Lecture 3: Recursion

• 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

- 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

\[
\text{isDigesting}(X,Y) :- \text{justAte}(X,Y).
\]

\[
\text{isDigesting}(X,Y) :- \text{justAte}(X,Z), \text{isDigesting}(Z,Y).
\]

\[
\text{justAte}(\text{mosquito},\text{blood}(\text{john})).
\]

\[
\text{justAte}(\text{frog},\text{mosquito}).
\]

\[
\text{justAte}(\text{stork},\text{frog}).
\]

?-
Picture of the situation
Picture of the situation

justAte

X → Y

isDigesting

justAte     isDigesting

X → Z → Y

isDigesting
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).
Another recursive definition

Another recursive definition

?- p.
p:- p.
Another recursive definition

\[ \text{p:- p.} \]

\[ \text{?- p. ERROR: out of memory} \]
Example 2: Descendant

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

descend(X, Y): - child(X, Y).
descend(X, Y): - child(X, Z), child(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), child(Z,Y).
Example 2: Descendant

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
?-
Example 2: Descendant

\[
\begin{align*}
\text{child(anna,bridget).} \\
\text{child(bridget,caroline).} \\
\text{child(caroline,donna).} \\
\text{child(donna,emily).} \\
\text{descend(X,Y):- child(X,Y).} \\
\text{descend(X,Y):- child(X,Z), child(Z,Y).} \\
\text{descend(X,Y):- child(X,Z), child(Z,U), child(U,Y).} \\
\end{align*}
\]
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: Descendant

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).
Search tree

• Draw search tree for

?- descend(anna,donna).
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 $\text{succ}(X)$. 
Example 3: Successor

umeral(0).
umeral(succ(X)) :- numeral(X).
Example 3: Successor

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

?- numeral(succ(succ(succ(0)))).
yes
?-
Example 3: Successor

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

?- numeral(X).
Example 3: Successor

```prolog
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))));
X=succ(succ(succ(succ(succ(0)))))
```
Example 4: Addition

?- add(succ(succ(0)), succ(succ(succ(0))), Result).
Result = succ(succ(succ(succ(succ(0))))))
yes
Example 4: Addition

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

?- add(succ(succ(0)),succ(succ(succ(succ(0)))), Result).
Result=succ(succ(succ(succ(succ(succ(succ(succ(succ(succ(0)))))))))))
yes
Example 4: Addition

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

add(succ(X), Y, succ(Z)):-
    add(X, Y, Z).

add(succ(succ(0)), succ(succ(succ(0))), Result).
Result = succ(succ(succ(succ(succ(0)))))
yes
```
Draw search tree
Prolog and Logic

• 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 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
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
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
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
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).
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
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