Programação em Lógica estendida e Conhecimento imperfeito

Universidade do Minho, Campus de Gualtar

Bruno Miguel Salgado Machado	74941
Carlos Manuel Magalhães da Silva	75107
Gonçalo Leal da Mota Meireles Moreira	73591
João Rui de Sousa Miguel	74237
Paulo Jorge Machado Guedes	74411

Resumo

Para este trabalho foi-nos solicitado a adição da representação de conhecimento imperfeito ao trabalho realizado anteriormente sobre caracterização de um universo de discurso, na área de prestação de cuidados de saúde. Assim pretende-se que este trabalho contenha todas as funcionalidades da fase anterior, acrescentando-se a representação de conhecimento imperfeito.

Para a representação de conhecimento imperfeito foi necessário criar predicados correspondentes aos três tipos de conhecimento imperfeito (incerto, impreciso e interdito), para os predicados referentes ao utente, ato médico e cuidado prestado, assim como respetivos invariantes dependendo das suas características inerentes. Foi também necessário a alteração de alguns predicados criados na fase anterior nomeadamente os predicados de inserção de conhecimento, entre outros. Adicionalmente foi criado um novo predicado demo, que dá resposta a uma lista de predicados.

Assim, neste relatório, serão abordados os novos predicados criados bem como as alterações feitas nos que transitaram do primeiro exercício e análise de resultados do trabalho efetuado.

Índice

Resumo 1
Índice 2
Índice de Imagens 4
Introdução 5
Descrição do Trabalho 6
Meta-Predicado demoC 6
Meta-Predicado demoD 6
Meta-Predicado mydemo 7
Meta-Predicado demo 7
Meta-Predicado demoLista 7
Meta-Predicado InvolucaoL 8
Predicado exceção 9
Predicado nulo 9
Predicado incerto 10
Predicado impreciso 11
Predicado interdito 11
Predicado registarUtente, registarCuiPrest e registarAmed 13
Invariantes +utente 15
Invariantes -utente 15
Invariantes +cuiprest 16
Invariantes -cuiprest 16
Invariantes +amed 17
Invariantes -amed 17
Análise de Resultados 18
Sistema de Inferência demoLista 18
Predicado incerto 19

Índice

Predicado impreciso 22
Predicado interdito 24
Conclusão 27

Índice de Imagens

Figura 1 - Exemplificação de uma conjunção num caso desconhecido e verdadeiro 18
Figura 2 - Exemplificação de uma disjunção num caso falso e verdadeiro 18
Figura 3 - Exemplificação de uma disjunção num caso falso e falso 18
Figura 4 - Listagem inicial de Utentes no sistema. 19
Figura 5 - Listagem inicial de atos médicos presentes no sistema. 20
Figura 6 - Listagem inicial de cuidados prestados presentes no sistema. 20
Figura 7 - Exempro de inserção de conhecimento incerto. 20
Figura 8 - Exceção para um utente com idade incerta. 21
Figura 9 - Verificação da não alteração do predicado utente. 21
Figura 10 - Registo do cliente, após se conhecer a sua idade. 21
Figura 11 - Verificação do predicado exceção. 21
Figura 12 - Exemplificação do predicado utente quando se introduz novo conhecimento que era imperfeito. 21
Figura 13 - Inserção de conhecimento desconhecido relativo à cidade de um cuidado prestado. 22
Figura 14 - Demonstração de que Coimbra não é opção de cidade válida. 23
Figura 15 - Demonstração de que Braga é uma possível cidade onde este cuidado é prestado. 23
Figura 16 - Introdução de um novo cuidado prestado. 23
Figura 17 - Validação da incerteza relativa á cidade de um cuidado prestado. 23
Figura 18 - Nova listagem de cuidados prestados, após introdução de conhecimento previamente impreciso. 24
Figura 19 - Listagem de exceções relativas ao conhecimento interdito de atos médicos 24
Figura 20 - Demonstração da inserção de conhecimento interdito no predico amed 25
Figura 21 - Demonstração que o preço é desconhecido do ato médico interdito inserido 25
Figura 22 - Listagem dos atos médicos após inserção do conhecimento interdito 25
Figura 23 - Listagem do predicado nulo existente dos utentes, atos médicos e cuidados prestados 26
Figura 24 - Demonstração da negação na atualização do ato médico 26

Introdução

Neste projeto foi-nos solicitado a adição da extensão à programação em lógica ao projeto realizado anteriormente onde se representou um discurso na área da prestação de cuidados de saúde pela realização de cuidados e atos médicos.

Para a representação do conhecimento imperfeito foi decidido a representação de 3 tipos de conhecimento, nomeadamente o conhecimento incerto, impreciso e interdito. Para tal foi necessário a utilização de valores nulos e exceções em conjunto com a criação de novos invariantes e a criação e alteração de predicados presentes no exercício anterior.

Descrição do Trabalho

Meta-Predicado demoC

Este meta-predicado, serve o propósito, de simular a conjunção de valores de verdade. A partir de dois valores booleanos, dá uma resposta, dependendo dos valores inseridos, aplicando a regra da conjunção.

```
demoC(verdadeiro,verdadeiro,verdadeiro).
demoC(verdadeiro,falso,falso).
demoC(verdadeiro,desconhecido,desconhecido).
demoC(falso,falso, falso).
demoC(falso,verdadeiro,falso).
demoC(falso,desconhecido,desconhecido).
demoC(desconhecido,desconhecido,desconhecido).
demoC(desconhecido,verdadeiro,desconhecido).
demoC(desconhecido,falso,desconhecido).
```

Meta-Predicado demoD

Este meta-predicado, serve o propósito, de simular a disjunção de valores de verdade. A partir de dois valores booleanos, dá uma resposta, dependendo dos valores inseridos, aplicando a regra da disjunção.

```
demoD(verdadeiro,verdadeiro,verdadeiro).
demoD(verdadeiro,falso,verdadeiro).
demoD(verdadeiro,desconhecido,verdadeiro).
demoD(falso,falso, falso).
demoD(falso,verdadeiro,verdadeiro).
demoD(falso,desconhecido,falso).
demoD(desconhecido,desconhecido,desconhecido).
demoD(desconhecido,verdadeiro,verdadeiro).
demoD(desconhecido,falso,falso).
```

Meta-Predicado mydemo

Utiliza-se este predicado, para, a partir de duas questões, e com o operador conjunção (“e”) ou disjunção (“v”), calcular a combinação das duas questões, dependendo do operador usado, dando como resposta o resultado dessa combinação.

Meta-Predicado demo

O predicado demo, apresenta uma resposta, mediante uma questão, calculando o valor de verdade dessa mesma questão. Com a inserção de conhecimento imperfeito, surge a necessidade da criação do predicado, o qual requer que seja representado o valor de verdade, Desconhecido. Assim, este predicado demo, poderá responder de três formas: Verdadeiro, Falso, Desconhecido. Este ultimo, que implica que haja uma forma de representar conhecimento que não haja prova contraditória nem válida sobre determinada informação.

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

Meta-Predicado demoLista

O predicado demoLista, uma versão estendida do predicado demo, possibilita a inferência de uma resposta mediante um conjunto de questões apresentadas e separadas por um operador de conjunção e/ou disjunção. Este predicado, utiliza outros predicados como demoC, demoD e mydemo, de forma a que aplicando as suas regras se consiga atingir o objetivo de descobrir o valor de verdade correspondente ao conjunto de questões. De notar que este que as respostas a este conjunto de questões podem ser, também, Verdadeiro, Falso, ou Desconhecido, pelas razões enunciadas acima.

```
demoLista([Q2],R).
demoLista([Q1,e,Q2|T],R)  :-
        mydemo(Q1,e,Q2,R1),
        demoLista([Q2|T],R2),
        demoC(R1,R2,R).

demoLista([Q1,v,Q2|T],R)  :-
        mydemo(Q1,v,Q2,R1),
        demoLista([Q2|T],R2),
        demoD(R1,R2,R).
```

Meta-Predicado InvolucaoL

Este meta-predicado garante que todo o conhecimento imperfeito sob forma de gama de valores é removido aquando a inserção de um novo conhecimento que os venha a substituir. Quando se pretende inserir conhecimento impreciso, dever-se-á inserir uma gama de valores pelos quais estes servirão como opcionais, deste modo, cada valor alternativo adicionará uma exceção nova à nossa base de conhecimento.

Para que todas as exceções adicionadas sejam removidas corretamente, deveremos invocar este predicado com os argumentos chave de cada conhecimento, ou seja, no caso do conhecimento sobre utente, identificamos pelo id de utente, no conhecimento sobre os cuidados prestados, identificamos pelo id do cuidado prestado e por fim, no conhecimento sobre atos médicos, identificamos pelo tuplo (data, id de utente, cuidado prestado) visto que não pode ser realizado ao mesmo utente, mo mesmo dia, o mesmo cuidado prestado.

```
involucaoL(excecao(utente(U,N,I,M))) :- solucoes(excecao(utente(U,X,Y,Z)),
excecao(utente(U,X,Y,Z)), Li),
                                removerL(Li).
involucaoL(excecao(cuiprest(U,D,I,C))) :- solucoes(excecao(cuiprest(U,X,Y,Z)),
excecao(cuiprest(U,X,Y,Z)), Li),
                                removerL(Li).
involucaoL(excecao(amed(D,U,S,P))) :- solucoes(excecao(amed(D,U,S,Z)),
excecao(amed(D,U,S,Z)), Li),
                                removerL(Li).
```

Predicado exceção

O predicado exceção tem o propósito de auxiliar na definição de conhecimento incerto, impreciso e interdito. Sempre que existir uma duvida relativa a, por exemplo, a idade de um utente, poderemos definir uma exceção para essa idade, estabelecendo que será uma exceção ao conhecimento perfeito. Deste modo, sempre que se pretender verificar a idade do utente cuja idade é desconhecida, será ativada a exceção. Isto possibilita não só estipular elementos que são atualmente desconhecidos, como definir um intervalo de valores possíveis, e, definir ainda valores que nunca poderão ser descobertos (com auxilio do predicado nulo, que falaremos de seguida).

```
excecao(utente(U,N,I,M)) :- utente(U,N,idade,M).

excecao(cuiprest(Id,D,I,C)) :- cuiprest(Id,D,instituicao,C).

excecao(amed(D,U,S,P)) :- amed(D,U,S,preco).
```

Predicado nulo

O predicado nulo tem o intuito de auxiliar no processo de definição de conhecimento imperfeito. Este predicado vai ser utilizado sempre que se pretenda definir um determinado elemento como impossível de se vir a atualizar. Para que tal seja possível, o utilizador indica qual argumento quer que seja impossível de se vir a atualizar, e esse argumento passará a

ser o argumento nulo respetivo. Posteriormente é possível verificar se um conhecimento presente na nossa base tem como elementos um atributo nulo, fazendo com que se torne impossível a alteração dos valores a si associados.

```
nulo(idade).
nulo(morada).
nulo(nome).

nulo(descricao).
nulo(instituicao).
nulo(cidade).

nulo(preco).
```

Predicado incerto

Para representarmos conhecimento imperfeito incerto, foi criado o predicado incerto para que seja possível adicionar esse tipo de conhecimento nos utentes, cuidados prestados e atos médicos como vemos no código apresentado abaixo. Para se adicionar conhecimento incerto, diz-se ao programa que o índice do parâmetro que se desconhece ou seja, neste exemplo: incerto (utente(u7, nome('tiago'), 20, morada('rua de nada), [3]), significaria que a idade, seria o parâmetro desconhecido, fazendo-se posteriormente uma evolução da exceção com esse utente, com "_" na idade, o que queria dizer que esse parâmetro não iria ser contabilizado.

Considerou-se que o id do utente, no conhecimento do utente, o id do serviço, no cuidado prestado, e o id do utente, no ato médico, nunca poderão ser desconhecidos.

```
incerto(utente(U,N,I,M), [2]) :- nao(utente(U,_,_,_)),
                                 evolucao(excecao(utente(U,_,I,M))).

incerto(cuiprest(Id,D,I,C), [2]) :- nao(cuiprest(Id,_,_,_)),
                                    evolucao(excecao(cuiprest(Id,_,I,C))).

incerto(amed(D,U,S,P), [1]) :- evolucao(excecao(amed(_,U,S,P))).
```

Predicado impreciso

Para representarmos conhecimento imperfeito impreciso, foi criado o predicado impreciso para que seja possível inserir esse tipo de conhecimento nos utentes, cuidados, prestados e atos médicos como vemos com código apresentado abaixo. Em qualquer dos casos (utentes, etc) é possível adicionar conhecimento impreciso. Para isso é feita uma evolução como exceção do que se quer adicionar, sem modificar quaisquer dos parâmetros. Pode ser adicionado qualquer tipo de conhecimento impreciso. Nos utentes, pode ainda ser acrescentado conhecimento impreciso na idade como gama de valores, ou seja, por exemplo, pode ser adicionado um utente com uma idade superior a 20 anos, inferior a 20 anos ou entre 20 e 30 anos.

```
impreciso(utente(U,N,I,M), I>R) :- nao(utente(U,_,_,_)),
                                   evolucao((excecao(utente(U,N,X,M)) :- X>R)).
impreciso(utente(U,N,I,M), I<R) :- nao(utente(U,_,_,_)),
                                   evolucao((excecao(utente(U,N,X,M)) :- X<R)).
impreciso(utente(U,N,I,M), I>R1, I<R2) :- nao(utente(U,_,_,_)),
                                          evolucao((excecao(utente(U,N,X,M)) :- X>R1, X<R2)).
impreciso(utente(U,N,I,M)) :- nao(utente(U,_,_,_)),
                              evolucao(excecao(utente(U,N,I,M))).

impreciso(cuiprest(Id,D,I,C)) :- nao(cuiprest(Id,_,_,_)),
                                 evolucao(excecao(cuiprest(Id,D,I,C))).

impreciso(amed(D,U,S,P)) :- nao(amed(D,U,S,_)),
                            evolucao(excecao(amed(D,U,S,P))).
```

Predicado interdito

Para se representar conhecimento imperfeito interdito, foi criado o predicado interdito para que seja possível inserir esse tipo de conhecimento nos utentes, cuidados prestados e atos médicos como vemos no código abaixo. Para este procedimento, os parâmetros que se querem interditos, são indicados os indicies onde estes pertencem através de uma lista. Estes parâmetros são substituídos por um valor designado para o propósito de definir a impossibilidade de conhecer determinado valor e introduzido o conhecimento pretendido com

esse valor. É também criada uma exceção com esse valor que será geral para cada parâmetro do conhecimento que se quer inserir. É por fim designado como nulo esse valor.

No caso dos utentes, o id do utente, não pode ser interdito nunca, sendo que os restantes parâmetros o podem ser, tanto ao mesmo tempo como separadamente.

No caso dos cuidados prestados, o id do cuidado terá de ser igualmente também sempre conhecido. No entanto, os restantes parâmetros podem ser interditos.

Finalmente, nos atos médicos, somente o preço poderá ser interdito, o resto dos parâmetros terão de ter a possibilidade de evolução.

```
interdito(utente(U,N,I,M),  [3]) :- nao(utente(U,_,_,_)),
                                    evolucao(utente(U,N,idade,M)).
interdito(utente(U,N,I,M),  [2]) :- nao(utente(U,_,_,_)),
                                    evolucao(utente(U,nome,I,M)).
interdito(utente(U,N,I,M),  [4]) :- nao(utente(U,_,_,_)),
                                    evolucao(utente(U,N,I,morada)).
interdito(utente(U,N,I,M),  [3, 4]) :- nao(utente(U,_,_,_)),
                                      evolucao(utente(U,N,idade,morada)).
interdito(utente(U,N,I,M),  [2, 4]) :- nao(utente(U,_,_,_)),
                                      evolucao(utente(U,nome,I,morada)).
interdito(utente(U,N,I,M),  [2, 3]) :- nao(utente(U,_,_,_)),
                                      evolucao(utente(U,nome,idade,M)).
interdito(utente(U,N,I,M),  [2,3, 4]) :- nao(utente(U,_,_,_)),
                                      evolucao(utente(U,nome,idade,morada)).
```

```
interdito(cuiprest(Id,D,I,C),  [2]) :- nao(cuiprest(Id,_,_,_)),
                                evolucao(cuiprest(Id,descricao,I,C)).
interdito(cuiprest(Id,D,I,C),  [3]) :- nao(cuiprest(Id,_,_,_)),
                                evolucao(cuiprest(Id,D,instituicao,C)).
interdito(cuiprest(Id,D,I,C),  [4]) :- nao(cuiprest(Id,_,_,_)),
                                evolucao(cuiprest(Id,D,instituicao,C)).
interdito(cuiprest(Id,D,I,C),  [2,3]) :- nao(cuiprest(Id,_,_,_)),
                                 evolucao(cuiprest(Id,descricao,instituicao,C)).
interdito(cuiprest(Id,D,I,C),  [2,4]) :- nao(cuiprest(Id,_,_,_)),
                                 evolucao(cuiprest(Id,descricao,I,cidade)).
interdito(cuiprest(Id,D,I,C),  [3,4]) :- nao(cuiprest(Id,_,_,_)),
                                 evolucao(cuiprest(Id,D,instituicao,cidade)).
interdito(cuiprest(Id,D,I,C),  [2,3,4]) :- nao(cuiprest(Id,_,_,_)),
                                   evolucao(cuiprest(Id,descricao,instituicao,morada)).

interdito(amed(D,U,S,C),  [4]) :- nao(amed(D,U,S,_)),
                               evolucao(amed(D,U,S,custo)).
```

Predicado registarUtente, registarCuiPrest e registarAmed

Estes predicados, registarUtente, registarCuiPrest e registarAmed, além do que foi apresentado no exercício anterior, foi acrescentado a involução às suas regras/procedimentos, devido á possibilidade de se evoluir conhecimento imperfeito, nomeadamente, conhecimento imperfeito incerto e impreciso. Desta forma, a involução retirará qualquer exceção que for de encontro ao conhecimento que se quer atualizar como perfeito.

Acrescentamos também regras de evolução para que que tal não seja possível na eventualidade do conhecimento que se pretenda evoluir estiver classificado como conhecimento interdito.

```
registarUtente(U,N,I,M):-  evolucao(utente(U,N,I,M)),
                           N,
                           M,
                           involucaoL(excecao(utente(U,_,_,_))).
registarUtente(U,N,I,M):-  evolucao(utente(U,N,I,M)),
                           N,
                           evolucao(M),
                           involucaoL(excecao(utente(U,_,_,_))).
registarUtente(U,N,I,M):-  evolucao(utente(U,N,I,M)),
                           evolucao(N),
                           M,
                           involucaoL(excecao(utente(U,_,_,_))).
registarUtente(U,N,I,M):-  evolucao(utente(U,N,I,M)),
                           evolucao(N),
                           evolucao(M),
                           involucaoL(excecao(utente(U,_,_,_))).
```

```
registaAMed(D,U,I,C)  :- evolucao(D),
                         evolucao(amed(D,U,I,C)).
                         involucaoL(excecao(amed(D,U,I,_))).

registaAMed(D,U,I,C)  :- evolucao(amed(D,U,I,C)).
                         involucaoL(excecao(amed(D,U,I,_))).
```

```
registarCuiPrest(U,D,I,C):-  evolucao(cuiprest(U,D,I,C)),
                             D,
                             I,
                             C,
                             involucaoL(excecao(cuiprest(U,_,_,_))).
registarCuiPrest(U,D,I,C):-  evolucao(cuiprest(U,D,I,C)),
                             evolucao(D),
                             evolucao(I),
                             evolucao(C),
                             involucaoL(excecao(cuiprest(U,_,_,_))).
```

Invariantes +utente

Este invariante estrutural, foi criado para que na evolução de conhecimento do predicado utente, não seja inserido mais que um utente com o mesmo id.

```
+utente(U,N,I,M) :: (
                    solucoes( U , utente( U , _ , _ , _ ) , S ),
                    comprimento( S , T ),
                    T =< 1
                   ).
```

Invariantes -utente

O primeiro invariante, invariante referencial, apresentado abaixo, foi criado para que não seja possível retirar conhecimento do predicado utente, quando este utente está ligado a um ato médico, para que não haja atos médicos com utentes que já não constam na base de conhecimento.

O segundo invariante, invariante estrutural, apresentado abaixo, foi criado para impor o pressuposto do mundo fechado ao predicado utente. Este invariante diz que se não houver conhecimento positivo nem conhecimento incompleto sobre alguma questão então essa questão é falsa.

```
-utente(U,N,I,M) :: (
                    solucoes( U, amed( D, U, S, P), S),
                    comprimento(S,T),
                    T==0
                   ).
-utente(U,N,I,M) :- nao( utente( U, N, I, M ) ),
                    nao( excecao( utente( U, N, I, M ) ) ).
```

Invariantes +cuiprest

Este invariante estrutural, foi criado para que na evolução de conhecimento do predicado cuiprest (cuidado prestado), não seja inserido mais que um cuiprest com o mesmo id.

```
+cuiprest(U,D,I,C) :: (
                    solucoes( U , cuiprest( U , _ , _ , _ ) , S ),
                    comprimento( S , T ),
                    T =< 1
                   ).
```

Invariantes -cuiprest

O primeiro invariante apresentado abaixo, invariante referencial, foi criado para que não seja retirado um cuidado prestado que esteja ligado a um ato médico. Desta forma não é permitida a existência de um ato médico que tenha um id de cuidado prestado ligado a um cuidado prestado inexistente na base de conhecimento.

O segundo invariante apresentado abaixo, invariante estrutural, impõe o pressuposto do mundo fechado ao predicado cuiprest (cuidado prestado). Este invariante diz que se não houver conhecimento positivo nem conhecimento incompleto sobre alguma questão então essa questão é falsa.

```
-cuiprest(U,D,I,C) :: (
                    solucoes( S , amed( _ , _ , S , _ ) , R ),
                    comprimento( R , T ),
                    T == 0
                   ).
-cuiprest( Id, D, I, C ) :- nao( cuiprest( Id, D, I, C ) ),
                            nao( excecao( cuiprest( Id, D, I, C ) ) ).
```

Invariantes +amed

Este invariante estrutural, foi criado para que na evolução de conhecimento do predicado amed (ato médico), não seja inserido mais que um ato médico com a mesma data, id de utente e id de cuidado prestado, ou seja, considera-se que no mesmo dia não é permitido um ato medico com o mesmo utente e cuidado prestado.

```
+amed(D,U,S,P) :: (
                    solucoes((D, U, S), amed(D, U, S, _), L),
                    comprimento(L, N),
                    N =< 1
                  ).
```

Invariantes -amed

O invariante apresentado abaixo, invariante estrutural, impõe o pressuposto do mundo fechado ao predicado amed (ato médico). Este invariante diz que se não houver conhecimento positivo nem conhecimento incompleto sobre alguma questão então essa questão é falsa.

```
-amed(D,U,S,P) :- nao(amed(D,U,S,P)),
                  nao(excecao(amed(D,U,S,P))).
```

Análise de Resultados

Dar-se-á agora alguns exemplos de predicados utilizados para testar toda a nossa base de conhecimento para esta segunda fase do trabalho. Utilizou-se, uma vez mais, a ferramenta adequada para tal, *SICSTUS*.

Sistema de Inferência demoLista

Este meta-predicado demoLista permite obter resposta a uma lista de conjunções e disjunções de questões. O tipo de respostas possíveis será "verdadeiro", "falso" e "desconhecido". Este meta-predicado assim como o predicado demo, foram desenvolvidos de forma a que deem resposta ao novo tipo de conhecimento inserido, ao conhecimento imperfeito.

Alguns exemplos são apresentados de seguida.

```
| ?- demoLista([amed(data(25,05,2017),u1,c4,100),e,utente(u1,nome('joao miguel'),30,morada('rua de barros'))],R).
R = desconhecido ?
yes
```

Figura 1 - Exemplificação de uma conjunção num caso desconhecido e verdadeiro

```
| ?- demoLista([amed(data(25,05,2017),u1,c9,100),v,utente(u1,nome('joao miguel'),30,morada('rua de barros'))],R).
R = verdadeiro ?
yes
```

Figura 2 - Exemplificação de uma disjunção num caso falso e verdadeiro

```
| ?- demoLista([amed(data(25,05,2017),u1,c9,100),v,utente(u1,nome('joao miguel'),32,morada('rua de barros'))],R).
R = falso ?
yes
```

Figura 3 - Exemplificação de uma disjunção num caso falso e falso

Na figura 1 é apresentada uma lista com uma conjunção de uma questão desconhecida, que advém duma inserção de conhecimento interdito no preço do predicado ato médico e de uma questão verdadeira, referente a um predicado utente já presente na base de conhecimento.

Na figura 2 é apresentada uma lista com uma disjunção de uma questão falsa, pois o id do cuidado prestado é inexistente e de uma questão verdadeira, referente a um predicado utente já presente na base de conhecimento.

Na figura 3 é apresentada uma lista com uma disjunção de uma questão falsa, pois o id do cuidado prestado é inexistente e de uma questão também falsa, pois a idade do utente em questão não corresponde à verdadeira presente na base de conhecimento

Predicado incerto

Quando o utilizador inicializa o sistema, depara-se com casos de testes, pré-carregados, tal como tinha sido efetuado para a primeira fase deste trabalho prático. Sendo que a compatibilidade com a fase anterior está totalmente assegurada, faz-se, de seguida, a síntese de todos os testes realizados após acomodação dos novos casos considerados para representação de conhecimento imperfeito.

Relembre-se assim os casos pré-carregados para os predicados utente, amed e cuiprest. (Figuras 4, 5 e 6)

```
| ?- listing(utente).
utente(u1, nome('joao miguel'), 30, morada('rua de barros')).
utente(u2, nome('bruno machado'), 20, morada('rua de santa apolonia')).
utente(u3, nome('carlos silva'), 20, morada('rua do souto')).
utente(u4, nome('paulo guedes'), 20, morada('rua penedo da forca')).
utente(u5, nome('goncalo moreira'), 20, morada('rua padre francisco babo')).
```

Figura 4 - Listagem inicial de Utentes no sistema.

```
| ?- listing(amed).
amed(data(21,2,2017), u1, c4, 100).
amed(data(21,2,2017), u1, c2, 100).
amed(data(12,1,2017), u2, c5, 10).
amed(data(30,1,2017), u3, c3, 5).
amed(data(24,2,2017), u4, c1, 20).
amed(data(24,2,2017), u3, c1, 20).
amed(data(15,3,2017), u5, c2, 45).
```

Figura 5 - Listagem inicial de atos médicos presentes no sistema.

```
| ?- listing(cuiprest).
cuiprest(c1, descricao('remocao de sinal nas costas'), instituicao('sao joao'), cidade(porto)).
cuiprest(c2, descricao('curativo pos operatorio'), instituicao('unidade local de saude'), cidade(matosinhos)).
cuiprest(c3, descricao(injecao), instituicao('sao joao'), cidade(porto)).
cuiprest(c4, descricao(eletrocardiograma), instituicao('hospital beatriz angelo'), cidade(loures)).
cuiprest(c5, descricao('analise clinica'), instituicao('centro hospitalar'), cidade(braga)).
```

Figura 6 - Listagem inicial de cuidados prestados presentes no sistema.

Teste-se agora o caso em que se pretende introduzir um novo utilizador ao qual se desconhece a sua idade. Para tal dever-se-á utilizar o predicado desenvolvido para inserção de conhecimento incerto. Este possibilitará respostas adequadas para questões que não as possuem. A figura 7 demostra a inserção de um novo utilizador, do qual se desconhece a idade.

```
| ?- incerto( utente(u6,nome('antonio alemao'),idade,morada('praca da republica')) , [3] ).
yes
```

Figura 7 - Exempro de inserção de conhecimento incerto.

Note-se que o segundo argumento do predicado incerto é uma lista contendo os índices para os quais se desconhecem resposta, neste caso seleciona-se a idade, o 3º argumento do predicado utente. De seguida verifica-se a nossa listagem de exceções (Figura 8), indicativa de que para este utente, não se conhece idade. Verifica-se ainda que este não foi introduzido na listagem de utentes (Figura 9), uma vez que não se possui informação relativa a todos os seus atributos (podendo estes vir a ser atualizados).

```
| ?- listing(excecao).

excecao(utente(u6,nome('antonio alemao'),_,morada('praca da republica'))).

yes
```

Figura 8 - Exceção para um utente com idade incerta.

```
| ?- listing(utente).
utente(u1, nome('joao miguel'), 30, morada('rua de barros')).
utente(u2, nome('bruno machado'), 20, morada('rua de santa apolonia')).
utente(u3, nome('carlos silva'), 20, morada('rua do souto')).
utente(u4, nome('paulo guedes'), 20, morada('rua penedo da forca')).
utente(u5, nome('goncalo moreira'), 20, morada('rua padre francisco babo')).
```

Figura 9 - Verificação da não alteração do predicado utente.

Quando finalmente se vir a conhecer a idade deste utilizador, poder-se-á atualizar as informações relativas a este, passando, deste modo, a pertencer à listagem de utentes registados no sistema. Para isto utiliza-se, tal como efetuado para a fase anterior, o predicado registarUtente. Após o registo com toda a informação, o utente deixará de integrar a listagem de exceções. (Figuras 10, 11 e 12).

```
| ?- registarUtente(u6, nome('antonio alemao'), 20, morada('praca da republica')).
yes
```

Figura 10 - Registo do cliente, após se conhecer a sua idade.

```
| ?- listing(excecao).
yes
```

Figura 11 - Verificação do predicado exceção.

```
| ?- listing(utente).
utente(u1, nome('joao miguel'), 30, morada('rua de barros')).
utente(u2, nome('bruno machado'), 20, morada('rua de santa apolonia')).
utente(u3, nome('carlos silva'), 20, morada('rua do souto')).
utente(u4, nome('paulo guedes'), 20, morada('rua penedo da forca')).
utente(u5, nome('goncalo moreira'), 20, morada('rua padre francisco babo')).
utente(u6, nome('antonio alemao'), 20, morada('praca da republica')).
```

Figura 12 - Exemplificação do predicado utente quando se introduz novo conhecimento que era imperfeito.

O mesmo predicado funciona para os casos onde se pretendem introduzir conhecimentos incertos para os atos médicos e para os cuidados prestados, sendo por isso necessário invocar o predicado, passando os argumentos pretendidos. Por exemplo: incerto(amed(data, utente, cuidado, preco), [1, 3]). ou incerto(cuiprest(id, desc, inst, cidade), [4]).

Predicado impreciso

Exemplifica-se agora um dos métodos utilizados para efetuar o teste da representação de conhecimento impreciso. Seja notado, antes de mais, que qualquer conhecimento impreciso está dependente de uma gama de valores pela qual não se tem a certeza de resposta.

Comecemos por testar a introdução de conhecimento impreciso relativo à cidade onde um cuidado prestado poderá ter sido efetuado. Para tal dever-se-á recorrer ao predicado impreciso. A listagem de conhecimento impreciso antecedente à inserção equivale à presente na figura 6. A listagem de conhecimento exceções relacionada com conhecimento impreciso também é igual à demonstrada pela figura 11.

```
| ?- impreciso( cuiprest(c6,descricao('operacao'),instituicao('hospital'),cidade('braga')) ).
yes
| ?- impreciso( cuiprest(c6,descricao('operacao'),instituicao('hospital'),cidade('porto')) ).
yes
| ?- impreciso( cuiprest(c6,descricao('operacao'),instituicao('hospital'),cidade('lisboa')) ).
yes
| ?- impreciso( cuiprest(c6,descricao('operacao'),instituicao('hospital'),cidade('faro')) ).
yes
```

Figura 13 - Inserção de conhecimento desconhecido relativo à cidade de um cuidado prestado.

Pela figura 13 verifica-se que se desconhece se o cuidado c6 foi prestado nas cidades de Braga, Porto, Lisboa ou Faro. Assuma-se agora que se pretende verificar se este cuidado foi realizado num sitio especifico, por exemplo em Coimbra, a resposta deverá ser falso, visto que esta cidade não se encontra na lista de cidades possíveis, no entanto quando se verifica se foi efetuado em qualquer outra cidade (Braga, Porto, Lisboa ou Faro) o resultado será desconhecido, uma vez que não se têm a certeza da sua realização. (Figuras 14 e 15)

```
| ?- demo(cuiprest(c6, descricao('operacao'), instituicao('hospital'), cidade('coimbra')), R).
R = falso ?
yes
```

Figura 14 - Demonstração de que Coimbra não é opção de cidade válida.

```
| ?- demo(cuiprest(c6, descricao('operacao'), instituicao('hospital'), cidade('braga')), R).
R = desconhecido ?
yes
```

Figura 15 - Demonstração de que Braga é uma possível cidade onde este cuidado é prestado.

Imagine-se agora que se descobriu que este cuidado é efetuado em Faro, dever-se-á atualizar a base de conhecimento, efetuando o registo deste cuidado com o predicado registarCuiPrest, que se assegura que as opções alternativas são eliminadas. (Figuras 16, 17 e 18)

```
| ?- registarCuiPrest(c6, descricao('operacao'), instituicao('hospital'), cidade('faro')).
yes
```

Figura 16 - Introdução de um novo cuidado prestado.

```
| ?- demo(cuiprest(c6, descricao('operacao'), instituicao('hospital'), cidade('braga')), R).
R = falso ?
yes
| ?- demo(cuiprest(c6, descricao('operacao'), instituicao('hospital'), cidade('lisboa')), R).
R = falso ?
yes
| ?- demo(cuiprest(c6, descricao('operacao'), instituicao('hospital'), cidade('porto')), R).
R = falso ?
yes
| ?- demo(cuiprest(c6, descricao('operacao'), instituicao('hospital'), cidade('faro')), R).
R = verdadeiro ?
yes
```

Figura 17 - Validação da incerteza relativa á cidade de um cuidado prestado.

```
| ?- listing(cuiprest).
cuiprest(c1, descricao('remocao de sinal nas costas'), instituicao('sao joao'), cidade(porto)).
cuiprest(c2, descricao('curativo pos operatorio'), instituicao('unidade local de saude'), cidade(matosinhos)).
cuiprest(c3, descricao(injecao), instituicao('sao joao'), cidade(porto)).
cuiprest(c4, descricao(eletrocardiograma), instituicao('hospital beatriz angelo'), cidade(loures)).
cuiprest(c5, descricao('analise clinica'), instituicao('centro hospitalar'), cidade(braga)).
cuiprest(c6, descricao(operacao), instituicao(hospital), cidade(faro)).

yes
```

Figura 18 - Nova listagem de cuidados prestados, após introdução de conhecimento previamente impreciso.

Este predicado, tal como o predicado incerto, poderá ser aplicado aos conhecimentos relativos a utentes e a atos médicos. Por exemplo:

1. impreciso(utente(u6, nome('jose'), 20, morada('bairro das tulipas'))).
2. impreciso(utente(u6, nome('jose'), 21, morada('bairro das tulipas'))).
3. impreciso(utente(u6, nome('jose'), 22, morada('bairro das tulipas'))).
4. impreciso(utente(u6, nome('jose'), 23, morada('bairro das tulipas'))).

Sendo a incerteza relativa à sua idade, este será apresentado como uma idade desconhecida, até que se proceda à atualização da base de conhecimento, passando o sistema a ser capaz de calcular a resposta verdadeira para este predicado.

Predicado interdito

Finalmente, demonstrar-se-á o funcionamento do último dos três tipos de conhecimento imperfeito, o conhecimento impreciso. Todo e qualquer conhecimento interdito não será possível atualizar-se depois de inserido.

Apresenta-se agora um caso, a título de exemplo, em que um ato médico tem argumentos interditos na sua construção. Para tal, utiliza-se o predicado interdito para que seja possível adicionar este tipo de conhecimento. A listagem de conhecimento interdito antecedente à inserção equivale à presente na figura 5. A listagem de conhecimento exceções relacionada com conhecimento interdito é representado na figura 19.

```
| ?- listing(excecao).
excecao(amed(A,B,C,_)) :-
        amed(A, B, C, preco).

yes
```

Figura 19 - Listagem de exceções relativas ao conhecimento interdito de atos médicos

```
| ?- interdito(amed(data(25,05,2017),u1,c4,nulo),[4]).
yes
```

Figura 20 - Demonstração da inserção de conhecimento interdito no predico amed

Como se pode verificar na figura 20, o preço no ato médico é interdito, ou seja, nunca poderá ser evoluído e questões sobre esse conhecimento inserido serão desconhecidas por não se saber o preço. Tudo o resto é conhecido. Pretende-se então agora, verificar se é conhecido o preço desse determinado ato médico, o qual deverá ser, como já abordado, desconhecido (figura 21). Podemos também verificar na figura 22, que o conhecimento interdito, é adicionado como conhecimento do ato médico com o valor preco no lugar do preço real pois este não é conhecido. Para este tipo de conhecimento foi criado também o predicado nulo que armazena os parâmetros nulos/ interditos, neste exemplo, do ato médico (figura 23).

```
| ?- demo(amed(data(25,05,2017),u1,c4,100),R).
R = desconhecido ?
yes
```

Figura 21 - Demonstração que o preço é desconhecido do ato médico interdito inserido

```
| ?- listing(amed).
amed(data(21,2,2017), u1, c4, 100).
amed(data(21,2,2017), u1, c2, 100).
amed(data(12,1,2017), u2, c5, 10).
amed(data(30,1,2017), u3, c3, 5).
amed(data(24,2,2017), u4, c1, 20).
amed(data(24,2,2017), u3, c1, 20).
amed(data(15,3,2017), u5, c2, 45).
amed(data(25,5,2017), u1, c4, preco).

yes
```

Figura 22 - Listagem dos atos médicos após inserção do conhecimento interdito

```
| ?- listing(nulo).
nulo(idade).
nulo(morada).
nulo(nome).
nulo(descricao).
nulo(instituicao).
nulo(cidade).
nulo(preco).

yes
```

Figura 23 - Listagem do predicado nulo existente dos utentes, atos médicos e cuidados prestados

Passa-se agora a demonstrar a impossibilidade de evoluir conhecimento sobre o ato médico inserido como interdito com o preço desconhecido (figura 24). De salientar que não deverá ser possível evoluir este conhecimento. O estado final após a tentativa de inserção de conhecimento perfeito sobre o ato médico em questão é representado na figura 22.

```
| ?- registarAmed(data(25,05,2017),u1,c4,100).
no
```

Figura 24 - Demonstração da negação na atualização do ato médico

Por fim é também possível, como já referido inserir conhecimento interdito para os predicados utente e cuiPrest (cuidados prestados). Exemplos disso são:

1. interdito(utente(u7, nome(tiago), nulo, nulo1),[3,4]).
2. Interdito(cuiprest(c7, nulo, instituição('hospital padre américo'), cidade('penafiel')), [2]).

Nestes dois exemplos podemos verificar que no caso do utente a idade e a morada são desconhecidos e no cuiprest a descrição é desconhecida.

Conclusão

Neste trabalho foi nos solicitado a extensão do conhecimento imperfeito á nossa base de conhecimento, realizado no trabalho anterior. Assim sendo, foram criadas várias regras de forma a manter a integridade e a viabilidade do conhecimento do trabalho.

Após concluído, foi adicionado o valor de verdade Desconhecido á representação de conhecimento no trabalho, possibilitando a introdução de conhecimento que não é nem totalmente conhecido, nem totalmente falso. Com isto, possibilitou-se uma maior representação de conhecimento, facilitando o funcionamento do trabalho.

Apesar de algumas dificuldades que surgiram durante a realização do trabalho, pensamos que todos os objetivos propostos pelo enunciado do projeto foram realizados com sucesso.