ISLab

Universidade do Minho
Escola de Engenharia
Departamento de Informática

Prolog

Introdução

LICENCIATURA EM ENGENHARIA INFORMÁTICA
MESTRADO integrado EM ENGENHARIA INFORMÁTICA
Inteligência Artificial
2022/23

ISLab
Synthetic Intelligence Lab

Prolog

- Descrever o problema;
- Colocar uma questão.

Prolog

- Deduz logicamente novos fatos sobre o problema;
- Devolve as deduções como respostas.

ISLab
Synthetic Intelligence Lab

Conceitos básicos

- Factos - correspondem a axiomas (os objetos que caracterizam o contexto em estudo)

rio(minho).

pai(pedro, raquel).

- Regras – correspondem as relações entre os objetos

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

- Questões – permitem interrogar a BC

?-pai(P, raquel).

?-pai(pedro, raquel).

?-neto(rui, A).

ISLab
Synthetic Intelligence Lab

Facto

- Tem um functor (nome);
- Tem zero ou mais argumentos, englobados por parêntesis e separados por vírgulas, e termina com um ponto;
- Argumentos são
  - Átomos - valores constantes, números, strings...
  - Variáveis - começadas por uma maiúscula.

ISLab
Synthetic Intelligence Lab

Questões sobre a BC

localizacao(porto,portugal).	falha
localizacao(lisboa,portugal).	falha
localizacao(coimbra,portugal).	falha
localizacao(caminha,portugal).	falha
localizacao(madrid,espanha).	sucesso
localizacao(barcelona,espanha).	
localizacao(zamora,espanha).	
localizacao(orense,espanha).	
localizacao(toledo,espanha).	

?-localizacao(madrid,espanha).

yes

atravessa(douro,porto).	falha
atravessa(douro,zamora).	falha
atravessa(tejo,lisboa).	falha
atravessa(tejo,toledo).	falha
atravessa(minho,caminha).	falha
atravessa(minho,orense).	falha

?-atravessa(mondego,coimbra).

no

ISLab
Synthetic Intelligence Lab

Variáveis nas questões sobre a BC

```
localizacao(porto,portugal).
localizacao(lisboa,portugal).
localizacao(coimbra,portugal).
localizacao(caminha,portugal).
localizacao(madrid,espanha).
localizacao(barcelona,espanha).
localizacao(zamora,espanha).
localizacao(orense,espanha).
localizacao(toledo,espanha).
```

falha
falha
falha
falha
sucesso

```
?-localizacao(X,espanha).
X=madrid <cr>
yes
```

```
atravessa(douro,porto).
atravessa(douro,zamora).
atravessa(tejo,lisboa).
atravessa(tejo,toledo).
atravessa(minho,caminha).
atravessa(minho,orense).
```

falha
falha
sucesso

```
?- atravessa(tejo,C).
C=lisboa <cr>
yes
```

```
?-localizacao(X,Y).
X=porto Y=portugal <cr>
yes
```

ISLab
Synthetic Intelligence Lab

Questões com conjunção

localizacao(porto,portugal). sucesso
localizacao(lisboa,portugal).
localizacao(coimbra,portugal).
localizacao(caminha,portugal).
localizacao(madrid,espanha).
localizacao(barcelona,espanha).
localizacao(zamora,espanha).
localizacao(orense,espanha).
localizacao(toledo,espanha).

atravessa(douro,porto). sucesso
atravessa(douro,zamora).
atravessa(tejo,lisboa).
atravessa(tejo,toledo).
atravessa(minho,caminha).
atravessa(minho,orense).

?-localizacao(X,portugal),atravessa(R,X).
X=porto R=douro *<cr>*
yes

E se fossem pedidas alternativas com o ; ?

?-localizacao(X,portugal),atravessa(R,X).
X=porto R=douro *;*
X=lisboa R=tejo *;*
X=caminha R=minho

ISLab
Synthetic Intelligence Lab

Questões com disjunção

```
localizacao(porto,portugal).
localizacao(lisboa,portugal).
localizacao(coimbra,portugal).
localizacao(caminha,portugal).
localizacao(madrid,espanha).
localizacao(barcelona,espanha).
localizacao(zamora,espanha).
localizacao(orense,espanha).
localizacao(toledo,espanha).

atravessa(douro,porto).
atravessa(douro,zamora).
atravessa(tejo,lisboa).
atravessa(tejo,toledo).
atravessa(minho,caminha).
atravessa(minho,orense).
```

sucesso

```
?-localizacao(X,portugal);localizacao(X,espanha).
X=porto <cr>
yes
```

E se fossem pedidas alternativas com o ; ?

```
?-localizacao(X,portugal);localizacao(X,espanha).
X=porto ;
X=lisboa ;
X=coimbra ;
X=caminha ;
X=madrid ;
X=barcelona ;
X=zamora ;
X=orense ;
X=toledo
yes
```

ISLab
Synthetic Intelligence Lab

Questões com negação

localizacao(porto,portugal).	sucesso
localizacao(lisboa,portugal).	
localizacao(coimbra,portugal).	
localizacao(caminha,portugal).	
localizacao(madrid,espanha).	
localizacao(barcelona,espanha).	
localizacao(zamora,espanha).	
localizacao(orense,espanha).	
localizacao(toledo,espanha).	
atravessa(douro,porto).	falha
atravessa(douro,zamora).	falha
atravessa(tejo,lisboa).	falha
atravessa(tejo,toledo).	falha
atravessa(minho,caminha).	falha
atravessa(minho,orense).	falha

```
?-not(localizacao(porto,portugal)).
no

?-not(atravessa(mondego,coimbra)).
yes
```

ISLab
Synthetic Intelligence Lab

Regras

- Uma regra tem 1 termo (functor e argumentos) do lado esquerdo que corresponde àquilo que se pretende provar (conclusão), seguido do :-
- À direita do :- podem aparecer outros termos afectados por operadores (, ; not) e por primitivas de controlo
- O lado direito corresponde às condições da regra
- Tal como nos factos, podemos ter regras com o mesmo nome e mesmo nº de argumentos, ou até com o mesmo nome e nº de argumentos diferentes

ISLab
Synthetic Intelligence Lab

Exemplos de Regras

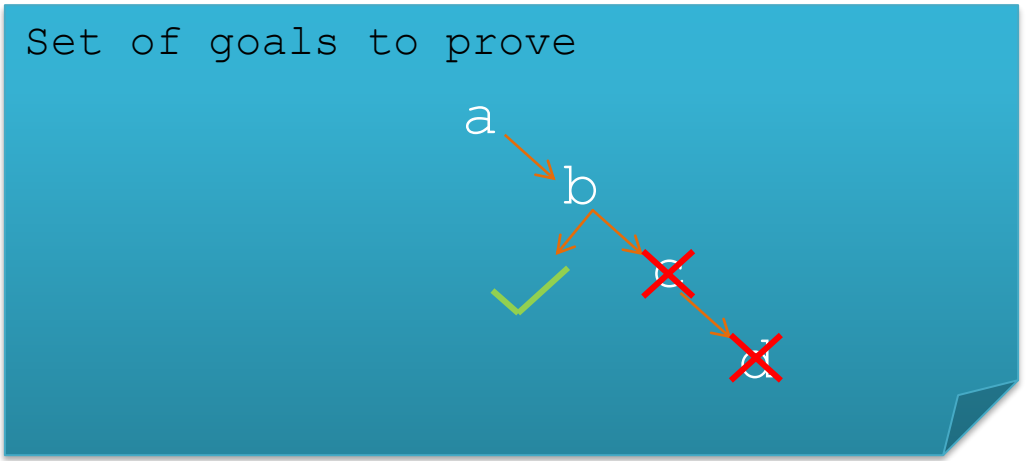
```
filho(X,Y):-homem(X),
(descendente(X,Y,_);descendente(X,_,Y)).


potência(_,0,1):-!.
potência(X,N,P):- N1 is N-1,
                  potência(X,N1,P1),
                  P is X*P1.
```

ISLab
Synthetic Intelligence Lab

Como funciona o Prolog?

- a :- b.
- b :- c.
- c :- d.
- b.

- Qual o resultado de **?- a.**



ISLab
Synthetic Intelligence Lab

Questões sobre Regras e Factos numa BC

```
localizacao(porto,portugal).
localizacao(lisboa,portugal).
localizacao(coimbra,portugal).
localizacao(caminha,portugal).
localizacao(madrid,espanha).
localizacao(barcelona,espanha).
localizacao(zamora,espanha).
localizacao(orense,espanha).
localizacao(toledo,espanha).

atravessa(douro,porto).
atravessa(douro,zamora).
atravessa(tejo,lisboa).
atravessa(tejo,toledo).
atravessa(minho,caminha).
atravessa(minho,orense).

rio_português(R):-atravessa(R,C),localizacao(C,portugal).
mesmo_rio(C1,C2):-atravessa(R,C1),atravessa(R,C2),C1\==C2.
```

sucesso (2)

sucesso (1)

?-rio_português(Rio).

Rio=douro

yes

Na chamada à regra, do lado esquerdo, Rio e R atravessam a ser a mesma variável

atravessa(R,C) tem sucesso com R=douro e C=porto

A chamada seguinte já é feita com C já instanciada com porto, na prática essa chamada é feita como sendo localizacao(porto,portugal)

Quando se atinge o ponto a regra tem sucesso

ISLab
Synthetic Intelligence Lab

Questões sobre Regras e Factos numa BC

```
localizacao(porto,portugal).          falha
localizacao(lisboa,portugal).          sucesso (2)
localizacao(coimbra,portugal).
localizacao(caminha,portugal).
localizacao(madrid,espanha).
localizacao(barcelona,espanha).
localizacao(zamora,espanha).
localizacao(orense,espanha).
localizacao(toledo,espanha).

atravessa(douro,porto).               falha
atravessa(douro,zamora).              falha
atravessa(tejo,lisboa).               sucesso (1)
atravessa(tejo,toledo).
atravessa(minho,caminha).
atravessa(minho,orense).

rio_português(R):-atravessa(R,C),localizacao(C,portugal).
mesmo_rio(C1,C2):-atravessa(R,C1),atravessa(R,C2),C1\==C2.
```

```
?-rio_português(tejo).
yes
```

Na chamada à regra, do lado esquerdo, R localizacao instanciada com o valor tejo

Dentro da regra (lado direito) a 1ª chamada já é feita como sendo atravessa(tejo,C) e tem sucesso com R=tejo e C=lisboa

A chamada seguinte já é feita com C já instanciada com lisboa, na prática essa chamada é feita como sendo localizacao (lisboa,portugal)

Quando se atinge o ponto a regra tem sucesso

ISLab
Synthetic Intelligence Lab

A ordem dos factos importa

```
localizacao(porto,portugal).      falha
localizacao(lisboa,portugal).     falha
localizacao(coimbra,portugal).    falha
localizacao(caminha,portugal).    falha
localizacao(madrid,espanha).      falha
localizacao(barcelona,espanha).   falha
localizacao(zamora,espanha).      falha
localizacao(orense,espanha).      falha
localizacao(toledo,espanha).      falha

atravessa(douro,porto).           falha
atravessa(douro,zamora).          falha
atravessa(tejo,toledo).           sucesso (1)
atravessa(tejo,lisboa).           sucesso (2)
atravessa(minho,caminha).
atravessa(minho,orense).

rio_português(R):-atravessa(R,C), localizacao(C,portugal).
mesmo_rio(C1,C2):-atravessa(R,C1), atravessa(R,C2),C1\==C2.
```

```
?-rio_português(tejo).
yes
```

Na chamada à regra, do lado esquerdo, R localizacao instanciada com o valor tejo;

Dentro da regra (lado direito) a 1ª chamada já é feita como sendo atravessa(tejo,C) e tem sucesso com R=tejo e C=toledo;

A chamada seguinte já é feita com C já instanciada com toledo, na prática essa chamada é feita como sendo localizacao(toledo,portugal) e falha

Volta-se atrás (*backtracking*) e é tentada uma nova solução para atravessa(tejo,C), localizacaondo C=lisboa

A chamada seguinte já é feita com C já instanciada com lisboa, na prática essa chamada é feita como sendo localizacao(lisboa,portugal)

Quando se atinge o ponto a regra tem sucesso

ISLab
Synthetic Intelligence Lab

O predicado cut

- Quando não é necessário encontrar todas as soluções possíveis pode ser usar o predicado cut: !
- Permite indicar ao prolog objetivos já satisfeitos e que não precisam ser reconsiderados no processo de backtracking

```
a1(X, Y) :- b(X), c(Y).

b(1).
b(2).
b(3).

c(1).
c(2).
c(3).
```

a1(X,Y).
X = Y, Y = 1 ;
X = 1, Y = 2 ;
X = 1, Y = 3 ;
X = 2, Y = 1 ;
X = Y, Y = 2 ;
X = 2, Y = 3 ;
X = 3, Y = 1 ;
X = 3, Y = 2 ;
X = Y, Y = 3.

ISLab
Synthetic Intelligence Lab

O predicado cut

- Quando não é necessário encontrar todas as soluções possíveis pode ser usar o predcado cut: !
- Permite indicar ao prolog objetivos já satisfeitos e que não precisam ser reconsiderados no processo de backtracking

```
a2(X, Y) :- b(X), !, c(Y).

b(1).
b(2).
b(3).

c(1).
c(2).
c(3).
```

```
a2(X,Y).
X = Y, Y = 1 ;
X = 1, Y = 2 ;
X = 1, Y = 3.
```

ISLab
Synthetic Intelligence Lab

Variáveis

- As variáveis em PROLOG têm um desempenho diferente das variáveis de outras linguagens;
- Em PROLOG uma variável pode estar apenas em dois estados: não instanciada ou instanciada;
- Uma vez instanciada uma variável só pode mudar de valor pelo processo de *backtracking*, ou seja, voltando a localização como não instanciada para tomar outro valor

ISLab
Synthetic Intelligence Lab

Variáveis

- Em PROLOG o incremento de uma variável nunca pode ser feito como N is N+1 (is é a atribuição numérica)
  - Se N não estiver instanciado ocorre uma falha quando se tenta avaliar N+1
  - Se N estiver instanciado não poderemos obrigar a mudar o seu valor
  - Pode ser usado N1 is N+1
- Quando num facto ou regra não interesse o valor de uma variável, esta pode ser substituída por _

ISLab
Synthetic Intelligence Lab

Operadores Lógicos

- Já foi visto que:
  - , para a conjunção
  - ; para a disjunção
  - not para a negação
- Podemos usar os () para tirar ambiguidades ou forçar as expressões pretendidas
- Vamos considerar a seguinte base de factos:

a.

b.

c:-fail. /* origina uma falha */

d.

ISLab
Synthetic Intelligence Lab

Operadores Lógicos

Base de Factos

```
a.
b.
c:-fail.
d.
```

Questões:

```
?- a.
yes
?- c.
no
?- not(a).
no
?- not(c).
yes
?- a,b.
yes
?- a,c.
no
?- a;c.
yes
?- (a,c);(not(a);b).
yes
```

ISLab
Synthetic Intelligence Lab

Operadores Aritméticos

▪ Adição		X + Y
▪ Subtracção		X - Y
▪ Multiplicação		X * Y
▪ Divisão		X / Y
▪ Divisão inteira		X // Y
▪ Resto da divisão inteira	X mod Y	
▪ Potência		X ^ Y
▪ Simétrico		- X

ISLab
Synthetic Intelligence Lab

Funções Aritméticas

- ip(X) parte inteira de X
- ln(X) logaritmo natural de X
- log(X) logaritmo decimal de X
- max(X,Y) máximo entre X e Y
- min(X,Y) mínimo entre X e Y
- rand(X) gera um número aleatório entre 0 e X
- sign(X) sinal de X
- sin(X) seno de X (graus)
- sqrt(X) raiz quadrada de X
- tan(X) tangente de X (graus)

ISLab
Synthetic Intelligence Lab

Operadores Relacionais

- Igualdade `X==Y`
- Diferença `X\==Y`
- Maior `X>Y`
- Menor `X<Y`
- Menor ou igual `X=<Y`
- Maior ou igual `X >= Y`

ISLab
Synthetic Intelligence Lab

Atribuição

- = para a atribuição simbólica X=a
- is para a atribuição numérica X is 5

- A atribuição simbólica é bidireccional

- Para X=Y
  - Se X não está instanciado e Y está então temos X←Y
  - Se X está instanciado e Y não está então temos X→Y
  - Se nenhum está instanciado então atravessam a ser a mesma variável
  - Se ambos estão instanciados com o mesmo valor então há sucesso
  - Se ambos estão instanciados com valores diferentes então ocorre uma falha

ISLab
Synthetic Intelligence Lab

Atribuição

- A atribuição numérica é unidireccional
- Do lado direito do *is*, se estiverem envolvidas variáveis, elas devem estar instanciadas
- Do lado esquerdo a variável não deve estar instanciada, senão ocorre uma falha
- Do lado direito as variável que apareçam devem estar instanciadas
- Em PROLOG N is N+1 nunca tem sucesso

ISLab
Synthetic Intelligence Lab

Recursividade

- O uso da recursividade é muito comum na linguagem PROLOG;
- Na implementação de um predicado recursivo devemos ter em atenção que deverá haver sempre uma alternativa para finalizar as chamadas recursivas:
  - Por uma regra, ou facto, que não efectua essa chamada;
  - Por uma das alternativas de uma disjunção.

factorial(0,1):-!.

factorial(N,F):-N1 is N-1,factorial(N1,F1),F is N*F1.

Vejamos o que acontece quando se efetua a chamada ?-factorial(3,F).

F=6

factorial(0,1) falha

factorial(3,F):-N1 ⟵ 3-1,factorial(2,F1) , F is 3*2. sucesso (c/ F ⟵ 6)

factorial(0,1) falha

factorial(2,F):-N1 ⟵ 2-1,factorial(1,F1) , F is 2*1. sucesso (c/ F ⟵ 2)

factorial(0,1) falha

factorial(1,F):-N1 ⟵ 1-1,factorial(0,F1) , F is 1*1. sucesso (c/ F ⟵ 1)

factorial(0,1):-!. sucesso

Listas

- Em PROLOG as listas podem ser:
  - Não vazias, tendo uma cabeça (1° elemento da lista) e uma cauda (lista com os restantes elementos)
  - Vazias, quando não têm nenhum elemento
- As listas podem ser representadas
  - Pela enumeração dos seus elementos separados por vírgulas e envolvidos por [e] (por exemplo [a,b,c,d])
  - Pela notação cabeça-cauda separadas pelo | e envolvidas por [e] (por exemplo [H|T])
- Em PROLOG os elementos das listas não têm de ser do mesmo tipo (por exemplo, [a,2,abc,[x,1,zzz]])

ISLab
Synthetic Intelligence Lab

Listas

- [] é a lista vazia
- [a] é uma lista com 1 elemento (a)
- [X] é uma lista com 1 elemento (a variável X)
- [b,Y] é uma lista com 2 elementos (b e a variável Y)
- [X,Y,Z] é uma lista com 3 elementos (as variáveis X,Y e Z)
- [H|T] é uma lista com cabeça H e cauda T

Vejamos algumas questões PROLOG:

```
?- [H|T]=[a,b,c,d].
H = a ,
T = [b,c,d]

?- [H|T]=[a,b,X].
H = a ,
T = [b,X] ,
X = _

?- [H|T]=[a].
H = a ,
T = []

?- [H|T]=[[a,b],3,[d,e]].
H = [a,b] ,
T = [3,[d,e]]

?- [H|T]=[].
no
```

ISLab
Synthetic Intelligence Lab

Alteração dinâmica de conhecimento

- Podemos inserir, dinamicamente, novos factos ou regras na Base de Conhecimento
- Tais factos ou regras têm que ser declaradas como sendo dinâmicas através de
- dynamic *predicado/aridade*

- asserta, assertz
- retract

ISLab
Synthetic Intelligence Lab

Alteração dinâmica de conhecimento

```
:-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

Soluções Múltiplas

bagof(X,Q,L)

- L é a lista dos X que atendem a Q, se não houver solução bagof falha

setof(X,Q,L)

- L é a lista dos X que atendem a Q, L vem ordenada, elementos repetidos são eliminados, se não houver solução setof falha

findall(X,Q,L)

- L é a lista dos X que atendem a Q, se não houver solução findall tem sucesso com L=[]

ISLab
Synthetic Intelligence Lab

Soluções Múltiplas

```
f(a,1).
f(a,2).
f(a,2).
f(z,6).
f(z,5).
f(x,4).
f(x,3).
```

```
?- findall(f(L,N),f(L,N),Lista).
L = _ ,
N = _ ,
Lista = [f(a,1),f(a,2),f(a,2),f(z,6),f(z,5),f(x,4),f(x,3)]

?- setof(f(L,N),f(L,N),Lista).
L = _ ,
N = _ ,
Lista = [f(a,1),f(a,2),f(x,3),f(x,4),f(z,5),f(z,6)]

?- bagof(f(L,N),f(L,N),Lista).
L = _ ,
N = _ ,
Lista = [f(a,1),f(a,2),f(a,2),f(z,6),f(z,5),f(x,4),f(x,3)]

?- findall(f(L,N),(f(L,N),N>7),Lista).
L = _ ,
N = _ ,
Lista = []

?- setof(f(L,N),(f(L,N),N>7),Lista).
no

?- bagof(f(L,N),(f(L,N),N>7),Lista).
no
```

ISLab
Synthetic Intelligence Lab

Alguns predicados úteis

- member/2
- append/3
- reverse/2
- delete/3
- Outros: http://www.swi-prolog.org/pldoc/man?section=builtin

ISLab
Synthetic Intelligence Lab

Referências

- Prolog Programming for Artificial Intelligence (4th Edition), Ivan Bratko, ISBN-13: 978-0321417466, 2011.
- Artificial Intelligence: A Modern Approach, Stuart Russell and Peter Norvig, (3rd Edition), ISBN 978-9332543515, 2015.
- Inteligência Artificial-Fundamentos e Aplicações, E.Costa, A.Simões; FCA, ISBN: 978-972-722-340-4, 2008.

ISLab
Synthetic Intelligence Lab

Implementações de prolog

SWI-Prolog - A Free Software Prolog environment, licensed under the Lesser GNU public license. This popular interpreter was developed by Jan Wielemaker. This is the interpreter we used while developing this book.
http://www.swi-prolog.org/

SICStus Prolog - Industrial strength Prolog environment from the Swedish Institute of Computer Science. **http://www.sics.se/sicstus/**

GNU Prolog - Another more widely used free Prolog compiler developed by Daniel Diaz. http://www.gprolog.org

YAP Prolog - A Prolog compiler developed at the Universidade do Porto and Universidade Federa do Rio de Janeiro. Free for use in academic environments.
http://www.ncc.up.pt/~vsc/Yap/

ISLab

Universidade do Minho
Escola de Engenharia
Departamento de Informática

Prolog

introdução

LICENCIATURA EM ENGENHARIA INFORMÁTICA
MESTRADO integrado EM ENGENHARIA INFORMÁTICA
Inteligência Artificial
2022/23