Programming Paradigms 2021-2022
Prolog - Reeks 1 & 2

Brent van Bladel
brent.vanbladel@uantwerpen.be

1. Try to answer the following questions first "by hand" and then verify your answers using a Prolog interpreter.

   - Which of the following are valid Prolog atoms?

     `f, loves(john,mary), Mary, _c1, 'Hello', this_is_it`

   - Which of the following are valid names for Prolog variables?

     `a, A, Paul, 'Hello', a_123, _, _abc, x2`

   - What would a Prolog interpreter reply given the following query?

     `?- f(a, b) = f(X, Y).`

   - Would the following query succeed?

     `?- loves(mary, john) = loves(John, Mary).`

     Why?

   - Assume a program consisting only of the fact a(B, B). How will the system react to the following query?

     `?- a(1, X), a(X, Y), a(Y, Z), a(Z, 100).`

     Why?

2. Represent the following in Prolog:

   - Butch is a killer.
   - Mia and Marcellus are married.
   - Zed is dead.
   - Marcellus kills everyone who gives Mia a foot massage.
   - Mia loves everyone who is a good dancer.
   - Jules eats anything that is nutritious or tasty.

3. Consider the following grammar:

```
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).
```

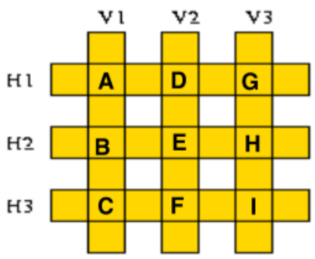
- What query do you have to pose in order to find out which sentences the grammar can generate?
- List all sentences that this grammar can generate.

4. Consider the following knowledge base:

```
%book(ISBN, title, pages)
book(1,'The art of Prolog',400).
book(23,'The mistery of Strawberries',42).
%person(name)
person('Statler').
person('Waldorf').
%author(person, ISBN)
author('Statler',1).
author('Waldorf',23).
%hates(personm ISBN)
hates('Statler',1).
%owns(person, ISBN)
owns('Waldorf',23).
```

- Formulate a query to retrieve ISBN numbers.
- Formulate a query to retrieve the names of books that are hated by their author.
- Formulate a query to retrieve the names of the books together with their author.
  Make Prolog print a line like this for each result: `the book` booktitle `is written by` author name.
  Hint: `write`
- Write a predicate `proud_author` which succeeds for all people that own at least one of the books they have written.

5. Consider the following 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. The capital letters visible in the grid above are of course variable. Which letter belongs where



will be determined as part of the solving process (and hence depends on the possible words in the db) 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` 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.

6. Consider the following definition for numerals:

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

- Define a predicate `addition` that takes two numerals in the notation that we just introduced as arguments and performs the addition.
- Define a predicate `greater_than` that takes two numerals as arguments and decides whether the first one is greater than the second one.
  E.g. `greater_than(succ(succ(succ(0))),succ(0)).` results in yes

7. Binary trees are trees where all internal nodes have exactly two children. The smallest binary trees consist of only one leaf node. We will represent leaf nodes as `leaf(Label)`. For instance, `leaf(3)` and `leaf(7)` are leaf nodes, and therefore small binary trees.
   Given two binary trees B1 and B2 we can combine them into one binary tree using the predicate tree: `tree(B1,B2)`. So, from the leaves `leaf(1)` and `leaf(2)` we can build the binary tree `tree(leaf(1), leaf(2))`. And from the binary trees `tree(leaf(1), leaf(2))` and `leaf(4)` we can build the binary tree `tree(tree(leaf(1), leaf(2)), leaf(4))`.
   Now, define a predicate `swap`, which produces a mirror image of the binary tree that is its first argument.
   For example:

```
?- swap(tree(tree(leaf(1), leaf(2)), leaf(4)),T).
T = tree(leaf(4), tree(leaf(2), leaf(1))).
yes
```