Lecture 2

- Theory
  - Unification
  - Unification in Prolog

Revised Definition 1/3

1. If T_1 and T_2 are constants, then
   T_1 and T_2 unify if they are the same atom, or
   the same number.

Revised Definition 2/3

1. If T_1 and T_2 are constants, then T_1 and T_2 unify if they are the same atom, or the same number.
2. If T_1 is a variable and T_2 is any type of term, then T_1 and T_2 unify, and T_1 is instantiated to T_2. (and vice versa)

Revised Definition 3/3

1. If T_1 and T_2 are constants, then T_1 and T_2 unify if they are the same atom, or the same number.
2. If T_1 is a variable and T_2 is any type of term, then T_1 and T_2 unify, and T_1 is instantiated to T_2. (and vice versa)
3. If T_1 and T_2 are complex terms then they unify if:
   a) They have the same functor and arity, and
   b) all their corresponding arguments unify, and
   c) the variable instantiations are compatible.

Prolog unification: =/2

```
?- mia = mia.

yes
?-
```

Prolog unification: =/2

```
?- mia = mia.
yes
?- mia = vincent.
no
?-
```

Prolog unification: =/2

```
?- mia = X.
X=mia
yes
?-
```

How will Prolog respond?

?- X=mia, X=vincent.

How will Prolog respond?

?- X=mia, X=vincent.

no

?-

Why? After working through the first goal, Prolog has instantiated X with **mia**, so that it cannot unify it with **vincent** anymore. Hence the second goal fails.

Example with complex terms

?- k(s(g),Y) = k(X,t(k)).

Example with complex terms

```
?- k(s(g),Y) = k(X,t(k)).
X=s(g)
Y=t(k)
yes
?-
```

Example with complex terms

?- k(s(g),t(k)) = k(X,t(Y)).

Example with complex terms

?- k(s(g),t(k)) = k(X,t(Y)).

X=s(g)

Y=k

yes

?-

One last example

?- loves(X,X) = loves(marsellus,mia).

Programming with Unification

```
vertical( line(point(X,Y),
               point(X,Z))).

horizontal( line(point(X,Y),
                 point(Z,Y))).
```

Programming with Unification

```
vertical( line(point(X,Y),
               point(X,Z))).

horizontal( line(point(X,Y),
                 point(Z,Y))).
```

```
?-
```

Programming with Unification

```
vertical( line(point(X,Y),
               point(X,Z))).

horizontal( line(point(X,Y),
                 point(Z,Y))).
```

```
?- vertical(line(point(1,1),point(1,3))).
yes
?-
```

Programming with Unification

```
vertical( line(point(X,Y),
               point(X,Z))).

horizontal( line(point(X,Y),
                 point(Z,Y))).
```

```
?- vertical(line(point(1,1),point(1,3))).
yes
?- vertical(line(point(1,1),point(3,2))).
no
?-
```

Programming with Unification

```
vertical( line(point(X,Y),
               point(X,Z))).

horizontal( line(point(X,Y),
                 point(Z,Y))).
```

```
?- horizontal(line(point(1,1),point(1,Y))).
Y = 1;
no
?-
```

Programming with Unification

```
vertical( line(point(X,Y),
               point(X,Z))).

horizontal( line(point(X,Y),
                 point(Z,Y))).
```

```
?- horizontal(line(point(2,3),Point)).
Point = point(_554,3);
no
?-
```

Exercises

```
word(article,a).
word(article,every).
word(noun,criminal).
word(noun,'big kahuna burger').
word(verb,eats).
word(verb,likes).

sentence(Word1,Word2,Word3,Word4,Word5) :-
        word(article,Word1),
        word(noun,Word2),
        word(verb,Word3),
        word(article,Word4),
        word(noun,Word5).
```

Exercises (cont.)

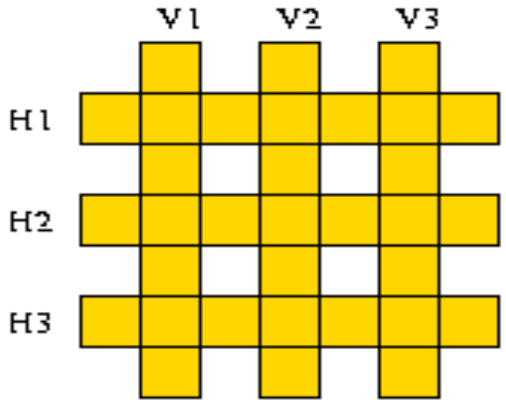
- What query do you have to pose in order to find out which sentences the grammar can generate?
- List all sentences that this grammar can generate in the order Prolog will generate them.
- Make sure that you understand why Prolog generates them in this order.

Exercises (cont.)

Here are six English words:

abalone, abandon, anagram, connect, elegant, enhance.

They are to be arranged in a crossword puzzle like fashion in the grid given below.



Exercises (cont.)

- The following knowledge base represents a lexicon containing these words.

```
word(abalone,a,b,a,l,o,n,e).
word(abandon,a,b,a,n,d,o,n).
word(enhance,e,n,h,a,n,c,e).
word(anagram,a,n,a,g,r,a,m).
word(connect,c,o,n,n,e,c,t).
word(elegant,e,l,e,g,a,n,t).
```

- Write a predicate crosswd/6 that tells us how to fill the grid, i.e. the first three arguments should be the vertical words from left to right and the following three arguments the horizontal words from top to bottom.