COMP2111 Week 4
Term 1, 2019
Predicate Logic I
Summary of topics

- Re-introduction to Predicate Logic
- Syntax of Predicate Logic
- Semantics of Predicate Logic
- Natural Deduction for Predicate Logic
Summary of topics

- Re-introduction to Predicate Logic
- Syntax of Predicate Logic
- Semantics of Predicate Logic
- Natural Deduction for Predicate Logic
Motivation

Predicate logic adds *expressiveness* to Propositional Logic.

- Examine how/why a proposition is true
- Define relationships between propositions
Motivating example

Consider the statement:

For all \( x, y \in X : (y = x + 1) \rightarrow (x \leq y) \)
Motivating example

Consider the statement:

For all $x, y \in X : (y = x+1) \rightarrow (x \leq y)$

$X = \{1, 2, 3\} : 18$ propositional variables:

$P_{11} = "1 = 1 + 1" \quad S_{11} = "1 \leq 1"$

$P_{12} = "2 = 1 + 1" \quad S_{12} = "1 \leq 2"

$\vdots \quad \vdots \quad \vdots \quad \vdots$

Final result: $(P_{11} \rightarrow S_{11}) \land (P_{12} \rightarrow S_{12}) \land \cdots \land (P_{33} \rightarrow S_{33})$

**NB**

“Normal arithmetic”, where $P_{11}$ is false, $P_{12}$ is true, etc is one of many possibilities.
Motivating example

Consider the statement:

For all $x, y \in X : (y = x + 1) \rightarrow (x \leq y)$

$X = \mathbb{N} : \infty$ propositional variables:

$P_{00} = "0 = 0 + 0" \quad S_{00} = "0 \leq 0"

P_{01} = "1 = 0 + 1" \quad S_{01} = "0 \leq 1"

\vdots \quad \vdots \quad \vdots \quad \vdots

Final result: $(P_{00} \rightarrow S_{00}) \land (P_{01} \rightarrow S_{01}) \land \cdots$ Not permitted!
Motivating example

Consider the statement:

\[ \text{For all } x, y \in X : (y = x+1) \rightarrow (x \leq y) \]

\[ X = \mathbb{N} : \infty \text{ propositional variables:} \]

\[ P_{00} = "0 = 0 + 0" \quad S_{00} = "0 \leq 0" \]
\[ P_{01} = "1 = 0 + 1" \quad S_{01} = "0 \leq 1" \]
\[ \vdots \quad \vdots \quad \vdots \quad \vdots \]

Final result: \((P_{00} \rightarrow S_{00}) \land (P_{01} \rightarrow S_{01}) \land \cdots \) Not permitted!
Motivating example

Consider the statement:

For all $x, y \in X : (y = x+1) \rightarrow (x \leq y)$

Predicate logic introduces:

- Predicates
- Functions
- Constants
- Variables, and
- Quantifiers
Motivating example

Consider the statement:

For all $x, y \in X : (y = x + 1) \rightarrow (x \leq y)$

Predicate logic introduces:

- **Predicates**
- **Functions**
- **Constants**
- **Variables, and**
- **Quantifiers**
Motivating example

Consider the statement:

For all $x, y \in X : (y = x+1) \rightarrow (x \leq y)$

Predicate logic introduces:

- Predicates
- Functions
- Constants
- Variables, and
- Quantifiers
Motivating example

Consider the statement:

\[ \text{For all } x, y \in X : (y = x+1) \rightarrow (x \leq y) \]

Predicate logic introduces:

- Predicates
- Functions
- Constants
- Variables, and
- Quantifiers
Motivating example

Consider the statement:

For all \( x, y \in X \): \((y = x + 1) \rightarrow (x \leq y)\)

Predicate logic introduces:

- Predicates
- Functions
- Constants
- Variables, and
- Quantifiers
Domain of discourse

Fundamental to interpreting formulas is the domain of discourse: the set of “ground objects” that we are referring to.

- Predicates: Relations on the domain
- Functions: Operators on the domain
- Constants: “Named” elements of the domain
- Variables: “Unnamed” elements of the domain (placeholders for elements)
- Quantifiers: Range over domain elements

Example

Consider: $\forall x C(x)$ where $C(x)$ represents “$x$ studies COMP2111”
It is true if the domain of discourse is the set of students in this room.
Domain of discourse

Fundamental to interpreting formulas is the domain of discourse: the set of “ground objects” that we are referring to.

- **Predicates**: Relations on the domain
- **Functions**: Operators on the domain
- **Constants**: “Named” elements of the domain
- **Variables**: “Unnamed” elements of the domain (placeholders for elements)
- **Quantifiers**: Range over domain elements

**Example**

Consider: $\forall x C(x)$ where $C(x)$ represents “$x$ studies COMP2111”

It is true if the domain of discourse is the set of students in this room.
Domain of discourse

Fundamental to interpreting formulas is the **domain of discourse**: the set of “ground objects” that we are referring to.

- **Predicates**: Relations on the domain
- **Functions**: Operators on the domain
- **Constants**: “Named” elements of the domain
- **Variables**: “Unnamed” elements of the domain (placeholders for elements)
- **Quantifiers**: Range over domain elements

**Example**

Consider: $\forall x C(x)$ where $C(x)$ represents “$x$ studies COMP2111”.

It is true if the domain of discourse is the set of students in this room.
Domain of discourse

Fundamental to interpreting formulas is the **domain of discourse**: the set of “ground objects” that we are referring to.

- **Predicates**: Relations on the domain
- **Functions**: Operators on the domain
- **Constants**: “Named” elements of the domain
- **Variables**: “Unnamed” elements of the domain (placeholders for elements)
- **Quantifiers**: Range over domain elements

**Example**

Consider: \( \forall x \, C(x) \) where \( C(x) \) represents “\( x \) studies COMP2111”. It is true if the domain of discourse is the set of students in this room.
Domain of discourse

Fundamental to interpreting formulas is the **domain of discourse**: the set of “ground objects” that we are referring to.

- **Predicates**: Relations on the domain
- **Functions**: Operators on the domain
- **Constants**: “Named” elements of the domain
- **Variables**: “Unnamed” elements of the domain (placeholders for elements)
- **Quantifiers**: Range over domain elements

**Example**

Consider: $\forall x C(x)$ where $C(x)$ represents “$x$ studies COMP2111”. It is true if the domain of discourse is the set of students in this room.
Domain of discourse

Fundamental to interpreting formulas is the **domain of discourse**: the set of “ground objects” that we are referring to.

- **Predicates**: Relations on the domain
- **Functions**: Operators on the domain
- **Constants**: “Named” elements of the domain
- **Variables**: “Unnamed” elements of the domain (placeholders for elements)
- **Quantifiers**: Range over domain elements

**Example**

Consider: \( \forall x C(x) \) where \( C(x) \) represents “\( x \) studies COMP2111”

It is **true** if the domain of discourse is the set of students in this room.
Domain of discourse

Fundamental to interpreting formulas is the **domain of discourse**: the set of “ground objects” that we are referring to.

- **Predicates**: Relations on the domain
- **Functions**: Operators on the domain
- **Constants**: “Named” elements of the domain
- **Variables**: “Unnamed” elements of the domain (placeholders for elements)
- **Quantifiers**: Range over domain elements

**Example**

Consider: $\forall x C(x)$ where $C(x)$ represents “$x$ studies COMP2111”

It is false if the domain of discourse is the set of students at UNSW.
Multiple domains of discourse

Is it possible to have multiple domains? Yes!

For example: the predicate studies\((x, y)\) representing “\(x\) (a student) studies \(y\) (a subject)”. 

- Take \(\text{STUDENTS} \cup \text{SUBJECTS}\) as the domain.
- Use unary predicates, e.g. \(\text{isStudent}(x)\), to restrict the domain.
- To restrict quantifiers (applies to any subset of the domain defined by a unary predicate):
  - \(\forall x \in \text{STUDENTS} : \phi \) is equivalent to: \(\forall x (\text{isStudent}(x) \rightarrow \phi)\)
  - \(\exists x \in \text{STUDENTS} : \phi \) is equivalent to: \(\exists x (\text{isStudent}(x) \land \phi)\)
Multiple domains of discourse

Is it possible to have multiple domains? Yes!

For example: the predicate $\text{studies}(x, y)$ representing “$x$ (a student) studies $y$ (a subject)”.

- Take $\text{STUDENTS} \cup \text{SUBJECTS}$ as the domain.
- Use unary predicates, e.g. $\text{isStudent}(x)$, to restrict the domain.
- To restrict quantifiers (applies to any subset of the domain defined by a unary predicate):
  - $\exists x \in \text{STUDENTS}$. $\phi$ is equivalent to: $\exists x (\text{isStudent}(x) \land \phi)$
  - $\forall x \in \text{STUDENTS}$. $\phi$ is equivalent to: $\forall x (\text{isStudent}(x) \rightarrow \phi)$
Multiple domains of discourse

Is it possible to have multiple domains? Yes!

For example: the predicate $\text{studies}(x, y)$ representing “$x$ (a student) studies $y$ (a subject)”.

- Take $\text{Students} \cup \text{Subjects}$ as the domain.
- Use unary predicates, e.g. $\text{isStudent}(x)$, to restrict the domain.
- To restrict quantifiers (applies to any subset of the domain defined by a unary predicate):
  - $\exists x \in \text{Students} : \varphi$ is equivalent to: $\exists x(\text{isStudent}(x) \land \varphi)$
  - $\forall x \in \text{Students} : \varphi$ is equivalent to: $\forall x(\text{isStudent}(x) \rightarrow \varphi)$
Function outputs, constants, and variables are interpreted as elements of the domain.

Predicates are truth-functional: they map elements of the domain to true or false.

Quantifiers (and the Boolean connectives) are predicate operators: they transform predicates into other predicates.
Example

Consider the following predicates and constants:

- \( K(x, y) \): \( x \) knows \( y \)
- \( S(x, y) \): \( x \) is not the son of \( y \)
- \( F(x, y) \): the fact that \( x \) is not the son of \( y \) (functional)

- \( J \): Jon Snow
- \( N \): Ned Stark
- \( B \): Bran Stark

Domain of discourse: \( \text{People} \cup \text{Facts} \)

The following are OK:

- \( S(B, J) \): Bran is not the son of Jon
- \( K(N, J) \): Ned knows Jon
- \( \forall x \neg K(J, x) \): Jon Snow knows nothing.

This is not:

- \( K(B, S(J, N)) \): Bran knows that Jon is not the son of Ned
Example

Consider the following predicates and constants:

\[ K(x, y): \text{x knows y} \]
\[ S(x, y): \text{x is not the son of y} \]
\[ F(x, y): \text{the fact that x is not the son of y (functional)} \]

\[ J: \text{Jon Snow} \]
\[ N: \text{Ned Stark} \]
\[ B: \text{Bran Stark} \]

Domain of discourse: \text{People} \cup \text{Facts}

The following are OK:

- \[ S(B, J): \text{Bran is not the son of Jon} \]
- \[ K(N, J): \text{Ned knows Jon} \]
- \[ \forall x \neg K(J, x): \text{Jon Snow knows no-one.} \]

This is not:

- \[ K(B, S(J, N)): \text{Bran knows that Jon is not the son of Ned} \]
Example

Consider the following predicates and constants:

\[ K(x, y): \ x \text{ knows } y \]
\[ S(x, y): \ x \text{ is not the son of } y \]
\[ F(x, y): \ \text{the fact that } x \text{ is not the son of } y \text{ (functional)} \]
\[ J: \text{ Jon Snow} \]
\[ N: \text{ Ned Stark} \]
\[ B: \text{ Bran Stark} \]

Domain of discourse: \text{ People } \cup \text{ Facts }

The following are OK:

- \[ S(B, J): \text{ Bran is not the son of Jon } \]
- \[ K(N, J): \text{ Ned knows Jon } \]
- \[ \forall x \neg K(J, x): \text{ Jon Snow knows no-one.} \]

This is not:

- \[ K(B, S(J, N)): \text{ Bran knows that Jon is not the son of Ned} \]
Example

Consider the following predicates and constants:

\[ K(x, y) : \text{x knows y} \]
\[ S(x, y) : \text{x is not the son of y} \]
\[ F(x, y) : \text{the fact that x is not the son of y (functional)} \]
\[ J : \text{Jon Snow} \]
\[ N : \text{Ned Stark} \]
\[ B : \text{Bran Stark} \]

Domain of discourse: \textbf{People} \cup \textbf{Facts}

The following are OK:

- \[ S(B, J) : \text{Bran is not the son of Jon} \]
- \[ K(N, J) : \text{Ned knows Jon} \]
- \[ \forall x \neg K(J, x) : \text{Jon Snow knows no-one.} \]

This is OK:

- \[ K(B, F(J, N)) : \text{Bran knows that Jon is not the son of Ned} \]
Example

Consider the following predicates and constants:

\[ K(x, y): \text{x knows y} \]
\[ S(x, y): \text{x is not the son of y} \]
\[ F(x, y): \text{the fact that x is not the son of y (functional)} \]
\[ J: \text{Jon Snow} \]
\[ N: \text{Ned Stark} \]
\[ B: \text{Bran Stark} \]

Domain of discourse: \text{People} \cup \text{Facts}

The following are OK:

- \[ S(B, J): \text{Bran is not the son of Jon} \]
- \[ K(N, J): \text{Ned knows Jon} \]
- \[ \forall x \neg K(J, x): \text{Jon Snow knows nothing.} \]

This is OK:

- \[ K(B, F(J, N)): \text{Bran knows that Jon is not the son of Ned} \]
Summary of topics

- Re-introduction to Predicate Logic
- Syntax of Predicate Logic
- Semantics of Predicate Logic
- Natural Deduction for Predicate Logic
A **vocabulary** indicates what **predicates**, **functions** and **constants** we can use to build up our formulas. Very similar to C header files, or Java interfaces.

A vocabulary $V$ is a set of:

- Predicate “symbols” $P, Q, \ldots$, each with an associated *arity* (number of arguments)
- Function “symbols” $f, g, \ldots$, each with an associated *arity* (number of arguments)
- Constant “symbols” $c, d, \ldots$ (also known as 0-arity functions)

**Example**

$V = \{\leq, +, 1\}$ where $\leq$ is a binary predicate symbol, $+$ is a binary function symbol, and $1$ is a constant symbol.
A **vocabulary** indicates what **predicates**, **functions** and **constants** we can use to build up our formulas. Very similar to C header files, or Java interfaces.

A vocabulary $V$ is a set of:

- Predicate “symbols” $P, Q, \ldots$, each with an associated **arity** (number of arguments)
- Function “symbols” $f, g, \ldots$, each with an associated **arity** (number of arguments)
- Constant “symbols” $c, d, \ldots$ (also known as 0-arity functions)

**Example**

$V = \{ \leq, +, 1 \}$ where $\leq$ is a binary predicate symbol, $+$ is a binary function symbol, and $1$ is a constant symbol.
A **term** is defined recursively as follows:

- A variable is a term
- A constant symbol is a term
- If $f$ is a function symbol with arity $k$, and $t_1, \ldots, t_k$ are terms, then $f(t_1, t_2, \ldots, t_k)$ is a term.

**NB**

*Terms will be interpreted as elements of the domain of discourse.*
Formulas

A formula of Predicate Logic is defined recursively as follows:

- If $P$ is a predicate symbol with arity $k$, and $t_1, \ldots, t_k$ are terms, then $P(t_1, t_2, \ldots, t_k)$ is a formula.
- If $t_1$ and $t_2$ are terms then $(t_1 = t_2)$ is a formula.
- If $\varphi, \psi$ are formulas then the following are formulas:
  - $\neg \varphi$
  - $(\varphi \land \psi)$
  - $(\varphi \lor \psi)$
  - $(\varphi \to \psi)$
  - $(\varphi \leftrightarrow \psi)$
  - $\forall x \varphi$
  - $\exists x \varphi$

NB

The base cases are known as atomic formulas: they play a similar role in the parse tree as propositional variables.
∀x∀y((y = x + 1) → (x ≤ y))
Free and Bound variables

A variable is **bound** to the closest matching quantifier that lies above it in the parse tree. A variable that is not bound is **free**.

Example

In $(\forall x \exists z \exists x P(x, y, z)) \land Q(x)$:

- $z$ is bound to $\exists z$
- $y$ is free

First $x$ is bound to $\exists x$

Second $x$ is free

A formula with no free variables is a **sentence**.
A variable is **bound** to the closest matching quantifier that lies above it in the parse tree. A variable that is not bound is **free**.

**Example**

In \((\forall x \exists z \exists x P(x, y, z)) \land Q(x)\):

- \(z\) is bound to \(\exists z\)
- \(y\) is free
- First \(x\) is bound to \(\exists x\)
- Second \(x\) is free

A formula with no free variables is a **sentence**.
Free and Bound variables

A variable is **bound** to the closest matching quantifier that lies above it in the parse tree. A variable that is not bound is **free**.

**Example**

In $$(\forall x \exists z \exists x P(x, y, z)) \land Q(x)$$:

- $z$ is bound to $\exists z$
- $y$ is free
- First $x$ is bound to $\exists x$
- Second $x$ is free

A formula with no free variables is a **sentence**.
Free and Bound variables

A variable is **bound** to the closest matching quantifier that lies above it in the parse tree. A variable that is not bound is **free**.

**Example**

In $$(\forall x \exists z \exists x P(x, y, z)) \land Q(x)$$:
- $z$ is bound to $\exists z$
- $y$ is free
- First $x$ is bound to $\exists x$
- Second $x$ is free

A formula with no free variables is a **sentence**.
Free and Bound variables

A variable is **bound** to the closest matching quantifier that lies above it in the parse tree. A variable that is not bound is **free**.

**Example**

In \((\forall x \exists z \exists x P(x, y, z)) \land Q(x)\):
- \(z\) is bound to \(\exists z\)
- \(y\) is free
- First \(x\) is bound to \(\exists x\)
- Second \(x\) is free

A formula with no free variables is a **sentence**.
Free and Bound variables

A variable is **bound** to the closest matching quantifier that lies above it in the parse tree. A variable that is not bound is **free**.

**Example**

In \( (\forall x \exists z \exists x P(x, y, z)) \land Q(x) \):

- \( z \) is bound to \( \exists z \)
- \( y \) is free
- First \( x \) is bound to \( \exists x \)
- Second \( x \) is free

A formula with no free variables is a sentence.
Free and Bound variables

A variable is **bound** to the closest matching quantifier that lies above it in the parse tree. A variable that is not bound is **free**.

**Example**

In \((\forall x \exists z \exists x P(x, y, z)) \land Q(x)\):

- \(z\) is bound to \(\exists z\)
- \(y\) is free
- First \(x\) is bound to \(\exists x\)
- Second \(x\) is free

A formula with no free variables is a **sentence**.
Free variables formally

We can define the set of free variables recursively on the structure of a formula:

- \( FV(x) = \{x\} \) for all variables \( x \)
- \( FV(c) = \emptyset \) for all constants \( c \)
- \( FV(f(t_1, \ldots, t_k)) = FV(t_1) \cup \cdots \cup FV(t_k) \) for all \( k \)-ary functions \( f \)
- \( FV(P(t_1, \ldots, t_k)) = FV(t_1) \cup \cdots \cup FV(t_k) \) for all \( k \)-ary predicates \( P \)
- \( FV(t_1 = t_2) = FV(t_1) \cup FV(t_2) \)
- \( FV(\neg \varphi) = FV(\varphi) \)
- \( FV(\psi \land \varphi) = FV(\psi \lor \varphi) = FV(\psi \rightarrow \varphi) = FV(\psi \leftrightarrow \varphi) = FV(\psi) \cup FV(\varphi) \)
- \( FV(\forall x \varphi) = FV(\exists x \varphi) = FV(\varphi) \setminus \{x\} \)
We can define the set of free variables recursively on the structure of a formula:

- \( FV(x) = \{x\} \) for all variables \( x \)
- \( FV(c) = \emptyset \) for all constants \( c \)
- \( FV(f(t_1, \ldots, t_k)) = FV(t_1) \cup \cdots \cup FV(t_k) \) for all \( k \)-ary functions \( f \)
- \( FV(P(t_1, \ldots, t_k)) = FV(t_1) \cup \cdots \cup FV(t_k) \) for all \( k \)-ary predicates \( P \)
- \( FV(t_1 = t_2) = FV(t_1) \cup FV(t_2) \)
- \( FV(\neg \varphi) = FV(\varphi) \)
- \( FV(\psi \land \varphi) = FV(\psi \lor \varphi) = FV(\psi \rightarrow \varphi) = FV(\psi \leftrightarrow \varphi) = FV(\psi) \cup FV(\varphi) \)
- \( FV(\forall x \varphi) = FV(\exists x \varphi) = FV(\varphi) \setminus \{x\} \)
Free variables formally

We can define the set of free variables recursively on the structure of a formula:

- $FV(x) = \{x\}$ for all variables $x$
- $FV(c) = \emptyset$ for all constants $c$
- $FV(f(t_1, \ldots, t_k)) = FV(t_1) \cup \cdots \cup FV(t_k)$ for all $k$-ary functions $f$
- $FV(P(t_1, \ldots, t_k)) = FV(t_1) \cup \cdots \cup FV(t_k)$ for all $k$-ary predicates $P$
- $FV(t_1 = t_2) = FV(t_1) \cup FV(t_2)$
- $FV(\neg \varphi) = FV(\varphi)$
- $FV(\psi \land \varphi) = FV(\psi \lor \varphi) = FV(\psi \rightarrow \varphi) = FV(\psi \leftrightarrow \varphi) = FV(\psi) \cup FV(\varphi)$
- $FV(\forall x \varphi) = FV(\exists x \varphi) = FV(\varphi) \setminus \{x\}$
Free variables formally

We can define the set of free variables recursively on the structure of a formula:

- $\text{FV}(x) = \{x\}$ for all variables $x$
- $\text{FV}(c) = \emptyset$ for all constants $c$
- $\text{FV}(f(t_1, \ldots, t_k)) = \text{FV}(t_1) \cup \cdots \cup \text{FV}(t_k)$ for all $k$-ary functions $f$
- $\text{FV}(P(t_1, \ldots, t_k)) = \text{FV}(t_1) \cup \cdots \cup \text{FV}(t_k)$ for all $k$-ary predicates $P$
- $\text{FV}(t_1 = t_2) = \text{FV}(t_1) \cup \text{FV}(t_2)$
- $\text{FV}(\neg \varphi) = \text{FV}(\varphi)$
- $\text{FV}(\psi \land \varphi) = \text{FV}(\psi \lor \varphi) = \text{FV}(\psi \rightarrow \varphi) = \text{FV}(\psi \leftrightarrow \varphi) = \text{FV}(\psi) \cup \text{FV}(\varphi)$
- $\text{FV}(\forall x \varphi) = \text{FV}(\exists x \varphi) = \text{FV}(\varphi) \setminus \{x\}$
Substitution

If $t$ is a term, $\varphi$ a formula, and $x \in FV(\varphi)$, then the substitution of $t$ for $x$ in $\varphi$ (denoted $\varphi[t/x]$) is the formula obtained by replacing every free occurrence of $x$ with $t$.

It can be useful to have “access” to the free variables of a formula. So if $x_1, \ldots, x_k$ are the free variables of $\varphi$, we may denote this as $\varphi(x_1, \ldots, x_k)$. Substitution can be easily presented: $\varphi(t)$ for $\varphi(x)[t/x]$.

Note

Variable names matter: $\varphi(x)$ and $\varphi(y)$ are different formulas!
Substitution

If $t$ is a term, $\varphi$ a formula, and $x \in FV(\varphi)$, then the substitution of $t$ for $x$ in $\varphi$ (denoted $\varphi[t/x]$) is the formula obtained by replacing every free occurrence of $x$ with $t$.

It can be useful to have “access” to the free variables of a formula. So if $x_1, \ldots, x_k$ are the free variables of $\varphi$, we may denote this as $\varphi(x_1, \ldots, x_k)$. Substitution can be easily presented: $\varphi(t)$ for $\varphi(x)[t/x]$.

Note

Variable names matter: $\varphi(x)$ and $\varphi(y)$ are different formulas!
Summary of topics

- Re-introduction to Predicate Logic
- Syntax of Predicate Logic
- Semantics of Predicate Logic
- Natural Deduction for Predicate Logic
Models

Predicate formulas are interpreted in **Models**.

Given a vocabulary $V$ a model $\mathcal{M}$ defines:

- A (non-empty) domain $D = \text{Dom}(\mathcal{M})$
- For every predicate symbol $P \in V$ with arity $k$: a $k$-ary relation $P^\mathcal{M}$ on $D$
- For every function symbol $f \in V$ with arity $k$: a function $f^\mathcal{M} : D^k \rightarrow D$
- For every constant symbol $c \in V$: an element, $c^\mathcal{M}$ of $D$

**Example**

For the vocabulary $V = \{\leq, +, 1\}$: one model could be $\mathbb{N}$ with the standard definitions.
Models

Predicate formulas are interpreted in **Models**.

Given a vocabulary \( V \) a model \( \mathcal{M} \) defines:

- A (non-empty) domain \( D = \text{Dom}(\mathcal{M}) \)
- For every predicate symbol \( P \in V \) with arity \( k \): a \( k \)-ary relation \( P^\mathcal{M} \) on \( D \)
- For every function symbol \( f \in V \) with arity \( k \): a function \( f^\mathcal{M} : D^k \to D \)
- For every constant symbol \( c \in V \): an element, \( c^\mathcal{M} \) of \( D \)

**Example**

For the vocabulary \( V = \{ \leq, +, 1 \} \): one model could be \( \mathbb{N} \) with the standard definitions.
Environments

Given a model $\mathcal{M}$, an environment (or lookup table), $\eta$, is a function from the set of variables to $\text{Dom}(\mathcal{M})$.

Given an environment $\eta$, we denote by $\eta[\mathcal{x} \mapsto c]$ the environment that agrees with $\eta$ everywhere except possibly at $x$ (where it has value $c$).
Environments

Given a model $\mathcal{M}$, an environment (or lookup table), $\eta$, is a function from the set of variables to $\text{Dom}(\mathcal{M})$.

Given an environment $\eta$, we denote by $\eta[x \mapsto c]$ the environment that agrees with $\eta$ everywhere except possibly at $x$ (where it has value $c$).
Interpretations

An interpretation is a pair $(\mathcal{M}, \eta)$ where $\mathcal{M}$ is a model and $\eta$ is an environment.
Interpretations

An interpretation is a pair \((\mathcal{M}, \eta)\) where \(\mathcal{M}\) is a model and \(\eta\) is an environment.

An interpretation \((\mathcal{M}, \eta)\) maps terms to elements of \(\text{Dom}(\mathcal{M})\) recursively as follows:

- \([x]_\mathcal{M}^\eta = \eta(x)\)
- \([c]_\mathcal{M}^\eta = c^\mathcal{M}\)
- \([f(t_1, \ldots, t_k)]_\mathcal{M}^\eta = f^\mathcal{M}([t_1]_\mathcal{M}^\eta, \ldots, [t_k]_\mathcal{M}^\eta)\)
Interpretations

An interpretation is a pair \((M, \eta)\) where \(M\) is a model and \(\eta\) is an environment.

An interpretation \((M, \eta)\) maps formulas to \(\mathbb{B}\) recursively as follows:

- \([P(t_1, \ldots, t_k)]^\eta_M = \text{true}\) if \(P^M([t_1]^\eta_M, \ldots, [t_k]^\eta_M)\) holds.
- \([t_1 = t_2]^\eta_M = \text{true}\) if \([t_1]^\eta_M = [t_2]^\eta_M\)
- \([\forall x \varphi]^\eta_M = \text{true}\) if \([\varphi]^{[x \mapsto c]}_M = \text{true}\) for all \(c \in \text{Dom}(M)\)
- \([\exists x \varphi]^\eta_M = \text{true}\) if \([\varphi]^{[x \mapsto c]}_M = \text{true}\) for some \(c \in \text{Dom}(M)\)
- \([\varphi]^\eta_M\) defined in the same way as Propositional Logic for all other formulas \(\varphi\).
Example

\[ \forall x \forall y ((y = x + 1) \rightarrow (x \leq y)) \]

- \(<\mathbb{N}, \leq, +, 1>\): true
- \(<\mathbb{N}, >, +, 1>\): false
- \(<\{0\}, \{(0, 0)\}, +, 0>\): true
Example

\[ \forall x \forall y ((y = x + 1) \rightarrow (x \leq y)) \]

- \( \langle \mathbb{N}, \leq, +, 1 \rangle \): true
- \( \langle \mathbb{N}, >, +, 1 \rangle \): false
- \( \langle \{0\}, \{(0,0)\}, +, 0 \rangle \): true
Example

\[ \forall x \forall y ((y = x + 1) \rightarrow (x \leq y)) \]

- \( \langle \mathbb{N}, \leq, +, 1 \rangle \): true
- \( \langle \mathbb{N}, >, +, 1 \rangle \): false
- \( \langle \{0\}, \{(0, 0)\}, +, 0 \rangle \): true
Example

\[ \forall x \forall y ((y = x + 1) \rightarrow (x \leq y)) \]

- \( \langle \mathbb{N}, \leq, +, 1 \rangle \): true
- \( \langle \mathbb{N}, >, +, 1 \rangle \): false
- \( \langle \{0\}, \{(0,0)\}, +, 0 \rangle \): true
Example

\[ \forall x \forall y ((y = x + 1) \rightarrow (x \leq y)) \]

- \( \langle \mathbb{N}, \leq, +, 1 \rangle \): true
- \( \langle \mathbb{N}, >, +, 1 \rangle \): false
- \( \langle \{0\}, \{(0,0)\}, +, 0 \rangle \): true
Example

\[ \forall x \forall y ((y = x + 1) \rightarrow (x \leq y)) \]

- \( \langle \mathbb{N}, \leq, +, 1 \rangle \): true
- \( \langle \mathbb{N}, >, +, 1 \rangle \): false
- \( \langle \{0\}, \{(0,0)\}, +, 0 \rangle \): true