Administrivia

• Today’s Topics
  1. Homework 2 review
  2. Lists (Chapter 4 of learnprolognow.org)
     Lab Exercises
Homework Exercise 1

• Consider Exercise 4 again.
  loves(vincent,mia).
  loves(marcellus,mia).

|jealous(A,B):- loves(A,C), loves(B,C).

• Assume one cannot be jealous of oneself.

• How would you change the rule or query so that this case is excluded from the answers?

Query:
jealous(A,B), \+ A = B.
Consider Exercise 4 again.

\[\text{loves(vincent,mia).} \]
\[\text{loves(marcellus,mia).} \]

\[\text{jealous(A,B)} : \neg \text{loves(A,C), loves(B,C).} \]

Assume one cannot be jealous of oneself.

How would you change the rule or query so that this case is excluded from the answers?

**Rule:**
\[\text{jealous(A,B)} : \neg \text{loves(A,C), loves(B,C), \neg A = B.} \]

**Query:**
\[\text{jealous(A,B)}.\]
Homework Exercise 2

• A word that’s a palindrome is spelt the same way backwards and forwards, e.g. kayak, radar or noon. We can check for “palindrome-­‐hood” using Prolog lists.

• Run the queries:
  – [k,a,y,a,k] = [X,Z,Y,Z,X]. (5 letter palindrome)
  – [c,a,n,o,e] = [X,Z,Y,Z,X].

1. Where can we use the anonymous variable (_,) in the palindrome check above?
2. What’s the four letter version of the palindrome check?

  1. [X, Z, _, Z, X]  
  2. [X, Z, Z, X]
Homework Exercise 3

• There’s a built-in predicate called reverse(List₁, List₂). Run the following queries:
  – reverse([1, 2, 3], [3, 2, 1]).
  – reverse([1, 2, 3], L).
  – reverse(L, [1, 2, 3]).

• Explain how you can use reverse/2 to check for palindromes of any length.

Query:
Word = [r, a, c, e, c, a, r], reverse(Word, Word).
Homework Exercise 4

• Extra Credit. Write a rule and query (or query) that solves the following puzzle:

Exercise 2.4 Here are six Italian words:
astante, astoria, baratto, cobalto, pistola, statale.
They are to be arranged, crossword puzzle fashion, in the following grid:

Constraint is:

H1 = [__,H1V1,__,H1V2,__,H1V3,__],
H2 = [__,H2V1,__,H2V2,__,H2V3,__],
H3 = [__,H3V1,__,H3V2,__,H3V3,__],
V1 = [__,H1V1,__,H2V1,__,H3V1,__],
V2 = [__,H1V2,__,H2V2,__,H3V2,__],
V3 = [__,H1V3,__,H2V3,__,H3V3,__].
Extra Credit. Write a rule and query (or query) that solves the following puzzle:

```
word([a,s,t,a,n,t,e]).
word([a,s,t,o,r,i,a]).
word([b,a,r,a,t,t,o]).
word([c,o,b,a,l,t,o]).
word([p,i,s,t,o,l,a]).
word([s,t,a,t,a,l,e]).
```

```
word(H1),
word(H2), \+ H1 = H2,
word(H3), \+ H1 = H3, \+ H2 = H3,
word(V1), \+ H1 = V1, \+ H2 = V1, \+ H3 = V1,
word(V2), \+ H1 = V2, \+ H2 = V2, \+ H3 = V2, \+ V1 = V2,
word(V3), \+ H1 = V3, \+ H2 = V3, \+ H3 = V3, \+ V1 = V3, \+ V2 = V3,
```
Homework Exercise 4

- Final program: put all the constraints together.

```prolog
puzzle([H1,H2,H3,V1,V2,V3]) :-
  word(H1),
  word(H2), \+ H1 = H2,
  word(H3), \+ H1 = H3, \+ H2 = H3,
  word(V1), \+ H1 = V1, \+ H2 = V1, \+ H3 = V1,
  word(V2), \+ H1 = V2, \+ H2 = V2, \+ H3 = V2, \+ V1 = V2,
  word(V3), \+ H1 = V3, \+ H2 = V3, \+ H3 = V3, \+ V1 = V3, \+ V2 = V3,
H1 = [_,H1V1,_,H1V2,_,H1V3,_,]
H2 = [_,H2V1,_,H2V2,_,H2V3,_,]
H3 = [_,H3V1,_,H3V2,_,H3V3,_,]
V1 = [_,H1V1,_,H2V1,_,H3V1,_,]
V2 = [_,H1V2,_,H2V2,_,H3V2,_,]
V3 = [_,H1V3,_,H2V3,_,H3V3,_,].
```

```prolog
word([a,s,t,a,n,t,e]).
word([a,s,t,o,r,i,a]).
word([b,a,r,a,t,t,o]).
word([c,o,b,a,l,t,o]).
word([p,i,s,t,o,l,a]).
word([s,t,a,t,a,l,e]).
```
Homework Exercise 4

Solution 1:

<table>
<thead>
<tr>
<th></th>
<th>V1</th>
<th>V2</th>
<th>V3</th>
</tr>
</thead>
<tbody>
<tr>
<td>H1</td>
<td>abstante</td>
<td></td>
<td></td>
</tr>
<tr>
<td>H2</td>
<td>cobaltol</td>
<td></td>
<td></td>
</tr>
<tr>
<td>H3</td>
<td>pistola</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Solution 2:

<table>
<thead>
<tr>
<th></th>
<th>V1</th>
<th>V2</th>
<th>V3</th>
</tr>
</thead>
<tbody>
<tr>
<td>H1</td>
<td>astoria</td>
<td></td>
<td></td>
</tr>
<tr>
<td>H2</td>
<td>barattoo</td>
<td></td>
<td></td>
</tr>
<tr>
<td>H3</td>
<td>statale</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Last Time

• Recursive definitions:
  – Base case
  – Recursive case

• Example:
  factorial(0,1).
  factorial(N,F) :-
    N > 0,
    N1 is N-1,
    factorial(N1,F1),
    F is N*F1.

• Example:
  length_of([],0).
  length_of([_|List],Len) :-
    length_of(List,N),
    Len is N+1.

• Example:
  numeral(0).
  numeral(succ(X)) :-
    numeral(X).
member

• Example:
  – list [1,3,5]
  – ask: is 5 a member of this list? (Yes: true)
  – ask: is 4 a member of this list? (No: false)

• In Prolog:
  member(X,[X|T]).
  member(X,[H|T]) :- member(X,T).

• Queries:
  ?- member(yolanda,[yolanda,trudy,vincent,jules]).
  ?- member(vincent,[yolanda,trudy,vincent,jules]).
  ?- member(X,[yolanda,trudy,vincent,jules]).
Exercise 1

Same number of a’s and b’s

• **4.3 Recursing down Lists**

• Let’s suppose we need a predicate a2b/2 that takes two lists as arguments, and succeeds if the first argument is a list of a’s, and the second argument is a list of b’s of exactly the same length.

• **Examples:**

  – ?- a2b([a,a,a,a],[b,b,b,b]). true
  – ?- a2b([a,a,a,a],[b,b,b]). false
  – ?- a2b([a,c,a,a],[b,b,5,4]). false
Exercise 1

Same number of a’s and b’s

• Part 1: *what is the base case?*
  – In an empty list, we have the same number of a’s and b’s...
  – Write the Prolog code
Exercise 1

Same number of a’s and b’s

• Part 2: what is the recursive case?
  – Given two lists \([X|L1]\) for a’s and \([Y|L2]\) for b’s, what needs to be recursively true...
  – Write the Prolog code
Exercise 2

• Exercise 4.5:
  database
  tran(eins,one).
  tran(zwei,two).
  tran(drei,three).
  tran(vier,four).
  tran(fuenf,five).
  tran(sechs,six).
  tran(sieben,seven).
  tran(acht,eight).
  tran(neun,nine).

• Write a predicate listtran(G,E) which translates a list of German number words to the corresponding list of English number words.

• For example:
  ?- listtran([eins,neun,zwei],X).
      X = [one,nine,two]
  ?- listtran(X,[one,seven,six,two]).
      X = [eins,sieben,sechs,zwei]
Exercise 2

German/English word list translation

• Part 1: *what is the base case?*
  – The empty list ...
  – Write the Prolog code
Exercise 2

German/English word list translation

• Part 2: *what is the recursive case?*
  
  – Given two lists `[G|L1]` for German and `[E|L2]` for English, what needs to be recursively true...
  
  – Write the Prolog code