

# Inteligência Artificial

Programação em Lógica I

Licenciatura em Engenharia Informática  
2024/2025

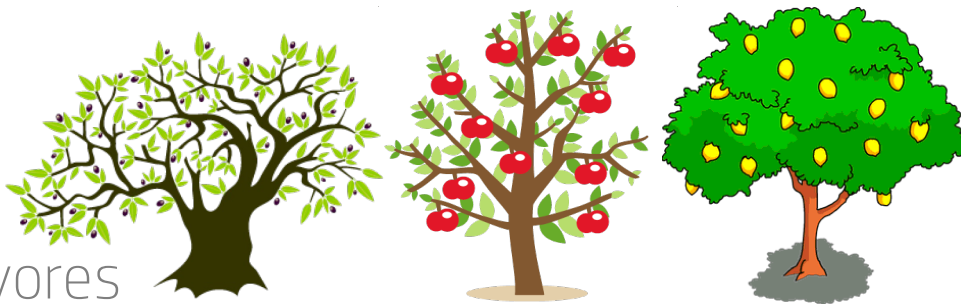
## \_introdução

- A Inteligência Artificial procura simular/implementar *comportamentos inteligentes*
- Uma das “escolas” da IA diz que o comportamento inteligente é resultado de processos de raciocínio corretos sobre o conhecimento disponível
- Tanto o conhecimento como os processos de raciocínio podem ser modelados numa linguagem formal lógica
- O formalismo lógico mais simples é a lógica proposicional
  - Linguagem formal
  - Métodos de inferência

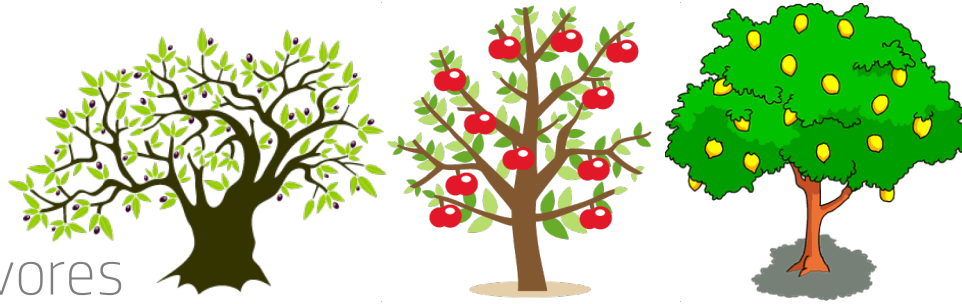
## \_introdução

- Muitas vezes, quando não prestamos atenção, é fácil aceitar conclusões inválidas
- Qual dos argumentos a seguir é válido?
  - Se chove, o chão fica molhado. Está a chover. Logo, o chão está molhado
  - Se chove, o chão fica molhado. O chão está molhado. Logo, choveu

Toda a fruta cresce em árvores



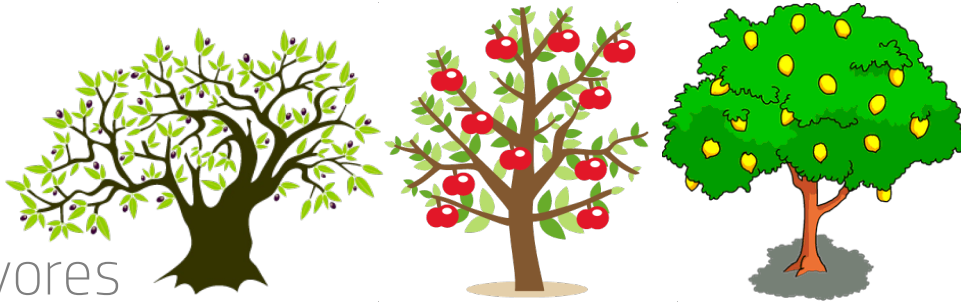
Toda a fruta cresce em árvores



A laranja é uma fruta



Toda a fruta cresce em árvores

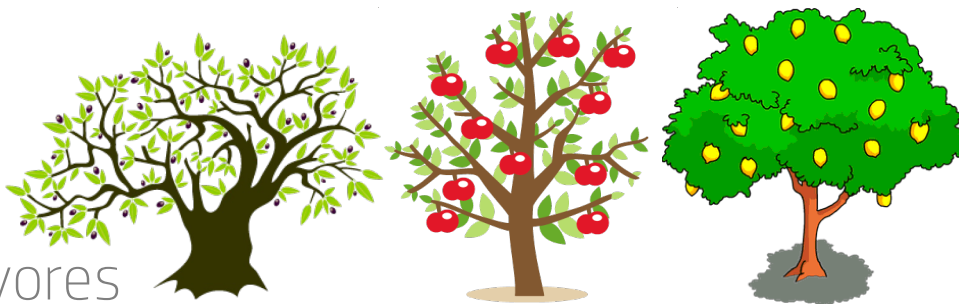


A laranja é uma fruta



A laranja cresce em árvores

# Dedução



(Premissa maior) Toda a fruta cresce em árvores

(Premissa menor)

A laranja é uma fruta

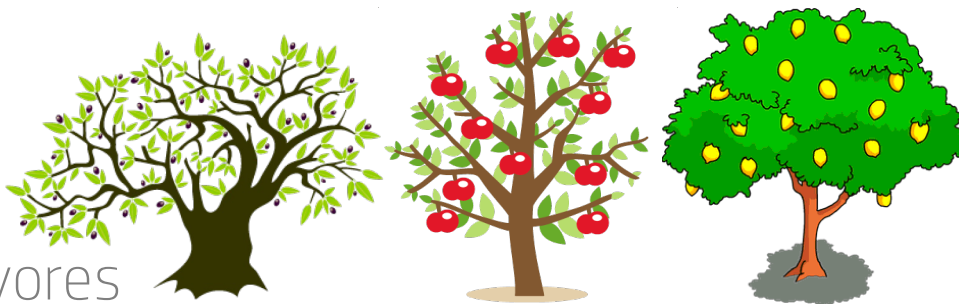


(Conclusão)

A laranja cresce em árvores

# Dedução

(Premissa maior) Toda a fruta cresce em árvores



(Premissa menor)

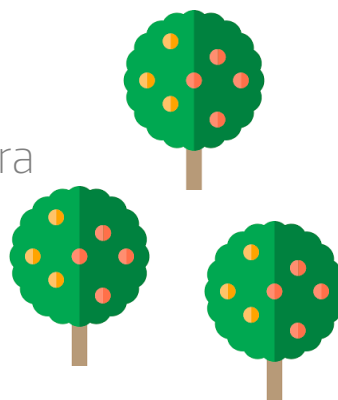
A laranja é uma fruta



(Conclusão)

A laranja cresce em árvores

(no fim montamos uma **experiência** para  
recolher dados e verificar se  
efetivamente a **teoria** se verifica)



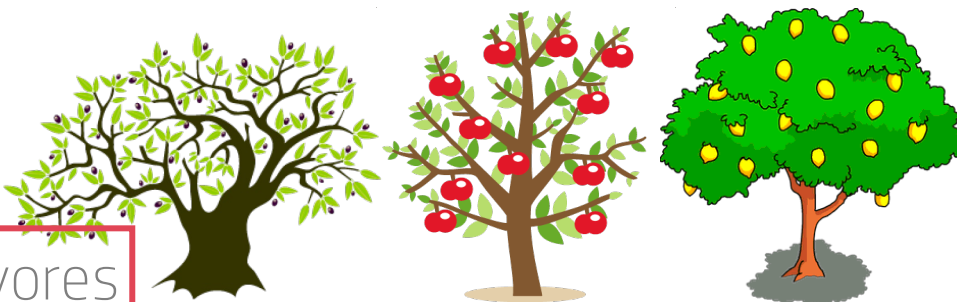


## \_introdução

- Dedução
  - Chegar a conclusões específicas/particulares a partir de ideias que são geralmente aceites
  - Se os factos, propriedades ou premissas são verdadeiros, e aplicamos a lógica corretamente, então as conclusões também são verdadeiras
  - Também conhecida como raciocínio *top-down*
- Para ser correto, um argumento dedutivo necessita passar dois requisitos
  - Validade – quando a estrutura lógica está correta
  - Solidez – quando um argumento é válido, e todas as premissas são verdadeiras

# Dedução

(Premissa maior) Toda a fruta cresce em árvores



(Premissa menor) A laranja é uma fruta



(Conclusão) A laranja cresce em árvores

Argumento válido, não sólido

## \_introdução

- Existem 3 grandes tipos de raciocínio dedutivo
  - Silogismo (raciocínio categórico)
    - Silogismo hipotético
  - *Modus ponens* (regra da afirmação)
  - *Modus tollens* (regra da negação)

## \_silogismo

- Raciocínio categórico
- Baseia-se na noção de **pertença** a uma categoria
- Não é uma implicação condicional (se... então...)
  - Lida com categorias e pertença
- Exemplo
  - Premissa maior: Todos os cães são mamíferos
  - Premissa menor: Bobi é um cão
  - Conclusão: Bobi é mamífero

## *\_modus ponens*

- Regra da afirmação
- Baseia-se em uma **implicação condicional**, não existindo a noção de pertença
- Afirma-se o antecedente para concluir o consequente
  - Usa a afirmação para provar
- Exemplo
  - Premissa maior: Se está a chover, a rua está molhada
  - Premissa menor: Está a chover
  - Conclusão: A rua está molhada

## *\_modus tollens*

- Regra da negação
- Baseia-se na **negação da consequência**, para negar o antecedente
- Útil para provar que algo é falso
  - Usa a negação para refutar
- Exemplo
  - Premissa maior: Se está a chover, a rua está molhada
  - Premissa menor: A rua não está molhada
  - Conclusão: Não está a chover

## \_silogismo hipotético

- Tipo especial de silogismo, baseado em **implicações condicionais**
  - Não existe a noção de pertença
  - Não existem premissas maiores/menores, pois as condições estão geralmente ao mesmo nível (são mais encadeadas que hierárquicas)
- Liga duas implicações numa só, podendo-se “saltar” o termo intermédio
- Exemplo
  - Condição 1: Se estudo prolog, passo no exame de IA
  - Condição 2: Se passo no exame de IA, termino o curso
  - Conclusão: Se estudo prolog, termino o curso

Geral

## Dedução

(Premissa maior) Toda a fruta cresce em árvores

(Premissa menor) A laranja é uma fruta



(Conclusão) A laranja cresce em árvores



Particular

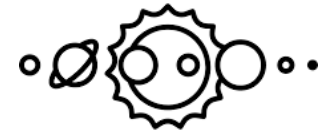
A Terra, Marte, e Júpiter são planetas

A terra é redonda

Marte é redondo

Júpiter é redondo

...





# Geral

## Dedução

(Premissa maior) Toda a fruta cresce em árvores

(Premissa menor) A laranja é uma fruta



(Conclusão) A laranja cresce em árvores



# Particular

Todos os planetas são redondos



A Terra, Marte, e Júpiter são planetas

A terra é redonda

Marte é redondo

Júpiter é redondo

...



Geral

## Dedução

(Premissa maior) Toda a fruta cresce em árvores

(Premissa menor) A laranja é uma fruta

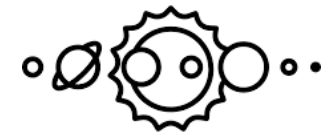


(Conclusão) A laranja cresce em árvores

Particular

## Indução

Todos os planetas são redondos (serão?)



A Terra, Marte, e Júpiter são planetas

A terra é redonda

Marte é redondo

Júpiter é redondo

...

Geral

## Dedução



## Indução



Particular

Geral

Dedução

Teoria



Hipótese



Observação



Conclusão



Particular

Indução

Teoria



Hipótese



Padrão



Observação

Machine Learning

Lógica,  
Escola  
Simbólica

## \_resumindo...

- Método dedutivo
  - Método de **raciocínio lógico** que usa **dedução** para obter uma **conclusão** a partir de determinadas premissas
  - Se as premissas são **verdadeiras** e o raciocínio for logicamente **válido**, as conclusões também são verdadeiras
- Método indutivo
  - Método de raciocínio em que as premissas são vistas como fornecendo **algumas provas** sobre a veracidade da conclusão
  - A veracidade da conclusão num argumento indutivo é **provável**, com base nas provas observadas

# A linguagem Prolog

## \_a escola simbólica

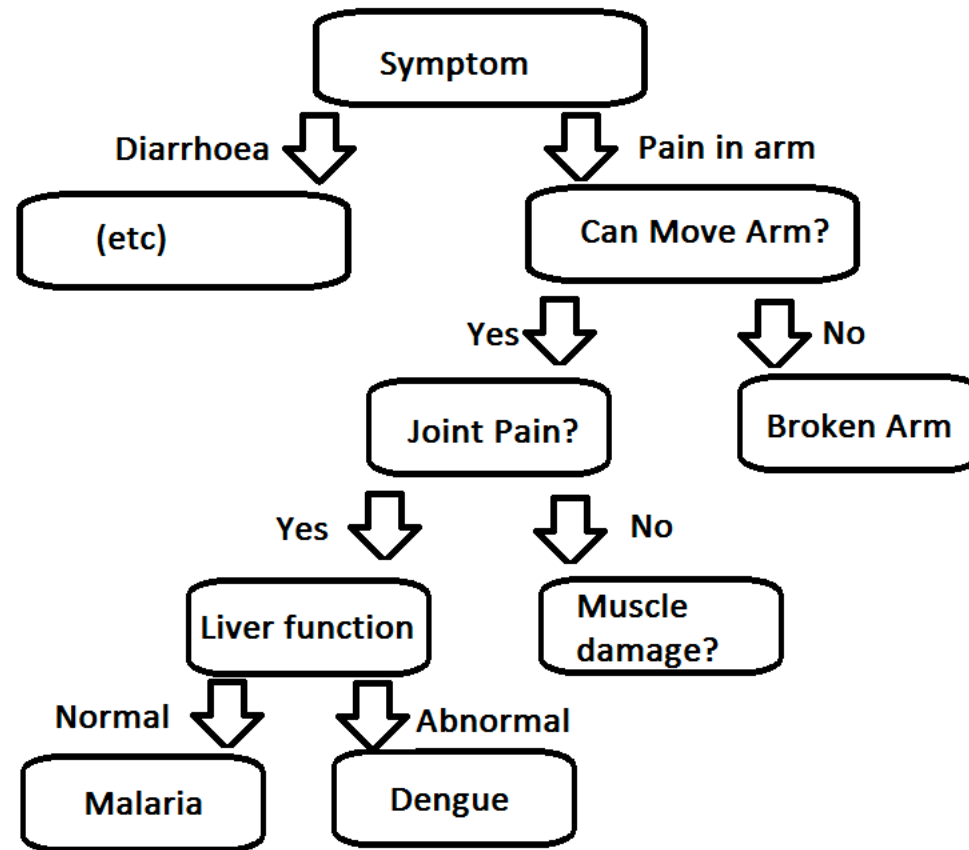
- Conjunto de métodos da IA que usam **representações simbólicas** (e.g. as próprias palavras/conceitos), de alto nível, que um Humano consegue **perceber** e **interpretar**
- Abordagem baseada na ideia de que muitos aspetos da inteligência são conseguidos através da manipulação de símbolos
  - E.g. raciocinamos com “palavras”
- Foram a forma dominante de IA desde o seu início, nos anos 50, até aproximadamente 1980

## \_expert systems

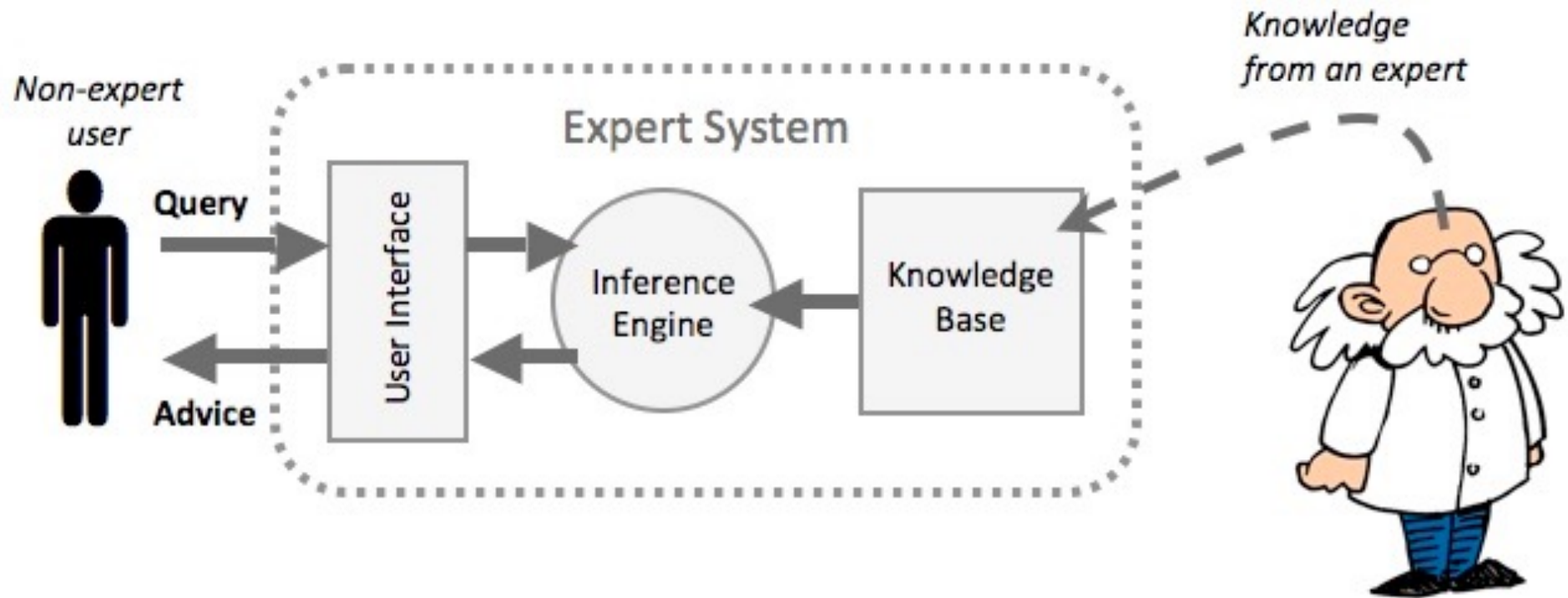
- Os chamados Expert Systems (sistemas periciais) são provavelmente a forma mais conhecida de IA simbólica
- Usam redes de **production rules**
  - Ligam símbolos através de relações similares a instruções if-then
  - Relações são criadas com base em peritos humanos e outras fontes
  - O sistema processa as regras para fazer deduções e **determinar novas informações**
  - Utiliza símbolos legíveis para um Humano



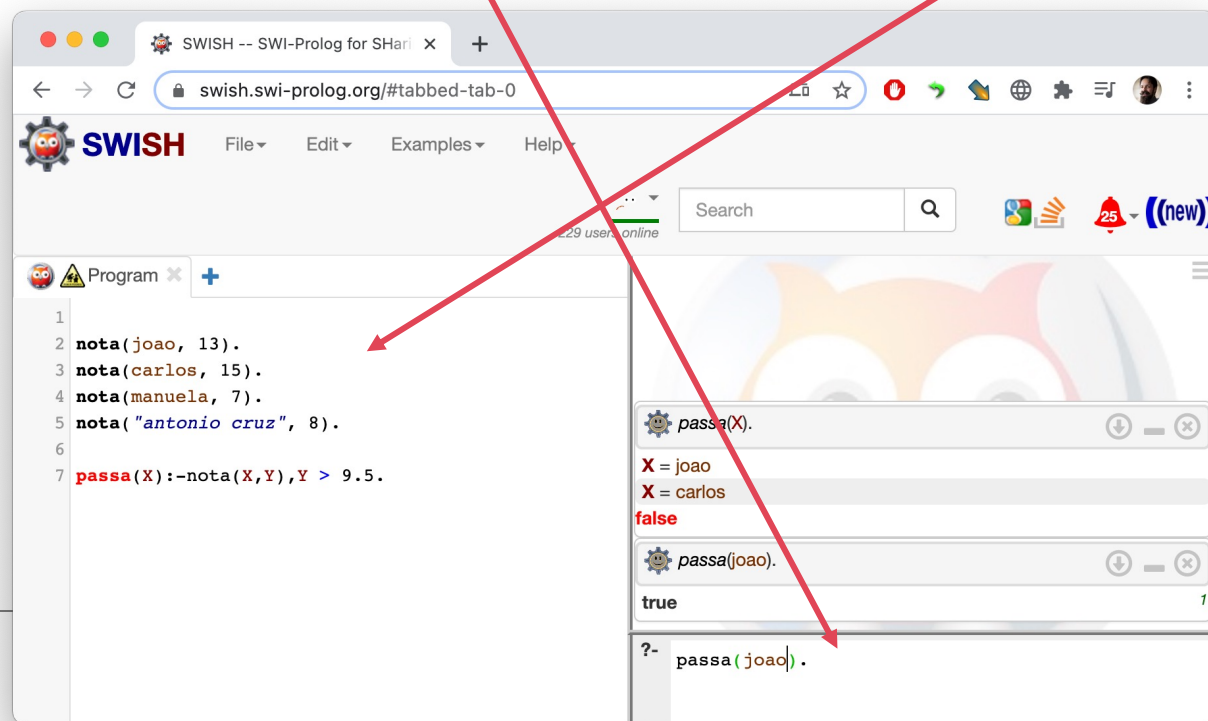
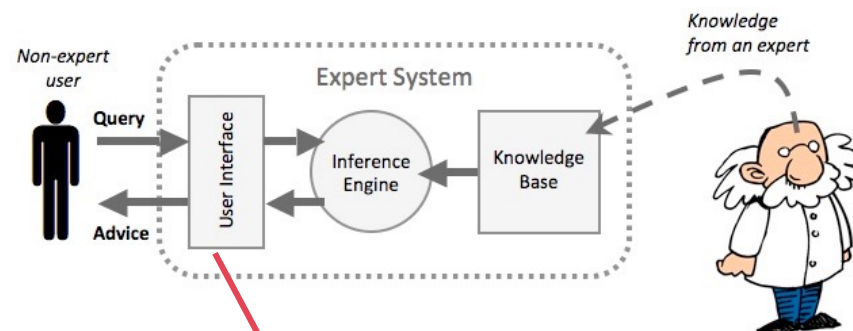
# \_expert systems



## \_expert systems



## \_expert systems



## \_expert systems

- A aquisição e manutenção de regras é geralmente dispendiosa
- É desejável que estes processos sejam feitos de forma automática
- Existem algoritmos e aplicações que ajudam neste processo (rules mining)
  - E.g. Zingtree
- Vamos estudar duas abordagens (em aulas futuras)
  - Árvores de Decisão
  - Regras de associação

## \_prolog

- A Linguagem PROLOG foi criada nos anos 70 por Alain Colmareur, na Universidade de Marselha
- O nome da linguagem vem de **PRO**gramming in **LOG**ic, ou seja, segue o paradigma da Programação em Lógica
- É conjuntamente com a linguagem LISP, criada nos anos 50, uma das linguagens específicas para o desenvolvimento de Sistemas de Inteligência Artificial
- Enquanto que a linguagem LISP teve impacto nos EUA, o PROLOG alcançou notoriedade na Europa e no Japão
- O principal standard de PROLOG foi proposto em Edimburgo, por Clocksin & Mellish

teste.plUNREGISTERED

teste.pl

```
1
2 hello(world).
3
4 ai_is_cool.
5
6 uc('inteligencia artificial', lei).
7 uc('fundamentos de programação', lei).
8 uc('tecnologias escaláveis para análise de dados', mei).
9
10
```

Line 9, Column 1Tab S

```
Welcome to SWI-Prolog (threaded, 64 bits, version 8.0.2)
SWI-Prolog comes with ABSOLUTELY NO WARRANTY. This is free software.
Please run ?- license. for legal details.

For online help and background, visit http://www.swi-prolog.org
For built-in help, use ?- help(Topic). or ?- apropos(Word).

?- consult(['Users/davidecarneiro/Downloads/teste.pl']).
true.

?- hello(X).
X = world.
|
?- ai_is_cool.
true.

?- ai_is_not_cool.
ERROR: Undefined procedure: ai_is_not_cool/0 (DWIM could not correct goal)
?- uc(X, lei).
X = 'inteligencia artificial'
X = 'fundamentos de programação'
?- uc('inteligencia artificial', mei).
false.

?-
```

<https://swish.swi-prolog.org/>

The screenshot shows the SWISH web interface in a browser. The address bar displays `swish.swi-prolog.org/#tabbed-tab-0`. The page header includes the SWISH logo, navigation menus (File, Edit, Examples, Help), a search bar, and a notification for 229 users online. The main content area is divided into two panels. The left panel, titled "Program", contains the following Prolog code:

```
1 nota(joao, 13).
2 nota(carlos, 15).
3 nota(manuela, 7).
4 nota("antonio cruz", 8).
5
6
7 passa(X):-nota(X,Y),Y > 9.5.
```

The right panel displays the execution results for the query `passa(X).`. It shows two bindings for `X`: `X = joao` and `X = carlos`. The result for the first binding is `false`, and the result for the second binding is `true`. The interface also shows a query prompt `?- passa(joao).` at the bottom.

## \_conceitos básicos

- Em Prolog, é possível representar conhecimento de duas formas
  - Explícita – através de factos e regras
  - Implícita – através da consequência lógica da aplicação de factos e regras



## \_conceitos básicos

- **Factos** – correspondem a axiomas

```
rio(douro) .
```

```
pai(pedro, ana) .
```

- **Regras** – correspondem a implicações (*a abordar na próxima aula*)

```
neto(N,A) :- filho(N,P), (descendente(P,A,_); descendente(P,_,A)) .
```

- **Questões** – permitem interrogar a BC

```
?-pai(P, ana) .
```

```
?-pai(pedro, ana) .
```

```
?-neto(rui, A) .
```

## \_pesquisa

- Quando colocamos uma questão ao interpretador, há um processo de **pesquisa** (*search*) cujo objetivo é tentar **encontrar uma forma de provar que a questão é verdadeira**
- A base de conhecimento é percorrida **sequencialmente**, parando quando se encontra **a primeira prova** (se existir)
  - Neste momento, é possível ao utilizador **pedir mais alternativas**
  - Neste caso, o processo de pesquisa continua
- O processo de pesquisa termina quando **todas as respostas forem encontradas**, ou se o utilizador **não pedir mais alternativas**

SWISH -- SWI-Prolog for SHari x

swish.swi-prolog.org/#tabbed-tab-0

SWISH File Edit Examples Help

201 users online Search

Program x +

```
1
2 %nota(nome do aluno, valor)
3 nota(joao, 13).
4 nota(carlos, 15).
5 nota(manuela, 7).
6 nota("antonio cruz", 8).
7
8 %passa(nome do aluno)
9 passa(X):-nota(X,Y),Y > 9.5.
```

nota("antonio cruz", 8).

true

?- nota("antonio cruz", 8).

# \_unificação

- Quando utilizamos **variáveis** numa questão e o prolog encontra uma resposta para a questão, as variáveis são **unificadas com o respetivo valor**
  - Nesse momento, a variável fica **instanciada** com esse valor
- Quando num facto ou regra não interesse o valor de uma variável, esta pode ser substituída por um '\_'
  - Faz com que unifique com qualquer valor

SWISH -- SWI-Prolog for SHarI x +

swish.swi-prolog.org/#tabbed-tab-0

SWISH File Edit Examples Help

202 users online

Search

Program

```
1
2 %nota(nome do aluno, valor)
3 nota(joao, 13).
4 nota(carlos, 15).
5 nota(manuela, 7).
6 nota("antonio cruz", 8).
7
8 %passa(nome do aluno)
9 passa(X):-nota(X,Y),Y > 9.5.
```

nota(Aluno, Nota).

Aluno = joao,  
Nota = 13

Next 10 100 1,000 Stop

?- nota(Aluno, Nota).

SWISH -- SWI-Prolog for SHarI x +

swish.swi-prolog.org/#tabbed-tab-0

SWISH File Edit Examples Help

204 users online

Search

Program

```
1
2 %nota(nome do aluno, valor)
3 nota(joao, 13).
4 nota(carlos, 15).
5 nota(manuela, 7).
6 nota("antonio cruz", 8).
7
8 %passa(nome do aluno)
9 passa(X):-nota(X,Y),Y > 9.5.
```

nota(Aluno, Nota).

Aluno = joao,  
Nota = 13  
Aluno = carlos,  
Nota = 15  
Aluno = manuela,  
Nota = 7  
Aluno = "antonio cruz",  
Nota = 8

?- nota(Aluno, Nota).

SWISH -- SWI-Prolog for SHari

swish.swi-prolog.org/#tabbed-tab-0

172 users online

Search

25 (new)

SWISH

File Edit Examples Help

Program

1

2 %nota(nome do aluno, valor)

3 nota(joao, 13).

4 nota(carlos, 15).

5 nota(manuela, 7).

6 nota("antonio cruz", 8).

7

8 %passa(nome do aluno)

9 passa(X):-nota(X,Y),Y > 9.5.

10

11

12

13

14

15

16

17

18

19

20

21

22

23

nota(joao, 15).

false

nota(joao, \_).

true

1

nota(X, \_).

X = joao

X = carlos

X = manuela


X = "antonio cruz"

?- nota(X, \_).


Examples History Solutions

☐ table results




Run!

**SWISH**

File Edit Examples Help


  
184 users online

Search


  
25

Program


```
1
2 tipo(banana, fruta).
3 tipo(laranja, fruta).
4 tipo(pera, fruta).
5 tipo(abacate, fruta).
6 tipo(cenoura, legume).
7 tipo(batata, tuberculo).
8
9 %cresce(banana, arvore).
10 %cresce(laranja, arvore).
11 %cresce(pera, arvore).
12 %cresce(abacate, arvore).
13
14 cresce(X, arvore):-tipo(X, fruta),!.
15 cresce(cenoura, terra).
16 cresce(batata, terra).
```

 cresce(cenoura, X).


X = terra

 cresce(abacate, X).

X = arvore

 cresce(laranja, X).

X = arvore

 cresce(pera, X).

X = arvore


?- cresce(pera, X).

Examples History Solutions


☐ table results Run!

### Dedução

(Premissa maior) Toda a fruta cresce em árvores



(Premissa menor) A laranja é uma fruta



(Conclusão) A laranja cresce em árvores

# Inteligência Artificial

Programação em Lógica I

Licenciatura em Engenharia Informática  
2024/2025