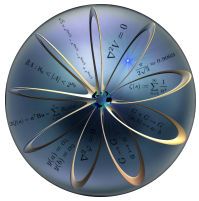

planetmath.org

Math for the people, by the people.

logical language

Canonical name	LogicalLanguage
Date of creation	2013-03-22 12:59:55
Last modified on	2013-03-22 12:59:55
Owner	Henry (455)
Last modified by	Henry (455)
Numerical id	14
Author	Henry (455)
Entry type	Definition
Classification	msc 03B15
Classification	msc 03B10
Related topic	QuantifierFree
Defines	term
Defines	formula
Defines	constant

In its most general form, a *logical language* is a set of rules for constructing *formulas* for some logic, which can then be assigned truth values based on the rules of that logic.

A logical language $\mathcal{L}$ consists of:

- A set F of function symbols (common examples include $+$ and $\cdot$)
- A set R of relation symbols (common examples include $=$ and $<$)
- A set C of logical connectives (usually $\neg$, $\wedge$, $\vee$, $\rightarrow$ and $\leftrightarrow$)
- A set Q of quantifiers (usuallly $\forall$ and $\exists$)
- A set V of variables

Every function symbol, relation symbol, and connective is associated with an arity (the set of n-ary function symbols is denoted F_n, and similarly for relation symbols and connectives). Each quantifier is a generalized quantifier associated with a quantifier type $\langle n_1, \ldots, n_n \rangle$.

The underlying logic has a (possibly empty) set of types T. There is a function Type : $F \cup V \rightarrow T$ which assignes a type to each function and variable. For each arity n is a function $\text{Inputs}_n : F_n \cup R_n \rightarrow T^n$ which gives the types of each of the arguments to a function symbol or relation. In addition, for each quantifier type $\langle n_1, \ldots, n_n \rangle$ there is a function $\text{Inputs}_{\langle n_1, \ldots, n_n \rangle}$ defined on $Q_{\langle n_1, \ldots, n_n \rangle}$ (the set of quantifiers of that type) which gives an n-tuple of n_i-tuples of types of the arguments taken by formulas the quantifier applies to.

The *terms* of $\mathcal{L}$ of type $t \in T$ are built as follows:

1. If v is a variable such that $\text{Type}(v) = t$ then v is a term of type t
2. If f is an n-ary function symbol such that $\text{Type}(f) = t$ and $t_1, \ldots, t_n$ are terms such that for each $i < n$ $\text{Type}(t_i) = (\text{Inputs}_n(f))_i$ then $ft_1, \ldots, t_n$ is a term of type t

The *formulas* of $\mathcal{L}$ are built as follows:

1. If r is an n-ary relation symbol and $t_1, \ldots, t_n$ are terms such that $\text{Type}(t_i) = (\text{Inputs}_n(r))_i$ then $rt_1, \ldots, t_n$ is a formula

2. If c is an n-ary connective and $f_1, \ldots, f_n$ are formulas then $cf_1, \ldots, f_n$ is a formula

3. If q is a quantifier of type $\langle n_1, \ldots, n_n \rangle$, $v_{1,1}, \ldots, v_{1,n_1}, v_{2,1}, \ldots, v_{n,1}, \ldots, v_{n,n_n}$ are a sequence of variables such that $\text{Type}(v_{i,j}) = ((\text{Inputs}_{\langle n_1,\ldots,n_n \rangle}(q))_j)_i$ and $f_1, \ldots, f_n$ are formulas then $qv_{1,1}, \ldots, v_{1,n_1}, v_{2,1}, \ldots, v_{n,1}, \ldots, v_{n,n_n} f_1, \ldots, f_n$ is a formula

Generally the connectives, quantifiers, and variables are specified by the appropriate logic, while the function and relation symbols are specified for particular languages. Note that 0-ary functions are usually called *constants*.

If there is only one type which is equated directly with truth values then this is essentially a propositional logic. If the standard quantifiers and connectives are used, there is only one type, and one of the relations is $=$ (with its usual semantics), this produces first order logic. If the standard quantifiers and connectives are used, there are two types, and the relations include $=$ and $\in$ with appropriate semantics, this is second order logic (a slightly different formulation replaces $\in$ with a 2-ary function which represents function application; this views second order objects as functions rather than sets).

Note that often connectives are written with infix notation with parentheses used to control order of operations.