Escola de Engenharia
Departamento de Informática
Licenciatura em Engenharia Informática

Sistemas de Representação de Conhecimento e Raciocínio

"Sistemas Multi-Agente Baseados em Quadros Negros"

A61021 Mário Leite
A61031 Diana Lemos
A61041 Mariana Medeiros
A61049 Miguel Pinto

Grupo 2

Braga, Maio de 2014

Resumo

Com este trabalho pretende-se desenvolver agentes inteligentes e sistemas multi-agente para a resolução de problemas, utilizando mecanismos de representação de conhecimento e de raciocínio de base hierárquica, aproveitando a herança como mecanismo de raciocínio por defeito.

Foi desenvolvido um sistema de computação distribuída baseado em quadros negros, como mecanismo de troca de informação entre os vários agentes. Para tal, foram utilizadas as bibliotecas LINDA do SICStus Prolog.

Foi também necessário a construção de uma base de conhecimento para cada agente inteligente, assim como a definição do modo de distribuição da computação e da linguagem a ser usada para a comunicação.

Tabela de Conteúdos

Tabela de figuras

1. Introdução

Foi proposto aos alunos inscritos na Unidade Curricular de Sistemas de Representação de Conhecimento e Raciocínio no presente ano letivo 2013/2014 que desenvolvessem um sistema de representação de conhecimento e raciocínio referente à representação do conhecimento segundo um modelo hierárquico com herança, recorrendo a sistemas multi-agente. A resolução do trabalho prático aqui descrita refere-se a essa mesma representação de conhecimento.

Assim, neste trabalho é desenvolvida uma base de conhecimento para todas as entidades presentes no sistema e também desenvolvida uma linguagem adequada para a comumicação entre as mesmas, linguagem esta desenvolvida em PROLOG.

Todo o sistema desenvolvido será abordado em termos do desenvolvimento de um sistema de computação distribuída, baseado no modelo de quadros negros, disponibilizado pelas bibliotecas LINDA do SICStus Prolog.

Neste relatório será efetuada uma descrição e análise de resultados, bem como apresentadas as conclusões obtidas após a realização do trabalho prático.

2. Descrição do trabalho

O sistema em estudo diz respeito a um sistema hierárquico de representação de conhecimento que tem como objectivo disponibilizar formas de raciocínio por defeito, possibilitar o tratamento de informação incompleta e permitir a aplicação de mecanismos de herança. A herança permite que as propriedades e relações de qualquer classe possam ser herdadas por todas as suas subclasses.

A representação da informação é realizada de uma forma estruturada, baseada na implementação de um grafo, em que os arcos representam o relacionamento entre as entidades estabelecendo a ligação entre dois agentes da mesma. No que diz respeito aos nodos do grafo, estes dizem respeito às entidades da estrutura. A cada entidade é associado um conjunto de propriedades.

2.1. Estrutura Hierárquica

Esta representação da informação com vários agentes trata-se de um sistema de computação distribuída. Este tipo de sistema permite a comunicação entre as várias entidades, através da implementação das bibliotecas LINDA do SICStus prolog.

Cada agente é definido num ficheiro e cada um deles contém toda a informação das suas caraterísticas. Para se manter a relação entre os agentes, cada um deles tem definido o predicado *e_um*. Este predicado indica qual é a classe superior do agente em questão.

Caso um agente não consiga responder a uma dada questão, a questão é reencaminhada para os superiores (SuperClasse) do agente em causa, caso estes não tenham informação suficiente dão uma resposta de *não*.

2.2. Herança Múltipla

A herança é um fato importante no sistema hierárquico. Contudo, existem situações mais complicadas do que uma simples herança, designadas por herança múltipla.

O problema da herança múltipla surge quando uma subclasse, em particular, está relacionada com mais do que uma superclasse, das quais pode herdar propriedades que entrem em conflito. Como exemplo, neste sistema é possível encontrar o agente Batman que herda propriedades de dois agentes superiores. Neste caso, o batman herda a propriedade ’revestimento’ da ave, completamente oposta à propriedade ’revestimento’ que herda do mamífero, não fazendo sentido o batman possuir dois tipos diferentes de revestimento.

3. Rede Semântica Hierárquica

Uma rede semântica hierárquica pode ser vista como uma estrutura de organização de tipos, agrupados por níveis de generalidade. No topo da hierarquia os conceitos são mais genéricos e na base estão representados elementos individuais ou instâncias.

A figura seguinte trata-se de uma representação gráfica de um sistema de conhecimento hierárquico:

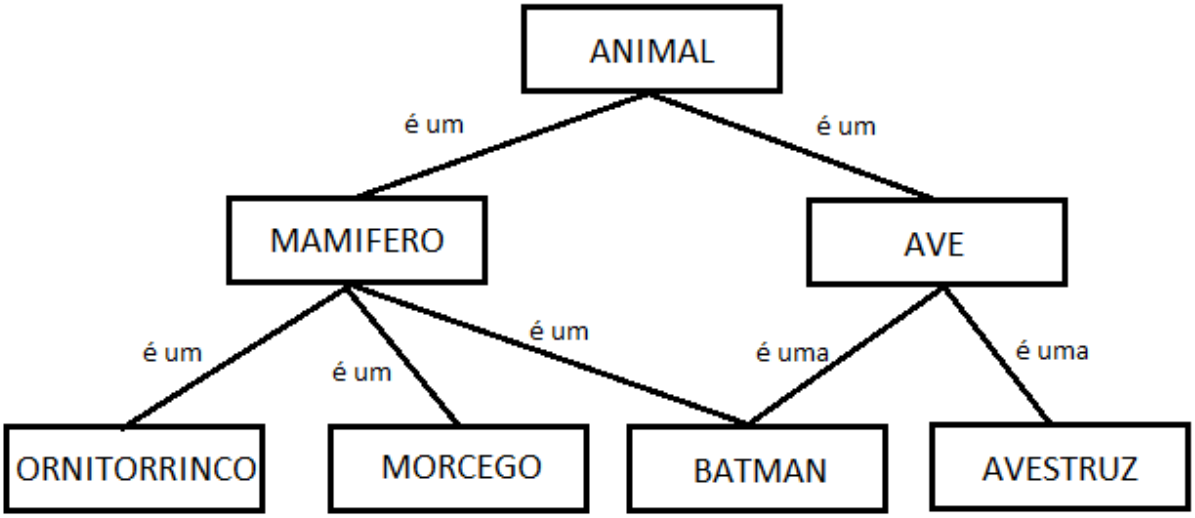

Figura 1 - Rede semântica.

4. Representação da base de conhecimento

O sistema hierárquico apresentado é composto por 7 agentes onde cada um possui as suas próprias características. Assim, cada agente deve saber responder às questões que lhe dizem respeito diretamente. Caso não saiba a resposta a um predicado deve solicitar ao agente hierarquicamente superior a ele, se existir a resposta, assim definido no agente, se não existir nesta situação reporta que está indefinido o predicado.

As propriedades existentes são relativas aos animais. Desta forma, foram definidos para os agentes predicados que definem o tipo de revestimento, cor, locomoção entre outras existentes no sistema e explicadas a seguir.

- **Animal**

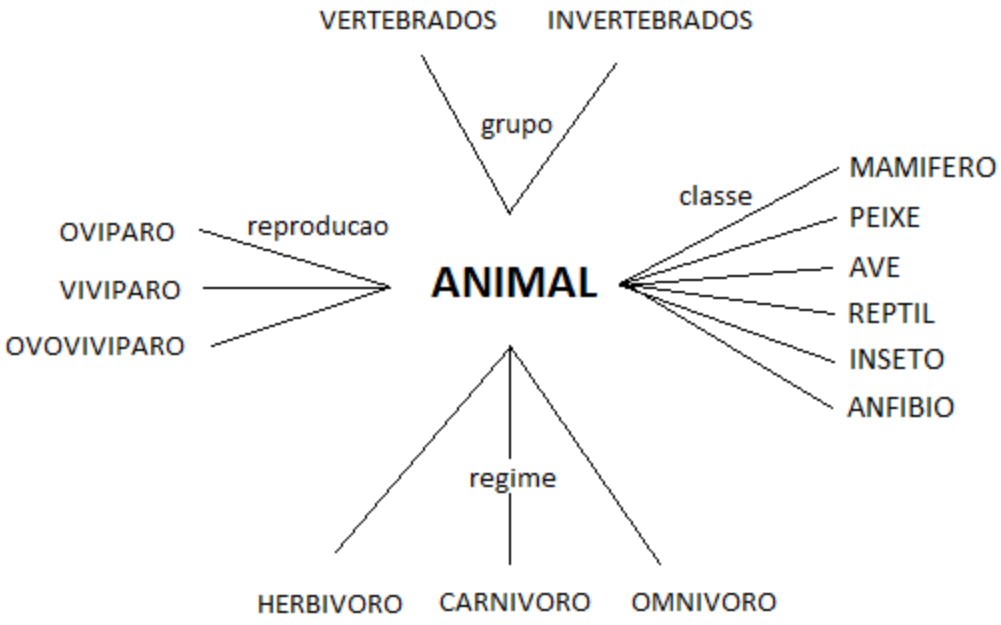

Figura 2 - Propriedades do animal.

- **Mamífero**

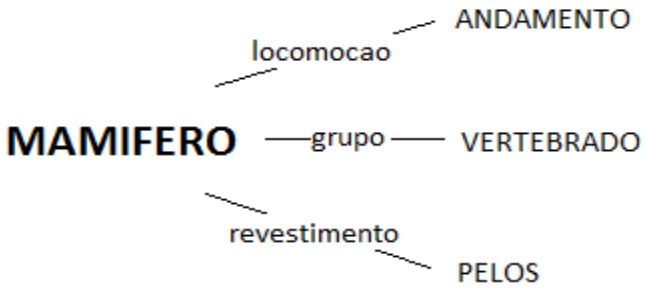

Figura 3 - Propriedades do mamifero.

- **Ave**

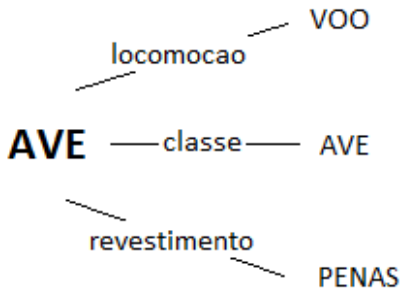

Figura 4 - Propriedades da ave.

- **Batman**

Figura 5 - Propriedades do batman.

- **Avestruz**

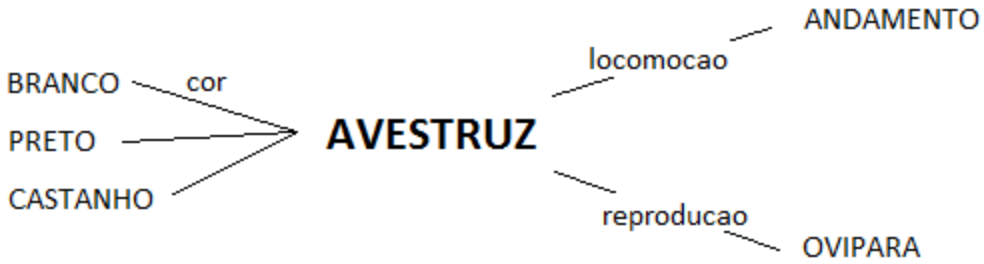

Figura 6 - Propriedades da avestruz.

- **Morcego**

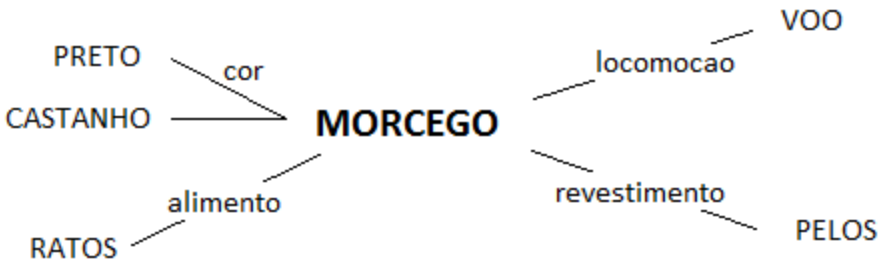

Figura 7 - Propriedades do morcego.

- **Ornitorrinco**

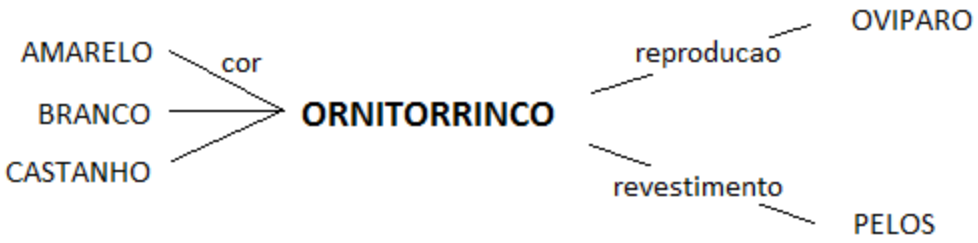

Figura 8 - Propriedades do ornitorrinco.

- **Predicado e_um**

De forma a ter acesso às carateristicas do agente que se encontra no nivel acima do agente em questão, existe este predicado.

Este predicado recebe como argumentos, o nome do próprio agente e o nome da classe hierarquicamente superior a ele.

- e_um: SubClasse,Classe -> {V,F}

Exemplo: *e_um(avestruz,ave).*

- **Evolução da base de conhecimento**

A base de conhecimento permite a inserção de conhecimento, isto é, é possível fazer inserções de novos predicados. O predicado *evolucao* recebe como argumento o termo a ser inserido.

```
evolucao( Termo ) :-
        solucoes( Invariante,+Termo::Invariante,Lista ),
        insercao( Termo ),
        teste( Lista ).

solucoes(X,Y,Z):- findall(X, Y, Z).

insercao(Termo):- assert(Termo).
insercao(Termo):- retract(Termo), !, fail.

teste([]).
teste([R|LR]):- R, teste(LR).
```

- **Remoção na base de conhecimento**

Para possibilitar a remoção de predicados já existentes na base de conhecimento, existe o predicado *remocao*.

```
remocao( Termo ) :-
        solucoes( Invariante,-Termo::Invariante,Lista ),
        teste( Lista ),
        remover( Termo ).

remover( Termo ) :- retract( Termo ).
```

- **Extensão do meta-predicado não**

  nao(X) :- X,!,fail.
  nao(X).

- **Extensão do predicado comprimento**

  comprimento([], 0) .
  comprimento([H|T], R) :- comprimento(T, X), R is 1+X .

5. Conhecimento positivo e negativo

O PROLOG não permite chegar a conclusões negativas, daí ser necessário declarar explicitamente informações falsas. Para tal, na base de conhecimento foi incluído conhecimento negativo.

Este tipo de conhecimento é caracterizado pelo caracter '-', que é colocado antes da informação que se pretende que seja negativa.

- **Conhecimento Negativo**

  Exemplo:
  -alimento(A) :- nao(alimento(A)), nao(excecao(alimento(A))).

6. Linguagem de Comunicação

Foi necessário definir um protocolo de comunicação entre os vários agentes existentes no sistema para que possam comunicar uns com os outros e responder a perguntas efetuadas pela interface. Assim, como modelo de comunicação temos o seguinte:

- **Predicado demo [*demo: Agente,Questao -> {V,F,D}*]**

As perguntas são efetuadas através da inserção de um meta-predicado que contém dois argumentos.

```
demo(Agente,Questao):-
        Agente::Questao,
        write((1,Agente::Questao)),nl,
        out(prova(Agente,Questao)).

demo(Agente,Questao):-
        e_um(Agente,Classe),
        write((2,e_um(Agente,Classe))),nl,
        out(demo(Classe,Questao)).

demo(Agente,Questao) :-
        write((3,nao)),nl,
        out(prova(Agente,nao)).
```

- **Predicado prova [*prova: Agente,Resposta -> {V,F,D}*]**

As respostas quando são encontradas devem ser da definidas pela forma anterior, em que o segundo argumento contém a resposta.

7. Conhecimento imperfeito

Quando é utilizado o predicado *demo* e quando não existem provas na base de conhecimento, o valor é *desconhecido*.

8. Comunicação e quadros negros

A comunicação através dos quadros negros é efectuada através das bibliotecas LINDA. Para existir comunicação, é necessário existirem, pelo menos, três processos a decorrer, um suportado por uma estrutura tipica de um quadro negro e outros dois, que comunicam por essa estrutura. No entanto, é obrigatório que um desses processos seja do tipo server, responsável pela transferência de informação.

Os outros processos dizem respeito aos sete agentes do sistema. Para iniciar a comunicação estes precisam de se ligar ao servidor.

Para a troca de mensagens é utilizada uma interface feita em Java com a utilização da biblioteca Jasper.

9. Interface

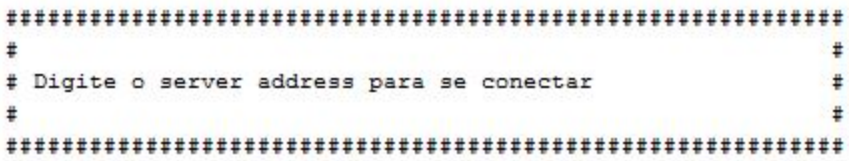

Figura 9 - Ligar server.

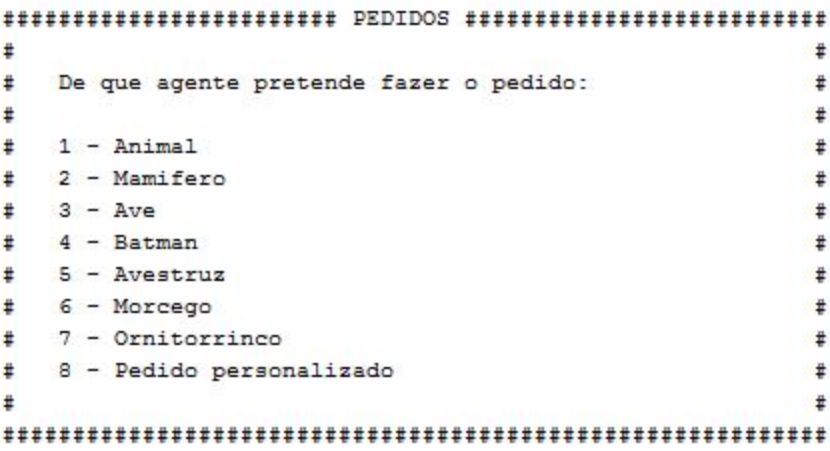

Figura 10 - Menu.

```
####################### Mamifero ############################
#                                                           #
#   Qual a questão:                                         #
#                                                           #
#   1 - Locomoção                                           #
#   2 - Grupo                                               #
#   3 - Revestimento                                        #
#                                                           #
#############################################################
3
{RESPOSTA=revestimento(pelos), X=_331}
```

Figura 11 - Pedido mamifero revestimento.

```
######################### PEDIDOS ###########################
#                                                           #
#   De que agente pretende fazer o pedido:                  #
#                                                           #
#   1 - Animal                                              #
#   2 - Mamifero                                            #
#   3 - Ave                                                 #
#   4 - Batman                                              #
#   5 - Avestruz                                            #
#   6 - Morcego                                             #
#   7 - Ornitorrinco                                        #
#   8 - Pedido personalizado                                #
#                                                           #
#############################################################
8

################ Pedido Personalizado ######################
#                                                           #
#   Efetue o pedido:                                        #
#                                                           #
#############################################################
p(mamifero,cor(X),RESPOSTA).
{RESPOSTA=indefinido, X=_608}
```

Figura 12 - Pedido personalizado.

10. Conclusões

Em modo de conclusão, vemos este projeto como mais uma forma de pôr em prática e consolidar os conhecimentos adquiridos e as ferramentas disponibilizadas na Unidade Curricular de Sistemas de Representação de Conhecimento e Raciocínio .

O projeto foi desenvolvido de maneira a conseguir dar resposta a tudo o que nos foi solicitado.

A principal dificuldade com que o grupo se deparou foi elaborar um caso prático que tornasse possível respeitar as necessidades de demonstração de todas as funcionalidades pedidas.