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:

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

Motivating example

Consider the statement:

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

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

$$\begin{array}{llll} P_{11} & = \text{“}1 = 1 + 1\text{”} & S_{11} & = \text{“}1 \leq 1\text{”} \\ P_{12} & = \text{“}2 = 1 + 1\text{”} & S_{12} & = \text{“}1 \leq 2\text{”} \\ \vdots & \vdots & \vdots & \vdots \end{array}$$

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

NB

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

Motivating example

Consider the statement:

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

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

$$\begin{array}{llll} P_{00} & = \text{“}0 = 0 + 0\text{”} & S_{00} & = \text{“}0 \leq 0\text{”} \\ P_{01} & = \text{“}1 = 0 + 1\text{”} & S_{01} & = \text{“}0 \leq 1\text{”} \\ \vdots & \vdots & \vdots & \vdots \end{array}$$

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

Motivating example

Consider the statement:

$$\text{For all } x, y \in X : (y = x{+}1) \to (x \leq y)$$

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

$$\begin{array}{llll} P_{00} = \text{“}0 = 0 + 0\text{”} & S_{00} = \text{“}0 \leq 0\text{”} \\ P_{01} = \text{“}1 = 0 + 1\text{”} & S_{01} = \text{“}0 \leq 1\text{”} \\ \vdots \quad \vdots & \vdots \quad \vdots \end{array}$$

Final result: $(P_{00} \to S_{00}) \wedge (P_{01} \to S_{01}) \wedge \cdots$ Not permitted!

Motivating example

Consider the statement:

$$\text{For all } x, y \in X : (y = x{+}1) \to (x \leq y)$$

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:

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

Predicate logic introduces:

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

Motivating example

Consider the statement:

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

Predicate logic introduces:

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

Motivating example

Consider the statement:

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

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 $\textsc{Students} \cup \textsc{Subjects}$ as the domain.
- Use unary predicates, e.g. isStudent(x), to restrict the domain.
- To restrict quantifiers (applies to any subset of the domain defined by a unary predicate):

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 $\textsc{Students} \cup \textsc{Subjects}$ as the domain.
- Use unary predicates, e.g. isStudent(x), to restrict the domain.
- To restrict quantifiers (applies to any subset of the domain defined by a unary predicate):
  - $\exists x \in \textsc{Students} \cdot \varphi$ is equivalent to: $\exists x(\text{isStudent}(x) \wedge \varphi)$
  - $\forall x \in \textsc{Students} \cdot \varphi$ is equivalent to: $\forall x(\text{isStudent}(x) \rightarrow \varphi)$

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 $\textsc{Students} \cup \textsc{Subjects}$ 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):
  - $\exists x \in \textsc{Students} : \varphi$ is equivalent to: $\exists x(\text{isStudent}(x) \wedge \varphi)$
  - $\forall x \in \textsc{Students} : \varphi$ is equivalent to: $\forall x(\text{isStudent}(x) \rightarrow \varphi)$

Domain of discourse

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:

$K(x, y)$:	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: $\textsc{People} \cup \textsc{Facts}$

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 no-one.

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:

$K(x, y)$:	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: $\textsc{People} \cup \textsc{Facts}$

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 no-one.

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:

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

Domain of discourse: $\textsc{People} \cup \textsc{Facts}$

The following are OK:

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

This is OK:

- $\mathsf{K}(\mathsf{B}, \mathsf{F}(\mathsf{J}, \mathsf{N}))$: Bran knows that Jon is not the son of Ned

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: $\textsc{People} \cup \textsc{Facts}$

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 OK:

- $K(B, F(J, N))$: 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

Vocabulary

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, . . . , each with an assoicated *arity* (number of arguments)
- Function "symbols" f, g, . . . , each with an assoicated *arity* (number of arguments)
- Constant "symbols" c, d, . . . (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.

Vocabulary

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, . . . , each with an assoicated *arity* (number of arguments)
- Function "symbols" f, g, . . . , each with an assoicated *arity* (number of arguments)
- Constant "symbols" c, d, . . . (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.

Terms

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, ..., 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, ..., 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 φ, ψ are a formulas then the following are formulas:
  - $\neg\varphi$
  - $(\varphi \wedge \psi)$
  - $(\varphi \vee \psi)$
  - $(\varphi \rightarrow \psi)$
  - $(\varphi \leftrightarrow \psi)$
  - $\forall x \varphi$
  - $\exists x \varphi$

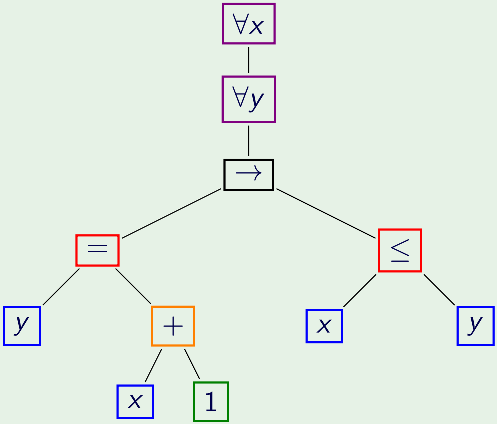
NB

The base cases are known as **atomic** *formulas: they play a similar role in the parse tree as propositional variables.*

Parse trees

Example

$$\forall x \forall y((y = x + 1) \rightarrow (x \leq y))$$



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

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

- 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 $(\forall x \exists z \exists x P(x, y, z)) \wedge Q(x)$:

- 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 $(\forall x \exists z \exists x P(x, y, z)) \wedge Q(x)$:

- 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 $(\forall x \exists z \exists x P(x, y, z)) \wedge Q(x)$:

- 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 $(\forall x \exists z \exists x P(x, y, z)) \wedge Q(x)$:

- 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 $(\forall x \exists z \exists x P(x, y, z)) \wedge Q(x)$:

- 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
- $FV(t_1 = t_2) = FV(t_1) \cup FV(t_2)$
- $FV(\neg\varphi) = FV(\varphi)$
- $FV(\psi \wedge \varphi) = FV(\psi \vee \varphi) = FV(\psi \to \varphi) = FV(\psi \leftrightarrow \varphi) = FV(\psi) \cup FV(\varphi)$
- $FV(\forall x \varphi) = FV(\exists x \varphi) = FV(\varphi) \setminus \{x\}$

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
- $FV(t_1 = t_2) = FV(t_1) \cup FV(t_2)$
- $FV(\neg\varphi) = FV(\varphi)$
- $FV(\psi \wedge \varphi) = FV(\psi \vee \varphi) = FV(\psi \to \varphi) = FV(\psi \leftrightarrow \varphi) = FV(\psi) \cup FV(\varphi)$
- $FV(\forall x \varphi) = FV(\exists x \varphi) = FV(\varphi) \setminus \{x\}$

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
- $FV(t_1 = t_2) = FV(t_1) \cup FV(t_2)$
- $FV(\neg\varphi) = FV(\varphi)$
- $FV(\psi \wedge \varphi) = FV(\psi \vee \varphi) = FV(\psi \rightarrow \varphi) = FV(\psi \leftrightarrow \varphi) = FV(\psi) \cup FV(\varphi)$
- $FV(\forall x \varphi) = FV(\exists x \varphi) = FV(\varphi) \setminus \{x\}$

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
- $FV(t_1 = t_2) = FV(t_1) \cup FV(t_2)$
- $FV(\neg\varphi) = FV(\varphi)$
- $FV(\psi \wedge \varphi) = FV(\psi \vee \varphi) = FV(\psi \to \varphi) = FV(\psi \leftrightarrow \varphi) = FV(\psi) \cup FV(\varphi)$
- $FV(\forall x\varphi) = FV(\exists x\varphi) = FV(\varphi) \setminus \{x\}$

Substitution

If t is a term, φ a formula, and $x \in FV(\varphi)$, then the **substitution of t for x in φ** (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 φ, 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, φ a formula, and $x \in FV(\varphi)$, then the **substitution of t for x in φ** (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 φ, 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 $D = \mathrm{Dom}(\mathcal{M})$
- 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.

Models

Predicate formulas are interpreted in **Models**.

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

- A (non-empty) domain $D = \mathrm{Dom}(\mathcal{M})$
- 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**), η, is a function from the set of variables to $\mathrm{Dom}(\mathcal{M})$.

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

Environments

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

Given an environment η, we denote by $\eta[x \mapsto c]$ the environment that agrees with η 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 η is an environment.

Interpretations

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

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

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

Interpretations

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

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

- $[\![P(t_1, \ldots, t_k)]\!]^{\eta}_{\mathcal{M}} = \texttt{true}$ if $P^{\mathcal{M}}([\![t_1]\!]^{\eta}_{\mathcal{M}}, \ldots, [\![t_k]\!]^{\eta}_{\mathcal{M}})$ holds.
- $[\![t_1 = t_2]\!]^{\eta}_{\mathcal{M}} = \texttt{true}$ if $[\![t_1]\!]^{\eta}_{\mathcal{M}} = [\![t_2]\!]^{\eta}_{\mathcal{M}}$
- $[\![\forall x \varphi]\!]^{\eta}_{\mathcal{M}} = \texttt{true}$ if $[\![\varphi]\!]^{\eta[x \mapsto c]}_{\mathcal{M}} = \texttt{true}$ for all $c \in \mathrm{Dom}(\mathcal{M})$
- $[\![\exists x \varphi]\!]^{\eta}_{\mathcal{M}} = \texttt{true}$ if $[\![\varphi]\!]^{\eta[x \mapsto c]}_{\mathcal{M}} = \texttt{true}$ for some $c \in \mathrm{Dom}(\mathcal{M})$
- $[\![\varphi]\!]^{\eta}_{\mathcal{M}}$ defined in the same way as Propositional Logic for all other formulas φ.

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

$$\forall x \forall y ((y = x + 1) \to (x \leq y))$$

- $\langle \mathbb{N}, \leq, +, 1 \rangle$: `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`