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
Motivating example

Consider the statement:

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

Can we encode this statement in propositional logic?
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:

\[
\begin{align*}
P_{11} &= \text{“1 = 1 + 1”} & S_{11} &= \text{“1 \leq 1”} \\
P_{12} &= \text{“2 = 1 + 1”} & S_{12} &= \text{“1 \leq 2”} \\
&\vdots & & \vdots \\
\end{align*}
\]

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 just 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:

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

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

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

Final result: $(P_{00} \rightarrow S_{00}) \land (P_{01} \rightarrow S_{01}) \land \cdots$
Motivating example

Consider the statement:

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

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

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

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

$\vdots$ $\vdots$ $\vdots$ $\vdots$

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

Consider the statement:

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

Predicate logic introduces:

- Predicates
Motivating example

Consider the statement:

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

Predicate logic introduces:

- Predicates
- Functions
Motivating example

Consider the statement:

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

Predicate logic introduces:

- Predicates
- Functions
- Constants
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
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
Q: Is this a true statement?

\[ \forall x y. x = y \]
Q: Is this a true statement?

\[ \forall x \, y. \ x = y \]

A: depends on what the domain of discourse is.
Domain of discourse

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

Example:

\[ \forall x \ C(x) \]

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)

Example: Consider:

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

where \( \text{C(x)} \) 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:

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

where \( \text{C}(x) \) 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

Multiple domains can be combined into one as follows.

For example: the predicate \texttt{studies}(x, y) representing “x (a student) studies y (a subject)”. 
Multiple domains of discourse

Multiple domains can be combined into one as follows.

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.
Multiple domains of discourse

Multiple domains can be combined into one as follows.

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: \( \forall x (\text{isStudent}(x) \rightarrow \varphi) \)
**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 \( y \)

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

Domain of discourse: People

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.
Example

Consider the following predicates and constants:

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

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

Domain of discourse: PEOPLE

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.
Example

Consider the following predicates and constants:

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

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

Domain of discourse: PEOPLE

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:

- $K(x, y)$: $x$ 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 no-one.

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
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, or database schemas.

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*
- 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, or database schemas.

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*
- 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 database schema identifies the various tables, their attributes, and their attributes’ types. For example:

<table>
<thead>
<tr>
<th>Person</th>
<th>Employee</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name: String</td>
<td>ID: int</td>
</tr>
<tr>
<td>Surname: String</td>
<td>Surname: String</td>
</tr>
<tr>
<td>Address: String</td>
<td></td>
</tr>
</tbody>
</table>

Tables relate a number of attributes (over several domains).
A database schema identifies the various tables, their attributes, and their attributes’ types. For example:

<table>
<thead>
<tr>
<th>Person</th>
<th>Employee</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name: String</td>
<td>ID: int</td>
</tr>
<tr>
<td>Surname: String</td>
<td>Surname: String</td>
</tr>
<tr>
<td>Address: String</td>
<td></td>
</tr>
</tbody>
</table>

Tables relate a number of attributes
The above schema would be represented by the vocabulary:

\[ DB = \{ \text{Person}, \text{Employee} \} \]

where Person is a ternary predicate symbol and Employee is a binary predicate symbol.
Vocabulary: example (databases)

Example

A database schema identifies the various tables, their attributes, and their attributes’ types. For example:

<table>
<thead>
<tr>
<th>Person</th>
<th>Employee</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name: String</td>
<td>ID: int</td>
</tr>
<tr>
<td>Surname: String</td>
<td>Surname: String</td>
</tr>
<tr>
<td>Address: String</td>
<td></td>
</tr>
</tbody>
</table>

Tables relate a number of attributes (over several domains). The above schema would be represented by the vocabulary:

\[ DB = \{ \text{Person, Employee, isString, isInteger} \} \]

where **Person** is a ternary predicate symbol and **Employee** is a binary predicate symbol and **isString** and **isInteger** are unary predicate symbols.
A term is defined inductively 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.
Terms: examples

Example

Over $V = \{\leq, +, 1\}$, the following are all terms:

- $x$
- $1$
- $(y, 1)$
- $(y, + (x, 1))$
Formulas

A formula of Predicate Logic is defined inductively 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:
  - $\neg \varphi$
  - $(\varphi \land \psi)$
  - $(\varphi \lor \psi)$
  - $(\varphi \rightarrow \psi)$
  - $(\varphi \leftrightarrow \psi)$
  - $\forall x \varphi$
  - $\exists x \varphi$

**NB**

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

\(\forall x \forall y ((y = x + 1) \rightarrow (x \leq y))\)
Over $V = \{\leq, +, 1\}$, the following are all formulas:

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

Feel free to write predicates and functions in infix for readability.
In relational databases, formulas correspond to (select-)queries.

**Example**

For the vocabulary

\[ DB = \{ \text{Person, Employee, isString, isInteger} \} : \]

\[
\text{Select } * \\
\text{from Person} \\
\text{Person(x, y, z)}
\]
In relational databases, formulas correspond to (select-)queries.

Example
For the vocabulary $DB = \{\text{Person, Employee, isString, isInteger, Alice}\}$:

Select *
from Person
where Person.name = "Alice"

$\text{Person}(x, y, z) \land (x = \text{Alice})$
In relational databases, formulas correspond to (select-)queries.

Example

For the vocabulary $DB = \{\text{Person, Employee, isString, isInteger, Alice}\}$:

\[
\text{Select } * \\
\text{from Person inner join Employee} \\
\text{on Person.surname = Employee.surname}
\]

$\text{Person}(x, y, z) \lor \text{Employee}(w, y)$
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**.
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 by \(\exists z\)
- y is free
- First x is bound by \(\exists x\)
- 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 by \(\exists z\)
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 by $\exists z$
- $y$ is free
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 by \(\exists z\)
- \(y\) is free
- First \(x\) is bound by \(\exists x\)
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 by \(\exists z\)
- \(y\) is free
- First \(x\) is bound by \(\exists x\)
- Second \(x\) is free
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 by \(\exists z\)
- \(y\) is free
- First \(x\) is bound by \(\exists x\)
- Second \(x\) is free

A formula with no free variables is a **sentence**.
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 $f$
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 \( P \)
- \( FV(t_1 = t_2) = FV(t_1) \cup FV(t_2) \)
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 $P$
- $FV(t_1 = t_2) = FV(t_1) \cup FV(t_2)$
- $FV(\neg \varphi) = FV(\varphi)$
- $FV(\psi \land \varphi) = FV(\psi \lor \varphi) = FV(\psi \rightarrow \varphi) = FV(\psi \leftrightarrow \varphi) = FV(\psi) \cup FV(\varphi)$
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 $P$
- $FV(t_1 = t_2) = FV(t_1) \cup FV(t_2)$
- $FV(\neg \varphi) = FV(\varphi)$
- $FV(\psi \land \varphi) = FV(\psi \lor \varphi) = FV(\psi \rightarrow \varphi) = FV(\psi \leftrightarrow \varphi) = FV(\psi) \cup FV(\varphi)$
- $FV(\forall x \varphi) = FV(\exists x \varphi) = FV(\varphi) \setminus \{x\}$
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
Q: Is this a true statement?

There is nothing going on between us.
Q: Is this a true statement?

There is nothing going on between us.

A: It depends upon what the meaning of the word ’is’ is.
Q: Is this a true statement?

1 + 1 = 2
Models

Q: Is this a true statement?

1 + 1 = 2

A: It depends on what the model of 1, 2 and + is.
Models

\{\forall, \exists, =, \land, \lor, \neg, \rightarrow, \leftrightarrow\} have a fixed meaning in first-order logic.

All other symbols are meaningless, unless we specify a model.
Models

Predicate formulas are interpreted in Models.

Given a vocabulary $V$ a model $M$ defines:

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

Predicate formulas are interpreted in Models.

Given a vocabulary $V$ a model $M$ defines:

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

Example

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

1. $\mathbb{N}$ with the standard definitions of $\leq$, $+$, and $1$.
2. $\{0, 1, 2, 3, 4\}$ with the standard definition of $\leq$ and $1$, and $m + n$ defined as $m + n \pmod{5}$.
3. The directed graph $G = (V, E)$ shown below with $\leq = E$; and $v + w$ defined to be $w$.

![Directed Graph Diagram]

- $1$ is connected to $2$, $3$, and $4$.
- $2$ is connected to $1$ and $3$.
- $3$ is connected to $2$ and $4$.
- $4$ is connected to $1$, $2$, and $3$.
Example

For the vocabulary $DB = \{\text{Person}, \text{Employee}, \text{isString}, \text{isInteger}\}$, the following database is a model:

<table>
<thead>
<tr>
<th>Name</th>
<th>Surname</th>
<th>Address</th>
<th>Person</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rapunzel</td>
<td>-</td>
<td>Tower</td>
<td></td>
</tr>
<tr>
<td>Cinderella</td>
<td>-</td>
<td>c/o Stepmum</td>
<td></td>
</tr>
<tr>
<td>Snow</td>
<td>White</td>
<td>Cottage</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>ID</th>
<th>Surname</th>
</tr>
</thead>
<tbody>
<tr>
<td>31415</td>
<td>Psmith</td>
</tr>
<tr>
<td>27182</td>
<td>Ukridge</td>
</tr>
<tr>
<td>16180</td>
<td>Wooster</td>
</tr>
</tbody>
</table>

isString and isInteger are defined by what values are permitted in each of the columns (sanitizing the input).
Environments

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

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

\[
\begin{align*}
\llbracket x \rrbracket_M^\eta &= \eta(x) \\
\llbracket c \rrbracket_M^\eta &= c^M \\
\llbracket f(t_1, \ldots, t_k) \rrbracket_M^\eta &= f^M(\llbracket t_1 \rrbracket_M^\eta, \ldots, \llbracket t_k \rrbracket_M^\eta)
\end{align*}
\]
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 \([\varphi]^\eta_M[x \mapsto c] = \text{true}\) for all \(c \in \text{dom}(M)\)
- \([\exists x \varphi]^\eta_M = \text{true}\) if \([\varphi]^\eta_M[x \mapsto c] = \text{true}\) for some \(c \in \text{dom}(M)\)
- \([\varphi]^\eta_M\) defined in the same way as Propositional Logic for all other formulas \(\varphi\).
Interpretations: examples

Example

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

- \( \mathbb{N} \) with the standard definitions of \( \leq \), +, and 1:
Interpretations: examples

Example

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

- \( \mathbb{N} \) with the standard definitions of \( \leq, +, \) and 1: true
Interpretations: examples

Example

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

- \( \mathbb{N} \) with the standard definitions of \( \leq \), +, and 1: true
- \( \{0, 1, 2, 3, 4\} \) with the standard definition of \( \leq \) and 1, and \( m + n \) defined as \( m + n \ (\text{mod } 5) \):
Example

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

- \( \mathbb{N} \) with the standard definitions of \( \leq, + \), and \( 1 \): true
- \( \{0, 1, 2, 3, 4\} \) with the standard definition of \( \leq \) and \( 1 \), and \( m + n \) defined as \( m + n \) (mod 5): false
Interpretations: examples

Example

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

- $\mathbb{N}$ with the standard definitions of $\leq$, $+$, and 1: true
- $\{0, 1, 2, 3, 4\}$ with the standard definition of $\leq$ and 1, and $m + n$ defined as $m + n \pmod{5}$: false
- The directed graph $G = (V, E)$ shown below with $\leq = E$, 1 be the vertex 1, and $v + w$ defined to be $w$.  

```
1
/|
/ |\            2
|   |
|   |
|   v
3 ---- 4
```
Interpretations: examples

Example

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

- $\mathbb{N}$ with the standard definitions of $\leq$, $+$, and $1$: true
- $\{0, 1, 2, 3, 4\}$ with the standard definition of $\leq$ and $1$, and $m + n$ defined as $m + n \pmod{5}$: false
- The directed graph $G = (V, E)$ shown below with $\leq = E$, 1 be the vertex 1, and $v + w$ defined to be $w$. true
Why separate the environment from the model?

In the definition of $\llbracket \varphi \rrbracket^\eta_M$, $\eta$ is only used to define values for the free variables. In particular, if $\varphi$ is a sentence then $\llbracket \varphi \rrbracket^\eta_M$ is independent of $\eta$. 

Define $\llbracket \cdot \rrbracket^M$ by "delaying" the assigning of values to free variables, and propagating them out. That is, define:

$$\llbracket \varphi(x_1, x_2, \ldots, x_n) \rrbracket^M = \llbracket \varphi \rrbracket^M(x_1, x_2, \ldots, x_n)$$

where $\llbracket \varphi \rrbracket^M : \text{dom}(M)^n \to \mathcal{B}$; that is, $\llbracket \varphi \rrbracket^M$ is an $n$-ary relation on $\text{dom}(M)$. 

78
Why separate the environment from the model?

In the definition of $[[\varphi]]^n_M$, $\eta$ is only used to define values for the free variables. In particular, if $\varphi$ is a sentence then $[[\varphi]]^n_M$ is independent of $\eta$.

Define $[[\cdot]]_M$ by “delaying” the assigning of values to free variables, and propagating them out.
Why separate the environment from the model?

In the definition of $[\varphi]_M^n$, $\eta$ is only used to define values for the free variables. In particular, if $\varphi$ is a sentence then $[\varphi]_M^n$ is independent of $\eta$.

Define $[\cdot]_M$ by “delaying” the assigning of values to free variables, and propagating them out. That is, define:

$$[\varphi(x_1, x_2, \ldots, x_n)]_M = [\varphi]_M(x_1, x_2, \ldots, x_n)$$

where $[\varphi]_M$:
Why separate the environment from the model?

In the definition of $[[\varphi]]^\eta_M$, $\eta$ is only used to define values for the free variables. In particular, if $\varphi$ is a sentence then $[[\varphi]]^\eta_M$ is independent of $\eta$.

Define $[[\cdot]]_M$ by “delaying” the assigning of values to free variables, and propagating them out. That is, define:

$$[[\varphi(x_1, x_2, \ldots, x_n)]]_M = [[\varphi]]_M(x_1, x_2, \ldots, x_n)$$

where $[[\varphi]]_M : \text{dom}(M)^n \rightarrow \mathbb{B}$;
Why separate the environment from the model?

In the definition of $\llbracket \varphi \rrbracket^\eta_M$, $\eta$ is only used to define values for the free variables. In particular, if $\varphi$ is a sentence then $\llbracket \varphi \rrbracket^\eta_M$ is independent of $\eta$.

Define $\llbracket \cdot \rrbracket_M$ by “delaying” the assigning of values to free variables, and propagating them out. That is, define:

$$\llbracket \varphi(x_1, x_2, \ldots, x_n) \rrbracket_M = \llbracket \varphi \rrbracket_M(x_1, x_2, \ldots, x_n)$$

where $\llbracket \varphi \rrbracket_M : \text{dom}(M)^n \to \mathbb{B}$; that is, $\llbracket \varphi \rrbracket_M$ is an $n$-ary relation on $\text{dom}(M)$. 
Interpretations: example (databases)

**Example**

- **Vocabulary:** database schema
- **Formulas:** queries ($\varphi$)
- **Models:** databases ($D$)
- **Interpretation:**

\[
\begin{align*}
\text{Environment:} & \text{ looks up an entry in a (derived) table and returns whether the lookup was successful.}
\end{align*}
\]

\[
\begin{align*}
\text{\textit{η}}(D) & \text{ Success/fail outcome of looking up a specific entry in a query result on } D.
\end{align*}
\]
Interpretations: example (databases)

Example

- **Vocabulary:** database schema
- **Formulas:** queries ($\varphi$)
- **Models:** databases ($D$)
- **Interpretation:** $\llbracket \varphi \rrbracket_D$ is a relation on $\text{dom}(D)$, i.e., a (derived) table in $D$
Interpretations: example (databases)

Example

- Vocabulary: database schema
- Formulas: queries ($\varphi$)
- Models: databases ($D$)
- Interpretation: $[\varphi]_D$ is a relation on $\text{dom}(D)$, i.e. a (derived) table in $D$
- Environment:
Interpretations: example (databases)

Example

- Vocabulary: database schema
- Formulas: queries ($\varphi$)
- Models: databases ($D$)
- Interpretation: $\models_{D}$ is a relation on $\text{dom}(D)$, i.e. a (derived) table in $D$
- Environment: looks up an entry in a (derived) table and returns whether the lookup was successful
Interpretations: example (databases)

Example

- Vocabulary: database schema
- Formulas: queries ($\varphi$)
- Models: databases ($D$)
- Interpretation: $[\varphi]_D$ is a relation on $\text{dom}(D)$, i.e. a (derived) table in $D$
- Environment: looks up an entry in a (derived) table and returns whether the lookup was successful
- $[\varphi]_D^\eta$: Success/fail outcome of looking up a specific entry in a query result on $D$. 
Satisfiability, truth, validity

A formula $\varphi$ of predicate logic is:

- **satisfiable** if there is some model $M$ and some environment $\eta$ such that $\llbracket \varphi \rrbracket_{M}^{\eta} = \text{true}$.
- **true in a model** $M$ if for all environments $\eta$ we have $\llbracket \varphi \rrbracket_{M}^{\eta} = \text{true}$
- a **logical validity** if it is true in all models.

**NB**

*For sentences the first two definitions coincide.*
Satisfiability, truth, validity

A formula $\varphi$ of predicate logic is:
- **satisfiable** if there is some model $M$ and some environment $\eta$ such that $\left[\varphi\right]_{M}^{\eta} = \text{true}$.
- **true in a model** $M$ if for all environments $\eta$ we have $\left[\varphi\right]_{M}^{\eta} = \text{true}$
- a **logical validity** if it is true in all models.

**NB**

*For sentences the first two definitions coincide.*

**Example**

The sentence $\forall x \forall y ((y = x + 1) \rightarrow (x \leq y))$ is satisfiable but it is not a logical validity.
Entailment, Logical equivalence

- A theory $T$ entails a formula $\varphi$, $T \models \varphi$, if $\varphi$ is satisfied by any interpretation that satisfies all formulas in $T$.

- $\varphi$ is logically equivalent to $\psi$, $\varphi \equiv \psi$, if $[\varphi]_M^n = [\psi]_M^n$ for all interpretations $(M, \eta)$.
A theory $T$ **entails** a formula $\varphi$, $T \models \varphi$, if $\varphi$ is satisfied by any interpretation that satisfies all formulas in $T$.

$\varphi$ is **logically equivalent** to $\psi$, $\varphi \equiv \psi$, if $[\varphi]_M = [\psi]_M$ for all interpretations $(M, \eta)$.

**Theorem**

- $\varphi_1, \ldots, \varphi_n \models \psi$ if, and only if, $(\varphi_1 \land \cdots \land \varphi_n) \rightarrow \psi$ is a logical validity.
- $\varphi \equiv \psi$ if, and only if, $\varphi \leftrightarrow \psi$ is a logical validity.
Summary of topics

- Re-introduction to Predicate Logic
- Syntax of Predicate Logic
- Semantics of Predicate Logic
- Natural Deduction for Predicate Logic (not today)