Declarative Programming
CS-205
Logic Programming (Prolog)

Monika Seisenberger, Ulrich Berger

Department of Computer Science
College of Science
Swansea University

CS-205 - Week8-B

Recursion

An important method of programming in Prolog is *recursion*.
Already met in Java and Haskell.

- As a first example, the factorial program (which looks at first glance, different to what we are used to, but we show it anyway already now. Note the examples discussed on the next slides look simpler to start with.)

```
fact(0,1).              % fact(0) = 1

fact(N,F) :-            % fact(n) = n * fact(n-1)
        N>0,
        N1 is N-1,
        fact(N1,F1),
        F is N*F1.
```

  Note this program requires arithmetic and the `is` predicate. [Both is introduced in a later chapter.]

Examples from Learn Prolog Now, Chapter 3

```
isDigesting(X,Y):- justAte(X,Y).
isDigesting(X,Y):- justAte(X,Z), isDigesting(Z,Y).

justAte(mosquito,blood(john)).
justAte(frog,mosquito).
justAte(stork,frog).
```

Who is digsting whom? How many answers will Prolog give?

Descendants

```
child(bridget,caroline).
child(caroline,donna).
```

```
descend(X,Y):- child(X,Y).
descend(X,Y):- child(X,Z), child(Z,Y).
```

Who is a descendant of whom?

Descendants (2)

```
child(anne,bridget).
child(bridget,caroline).
child(caroline,donna).
child(donna, emily).

descend(X,Y):- child(X,Y).
descend(X,Y):- child(X,Z), child(Z,Y).
```

Who many different answers does descend(A,B) give?
Who is not a descendant in this program?
How to improve this program?

Descendants (2)

```
child(anne,bridget).
child(bridget,caroline).
child(caroline,donna).
child(donna, emily).

descend(X,Y):- child(X,Y).
descend(X,Y):- child(X,Z), descend(Z,Y).
```

(1) Who many different answers does descend(A,B) give now?
(2) What happens if we swap the order of rules or the goals in the rules?

The shortest recursive program

```
p :- p
```

Programs with recursive calls, always need a second goal or rule, to terminate.

Prolog and Logic

- Programmer gives a declarative specification of the problem, using the language of logic
- The programmer should not have to tell the computer what to do
- To get information, the programmer simply asks a query
- As we've seen, Prolog has a specific way of answering queries:
  1. Search knowledge base from top to bottom
  2. Processes bodies of clauses from left to right
  3. Backtracking to recover from bad choices or select other choices
- Look at different versions of descendant

Numerals

```
numeral(0).
numeral(succ(X)):- numeral(X).
```

```
?- numeral(succ(succ(succ(0)))).
```

What does `?- numeral(X).` yield?

Defining Addition recursively.

We want a program that adds two numbers. It should (1) be able to show the correctness of an addition, and (2) also give results as below:

```
?- add(succ(succ(0)),succ(succ(succ(0))), Result).
Result=succ(succ(succ(succ(succ(0)))))
```

add/3 for numerals

```
add(0,X,X).                  %%% base clause

add(succ(X),Y,succ(Z)):-     %%% recursive clause
      add(X,Y,Z).
```

Optional Tasks: Week 8

Define a predicate `double/2` that doubles a numeral. Do this in two ways, once using the `add/3` predicate, and once defining it from scratch using recursion. For each do at least two queries, one verifiying a computation with numerals, and one that will indeed compute the double.

Optional task: descend4.pl

```
child(anne,bridget).
child(bridget,caroline).
child(caroline,donna).
child(donna, emily).

descend(X,Y):- child(X,Y).
descend(X,Y):- descend(Z,Y),child(X,Z).
```

Task: The book claims that the query `?-descend(anne, emily)` will loop. Try what result you get. Can you explain? Does this program behave exactly as descend1.pl? Explain the difference, eg by trying `?-descend(anne,X).`