5.3 Operators

Dienstag, 20. Juni 2017 09:30

Standard notation for predicates and functions is prefix notation:

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

But sometimes one wants to have infix (or postfix) notation: $2 + 3$ instead of $+(2, 3)$

they are considered syntactically identical

This is possible because + has been declared as an infix operator.

```
?- 2+3 == +(2,3).
true
```

To declare operators, one has to add a <u>directive</u> of the following form to the program:

```
:- op(Precedence, Type, Name(s)).
```

Directives are queries written in the program.

They are proved when loading the program. This is interesting if the predicates in the directive perform side-effects. Here, the side effect is that certain symbols can be used in infix-notation etc. after having evaluated this predicate.

For +, -, *, there are the following pre-defined directives:

```
:- op(500, yfx, [+, -]).
:- op(400, yfx, *).
```

Precedence (number between 0 and 1200). States which operation has stronger binding than another

needed to declare association to the left

Name of the symbol that is declared as an operator

A small precedence means strong binding

1 + 2 * 3 should stand for 1 + (2 * 3)

5 - 4 - 3 should stand for 5 - (4 - 3) or (5 - 4) - 3.

Should it associate to the right or to the left?

Types:

xfx , yfx , xfy — are types for binary infix operators

fx , fy — types for prefix operators

xf , yf — types for postfix operators

f $\hat{=}$ operator

x $\hat{=}$ argument whose precedence is smaller than prec. of f

y $\hat{=}$ —— " ———— is smaller or equal to the precedence of f.

Precedence of an argument: Prec. of its leading operator.

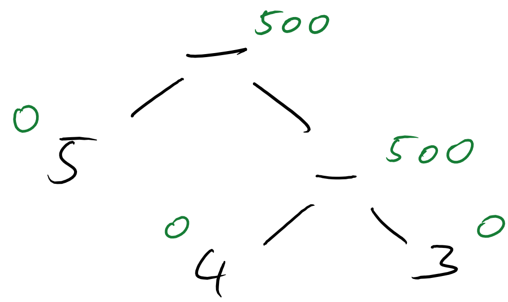
Precedence of standard (non-operator) pred. and fct. symbols is 0.

"Functors" $\rightarrow$ standard (non-operator) pred. and fct. symbols

Arguments in (...) also have prec. 0.

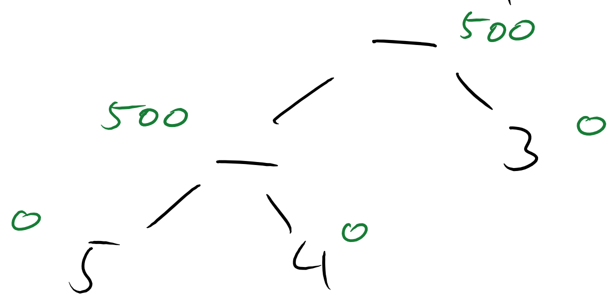
So yfx means that arguments with the same preced. as f can only occur on f's left-hand side.

5 - 4 - 3 Can this stand for 5-(4-3)?

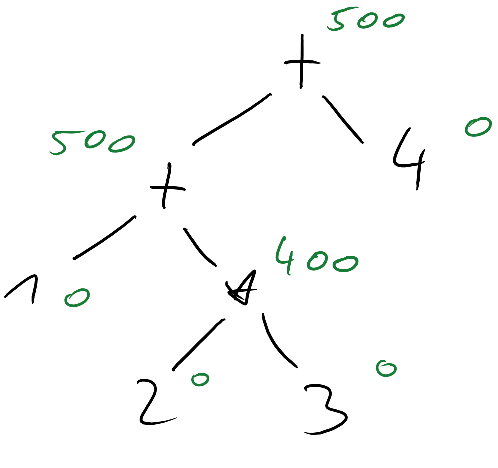


No, because then the right argument would also have prec. 500.

So it stands for



What does $1 + 2 * 3 + 4$ stand for?



Both + and * have type yfx.

$$(1 + (2 * 3)) + 4$$

yfx : association to the left

xfy : ——— " ——— right

xfx : no association

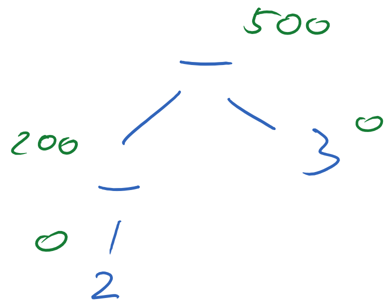
(1+2+3 would not be allowed)

Overloading of operators is possible.

Ex: In addition to the binary -
there is also a unary -.

:- op(200, fy, -).

-2-3 stands for



i.e., for (-2)-3

Operators can also be used for a simple form of natural language processing.

"was" is used as a binary operator in infix notation.
It should not have any association:

"laura was young was beautiful"
does not make sense.
⇒ Type xfx

"of": binary operator in infix notation,
should associate to the right

"secretary of son of john" stands for

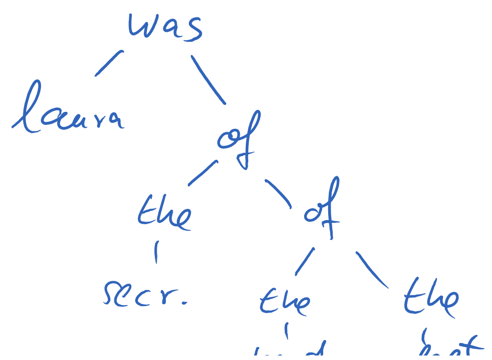
"secretary of (son of john)" ⇒ Type xfy

"the": unary operator, no association
"the the dog" makes no sense ⇒ Type fx

```
:- op(300, xfx, was).
:- op(250, xfy, of).     ← "of" binds stronger than "was"
:- op(200, fx, the).     ← "the" binds stronger than "of".
```

laura was the secretary of the head of the department.
stands for

was(laura, of(the(secr), of(...))).



secr. the head the dept

?- 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