Universidade do Minho
Mestrado Integrado em Engenharia Informática
Departamento de Informática

Sistemas de Representação de Conhecimento e Raciocínio

Exercício 1

Isabel Sofia da Costa Pereira A76550
José Francisco Gonçalves Petejo e Igreja Matos A77688
Maria de La Salete Dias Teixeira A75281
Tiago Daniel Amorim Alves A78218

18 de Março de 2018

1 Resumo

Este trabalho baseia-se na elaboração de um sistema de saúde. Desta forma, são apresentadas as entidades que permitem desenvolver a base de conhecimento e os invariantes necessários para a correta inserção e remoção de conhecimento. Além disso, são expostos os predicados requisitados inicialmente e predicados adicionais que propocionam novos critérios de procura. Para garantir o correto funcionamento do sistema também são demostrados os respetivos resultados dos predicados.

Conteúdo

2 Introdução

No âmbito da unidade curricular de SRCR foi-nos proposto o desenvolvimento de um sistema capaz de gerir cuidados de saúde, desde de utentes, pretadores, instituições e as próprias consultas. Para tal, foi necessária a elaboração de vários predicados na linguagem de progamação em lógica, *PROLOG*.

3 Preliminares

Um sistema de saúde, capaz de gerir todos os cuidados necessários para a população, tem que considerar todo um conjunto de pessoas, instituições e cuidados. Quando falamos de prestação de cuidados de saúde, temos 2 grandes grupos, os utentes e os prestadores. Os utentes, pessoas que recorrem aos cuidados, devem ser definidos por um identificador único, para diferenciar utentes com o mesmo nome completo, e também definidos pelo nome, idade e morada. Isto porque a idade é importante quando queremos avaliar a saúde de um indivíduo e a morada é importante para se tentar encaminhar o indivíduo para a instituição mais próxima. Os prestadores, que vão desde médicos e enfermeiros a nutricionista, tal como os utentes, devem ter um identificador único e nome. Para além disso, um prestador deve também ser definido pela sua especialidade, fator essencial na marcação de cuidados, e pela instituição em que trabalha.

Isto leva-nos a mais um conjunto essencial num sistema de saúde, as instituições. Uma instituição deve ser definida por um identificar único, porque, por exemplo, pode haver mais do que um hospital com o mesmo nome, o nome e a localidade. A localidade é importante para facilitar a marcação de cuidados de acordo com a morada de cada utente.

Por último, o essencial para gerir cuidados de saúde são os cuidados em si, que devem ser todos marcados no sistema. Assim, considera-se importante para a marcação de um cuidado, armazenar a data e a hora a que ocorrerá/ocorreu, o utente a ser atendido, o prestador que efetuou o cuidado, uma descrição informativa e o custo do cuidado.

Tendo em conta todos estes fatores, o grupo considerou essencial o desenvolvimento de quatro entidades para a base de conhecimento para o sistema de saúde, sendo estas as evidenciadas acima (utente, prestador, instituição e cuidado). Como uma base de conhecimento deve ter a capacidade de ser gerida e analisada de acordo com as necessidades existentes por o(s) utilizadore(s) do sistema, foram também desenvolvidos todos os invariantes e predicados necessários para tal.

4 Descrição do Trabalho e Ánalise dos Resultados

Com o objetivo de descrever um sistema de saúde foi essencial estabelecer quais os conhecimentos necessários e aplicar predicados sobre estes para assim podermos inferir factos.

4.1 Base de Conhecimento

A fim da construção do sistema foram adicionados os predicados utente, prestador e cuidado. Cada utente é identificado por um número, um nome, a sua idade e a sua morada. Os prestadores tem associados a si um identificador, um nome, a sua especialidade e o identificador da instituição a que pertencem. Por sua vez, os cuidados, que representam atos médicos, contêm informações pertinentes tais como a data, a hora, o identificador do utente e do prestador, uma descrição do porquê da consulta e o custo da mesma. O formato escolhido para a representação da data foi *ANO-MÊS-DIA*.

Para além do conhecimento descrito anteriormente, achou-se conveniente a criação de um novo predicado, a instituição. Neste predicado pode se encontrar informação sobre o nome da instituição, o local desta e o número que a identifica.

```
utente(1, tiago_alves, 20, vila_verde).
prestador(3, filipa_ferreira, pediatria, 1).
instituicao(1, hospital_de_braga, braga).
cuidado(2018-4-11, 17:00, 5, 7, rotina, 45).
```

Figura 1: Entidades do Sistema

Desta forma, é válido fazer certas questões ao sistema tal como quais os dados do utente com identificador 1 ou qual o cuidado com uma determinada data e hora.

```
| ?- utente(1,Nome,Idade,Morada).
Nome = tiago_alves,
Idade = 20,
Morada = vila_verde ?
yes
| ?- cuidado(2018-4-11,17:00,IdUt,IdPres,Descricao,Custo).
IdUt = 5,
IdPres = 7,
Descricao = rotina,
Custo = 45 ?
yes
| ?-
```

Figura 2: Possíveis questões ao Sistema e respetivas respostas

4.2 Predicados auxiliares

Ao longo da realização do trabalho, achamos pertinente a criação de predicados auxiliares, tais como:

- *comprimento*: devolve o tamanho de uma lista;
- *solucoes*: retorna todas as soluções encontradas a uma determinada questão;
- *evolucao*: adiciona um termo à base de conhecimento testando se nunhum invariante de inserção sobre esse termo não é infringido;
- *inevolucao*: remove um termo da base de conhecimento testanto se nenhum invariante de remoção sobre esse termo é desobedecido;
- *apagaT*: recebe uma lista e um elemento e prossegue à eliminação de todas as ocorrências do elemento na lista;
- *apagaRepetidos*: recebe uma lista e elimina os elementos repetidos presentes nesta;
- *somaLista*: dada uma lista, calcula a soma dos seus elementos;
- *dataMaior*: aceita duas datas e determina qual a maior;

- *custoPorDatas*: dada duas datas e uma lista de tuplos no formato (data,custo), retorna uma lista com os custos que se encontravam entre as datas definidas;
- *quantidade*: recebe uma lista e um elemento e especifica o número de vezes que esse ocorreu na lista utilizando o formato (elemento,número);
- *listaTuplos*: aceita uma lista e cria uma nova em que a cada elemento da primeira está associado o número de vezes que este surgiu, isto é, uma lista de tuplos no formato (elemento,número);
- *maiorElemento*: dada uma lista de tuplos onde o segundo elemento de cada tuplo é um número, determina qual o tuplo com maior segundo elemento;
- *topN*: recebe um número N e uma lista de tuplos, no formato (elemento,número), e determina quais os N tuplos maiores da lista;
- *apagaTuplos*: recebe uma lista de tuplos e um tuplo (X,Y) e elimina os tuplos da lista que têm como primeiro elemento X;
- *custoElemento*: dado um elemento e uma lista de tuplos, no formato (elemento,custo), determina a soma dos custos associados ao elemento solicitado;
- *custoTuplos*: aceita uma lista de tuplos, no formato (elemento,custo), e determina a soma dos custos associados a cada elemento;
- *mediaLista*: dada uma lista, calcula a média dos seus elementos;
- *verificaDataCuidado*: recebe três datas e verifica se a terceira está entre as duas primeiras.

4.3 Invariantes

Para uma base de conhecimento consistente foram desenvolvidos invariantes estruturais e referenciais.

Os estruturais tem como objetivo não permitir a inserção de conhecimento repetido, por exemplo, dentro dos utentes/prestadores/instituições existir dois com um identificador igual.

Os invariantes referenciais tratam as inconsistências lógicas no conhecimento, como por exemplo, a proibição da inserção de cuidados com a mesma data e hora para o mesmo utente ou prestador e, além disso, garantir que o utente e o prestador do cuidado a adicionar existem na base de conhecimento nao existe. Também impede a remoção de conhecimento que esteja a ser utilizado, tal como a remoção de utentes e prestadores que ainda tenham cuidados associados a si ou instituições que ainda possuam prestadores.

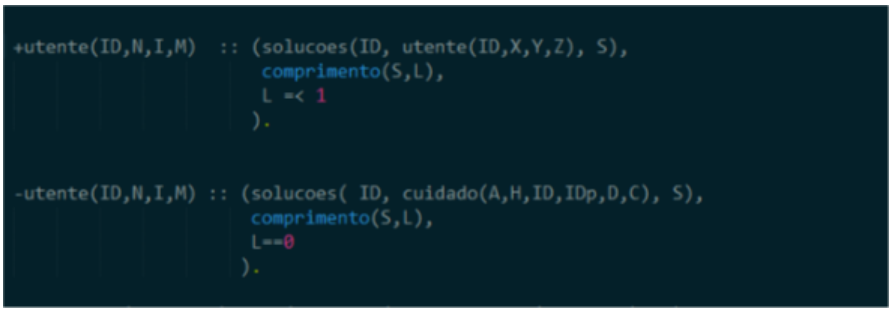

Figura 3: Invariantes do utente

```
| ?- evolucao(utente(11,pedro_silva,13,gondomar)).
yes
| ?- evolucao(utente(8,joao_soares,24,matosinhos)).
no
| ?- listing(utente).
utente(1, tiago_alves, 20, vila_verde).
utente(2, isabel_pereira, 21, vila_verde).
utente(3, francisco_matos, 16, tenoes).
utente(4, maria_teixeira, 21, gondizalves).
utente(5, rita_pereira, 45, braga).
utente(6, antonio_silva, 33, guimaraes).
utente(7, mario_duarte, 64, gualtar).
utente(8, helena_dias, 75, amares).
utente(9, sara_pires, 6, povoa_de_lanhoso).
utente(10, hugo_antunes, 83, esposende).
utente(11, pedro_silva, 13, gondomar).

yes
| ?-
```

Figura 4: Inserção de utente

```
| ?- listing(cuidado).
cuidado(2018-4-10, 16:0, 2, 1, ansiedade, 4).
cuidado(2018-4-11, 17:0, 5, 7, rotina, 45).
cuidado(2018-5-16, 9:0, 2, 4, alergia_na_pele, 60).
cuidado(2018-5-30, 9:0, 2, 4, alergia_na_pele, 60).
cuidado(2018-6-2, 15:30, 3, 3, checkup, 10).
cuidado(2018-9-30, 11:0, 3, 5, stress, 30).
cuidado(2018-10-20, 10:45, 1, 4, acne, 60).
cuidado(2018-10-22, 14:25, 10, 10, infecao, 5).
cuidado(2018-11-5, 11:30, 4, 11, ecografia, 35).
cuidado(2018-11-5, 11:0, 7, 2, rotina, 70).
cuidado(2018-11-27, 13:45, 6, 9, miopia, 55).
cuidado(2018-12-4, 12:0, 8, 8, dores_de_cabeca, 4).

yes
| ?- inevolucao(utente(9,sara_pires,6,povoa_de_lanhoso)).
yes
| ?- inevolucao(utente(2,isabel_pereira,21,vila_verde)).
no
| ?-
```

Figura 5: Remoção de utente

4.4 Funcionalidades

Foram implementadas procuras base sobre utentes, prestadores, instituições e cuidados usando cada um dos seus parâmetros. Apesar de serem pesquisas simples podem tornar-se bastante úteis para obter informações tais como quais os utentes de uma determinada zona, os prestadores de uma especialidade em questão, os prestadores de uma certa instituição ou os cuidados que um utente já realizou podendo assim informá-lo qual o nome do seu médico.

```
identificaPrestadorEsp(E,S) :- solucoes((ID,N), prestador(ID,N,E,IDi), S).

identificaPrestadorIDInst(IDi,S) :- solucoes((ID,N), prestador(ID,N,E,IDi), S).

identificaCuidadoIDU(IDu,S) :- solucoes((D,H,IDu,IDp,DC,C), cuidado(D,H,IDu,IDp,DC,C), S).
```

Figura 6: Exemplos de predicados de procura

```
| ?- identificaPrestadorEsp(pediatria,S).
S = [(3,filipa_ferreira),(6,renato_torres)] ?
yes
| ?- identificaPrestadorIDInst(2,S).
S = [(2,mario_oliveira),(4,joao_machado),(8,andre_fernandes),(12,filipe_alves)] ?
yes
| ?- identificaCuidadoIDU(2,S).
S = [(2018-4-10,16:0,2,1,ansiedade,4),(2018-5-16,9:0,2,4,alergia_na_pele,60),(2018-5-30,9:0,2,4,alergia_na_pele,60)] ?
yes
| ?-
```

Figura 7: Respostas a esses mesmos predicados

Além disso, também se desenvolveu predicados que ao receberem uma lista de identificadores, associam ao identificador o nome do utente/prestador/instituição. Para os utentes, também se implementou um predicado que dado vários identificadores de utentes, substitui cada um destes pela idade correspondente do utente.

Com estes predicados, se o utilizador pretender obter informações sobre vários utentes, por exemplo, o nome dos cinco primeiros utentes do sistema, basta evocar o predicado correspondente com a lista [1,2,3,4,5].

```
nomesUtentes([],[]).
nomesUtentes([ID|T],[(ID,N)|R]) :- utente(ID,N,_,_),
                                   nomesUtentes(T,R).
```

Figura 8: Predicado *nomesUtentes*

```
| ?- nomesUtentes([1,2,3,4,5],S).
S = [(1,tiago_alves),(2,isabel_pereira),(3,francisco_matos),(4,maria_teixeira),(5,rita_pereira)] ?
yes
| ?- listing(utente).
utente(1, tiago_alves, 20, vila_verde).
utente(2, isabel_pereira, 21, vila_verde).
utente(3, francisco_matos, 16, tenoes).
utente(4, maria_teixeira, 21, gondizalves).
utente(5, rita_pereira, 45, braga).
utente(6, antonio_silva, 33, guimaraes).
utente(7, mario_duarte, 64, gualtar).
utente(8, helena_dias, 75, amares).
utente(9, sara_pires, 6, povoa_de_lanhoso).
utente(10, hugo_antunes, 83, esposende).

yes
| ?-
```

Figura 9: Resposta ao respetivo predicado

Estando as procuras base definidas, achou-se pertinente interligar várias informações da base de conhecimento para obteção de respostas gerando assim mais conhecimento benéfico. Tal como:

- Cuidados/Identificador da Instituição
- Cuidados/Nome da Instituição
- Cuidados/Localidade da Instituição
- Utentes/Identificador do Prestador
- Utentes/Nome do Prestador
- Utentes/Especialidade
- Utentes/Identificador da Instituição
- Utentes/Nome da Instituição
- Utentes/Localidade da Instituição
- Instituições e Prestadores/Identificador de Utente
- Número de Utentes/Identificador de Prestador
- Número de Utentes/Identificador da Instituição
- Número de Prestadores/Identificador da Instituição
- Número de Especialidades/Identificador da Instituição

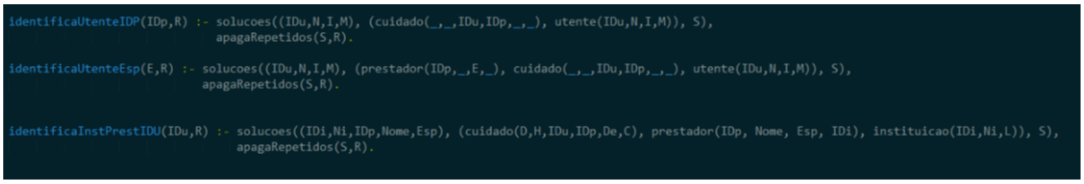

Figura 10: Exemplos de Predicados

```
| ?- identificaUtenteIDP(4,S).
S = [(2,isabel_pereira,21,vila_verde),(1,tiago_alves,20,vila_verde)] ?
yes
| ?- identificaUtenteEsp(pediatria,S).
S = [(3,francisco_matos,16,tenoes)] ?
yes
| ?- identificaInstPrestIDU(6,S).
S = [(3,hospital_sao_joao,9,filipe_alves,oftalmologia)] ?
yes
| ?-
```

Figura 11: Respostas aos Respetivos Predicados

Uma outra funcionalidade que foi implementada foi o cálculo do custo total dos cuidados de saúde de acordo com um determinado parâmetro, nomeadamente:

- Identificador do Utente
- Especialidade
- Identificador do Prestador
- Identificador da Instituição
- Data
- Duas datas

Estas funcionalidades podem ser bastante vantajosas para se obter uma melhor perceção de quanto uma especialidade, prestador ou instituição fatura. Para além disso, com o uso das datas, tem-se uma noção mais clara de quais os anos ou meses, por exemplo, que houve maiores gastos por parte dos utentes no serviço de saúde.

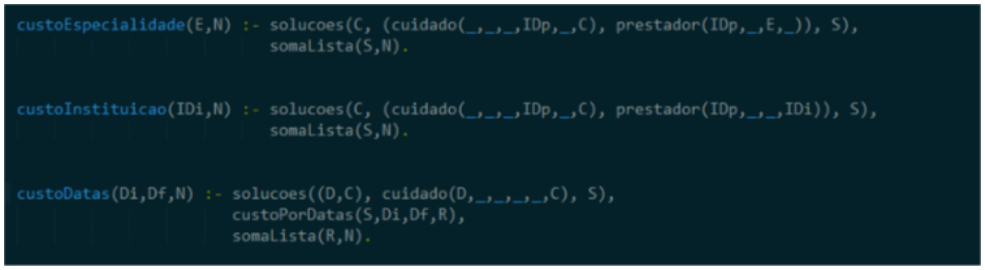

Figura 12: Exemplos de Predicados relacionados com Custos

```
| ?- custoEspecialidade(dermatologia,S).
S = 180 ?
yes
| ?- custoInstituicao(3,S).
S = 59 ?
yes
| ?- custoDatas(2018-11-1,2018-12-1,S).
S = 160 ?
yes
| ?- listing(cuidado).
cuidado(2018-4-10, 16:0, 2, 1, ansiedade, 4).
cuidado(2018-4-11, 17:0, 5, 7, rotina, 45).
cuidado(2018-5-16, 9:0, 2, 4, alergia_na_pele, 60).
cuidado(2018-5-30, 9:0, 2, 4, alergia_na_pele, 60).
cuidado(2018-6-2, 15:30, 3, 3, checkup, 10).
cuidado(2018-9-30, 11:0, 3, 5, stress, 30).
cuidado(2018-10-20, 10:45, 1, 4, acne, 60).
cuidado(2018-10-22, 14:25, 10, 10, infecao, 5).
cuidado(2018-11-5, 11:30, 4, 11, ecografia, 35).
cuidado(2018-11-5, 11:0, 7, 2, rotina, 70).
cuidado(2018-11-27, 13:45, 6, 9, miopia, 55).
cuidado(2018-12-4, 12:0, 8, 8, dores_de_cabeca, 4).

yes
| ?-
```

Figura 13: Respostas aos Respetivos Predicados

Outra funcionalidade interessante foi a construção de rankings. Desta forma, é possível estudar quais as N especialidades ou instituições com maior lucro ou então os utentes com maiores gastos numa determinada instituição. Além disso, também se pode tomar conhecimento de quais as especialidades, prestadores ou instituições mais procurados pelos utentes do sistema.

```
especialidadesMaisLucrativas(N,R) :- solucoes((E,C), (prestador(IDp,_,E,_), cuidado(_,_,_,IDp,_,C)), S),
                                     custoTuplos(S,T),
                                     topN(T,N,R).


topNPrestadores(N,R) :- solucoes(IDp, (prestador(IDp,_,_,_), cuidado(_,_,_,IDp,_,_)), S),
                        listaTuplos(S,T),
                        topN(T,N,X),
                        nomesPrestadores(X,R).


utentesMaiorCustoInst(IDi,N,R) :- solucoes((IDu,C), (prestador(IDp,_,_,IDi), cuidado(_,_,IDu,IDp,_,C)), S),
                                  custoTuplos(S,T),
                                  topN(T,N,X),
                                  nomesUtentes(X,R).
```

Figura 14: Exemplos de Predicados relativos a rankings

```
| ?- especialidadesMaisLucrativas(5,S).
S = [dermatologia,clinica_geral,oftalmologia,cardiologia,ginecologia] ?
yes
| ?- topNPrestadores(5,S).
S = [(4,joao_machado),(1,tania_fernandes),(2,mario_oliveira),(3,filipa_ferreira),(5,andre_correia)] ?
yes
| ?- utentesMaiorCustoInst(1,3,S).
S = [(4,maria_teixeira),(3,francisco_matos),(10,hugo_antunes)] ?
yes
```

Figura 15: Respostas aos respetivos predicados

Para estudos estatisticos foram desenvolvidos predicados relacionados com a idade dos utentes. Por exemplo, *mediaIdadeGeral*, *mediaIdadeInstituicao* e *mediaIdadeEspecialidade*. Com esta última, se o sistema contiver uma base de conhecimento ampla, é prático tirar conclusões sobre quais problemas afetam uma determinada faixa etária.

```
mediaIdadeEspecialidade(E,R) :- solucoes(IDu, (prestador(IDp,_,E,_), cuidado(_,_,IDu,IDp,_,_)), S),
                                apagaRepetidos(S,T),
                                idadesUtentes(T,I),
                                mediaLista(I,R).
```

Figura 16: Exemplo de um predicado

```
| ?-
| ?-
| ?- mediaIdadeEspecialidade(dermatologia,S).
S = 20.5 ?
yes
| ?-
| ?-
| ?-
```

Figura 17: Resposta ao respetivo predicado

Com o intuito de manipular o conhecimento através das datas presentes nos cuidados, foram elaborados predicados que efetuam procuras tal como, quais foram os utentes que procuraram uma certa especialidade/prestador entre duas datas, quais foram os prestadores procurados entre duas datas, uu então quais os cuidados efetuados, no geral ou de uma certa especialidade, entre duas determinadas datas.

```
utentesPrestadorDatas(IDp,D1,D2,R) :- solucoes((IDu,Nu), (prestador(IDp,_,_,_), cuidado(Dc,_,IDu,IDp,_,_), verificaDataCuidado(D1,D2,Dc), utente(IDu,Nu,_,_)), S),
                                      apagaRepetidos(S,R).


cuidadosEntreDatas(D1,D2,S) :- solucoes((Dc,H), (cuidado(Dc,H,_,_,_,_), verificaDataCuidado(D1,D2,Dc)), S).
```

Figura 18: Exemplos de predicados relativos a datas

```
| ?- utentesPrestadorDatas(4,2018-3-1,2018-11-1,S).
S = [(2,isabel_pereira),(1,tiago_alves)] ?
yes
| ?- cuidadosEntreDatas(2018-3-1,2018-6-1,S).
S = [(2018-4-10,16:0),(2018-4-11,17:0),(2018-5-16,9:0),(2018-5-30,9:0)] ?
yes
| ?-
| ?-
| ?
```

Figura 19: Resposta ao respetivo predicado

5 Conclusões e Sugestões

A elaboração deste primeiro trabalho permitiu-nos aprofundar os nossos conhecimentos em relação à linguagem de programacão em lógica, *PROLOG*.

Todos os requisitos propostos para este trabalho foram cumpridos, estando todos os predicados implementados e operacionais. Foi desenvolvido uma entidade extra, instituição, pelo facto de acharmos importante ter mais informações sobre cada instituição para além do seu identificador. Além disso, considerou-se essencial o desenvolvimento de predicados extras de forma a alcançar uma exploração mais profunda da base de conhecimento do sistema de saúde e disponibilizar mais funcionalidades ao utilizador da mesma.

Tendo em conta os resultados obtidos, consideramos que o trabalho foi realizado com sucesso e que desenvolvemos todas as funcionalidades necessárias para esta base de conhecimento.

6 Referências

Bratko, Ivan

"PROLOG: Programming for Artificial Intelligence"

Pearson Education, Edimburgo, 2001