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

Inteligência Artificial 2015/2016

Época Especial, 2016/07/11

Lê cuidadosamente as instruções desta prova feita em moldes não habituais.

Para garantires uma classificação de 12 valores, tens de responder acertadamente às três perguntas do grupo 1, para o que dispões de 15 minutos. Poderás passar com uma nota inferior a 12, desde que respondas quase certo a todas as perguntas.

Para teres uma classificação de 0 a 20, além de responder acertadamente às perguntas do grupo 1, tens de fazer a pergunta do grupo 2 (a qual só conta se a pergunta do grupo 1 estiver completamente certa), para o que dispões de 15 minutos adicionais.

Grupo 1 – Para 12 Valores (15 Minutos)

1 – Qual é o resultado da aplicação do primeiro passo da conversão para forma clausal da seguinte proposição?

$$\neg\forall x\,[P(x) \Rightarrow \exists y(Q(x, y) \vee R(x, y))]$$

Inicia a tua resposta indicando a transformação geral operada pelo primeiro passo da conversão.

R:

Primeiro passo da conversão para forma clausal: Substituir ($A \Rightarrow B$) por ($\neg A \vee B$)

$$\neg\forall x\,[\neg P(x) \vee \exists y(Q(x, y) \vee R(x, y))]$$

2 – Imagina que tens factos com informação sobre os docentes dos alunos nas suas cadeiras, por exemplo

```
profs(inês, [(ia, isabel), (redes1, rui), (redes1, paulo)]).
profs(daria, [(redes3, andre), (ia, luis), (redes3, joao)]).
```

Sem definir nenhum predicado, escreve a interrogação que escreverias na linha de comando do interpretador de *Prolog* para perguntar qual é ou quais são os professores de Redes 3 da Dária

R:

```
?- profs(daria, Profs), member((redes3, P), Profs).
```

3 – O malfeitor, estando acercado da vítima, pergunta "*A bolsa ou a vida?*", empunhando uma pistola. Se a vítima responder "*A bolsa*", o malfeitor rouba a carteira da vítima e dá-lhe um sopapo. Se a vítima responder "*A vida*", o malfeitor dá-lhe um sopapo, rouba-lhe a carteira e mata-a que nem um cão.

O seguinte programa usa a ferramenta *PSys* para controlar o malfeitor exatamente conforme descrito

```
executa_malfeitor(Malfeitor, Vitima):-
      assert(malfeitor(Malfeitor)),
      assert(vitima(Vitima)),
      psys.
```

As duas regras que se seguem determinam parte do comportamento do malfeitor.

```
if   (malfeitor(M) and vitima(V) and resposta(bolsa))
then (roubar_carteira(M, V),
      dar_sopapo(M, V),
      retract(malfeitor(_)),
      retract(vitima(_)),
      retract(resposta(_))).

if   (malfeitor(M) and vitima(V) and resposta(vida))
then (dar_sopapo(M, V),
      roubar_carteira(M, V),
      matar(M, V),
      retract(malfeitor(_)),
      retract(vitima(_)),
      retract(resposta(_))).
```

Escreve a regra que falta. Para isso, imagina que dispões da ação perguntar(X, Y) com a seguinte descrição:

perguntar(X, Y)	X pergunta a Y: *"A bolsa ou a vida?"*, empunhando uma pistola apontada a Y. X responde à pergunta. Em resultado, surge o facto resposta(Resposta), por exemplo resposta(bolsa).

R:

```
if   (malfeitor(M) and vitima(V) and \+ resposta(_))
then perguntar(M, V).
```

Grupo 2 – Para uma classificação arbitrária (15 Minutos)

Escreve um predicado Prolog para contar a frequência relativa dos elementos de uma lista. Exemplo:

```
?- relative_frequences([a, b, c, c, a, a], Freqs).
Freqs = [(a, 0.5), (c, 0.333), (b, 0.1666)]
```

Não podes usar o predicado *length/2*, já definido na linguagem.

NOTAS:

1) A ordem pela qual os elementos da lista de saída estão organizados é irrelevante neste enunciado.

2) Se necessitares, podes usar o predicado *select/3* sem o implementar. select(X, L1, L2): L2 é a lista formada pelos elementos de L1 exceto uma das ocorrências de X. Se X não pertencer a L1, o predicado falha. Exemplos:

```
?- select(f, [a, b, c, a], L).
false
?- select(a, [a, b, c, a], L).
L = [b, c, a];
L = [a, b, c];
false
?- select(X, [a, b], L).
X = a   L = [b];
X = b   L = [a];
false
```

R:

```
relative_frequences(List, Freqs):-
     counts(List, [], N, Counts),
     relative_frequences(Counts, N, Freqs).

counts([X|Rest], Acc, N1, Counts):-
     select((X, Count), Acc, RemainingAcc),
     !,
     Count1 is Count + 1,
     counts(Rest, [(X, Count1)|RemainingAcc], N, Counts),
     N1 is N + 1.
counts([X|Rest], Acc, N1, Counts):-
     counts(Rest, [(X, 1)|Acc], N, Counts),
     N1 is N + 1.
counts([], Acc, 0, Acc).

relative_frequences([(X, Count)|Counts], N, [(X, Freq)|Freqs]):-
     Freq is Count / N,
     relative_frequences(Counts, N, Freqs).
relative_frequences([], _, []).
```