

Universidade do Minho - Escola de Engenharia

Mestrado Integrado em Engenharia Informática

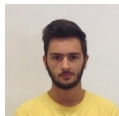
Sistemas de Representação de Conhecimento e Raciocínio

2017/2018 - 2º semestre

Exercício nº2 - Programação em Lógica Estendida e Conhecimento Imperfeito

Autores :

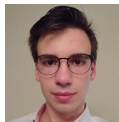
Carlos Campos (A74745)



Diana Costa (A78985)



Marcos Pereira (A79116)



Vitor Castro (A77870)



Braga, 20 de Abril de 2018

Resumo

Perante a proposta de realizar um sistema de representação de conhecimento e raciocínio, com capacidade de caracterizar um universo de discurso na área da prestação de cuidados de saúde, houve um impasse inicial devido ao surgimento do valor nulo e à necessidade de uma boa estruturação do problema. Tudo isto requereu a escolha e manipulação acertada de predicados, e bom uso da linguagem de programação em lógica estendida, *PROLOG*, de modo a que a resolução destes fosse a mais clara e simples possível.

Depois de algum tempo e trabalho, o resultado encontrado foi satisfatório, e os objetivos e respostas às questões do enunciado proposto cumpridos.

Conteúdo

1	Introdução	3
2	Preliminares	4
3	Descrição do trabalho e Análise de resultados	5
3.1	Representação de conhecimento positivo e negativo	5
3.2	Representação de casos de conhecimento imperfeito	7
3.2.1	Conhecimento imperfeito incerto	7
3.2.2	Conhecimento imperfeito impreciso	7
3.2.3	Conhecimento imperfeito interdito	8
3.3	Invariantes para inserção e remoção de conhecimento no sistema	10
3.4	Evolução do conhecimento	12
3.4.1	Evolução do conhecimento incerto	12
3.4.2	Evolução do conhecimento impreciso	13
3.4.3	Evolução do conhecimento interdito	14
3.4.4	Evolução para outros predicados	15
3.5	Sistema de Inferência	15
3.6	Outros Predicados	17
4	Conclusões e Sugestões	20

1 Introdução

Este segundo projeto surge no âmbito da unidade curricular de Sistemas de Representação de Conhecimento e Raciocínio, do Mestrado Integrado em Engenharia Informática da Universidade do Minho. Era requerido, como primeiro exercício, que fosse desenvolvido um sistema de representação de conhecimento e raciocínio com capacidade para caracterizar um universo de discurso na área da prestação de cuidados de saúde. O grupo decidiu utilizar a representação de conhecimento do primeiro projeto (incluindo o conhecimento extra enunciado "recibo") e manipular a mesma, de forma a que fosse introduzido o valor "desconhecido".

Assim, como primeira instância, foi descrita a seguinte caracterização de conhecimento,

- utente: $\#IdUt, Nome, Idade, Morada \rightsquigarrow \{\mathbb{V}, \mathbb{F}, \mathbb{D}\}$
- prestador: $\#IdPrest, Nome, Especialidade, Instituição \rightsquigarrow \{\mathbb{V}, \mathbb{F}, \mathbb{D}\}$
- cuidado: $Data, \#IdUt, \#IdPrest, Descrição, Custo \rightsquigarrow \{\mathbb{V}, \mathbb{F}, \mathbb{D}\}$

à qual o grupo adicionou:

- recibo: $\#IdUt, NomeUt, \#IdPrest, Especialidade, Data, Custo \rightsquigarrow \{\mathbb{V}, \mathbb{F}, \mathbb{D}\}$

Como segunda tarefa, era necessário que se representasse conhecimento positivo e negativo, uma vez que, com a introdução do "desconhecido", é preciso definir bem o que significa um predicado retornar verdadeiro ou falso.

Por conseguinte, e de forma a dar uso a este novo valor de verdade, foram definidos, na base de conhecimento, todos os tipos de nulo estudados nas aulas - nulo incerto, impreciso e interdito -, que permitem ao grupo testar e dar vida ao sistema.

Por fim, teriam de ser definidos os invariantes, de forma a manter a consistência nas inserções e remoções de conhecimento, e lidar com a problemática da evolução de conhecimento do sistema.

Em suma, a Secção 2 descreve os preliminares necessários ao projeto, enquanto que a Secção 3 descreve a resolução do trabalho, desde a representação do conhecimento positivo e negativo(3.1), representação de casos de conhecimento imperfeito(3.2), descrição de invariantes (3.3), evolução do conhecimento do sistema(3.4) e sistema de inferência(3.5). Esta secção termina com a consequente análise geral de predicados que foram usados com mais frequência ao longo do exercício(Secção 3.6). Ao longo do trabalho há algumas evidências da aplicação dos diferentes predicados.

O relatório termina com uma breve conclusão na Secção 4, onde é feito um balanço do trabalho realizado, tendo em conta as dificuldades ao longo do desenvolvimento do mesmo.

2 Preliminares

Para o desenvolvimento deste exercício prático não foi necessário o estudo de matérias adicionais, apenas da matéria lecionada, até então, na presente unidade curricular. Ainda assim, o grupo considera pertinente declarar, aqui, o estudo inicial que permitiu fazer a transição entre o primeiro e o segundo projeto, correspondente aos pressupostos do mundo fechado, nomes únicos e domínio fechado, e à representação de conhecimento imperfeito incerto, impreciso e interdito.

As linguagens de manipulação de informação num sistema de representação de conhecimento alicerçam-se, basicamente, nos seguintes pressupostos:

- **Pressuposto do Mundo Fechado (PMF)** - toda a informação que não existe mencionada na base de dados é considerada falsa;
- **Pressuposto dos Nomes Únicos (PNU)** - duas constantes diferentes (que definam valores atômicos ou objetos) designam, necessariamente, duas entidades diferentes no universo de discurso;
- **Pressuposto do Domínio Fechado (PDF)** - não existem mais objetos no universo de discurso para além daqueles designados por constantes na base de dados.

Estes pressupostos das linguagens tradicionais de Programação em Lógica (PL) não permitem a representação de conhecimento do senso comum: nem sempre se pretende assumir que a informação representada é a única que é válida e, muito menos, que as entidades representadas sejam as únicas existentes no mundo exterior. Deste modo, e com o intuito de retificar este problema, surgem esquemas de raciocínio não-monótono, que proporcionam uma maior flexibilidade no mecanismo de inferência e no processo de obtenção de conclusões mas fazem-no recorrendo à negação por falha, adotando o PMF. Surgem assim, na Programação em Lógica Estendida, os PMA e PDA, sendo que o PNU se mantém:

- **Pressuposto do Mundo Aberto (PMA)** - podem existir outros factos ou conclusões verdadeiros para além daqueles representados na base de conhecimento;
- **Pressuposto dos Nomes Únicos (PNU)** - duas constantes diferentes (que definam valores atômicos ou objetos) designam, necessariamente, duas entidades diferentes no universo de discurso;
- **Pressuposto do Domínio Aberto (PDA)** - podem existir mais objetos no universo de discurso para além daqueles designados pelas constantes na base de dados.

Por conseguinte, surge, então, mais um valor de verdade para além do **verdadeiro** ou **falso**, designado como **desconhecido**. Este "desconhecido", na representação de conhecimento, surge em coordenação com o **conhecimento imperfeito**, que pode ser de três tipos:

- **Nulo Incerto** - É um valor nulo do tipo desconhecido, e de um conjunto indeterminado de valores;
- **Nulo Impreciso** - Valor nulo desconhecido, mas de um conjunto determinado de valores (conjunto de valores bem definidos);
- **Nulo Interdito** - Representam valores desconhecidos e caracterizam, ainda, um tipo de dados que não se pretende admitir que surjam na base de conhecimento.

Todos estes conceitos anteriores serão aplicados e abordados neste segundo projeto, sendo que, pela breve explicação que se fez nesta secção, poupar-se-ão futuras explicações pormenorizadas. Considera-se também que a leitura e compreensão deste relatório são facilitadas através de conhecimentos de programação em lógica estendida com invariantes, consciência da sintaxe de *Prolog* e entendimento para representar conhecimento e construir mecanismos de raciocínio para dar solução aos problemas propostos.

3 Descrição do trabalho e Análise de resultados

Nos seguintes tópicos são apresentados os predicados exigidos à representação do conhecimento e raciocínio pedida.

3.1 Representação de conhecimento positivo e negativo

Relembra-se, tendo por base o primeiro projeto, o conhecimento positivo representado pelo grupo:

```
% Extensao do predicado utente: IdUt, Nome, Idade, Morada -> {V,F,D}
utente(1, carlos, 21, famalicao).
utente(2, diana, 20, trofa).
utente(3, marcos, 20, anais).
utente(4, vitor, 20, guimaraes).
utente(5, antonio, 6, povoadevarzim).
utente(6, rita, 26, pontedelima).
utente(7, marta, 66, guimaraes).
utente(8, paulo, 16, barcelos).
utente(9, maria, 43, barcelos).
utente(10, luis, 51, vizela).
```

Figura 1: Representação de conhecimento positivo - Utente

```
% Extensao do predicado prestador: IdPrest, Nome, Especialidade, Instituicao -> {V,F,D}
prestador(1, pres1, pediatria, csjoane).
prestador(2, pres2, cardiologia, hospitalbraga).
prestador(3, pres3, cirurgia, hospitalbraga).
prestador(4, pres4, ginecologia, hospitalbraga).
prestador(5, pres5, neurologia, hsmm).
prestador(6, pres6, psiquiatria, hsg).
prestador(7, pres7, oftamologia, htrofa).
prestador(8, pres8, reumatologia, htrofa).
prestador(9, pres9, psiquiatria, hospitalbraga).
```

Figura 2: Representação de conhecimento positivo - Prestador

```
% Extensao do predicado cuidado: Data, IdUt, IdPrest, Descricao, Custo -> {V,F,D}
cuidado(01-02-2017, 1, 6, hipnose, 15).
cuidado(13-02-2017, 3, 4, papanico, 30).
cuidado(13-02-2017, 2, 5, cerebrotroca, 30).
cuidado(14-02-2017, 2, 7, olhoterapia, 7).
cuidado(20-03-2017, 4, 2, pacemaker, 20).
cuidado(02-04-2017, 7, 4, ovarilogia, 5).
cuidado(03-04-2017, 3, 5, neuroterapia, 25).
cuidado(20-04-2017, 6, 7, retinoterapia, 35).
cuidado(22-05-2017, 6, 2, cardioterapia, 55).
cuidado(04-06-2017, 2, 2, cardiograma, 99).
cuidado(15-06-2017, 1, 1, terapiafala, 5).
cuidado(18-06-2017, 2, 8, reumatografia, 350).
cuidado(18-06-2017, 2, 9, cbt, 10).
```

Figura 3: Representação de conhecimento positivo - Cuidado

```
% Extensao do predicado recibo: IdRecibo, IdUt, NomeUt, Morada, Especialidade, Instituicao, Data, Descricao, Custo -> {V,F,D}
recibo(1, 3, marcos, anais, ginecologia, hospitalbraga, 13-02-2017, papanico, 30).
recibo(2, 4, vitor, guimaraes, cardiologia, hospitalbraga, 20-03-2017, pacemaker, 20).
```

Figura 4: Representação de conhecimento positivo - Recibo

De forma a dar continuidade ao segundo projeto, definiram-se predicados que possibilitassem deduzir falsidades, de acordo com o conhecimento negativo que o grupo pretendia representar. Esta necessidade surge pelo facto da aparição do valor desconhecido, que implica que o conceito de falso seja redefinido mais pormenorizadamente, através de **negação forte**. Assim, considera-se falso tudo o que não seja verdadeiro, e também que não seja desconhecido, isto é, uma exceção à verdade. Para este efeito, desenvolveram-se os predicados "excecao" e "nao", sendo que "excecao" surge para possibilitar a existência do conhecimento imperfeito, demonstrado nas próximas secções, e o último surge pela necessidade de verificar a existência ou não da questão que lhe é colocada.

```
% Definições das negações fortes (não é verdadeiro ^ não é desconhecido)

-utente(Id,N,I,M) :-
    nao(utente(Id,N,I,M)),
    nao(excecao(utente(Id,N,I,M))).

-prestador(Id,N,E,I) :-
    nao(prestador(Id,N,E,I)),
    nao(excecao(prestador(Id,N,E,I))).

-cuidado(D,IdU,IdP,De,C) :-
    nao(cuidado(D,IdU,IdP,De,C)),
    nao(excecao(cuidado(D,IdU,IdP,De,C))).

-recibo(IdR, IdU, N, M, E, I, D, De, C) :-
    nao(recibo(IdR, IdU, N, M, E, I, D, De, C)),
    nao(excecao(recibo(IdR, IdU, N, M, E, I, D, De, C))).
```

Figura 5: Representação de conhecimento negativo

3.2 Representação de casos de conhecimento imperfeito

Com o intuito de representar conhecimento imperfeito, o grupo usou da sua criatividade e criou alguns exemplos para cada tipo de conhecimento. Como tal, seguem-se, nas próximas secções, os exemplos desenvolvidos e a respetiva explicação.

3.2.1 Conhecimento imperfeito incerto

Seguem-se exemplos simples de nulos incertos na base de conhecimento. Para o predicado "utente", não se sabe qual a morada da Mariana, com id=12, que tem 43 anos.

```
% -----  
% Desconhece-se a morada do utente com o id=12, correspondente à Mariana de 43 anos.  
utente(12, mariana, 43, i1).  
excecao( utente( A, B, C, D ) ) :-  
    utente( A, B, C, i1).
```

Figura 6: Exemplo de um nulo incerto para o predicado "utente"

Para o predicado "prestador", desconhece-se qual é o nome do prestador com id=11 do Hospital de Braga, nem a especialidade que o mesmo exerce.

```
% -----  
% Não se sabe qual é o nome do prestador com o id=11 do hospital de braga,  
% nem qual a especialidade que o mesmo exerce.  
prestador(11, i2, i3, hospitalbraga).  
excecao( prestador( A, B, C, D ) ) :-  
    prestador( A, i2, C, D).  
excecao( prestador( A, B, C, D ) ) :-  
    prestador( A, B, i3, D).
```

Figura 7: Exemplo de um nulo incerto para o predicado "prestador"

3.2.2 Conhecimento imperfeito impreciso

Apresentam-se, de seguida, alguns exemplos de nulos imprecisos na base de conhecimento, definidos pelo grupo. Para o predicado "utente", desconhece-se se a utente Inês (id=11), residente em Lisboa, tem 23 ou 24 anos.

```
% -----  
% Não se sabe se a utente com id=11, com o nome Inês, residente em Lisboa, tem 23 ou 24 anos.  
excecao( utente(11, ines, 23, lisboa) ).  
excecao( utente(11, ines, 24, lisboa) ).
```

Figura 8: Exemplo de um nulo impreciso para o predicado "utente"

Para o prestador nº10, designado pres10, não se sabe se o mesmo exerce dermatologia no Hospital de Braga ou no Hospital de São João.

```
% -----  
% Desconhece-se se o prestador com id=10, pres10, exerce dermatologia no Hospital de Braga ou no Hospital de São João.  
excecao( prestador(10, pres10, dermatologia, hospitalbraga) ).  
excecao( prestador(10, pres10, dermatologia, hsaojoao) ).
```

Figura 9: Exemplo de um nulo impreciso para o predicado "prestador"

No predicado "cuidado", não se sabe se o cuidado "penso" com custo de 10€, prestado em 19-12-2017 pelo prestador nº8, foi proporcionado ao utente nº2 ou nº3.


```
% -----
% Não se sabe se o cuidado "penso" prestado em 19-12-2017, pelo prestador com id=8 e com o
% valor monetário de 10€ corresponde ao utente com id=2 ou id=3.
execcao( cuidado(19-12-2017, 2, 8, penso, 10) ).
execcao( cuidado(19-12-2017, 3, 8, penso, 10) ).
```

Figura 10: Exemplo de um nulo impreciso para o predicado "cuidado"

Mais uma vez, para o predicado "cuidado", desconhece-se se o cuidado provido em 12-05-2017, pelo prestador nº5 ao utente nº8 foi radioterapia ou quimioterapia, ou se o custo foi 50€ ou 70€.

```
% -----
% Desconhece-se se o cuidado prestado em 12-05-2017 pelo prestador com id=5 ao utente com id=8 foi
% quimioterapia ou radioterapia, e se o custo foi 50€ ou 70€.
execcao( cuidado(12-05-2017, 8, 5, radioterapia, 50) ).
execcao( cuidado(12-05-2017, 8, 5, radioterapia, 70) ).
execcao( cuidado(12-05-2017, 8, 5, quimioterapia, 50) ).
execcao( cuidado(12-05-2017, 8, 5, quimioterapia, 70) ).
```

Figura 11: Exemplo de um nulo impreciso para o predicado "cuidado"

Para o último predicado, sabe-se que existe um recibo do utente nº2, designado Diana, residente na Trofa, que declara uma consulta de rotina de nutrição, prestada no Hospital da Trofa, com o custo de 50€. No entanto, desconhece-se se este recibo foi emitido na data de 27-03-2017 ou 28-03-2017.

```
% -----
% Existe um recibo (idRecibo=3) do utente com id=2, intitulado Diana, que vive na Trofa e efetuou,
% no hospital da trofa, uma consulta de rotina na área da nutrição, com custo de 50€.
% Desconhece-se, no entanto, se o recibo foi emitido em 27-03-2017 ou 28-03-2017.
execcao( recibo(3, 2, diana, trofa, nutricao, htrofa, 27-03-2017, rotina, 50) ).
execcao( recibo(3, 2, diana, trofa, nutricao, htrofa, 28-03-2017, rotina, 50) ).
```

Figura 12: Exemplo de um nulo impreciso para o predicado "recibo"

Também para o predicado "recibo", sabe-se que existe um recibo (id=5) que descreve um serviço prestado ao utente número 10, denominado Luís, que vive em Vizela e efetuou, no Hospital de Braga, um cuidado de hipnose (na área de psiquiatria) a 02-11-2017. No entanto, desconhece-se o custo do recibo, mas sabe-se que se situa entre 35€ e 60€.

```
% -----
% Existe um recibo (idRecibo=5) do utente com id=10, intitulado luis, que vive em vizela e efetuou,
% no hospital de braga, hipnose na área da psiquiatria, na data 02-11-2017.
% Desconhece-se o custo declarado no recibo, no entanto sabe-se que se situa entre 35€ e 60€.
recibo(5, 10, luis, vizela, psiquiatria, hbraga, 02-11-2017, hipnose, i4).
execcao( recibo(A, B, C, D, E, F, G, H, I) ) :-
    recibo(A, B, C, D, E, F, G, H, i4).
recibo(5, 10, luis, vizela, psiquiatria, hbraga, 02-11-2017, hipnose, i4) :-
    i4 > 35, i4 < 60.
```

Figura 13: Exemplo de um nulo incerto para o predicado "recibo"

3.2.3 Conhecimento imperfeito interdito

Por último, apresentam-se dois exemplos simples de nulos interditos na base de conhecimento. Para o imediatamente abaixo, tem-se que existe um cuidado realizado a 13-08-2017, proporcionado pelo prestador nº7 ao utente nº10, com um custo respetivo de 75€. Ainda assim, nunca se poderá saber qual a descrição deste cuidado.

Chama-se à atenção para a última linha da imagem, na qual não se deixa inserir no sistema a descrição de um cuidado nestas condições. Isto é sensato, uma vez que se trata de conhecimento interdito, ou seja, inacessível em qualquer circunstância.

```
% -----
% Não se pode saber qual a descrição do cuidado realizado a 13-08-2017, prestado pelo prestador com id=7
% ao utente com id=10, sendo que o custo foi de 75€.
cuidado( 13-08-2017, 10, 7, i5, 45).
execcao( cuidado(A, B, C, D, E) ) :-
    cuidado(A, B, C, i5, E).
nulointerdito( i5 ).
+cuidado( A, B, C, D, E ) :: ( solucoes((A,B,C,E), (cuidado( 13-08-2017, 10, 7, i5, 45), nao(nulointerdito(i5))), List),
    comprimento( List, N ),
    N == 0
    ).
```

Figura 14: Exemplo de um nulo interdito para o predicado "cuidado"

Para o predicado "recibo", nunca se poderá saber, a partir do recibo nº4, em que instituição/hospital foi prestado um peeling químico, na área da dermatologia, ao utente nº7 designado Marta, em 04-12-2017, com o custo de 30€. Por consequência, nunca poderá ser inserido, na base de conhecimento, algum recibo com as condições descritas anteriormente, com o objetivo de lhe adicionar uma instituição, uma vez que este conhecimento é, e será sempre, interdito.

```
% -----
% Nunca se poderá saber, a partir do recibo com id=4, em que instituição foi prestado um peeling químico,
% na área da dermatologia, ao utente com id=7 e intitulado Marta, em 04-12-2017, com o custo de 30€.
recibo( 4, 7, marta, guimaraes, dermatologia, i6, 04-12-2017, peelingquimico, 30 ).
execcao( recibo(A, B, C, D, E, F, G, H, I) ) :-
    recibo(A, B, C, D, E, i6, G, H, I).
nulointerdito( i6 ).
+recibo(A, B, C, D, E, F, G, H, I) :: ( solucoes((A, B, C, D, E, G, H, I), (recibo( 4, 7, marta, guimaraes, dermatologia, i6, 04-12-2017, peelingquimico, 30),
    nao(nulointerdito(i6))), List ),
    comprimento( List, N ),
    N == 0
    ).
```

Figura 15: Exemplo de um nulo interdito para o predicado "recibo"

3.3 Invariantes para inserção e remoção de conhecimento no sistema

Nesta secção são usados os mesmos invariantes do exercício um, sem nenhuma alteração. Visto que, a inserção ou remoção de conhecimento, com esta nova extensão da programação em lógica mantêm-se constante.

```
+utente(Id,N,I,M)::(solucoes(Id,utente(Id,_,_,_),S),  
    comprimento(S,L),  
    L==1).
```

Figura 16: Invariante de Inserção de Utentes

```
+prestador(Id,N,E,I)::(solucoes(Id,prestador(Id,_,_,_),S),  
    comprimento(S,L),  
    L==1).
```

Figura 17: Invariante de Inserção de Prestadores

```
+cuidado(Dat,U,P,D,C)::(solucoes(P,prestador(P,_,_,_),LisP),  
    comprimento(LisP,NumP),  
    NumP==1,  
    solucoes(U,utente(U,_,_,_),LisU),  
    comprimento(LisU,NumU),  
    NumU==1).
```

Figura 18: Invariante de Inserção de Cuidados

```
+recibo(Id,U,N,M,E,I,Dat,D,C)::(solucoes(Id,recibo(Id,_,_,_,_,_,_,_),R),  
    comprimento(R,Lr),  
    Lr==1,  
    solucoes((U,N,M),utente(U,N,_,M),Uts),  
    comprimento(Uts,Lu),  
    Lu==1,  
    solucoes((P,E,I),prestador(P,_,E,I),Pres),  
    comprimento(Pres,Lp),  
    Lp>=1,  
    solucoes((Dat,U,P,D,C),cuidado(Dat,U,P,D,C),Cuids),  
    comprimento(Cuids,Lc),  
    Lc==1).
```

Figura 19: Invariante de Inserção de Recibos

```
-utente(Id,N,I,M)::(solucoes(Id,utente(Id,N,I,M),Uts),
    comprimento(Uts,Lu),
    Lu==1,
    solucoes(Id,cuidado(_,Id,_,_),Cuids),
    comprimento(Cuids,Lc),
    Lc==0).
```

Figura 20: Invariante de Remoção de Utentes

```
-prestador(Id,N,E,I)::(solucoes(Id,prestador(Id,_,_,_),Prests),
    comprimento(Prests,Lp),
    Lp==1,
    solucoes(Id,cuidado(_,_,Id,_,_),Cuids),
    comprimento(Cuids,Lc),
    Lc==0).
```

Figura 21: Invariante de Remoção de Prestadores

```
-cuidado(Dat,U,P,D,C)::(solucoes((Dat,U,P,D,C),cuidado(Dat,U,P,D,C),Cuids),
    comprimento(Cuids,L),
    L==1).
```

Figura 22: Invariante de Remoção de Cuidados

```
-recibo(Id,U,N,M,E,I,Dat,D,C)::fail.
```

Figura 23: Invariante de Remoção de Recibos

3.4 Evolução do conhecimento

3.4.1 Evolução do conhecimento incerto

Agora, possível adicionar conhecimento através de duas vertentes, uma sendo a que tem sido utilizada até agora, que é a simples inserção de conhecimento positivo, com ajuda do predicado inserir, e a outra é a possibilidade de poder alterar conhecimento, seja ele falso ou desconhecido, para conhecimento verdadeiro, com ajuda do predicado evolução.

Para o caso em que o conhecimento é desconhecido, porque é incerto, tomemos o caso da Mariana, que é a utente numero 12, tem 43 anos de idade, mas por algum motivo, não se sabe a sua morada.

```
% -----  
% Desconhece-se a morada do utente com o id=12, correspondente à Mariana de 43 anos.  
utente(12, mariana, 43, i1).  
excecao( utente( A, B, C, D ) ) :-  
    utente( A, B, C, i1).
```

Figura 24: Exemplo de um nulo incerto para o predicado "utente"

Se tomarmos atenção na definição do predicado evolução, na parte em que é usado referente aos utentes:

```
evolucao(utente(Id, Nome, Idade, Morada)) :-  
    demo(utente(Id, Nome, Idade, Morada), desconhecido),  
    solucoes(utente(Id, N, I, M), utente(Id, N, I, M), L),  
    remocaoL(utente(Id, Nome, Idade, Morada), L).  
  
evolucao(utente(Id, Nome, Idade, Morada)) :-  
    demo(utente(Id, Nome, Idade, Morada), falso),  
    inserir(utente(Id, Nome, Idade, Morada)).
```

Figura 25: Definição do predicado evolução, no que toca a parte dos utentes

É possível ver que, caso a demonstração da questão utente que vai ser inserida, for desconhecida, essa mesma, é removida da base de conhecimento, através do predicado remocaoL, que remove de uma lista um determinado elemento, essa mesma lista é fornecida pelo predicado soluções, depois de feita a interceção do utente a ser inserido, com a base de conhecimentos. Também é o predicado remocaoL, que trata da inserção do termo, que é dado ao predicado evolução, na base de conhecimento. Como é possível ver na sua definição:

```

remocaoL(Termo,L):-
    retractL(L),
    inserir(Termo).
remocaoL(Termo,L):-
    assertL(L),!,fail.

retractL([]).
retractL([X|L]):-
    retract(X),
    retractL(L).

assertL([]).
assertL([X|L]):-
    assert(X),
    assertL(L).

```

Figura 26: Definição do predicado remocaoL

Por ultimo, é mostrado um teste à base de conhecimentos, em que é atribuído à Mariana, a rua de Fafe como morada.

```

| ?- demo(utente(12,mariana,43,'rua de fafe'),R).
R = desconhecido ?
yes
| ?- evolucao(utente(12,mariana,43,'rua de fafe')).
yes
| ?- demo(utente(12,mariana,43,'rua de fafe'),R).
R = verdadeiro ?
yes
| ?- _

```

Figura 27: Evolução de conhecimento incerto

3.4.2 Evolução do conhecimento impreciso

Para o caso em que o conhecimento é desconhecido, porque é impreciso, o raciocínio é exatamente o mesmo, do que quando é incerto, tomemos o caso da Inês, que é a utente numero 11, que reside em Lisboa, mas por algum motivo, não se sabe se tem 23 ou 24 anos.

```

% -----
% Não se sabe se a utente com id=11, com o nome Inês, residente em Lisboa, tem 23 ou 24 anos.
excecao( utente(11,ines,23,lisboa) ).
excecao( utente(11,ines,24,lisboa) ).

```

Figura 28: Evolução de conhecimento impreciso

Por ultimo, é mostrado um teste à base de conhecimentos, em que é atribuído à Inês, a sua verdadeira idade.

```

| ?-
| ?- demo(utente(11,ines,24,lisboa),R).
R = desconhecido ?
yes
| ?- evolucao(utente(11,ines,24,lisboa)).
yes
| ?- demo(utente(11,ines,24,lisboa),R).
R = verdadeiro ?
yes
| ?-

```

Figura 29: Evolução de conhecimento impreciso

3.4.3 Evolução do conhecimento interdito

```

% -----
% Não se pode saber qual a descrição do cuidado realizado a 13-08-2017, prestado pelo prestador com id=7
% ao utente com id=10, sendo que o custo foi de 75€.
cuidado( 13-08-2017, 10, 7, i5, 45).
excecao( cuidado(A, B, C, D, E) ) :-
    cuidado(A, B, C, i5, E).
nulointerdito( i5 ).
+cuidado( A, B, C, D, E ) :: ( solucoes((A,B,C,E), (cuidado( 13-08-2017, 10, 7, i5, 45), nao(nulointerdito(i5))), List),
    comprimento( List, N ),
    N == 0
    ).

```

Figura 30: Exemplo de um nulo interdito para o predicado "cuidado"

A computação executada para esta evolução seria a mesma, que o exemplo anterior, mas como existe sempre um invariante relacionado com o valor nulo interdito, tal não acontece. Vejamos o caso do cuidado do cuidado realizado a 13-08-2017, prestado pelo prestador com identificador numero sete, ao utente com identificador numero dez, sendo que o custo foi de 45€.

```

% -----
% Não se pode saber qual a descrição do cuidado realizado a 13-08-2017, prestado pelo prestador com id=7
% ao utente com id=10, sendo que o custo foi de 75€.
cuidado( 13-08-2017, 10, 7, i5, 45).
excecao( cuidado(A, B, C, D, E) ) :-
    cuidado(A, B, C, i5, E).
nulointerdito( i5 ).
+cuidado( A, B, C, D, E ) :: ( solucoes((A,B,C,E), (cuidado( 13-08-2017, 10, 7, i5, 45), nao(nulointerdito(i5))), List),
    comprimento( List, N ),
    N == 0
    ).

```

Figura 31: Exemplo de um nulo interdito para o predicado "cuidado"

Como podemos ver, **a evolução de um valor nulo interdito, nunca acontece.**

```

| ?- demo(cuidado(13-08-2017,10,7,hipnose,45),R).
R = desconhecido ?
yes
| ?- evolucao(cuidado(13-08-2017,10,7,hipnose,45)).
no
| ?- demo(cuidado(13-08-2017,10,7,hipnose,45),R).
R = desconhecido ?
yes
| ?-
| ?- ■

```

Figura 32: Evolução do conhecimento interdito

3.4.4 Evolução para outros predicados

As evoluções para os predicados prestador, cuidado e recibo são homologas em relação à do utente, como se pode ver através da sua definição.

```
evolucao(prestador(Id, Nome, Especialidade, Instituicao)):-  
    demo(prestador(Id, Nome, Especialidade, Instituicao), desconhecido),  
    solucoes(prestador(Id, N, E, I), prestador(Id, N, E, I), L),  
    remocaoL(prestador(Id, Nome, Especialidade, Instituicao), L).  
  
evolucao(prestador(Id, Nome, Especialidade, Instituicao)):-  
    demo(prestador(Id, Nome, Especialidade, Instituicao), falso),  
    inserir(prestador(Id, Nome, Especialidade, Instituicao)).  
  
evolucao(cuidado(Data, IdU, IdP, Descricao, Custo)):-  
    demo(cuidado(Data, IdU, IdP, Descricao, Custo), desconhecido),  
    solucoes(cuidado(Data, IdU, IdP, Descricao, Custo), cuidado(Data, IdU, IdP, Descricao, Custo), L),  
    remocaoL(cuidado(Data, IdU, IdP, Descricao, Custo), L).  
  
evolucao(cuidado(Data, IdU, IdP, Descricao, Custo)):-  
    demo(cuidado(Data, IdU, IdP, Descricao, Custo), falso),  
    inserir(cuidado(Data, IdU, IdP, Descricao, Custo)).  
  
evolucao(recibo(Data, IdU, IdS, Custo)):-  
    demo(recibo(IdR, IdU, NomeU, Morada, Especialidade, Instituicao, Data, Descricao, Custo), desconhecido),  
    solucoes(recibo(IdR, IdU, NomeU, Morada, Especialidade, Instituicao, Data, Descricao, Custo),  
             recibo(IdR, IdU, NomeU, Morada, Especialidade, Instituicao, Data, Descricao, Custo), L),  
    remocaoL(recibo(IdR, IdU, NomeU, Morada, Especialidade, Instituicao, Data, Descricao, Custo), L).  
  
evolucao(recibo(IdR, IdU, NomeU, Morada, Especialidade, Instituicao, Data, Descricao, Custo)):-  
    demo(recibo(IdR, IdU, NomeU, Morada, Especialidade, Instituicao, Data, Descricao, Custo), falso),  
    inserir(recibo(IdR, IdU, NomeU, Morada, Especialidade, Instituicao, Data, Descricao, Custo)).
```

Figura 33: Evolução para outros predicados

É ainda de salientar, que as clausulas do predicado evolução que mudam um termo de falso para verdadeiro, são usadas quando nada se sabe sobre esse mesmo. Mesmo que usemos o pressuposto do mundo aberto, só sabemos que algo é desconhecido, se computacionalmente for indicado na base de conhecimento. E portanto, tudo que não é referenciado na base de conhecimento, é considerado falso. Por isso é que sempre que um termo é indicado como falso, pelo predicado demo, é logo inserido.

3.5 Sistema de Inferência

O sistema de inferência capaz de implementar os mecanismos de raciocínio adequados a estes sistemas tem três tipos de resposta possíveis: verdadeiro, falso e desconhecido.

Este meta-predicado foi desenvolvido tendo em conta três possibilidades:

- se existir conhecimento positivo de uma determinada Questão, esta é **verdadeira**
- se existir conhecimento negativo de uma determinada Questão, esta é **falsa**
- caso não haja conhecimento positivo nem negativo de uma determinada Questão, esta é então **desconhecida**

Com esta extensão à programação em lógica, é preciso usar um predicado que nos indique quando algo é nos desconhecido, já que o *Prolog*, por si só, não consegue fazer. É aqui que surge o predicado demo. Que indica que uma questão é verdadeira, se essa mesma for indicado como tal, na base de conhecimento. O mesmo acontece, quando indica que é falsa, porque encontrou uma

negação forte dessa mesma questão. O predicado *demo*, diz que algo é desconhecido quando não é verdade, mas também não é verdade que exista a sua negação.

```
demo(Questao,verdadeiro):-  
    Questao.  
demo(Questao,falso):-  
    -Questao.  
demo(Questao,desconhecido):-  
    nao(Questao),  
    nao(-Questao).
```

Figura 34: Predicado demo

Quando se fazem varias questões de uma só vez, o predicado *demo* torna-se pouco útil. E assim nasce o predicado *demoL*, que dado uma lista de questões, cria outra lista, com os valores de verdade correspondentes.

```
demoL([],[]).  
demoL([Q|TQ],[R|TR]):- demo(Q,R),demoL(TQ,TR).
```

Figura 35: Predicado demoL

Por fim foi definido o predicado *demComp*, que serve para indicar o valor lógico, de uma disjunção de questões, ou de uma conjunção de questões, sendo que essas questões, por si só, podem ser outra disjunção ou conjunção de mais questões. O resultado final é gerado com a ajuda dos predicados *conjuncao* e *disjuncao*, que são a representação informática das tabelas de verdade respectivas em programação lógica. Só com a diferença da adição do valor desconhecido, que no caso de conjunção, torna-se um elemento absorvente, menos quando conjugado com o falso. Enquanto que na disjunção, desconhecido com verdadeiro, resulta em verdade, e com outro desconhecido, ou com falso, resulta em desconhecido.

```
demoComp(Q1 e Q2,R):-  
    demo(Q1,R1),  
    demoComp(Q2,R2),  
    conjuncao(R1,R2,R).  
demoComp(Q1 ou Q2,R):-  
    demo(Q1,R1),  
    demoComp(Q2,R2),  
    disjuncao(R1,R2,R).  
demoComp(Q1,R):-  
    demo(Q1,R).
```

Figura 36: Predicado demoComp

```

conjuncao(verdadeiro,verdadeiro,verdadeiro).
conjuncao(verdadeiro,desconhecido,desconhecido).
conjuncao(verdadeiro,falso,falso).
conjuncao(desconhecido,verdadeiro,desconhecido).
conjuncao(desconhecido,desconhecido,desconhecido).
conjuncao(desconhecido,falso,falso).
conjuncao(falso,_,falso).

```

Figura 37: Predicado conjuncao

```

disjuncao(verdadeiro,verdadeiro,verdadeiro).
disjuncao(verdadeiro,desconhecido,verdadeiro).
disjuncao(verdadeiro,falso,verdadeiro).
disjuncao(desconhecido,verdadeiro,verdadeiro).
disjuncao(desconhecido,desconhecido,desconhecido).
disjuncao(desconhecido,falso,desconhecido).
disjuncao(falso,verdadeiro,verdadeiro).
disjuncao(falso,desconhecido,desconhecido).
disjuncao(falso,falso,falso).

```

Figura 38: Predicado disjuncao

3.6 Outros Predicados

Nesta secção são usados os mesmos predicados do exercício um, sem nenhuma alteração. Visto que, a inserção ou remoção de conhecimento, com esta nova extensão da programação em lógica mantém-se constante.

```

% Extensao do predicado removeDup: L,R -> {V,F}

removeDup([],[]).
removeDup([X|T],R):-
    pertence(X,T),
    removeDup(T,R).
removeDup([X|T],[X|R]):-
    nao(pertence(X,T)),
    removeDup(T,R).

```

Figura 39: Predicado removeDup

```

% Extensao do predicado quantosTem:A,Lista,Resultado -> {V,F}

quantosTem(A,[],0).
quantosTem(A,[H|T],Resultado):-
    (A == H), quantosTem(A,T,R), Resultado is R+1.
quantosTem(A,[H|T],Resultado):-
    (A \= H), quantosTem(A,T,Resultado).

```

Figura 40: Predicado quantosTem

```
% Extensao do predicado ordenarDecresc: L,Resultado -> {V,F}

ordenarDecresc([X],[X]).
ordenarDecresc([X|Y],T):-
    ordenarDecresc(Y,R),
    insereOrdenado(X,R,T).
```

Figura 41: Predicado ordenarDecresc

```
% -----
% Extensao do predicado solucoes: X,Y,Z -> {V,F}

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

Figura 42: Predicado solucoes

```
% -----
% Extensao do predicado comprimento: Lista, Resultado -> {V,F}

comprimento(X,Z):-
    length(X,Z).
```

Figura 43: Predicado comprimento

```
% -----
% Extensao do predicado teste: Lista -> {V,F}

teste([]).
teste([I|L]) :-
    I,
    teste(L).
```

Figura 44: Predicado teste

```
% -----
% Extensao do predicado somatorio: lista, resultado -> {V,F}

somatorio([], 0).
somatorio([X|Y], R) :-
    somatorio(Y,G),
    R is X+G.
```

Figura 45: Predicado somatorio

```
% Extensao do predicado pertence: X,L -> {V,F}

pertence(X,[]):- fail.
pertence(X,[X|T]):- X==X.
pertence(X,[H|T]):-
    X\=H,
    pertence(X,T).
```

Figura 46: Predicado pertence

```
% Extensao do predicado nao: Q -> {V,F}

nao(Q):- Q, !, fail.
nao(Q).
```

Figura 47: Predicado nao

```
% Extensao do predicado insereOrdenado: X,L,Resultado -> {V,F}

insereOrdenado((X1,Y1),[],[(X1,Y1)]).
insereOrdenado((X1,Y1),[(X2,Y2)|Z],[ (X1,Y1)|[(X2,Y2)|Z] ]):-
    Y1>Y2.
insereOrdenado((X1,Y1), [(X2,Y2)|Z], [(X2,Y2)|R2]) :-
    Y1<=Y2,
    insereOrdenado((X1,Y1),Z,R2).
```

Figura 48: Predicado insereOrdenado

```
% Extensao do predicado concatena(L1,L2,L3)->{V,F}

concatena([],L2,L2).
concatena([X|L1],L2,[X|L]) :-
    concatena(L1,L2,L).
```

Figura 49: Predicado concatena

4 Conclusões e Sugestões

A resolução deste segundo exercício prático foi bastante importante e enriquecedora, pois permitiu aos membros do grupo perceber e interiorizar melhor os conceitos abordados nas aulas práticas da Unidade Curricular de Sistemas de Representação de Raciocínio e Conhecimento. Deste modo, foram adquiridos conhecimentos acerca de programação em lógica estendida, com a introdução do valor "desconhecido" na base de conhecimento, que levaram à revisão de invariantes, representação de conhecimento e evolução do sistema. O grupo alcançou uma maior consciência da sintaxe de *Prolog* e da ferramenta *Sicstus*, e entendimento para representar conhecimento e construir mecanismos de raciocínio para dar solução a problemas, que certamente serão úteis para o posterior desenvolvimento dos conteúdos da UC.

No que diz respeito à representação de conhecimento, o grupo achou por bem, por forma a representar um sistema de cuidados de saúde mais preciso, adicionar o "recibo". Sobre todo este conhecimento, foram introduzidos os diversos tipos de nulos lecionados, o que requereu algum engenho e criatividade. Para além da representação de conhecimento, seria necessário, também, definir invariantes de modo a manter a consistência e integridade do conhecimento. Assim, o grau de dificuldade desta parte foi razoável, já que o mais labiríntico seria a definição correta dos invariantes, sendo que a equipa ficou contente com os resultados obtidos.

No que toca à evolução do conhecimento e ao sistema de inferência, apesar de alguma dificuldade inicial e necessidade de planear e decidir qual a abordagem a tomar, tornou-se cada vez mais fácil fazê-lo à medida que a experiência era maior. Assim, através da correta divisão de tarefas e revisão por toda a equipa, foi conseguida uma solução simples e eficaz.

Em suma, é feita uma apreciação positiva relativamente ao trabalho realizado, visto que a implementação de todas as funcionalidades propostas foram conseguidas com sucesso. O grupo conseguiu tirar partido dos conhecimentos adquiridos neste projeto, sentido-se capaz de, num contexto futuro, aplicar os conceitos subjacentes de forma eficaz. É evidente que, num outro contexto (como um projeto de grandes dimensões), seria benéfico que fossem implementadas um maior conjunto de funcionalidades adicionais para uma melhor gestão do sistema.

Referências

- [1] [Analide, 2011] ANALIDE, César, NOVAIS, Paulo, NEVES, José,
"Sugestões para a Redação de Relatórios Técnicos",
Relatório Técnico, Departamento de Informática, Universidade do Minho, Portugal, 2011.
- [2] [Bratko, 1986] BRAKTO, Ivan
"PROLOG: Programming for Artificial Intelligence",
British Library of Congress, Grã Bretanha, 2011.