Arithmetic (12A)

Young W. Lim
4/9/14

based on the following document:
http://www.learnprolognow.org/ Learn Prolog Now!

Please send corrections (or suggestions) to youngwlim@hotmail.com.

This document was produced by using LibreOffice/OpenOffice.

Arithmetic in Prolog

```
?- 8 is 6+2.
true.

?- X is 6+2.
X = 8

?- 6+2 is 8.
false.

?- 6+2 is X .
ERROR...
```

```
?- 8 = 6+2.
false.

?- X = 6+2.
X = 6+2

?- 6+2 = 8.
false

?- 6+2 = X.
X = 6+2
```

```
new_pred(X,Y) :- Y is (X+3)*2.

?- new_pred(1,X).
X = 8

?- new_pred(X,3).
Error

new_pred(X,3) :- 3 is (X+3)*2.
```

Term-based Arithmetic

3 + 2 ← a term
in a user-friendly notation

+(3,2) ← the same term

X is 3 + 2

is(X, +(3,2))

- expression need to be evaluated must go to the **right** of **is**
- let every variable be correctly instantiated

difference between the **procedural** and **declarative** meanings

Unification

testing whether X unifies with Y
not just testing mathematical equality

$$X = Y$$

- a variable
- an atom
- a complex term

- a variable
- an atom
- a complex term

two atoms, including numeric atoms,
unify if they are the same.

two more complex terms unify
if they have the same functor and
their corresponding arguments unify.

A variable always unifies with a term
(provided that it is not previously unified with something different)
by binding to that term

Arithmetic Operators (1)

X < Y.	x < y
X =< Y.	x ≤ y
X =:= Y.	x = y
X =\= Y.	x ≠ y
X >= Y	x ≥ y
X > Y	x > y

-Number is +Expr

True when Number is the value to which Expr evaluates. Typically, is/2 should be used with unbound left operand. If equality is to be tested, =:=/2 should be used. For example:

?- 1 is sin(pi/2).
Fails! sin(pi/2) evaluates to the float 1.0, which does not unify with the integer 1.
?- 1 =:= sin(pi/2). Succeeds as expecte

Arithmetic Operators (2)

- +Expr
+ +Expr
+Expr1 + +Expr2
+Expr1 - +Expr2
+Expr1 * +Expr2
+Expr1 / +Expr2
+IntExpr1 mod +IntExpr2
+IntExpr1 // +IntExpr2
Integer division
div(+IntExpr1, +IntExpr2)
(IntExpr1 - IntExpr1 mod IntExpr2) // IntExpr2.
+RatExpr rdiv +RatExpr
Rational number division.
+IntExpr1 gcd +IntExpr2
abs(+Expr)
sign(+Expr)
copysign(+Expr1, +Expr2)
matches the sign of Expr2.
max(+Expr1, +Expr2)
min(+Expr1, +Expr2)
random(+IntExpr)
Evaluate to a random integer i for which 0 =< i < IntExpr.

round(+Expr)
integer(+Expr)
float(+Expr)
rational(+Expr)
rationalize(+Expr)
float_fractional_part(+Expr)
float_integer_part(+Expr)
truncate(+Expr)
floor(+Expr)
ceiling(+Expr)
ceil(+Expr)

Arithmetic Operators (3)

+IntExpr1 >> +IntExpr2
+IntExpr1 << +IntExpr2
+IntExpr1 \/ +IntExpr2
+IntExpr1 /\ +IntExpr2
+IntExpr1 xor +IntExpr2
\ +IntExpr

sqrt(+Expr)
sin(+Expr)
cos(+Expr)
tan(+Expr)
asin(+Expr)
acos(+Expr)
atan(+Expr)
atan2(+YExpr, +XExpr)
atan(+YExpr, +XExpr)
sinh(+Expr)
cosh(+Expr)
tanh(+Expr)
asinh(+Expr)
acosh(+Expr)
atanh(+Expr)

log(+Expr)
log10(+Expr)
exp(+Expr)
+Expr1 ** +Expr2
+Expr1 ^ +Expr2
powm(+IntExprBase, +IntExprExp, +IntExprMod)
    Result = (IntExprBase**IntExprExp) modulo IntExprMod.
lgamma(+Expr)
erf(+Expr)
erfc(+Expr)
pi
e
epsilon
cputime
eval(+Expr)

msb(+IntExpr)
lsb(+IntExpr)
popcount(+IntExpr)
Return the number of 1s in the binary representation of the non-negative integer IntExpr.

Arithmetic and Lists

```
len([], 0).

len([_|T], N) :-  len(T, X),  N  is  X+1.
```

The empty list has length zero.
A non-empty list has length 1 + len (Tail)

```
accLen (List, Acc, Length)

accLen([_|T], A, L)  :-   Anew  is  A+1,  accLen(T, Anew, L).

accLen([], A, A).
```

```
leng(List, Length)  :-
        accLen(List, 0, Length).
```

```
accMax([H|T], A, Max)  :-    H  >  A,
                             accMax(T, H, Max).

accMax([H|T], A, Max)  :-    H  =<  A,
                             accMax(T, A, Max).

accMax([],A, A).


max(List, Max)  :-
        List  =  [H|_],
        accMax(List, H, Max)
```

Arithmetic in Prolog

References

[1] en.wikipedia.org
[2] en.wiktionary.org
[3] U. Endriss, "Lecture Notes : Introduction to Prolog Programming"
[4] http://www.learnprolognow.org/ Learn Prolog Now!
[5] http://www.csupomona.edu/~jrfisher/www/prolog_tutorial
[6] www.cse.unsw.edu.au/~billw/cs9414/notes/prolog/intro.html
[7] www.cse.unsw.edu.au/~billw/dictionaries/prolog/negation.html