Laboratório 2 – Sistemas de regras baseadas em lógica vaga

Esta aula de laboratório tem por objetivo o exercício de sistemas baseados em conhecimento cujo método de representação e raciocínio são as regras condição-conclusão, baseadas na lógica vaga, com encadeamento para trás. O `FUZ-is`, disponibilizado no sítio web da cadeira, é a ferramenta usada para este laboratório.

1 Como utilizar o *FUZ-is*

1 – Aceder ao sítio web da cadeira e obter o `FUZ-is`. Desempacotar o ficheiro `.zip` e guardar os ficheiros de código fonte (Prolog) numa diretoria à escolha. A distribuição do FUZ-is contém os seguintes ficheiros:

FUZ-is.doc	Manual
fuz-is.pl	Ficheiro principal. É suposto que este ficheiro carregue os outros
inference.pl	Motor de inferência
methods.pl	Diversos métodos alternativos para os vários passos da inferência
vardef.pl	Definição de uma forma alternativa de especificar variáveis vagas (é melhor não usar)

Para que o `FUZ-is` possa funcionar, tem que se editar o ficheiro `fuz-is.pl` (com o *Prolog* ou com um editor de texto qualquer, por exemplo, o *Notepad*)

O ficheiro `fuz-is.pl` contém o seguinte facto, o qual é necessário alterar:

```
fuzis_home_dir('C:/Users/Luis/Dropbox/Systems/FUZ-is/').
```

Modificar o *pathname* da DIRETORIA que contém os ficheiros da distribuição do FUZ-is, de modo a indicar a diretoria onde o sistema foi arrumado, por exemplo.

```
fuzis_home_dir('Z:/Ana/FerramentasIA/FUZ-is/').
```

(terminado com a barra, /)

2 – O ficheiro com a base de conhecimento tem de ter a seguinte diretiva para se poder usar o `FUZ-is`:

```
:- ensure_loaded(pathname do FUZ-IS).
```

Em que o *pathname do FUZ-IS* é o caminho completo do ficheiro de código fonte Prolog do `FUZ-is`, por exemplo

```
'Z:/Ana/FerramentasIA/FUZ-is/FUZ-is.pl'
```

Quando o caminho do `FUZ-is` estiver corretamente especificado, ele aparecerá com a cor azul.

3 – Após a importação do `FUZ-is`, podem escrever-se as regras e os factos, e pode configurar-se o sistema.

2 Exemplo

2.1 Enunciado

Criar e testar um sistema baseado em conhecimento com regras baseadas em lógica vaga, cuja base de conhecimento capte o seguinte:

> *Se a impressão da entrevista for boa e o curriculum for bom então a avaliação é boa.*
>
> *Se a impressão da entrevista for media e o curriculum for médio então a avaliação é média.*
>
> *Se a impressão da entrevista for má e o curriculum for mau então a avaliação é má.*

Para representar este conhecimento, usam-se as variáveis entrevista (para representar a impressão que o candidato produziu na entrevista), cv (para representar a qualidade do *curriculum vitae* do candidato), e avaliação (para representar a avaliação do candidato. A representação das regras exemplificadas recorre a essas variáveis, as quais têm também de ser definidas. Além da representação das regras e variáveis, é necessário configurar o sistema.

2.2 Resolução

```
:- ensure_loaded('C:/Users/Luis/Dropbox/Systems/Fuz-is/Fuz-is.pl').


% Configuração
% Conjunções, disjunções e negações
%
conjunctionTruthMethod(minMethod).

disjunctionTruthMethod(maxMethod).

negationTruthMethod(complement).


% Conclusion truth value
%
condition2conclusionPropagationMethod(shrink).


% Else link
%
else_linkMethod(probor_link).



% Regras
%
if  ((entrevista is boa) and (cv is bom)) then (avaliacao is boa).
```

```
if  ((entrevista is media) and (cv is medio))
then (avaliacao is media).
if  ((entrevista is ma) and (cv is mau)) then (avaliacao is ma).



% Variáveis vagas
%
% entrevista
%
variableUniverse(entrevista, [1, 2, 3, 4, 5]).

variableValueDef(entrevista, boa, [1/0, 2/0, 3/0.3, 4/0.8, 5/1]).
variableValueDef(entrevista, media, [1/0, 2/0.5, 3/1, 4/0.5, 5/0]).
variableValueDef(entrevista, ma, [1/1, 2/0.8, 3/0.3, 4/0, 5/0]).



% CV
%
variableUniverse(cv, [1, 2, 3, 4, 5]).

variableValueDef(cv, bom, [1/0, 2/0, 3/0.3, 4/0.8, 5/1]).
variableValueDef(cv, medio, [1/0, 2/0.5, 3/1, 4/0.5, 5/0]).
variableValueDef(cv, mau, [1/1, 2/0.8, 3/0.3, 4/0, 5/0]).



% avaliacao
%
variableUniverse(avaliacao, [1, 2, 3, 4, 5]).

variableValueDef(avaliacao, boa, [1/0, 2/0, 3/0.3, 4/0.8, 5/1]).
variableValueDef(avaliacao, media, [1/0, 2/0.5, 3/1, 4/0.5, 5/0]).
variableValueDef(avaliacao, ma, [1/1, 2/0.8, 3/0.3, 4/0, 5/0]).

variableDefuzMethod(avaliacao, centroid).
```

Interação com o sistema

Na janela de consola do interpretador de Prolog, usa-se o predicado fuzis/3 para interagir com o sistema

```
?- fuzis([cv = 3, entrevista = 2], avaliacao, Aval).
Aval = 2.4885245901639346
```

Alternativamente, pode definir-se um predicado de interface, no ficheiro com o código do sistema, por exemplo

```
aval(CV, Entrevista, Aval):-
      fuzis([cv = CV, entrevista = Entrevista], avaliacao, Aval).
```

Depois, no interpretador, já se pode usar o predicado definido:

```
?- aval(3, 2, Aval).
Aval = 2.4885245901639346
```

3 Exercícios

1. Usando o *FUZ-is*, implementa o sistema baseado em conhecimento para determinar a gratificação a atribuir a um empregado de mesa em função da qualidade da comida e da qualidade do serviço.

Usa as seguintes regras:

Se a comida é fraca ou o serviço é mau então a gratificação é baixa.

Se a comida é boa e o serviço é bom então a gratificação é elevada.

Se a comida é boa e o serviço é médio então a gratificação é média.

Se a comida é média e o serviço é bom então a gratificação é média.

Testa várias configurações diferentes e tenta descortinar qual a que melhor corresponde às tuas expectativas. Verifica se essa configuração é igualmente a melhor para problemas futuros.

   a) O sistema deve ter, como interface, o predicado Prolog *gorjeta/3*, tal que gorjeta(Servico, Comida, Gorjeta) significa que *Gorjeta* é o valor da gratificação a dar ao empregado quando a qualidade do serviço é *Servico* e a qualidade da comida é *Comida*.
   b) O sistema deve ter, como interface, o predicado Prolog *gorjeta/0*, o qual pede ao utilizador os valores da qualidade da comida e da qualidade do serviço, determina o valor da gorjeta recorrendo ao sistema baseado em conhecimento, e imprime esse valor no ecrã do computador.

2. Usando o predicado *gorjeta/3* da alínea a) do exercício 1, faz um programa que imprime, linha a linha, uma tabela com todas as combinações dos valores possíveis da qualidade de comida e da qualidade do serviço (combinações de valores dos respectivos universos de discurso) e a respectiva gratificação.