Sistemas de Representação de Conhecimento e Raciocínio (3º ano de MIEI)

Segundo Exercício

Relatório de Desenvolvimento

André Gonçalves (a80368)

João Queirós (a82422)

Luís Alves (a80165)

Rafaela Rodrigues (a80516)

22 de Junho de 2019

Resumo

Este relatório inicia-se com uma breve introdução, especificando o contexto sob o qual foi realizado o trabalho proposto. De seguida, é descrito o resultado final de desenvolvimento, sendo declaradas decisões de desenvolvimento e apresentadas respostas da base de conhecimento de forma a demonstrar as funcionalidades pedidas, nomeadamente: um sistema de inferência, evolução do conhecimento e representação de conhecimento imperfeito e perfeito. Por fim, foram elaboradas algumas conclusões relativas ao trabalho desenvolvido, e apresentadas algumas sugestões para trabalho futuro.

Conteúdo

Lista de Figuras

Lista de Tabelas

Capítulo 1

Introdução

Este trabalho foi desenvolvido no âmbito da Unidade Curricular de Sistemas de Representação de Conhecimento e Raciocínio, que se insere no 2º semestre do 3º ano do Mestrado Integrado em Engenharia Informática.

É o segundo de 3 exercícios propostos que constituem o trabalho prático da UC.

Para este segundo exercício foi proposto, tal como para o primeiro, o desenvolvimento de um sistema de representação de conhecimento e raciocínio com capacidade para caracterizar um universo de discurso na área da prestação de cuidados de saúde.

Com este trabalho prático pretende-se estender o universo criado anteriormente, de forma a ser possível trabalhar numa extensão à programação em lógica, recorrendo a valores nulos que representem conhecimento imperfeito.

Capítulo 2

Descrição do Trabalho e Análise de Resultados

2.1 Conhecimento Positivo e Negativo

O conhecimento positivo existente na base de conhecimento, é igual ao que foi desenvolvido para o exercício anterior, pelo que a sua descrição se omite aqui.

No entanto, para este segundo exercício, foi desenvolvido um novo tipo de conhecimento: negativo. Este é representado através da preposição '-'. Assim, um determinado predicado apenas poderá ser considerado falso (isto é, tem conhecimento negativo), se não existir uma prova de que este é verdadeiro e não existir uma exceção a esse predicado. Caso isso aconteça, então estamos perante um caso em que não é possível determinar se o conhecimento é positivo ou negativo, e que será abordado mais à frente.

Para cada um dos 3 predicados (utente, prestador e cuidado), foi elaborada uma definição de conhecimento negativo generalizada, que se apresenta de seguida para o prestador (as outras duas são similares):

```
-prestador( Id , Nome , Esp , Inst) :-
    nao(prestador(Id,Nome,Esp,Inst)),
    nao(excecao(prestador(Id,Nome,Esp, Inst))).
```

Figura 2.1: Indicação de conhecimento negativo de prestador

Para além disso, também poderá ser indicado conhecimento negativo da seguinte forma:

```
-prestador( 41 , 'Joaquim da Graca' , 'Urologia' , 'Hospital de Guimaraes').
```

Figura 2.2: Exemplo de conhecimento negativo explícito

Com o predicado acima, é possível concluir que é mentira que o prestador com ID 41, chamado Joaquim da Graça preste Urologia no Hospital de Guimarães.

De forma a ser possível obter as respostas a estas questões, foi necessário desenvolver um sistema de inferência que indicasse se a resposta seria verdadeira, falsa ou desconhecida. É o que se apresenta na secção seguinte.

2.2 Sistema de inferência

De forma a dar resposta à necessidade de termos 3 valores possíveis de resposta (verdadeiro, falso e desconhecido), foi elaborado o seguinte sistema de inferência.

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

Figura 2.3: Sistema de Inferência

No entanto, este sistema não permite obter informação relativa a disjunções ou conjunções, pelo que foi estendido para as suportar.

```
si( e(Q1,Q2) , Resposta ) :-
    si(Q1 , R1), si(Q2, R2),
    e(R1,R2,Resposta), !.
si( ou(Q1,Q2) , Resposta) :-
    si(Q1 , R1), si(Q2, R2),
    ou(R1,R2,Resposta), !.
```

Figura 2.4: Sistema de Inferência estendido

Para os dois casos, foi necessário colocar um *cut* no fim de cada predicado, uma vez que não queremos reavaliar o sistema de inferência para um caso que não aquele que foi inicialmente pretendido, isto é, não quereríamos que uma conjunção pudesse vir a ser avaliada como uma disjunção.
Em relação aos predicados `e` e `ou`, estes tiveram por base as seguintes tabelas:

Questão 1	Questão 2	Resultado
Verdadeiro	Verdadeiro	Verdadeiro
Verdadeiro	Desconhecido	Desconhecido
Desconhecido	Verdadeiro	Desconhecido
Falso	Falso	Falso
Falso	Desconhecido	Falso
Falso	Verdadeiro	Falso
Desconhecido	Falso	Falso
Verdadeiro	Falso	Falso

Tabela 2.1: Tabela de resultados do e

É possível assim efetuar à base de conhecimento várias questões avulso, com uma conjunção ou uma disjunção.

Questão 1	Questão 2	Resultado
Verdadeiro	Verdadeiro	Verdadeiro
Verdadeiro	Desconhecido	Verdadeiro
Verdadeiro	Falso	Verdadeiro
Desconhecido	Verdadeiro	Verdadeiro
Falso	Verdadeiro	Verdadeiro
Falso	Falso	Falso
Falso	Desconhecido	Desconhecido
Desconhecido	Falso	Desconhecido
Desconhecido	Desconhecido	Desconhecido

Tabela 2.2: Tabela de resultados do ou

2.3 Conhecimento Imperfeito

2.3.1 Conhecimento Incerto

Caso exista conhecimento do tipo incerto, então isso significa que se desconhece o valor de um determinado parâmetro de um predicado, sendo que esse valor é parte de um conjunto indeterminado.

Por exemplo, não se sabe qual a especialidade que Hugo Aníbal (com ID 10) presta, uma vez que não foi preenchido o formulário 37423_B aquando da sua entrada a prestador no Hospital da Luz. Essa informação poderá ser representada da seguinte forma:

```
prestador( 10 , 'Hugo Anibal' , nulo(incerto) , 'Hospital da Luz').
excecao(prestador(Id,Nome,Especialidade,Instituicao)) :-
    prestador(Id,Nome,nulo(incerto), Instituicao).
```

Figura 2.5: Conhecimento incerto sobre um prestador

A introdução da exceção é importante para o sistema de inferência anteriormente desenvolvido.

Assim, quando o invocarmos questionando-o sobre uma especialidade do Hugo Aníbal, a resposta será *desconhecido*, uma vez que não há conhecimento perfeito positivo ou negativo que permita verificar a veracidade positiva ou negativa da questão. Tal pode ser verificado na imagem que se segue:

```
| ?- si(prestador(10,'Hugo Anibal','Genecologia','Hospital da Luz'),X).
X = desconhecido ?
yes
```

Figura 2.6: Resposta a questão sobre conhecimento do tipo incerto

2.3.2 Conhecimento Impreciso

No caso de existir conhecimento do tipo impreciso, então isso significa que o valor de um determinado parâmetro de um predicado pertence a um conjunto finito de valores. Por exemplo, quando o prestador com o ID 13 foi registado, não foi registado o seu nome. Posteriormente foi adicionado que poderia ser João Teixeira, ou Alberto Ricardo, uma vez que foram encontradas assinaturas com estes dois nomes em vários papéis. Essa informação pode ser representada da seguinte forma:

```
excecao(prestador( 13 , 'Joao Teixeira' , 'Cardiologia' , 'Hospital de Faro')).
excecao(prestador( 13 , 'Alberto Ricardo' , 'Cardiologia' , 'Hospital de Faro')).
```

Figura 2.7: Conhecimento impreciso sobre um prestador

Assim, quando se usa o sistema de inferência para questionar a base de conhecimento, a resposta à questão de se o prestador com ID 13 se chama João Teixeira e é especialista em Cardiologia no Hospital de Faro será desconhecida. No entanto, caso inquiramos se o nome é João Adalberto, a resposta será falsa, uma vez que não faz parte do conjunto de valores possivelmente verdadeiros do predicado. Isto pode ser comprovado nas imagens que se seguem.

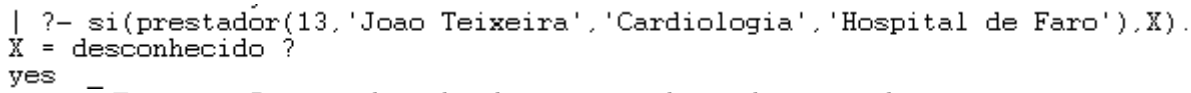

Figura 2.8: Resposta desconhecida a questão sobre conhecimento do tipo impreciso

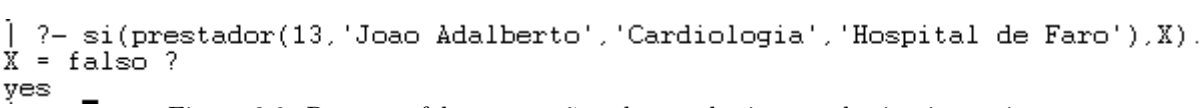

Figura 2.9: Resposta falsa a questão sobre conhecimento do tipo impreciso

Para além disso, também foi desenvolvido um exemplo de um predicado cujo valor poderá oscilar dentro de um intervalo. Por exemplo, a utente com ID 20, chamada Alexandra que mora na Avenida 25 de Abril em Santarém, não indicou a sua idade. No entanto, a funcionária que procedeu ao seu registo indicou que, pela sua aparência, teria entre 18 a 23 anos de idade. Este caso é representado na base de conhecimento da forma que se segue.

```
excecao(utente( 20,'Alexandra',I,'Avenida 25 de Abril, Santarem')) :-
    I >= 18 , I =< 23.
```

Figura 2.10: Conhecimento impreciso sobre um utente relativo a um intervalo

Tendo este predicado, poderemos usar o sistema de inferência para obter informação útil relativamente à utente 20. Se questionarmos se esta tem uma idade dentro do intervalo 18 a 23, a resposta será desconhecida. Mas se questionarmos se esta tem uma idade fora deste intervalo, a resposta será falsa, uma vez que temos a certeza que estará dentro desse intervalo. Isto é comprovado nas figuras que se apresentam.

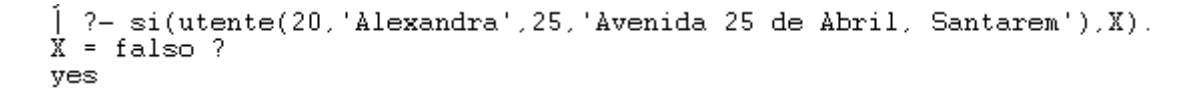

Figura 2.11: Resposta falsa a questão sobre conhecimento intervalar do tipo impreciso

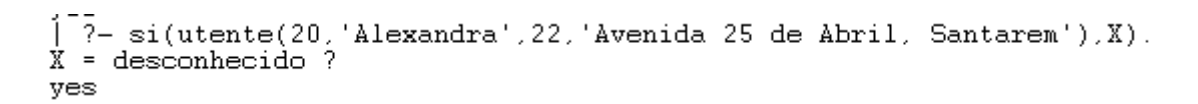

Figura 2.12: Resposta desconhecida a questão sobre conhecimento intervalar do tipo impreciso

2.3.3 Conhecimento Interdito

No caso de existir conhecimento do tipo interdito, então isso significa que o valor de um determinado parâmetro é desconhecido e não poderá vir a ser conhecido, através da evolução da base de conhecimento. Por exemplo, o prestador Adelino Gomes é um prestador VIP, pelo que não permite que ninguém saiba qual a sua especialidade. Isto pode ser representado da seguinte forma:

```
prestador( 14 , 'Adelino Gomes', nulo(interdito), 'Hospital de Santarem').
excecao(prestador(Id, Nome, Especialidade, Instituicao)) :-
    prestador(Id,Nome,nulo(interdito),Instituicao).

+prestador(Id,Nome,Especialidade,Instituicao) :: (
    solucoes(Id, prestador(Id,Nome,nulo(interdito),Instituicao), Lista),
    comprimento(Lista,N),
    N == 0
).
```

Figura 2.13: Conhecimento Interdito sobre um prestador

Assim, caso se inquira a base de conhecimento se o prestador com ID 14, chamado Adelino Gomes é especialista em Cardiologia no Hospital de Santarém, a resposta deverá ser desconhecida. Caso se tente atualizar a base de conhecimento com informação relativa à especialidade do prestador Adelino Gomes, tal não deverá ser permitido. É o que sucede nas figuras que se seguem.

```
| ?- si(prestador(14,'Adelino Gomes','Cardiologia','Hospital de Santarem'),X).
X = desconhecido ?
yes
```

Figura 2.14: Resposta desconhecida a questão sobre conhecimento interdito

```
| ?- evolucao(prestador(14,'Adelino Gomes','Cardiologia','Hospital de Santarem'
),X).
no
```

Figura 2.15: Interdição na evolução de conhecimento interdito

Para além disso, também foram definidas algumas especialidades de prestadores que não podem ser adicionadas ao sistema, através dos predicados que se seguem.

```
interdito('Homeopatia').
interdito('Acupuntura').
interdito('Aromaterapia').

+prestador(Id,Nome,Especialidade,Instituicao) :: (nao(interdito(Especialidade))).
```

Figura 2.16: Especialidades Interditas

Assim, caso se tente adicionar um prestador de uma dessas especialidades, a base de conhecimento não o permitirá, como é possível de constatar na figura que se segue.

```
| ?- evolucao(prestador(69,'Joaquim dos Aromas','Aromaterapia','Clinica de Medi
cinas Alternativas')).
no
```

Figura 2.17: Interdição na evolução de prestador de especialidades interditas

2.4 Evolução do conhecimento

Para ser possível adicionar ou remover conhecimento da base de conhecimento, foram desenvolvidos dois métodos: evolução e involução, sendo que é possível remover conhecimento perfeito positivo ou negativo, e adicionar conhecimento perfeito positivo, e negativo caso não seja impreciso.

Como o desenvolvimento destes predicados envolvia a repetição de vários casos para os diferentes predicados do universo que queremos representar, o grupo optou por demonstrar apenas um: para evolução/involução de um prestador. Os restantes, poderiam ser desenvolvidos de forma análoga.

Nas secções seguintes são explicitados em detalhe.

2.4.1 Evolução

Existem dois casos distintos para a evolução de conhecimento perfeito positivo. A primeira é caso não exista conhecimento imperfeito relativamente ao prestador, e a segunda caso exista. Debrucemo-nos então sobre a primeira.

O predicado que a representa é o seguinte:

```
evolucao( Termo ) :-
    nao(imperfeito(Termo)),
    solucoes( Invariante,+Termo::Invariante,Lista ),
    teste( Lista ),
    assert(Termo).
```

Figura 2.18: Evolução de conhecimento perfeito positivo, não havendo conhecimento imperfeito

Caso se deseje inserir um novo prestador, este não poderá ter conhecimento incerto ou impreciso na base de conhecimento. Para além disso, caso os invariantes sejam verificados, poderá então ser adicionado à base de conhecimento. Os invariantes definidos para a adição de um prestador são os que se indicam de seguida.

```
+prestador(Id,Nome,Especialidade,Instituicao) :: (
    solucoes(Id, prestador(Id,Nome,Especialidade,Instituicao), L),
    comprimento(L,N),
    N == 0
).
+prestador(Id,Nome,Especialidade,Instituicao) :: (
    solucoes(Id, -prestador(Id,Nome,Especialidade,Instituicao), L),
    comprimento(L,N),
    N =< 1
).
+prestador(Id,Nome,Especialidade,Instituicao) ::
    (nao(interdito(Especialidade))).

+prestador(Id,Nome,Especialidade,Instituicao) :: (
    solucoes(Id, prestador(Id,Nome,nulo(interdito),Instituicao), Lista),
    comprimento(Lista,N),
    N == 0
).
```

Figura 2.19: Invariantes relativos à adição de prestadores à base de conhecimento

O primeiro indica que apenas poderá existir um prestador com o mesmo ID na base de conhecimento; o segundo que não deverá haver informação contraditória, isto é, ou é verdade um predicado, ou mentira, não podendo ambos co-existir; o terceiro que não deverá ser possível adicionar prestadores de especialidades interditas e o quarto que não poderá ser adicionada informação sobre um prestador cuja especialidade é interdita.

Para o caso em que exista conhecimento impreciso na base de conhecimento, o predicado usado é o que se apresenta de seguida.

```
evolucao( Termo ) :-
    imperfeito( Termo ),
    si(Termo,desconhecido),
    solucoes(Invariante,+Termo::Invariante, Lista),
    teste(Lista),
    assert(Termo),
    remocaoImperfeito(Termo).
```

Figura 2.20: Evolução de conhecimento perfeito positivo, havendo conhecimento imperfeito

Assim, caso o termo seja imperfeito, isto é, seja do tipo incerto ou impreciso, será verificado se o termo a adicionar for de uma veracidade desconhecida. De seguida, são testados os diversos invariantes relativos à adição desse termo (ver secção acima relativa aos invariantes do prestador). É por fim feita a evolução da base com esse Termo, sendo removidas as referências a conhecimento imperfeito relativo a ele.

Uma implicação desta abordagem, é que se torna interdito adicionar conhecimento que em nada implicaria uma contradição do conhecimento impreciso. Espera-se então que a próxima evolução à base de conhecimento, seja sempre uma que remova as imperfeições lá presentes.

Passando a exemplificar, tentemos adicionar à base de conhecimento o seguinte prestador:

```
prestador(10, 'Hugo Anibal', 'Medicina Geral', 'Hospital da Luz')
```

Tenha-se em consideração o atual estado da base de conhecimento:

```
prestador( 10 , 'Hugo Anibal' , nulo(incerto) , 'Hospital da Luz').
excecao(prestador(Id,Nome,Especialidade,Instituicao)) :-
    prestador(Id,Nome,nulo(incerto), Instituicao).
incerto(prestador(10)).
```

Caso se tente adicionar um prestador com ID 10 e nome e instituição diferentes que as presentes no excerto acima, tal não será permitido. É o que é apresentado nas figuras seguintes.

```
| ?- evolucao(prestador(10,'Hugo Anibal','Ortopedia','Hospital de Braga')).
no
```

Figura 2.21: Tentativa de evolução não permitida

```
| ?- listing(prestador).
prestador(0, 'Joaquim da Graca', 'Urologia', 'Hospital de Braga').
prestador(1, 'Marafa da Derme', 'Dermatologia', 'Hospital de Braga').
prestador(2, 'Manel da Costa', 'Pediatria', 'Hospital de Guimaraes').
prestador(3, 'Serafim Peixoto', 'Gastrenterologia', 'Hospital da Luz').
prestador(4, 'Miguel Dourado', 'Pediatria', 'Hospital da Luz').
prestador(5, 'Estevao Esteves', 'Cardiologia', 'Hospital de Faro').
prestador(6, 'Sousa Costa', 'Gastrenterologia', 'Hospital de Faro').
prestador(6, 'Sousa Costa', 'Otorrinolaringologia', 'Hospital de Faro').
prestador(7, 'Dolores da Cunha', 'Cardiologia', 'Hospital de Faro').
prestador(8, 'Manuel Oliveira', 'Cardiologia', 'Hospital de Faro').
prestador(9, 'Diogo Teixeira', 'Cardiologia', 'Hospital de Faro').
prestador(10, 'Hugo Anibal', nulo(incerto), 'Hospital da Luz').
prestador(11, 'Andre', 'Genecologia', nulo(incerto)).
prestador(14, 'Adelino Gomes', nulo(interdito), 'Hospital de Santarem').
```

Figura 2.22: Base de conhecimento mantém-se inalterada

No entanto, caso se tente adicionar o prestador referenciado acima, a evolução sucede e são removidas todas as referências existentes relativas ao conhecimento imperfeito que existia anteriormente.

```
| ?- evolucao(prestador(10,'Hugo Anibal','Medicina Geral','Hospital da Luz')).
yes
```

Figura 2.23: Evolução de prestador sucede

```
| ?- listing(prestador).
prestador(0, 'Joaquim da Graca', 'Urologia', 'Hospital de Braga').
prestador(1, 'Marafa da Derme', 'Dermatologia', 'Hospital de Braga').
prestador(2, 'Manel da Costa', 'Pediatria', 'Hospital de Guimaraes').
prestador(3, 'Serafim Peixoto', 'Gastrenterologia', 'Hospital da Luz').
prestador(4, 'Miguel Dourado', 'Pediatria', 'Hospital da Luz').
prestador(5, 'Estevao Esteves', 'Cardiologia', 'Hospital de Faro').
prestador(6, 'Sousa Costa', 'Gastrenterologia', 'Hospital de Faro').
prestador(6, 'Sousa Costa', 'Otorrinolaringologia', 'Hospital de Faro').
prestador(7, 'Dolores da Cunha', 'Cardiologia', 'Hospital de Faro').
prestador(8, 'Manuel Oliveira', 'Cardiologia', 'Hospital de Faro').
prestador(9, 'Diogo Teixeira', 'Cardiologia', 'Hospital de Faro').
prestador(11, 'Andre', 'Genecologia', nulo(incerto)).
prestador(14, 'Adelino Gomes', nulo(interdito), 'Hospital de Santarem').
prestador(10, 'Hugo Anibal', 'Medicina Geral', 'Hospital da Luz').
```

Figura 2.24: Base de conhecimento contém informação perfeita sobre o prestador

```
excecao(utente(A,_,B,C)) :-
        utente(A, nulo(incerto), B, C).
excecao(utente(A,B,C,_)) :-
        utente(A, B, C, nulo(incerto)).
excecao(utente(A,B,_,C)) :-
        utente(A, B, nulo(incerto), C).
excecao(prestador(A,B,_,C)) :-
        prestador(A, B, nulo(incerto), C).
excecao(prestador(A,B,C,_)) :-
        prestador(A, B, C, nulo(incerto)).
excecao(cuidado(_,A,B,C,D)) :-
        cuidado(nulo(incerto), A, B, C, D).
excecao(cuidado(A,B,C,D,_)) :-
        cuidado(A, B, C, D, nulo(incerto)).
excecao(cuidado(data(10,7,2019),3,2,'Consulta breve',40)).
excecao(cuidado(data(10,7,2019),3,2,'Consulta breve',60)).
excecao(prestador(13,'Joao Teixeira','Cardiologia','Hospital de Faro')).
excecao(prestador(13,'Alberto Ricardo','Cardiologia','Hospital de Faro')).
excecao(utente(20,'Alexandra',A,'Avenida 25 de Abril, Santarem')) :-
        A>=18,
        A=<23.
excecao(prestador(A,B,_,C)) :-
        prestador(A, B, nulo(interdito), C).
```

Figura 2.25: Exceções relativas ao prestador 10 foram removidas

Caso o conhecimento seja do tipo impreciso, o procedimento é o mesmo. Ainda assim, demonstra-se de seguida.

```
excecao(prestador( 13 , 'Joao Teixeira' , 'Cardiologia' , 'Hospital de Faro')).
excecao(prestador( 13 , 'Alberto Ricardo' , 'Cardiologia' , 'Hospital de Faro')).
impreciso(prestador(13)).
```

Figura 2.26: Exceções relativas a conhecimento impreciso do prestador 13

```
| ?- evolucao(prestador(13,'Alberto Ricardo','Cardiologia','Hospital de Faro'))
.
yes
```

Figura 2.27: Remoção de imprecisão da base de conhecimento

```
| ?- listing(excecao).
excecao(utente(A,_,B,C)) :-
        utente(A, nulo(incerto), B, C).
excecao(utente(A,B,C,_)) :-
        utente(A, B, C, nulo(incerto)).
excecao(utente(A,B,_,C)) :-
        utente(A, B, nulo(incerto), C).
excecao(prestador(A,B,_,C)) :-
        prestador(A, B, nulo(incerto), C).
excecao(prestador(A,B,C,_)) :-
        prestador(A, B, C, nulo(incerto)).
excecao(cuidado(_,A,B,C,D)) :-
        cuidado(nulo(incerto), A, B, C, D).
excecao(cuidado(A,B,C,D,_)) :-
        cuidado(A, B, C, D, nulo(incerto)).
excecao(cuidado(data(10,7,2019),3,2,'Consulta breve',40)).
excecao(cuidado(data(10,7,2019),3,2,'Consulta breve',60)).
excecao(utente(20,'Alexandra',A,'Avenida 25 de Abril, Santarem')) :-
        A>=18,
        A=<23.
excecao(prestador(A,B,_,C)) :-
        prestador(A, B, nulo(interdito), C).
```

Figura 2.28: Exceções relativas ao prestador 13 foram removidas

Por fim, resta mencionar o predicado utilizado para evoluir a base de conhecimento com conhecimento perfeito negativo, caso não exista conhecimento imperfeito relacionado com o predicado que se pretende inserir. Este predicado é similar ao utilizado para o conhecimento perfeito positivo, diferenciando-se apenas nos invariantes que tem de respeitar.

```
evolucao(-Termo) :-
    nao(impreciso(Termo)),
    solucoes(Inv,+(-Termo)::Inv,Lista),
    teste(Lista),
    assert(-Termo).
```

Figura 2.29: Evolução de conhecimento perfeito negativo, não havendo conhecimento imperfeito

Apresentam-se de seguida os invariantes a respeitar.

```
+(-prestador(Id,Nome,Especialidade,Instituicao)) :: (
    solucoes(Id, prestador(Id,Nome,Especialidade,Instituicao), L),
    comprimento(L,N),
    N == 0
).

+(-prestador(Id,Nome,Especialidade,Instituicao)) :: (
    solucoes(Id, -prestador(Id,Nome,Especialidade,Instituicao), L),
    comprimento(L,N),
    N =< 1
).
```

Figura 2.30: Invariantes relativos à evolução de conhecimento perfeito negativo

O primeiro é relativo à impossibilidade de adicionar conhecimento perfeito negativo caso exista conheci-

mento perfeito positivo relativo ao mesmo predicado. O segundo, refere-se à impossibilidade de se adicionar conhecimento perfeito negativo repetido.
Assim, caso estes dois predicados sejam integralmente cumpridos, é possível efetuar a evolução da base de conhecimento com o predicado desejado.

2.4.2 Involução

Para ser possível efetuar a involução da base de conhecimento, os predicados desenvolvidos são idênticos aos do exercício anterior, tendo apenas sido adicionado um novo predicado para remover conhecimento negativo. São estes dois predicados os que se seguem.

```
involucao( T ) :- T,
    solucoes(I, -T::I, LInv),
    teste(LInv),
    retract(T).

involucao(-T) :- -T,
    solucoes(I,-(-T)::I,LInv),
    teste(LInv),
    retract(-T).
```

Figura 2.31: Predicados de involução do conhecimento

Assim, para se remover um dado termo, este apenas necessita de existir e verificar os invariantes definidos. Neste caso, apenas um, que impede a remoção de um prestador se este tiver cuidados associados. É o que se apresenta de seguida.

```
-prestador(Id,_,_,_) :: (
    solucoes(Id, cuidado(_,_,Id,_,_), L),
    comprimento(L,N),
    N == 0
).
```

Figura 2.32: Invariante relativo à involução de conhecimento

Passando à exemplificação de uma involução, pretende-se remover informação relativa ao prestador 4, que não tem qualquer cuidado associado. A involução sucede, e a base de conhecimento fica sem esse registo, tal como demonstrado nas figuras seguintes.

```
| ?- involucao(prestador(4,'Miguel Dourado','Pediatria','Hospital da Luz')).
yes
```

Figura 2.33: Involução do conhecimento bem sucedido

```
| ?- listing(prestador).
prestador(0, 'Joaquim da Graca', 'Urologia', 'Hospital de Braga').
prestador(1, 'Marafa da Derme', 'Dermatologia', 'Hospital de Braga').
prestador(2, 'Manel da Costa', 'Pediatria', 'Hospital de Guimaraes').
prestador(3, 'Serafim Peixoto', 'Gastrenterologia', 'Hospital da Luz').
prestador(5, 'Estevao Esteves', 'Cardiologia', 'Hospital de Faro').
prestador(6, 'Sousa Costa', 'Gastrenterologia', 'Hospital de Faro').
prestador(6, 'Sousa Costa', 'Otorrinolaringologia', 'Hospital de Faro').
prestador(7, 'Dolores da Cunha', 'Cardiologia', 'Hospital de Faro').
prestador(8, 'Manuel Oliveira', 'Cardiologia', 'Hospital de Faro').
prestador(9, 'Diogo Teixeira', 'Cardiologia', 'Hospital de Faro').
prestador(10, 'Hugo Anibal', nulo(incerto), 'Hospital da Luz').
prestador(11, 'Andre', 'Genecologia', nulo(incerto)).
prestador(14, 'Adelino Gomes', nulo(interdito), 'Hospital de Santarem').
```

Figura 2.34: Base de conhecimento após a involução

Caso se tente remover um prestador que tem cuidados associados, como o prestador 1, então a base de conhecimento permanecerá intacta, como comprovado nas figuras que se seguem.

```
| ?- involucao(prestador(1,'Marafa da Derme','Dermatologia','Hospital de Braga'
)).
no
```

Figura 2.35: Tentativa de involução que contraria invariante

```
cuidado(data(5,10,2018), 0, 1, 'cuidado ao Coracao', 20).
```

Figura 2.36: Cuidado associado ao prestador 1

Capítulo 3

Conclusões

Com este trabalho foi possível identificar as diversas problemáticas associadas à representação de conhecimento imperfeito numa base de conhecimento. Para além disso, consideramos que o trabalho desenvolvido permitiu demonstrar todas as funcionalidades requisitadas, apesar de a evolução e involução do conhecimento ter sido apenas elaborada para o caso dos prestadores. Isto deveu-se à similitude entre os predicados utente, prestador e cuidado, pelo que o grupo considerou que para efeitos de demonstração de conhecimento, a elaboração de um era suficiente. Ainda assim, para um trabalho futuro, este poderá passar por definir esses mesmos predicados e por expandir a informação presente na base de conhecimento. Para além disso, também poderá ser desenvolvido um predicado que permita inserir conhecimento negativo quando existe conhecimento imperfeito.