Mathematical Logic and Linguistics

Glyn Morrill & Oriol Valentín

Department of Computer Science
Universitat Politècnica de Catalunya
morrill@cs.upc.edu & oriol.valentin@gmail.com

BGSMath Course
Class 6

Higher-order logic as a simply typed lambda-calculus with logical constants

Higher-order logic as a simply typed lambda-calculus with logical constants

We can express indefinitely high-order logic by adding logical constants to typed lambda calculus (Church 1940).

Higher-order logic as a simply typed lambda-calculus with logical constants

We can express indefinitely high-order logic by adding logical constants to typed lambda calculus (Church 1940).

Let us assume basic types e for entities (D_e a nonempty set) and t for truth values ($D_t = \{\{\emptyset\}, \emptyset\}$).

Higher-order logic as a simply typed lambda-calculus with logical constants

We can express indefinitely high-order logic by adding logical constants to typed lambda calculus (Church 1940).

Let us assume basic types e for entities (D_e a nonempty set) and t for truth values ($D_t = \{\{\emptyset\}, \emptyset\}$).

Logical constants are typed constants as usual, except that their denotations are constrained.

Higher-order logic as a simply typed lambda-calculus with logical constants

We can express indefinitely high-order logic by adding logical constants to typed lambda calculus (Church 1940).

Let us assume basic types e for entities (D_e a nonempty set) and t for truth values ($D_t = \{\{\emptyset\}, \emptyset\}$).

Logical constants are typed constants as usual, except that their denotations are constrained.

For example, we may assume a logical constant **And** for conjunction which is of type $t \to (t \to t)$. But as a constant which is logical, rather than considering valuations in which it is interpreted as any function this type, we include only those valuations in which it is interpreted specifically according to the truth table of conjunction.

Logical constants

constant	type	constraint
Not	$t \to t$	$f(\textbf{Not})(m) = \overline{m}^{\{\emptyset\}}$
And	$t \to (t \to t)$	$f(\textbf{And})(m)(m') = m \cap m'$
Or	$t \to (t \to t)$	$f(\textbf{Or})(m)(m') = m \cup m'$
Imp	$t \to (t \to t)$	$f(\textbf{Imp})(m)(m') = \overline{m}^{\{\emptyset\}} \cup m'$
Eq	$t \to (t \to t)$	$f(\textbf{Eq})(m)(m') = \{\emptyset\}$ if $m = m'$ else $\emptyset$
All	$(e \to t) \to t$	$f(\textbf{All})(m) = \bigcap_{m' \in D_e} m(m')$
Exst	$(e \to t) \to t$	$f(\textbf{Exst})(m) = \bigcup_{m' \in D_e} m(m')$
Iota	$(e \to t) \to e$	$f(\textbf{Iota})(\{m\}) = m$

Translation from first-order logic notation into higher-order logic

$$
\begin{array}{rcll}
|x| & = & x & \text{for individual variable } x \\
|a| & = & a & \text{for individual constant } a \\
|f(t_0, \ldots, t_n)| & = & (\cdots(|f|\,|t_0|)\ \cdots\ |t_n|) & \\
|Pt_1 \ldots t_n| & = & (\cdots(|P|\,|t_1|)\ \cdots\ |t_n|) & \\
|\neg A| & = & (\textbf{Not}\ |A|) & \\
|A \wedge B| & = & ((\textbf{And}\ |A|)\ |B|) & \\
|A \vee B| & = & ((\textbf{Or}\ |A|)\ |B|) & \\
|A \rightarrow B| & = & ((\textbf{Imp}\ |A|)\ |B|) & \\
|\forall x A| & = & (\textbf{All}\ \lambda x |A|) & \\
|\exists x A| & = & (\textbf{Exst}\ \lambda x |A|) &
\end{array}
$$