Contents

Arithmetic in Prolog

- Prolog provides a number of basic arithmetic tools
- Integer and real numbers

```
?- 5 is 2+3.
true.
?- 12 is 3*4.
true.
?- 2 is 5-4.
false.
?- X is 3-5.
X=-2
true.
?- 2 is 4/2.
true.
?- X is mod(7,2).
X=1
true.
```

Defining

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

?- addThreeAndDouble(1, X).
X=8
```

```
true.
?- addThreeAndDouble(2, X).
X=10
true.
```

A Closer Look

- It is important to know that `+`, `-`, `/` and `*` do not carry out any arithmetic
- Expressions such as `3+2`, `4-7`, `5/5` are ordinary Prolog terms
  - Functor: +, -, /, *
  - Arity: 2
  - Arguments: integers

```
?- X=3+2.
X=3+2
true.
?- 3+2=X
X=3+2
true.
```

The is/2 Predicate

- To force Prolog to actually evaluate arithmetic expressions, we have to use `is`
- This is an instruction for Prolog to carry out calculations
- Because this is not an ordinary Prolog predicate, there are some restrictions

```
?- X is 3+2.
X=5
true.
?- 3+2 is X.
ERROR: is/2: Arguments are not sufficiently instantiated
```

Restrictions

- We are free to use variables on the right hand side of the is predicate
- But when Prolog actually carriers out the evaluation, the variables must be instantiated with a variable-free Prolog term
- This Prolog term must be an arithmetic expression

Notation

- Two final remarks on arithmetic expressions
  - `3+2`, `4/2`, `4-5` are just ordinary Prolog terms in a use friendly notation: `3+2` is really `+(3, 2)` and so on
  - Also the `is` predicate is a two-place Prolog predicate

```
?- is(X, +(3, 2)).
X=5
true.
```

Arithmetic and Lists

- How long is a list?
  - The empty list has length zero
  - A non empty list has a length one plus length of its tail

```
len([], 0).
len([_|L], N):- len(L, X), N is X+1.

?- len([a, b, c, d, e, [a, x], t), X).
X=7
true.
```

Accumulators

- This is quite a good program
  - Easy to understand
  - Relatively efficient
- But there is another method of finding the length of a list
  - Introduce the idea of accumulators
  - Accumulators are variables that hold intermediate results
- The predicate `acclen/3` has three arguments
  - The list whose length we want to find
  - The length of the list, an integer
  - An accumulator, keeping track of the intermediate values for the length
- The accumulator of `acclen/3`

- Initial value of the accumulator is `0`
- Add `1` to accumulator each time we can recursively take the head of a list
- When we reach the empty list, the accumulator contains the length of the list