More Prolog

Hacks and features of Prolog making it into a full programming language:

- General data structures and lists
- Control facilities
- Arithmetic in Prolog
- Syntactic extensibility: Operator notation
- (Self-inspection and modification)

That's really all of it!
Basic notions, now adding **structures**

- **predicates**: `parent`
  - defines a **relation**
  - given by **facts**, **rules**, coll. called **clauses**
- **constants**: `tom`, `bob`, `x`, `y`
- **variables**: `X`, `Y`, `Tom`
- **atoms**: `parent(A,a)`
- Arguments to predicates can also be **structures**:
  ```
  point(1,1)
  line_segment(point(1,1),point(3,3))
  ```

**NB**: Looks like pred's with arguments, ...
An example of using structures

Intuitive interpretation of structure:

This is a program:

```
vertical( line_segment(point(X,Y), point(X,Y1))).
horizontal( line_segment(point(X,Y), point(X1,Y))).
```

Querying it:

```
?- vertical(line_segment(point(1,1),point(2,Y))).
   no
?- horizontal(line_segment(point(1,1),point(2,Y))).
   Y = 1 ?
```
Lists, an important sort of structures

List syntax ≈ syntactic sugar; no new semantics

?- write([1,2,3,4,5,6]).
[1,2,3,4,5,6]

?- write_canonical([1,2,3,4,5,6]).
'(1,''(2,''(3,''(4,''(5,''(6,''[])'))'))

?- [1,2,3,4,5,6] = [Head | Tail].
Head = 1, Tail = [2,3,4,5,6]

?- [First, Second | Tail2] = [a,b,c,d,e,f].
First = a, Second = b, Tails = [c,d,e,f]
Working with lists; the member predicate

A built-in predicate; in SICStus (v. 3, not 4, sic!) remember this:
:- use_module(library(lists)).

?- member(a,[a,b,c]).
yes

Member is also a list constructor:

?- member(a,L), member(b,L), member(c,L).
L = [a,b,c|_A]

Implementation of member

member(X, [X | _] ).
member(X, [_|L]):- member(X,L).
"append": List concat'n & decomp'n

Examples:

?- append([a,b],[c,d], L).
L = [a,b,c,d]

?- append(X,Y,[a,b,c]).
X = [], Y = [a,b,c] ? ;
X = [a], Y = [b,c] ? ;
X = [a,b], Y = [c] ? ;
X = [a,b,c], Y = [] ? ;
A definition of "append"

append([], L, L).

append([X|L1], L2, [X|L3]) :- append(L1, L2, L3).
Arithmetic, a stepchild in Prolog

?- X is 2 + 2 * 3.
X = 8 ?

?- X is 2 + Y * 3.

! Instantiation error in argument 2 of is/2
! goal: _79 is 2+_73*3

Remember points about
• range-restrictedness
• left-to-right execution
Exercises

• 5.1, p. 46
• 5.2, p. 46–47.
  – Only triangles, \texttt{identical_triangles, segment_length}
    and possibly \texttt{area(<triangle>, <length>)}
• 5.3, p.47.
• Extra: Define, using append, a predicate \texttt{find_abc(L)}, which is satisfied iff \([a, b, c]\) is a
  “sublist” of \texttt{L}, e.g.
  
  \begin{verbatim}
  ?- find_abc([k, l, m, n, a, b, c, d, e])
  yes
  ?- find_abc([k, l, m, n, a, b, z, z, c, d, e])
  no
  \end{verbatim}

  (can be done with just one call to \texttt{append})
Useful built-ins (use with care)

\[\text{var}(\text{arg}) \quad \text{— argument currently uninstantiated?}\]
\[\text{nonvar}(\text{arg}) \quad \text{— the opposite}\]
\[\text{ground}(\text{arg}) \quad \text{— is current value of arg ground, i.e., variable-free?}\]
\[\text{atom}(\text{arg}) \quad \text{— current value constant that is not a number?}\]
\[\text{integer}(\text{arg}) \quad \text{— current value an integer number?}\]
\[\text{atomic}(\text{arg}) \quad \text{— current value a constant?}\]

Splitting terms by "+="

?- \text{f}(a,b) =.. [F|\text{Args}].
\text{F} = \text{f}, \text{Args} = [a,b]
?- \text{f}(a,b) =.. [F|\text{Args}], \text{NewTerm} =.. [F,\text{new}|\text{Args}].
... , \text{NewTerm} = \text{f}(\text{new},a,b)
Control of backtracking by "!" (cut)

```prolog
salary(S, 0):- student(S), !.
salary(S, 1000000).
student(peter).

?- salary(peter, S).
S = 0 ;
no
?- salary(jane, S).
S = 1000000 ;
no

But trying to generate all solution :(

?- salary(X, S).
X = peter, S = 0 ;
no
```

Be careful:
- Destroys logic
- Introduces assumptions about how predicates are called
Conditionals

salary(X,S):-
    student(X) -> S=0
;
    director(X) -> S=1000000
;
    professor(X) -> S=500000
;
    S = 10.

Like a "soft-cut", successful-test-and-choice not backtracked, but subsequent clause may be used.
Operators: Extensible syntax

:- op(700, xfx, sparker).
manden sparker hunden.
:- op(700, xfx, bider).
X bider Y :- Y sparker X.

Important: Only syntactic sugar, no new semantics

?- current_op(X, Y, Z).
X = 1200, Y = xfx, Z = :- ? ;
X = 1200, Y = xfx, Z = --> ? ;
...
X = 1000, Y = xfy, Z = ',' ? ;
...
X = 500, Y = yfx, Z = + ? ;
...
X = 400, Y = yfx, Z = * ? ;
Example of program with operators

:- op(700, xfx, er).
:- op(100, fx, [en,et]).
en mand er et menneske.
en kvinde er et menneske.
et menneske er et dyr.
en ko er et dyr.
peter er en mand.
X er Z :- X er Y, Y er Z.
Other facilities

Generating all solutions:

\texttt{setof, bagof, findall}

— read about them when you need them

Input-output:

\texttt{write('Hello')} useful for test prints...

Inspecting and modifying the program

\texttt{clause, asserta, assertz, retract}

We may see those guys later in the course

This is really all of Prolog!