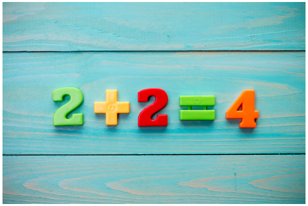
Logic Programming

Arithmetic

Lecturer: Vítor Santos Costa
DCC-FCUP
vsc@dcc.fc.up.pt (room: 1.45)

These slides are largely based on Prof. Inês Dutra's and Prof. Alípio Jorge

Arithmetics in Prolog



In Prolog we can do artihmetics in at least two ways:

- Using logic only and functors;
- Using built-in predicates.

The first option is worth visiting, but the practical solution is usually the second one.

Using the `s/1` functor

The *natural number*.

- The number zero is represented as 0.
- one as $s(0)$, two as $s(s(0))$ and so on.

s is a **functor** of arity 1, i.e. it has one argument.

```
natural(0).
natural(s(X)):-natural(X).
```

Using the s/1 functor

The *sum* example.
How to define the predicate **sum/3** ?

```
sum(0,X,X).
sum(s(X),Y,s(Z)):-sum(X,Y,Z).
```

Using the s/1 functor

The *sum* example.
We can check type.

```
sum(0,X,X):-natural(X).
sum(s(X),Y,s(Z)):-sum(X,Y,Z).
```

Using the `s/1` functor

Querying:

```
?- sum(s(0),s(s(0)),X).
X=s(s(s(0)))
```

We can ask which two numbers added result in a given number.

Using the s/1 functor

Which two numbers added result in a given number.

```
?- sum(X,Y,s(s(s(0)))).

X = 0,
Y = s(s(s(0)))
X = s(0),
Y = s(s(0))
X = s(s(0)),
Y = s(0)
X = s(s(s(0))),
Y = 0
```

Using the `s/1` functor

Is a given number pair?

```
pair(X): - sum(Y,Y,X).
```

Multiplication

Multiplication

```
mult(0,_,0).
mult(s(X),Y,Z):-mult(X,Y,A),sum(Y,A,Z).
```

Other predicates

Define predicates

- **gte/2** (greater than or equal).
- **minimum/3**.
- **mod/3**.

Power

Define predicate **exp/3**, such that `exp(X,N,Y)` is true if $X^N = Y$.

```
exp(0,s(0),0).
...
```

Factorial

Define predicate **fact/2**, such that `fact(X,Y)` is true if $Y = X!$.

```
fact(0,s(0)).
...
```

Using built in operators

```
sum(X,Y,Z) :- Z is X+Y. % X and Y must be numbers.
mult(X,Y,Z) :- Z is X*Y. % Works for floats as well.
gte(X,Y) :- X>=Y.
```

- Common operators are available.
- Common functions are available too: *sin*(x), *exp*(x), *log*(x), …

For SWI prolog, you can find more in `http://www.swi-prolog.org/pldoc/man?section=arith`.

Using built in operators

- Define `fact/2` using built in predicates.
- Common functions are available too: $sin(x)$, $exp(x)$, $log(x)$, ...