We can already do a lot with propositional logic. It is, however, annoying that there is no structure in the atomic propositions.

Example:

“All blocks are red”
“There is a block A”

It should follow that “A is red”

But propositional logic cannot handle this.

Idea: We introduce individual variables, predicates, functions, . . . .

→ First-Order Predicate Logic (PL1)
Lecture Overview

1 Syntax and Semantics

2 Reduction to Propositional Theories

3 Summary
Symbols:

- Operators: \( \neg, \lor, \land, \forall, \exists, = \)
- Variables: \( x, x_1, x_2, \ldots, x', x'', \ldots, y, \ldots, z, \ldots \)
- Brackets: \( (, [) \), (, [) \)
- Function symbols (e.g., \( \text{weight}(), \text{color}() \))
- Predicate symbols (e.g., \( \text{Block}(), \text{Red}() \))

Predicate and function symbols have an arity (number of arguments).
- 0-ary predicate \( \equiv \) propositional logic atoms: \( P, Q, R, \ldots \)
- 0-ary function \( \equiv \) constants: \( a, b, c, \ldots \)

We assume a countable set of predicates and functions of any arity.

“\( = \)” is usually not considered a predicate, but a logical symbol.
Terms (represent objects):
1. Every variable is a term.
2. If \( t_1, t_2, \ldots, t_n \) are terms and \( f \) is an \( n \)-ary function, then
\[
f(t_1, t_2, \ldots, t_n)
\]
is also a term.
Terms without variables: ground terms.

Atomic Formulae (represent statements about objects)
1. If \( t_1, t_2, \ldots, t_n \) are terms and \( P \) is an \( n \)-ary predicate, then
\[
P(t_1, t_2, \ldots, t_n)
\]
is an atomic formula.
2. If \( t_1 \) and \( t_2 \) are terms, then \( t_1 = t_2 \) is an atomic formula.
Atomic formulae without variables: ground atoms.
The Grammar of First-Order Predicate Logic (2)

Formulae:
1. Every atomic formula is a formula.
2. If \( \varphi \) and \( \psi \) are formulae and \( x \) is a variable, then
   \[
   \neg \varphi, \ \varphi \land \psi, \ \varphi \lor \psi, \ \varphi \Rightarrow \psi, \ \varphi \Leftrightarrow \psi, \ \exists x \varphi \text{ and } \forall x \varphi
   \]
   are also formulae.
   \( \forall, \ \exists \) are as strongly binding as \( \neg \).

Propositional logic is part of the PL1 language:
1. Atomic formulae: only 0-ary predicates
2. Neither variables nor quantifiers.
### Alternative Notation

<table>
<thead>
<tr>
<th>Here</th>
<th>Elsewhere</th>
</tr>
</thead>
<tbody>
<tr>
<td>\neg \varphi</td>
<td>\sim \varphi \ \bar{\varphi}</td>
</tr>
<tr>
<td>\varphi \land \psi</td>
<td>\varphi &amp; \psi \ \varphi \bullet \psi \ \varphi, \psi</td>
</tr>
<tr>
<td>\varphi \lor \psi</td>
<td>\varphi</td>
</tr>
<tr>
<td>\varphi \Rightarrow \psi</td>
<td>\varphi \rightarrow \psi \ \varphi \supset \psi</td>
</tr>
<tr>
<td>\varphi \Leftrightarrow \psi</td>
<td>\varphi \leftrightarrow \psi \ \varphi \equiv \psi</td>
</tr>
<tr>
<td>\forall x \varphi</td>
<td>(\forall x) \varphi \land x \varphi</td>
</tr>
<tr>
<td>\exists x \varphi</td>
<td>(\exists x) \varphi \lor x \varphi</td>
</tr>
</tbody>
</table>
Meaning of PL1-Formulae

Our example: $\forall x [Block(x) \Rightarrow Red(x)], \ Block(a)$

For all objects $x$: If $x$ is a block, then $x$ is red and $a$ is a block.

Generally:

- Terms are interpreted as objects.
- Universally-quantified variables denote all objects in the universe.
- Existentially-quantified variables represent one of the objects in the universe (made true by the quantified expression).
- Predicates represent subsets of the universe.

Similar to propositional logic, we define interpretations, satisfiability, models, validity, ...
Semantics of PL1-Logic

**Interpretation:** \( I = \langle D, \bullet^I \rangle \) where \( D \) is an arbitrary, non-empty set and \( \bullet^I \) is a function that

- maps \( n \)-ary function symbols to functions over \( D \):
  \[ f^I \in [D^n \rightarrow D] \]
- maps individual constants to elements of \( D \):
  \[ a^I \in D \]
- maps \( n \)-ary predicate symbols to relations over \( D \):
  \[ P^I \subseteq D^n \]

**Interpretation of ground terms:**

\[ (f(t_1, \ldots, t_n))^I = f^I(t_1^I, \ldots, t_n^I) \]

**Satisfaction of ground atoms** \( P(t_1, \ldots, t_n) \):

\[ I \models P(t_1, \ldots, t_n) \text{ iff } \langle t_1^I, \ldots, t_n^I \rangle \in P^I \]
\[ D = \{ d_1, \ldots, d_n \mid n > 1 \} \]
\[ a^I = d_1 \]
\[ b^I = d_2 \]
\[ c^I = \ldots \]
\[ \text{Block}^I = \{ d_1 \} \]
\[ \text{Red}^I = D \]
\[ I \models \text{Red}(b) \]
\[ I \not\models \text{Block}(b) \]
Example (2)

\[ D = \{1, 2, 3, \ldots\} \]
\[ 1^I = 1 \]
\[ 2^I = 2 \]
\[ \ldots \]
\[ Even^I = \{2, 4, 6, \ldots\} \]
\[ succ^I = \{(1 \mapsto 2), (2 \mapsto 3), \ldots\} \]
\[ I \models Even(2) \]
\[ I \not\models Even(succ(2)) \]
Semantics of PL1: Variable Assignment

Set of all variables $V$. Function $\alpha : V \to D$

Notation: $\alpha[x/d]$ is the same as $\alpha$ apart from point $x$.

For $x : \alpha[x/d](x) = d$.

Interpretation of terms under $I, \alpha$: 

$$x^{I,\alpha} = \alpha(x)$$

$$a^{I,\alpha} = a^I$$

$$(f(t_1, \ldots, t_n))^{I,\alpha} = f^I(t_1^{I,\alpha}, \ldots, t_n^{I,\alpha})$$

Satisfaction of atomic formulae:

$$I, \alpha \models P(t_1, \ldots, t_n) \text{ iff } \langle t_1^{I,\alpha}, \ldots, t_n^{I,\alpha} \rangle \in P^I$$
$Block^I = \{ d_1 \}$

$Red^I = D$

$\alpha = \{(x \mapsto d_1), (y \mapsto d_2)\}$

$I, \alpha \models Red(x)$

$I, \alpha[y/d_1] \models Block(y)$
A formula $\varphi$ is satisfied by an interpretation $I$ and a variable assignment $\alpha$, i.e., $I, \alpha \models \varphi$:

\[
\begin{align*}
I, \alpha &\models \top \\
I, \alpha &\not\models \bot \\
I, \alpha &\models \neg \varphi \iff I, \alpha \not\models \varphi \\
&\ldots
\end{align*}
\]

and all other propositional rules as well as

\[
\begin{align*}
I, \alpha &\models P(t_1, \ldots, t_n) \iff \langle t_1^I, \alpha, \ldots, t_n^I, \alpha \rangle \in P^I \\
I, \alpha &\models \forall x \varphi \iff \text{for all } d \in D, I, \alpha[x/d] \models \varphi \\
I, \alpha &\models \exists x \varphi \iff \text{there exists a } d \in D \text{ with } I, \alpha[x/d] \models \varphi
\end{align*}
\]
Example

\[ D = \{d_1, \ldots, d_n \mid n > 1\} \]

\[ a^I = d_1 \]

\[ b^I = d_2 \]

\[ Block^I = \{d_1\} \]

\[ Red^I = D \]

\[ \alpha = \{(x \mapsto d_1), (y \mapsto d_2)\} \]

Questions:

1. \( I, \alpha \models Block(b) \lor \neg Block(b) \)?

2. \( I, \alpha \models Block(x) \Rightarrow (Block(x) \lor \neg Block(y)) \)?

3. \( I, \alpha \models Block(a) \land Block(b) \)?

4. \( I, \alpha \models \forall x (Block(x) \Rightarrow Red(x)) \)?
∀x [R(y, z) ∧ ∃y (¬P(y, x) ∨ R(y, z))]

The boxed appearances of y and z are free. All other appearances of x, y, z are bound.

Formulae with no free variables are called closed formulae or sentences. We form theories from closed formulae.

Note: With closed formulae, the concepts logical equivalence, satisfiability, and implication, etc. are not dependent on the variable assignment α (i.e., we can always ignore all variable assignments).

With closed formulae, α can be left out on the left side of the model relationship symbol:

\[ I \models \varphi \]
An interpretation $I$ is called a model of $\varphi$ under $\alpha$ if

$$I, \alpha \models \varphi$$

A PL1 formula $\varphi$ can, as in propositional logic, be satisfiable, unsatisfiable, falsifiable, or valid.

Analogously, two formulae are logically equivalent ($\varphi \equiv \psi$) if for all $I, \alpha$:

$$I, \alpha \models \varphi \iff I, \alpha \models \psi$$

Note: $P(x) \not\equiv P(y)$!

Logical Implication is also analogous to propositional logic.

Question: How can we define derivation?
Lecture Overview

1. Syntax and Semantics

2. Reduction to Propositional Theories

3. Summary
Derivation in PL1: Possible Approaches

- We now know the semantics of PL1. How can we do inference in PL1?
- One way: Normalization + Skolemization + Resolution with Unification
- Alternative: Reduction to propositional logic by instantiation based on the so-called Herbrand Universe (all possible terms) $\mapsto$ infinite propositional theories
- It turns out that logical implication in PL1 is undecidable!
- Simple way for special case: If the number of objects is finite, instantiate all variables by possible objects (in fact, often used in AI systems, e.g. planning or ASP)
Let us assume that we only want to talk about a finite number of objects.

**Domain closure axiom (DCA):**
\[ \forall x [x = c_1 \lor x = c_2 \lor \ldots \lor x = c_n] \]

Often one also assumes that different names denote different objects (unique name assumption/axiom or UNA):
\[ \land_{i \neq j} [c_i \neq c_j] \]

→ Only important when counting or using \( \neq \) or \( = \) as a predicate.

Eliminate quantification by instantiating all variables with all possible values.
**Notation:** if \( \varphi \) is a formula, then \( \varphi[x/a] \) is the formula with all free
occurrences of \( x \) replaced by \( a \).

**Universally quantified formulas** are replaced by a conjunction of formulas
with the variable instantiated to all possible values (from DCA):
\[
\forall x \varphi \rightsquigarrow \bigwedge_i \varphi[x/c_i]
\]

**Existentially quantified variables** are replaced by a disjunction of
formulas with the variable instantiated to all possible values (from
DCA):
\[
\exists x \varphi \rightsquigarrow \bigvee_i \varphi[x/c_i]
\]

**Note:** does blow up the formulas exponentially in the **arity** of the
predicates!
\( \forall x \ (\text{Block}(x) \Rightarrow \text{Red}(x)) \)
\( \forall x \ (x = a \lor x = b \lor x = c) \)
\( \Rightarrow \)
\( (\text{Block}(a) \Rightarrow \text{Red}(a)) \land (\text{Block}(b) \Rightarrow \text{Red}(b)) \land (\text{Block}(c) \Rightarrow \text{Red}(c)) \)
Lecture Overview

1. Syntax and Semantics
2. Reduction to Propositional Theories
3. Summary
Summary

- PL1 makes it possible to structure statements, thereby giving us considerably more expressive power than propositional logic.
- Logical implication in PL1 is undecidable.
- If we only reason over a finite universe, PL1 can be reduced to propositional logic over finite theories (but the reduction is exponential in the arity of the predicates).