# Interpreting Algorithms Lists in Prolog

Remember that if List is not empty, Prolog admits that it has a structure [Head|Tail]. Head can be specified with a variable or a couple of variables, Tail must be a list.

For example, in

: test([a,b,c,d]).

: test([X,Y|Z])?

X = a

Y = b

Z = [c, d]

The query specifies Head with a couple of variables, but another query

test([X|Tail])?

specifies Head with one variable

**Interpreting How to Find Maximum**  **in the list.**

The program below has 4 lines. The first two lines define max between two numbers. The last two lines define maxlist, which is a maximal element of the list.

max(A,B,A) :- A>B;A=B.

max(A,B,B) :- B>A.

maxlist([X], X).

maxlist([X,Y|Rest], A) :- maxlist([Y|Rest], B), max(X,B,A).

maxlist([6,8,5,4,87,43], X)?

X = 87

**Program Interpretation**

**max(A,B,A) :- A>B;A=B.**

means that if A>B or A=B, A is max among A,B

**max(A,B,B) :- B>A**.

means that if B>A, B is max among A,B

**maxlist([X], X).**

means that in the list with a single element, this element is maximum in the list

**maxlist([X,Y|Rest], A) :- maxlist([Y|Rest], B), max(X,B,A).**

is applied when a list has at least two elements. Suppose we found maximum B in the sublist that can be obtained from the original list by deleting the first element. Now, compare B with X which is the first element of the original list. The max among B and X will be maximum in the original list we consider. Phrasing it differently, A is max in the original list specified as [X,Y|Rest] if there exists B such that B is maximum in the list [Y|Rest] and A is max among X and B. In other words, in the list

6,8,1,45,3

45 is maximum because it is maximum in the sublist

8,1,45,3

And it is greater than 6

Therefore, the **idea behind recursion** is that the goal of finding maximum in [6,8,1,45,3] is reduced to the goal of finding maximum in [8,1,45,3] and comparing it with 6. The goal of finding maximum in [8,1,45,3] is reduced to the goal of finding maximum in [1,45,3] and comparing it with 8. The goal of finding maximum in [1,45,3] is reduced to the goal of finding maximum in [45,3] and comparing it with 1. The goal of finding maximum in [45,3] is reduced to the goal of finding maximum in [3] and comparing it with 45. The goal of finding maximum in [3] can be solved with the first line of the rule: 3 is max is [3].

**Backtracking:**  3 is maximum in [3] and 45 >3, thus 45 is maximum in [45,3]. Since 45>1, 45 is maximum in [1,45,3]. Since 45>8, 45 is maximum in [8,1,45,3]. Since 45>6, 45 is maximum in [6,8,1,45,3].

**Interpreting How to Find the SUM of all elements** **in the list.**

The program below contains only two lines. It finds the sum of the all elements of the list.

: sumlist([],0).

: sumlist([Head|Tail], Sum) :- sumlist(Tail,S1), Sum is Head+S1.

: sumlist([9,8,7,6],X)?

X = 30

**sumlist([],0).**

Means that in the empty list the sum of all elements is 0

**sumlist([Head|Tail], Sum) :- sumlist(Tail,S1), Sum is Head+S1**

is applied when a list is not empty. Suppose we found the sum of all elements S1 in the sublist that can be obtained from the original list by deleting the first element. Now, take S1 and add the to the first element of the original list, we will get the sum of all elements in the original lest. Phrasing it differently, Sum is the sum of all elements in the original list specified as([Head|Tail] if there exists S1 such that S1 is the sum of all elements in Tail and Sum can obtained by adding S! to Head. . In other words, in order to find sumlist (the sum of all elements) in the list

[6,8,1,45,3]

we need to find sumlist in [8,1,45,3] and add it to 6. In order to find sumlist in [8,1,45,3], we need to find sumlist in [1,45,3] and add it to 8. In order to find sumlist in [1,45,3], we need to find sumlist in [45,3] and add it to 1. In order to find sumlist in [45,3], we need to find sumlist in [3] and add it to 45. In order to find sumlist in [3], we need to find sumlist in [] and add it to 3. In order to find sumlist in [], we use the first line of the rule which tells us that it is 0.

**Backtracking:** 0 is sumlist of [0], 3 is sumlist of [3], 45+3 is sumlist of[45,3], 48+1 is sumlist of[1,45,3], 49+8 is sumlist of[8,1,45,3], 57+6 is sumlist of[6,81,45,3],