Universidade do Minho

Escola de Engenharia

Licenciatura em Engenharia Informática

Unidade Curricular de Sistemas de Representação de Conhecimento e Raciocínio

Ano Letivo de 2014/2015

3º Ano 2º Semestre

Trabalho de grupo – 1º Exercício

Daniel Caldas a67691

José Cortez a67716

Susana Mendes a63464

Xavier Rodrigues a67676

Março, 2015

Trabalho de grupo – 1º Exercício

Daniel Caldas a67691
José Cortez a67716
Susana Mendes a63464
Xavier Rodrigues a67676

Março, 2015

Resumo

Com o presente relatório pretendemos expor sucintamente as decisões tomadas ao longo da criação do nosso sistema de representação de conhecimento e raciocínio. Iremos apresentar o caso de estudo seguido de uma pequena análise do problema. Posteriormente vamos expor a solução desenvolvida com o detalhe necessário. Posteriormente, será feita uma análise dos resultados e por fim apresentaremos as conclusões.

Área de Aplicação: Inteligência Artificial.

Índice

1 Introdução

1.1. Apresentação do Caso de Estudo

O sistema de representação de conhecimento e raciocínio deverá ser capaz de identificar/caracterizar relacionamentos (de parentesco) numa dada base de conhecimento, mais concretamente, uma árvore genealógica. Para além das relações familiares, é suposto que o mesmo sistema desenvolvido seja capaz de identificar o sexo, naturalidade, data de nascimento e morte de um dado indivíduo.

1.2. Motivação e Objetivos

Os conceitos teóricos da programação em lógica são difíceis de assimilar numa primeira fase, contudo a prática e a resolução de problemas como o proposto levam à melhor compreensão desses conceitos, como tal é estabelecido o objetivo de desenvolver um sistema de representação de conhecimento e raciocínio com as características descritas na apresentação do caso de estudo.

1.3. Estrutura do Relatório

O presente relatório encontra-se estruturado conforme os seguintes pontos:

- Análise do problema;
- Apresentação da solução desenvolvida:
    - Conhecimento representado na base (de conhecimento);
    - Predicados implementados;
    - Invariantes estruturais e referenciais;
- Conhecimento fornecido à base por asserções iniciais;
- Apresentação de resultados;
- Teste concretos;
- Análise dos resultados;
- Conclusões;

2. Preliminares

2.1 Estudo prévio

Na sequência do estudo da programação em lógica feito nas aulas da UC SRCR, estudamos regras e técnicas para construir simples mecanismos (programas) de inferência lógica baseando-nos nos pressupostos do domínio fechado e mundo fechado, nos quais um sistema apenas “raciocina” sobre factos que lhe são conhecidos, isto é, “se não está dito é mentira”.

Será portanto nestes moldes que será desenvolvida uma proposta de solução para o problema.

2.2 Análise do problema

Após uma análise do enunciado, foi elaborada uma pequena lista de predicados que permitirão ao sistema fazer deduções válidas do ponto de vista lógico (sempre em função dos dados que são fornecidos à base) no universo de discurso do problema.
Esses predicados são relativos a conexões familiares: **filho, pai, mãe, tio, sobrinho, primo e irmão.**

No que diz respeito ao sexo de cada indivíduo estendemos o predicado **sexo**.
Para que no sistema se possam reconhecer datas de nascimento e falecimento assim como naturalidade de um dado indivíduo definimos um só predicado **id**.

Outro problema que surge com frequência é a repetição de conhecimento na base de conhecimento, para tal foram implementados invariantes estruturais e referenciais que acrescentam a consistência necessária à nossa base de conhecimento. Portanto na nossa solução definimos os invariantes para que possamos afirmar:

- Qualquer indivíduo tem apenas um ou dois progenitores;
- Qualquer indivíduo tem apenas um sexo;
- Qualquer indivíduo tem apenas uma naturalidade, data de nascimento e morte;
- Dois indivíduos são irmãos se e só se são descendentes diretos do mesmo par de progenitores.

3 Apresentação da solução

3.1 Conhecimento representado

Por uma questão de simplicidade na abordagem ao problema, bem como para garantir uma base mais robusta e consistente, o grupo decidiu que seriam apenas feitas inserções de relações familiares pelo predicado filho, desta forma somos também capazes de testar a eficiência dos predicados que implementamos, pois nunca estão na base disponíveis factos diretos de relações familiares, passando a exemplificar:

filho(joao,carlos) .
filho(joao,ana).
irmao (joao,andre).

Como foi anteriormente descrito seria direta a conclusão de que o João e o André são irmãos enquanto, que em vez de declarar o André como irmão do João, seria declarado da seguinte forma:

filho(andre,carlos).
filho(andre,ana).

Desta forma teremos de verificar se o André é irmão do João através dos progenitores de cada um que neste caso em concreto seriam os mesmos.
Portanto no início do ficheiro com o nosso código em *PROLOG* veremos apenas estabelecimento de relações familiares pelo predicado filho (para além das de sexo e identidade).[1]

[1]O facto de termos utilizado este método para representar conhecimento, não significa que não se possam inserir **relações familiares independentes**, estão definidos invariantes estruturais e referenciais que permitem isso mesmo.

3.2 Predicados

Foram implementados predicados para identificação de relações familiares, naturalidade e identidade (entenda-se identidade como data de nascimento e morte).

3.2.1 Naturalidade e identidade

Para que a um dado indivíduo se possa atribuir uma naturalidade, data de nascimento e morte foram definidos dois predicados:

***naturalidade* (X,Y)** : em que X é o nome de um dado indivíduo e Y a sua naturalidade.
***id* (X,AN,AM)** : em que X é o indivíduo que nasceu no ano AN e faleceu no ano AM.

Foram também acrescentados à base de conhecimento os respetivos invariantes como iremos analisar com mais detalhe posteriormente, neste relatório.

```
naturalidade(erico,copenhaga).
naturalidade(ana,guimaraes).
naturalidade(joao,guimaraes).
```

Figura 1 - Exemplo de inserção de naturalidade.

```
id(erico,1216,1250).
id(ana,1200,1245).
id(joao,1216,1260).
```

Figura 2 - Exemplo de inserção de identidade.

3.2.2 Pai e Mãe

```
%------------------------------ - - - - - - - - - -  -  -  -  -  -
% Extensao do predicado pai: Pai,Filho -> {V,F}

pai( P,F ) :- filho( F,P ), sexo( P,masculino ).


%------------------------------ - - - - - - - - - -  -  -  -  -  -
% Extensao do predicado mae: mae,Filho -> {V,F}

mae( M,F ) :- filho( F,M ), sexo( M,feminino ).
```

Figura 3 - Extensão dos predicados *pai* e *mãe*.

F – Um dado indivíduo acerca do qual se quer inferir sobre os respetivos progenitores.
P – Indivíduo que pode ou não ser pai de F.
M – Indivíduo que pode ou não ser mãe de F.

Estes predicados, como podemos observar na figura 1 para além de verificarem se um determinado indivíduo P ou M é progenitor de um indivíduo F, é verificado também o sexo dos progenitores que são testados pelo predicado.

3.2.3 Irmão

```
%------------------------------ - - - - - - - - - -  -  -  -  -   -
% Extensao do predicado irmao: Irmao1,Irmao2 -> {V,F}

irmao( I1,I2 ) :- pai( X,I1 ), pai( X,I2 ), mae( Y,I1), mae( Y,I2 ), I1\=I2 .


%------------------------------ - - - - - - - - - -  -  -  -  -   -
% Extensao do predicado irmaos que determina a lista dos irmaos de um dado indivíduo: I,L -> {V,F}

irmaos( I,L ) :- solucoes( X,irmao(I,X), L).
```

Figura 4 - Extensões dos predicados *irmao* e *irmaos*.

No predicado *irmao*:
I1 – Indivíduo sobre o qual se quer verificar relação de parentesco irmão.
I2 – Indivíduo sobre o qual se quer verificar relação de parentesco irmão com o indivíduo I1.
X – Pai do indivíduo I1 (terá de ser o mesmo que o pai do indivíduo I2).
Y – Mãe do indivíduo I1 (terá de ser a mesmo que a mãe do indivíduo I2).

No predicado *irmaos*:
I – Indivíduo.
L – Lista dos irmãos do indivíduo I.

No primeiro predicado da imagem, podemos ver como dois indivíduos são considerados irmãos pelo sistema**. I1 e I2 serão irmãos se e só se são filhos do mesmo par de progenitores** e são diferentes, i.e eliminar a possibilidade de que a base considerasse verdadeiro irmao(I1,I1) ou irmao(I2,I2).

O segundo predicado diz-nos, para um dado indivíduo, quais os seus irmãos. Para tal apenas foi explorado o predicado irmão e o predicado que procure todas as soluções para uma determinada questão, obtemos a lista L dos irmãos do indivíduo I.

Esta técnica foi aplicada em todos os predicados de relação familiar pelo que, de forma a não constar demasiada informação repetida no relatório, fica aqui **uma explicação que serve como exemplo para as restantes**.

3.2.4 Tio

```
%------------------------------ - - - - - - - - - -  -  -  -  -  -
% Extensao do predicado tio que testa se um indivíduo T é tio (ou tia) de um indivíduo S: Tio,Lista -> {V,F}

tio( T,S ) :- pai( X,S ), irmao( T,X ).
tio( T,S ) :- mae( X,S ), irmao( T,X ).


%------------------------------ - - - - - - - - - -  -  -  -  -  -
% Extensao do predicado tios que testa se um indivíduo T é tios (e tias) de um indivíduo S: Tio,Lista -> {V,F}

tios( I,TS ) :- solucoes( T,tio(T,I), TS).
```

Figura 5 - Extensão dos predicados tio e tios.

No predicado tio:

T – Indivíduo sobre o qual se quer verificar relação de parentesco tio.

S – Indivíduo sobre o qual se quer verificar relação de parentesco tio com o indivíduo T.

X – Vai ser o Pai/Mãe do indivíduo S para posteriormente testar se este individuo é irmão de T.

No predicado tios:

I – Indivíduos.

TS – Lista de todos os sobrinhos do Individuo I.

T será considerado tio de S e I2 se e só se S é filho de um Pai ou Mãe X, cujo X é um irmão T.

O segundo predicado diz-nos para um dado indivíduo quais os seus tios. Para tal apenas foi explorado o predicado tio, e usando o predicado que procure todas as soluções (função *findall* já definida na biblioteca do PROLOG) para uma determinada questão, obtemos a lista L dos irmãos do indivíduo I.

3.2.5 Sobrinho

```
%------------------------------ - - - - - - - - - -  -  -  -  -  -
% Extensao do predicado sobrinho: Sobrinho,Tio -> {V,F}
sobrinho( S,T ) :- tio( T,S ).


%------------------------------ - - - - - - - - - -  -  -  -  -  -
% Extensao do predicado sobrinhos: indivíduo,Sobrinho -> {V,F}
sobrinhos( I,SS ):- solucoes( S,sobrinho(S,I),SS ).
```

Figura 6 - Extensão dos predicados *sobrinho e sobrinhos*.

No predicado Sobrinho:

S – Individuo sobre o qual se vai testar a relação de parentesco sobrinho.

T - Indivíduo sobre o qual se quer verificar a relação de parentesco sobrinho com o indivíduo S.

No predicado Sobrinhos:

S – Individuo.

SS – Lista de todos os tios.

Este predicado é muitos simples de explicar pois demos uso a um predicado já definido, o predicado tio.

S só será sobrinho de T se e só se T for tio (ou tia) de S.

Quanto ao predicado sobrinhos, usa o mesmo raciocínio já apresentado**.**

3.2.6 Todos os descendentes até grau N

De modo a determinar os descendentes de uma pessoa até grau N, foram criados os seguintes predicados:

1 – Predicado que calcula o grau entre duas pessoas

```
% Extensao do predicado grau que determina o grau de parentesco entre dois indivíduos: D,A,G -> {V,F}

grau( D,A,1 ) :- filho( D,A ).
grau( D,A,N ) :- filho( D,X ) , grau( X,A,R ), N is R+1.
```

Figura 7 - Predicado Grau

Onde:

D – Pessoa sobre a qual se quer fazer a questão

A – Antecessor de D

N – Grau entre D e A.

2 – Predicado que calcula todos os descendentes de um dado grau

```
% Extensao do predicado ngrau que determina todos os familiares de um determinado grau de um indivíduo: L1,L2,LF -> {V,F}

ngrau(I,0,[]).
ngrau( I,N,L) :- findall( (X),grau(I,X,N),L ).
```

Figura 8 – Predicado ngrau

Onde:

I - Indivíduo sobre o qual se pretende obter todos os descendentes de um dado grau

N – Grau pretendido

L – Lista final na qual serão mostrados os resultados

Finalmente, com o auxílio dos dois predicados anteriores, é possível determinar todos os descendentes de uma dada pessoa até grau N:

```
% Extensao do predicado graus que determina todos os familiares anteriores (ou do mesmo) dum indivído até um
% dado grau: I,G,L -> {V,F}

graus(I,0,[]).
graus(I,0,L).
graus(I,N,L) :- concatenar(L1,L2,L), ngrau(I,N,L1), M is N-1, graus(I,M,L2).
```

Figura 9 – Predicado graus

Onde:

I – Indivíduo sobre o qual se pretende obter os descendentes até determinado grau

N – Grau de descendência

L – Lista que irá conter os resultados finais

3.2.7 Relação familiar

A abordagem feita foi a seguinte:

Foram considerados todas as relações familiares e, para cada uma delas, foi criado um predicado com o respetivo resultado:

```
% Extensao do predicado relacao que determina a relacão familiar entre dois indivíduos: X,Y,R -> {V,F}

relacao( X,Y,'filho' ) :- filho(X,Y).
relacao( X,Y,'pai' ) :- pai(X,Y).
relacao( X,Y,'mae' ) :- mae(X,Y).
relacao( X,Y,'pai' ) :- pai(X,Y).
relacao( X,Y,'tio' ) :- tio(X,Y).
relacao( X,Y,'sobrinho' ) :- sobrinho(X,Y).
relacao( X,Y,'primo' ) :- primo(X,Y).
```

Figura 10 – Predicados 'relacao' para cada relação familiar

Onde:

X será o pai/filho/etc. de Y

3.2.8 Funcionalidade Extra - Idade

```
% Extensao do predicado idade que calcula a idade com que um dado indíviudo da base de conhecimento morreu: Nome,Idade -> {V,F}

idade( N,I ) :- id( N,X,Y ), I is Y-X.
```

Figura 11 – Predicado idade

Trata-se de um predicado que calcula a idade de um indivíduo aquando da sua morte, onde:

N – Indivíduo sobre o qual se pretende saber a idade no momento da sua morte

I – Idade

X – Data de nascimento

Y – Data de morte

3.2.9 Invariantes

Foram definidos invariantes para os seguintes predicados:

- naturalidade (estrutural) – Não permite inserção de conhecimento repetido para o predicado naturalidade.

```
+naturalidade( I,X ) :: (solucoes( (X),(naturalidade( I,X )),S ),
                         comprimento( S,N ),
                         N == 1
                         ).
```

Figura 12 – Invariante estrutural do predicado naturalidade

- naturalidade (Referencial) – Não admite mais do que uma naturalidade para o mesmo indivíduo

```
+naturalidade( I,X ) :: (solucoes( ( I,X ),(naturalidade( I,F )),S ),
        comprimento( S,N ),
        N =< 1
            ).
```

Figura 13 – Invariante referencial do predicado naturalidade

- id (Estrutural) – Não permite inserção de conhecimento repetido para o predicado id.

```
+id( I,X,Z ) :: (solucoes( (X,Z),(id( I,F,D )),S ),
        comprimento( S,N ),
        N =< 1
            ).
```

Figura 14 – Invariante estrutural para o predicado id

-id (Referencial) – Não admite mais do que um par (Data nascimento – Data morte) para o mesmo indivíduo.

```
+id( I,X,Z ) :: (solucoes( (I,X,Z),(id( I,F,D )),S ),
        comprimento( S,N ),
        N == 1
            ).
```

Figura 15 – Invariante referencial para o predicado id

- filho (Estrutural) – Não permite a inserção de conhecimento repetido para o predicado filho.

```
+filho( F,P ) :: (solucoes( (Pais),(filho( F,Pais )),S ),
            comprimento( S,N ), N =< 2
            ).
```

Figura 16 – Invariante estrutural para o predicado filho

- filho (Referencial) – Não admite mais do que dois progenitores para o mesmo indivíduo.

```
+filho( F,P ) :: (solucoes( (Pais),(filho( F,Pais )),S ),
            comprimento( S,N ), N =< 2
            ).
```

Figura 17 – Invariante referencial do predicado filho

- sexo (estrutural) – Não permite a inserção de conhecimento repetido para o predicado sexo.

```
+sexo( I,X ) :: (solucoes( (X),(sexo( I,X )),S ),
                 comprimento( S,N ),
                 N =< 1
                       ).
```

Figura 18 – Invariante estrutural do predicado sexo

- sexo (referencial) – Não admite mais do que um sexo por indivíduo.

```
+sexo( I,X ) :: (solucoes( (I,X),(sexo( I,F )),S ),
                 comprimento( S,N ),
                 N == 1
                       ).
```

Figura 19 – Invariante referencial do predicado sexo.

4 Análise de Resultados

Foram realizados diversos testes através de questões colocadas à base de conhecimento. Expomos aqui algumas dessas questões.

```
msec 157008 bytes
| ?- irmaos(berengaria,L).
L = [raimundo,constanca,afonsoII,pedro,fernando,branca] ?
yes
| ?- irmaos(raimundo,L).
L = [constanca,afonsoII,berengaria,pedro,fernando,branca] ?
yes
| ?- irmaos(constanca,L).
L = [raimundo,afonsoII,berengaria,pedro,fernando,branca] ?
yes
| ?-
```

Figura 20 - Teste do predicado *irmaos*.

Como podemos observar, este predicado, para um dado indivíduo, fornece-nos a lista dos respetivos irmãos.

Para comprovar o resultado, verificamos a árvore genealógica desenvolvida pelo grupo, que se encontra em anexo neste relatório (Anexo 1).

De seguida serão enunciados outros testes relevantes realizados.

```
| ?- tios(sancho,L).
L = [urraca,sancha,afonso] ?
yes
| ?- tios(erico,L).
L = [raimundo,constanca,afonsoII,pedro,fernando,branca] ?
yes
| ?- tios(joao,L).
L = [constanca,afonsoII,berengaria,pedro,fernando,branca] ?
yes
| ?-
```

Figura 21 - Teste do predicado *tios*.

```
| ?- graus(joao,2,L).
L = [sancho,dulce,raimundo,ana] ?
yes
| ?- graus(afonsoII,3,L).
L = [henrique,teresa,afonsoI,mafalda,ramon,petronila,sancho,dulce] ?
yes
| ?- graus(sancho,1,L).
L = [afonsoI,mafalda] ?
yes
```

Figura 22 - Teste do predicado *graus*.

```
filho(constanca, dulce).
filho(afonsoII, sancho).
filho(afonsoII, dulce).
filho(berengaria, sancho).
filho(berengaria, dulce).
filho(pedro, sancho).
filho(pedro, dulce).
filho(fernando, sancho).
filho(fernando, dulce).
filho(branca, sancho).
filho(branca, dulce).
filho(erico, valdemar).
filho(erico, berengaria).
filho(joao, raimundo).
filho(joao, ana).

yes
| ?- evolucao(filho(pedro,sancho)).
no
| ?- evolucao(filho(pedro,manuel)).
no
| ?-
```

Figura 23 - Teste do predicado evolucao.

Com o predicado *evolucao,* podemos testar a consistência da base de conhecimento. Pela figura 10 podemos observar que o indivíduo Pedro já tinha dois progenitores (um pai e uma mãe: Sancho e Dulce, respetivamente), portanto a base rejeitou a inserção de conhecimento repetido na primeira asserção enquanto na segunda rejeitou um novo progenitor para Pedro pois foi definido um invariante referencial por forma a impedir que um dado indivíduo tenha mais do que dois progenitores.

5 Conclusões

Este trabalho serviu como consolidação dos conhecimentos adquiridos até ao momento de programação em lógica segundo os princípios do mundo fechado e domínio fechado, usando como ferramenta de trabalho o PROLOG.

Os objetivos do trabalho foram cumpridos. No entanto, o grupo conclui que poderia ter explorada mais funcionalidades extra no contexto do problema que nos foi proposto.

Bibliografia

[Analide, 2001] ANALIDE, Cesar, NOVAIS, Paulo, NEVES, José, “Sugestões para a Elaboração de Relatórios”, Relatório Técnico, Departamento de Informática, Universidade do Minho, Portugal, 2001.

Anexos

I. Anexo 1- Árvore genealógica usada para testes