

## Lecture 3: Recursion

---

© Patrick Blackburn, Johan Bos & Kristina Striegnitz

- Theory
  - Introduce recursive definitions in Prolog
  - Four examples
  - Show that there can be mismatches between the declarative and procedural meaning of a Prolog program
- Exercises
  - Exercises of LPN chapter 3
  - Practical work

## Recursive Definitions

---

© Patrick Blackburn, Johan Bos & Kristina Striegnitz

- Prolog predicates can be defined recursively
- A predicate is recursively defined if one or more rules in its definition refers to itself

## Example 1: Eating

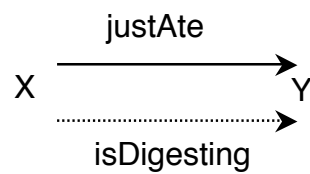
```
isDigesting(X,Y):- justAte(X,Y).  
isDigesting(X,Y):- justAte(X,Z), isDigesting(Z,Y).
```

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

?-

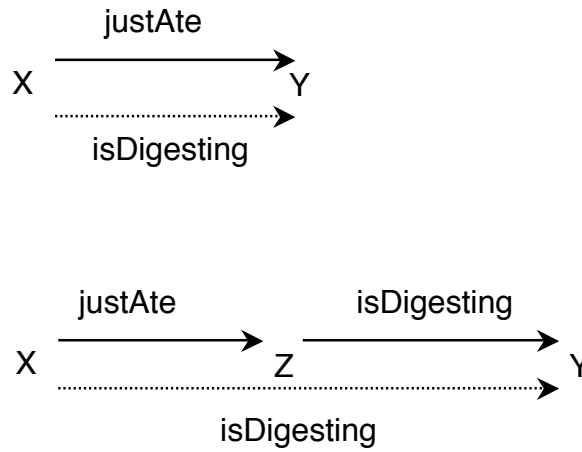
© Patrick Blackburn, Johan Bos & Kristina Striegnitz

## Picture of the situation



© Patrick Blackburn, Johan Bos & Kristina Striegnitz

## Picture of the situation



© Patrick Blackburn, Johan Bos & Kristina Striegnitz

## Example 1: Eating

```
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).
```

```
?- isDigesting(stork,mosquito).
```

© Patrick Blackburn, Johan Bos & Kristina Striegnitz

## Another recursive definition

p:- p.

?-

© Patrick Blackburn, Johan Bos & Kristina Striegnitz

## Another recursive definition

p:- p.

?- p.

© Patrick Blackburn, Johan Bos & Kristina Striegnitz

## Another recursive definition

p:- p.

?- p.  
ERROR: out of memory

© Patrick Blackburn, Johan Bos & Kristina Striegnitz

## Example 2: Decendant

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

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

© Patrick Blackburn, Johan Bos & Kristina Striegnitz

## Example 2: Decendant

```
child(anna,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).
```

© Patrick Blackburn, Johan Bos & Kristina Striegnitz

## Example 2: Decendant

```
child(anna,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).
```

```
?- descend(anna,donna).
no
?-
```

© Patrick Blackburn, Johan Bos & Kristina Striegnitz

## Example 2: Decendant

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

?-

## Example 2: Decendant

```
child(anna,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).
```

?-

## Example 2: Decendant

```
child(anna,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).
```

```
?- descend(anna,donna).
```

© Patrick Blackburn, Johan Bos & Kristina Striegnitz

## Search tree

- Draw search tree for

```
?- descend(anna,donna).
```

© Patrick Blackburn, Johan Bos & Kristina Striegnitz



## Example 3: Successor

- Suppose we use the following way to write numerals:
  1. **0** is a numeral.
  2. If **X** is a numeral, then so is **succ(X)**.

© Patrick Blackburn, Johan Bos & Kristina Striegnitz

## Example 3: Successor

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

© Patrick Blackburn, Johan Bos & Kristina Striegnitz

## Example 3: Successor

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

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

© Patrick Blackburn, Johan Bos & Kristina Striegnitz

## Example 3: Successor

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

```
?- numeral(X).
```

© Patrick Blackburn, Johan Bos & Kristina Striegnitz

## Example 3: Successor

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

```
?- numeral(X).  
X=0;  
X=succ(0);  
X=succ(succ(0));  
X=succ(succ(succ(0)));  
X=succ(succ(succ(succ(0))))
```

© Patrick Blackburn, Johan Bos & Kristina Striegnitz

## Example 4: Addition

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

© Patrick Blackburn, Johan Bos & Kristina Striegnitz

## Example 4: Addition

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

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

© Patrick Blackburn, Johan Bos & Kristina Striegnitz

## Example 4: Addition

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

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

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

© Patrick Blackburn, Johan Bos & Kristina Striegnitz

## Search tree

---

- Draw search tree

© Patrick Blackburn, Johan Bos & Kristina Striegnitz

## Exercises

---

© Patrick Blackburn, Johan Bos & Kristina Striegnitz

## Prolog and Logic

© Patrick Blackburn, Johan Bos & Kristina Striegnitz

- Prolog was the first reasonable attempt to create a logic programming language
  - 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

## Prolog and Logic

© Patrick Blackburn, Johan Bos & Kristina Striegnitz

- Prolog does some important steps in this direction, but nevertheless, Prolog is not a full logic programming language!
- Prolog has a specific way of answering queries:
  - Search knowledge base from top to bottom
  - Processes clauses from left to right
  - Backtracking to recover from bad choices

## descend1.pl

```
child(anna,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).
```

```
?- descend(A,B).
A=anna
B=bridget
```

© Patrick Blackburn, Johan Bos & Kristina Striegnitz

## descend2.pl

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

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

```
?- descend(A,B).
A=anna
B=emily
```

© Patrick Blackburn, Johan Bos & Kristina Striegnitz

## descend3.pl

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

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

```
?- descend(A,B).
ERROR: OUT OF LOCAL STACK
```

© Patrick Blackburn, Johan Bos & Kristina Striegnitz

## descend4.pl

```
child(anna,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).
```

```
?- descend(A,B).
```

© Patrick Blackburn, Johan Bos & Kristina Striegnitz



## Summary of this lecture

- In this lecture we introduced recursive predicates
- We also looked at the differences between the declarative and the procedural meaning of Prolog programs
- We have identified some of the shortcomings of Prolog seen as a logical programming language

© Patrick Blackburn, Johan Bos & Kristina Striegnitz

## Next lecture

- Introduce **lists** in Prolog
  - Important recursive data structure in Prolog programming
  - Define the member/2 predicate, a fundamental Prolog tool for working with lists
  - Discuss the idea of recursing down lists

© Patrick Blackburn, Johan Bos & Kristina Striegnitz