

5.3 Operators

Freitag, 12. Juni 2015 08:30

Usually, Prolog uses Prefix-notation for function and predicate symbols:

$$p(X, f(a))$$

Such symbols are also called functors.

Sometimes, one wants to use binary symbols in infix-notation: $2 + 3$ instead of $+(2, 3)$

Similarly, one might want to use unary symbols in prefix- or postfix notation (without brackets).

$$-X \quad \text{instead of} \quad -(X)$$

$$X- \quad \text{---} \quad \text{---}$$

Some pre-defined symbols are already declared as operators (e.g., $+$). Then $+(2, 3)$ and $2+3$ are considered to be syntactically equal.

To define operators, one uses a directive of the following form:

↑
queries contained
in the program. When
loading the program, Prolog

tries to prove those queries

$\text{op}(\text{Precedence, Type, Name(s)})$

Directive: clause with empty head.

determines how strong the binding of an operator is

determines whether it is an infix, prefix or postfix operator and the association of the operator

names of the symbols that are declared as operators

Pre-defined op-directives for $+$, $-$, $*$:

$\text{op}(500, \text{YfX}, [+,-])$

$\text{op}(400, \text{YfX}, *)$

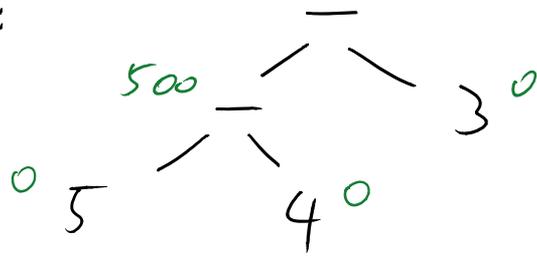
Precedence is needed to state how strong an operator binds its arguments. Smaller precedence means that the operator has a stronger binding.

e.g., $1 + 2 * 3$ stands for $1 + (2 * 3)$

because precedence of $*$ is smaller than precedence of $+$.

Type (e.g., YfX) determines the order of operator and arguments. Here, "f" stands for the operator and Y, x stand for arguments.

Reason:



$Y \# X$ means that the right argument must have smaller precedence (stronger binding)

$\Rightarrow Y \# X$ means association to the left

$X \# Y$ — " ————— right

$X \# X$ means: no association

$\therefore -op(500, X \# X, +++).$

Then $3 * 2 +++ 4 * 5$ stands for $(3 * 2) +++ (4 * 5)$

But $1 +++ 2 +++ 3$ is not allowed.

$(1 +++ 2) +++ 3$ is ok.

precedence 0

Types for prefix operators: $f x$, $f y$

Types for postfix operators: $x f$, $y f$

Pre-defined prefix operator for negation of numbers: $-$ is overloaded, i.e., there is a binary and a unary $-$.

$\therefore \text{-op}(200, \text{fy}, -)$.

Therefore $-2-3$ stands for $(-2)-3$

and $--2$ stands for $-(-2)$

$$\begin{array}{r} \text{---} 200 \\ | \\ \text{---} 200 \\ | \\ 20 \end{array}$$

Goal of operators: increase readability

Allows a simple form of natural language processing.

Verb: "was" should be used in infix notation

laura was beautiful instead of
 $\text{was}(\text{laura}, \text{beautiful})$

"was" should not associate to left or right:

laura was beautiful was young

Makes no sense

$\therefore \text{-op}(300, \text{xfx}, \text{was})$.

"of" should also be used in infix-notation, associates

to the right:

secretary of son of john stands for

secretary of (son of john)

"of" should bind stronger than "was":

laura was secretary of john stands for

laura was (secretary of john).

$\therefore \text{op}(250, \times, \text{of})$.

"the" unary prefix operator, type fx
(the the son makes no sense)

the secretary of the son stands for
(the secretary) of (the son)

\Rightarrow "the" binds stronger than "of"

$\therefore \text{op}(200, fx, \text{the})$.

laura was the secretary of the head of the department.

\uparrow Prolog fact which looks like natural language.

\Rightarrow goal: programming in (almost) natural language

This fact stands for
was (laura, of (the (secretary),
of (the (head), the (department))))

Prolog-Reg:

laura was the secretary of the head of the department.

? - Who was the secretary of the head of the department.

Who = laura.

? - laura was What.

What = the secretary of the head of the department.

? - Who was the secretary of the head of What.

Who = laura

What = the department