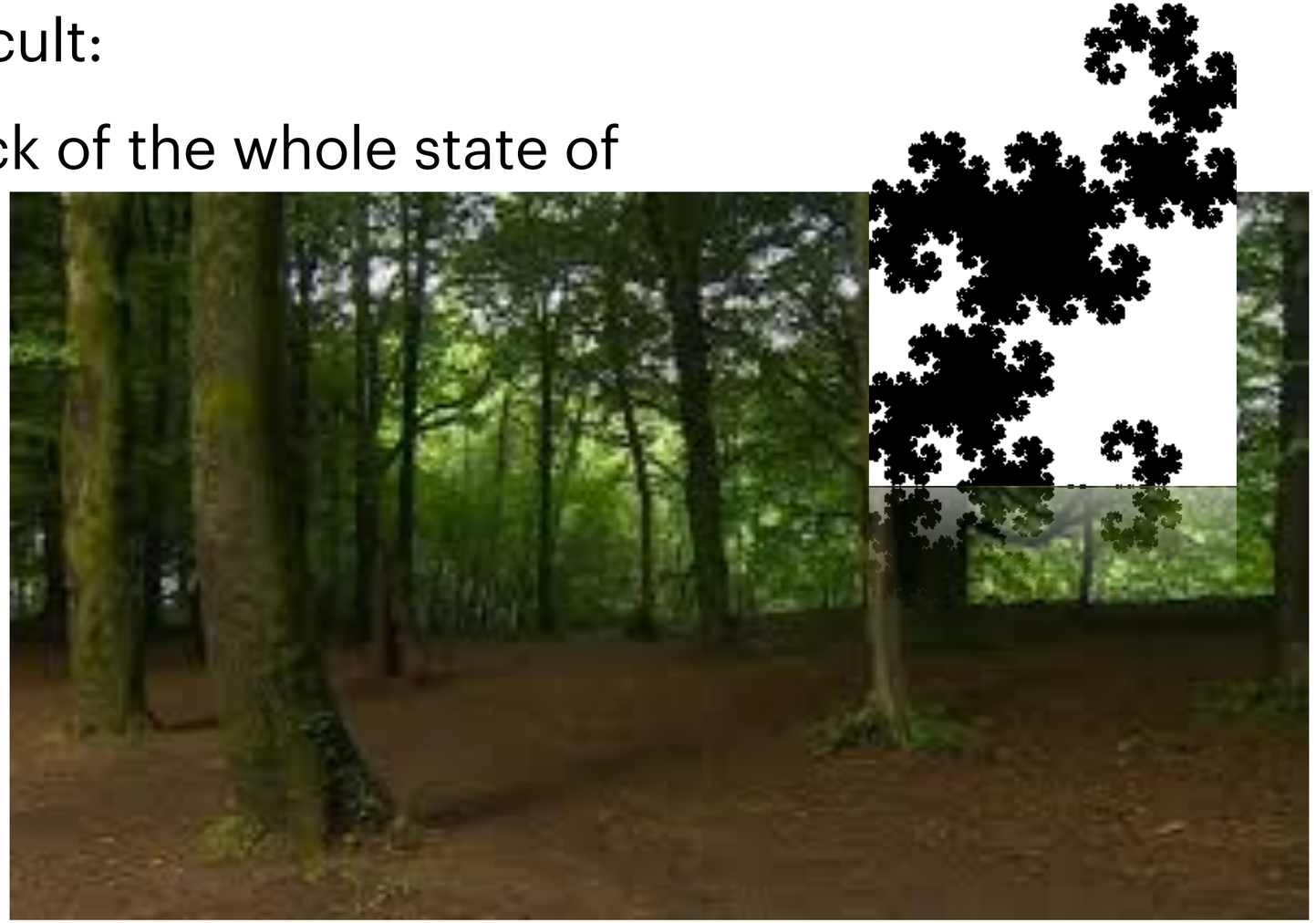
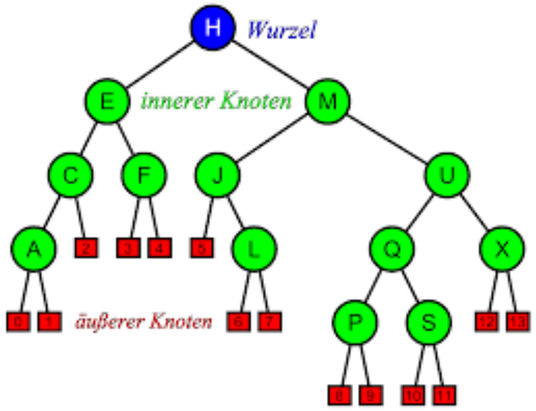
Programação em Lógica

Logic Programming

Programming

- Programming is difficult:
  - You must keep track of the whole state of
  - The world





Improving Programming


Talk to the Computer

We give the Problem

The computer gives the Solution

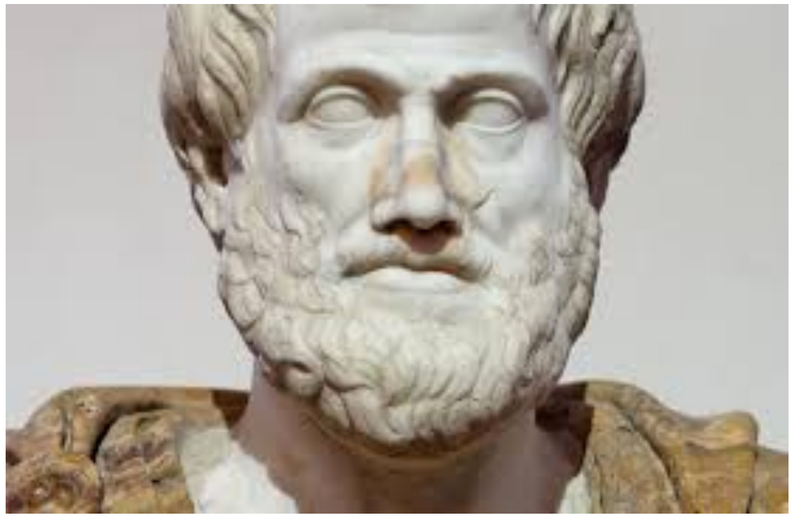
PROLOG

The Language

Prolog vs History of Logic

Aristotle, Alice, and Robinson

Logic is about the truth of statements

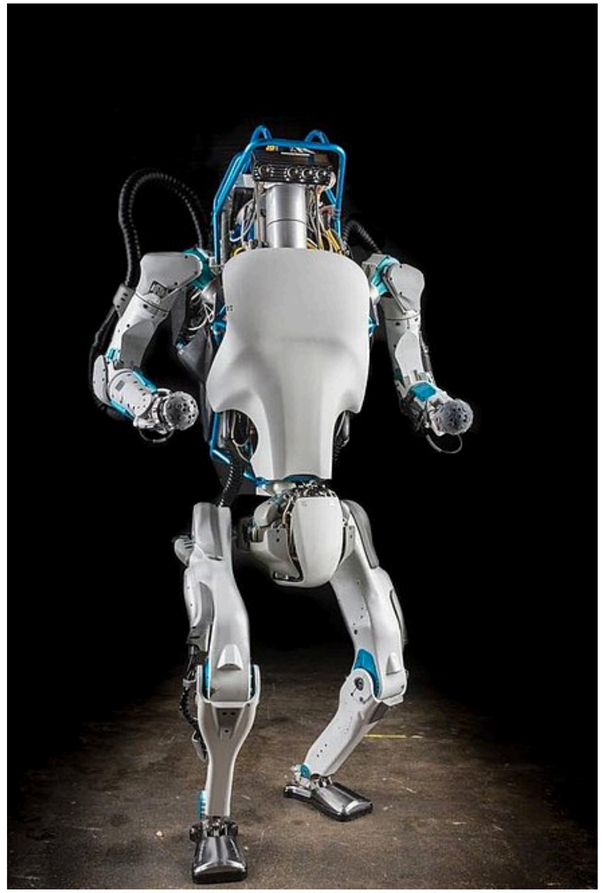


MP:

John is a robot

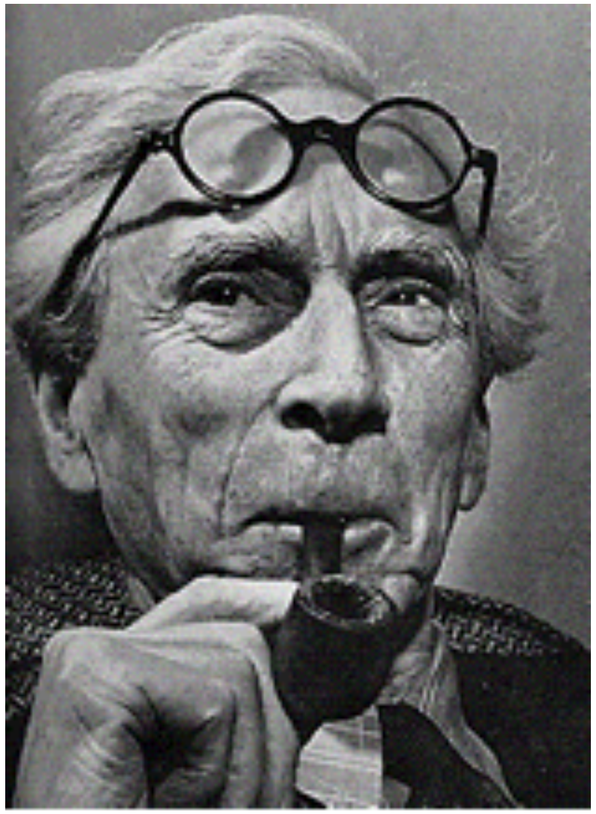
Robots eat oil

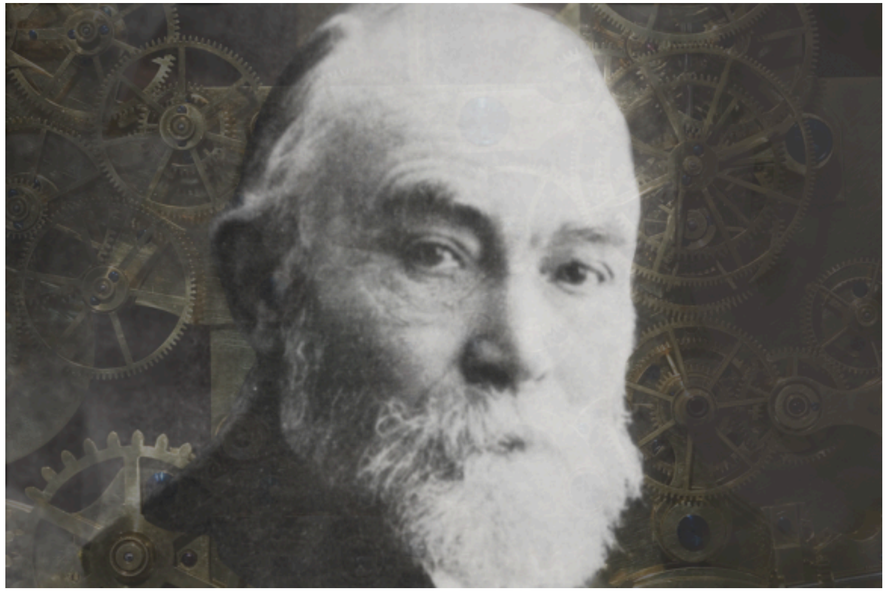
John eats oil.



Mathematical Logic

Prove-a-Proof ltd

- 



Predicate Calculus

Example	Frege Notation	Modern Notation
Everything is mortal	$\overset{\mathfrak{a}}{\smile} M(\mathfrak{a})$	$\forall x Mx$
Something is mortal	$\neg\overset{\mathfrak{a}}{\smile}\neg M(\mathfrak{a})$	$\neg\forall x\neg Mx$ i.e., $\exists x Mx$
Nothing is mortal	$\overset{\mathfrak{a}}{\smile}\neg M(\mathfrak{a})$	$\forall x\neg Mx$ i.e., $\neg\exists x Mx$
Every person is mortal	$\overset{\mathfrak{a}}{\smile}\begin{matrix} M(\mathfrak{a}) \\ \vdash P(\mathfrak{a}) \end{matrix}$	$\forall x(Px \rightarrow Mx)$
Some person is mortal	$\neg\overset{\mathfrak{a}}{\smile}\begin{matrix} M(\mathfrak{a}) \\ \vdash P(\mathfrak{a}) \end{matrix}$	$\neg\forall x(Px \rightarrow \neg Mx)$ i.e., $\exists x(Px \,\&\, Mx)$
No person is mortal	$\overset{\mathfrak{a}}{\smile}\begin{matrix} M(\mathfrak{a}) \\ \vdash P(\mathfrak{a}) \end{matrix}$	$\forall x(Px \rightarrow \neg Mx)$ i.e., $\neg\exists x(Px \,\&\, Mx)$
All and only persons are mortal	$\overset{\mathfrak{a}}{\smile} P(\mathfrak{a}) = M(\mathfrak{a})$	$\forall x(Px \equiv Mx)$

Resolution

McCarthy

Data Types in Prolog

Prolog Bricks

- Atoms: hello, j23_007b, +, /, ‘The hills are alive'
- Numbers 88, 0xFF, 3.3, 77777777777777777, 8//5
- Strings `With the Sound Of Music`

Prolog Syntax
Atoms

- Atoms
  - Start with Lower Caps [a-z]
  - Follow a number of [a-zA-Z0-9] and _
  - UTF-32: LL+LU,LT
  - Symbols (by themselves): ; , .
  - Symbols (group together): = > < / \
  - Quoted text ‘Most anything’

Syntax
Variables

- Start with upper case
  - Otherwise, similar to atoms
  - X, Y, Mr, Ms, Prof, Hello
  - Notice that syntax is different from std FOL
  - Variables
    - Do not need to be declared
    - Are local to a clause

Syntax
Strings

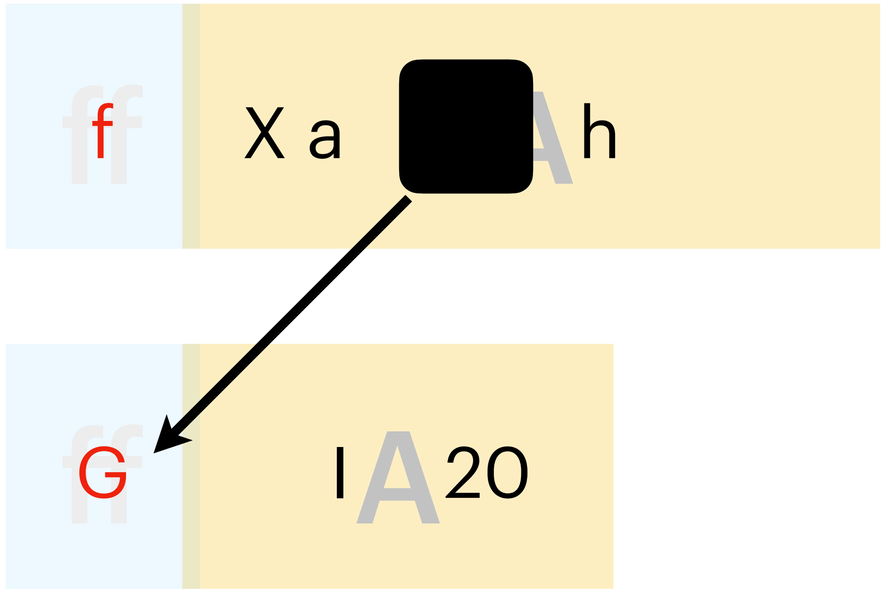
- Like traditional language strings:
- They are an atomic object
- But they are not permanent: they live in the stack

- `String is I

Compound Terms

Aka Complex or Applicative

- f(X,a,g(I,20),h)
  - Stump of a Tree,
  - Graph



Compound Terms: Tree

- folha(Val)
- no(folha(Val1), Val0, folha(Val2)
- nó(nó(folha(Val1), Val0, folha(Val2)),Val3, folha(4))
- nó(folha(4),Val3, nó(folha(Val1), Val0, folha(Val2)))
- nó(folha(4),Val3, nó(nó(folha(Val1), Val0, folha(Val2)), Val0, folha(Val2)))
-

Applications

- Also seen as a function call
  - $V = f(A_1, A_2, A_3)$
  - F is a generic function
  - Prolog should do the same operations over any node.

Functions

- Prolog comes from Logic:
  - There, we want to provea theorem independently of any interpretstion
- 2+3 = +(2,3)
- 5 = 2+3 fails
- To use arithmetic, we need a theory -> builtins
- 5 is 2+3 succeeds

Assembly -> C -> Java -> Haskell
Python

Prolog

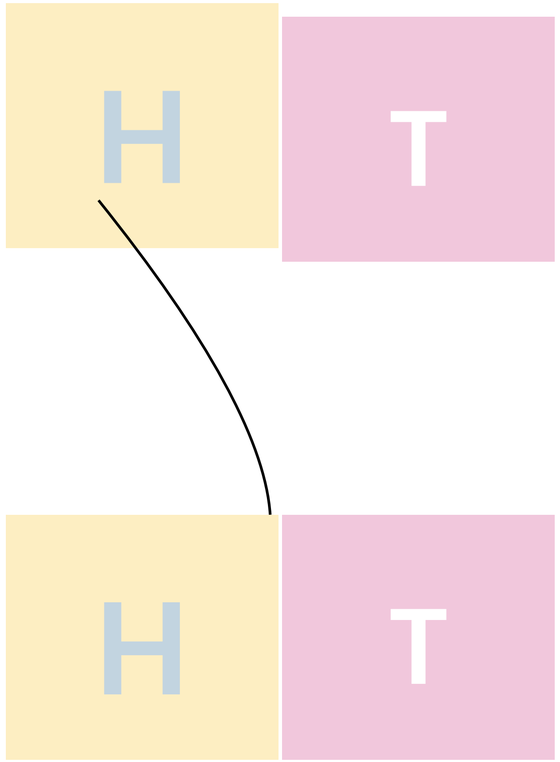
Provadores
Linguagem+
Solver

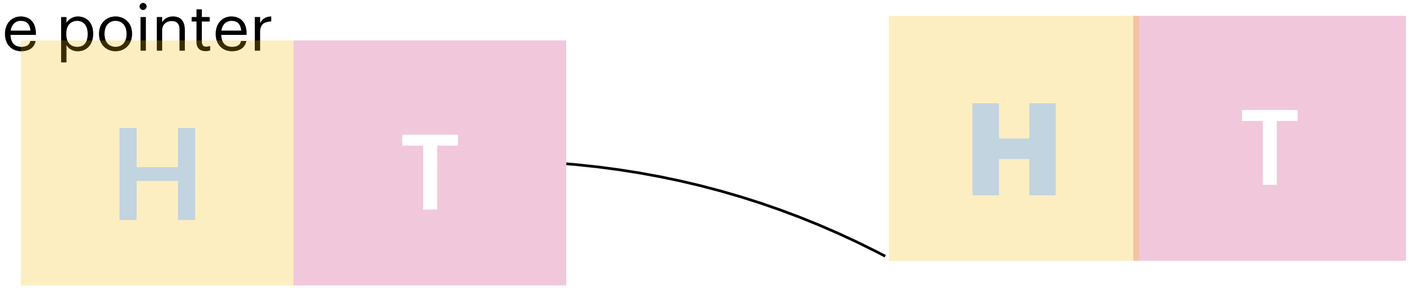
Lists

Pairs

- List = linked list

- True list:
  - Composed of pairs (Hi,Tl)
  - Usually such that:
    - H has the value
    - T has the pointer





Compilador

Programa -> Arvore Sintactica

Arvore Sintatica -> Codigo

Codigo -> Codigo Maquina

Lists

- Default list:
  - X=[a|Y], Y=[b|Z], Z = [c|W], W=[]
  - Aka
  - [a,b,c] , X=[a,b,c] Y =[b,c], Z = [c], W=[]

Lis

- Lista = [Head|Lista]

- Lista = []

Partial Lists

- What if we forgot the Z?
  - X=[a|Y], Y=[b|Z] or X =[a,b|Z]
  - This is called a **partial list**
  - In fact, 2 partial lists

Growing Lists

- append([X|L], Y,[X|NL]) :- append(L,Y,NL).
- append([], L, L).
- Similar to Python L1+L2 or L1 ## L2
- append([1,2],[3,4],L).

Append/3

The predicate that can do it

- ?- append([1,2,3],X,[1,2,3,4,5,6])
- ?- append(X,[5,6],[1,2,3,4,5,6])
- ?- append(X,[5,6],[5,6]).
- ?- append([_,A],_,L).
- ?- Middle of a list?

Sorting a List

Mergesort

- sort(L,SL) :- split(SL, L1, L2), sort(L1,S1), sort(L2,S2),merge(S1,S2,SL).
- sort([], []).

- split([H1,H2|L],[H1|L1],[H2|L2]) :- split(L,L1), split(L,L2).
- split([],[]).
- split([H],[H],[]).

- merge([],[],[]).
- merge([X|L],[],[X|L]).
- merge([],[X|L],[X|L]).
- merge([X1|L1],[X2|L2], NL) :- (X1 @=< X2 -> NL = [X1|NL1], merge(L1, [X2|L2], NL1).
- ; N2=[X2|NL2], merge([X1|L1], L2, NL2).
- ;

Mergesort
Example

- [1,5,3,2,6] ->[1,3,6]+ [5,2]
  - [1,3,6] -> [1,6] +[3]
    - [1,6]->[1]+[6]
    - [5,2] -> [5]+[2]
    - [1]++[6] -> [1,6]
    - [3] [1,6]++[3] => [1,3,6]
    - [5]++[2] -> [2,5]
      [1,3,6]+[2,5]=>[1,2,3,5,6]

Quicksort

In Prolog

- quicksort([A|L],S) :-
- split(A,LM,Lm),
- quicksort(LM, LOM),
- quicksort(Lm, Lom),
- append([Lom,[H|LOM],S).

Split

- split(A,[X|L],G,[X|S]) :- X@<A.
- split(A,[X|L],[X|G],S) :- X@>= A.

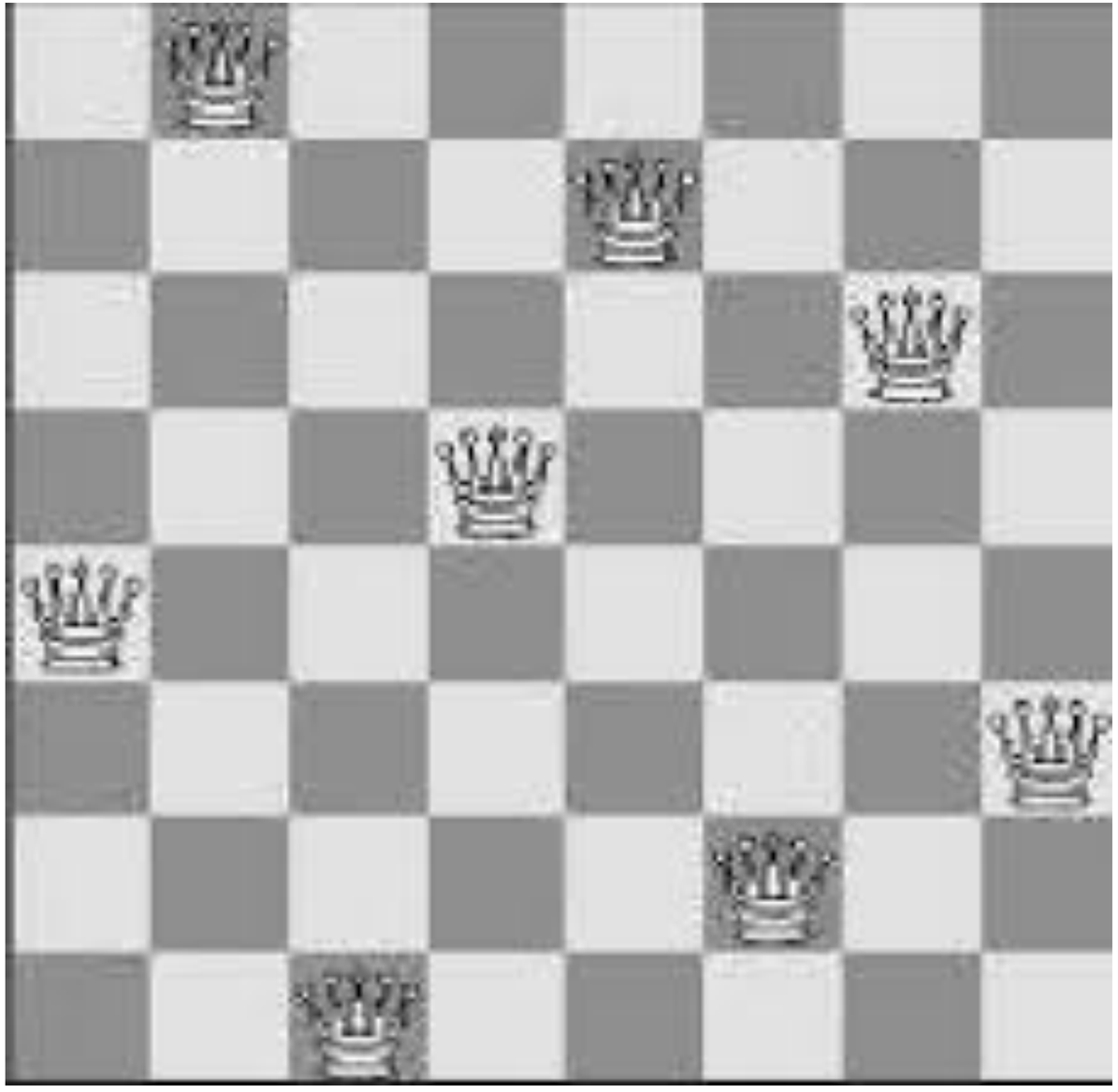
- Alguns truques:

- split(A,[X|L],G,[X|S]) :- X@<A, !.
- split(A,[X|L],[X|G],S).
-

Quicksort no append

- A better quicksort

```
quicksort([A|L] , S, S0):-
        split(A,LM,Lm),
         quicksort(LM, LOM, S0),
         quicksort(Lm, Lom, [A|LOM]),
```



Generate n Test

- Generate a solution
  - Each solution is a vector
  - A permutation of 1..N

- Test
  - Xi+Yi \== Xj+Yj
  - Xi-Yi \== Xj-Yj

Queens

- Instead of trying everything
- Try the Queen with more/less choices.
- How?
- queens(N,[[I,J]|Queens],Is,Js) :-
- findall(M-[I,J],score(I,J,N,M),MIJs),
- sort(Scores, Scored),
- member(M-[I,J],Scored),
- remove(I,J,Is,Js, NIs,NJs),
- queens(N,Queens, NIs,NJs).

A Scoring Function

A simple one

- Queens have the same line and cols
- But have diagonals of different length:
  - cost(I,J,C) :-
    - C is min(I,8-J)+min(8-I,J)
- What is the base case?

Builtin-Predicates

Interfacing the real world

- Library predicates:
  - X is .... -> arithmetic
  - X =:= Y -> arithmetic comp
  - X@=<Y -> term comparison
  - Write, write_term, format
  - Read, read_term
  - Hundreds more

Builtin Predicates

Tuning Prolog

- !
- first_queen(X,L) :- queens(X,L), !.

- But more:
- list(0,[]) :- !.
- list(N,[N|L]) :-
    - N1 is N-1,
    - list(N1,L).

Builtin Predicates

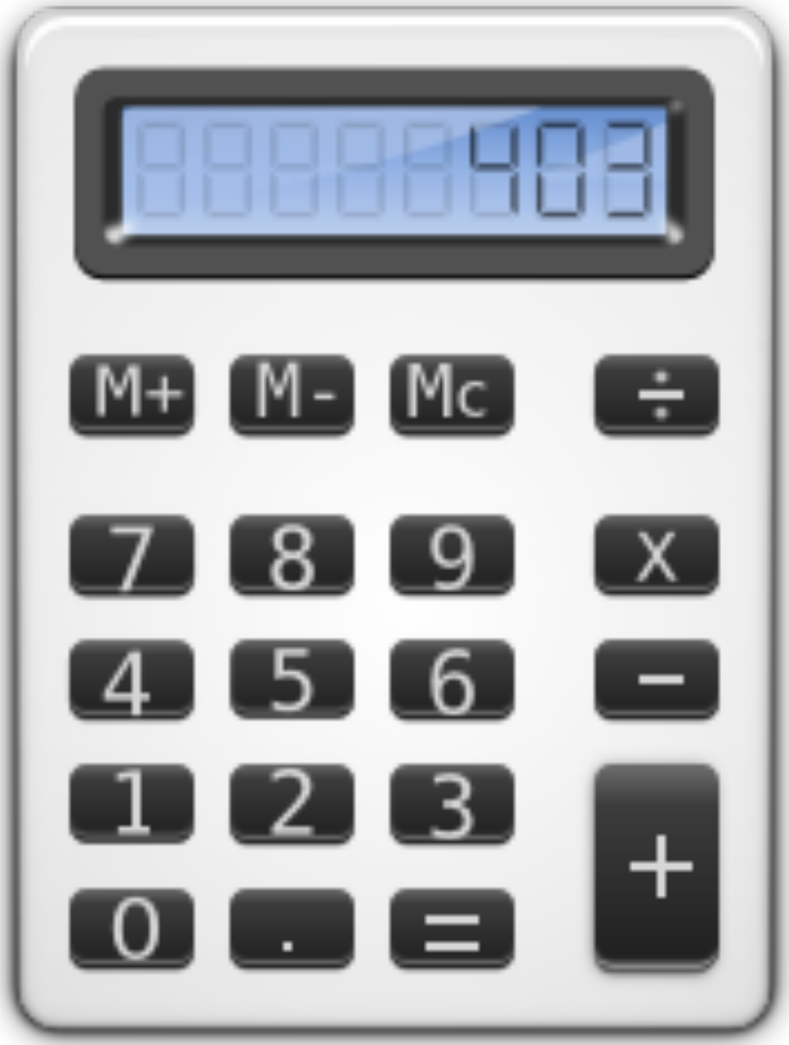
Collecting solutions

- findall(X,append(X,_,[1,2,3],Xs).

- findall([X,Y],append(X,Y,[1,2,3],XYs)

- findall(X,member(X,L),Xs).

Calculator



How To Implement?

- Scanner: chops into blocks
- Parser: creates a tree
- Evaluator: gets the result

Scanner