Lecture 16

Predicate Logic
Predicates, Variables, and Quantifiers

Properties are expressed by predicates. \( S, I, Y \) are \textit{predicates}.

\begin{align*}
  S(\text{andy}): & \text{ Andy is a student.} \\
  I(\text{paul}): & \text{ Paul is an instructor.} \\
  Y(\text{andy}, \text{paul}): & \text{ Andy is younger than Paul.}
\end{align*}

Variables are placeholders for concrete values.

\begin{align*}
  S(x): & \text{ } x \text{ is a student.} \\
  I(x): & \text{ } x \text{ is an instructor.} \\
  Y(x, y): & \text{ } x \text{ is younger than } y.
\end{align*}

Quantifiers make possible encoding the phrase:

\textit{“Every student is younger than some instructor.”}

Two quantifiers: \( \forall \) – \textit{forall}, and \( \exists \) – \textit{exists}.

Encoding of the above sentence:

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

Consider translating the sentence:

“Every son of my father is my brother”

Two alternatives:

- “Father of” relationship encoded as a predicate.

  \[ S(x, y) : x \text{ is the son of } y. \]
  \[ F(x, y) : x \text{ is the father of } y. \]
  \[ B(x, y) : x \text{ is the brother of } y. \]
  \[ m : \text{ constant, denoting “myself”}. \]

  **Translation:** \( \forall x \forall y (F(x, m) \land S(y, x) \rightarrow B(y, m)) \)

- “Father of” relationship encoded as a function.

  \[ f(x) : \text{ father of } x. \]

  **Translation:** \( \forall x (S(x, f(m)) \rightarrow B(x, m)) \)
Function

Every child is younger than its mother.

\( C(x): x \) is a child
\( M(x, y): x \) is mother of \( y \)
\( Y(x, y): x \) is younger than \( y \)

\( \forall x \forall y (C(x) \land M(y, x) \rightarrow Y(x, y)) \)

Andy and Paul have the same maternal grandmother.

\( \forall x \forall y \forall u \forall v (M(x, y) \land M(y, a) \land M(u, v) \land M(v, p) \rightarrow x = u) \).

\( m(m(a)) = m(m(p)) \)
Notes about Function

Ann likes Mary’s brother

\[ \exists x \ (B(x, m) \land L(a, x)) \]

\[ \forall x \ (B(x, m) \rightarrow L(a, x)) \]
Predicate Logic as a Formal Language

Two sorts of “things” in a predicate formula:

- Objects such as \( a \) (Andy) and \( p \) (Paul). Function symbols also refer to objects. These are modeled by terms.

- Expressions that can be given truth values. These are modeled by formulas.

A predicate vocabulary consists of 3 sets:

- **Predicate symbols** \( \mathcal{P} \);
- **Function symbols** \( \mathcal{F} \);
- **Constants** \( C \).

Each predicate and function symbol comes with a fixed arity (i.e., number of arguments).
Terms

**Definition:** *Terms* are defined as follows:

- Any variable is a term;
- Any constant in $C$ is a term;
- If $t_1, \ldots, t_n$ are terms and $f \in F$ has arity $n$, then $f(t_1, \ldots, t_n)$ is a term;
- Nothing else is a term.

**Backus-Naur definition:** $t ::= x | c | f(t, \ldots, t)$ where $x$ represents variables, $c$ represents constants in $C$, and $f$ represents function in $F$ with arity $n$.

**Remarks:**

- The first building blocks of terms are constants and variables.
- More complex terms are built from function symbols using previously built terms.
Formulas

**Definition:** We define the set of formulas over \((\mathcal{F}, \mathcal{P})\) inductively, using the already defined set of terms over \(\mathcal{F}\).

- If \(P\) is a predicate with \(n \geq 1\) arguments, and \(t_1, \ldots, t_n\) are terms over \(\mathcal{F}\), then \(P(t_1, \ldots, t_n)\) is a formula.
- If \(\Phi\) is a formula, then so is \(\neg \Phi\).
- If \(\Phi\) and \(\Psi\) are formulas, then so are \(\Phi \land \Psi, \Phi \lor \Psi, \Phi \rightarrow \Psi\).
- If \(\Phi\) is a formula and \(x\) is a variable, then \(\forall x \Phi\) and \(\exists x \Phi\) are formulas.
- Nothing else is a formula.

**BNF definition:**

\[
\Phi ::= P(t_1, \ldots, t_n) | (\neg \Phi) | (\Phi \land \Phi) | (\Phi \lor \Phi) | (\Phi \rightarrow \Phi) | (\forall x \Phi) | (\exists x \Phi)
\]

where \(P\) is a predicate of arity \(n\), \(t_i\) are terms, \(i \in \{1, \ldots, n\}\), \(x\) is a variable.

**Convention:** We retain the usual binding priorities of the connectives \(\neg, \land, \lor, \rightarrow\). We add that \(\forall x\) and \(\exists x\) bind like \(\neg\).
Introduction to Prolog


3. Other resources - Cygwin for windows: [https://cygwin.com/install.html](https://cygwin.com/install.html)

Support Linux, Windows & Mac

*Materials adapted from Geraint Wiggins*
Prolog is a logic language that is particularly suited to programs that involve symbolic or non-numeric computation. For this reason it is a frequently used language in Artificial Intelligence where manipulation of symbols and inference about them is a common task.

Prolog consists of a series of **FACTS** and **RULES**. A program is run by presenting some query and seeing if this can be proved against these known rules and facts.

- If the train arrives late and there are no taxis at the station then John is late for his meeting. John is not late for his meeting. The train did arrive late.

Materials adapted from Geraint Wiggins