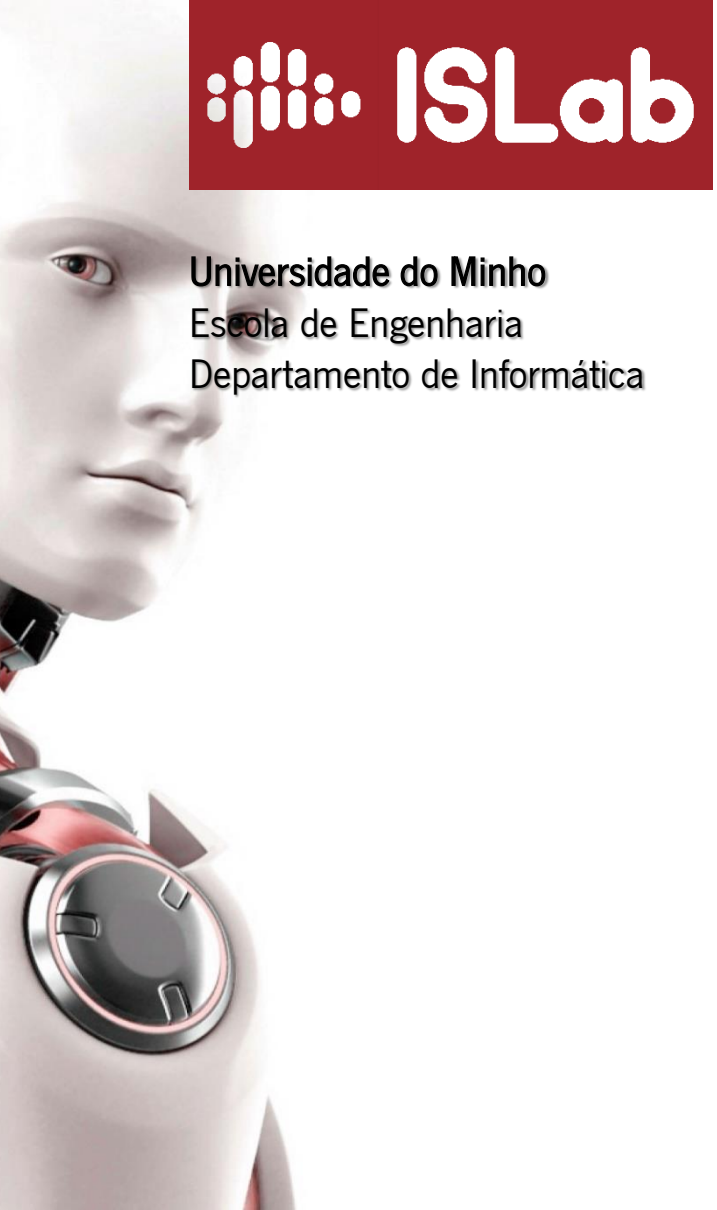


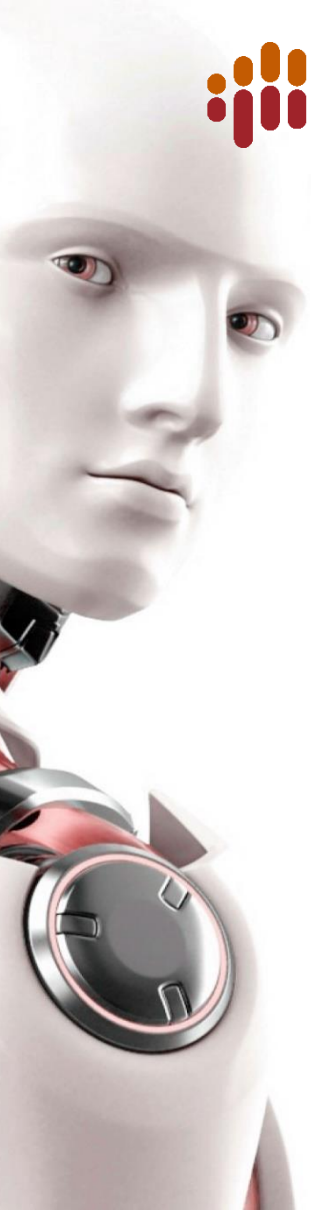
ISLab

Universidade do Minho
Escola de Engenharia
Departamento de Informática

Prolog

Aprofundamento

Mestrado Integrado em Engenharia Informática
Licenciatura em Engenharia Informática
Inteligência Artificial



ISLab
Synthetic Intelligence Lab

A closer look at terms

- Theory
  - Introduce the == predicate
  - Take a closer look at term structure
  - Introduce strings in Prolog
  - Introduce operators

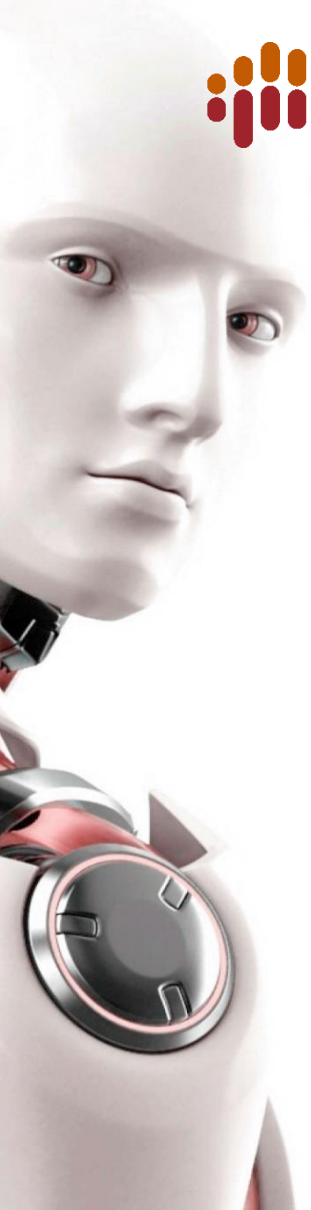
ISLab
Synthetic Intelligence Lab

Comparing terms: ==/2

- Prolog contains an important predicate for comparing terms:
  - This is the identity predicate

==/2

- The identity predicate ==/2 does not instantiate variables, that is, it behaves differently from =/2

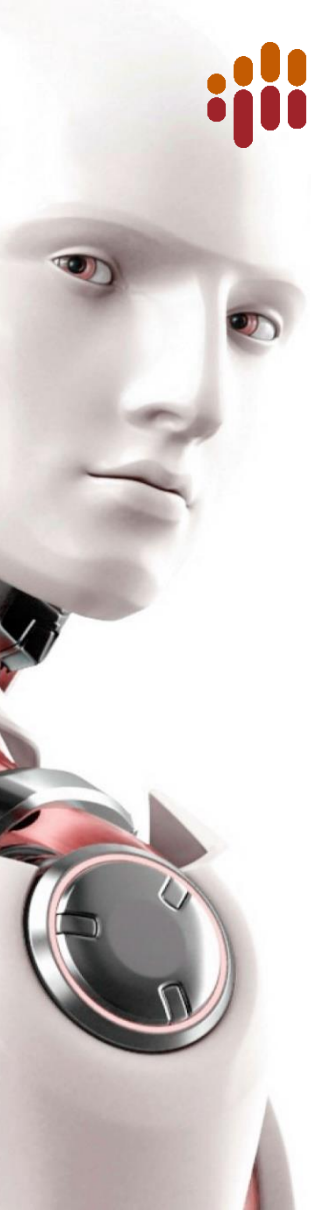


ISLab
Synthetic Intelligence Lab

Comparing terms: ==/2

- Prolog contains an important predicate for comparing terms
- This is the identity predicate ==/2
- The identity predicate ==/2 does not instantiate variables, that is, it behaves differently from =/2

```
?- a==a.
yes

?- a==b.
no

?- a=='a'.
yes

?- a==X.  X = _443
no
```

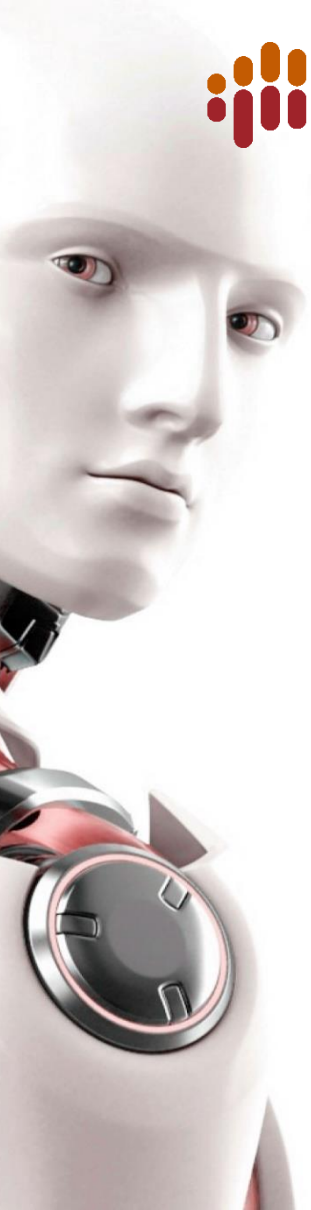


ISLab
Synthetic Intelligence Lab

Comparing variables

- Two different uninstantiated variables are not identical terms;
- Variables instantiated with a term T are identical to T

```
?- X==X.  X = _443
yes

?- Y==X.  Y = _442  X = _443
no

?- a=U, a==U.
U = _443
yes
```

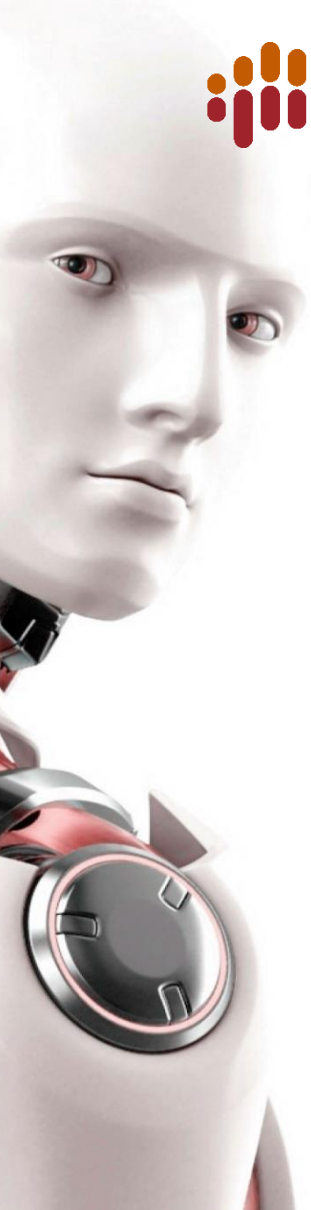


ISLab
Synthetic Intelligence Lab

Comparing terms: \==/2

- The predicate \==/2 is defined so that it succeeds in precisely those cases where
  - ==/2 fails
- In other words, it succeeds whenever two terms are not identical, and fails otherwise

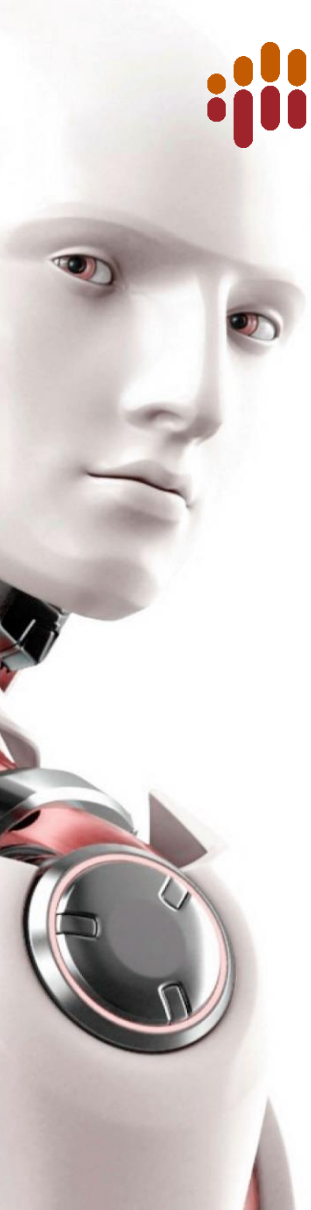
```
?- a \== a.  no
?- a \== b.  yes
?- a \== 'a'.  no
?- a \== X.  X = _443
yes
```



ISLab
Synthetic Intelligence Lab

Terms with a special notation

- For example: a and 'a', but there are many other cases
- Why does Prolog do this?
  - Because it makes programming more pleasant
  - More natural way of coding Prolog programs

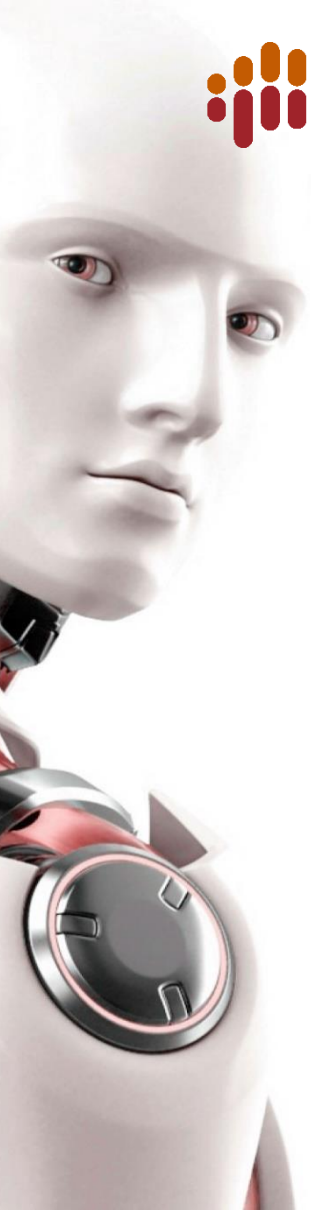


ISLab
Synthetic Intelligence Lab

Arithmetic terms

- Recall arithmetic:
- +, -, <, >, etc are functors and expressions such as 2+3 are actually ordinary complex terms;
- The term 2+3 is identical to the term +(2,3);

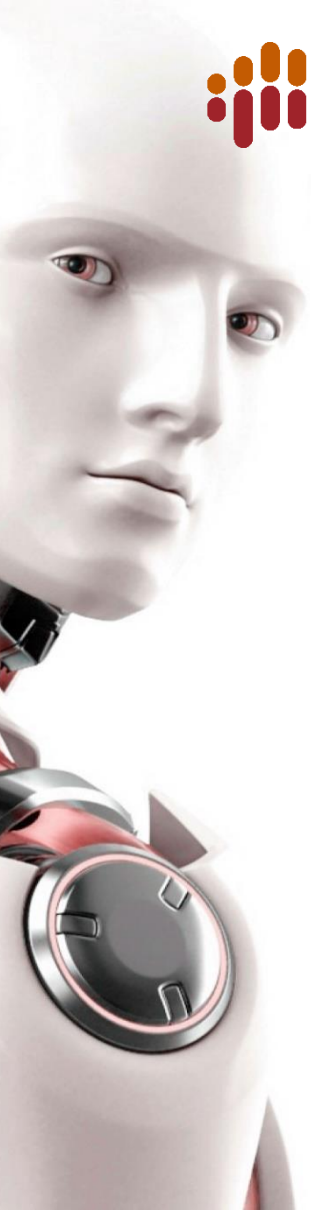
```
?- 2+3 == +(2,3).
yes

?- -(2,3) == 2-3.
yes

?- (4<2) == <(4,2).
yes
```



ISLab
Synthetic Intelligence Lab

Summary of comparison predicates

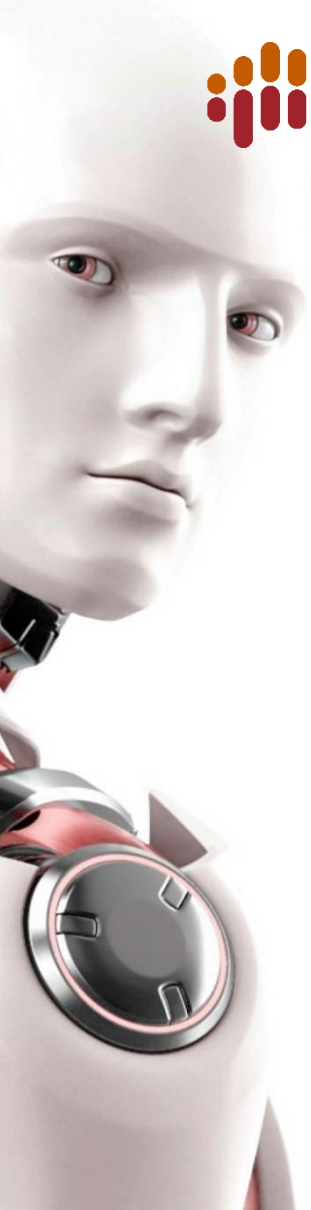
=	Unification predicate
\=	Negation of unification predicate
==	Identity predicate
\==	Negation of identity predicate
=:=	Arithmetic equality predicate
=\=	Negation of arithmetic equality predicate



ISLab
Synthetic Intelligence Lab

Checking the type of a term

atom/1	Is the argument an atom?
integer/1	... an integer?
float/1	... a floating point number?
number/1	... an integer or float?
atomic/1	... a constant?
var/1	... an uninstantiated variable?
nonvar/1	... an instantiated variable or another term that is not an uninstantiated variable



ISLab
Synthetic Intelligence Lab

Type checking: atom/1

?- atom(a).

yes

?- atom(7).

no

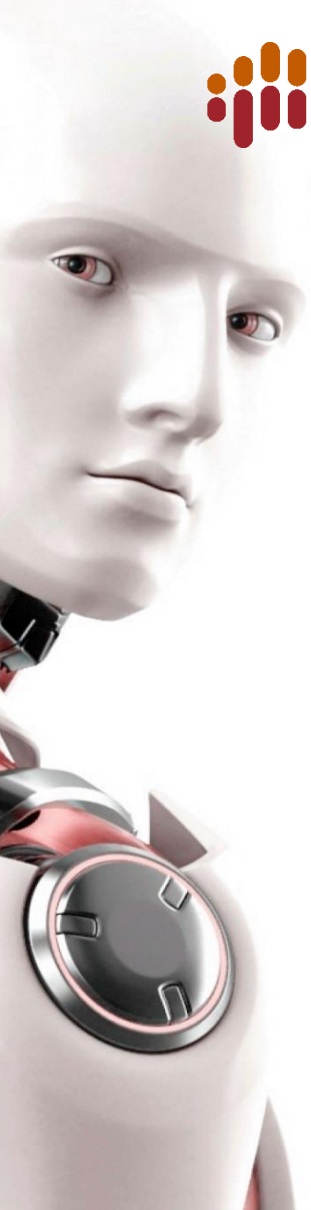
?- atom(X).

no

?- X=a, atom(X).

X = a yes

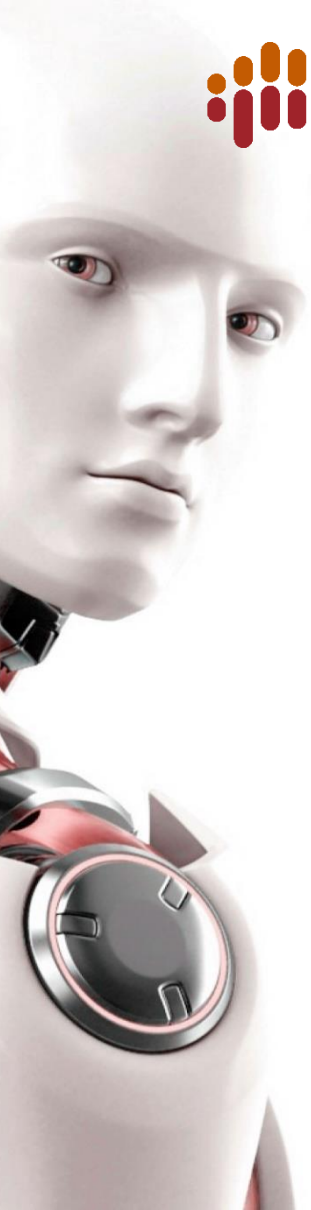
?- atom(X), X=a. no



ISLab
Synthetic Intelligence Lab

Type checking: atomic/1

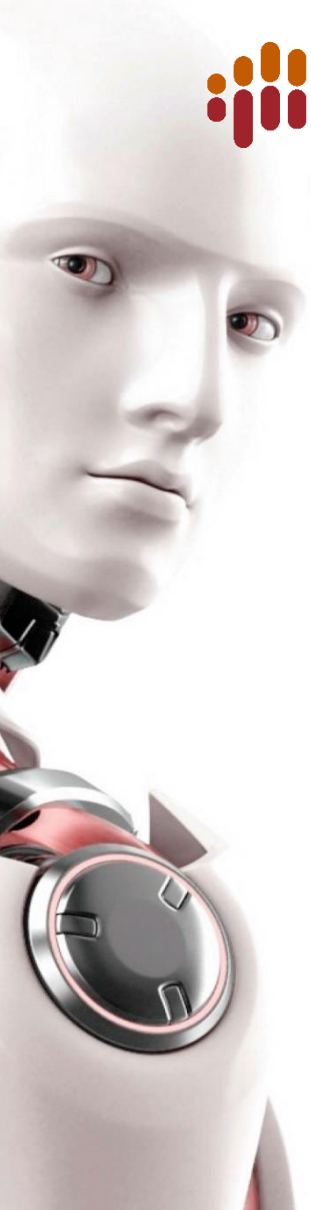
```
?- atomic(marcia).
yes

?- atomic(5).
yes

?- atomic(gosta(vicente,marcia)).
no
```



ISLab
Synthetic Intelligence Lab

Type checking: var/1

```
?- var(marcia).
no

?- var(X).
yes

?- X=5, var(X).
no
```



ISLab
Synthetic Intelligence Lab

Type checking: nonvar/1

?- nonvar(X).

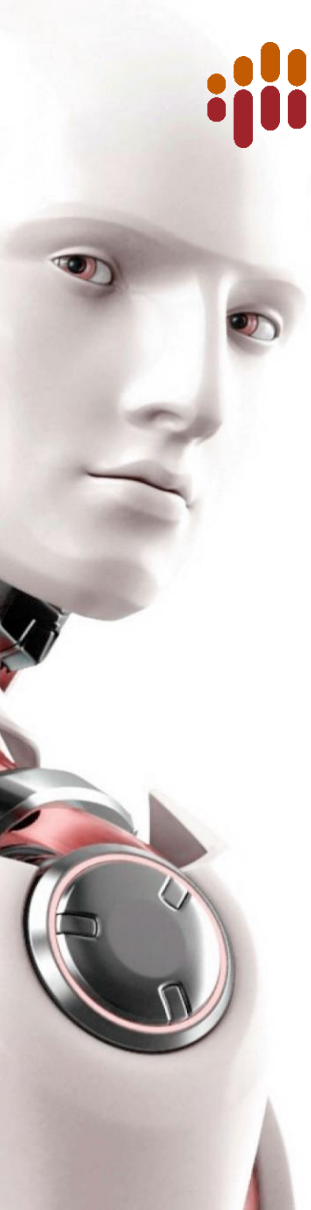
no

?- nonvar(marcia).

yes

?- nonvar(23).

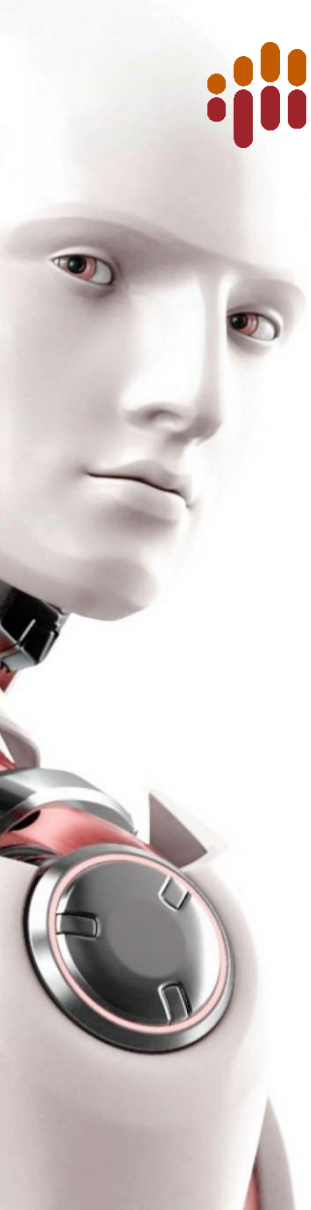
yes



ISLab
Synthetic Intelligence Lab

The structure of terms

- Given a complex term of unknown structure, what kind of information might we want to extract from it?
- Obviously:
  - The functor
  - The arity
  - The argument
- Prolog provides built-in predicates to produce this information.

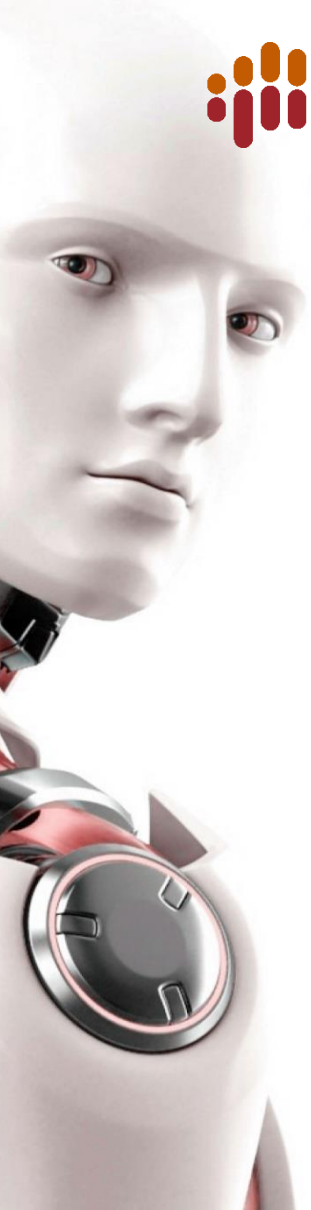


ISLab
Synthetic Intelligence Lab

The functor/3 predicate

- The functor/3 predicate gives the functor and arity of a complex predicate

```
?- functor(amigos(luisa,ana),F,A).
   F = amigos  A = 2
   yes
```

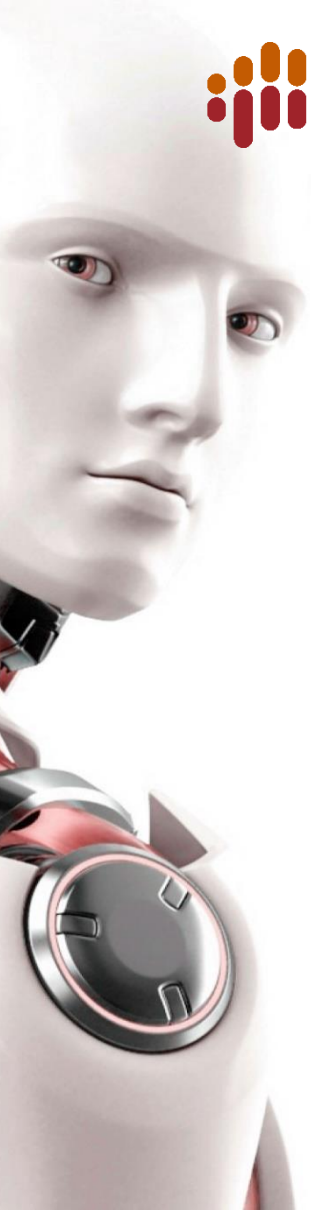


ISLab
Synthetic Intelligence Lab

functor/3 and constants

- What happens when we use functor/3 with constants?

```
?- functor(mia,F,A).
   F = mia  A = 0
   yes
?- functor(14,F,A).
   F = 14
   A = 0
   yes
```

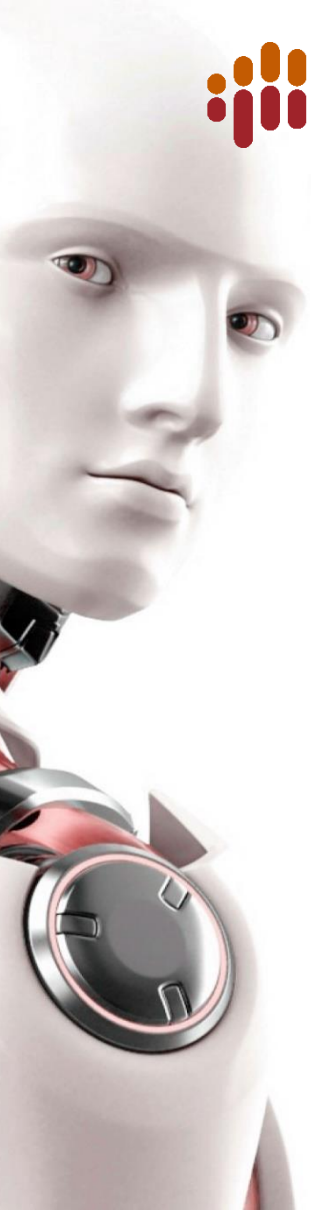


ISLab
Synthetic Intelligence Lab

functor/3 for constructing terms

- You can also use functor/3 to construct terms:

```
?- functor(Term,amigos,2).
   Term = amigos(_,_)
    yes
```



ISLab
Synthetic Intelligence Lab

Checking for complex terms

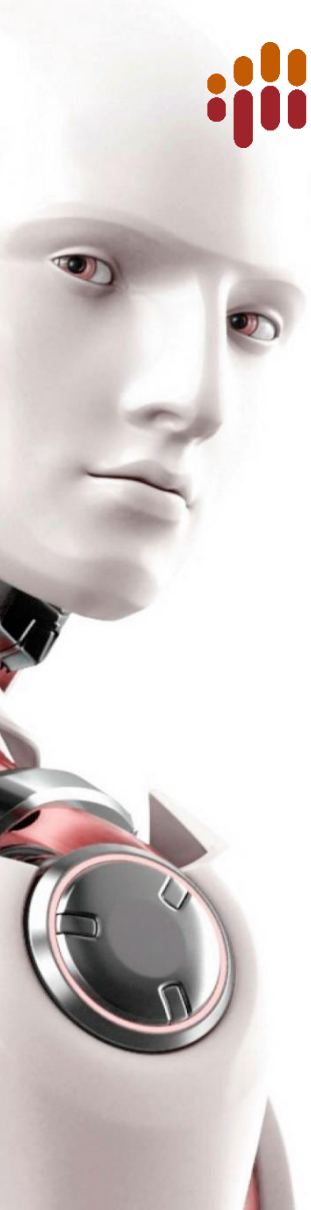
```
complexTerm(X):-
    nonvar(X),  functor(X,_,A),  A > 0.
```

ISLab
Synthetic Intelligence Lab

Arguments: arg/3

- Prolog also provides us with the predicate arg/3
- This predicate tells us about the arguments of complex terms
- It takes three arguments:
  - A number N
  - A complex term T
  - The Nth argument of T

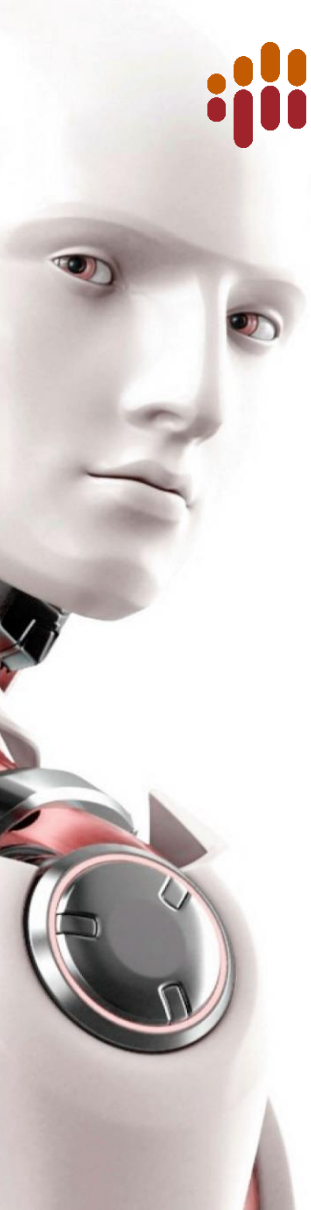
```
?- arg(2,gosta(luisa,ana),A).
A = ana
yes
```



ISLab
Synthetic Intelligence Lab

- Strings are represented in Prolog by a list of character codes;
- Prolog offers double quotes for an easy notation for strings.

```
?- atom_codes(maria,S).
S = [109,97,114,105,97]
yes
```

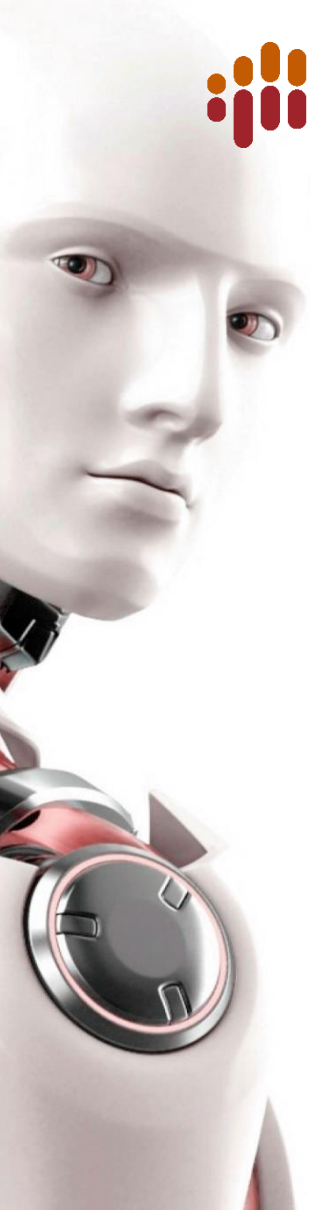


ISLab
Synthetic Intelligence Lab

Working with strings

- There are several standard predicates for working with strings
- A particular useful one is atom_codes/2

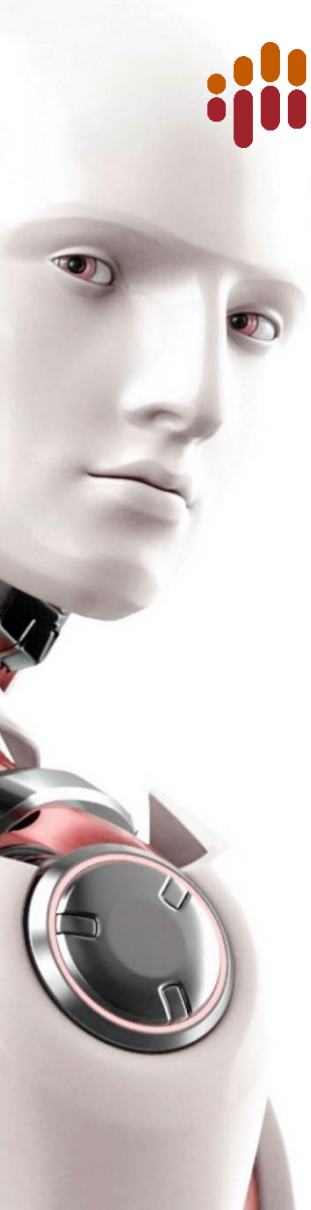
```
?- atom_codes(maria,S).
S = [109,97,114,105,97]
yes
```



ISLab
Synthetic Intelligence Lab

Properties of operators

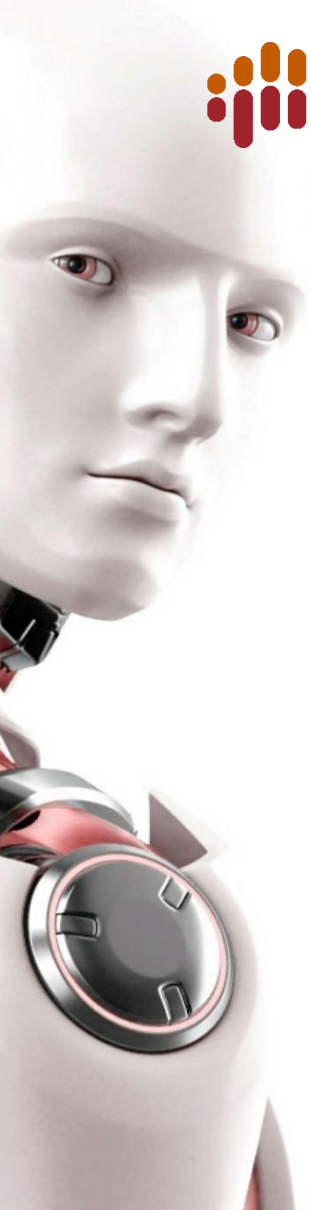
- Infix operators
  - Functors written between their arguments
  - – Examples: + - == , ; . –>
- Prefix operators
  - Functors written before their argument
  - Example: - (to represent negative numbers)
- Postfix operators
  - Functors written after their argument
  - Example: ++ in the C programming language



ISLab
Synthetic Intelligence Lab

Associativity

- Prolog uses associativity to disambiguate operators with the same precedence value;
- Example: 2+3+4
  - Does this mean (2+3)+4 or 2+(3+4)?
    - Left associative
    - Right associative
- Operators can also be defined as non- associative, in which case you are forced to use bracketing in ambiguous cases.



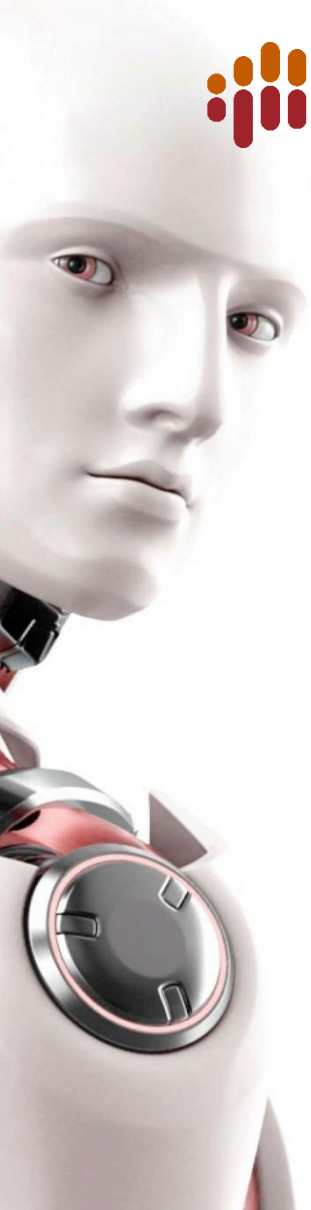
ISLab
Synthetic Intelligence Lab

Defining operators

- Prolog lets you define your own operators;
- Operator definitions look like this:

```
:- op(Precedence, Type, Name).
```

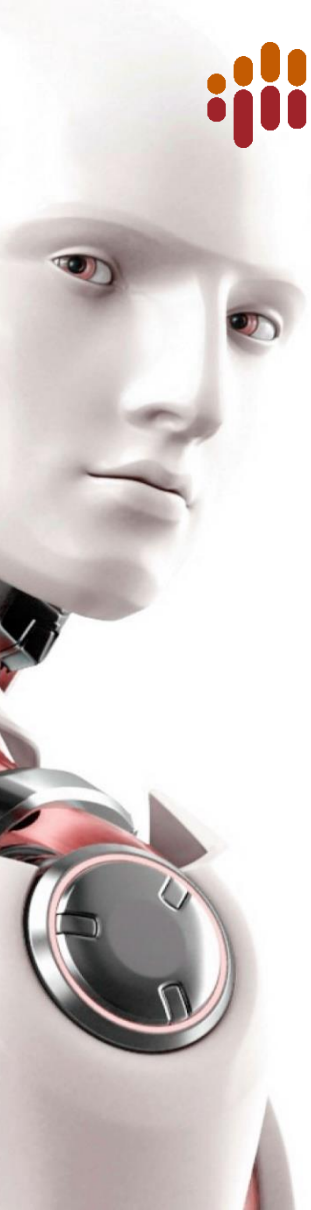
- Precedence:
  - number between 0 and 1200
- Type: the type of operator



ISLab
Synthetic Intelligence Lab

Types of operators in Prolog

yfx	left-associative, infix
xfy	right-associative, infix
xfx	non-associative, infix
fx	non-associative, prefix
fy	right-associative, prefix
xf	non-associative, postfix
yf	left-associative, postfix

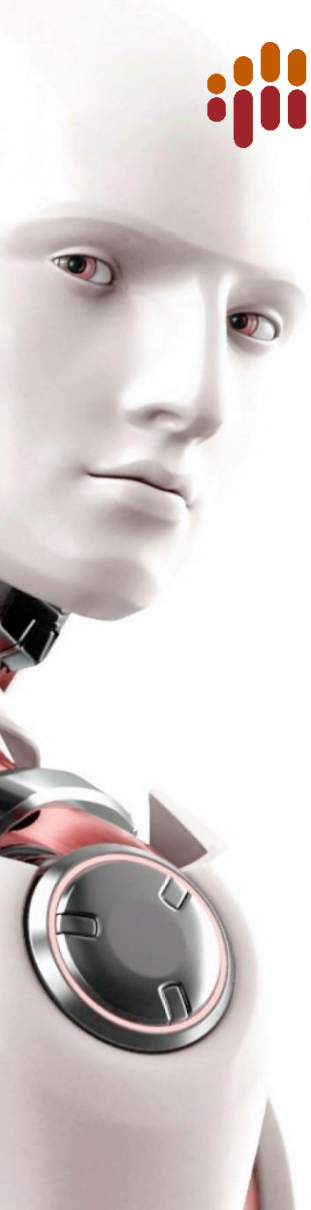


ISLab
Synthetic Intelligence Lab

Add/remove knowledge

```
:-dynamic figura/2.
figura(hexágono,6).
cria_figuras:- assertz(figura(triângulo,3)),
               asserta(figura(quadrado,4)),
               assertz(figura(pentagono,5)).
```

```
?- figura(F,NL).
F = hexágono ,
NL = 6

?- cria_figuras.
yes

?- figura(F,NL).
F = quadrado ,
NL = 4 ;

F = hexágono ,
NL = 6 ;

F = triângulo ,
NL = 3 ;
F = pentagono ,
NL = 5
```

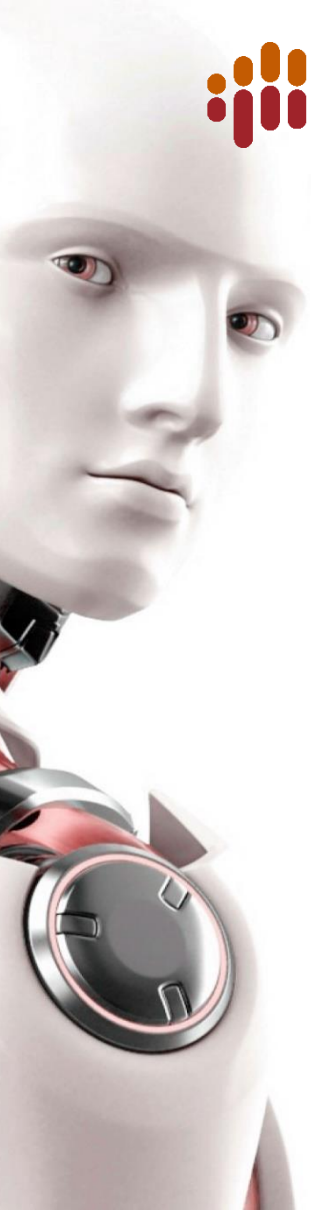


ISLab
Synthetic Intelligence Lab

Add/remove knowledge

```
:-dynamic figura/2.
figura(hexágono,6).
cria_figuras:- assertz(figura(triângulo,3)),
               asserta(figura(quadrado,4)),
               assertz(figura(pentagono,5)).
```

```
?- retract(figura(triângulo,X)).
X = 3

?- retract(figura(X,Y)).
 X = quadrado

Y = 4
```



ISLab
Synthetic Intelligence Lab

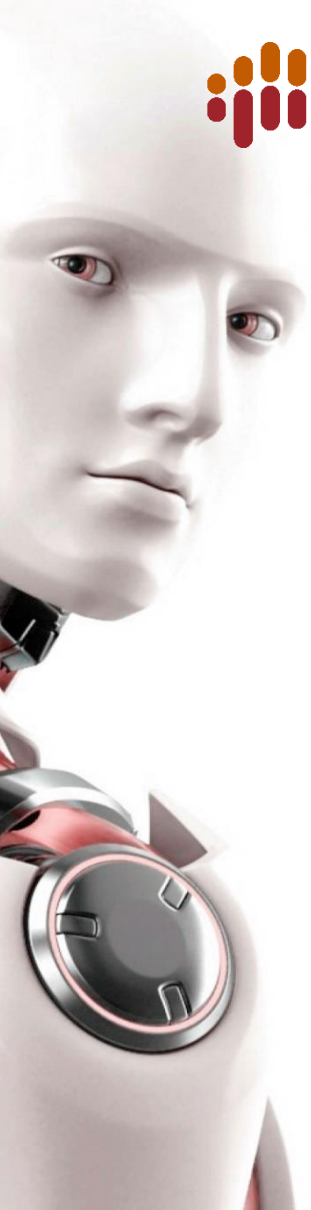
Database Manipulation

- Prolog has five basic database manipulation commands:
  - assert/1
  - asserta/1
  - assertz/1

  } Adding information

  - retract/1
  - retractall/1

  } Removing information

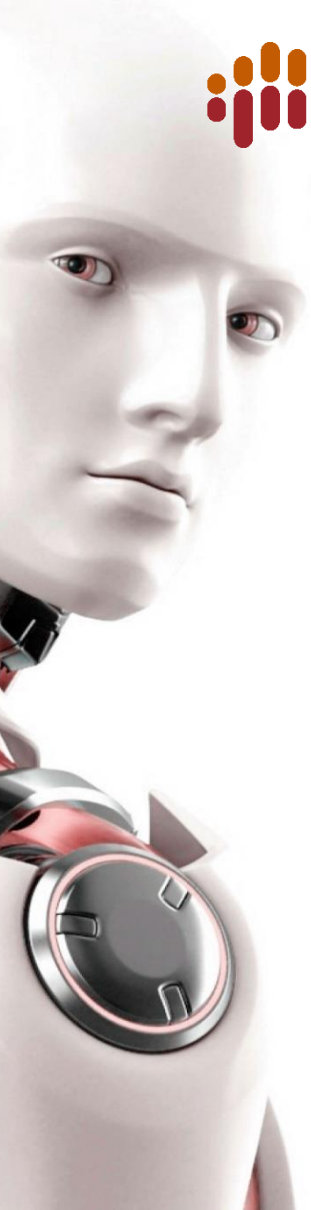


ISLab
Synthetic Intelligence Lab

Consider this database

```
filho(marta,carla).
filho(carla,carolina).
filho(carolina,laura).
filho(laura,rosa).

descendant(X,Y):- filho(X,Y).
descendant(X,Y):- filho(X,Z),
                  descendant(Z,Y).
```

```
?- descendant(marta,X).
X=carla;
X=carolina;
X=laura;
X=rosa;
no
```

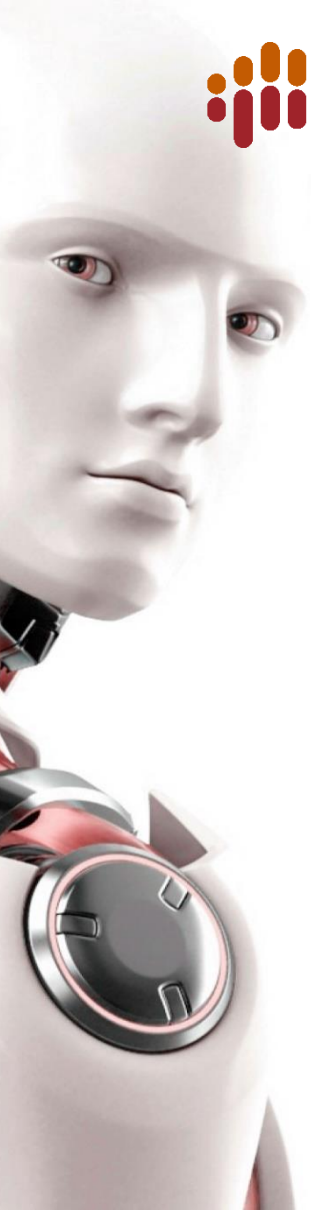


ISLab
Synthetic Intelligence Lab

Consider this database

```
filho(marta,carla).
filho(carla,carolina).
filho(carolina,laura).
filho(laura,rosa).

descendant(X,Y):- filho(X,Y).
descendant(X,Y):- filho(X,Z),
descendant(Z,Y).
```

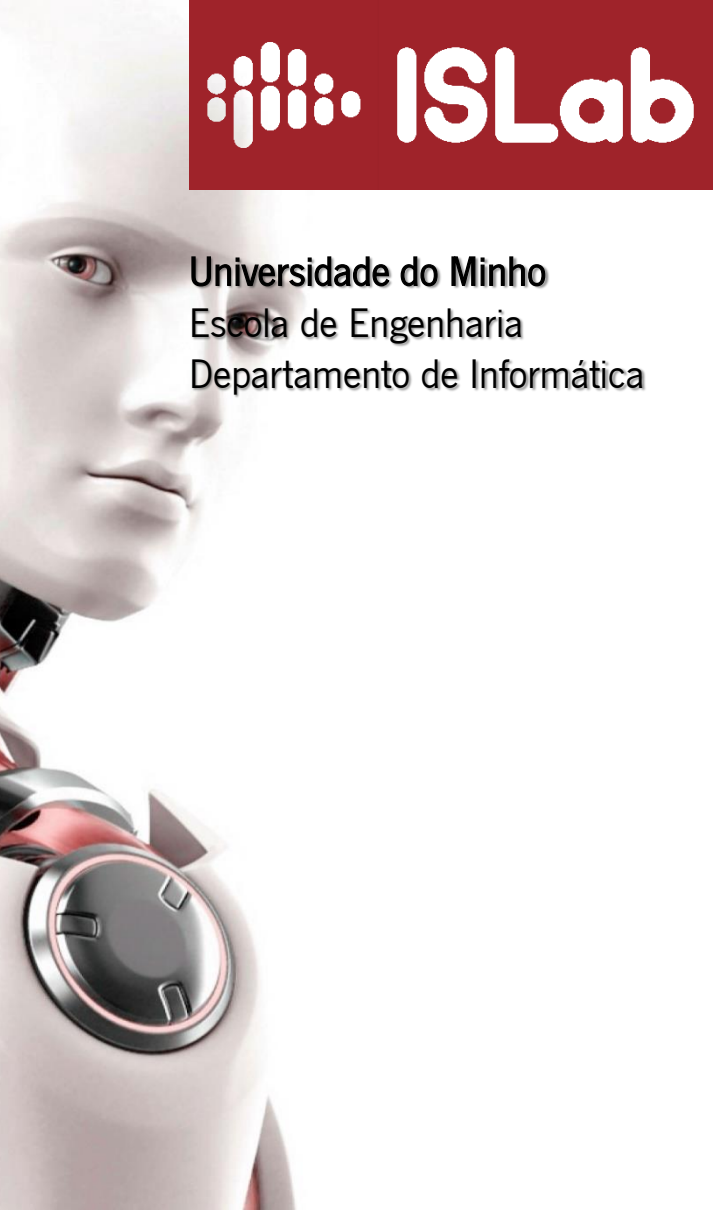
```
?- findall(X,descendant(marta,X),L).
L=[carla,carolina,laura,rosa]
 yes
```



ISLab
Synthetic Intelligence Lab

Database Manipulation

- bagof(X,Q,L) – L is a list of X that attend Q, if no solution, it fails.
- setof(X,Q,L) – L is the list of X that attend Q, L is ordered, repeated elements removed, if no solution, it fails.
- findall(X,Q,L) – L is the list of X that attend Q, if no solution findall succeeds with L=[]



ISLab

Universidade do Minho
Escola de Engenharia
Departamento de Informática

Prolog

Aprofundamento

Mestrado Integrado em Engenharia Informática
Licenciatura em Engenharia Informática
Inteligência Artificial