epsilon \), the empty string;
3. let \( \cdot^M \) be defined such that \( s_1 \cdot^M s_2 \) is the concatenation of the strings \( s_1 \) and \( s_2 \); and
4. let \( \leq^M \) be defined such that \( s_1 \leq^M s_2 \) iff \( s_1 \) is a prefix of \( s_2 \).

Some Elements of \( A \)
- 10001
- \( \epsilon \)
- \( 1010 \cdot^M 1100 = 10101100 \)
- \( 000 \cdot^M \epsilon = 000 \)
Equality Revisited

Interpretation of equality
Usually, we require that the equality predicate $=\$ is interpreted as same-ness.

Extensionality restriction
This means that allowable models are restricted to those in which $a =^M b$ holds if and only if $a$ and $b$ are the same elements of the model’s universe.
Example (continued)

1. Let $A$ be the set of binary strings over the alphabet $\{0, 1\}$;
2. let $e^M = \epsilon$, the empty string;
3. let $\cdot^M$ be defined such that $s_1 \cdot^M s_2$ is the concatenation of the strings $s_1$ and $s_2$; and
4. let $\leq^M$ be defined such that $s_1 \leq^M s_2$ iff $s_1$ is a prefix of $s_2$.

Equality in $M$

- $000 =^M 000$
- $001 \neq^M 100$
Another Example

Let $\mathcal{F} = \{z, s\}$ and $\mathcal{P} = \{\leq\}$.
Let model $\mathcal{M}$ for $(\mathcal{F}, \mathcal{P})$ be defined as follows:

1. Let $A$ be the set of natural numbers;
2. let $z^\mathcal{M} = 0$;
3. let $s^\mathcal{M}$ be defined such that $s(n) = n + 1$; and
4. let $\leq^\mathcal{M}$ be defined such that $n_1 \leq^\mathcal{M} n_2$ iff the natural number $n_1$ is less than or equal to $n_2$. 
How To Handle Free Variables?

Idea

We can give meaning to formulas with free variables by providing an environment (lookup table) that assigns variables to elements of our universe:

\[ I : \mathcal{V} \rightarrow A. \]

Environment extension

We define environment extension such that \( I[x \mapsto a] \) is the environment that maps \( x \) to \( a \) and any other variable \( y \) to \( I(y) \).
Satisfaction Relation

The model $\mathcal{M}$ satisfies $\phi$ with respect to environment $l$, written $\mathcal{M} \models _l \phi$:

- in case $\phi$ is of the form $P(t_1, t_2, \ldots, t_n)$, if $a_1, a_2, \ldots, a_n$ are the results of evaluating $t_1, t_2, \ldots, t_n$ with respect to $l$, and if $P^\mathcal{M}(a_1, a_2, \ldots, a_n) = T$;
- in case $\phi$ is of the form $P$, if $P^\mathcal{M} = T$;
- in case $\phi$ has the form $\forall x \psi$, if the $\mathcal{M} \models _l [x \mapsto a] \psi$ holds for all $a \in A$;
- in case $\phi$ has the form $\exists x \psi$, if the $\mathcal{M} \models _l [x \mapsto a] \psi$ holds for some $a \in A$;
• in case \( \phi \) has the form \( \neg \psi \), if \( M \models \psi \) does not hold;
• in case \( \phi \) has the form \( \psi_1 \lor \psi_2 \), if \( M \models \psi_1 \) holds or \( M \models \psi_2 \) holds;
• in case \( \phi \) has the form \( \psi_1 \land \psi_2 \), if \( M \models \psi_1 \) holds and \( M \models \psi_2 \) holds; and
• in case \( \phi \) has the form \( \psi_1 \rightarrow \psi_2 \), if \( M \models \psi_2 \) holds whenever \( M \models \psi_1 \) holds.
Satisfaction of Closed Formulas

If a formula $\phi$ has no free variables, we call $\phi$ a *sentence*. $\mathcal{M} \models_I \phi$ holds or does not hold regardless of the choice of $I$. Thus we write $\mathcal{M} \models \phi$ or $\mathcal{M} \not\models \phi$. 
Semantic Entailment and Satisfiability

Let $\Gamma$ be a possibly infinite set of formulas in predicate logic and $\psi$ a formula.

Entailment

$\Gamma \models \psi$ iff for all models $\mathcal{M}$ and environments $l$, whenever $\mathcal{M} \models_l \phi$ holds for all $\phi \in \Gamma$, then $\mathcal{M} \models_l \psi$.

Satisfiability of Formulas

$\psi$ is satisfiable iff there is some model $\mathcal{M}$ and some environment $l$ such that $\mathcal{M} \models_l \psi$ holds.

Satisfiability of Formula Sets

$\Gamma$ is satisfiable iff there is some model $\mathcal{M}$ and some environment $l$ such that $\mathcal{M} \models_l \phi$, for all $\phi \in \Gamma$. 
Semantic Entailment and Satisfiability

Let $\Gamma$ be a possibly infinite set of formulas in predicate logic and $\psi$ a formula.

Validity

$\psi$ is valid iff for all models $\mathcal{M}$ and environments $I$, we have $\mathcal{M} \models_I \psi$. 
The Problem with Predicate Logic

Entailment ranges over models

Semantic entailment between sentences: $\phi_1, \phi_2, \ldots, \phi_n \models \psi$
requires that in all models that satisfy $\phi_1, \phi_2, \ldots, \phi_n$, the sentence $\psi$ is satisfied.

How to effectively argue about all possible models?

Usually the number of models is infinite; it is very hard to argue on the semantic level in predicate logic.

Idea from propositional logic

Can we use natural deduction for showing entailment?
Coq Homework 2: out on module homepage; due 10/9, 9:30pm
Assignment 3: out soon; due 9/9, 11:00am
Monday, Wednesday: Office hours
Tuesday: Tutorials (Assignments 2 and 3)
Wednesday: Labs (Quiz 1 solution, Coq Homework 2)
Thursday: Lecture on