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:
\[ 2 + 3 \quad \text{instead of} \quad +\,(2,\,3) \]

Similarly, one might want to use unary symbols in prefix- or postfix notation (without brackets):
\[ -\,X \quad \text{instead of} \quad -(\,X\,) \]
\[ X- \quad \text{or} \quad X\,\_\_ \_\_ \]

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:

\[ \text{queries contained in the program. When loading the program, Prolog} \]
- op (Precedence, Type, Name(s)).

Pre-defined op-directives for +, -, *:
:- op (500, `,f`, `[+,-]`).
:- op (400, `,f`, `*`).

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., `,f`) determines the order of operator and arguments. Here, `f` stands for the operator and `,` stands for arguments.
Types for infix-operators:

\[ x \text{ if } x \]
\[ x \text{ if } y \]
\[ x \text{ if } x \]

\[ x = \text{argument with smaller precedence than } f \]
\[ y = \text{argument with smaller or equal precedence than } f \]

Precedence of an argument:

- If outermost symbol is an operator, then the precedence of the argument is the prec. of this operator:
  \[ 2 + 3 \text{ has precedence of } +, \text{i.e., } 500 \]
  \[ (6 \times 7) + 3 \text{ has precedence of } (, \text{i.e., } 1000 \]

- If argument is in brackets or a variable or built with a functor, then its precedence is 0.
  \[ (2 + 3) \text{ has precedence } 0 \]
  \[ 2 \]

Type is needed to determine whether the operator associates to the left or right:

\[-op \ (500, y \text{ if } x, [+,-]) \]

5 - 4 - 3 stands for (5 - 4) - 3

Reason:

\[ \begin{array}{c}
500 \\
500 \\
500 \\
\end{array} \]

\[ \text{y if x means } \]

\[ \begin{array}{c}
500 \\
500 \\
500 \\
\end{array} \]

\[ \text{...01} \]
Reason: \[ 500 \rightrightarrow 30 \rightrightarrow 0 \rightrightarrow 5 \rightrightarrow 40 \]

\( \Rightarrow \) \( \text{contains association to the left} \)

\( \times \times \text{ means association to the left} \)
\( \times \times \text{ right} \)
\( \times \times \text{ means: no association} \)

\[ \circ \quad \text{op(} 500, \times \times \text{, } \text{+++)}. \]

Then \( 3 \times 2 \text{++} 4 \times 5 \) \text{ stands for} \( (3 \times 2) \text{++} (4 \times 5) \)

But \( 1 \text{++} 2 \text{++} 3 \) \text{ is not allowed.} \( (1 \text{++} 2) \text{++} 3 \text{ is ok.} \)

prececedence 0

Types for prefix operators: \( \text{f} \times, \text{f} \times \)
Types for postfix operators: \( \times \text{f}, \text{f} \times \)

Pre-defined prefix operator for negation of numbers: \( \text{---} \text{ is overloaded, i.e., there is a binary and a unary} \text{---} \).
\[ :- \text{op}(200, f, -). \]

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

and \(- - 2\) stands for \(-(-2)\)

\[
\begin{array}{c}
\text{Goal of operators: increase readability} \\
\text{Allows a simple form of natural language processing.} \\
\text{Verb: "was" should be used in infix notation} \\
\text{Laura was beautiful instead of} \\
\text{was(Laura, beautiful)} \\
\text{"was" should not associate to left or right:} \\
\text{Laura was beautiful was young} \\
\text{makes no sense} \\
\end{array}
\]

\[ :- \text{op}(300, xf x, \text{was}). \]

"of" should also be used in infix-notation, associates
to thought:
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).
:- op(250, x+y, 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)
⇒ "the" binds stronger than 'of'
:- op(200, f x, the).

laura was the secretary of the head of the department.

^ Prolog fact which looks
like natural language.
⇒ Goal: programming in (almost)
natural language
This fact stands for
\text{was} (\text{laura}) \text{ of } (\text{the} (\text{secretary})
\text{ of } (\text{the} (\text{head}) \text{, the} (\text{department}))))

Robog-Reg: 
\text{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