Prolog

**Reading Material 1**

Read and make notes of the tutorial in <http://www.doc.gold.ac.uk/~mas02gw/prolog_tutorial/prologpages/> from simple facts to recursion. Also read and make notes of the from the document “Prolog\_extra.pdf”

* Pay specific attention to the logic behind predicates and unification, this will be the heart of everything we do.
* Use goolge or imdb to create a knowledge base in Prolog, (you will be consulting to see what actors star in which movies). That knowledge base should have at least the following:
  + - 1 movies with "Chuck Norris"
    - 2 movies with "Jackie Chan"
    - 2 movies with "Bruce Lee"
    - 1 movie with "Chuck Norris" and "Bruce Lee"

Query your knowledge base to know if you created it correctly.

A nice example is provided in this blog: <http://vadimcomanescu.net/2014/02/25/having-some-fun-with-prolog/>

**During class**

Research the following concepts, use at least 2 formal sources.

* What is first order logic
* Family example.
* What is the definition of logic programming?
* What are the characteristics of a language that uses logic programming paradigm?
* Name two advantages logic programming has over other paradigms
* Name two disadvantages that logic programming has compared with imperative, object oriented programing.
* Could this type of programming be done in parallel?
* What are the basic elements of logic programming languages?
* Family tree example or movie sample. **(family.pl)**
* Trace on Backtracking

**Lab 1**

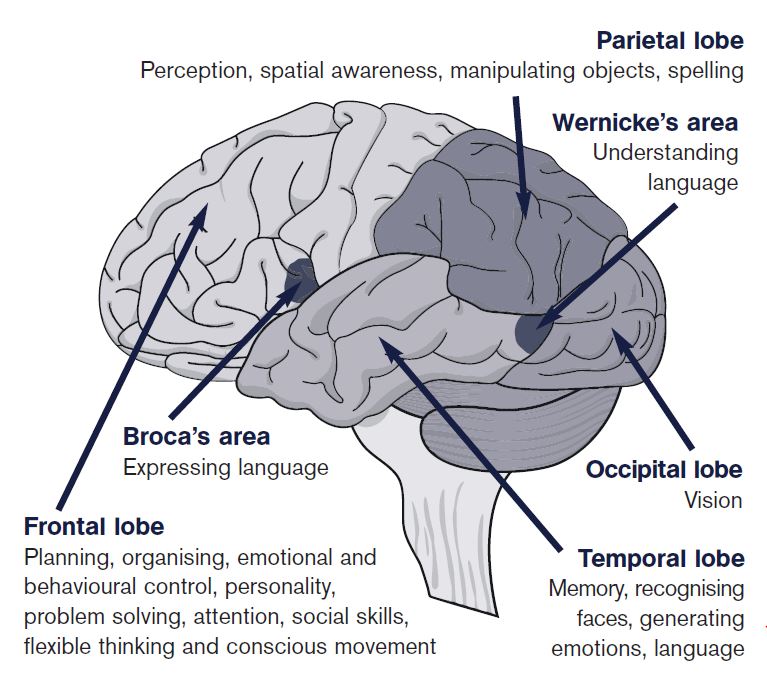
After doing this lab you should be able to understand how to represent facts and graphs in a logic programming language. You should also have a grasp of how Prolog’s basic operations work.

* Given the following facts

hobby(juan,kaggle).   
hobby(luis,hack).   
hobby(elena,tennis).   
hobby(midori,videogame).   
hobby(simon,sail).  
hobby(simon,kaggle).  
hobby(laura,hack).  
hobby(hans,videogame).   
  
Define a predicate “compatible(X,Y)”. We say that X and Y are compatible if they share at least 1 hobby.

* Use the following map to generate a graph of the regions of the brain, assume that the regions are connected only if you can see the in the image next to each other, assume they are one sided directional conections.

For example:

connect(brocas\_area,frontal\_lobe).

Define the predicate can\_get\_to(X, Y) which is true if there is a path (in the brain) that can get to X from Y.

e.g. can\_get\_to(brocas\_area, frontal\_lobe) is true.

Define the predicate size(X, Y, Z) which returns in Z the size of the path from X to Y.

* Define the predicate min(A, B, C, Z), which returns Z as the minimal value between A, B, and C.
* Define the predicate gcd(A, B, Z), which returns Z as the greatest common divisor (or highest common factor) of A and B.

**Reading Material 2**

Read and make notes from the following links [Lists](http://www.doc.gold.ac.uk/~mas02gw/prolog_tutorial/prologpages/lists.html) and [List Processing](http://kti.ms.mff.cuni.cz/%7Ebartak/prolog/lists.html).

* Are there any advantages to using recursion instead of iteration?

**During class**

* Compare a cycle between a procedural, functional and a logic paradigm. **(recusion\_example.pl, fibbo.pl)**
* Do example with head recursion and tail recursion **(make list.pl)**
* Understanding prolog compiler <http://www.swi-prolog.org/pldoc/man?section=compilation>
* Generate a compiled version of your program in prolog <http://www.swi-prolog.org/FAQ/MakeExecutable.html>

**Lab 2**

After this lab you should have a deeper understanding of recursion, and you should know how lists work in Prolog.

* Define the following predicates

Any positive.

?- any\_positive ([-1,-2,3,-4]).  
true.

Substitute.

?- substitute (2, 3, [1, 2, 2, 2, 3, 2], X).  
X = [1, 3, 3, 3, 3, 3]

Eiminate duplicates.

?- eliminate\_duplicates ([a, a, a, a, b, c, c, a, a, d, e, e, e, e],X).  
X = [a, b, c, d, e]

Intersect.

?- intersect ([a, b, c, d], [b, d, e, f], X).  
X = [b, d]

Invert.

?- invert([a, b, e, c, e],X).  
X = [e, c, e, b, a]

Less than.

?- less\_then(5, [1, 6, 5, 2, 7], X).  
X = [1, 2]

More than (or equal).

?- more\_than((5, [1, 6, 5, 2, 7], X).  
X = [6, 5, 7]

quicksort.

?- quicksort([1, 6, 5, 2, 7], X).  
X = [1, 2, 5, 6, 7]

rotate.

?- rotate([1, 6, 5, 2, 7], 3, X).  
X = [2, 7, 1, 6, 5]  
?- rotate([1, 6, 5, 2, 7], -3, X).  
X = [5, 2, 7, 1, 6]

Given the brain Knowledge base you used in the last lab, modify it so that the relations between part are bidirectional. Define the predicate size(X, Y, Z) which returns in Z the shortest path between X and Y.

**Reading Material 3**

Read and make notes from <http://cs.union.edu/~striegnk/learn-prolog-now/html/node88.html#sec.l10.cut> and the prolog\_cutfail.pdf, it is a compact presentation.

* In what cases should we use cut?
* What is the purpose of fail?

**During class**

* First depth search algorithm.

**Lab 3**

After this lab you should understand in what situations it is necessary to stop prologs backtracking and how to do so.

This is an implemented depth fist search algorithm in prolog:

<https://www.cs.unm.edu/~luger/ai-final/code/PROLOG.depth.html>

Read it carefully and use it as an example to implement your own **Limited Depth First Search Algorithm**

**Ideas for project:**

<https://www.cs.auckland.ac.nz/~j-hamer/07.363/explore.html>

Kevin Bacon center of the Universe.