EI (diurno e PL) **ETI (diurno e PL)** **IGE (diurno e PL)**

Inteligência Artificial 2017-2018

2º Exame

2018/02/07, 02:00H

Nota: nos exercícios que se seguem, a lógica e as ferramentas computacionais utilizadas funcionam exatamente como explicado nas aulas; as respostas devem seguir os mesmos padrões.

I – Perguntas avulsas

Responde às seguintes perguntas, sem justificar.

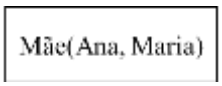
(1 Valor) 1a) Dos três componentes da arquitetura de um Sistema Baseado em Conhecimento (Base de Conhecimento, Motor de Inferência, e Interface), qual ou quais deles não integram os outros sistemas computacionais?

R: Base de Conhecimento e Motor de Inferência

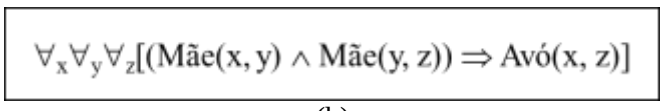
(1 Valor) 1b) Na base de conhecimento de um Sistema Baseado em Conhecimento, pode usar-se uma tecnologia não simbólica? **[Uma resposta errada desconta 0.5 Valores]**

R: Não

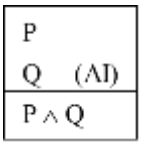
(0.5 Valores) 1c) Em que componente de um Sistema Baseado em Conhecimento se arrumaria cada uma das seguintes estruturas simbólicas:



(a)



(b)



(c)

[A pontuação é atribuída apenas a respostas totalmente certas]

R: (a) e (b): Base de Conhecimento. (c): Motor de Inferência

NOTA: O texto usa a palavra "arrumaria" (*em que componente se arrumaria*), o que excluiria a *Interface*. Nada se arruma na *Interface*. No entanto, considerámos certas as respostas em que as estruturas simbólicas (a) e (b) poderiam ser apresentadas à interface.

II – Lógica de Predicados

(2.5 Valores) 2 – Considera a seguinte base de conhecimento:

1. $\forall x$ [TemCalcanharViscoso(x) $\Rightarrow$ Atendimento(x, LavagemPedonal)]
2. $\forall x$ [TemComichãoCerebral(x) $\Rightarrow$ Atendimento(x, LavagemCerebral)]
3. $\forall x$ [($\neg$TemComichãoCerebral(x) $\wedge$ $\neg$TemCalcanharViscoso(x)) $\Rightarrow$ Atendimento(x, MarteloBicudo)]
4. $\neg$Atendimento(Limpinho, LavagemPedonal)
5. $\neg$Atendimento(Limpinho, LavagemCerebral)

Sem recorrer à forma clausal, mostra que se deriva da base de conhecimento que o tipo de atendimento do Limpinho é o martelo de bico (*i.e.*, Atendimento(Limpinho, MarteloBicudo))

R:

6. TemCalcanharViscoso(Limpinho) $\Rightarrow$ Atendimento(Limpinho, LavagemPedonal) 1 UI
7. TemComichãoCerebral(Limpinho) $\Rightarrow$ Atendimento(Limpinho, LavagemCerebral) 2 UI
8. ($\neg$TemComichãoCerebral(Limpinho) $\wedge$ $\neg$TemCalcanharViscoso(Limpinho)) $\Rightarrow$ Atendimento(Limpinho, MarteloBicudo)] 3 UI
9. $\neg$TemCalcanharViscoso(Limpinho) 4, 6 MT
10. $\neg$ TemComichãoCerebral(Limpinho) 5, 7 MT
11. ($\neg$TemComichãoCerebral(Limpinho) $\wedge$ $\neg$TemCalcanharViscoso(Limpinho)) 9, 10 AI
12. Atendimento(Limpinho, MarteloBicudo) 11, 8 MP

NOTA: Respostas que não indiquem as regras de inferência usadas ou que não indiquem as proposições de que se deriva cada conclusão sofrem descontos

(2.5 Valores) 3 - Converte a proposição seguinte em forma clausal:

$\forall x$ [($\neg$TemComichãoCerebral(x) $\wedge$ $\neg$TemCalcanharViscoso(x)) $\Rightarrow$ Atendimento(x, MarteloBicudo)]

R:

1. Substituir as implicações: $(P \Rightarrow Q) \equiv (\neg P \vee Q)$

$\forall x$ [$\neg$($\neg$TemComichãoCerebral(x) $\wedge$ $\neg$TemCalcanharViscoso(x)) $\vee$ Atendimento(x, MarteloBicudo)]

2. Mover as negações para os literais: $\neg(P \wedge Q) \equiv (\neg P \vee \neg Q)$

$\forall x$ [($\neg\neg$TemComichãoCerebral(x) $\vee$ $\neg\neg$TemCalcanharViscoso(x)) $\vee$ Atendimento(x, MarteloBicudo)]

$\neg\neg P \equiv P$

$\forall x$ [(TemComichãoCerebral(x) $\vee$ TemCalcanharViscoso(x)) $\vee$ Atendimento(x, MarteloBicudo)]

3. Skolemização

Não aplicável porque não existem quantificadores existenciais

4. Remover quantificadores universais

TemComichãoCerebral(x) $\vee$ TemCalcanharViscoso(x) $\vee$ Atendimento(x, MarteloBicudo)

5. Obter uma conjunção de disjunções

Não se aplica porque já temos uma disjunção de literais

6. Escrever cláusulas

{TemComichãoCerebral(x), TemCalcanharViscoso(x), Atendimento(x, MarteloBicudo)}

NOTA: Respostas em que faltem os passos da conversão, ou as transformações lógicas usadas, ou as razões pelas quais certos passos não se apliquem serão penalizadas.

III – Regras de Produção

Considera um sistema no âmbito de investimentos na bolsa. As decisões de investimento dependem das cotações dos vários produtos disponíveis. Para tal, as cotações de ações, ao longo do tempo, são mantidas por factos do predicado *cotação/4*, com a seguinte descrição:

cotação(Id, Data, Hora, Valor): no instante de tempo *Hora* do dia *Data*, as ações do produto *Id* tinham o valor *Valor*. Estes factos fazem parte do sistema de informação da Bolsa.

Exemplos:

`cotação(msft, 2017/11/13, 12:30, 84.56).` Cotação da Microsoft às 12:30 do dia 13 de Novembro de 2017.

`cotação(amzn, 2017/11/13, 12:30, 1109.15).` Cotação da Amazon

(2.5 Valores) 4 – Escreve uma regra que cria factos do predicado *valor_de_fecho/3* com os valores dos vários produtos na hora de fecho da bolsa, na data especificada. Por exemplo

`valor_de_fecho(ctt, 2017/11/13, 3.7):` O valor dos CTT, há hora do fecho da bolsa no dia 13 de Novembro de 2017, foi de 3.7 Euros.

As regras que desenhares podem contar com a existência do predicado *fecho/2* que mantem a hora de fecho da bolsa na data especificada, por exemplo

`fecho(2017/11/13, 16:30).` No dia 13 de novembro, a bolsa fechou às 16:30.

O valor de fecho de uma ação é o valor dessa ação à hora de fecho. Assume-se que existe sempre o facto *cotação/4* correspondente à hora de fecho.

R:

```
if    (fecho(Data, Hora) and cotacao(Id, Data, Hora, Valor) and
       \+ valor_de_fecho(Id, Data, _))
then assert(valor_de_fecho(Id, Data, Valor)).
```

(2.5 Valores) 5 – Escreve um conjunto de uma ou mais regras para criar factos do predicado *valor_máximo/3*, tal que valor_maximo(Id, Data, Valor) significa que, no dia *Data*, o valor mais elevado da ação *Id* foi *Valor*.

R:

```
if    (cotacao(Id, Data, _, ValorMax) and
       \+( (cotação(Id, Data, _, Valor) and Valor > ValorMax) ) and
       \+ valor_maximo(Id, Data, _))
then  assert(valor_maximo(Id, Data, ValorMax)).
```

IV – Prolog declarativo

Neste grupo não podes usar nem *cut*, nem *repeat*, nem *assert*, nem *retract*, nem nenhum outro mecanismo de controlo da linguagem Prolog.

Os exercícios apresentados neste grupo usam listas de pares (Id, Retorno) que representam o retorno obtido de um investimento identificado por *Id*.

(2.5 Valores) 6 – Escreve o predicado recursivo *lista_de_retornos/2* que recebe uma lista de investimentos e seus retornos e devolve a lista de retornos desses investimentos, por exemplo

```
?- lista_de_retornos([(i1, 10), (i2, -4), (i3, 19)], L).
L = [10, -4, 19]
```

O predicado deve verificar que o seu primeiro argumento é uma lista de investimentos e que o seu segundo argumento é uma variável ou uma lista de números. Em cada um dos pares que representam investimentos, o primeiro elemento é um átomo e o segundo é um número. Em alguma passagem, a tua resolução tem de ser recursiva.

R:

```
lista_de_retornos(Invests, Retornos):-
     is_list(Invests), (var(Retornos); is_list(Retornos)),
     retornos(Invests, Retornos).
retornos([(Nome, Ret) | LI], [Ret|LR]):-
     atom(Nome), number(Ret),
     retornos(LI, LR).
retornos([], []).
```

(2.5 Valores) 7 – Escreve o predicado recursivo *soma_dos_retornos/3* que recebe uma lista de investimentos e seus retornos e devolve a soma dos retornos negativos (segundo argumento), e a soma dos retornos positivos (terceiro argumento), por exemplo

```
?- soma_dos_retornos([(i1, 10), (i2, -4), (i3, 19)], Neg, Pos).
Neg = -4   Pos = 29
```

Não faças validações dos dados recebidos pelo predicado.

R:

```
soma_dos_retornos([(_, R)|Invests], Neg, Pos):-
    R >= 0,
    soma_dos_retornos(Invests, Neg, PTmp),
    Pos is PTmp + R.
soma_dos_retornos([(_, R)|Invests], Neg, Pos):-
    R < 0,
    soma_dos_retornos(Invests, NTmp, Pos),
    Neg is NTmp + R.
soma_dos_retornos([], 0, 0).
```

V – Prolog procedimental

Neste grupo não podes usar recursividade; tens de usar obrigatoriamente o mecanismo adequado

de repetição por falha. Respostas com a escolha errada do mecanismo de repetição terão classificação nula.

(2.5 Valores) 8 – Imagina que tens um conjunto de factos com investimentos, como os que se exemplificam:

`invest(google, 5000, 5500).` Investimento de 5000 euros na Google com retorno de 5500 euros.

```
invest(amazon, 15000, 14575).
```

```
invest(msft, 5000, 5250).
```

Escreve o programa *lucro_total/1* para calcular a soma dos lucros de todos os investimentos, por exemplo

```
?- lucro_total(X).
X = 325
?-
```

O lucro de um investimento é a diferença entre o valor retornado e o valor investido (ou seja, lucros negativos representam prejuízos). Não faças validações dos dados processados pelo programa.

R:

```
lucro_total(_):-
    assert(lucro(0)),
    invest(_, I, R),
    atualiza_lucro(I, R),
    fail.
lucro_total(X):-
    retract(lucro(X)).

atualiza_lucro(I, R):-
    retract(lucro(X)),
    X1 is X + (R-I),
    assert(lucro(X1)),
    !.
```