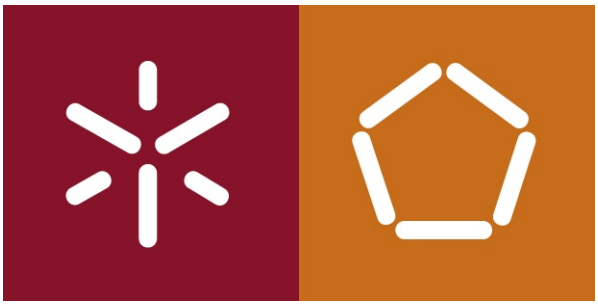
Universidade do Minho

Mestrado Integrado em Engenharia Informática



Sistema de Representação de Conhecimento e Raciocínio

Programação em Lógica Estendida e Conhecimento Imperfeito - Grupo 11

Trabalho realizado por: *Número*

Adriana Martins Gonçalves	A75119
Bruno Alexandre Martins Carvalho	A89476
Eduardo Jorge Santos Teixeira	A84707
Fábio Fernandes Silva	A82331
Tiago Miguel Carvalho e Cunha	A87978

1 *Resumo*

No âmbito da unidade curricular de **Sistemas de Representação de Conhecimento e Raciocínio**, foi-nos proposto a elaboração de um trabalho prático cuja finalidade é desenvolver um sistema capaz de caracterizar um universo de discurso na área da vacinação global da população portuguesa no contexto atual.

Este trabalho encontra-se dividido em duas partes, sendo que este relatório diz respeito à segunda e última fase deste projeto. A realização deste problema terá de ser desenvolvido através da utilização da linguagem de programação *PROLOG*, linguagem essa que foi abordada ao longo de toda a unidade curricular.

Ao longo do relatório iremos explicar as decisões e abordagens que tomamos ao longo do desenvolvimento do projeto, de forma a obter o resultado final desejado.

Conteúdo

2 *Introdução*

No âmbito da Unidade Curricular de Sistemas de Representação de Conhecimento e Raciocínio, foi-nos proposto o desenvolvimento de um projeto, divido em 2 partes, utilizando como linguagem de programação o *PROLOG*, sendo esta uma linguagem lógica e recomendada na representação de conhecimento e construção de mecanismos de raciocínio para a resolução de problemas.

Nesta segunda fase pretende-se que saibamos representar conhecimentos positivos e negativos, representar casos de conhecimento imperfeito (através da utilização de valores nulos de todos os tipos estudados), bem como a manipulação de invariantes que designem restrições à inserção e à remoção de conhecimento do sistema.

Para uma melhor organização de toda a informação, consideramos como base de conhecimento um **Utente**, um **Centro de Saúde**, o **Staff** e a **Vacinação** com adição de **Profissões de Risco** e **Fase**.

3 *Preliminares*

3.1 Representação de Conhecimento Imperfeito

Contrariamente à primeira fase do projeto em que tínhamos um sistema baseado num **mundo fechado** capaz de representar conhecimento perfeito onde a informação representada era única e válida, e as entidades representadas eram as únicas existentes no mundo exterior, nesta etapa pretende-se alterar o pressuposto para um **mundo aberto**, ou seja, implementar um sistema de representação de conhecimento capaz de implementar conhecimento imperfeito. Com isto podemos dizer que nem sempre podemos afirmar que um conhecimento é falso por não fazer parte da base de conhecimento.

Estas alterações permitem aproximar este sistema à realidade, no sentido em que quando não se sabe alguma coisa, apenas se assume essa coisa é **desconhecida**. Esta alteração permitirá catalogar uma resposta de outra forma para além de *verdadeiro* e *falso*. Sendo assim, uma questão *q(x)* poderá ter as seguintes opções como resposta:

- **Verdadeira:** $\exists x : q(x)$
- **Falsa:** $\exists x : \neg q(x)$
- **Desconhecida:** $\neg\exists x : q(x) \vee \neg q(x)$

3.2 Evolução do Conhecimento

Visto que nesta fase do projeto é preciso representar conhecimento imperfeito, houve a necessidade de mudar a forma de desenvolvimento da evolução e retrocesso de conhecimento desenvolvidos na primeira etapa. Para a resolução de todos os pressupostos, foi preciso alterar a nossa base de dados através da adição de conhecimento imperfeito pela utilização de valores nulos de todos os tipos estudados. No caso do conhecimento imperfeito vamos recorrer à definição de exceções, em que, em situações de valores nulos de terceiro tipo, teremos de definir invariantes para garantir que o conhecimento não pode evoluir.

3.3 Conhecimento Negativo

Geralmente um programa de lógica deduz que na falta de informação significa que a mesma é falsa. Apesar disso com uma extensão de um programa em lógica podemos incluir informação falsa explicitamente, assim como explicitar diretamente o pressuposto do mundo fechado para alguns predicados. A negação serve para casos onde não basta provar que uma informação está ausente da base de conhecimento para provar a sua falsidade e há até casos onde se verificam exceções à regra, os quais devem ser incluídos na expressão de negação.

Assim sendo, temos que p(x) é falso se não houver nenhuma prova verdadeira de p(x) nem nenhuma exceção.

4 *Descrição do Trabalho*

4.1 Conhecimento Perfeito

Neste tópico vamos apresentar alguns exemplos para cada predicado relativamente ao conhecimento perfeito.

```
%Conhecimento Perfeito Positivo

utente(1,'1234',eduardo,10/1/1999,'eduardo@mail.com','93436278',fafe,estudante,[asma,diabetes],1).%FASE2
utente(2,'2341',monteiro,07/5/1999,'mach@mail.com','93566278',porto,taxista,[],3).%FASE4
utente(3,'3412',torgal,01/9/1989,'dsaw@mail.com','93436988',guimaraes,medico,[],2).%FASE4
utente(4,'4123',mariana,19/3/1979,'ds@mail.com','91236278',braga,tenista,[asma,hipertensao],1).%FASE2
utente(5,'1124',ze,10/2/1972,'opisa@mail.com','91244012',chaves,camionista,[bronquite],3).%FASE2

utente(6,'1134',margarida,08/12/1953,'opisa@mail.com','91244536',lisboa,policia,[asma],3).%FASE 1
utente(7,'1572',daniela,15/03/1970,'opisa@mail.com','91243671',felgueiras,medica,[hipertensao],3). %FASE2
utente(8,'1002',martim,24/1/1978,'opisa@mail.com','91240116',chaves,enfermeiro,[],3).%FASE 3

utente(9,'2564',roberta,11/12/1973,'opisa@mail.com','91244536',lisboa,medica,[asma],3).%FASE 2
utente(10,'7072',david,13/03/1960,'opisa@mail.com','91243671',felgueiras,enfermeiro,[hipertensao],3).%FASE2
utente(11,'4102',manuel,20/01/1948,'opisa@mail.com','91240116',chaves,policia,[],3).%FASE1
```

Figura 1: Conhecimento Perfeito dos utentes

```
%Conhecimento Perfeito Positivo
centro_saude(1,'Alto_Ave',povoa,'234145167','alto@mail.com').
centro_saude(2,'Centro saude',fafe,'254458167','saude@mail.com').
centro_saude(3,'Chaves Centro',chaves,'234145006','ch@mail.com').
centro_saude(4,'Centro_L',lisboa,'254171267','za@mail.com').
centro_saude(5,'Riba Centro',lisboa,'250878167','riza@mail.com').
```

Figura 2: Conhecimento Perfeito dos centros de saúde

```
%Conhecimento Perfeito Positivo
staff(1,1,rita,'rti@mail.com').
staff(2,1,gilberto,'gilberto@mail.com').
staff(3,1,monica,'saledge@mail.com').
staff(1,2,joao,'juanito@mail.com').
staff(2,2,andreia,'zeca@mail.com').
staff(3,2,belo,'li4@mail.com').
staff(1,3,rui,'salmon@mail.com').
staff(2,3,ze,'salmonze@mail.com').
staff(3,3,ana,'cena@mail.com').
staff(1,4,guilherme,'ti@mail.com').
staff(2,4,carolina,'carol@mail.com').
```

Figura 3: Conhecimento Perfeito da staff

```
%Conhecimento Perfeito Positivo
vacinacao_Covid(5,6,14/02/21,pfeizer,1,1).
vacinacao_Covid(5,6,21/04/21,pfeizer,2,1).
vacinacao_Covid(1,11,01/03/21,pfeizer,1,1).
vacinacao_Covid(1,11,01/04/21,pfeizer,2,1).
vacinacao_Covid(3,1,21/03/21,pfeizer,1,2).
vacinacao_Covid(3,4,21/03/21,pfeizer,1,2).
vacinacao_Covid(3,4,21/04/21,pfeizer,2,2).
```

Figura 4: Conhecimento Perfeito da vacinação covid

```
%Conhecimento Perfeito Positivo
fase(6,1).
fase(11,1).
fase(1,2).
fase(4,2).
fase(5,2).
fase(7,2).
fase(9,2).
fase(10,2).
fase(8,3).
fase(2,4).
fase(3,4).
```

Figura 5: Conhecimento Perfeito das fases

Estes exemplos diferem dos que tínhamos na fase anterior apenas no contradomínio, uma vez que agora existe um terceiro valor de verdade. Debruçando-nos agora na capacidade de inserção de conhecimento negativo, apresentamos de seguida alguns exemplos de aplicação deste tipo de conhecimento em cada um dos predicados:

```
%Conhecimento Perfeito Negativo
-utente(12,'4444',rui,10/10/1993,'ruimanuel@email.com','912789234',fafe,bombeiro,[],4).
-utente(13,'4044',eva,03/7/1953,'evacoates@email.com','912509234',viseu,deputada,[],1).

%Conhecimento Perfeito Negativo
-centro_saude(6,'Clipovoa',povoa,'253498167','povoa@mail.com').

%Conhecimento Perfeito Negativo
-staff(2,4,ramiro,'ramen@mail.com').
-staff(3,4,valentina,'tina@mail.com').

%Conhecimento Perfeito Negativo
-vacinacao_Covid(3,5,21/04/21,pfeizer,1,2).
-vacinacao_Covid(3,5,21/06/21,pfeizer,2,2).

%Conhecimento Perfeito Negativo
-fase(11,2).
```

Figura 6: Conhecimento Perfeito Negativo

Com estes predicados, estamos a afirmar explicitamente que aqueles utentes, centros de saúde, staff, vacinação do covid e a fase não constam da base de conhecimento. Para os predicados utentes, centros de saúde, staff, vacinação do covid e a fase, recorremos ao Pressuposto do Mundo Fechado para explicitar que um utente/centro de saude/staff/vacinação do covid/-fase que não estejam definidos na base de conhecimento não existem, tirando em casos de exceção, que devem ser definidos.

```
% Pressuposto do mundo fechado para o predicado utente
-utente(IDU,NSEG,NOME,DNASC,EMAIL,TEL,MOR,PROF,DOEN,IDCENTRO):-
                            nao(utente(IDU,NSEG,NOME,DNASC,EMAIL,TEL,MOR,PROF,DOEN,IDCENTRO)),
                            nao(excecao(utente(IDU,NSEG,NOME,DNASC,EMAIL,TEL,MOR,PROF,DOEN,IDCENTRO))).


% Pressuposto do mundo fechado para o predicado centro_saude
-centro_saude(IDCENTRO,NOME,MOR,TEL,EMAIL):-
                            nao(centro_saude(IDCENTRO,NOME,MOR,TEL,EMAIL)),
                            nao(excecao(centro_saude(IDCENTRO,NOME,MOR,TEL,EMAIL))).


% Pressuposto do mundo fechado para o predicado staff
-staff(IDSTAFF, IDCENTRO, NOME, EMAIL):-
                            nao(staff(IDSTAFF, IDCENTRO, NOME, EMAIL)),
                            nao(excecao(staff(IDSTAFF, IDCENTRO, NOME, EMAIL))).

-vacinacao_Covid(STAFF, UT, DATA, VACINA, TOMA, FASE):-
                            nao(vacinacao_Covid(STAFF, UT, DATA, VACINA, TOMA, FASE)),
                            nao(excecao(vacinacao_Covid(STAFF, UT, DATA, VACINA, TOMA, FASE))).

-fase(IDU,FASE):-
            nao(fase(IDU,FASE))  ,
            nao(excecao(fase(IDU,FASE))).
```

Figura 7: Pressuposto do Mundo Fechado

4.2 Representação de Informação Incompleta

Este trabalho realiza-se no sentido de dar a um sistema de representação de conhecimento características capazes de o fazer raciocinar segundo o pressuposto do mundo aberto, de forma a permitir tratar informação incompleta. Para abordar a representação de informação incompleta, reparamos que há três possíveis conclusões para uma questão: verdadeira, falsa ou, quando não se pode concluir nenhuma das anteriores, desconhecida. Assim, representamos as possíveis respostas a uma dada questão através do predicado inf (de inferência), apresentado de seguida:

```
% Extensao do meta-predicado inf: Questao,Resposta
%                                 Resposta = {verdadeiro,falso,desconhecido}
% capaz de responder a uma única questão


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

Figura 8: predicado de inferência

O predicado anterior consiste num sistema de inferência que aceita como argumento apenas uma questão. De forma a estender este mecanismo de raciocínio, definimos também um sistema capaz de responder a um conjunto de questões, apresentando um resultado baseado na conjunção e disjunção dos resultados individuais de cada questão. Para isto acontecer criamos também um sistema lógico novo para permitir fazer conjunções e disjunções com a opção de ser desconhecido, como se pode ver nas figuras seguintes:

```
%- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
% Extensao do meta-predicado infConjDisj: [Questao], Resposta -> {V,F,D}
% capaz de fazer a conjunção e/ou disjunção de uma lista de questões
% produzindo um resultado final
infConjDisj([],verdadeiro).
infConjDisj([Questao],R) :- inf(Questao,R).
infConjDisj([Questao1,'AND'|Questoes],R) :- inf(Questao1,R1),
                            infConjDisj(Questoes,R2),
                            conjuncao(R1,R2,R).
infConjDisj([Questao1,'OR'|Questoes],R) :- inf(Questao1,R1),
                           infConjDisj(Questoes,R2),
                           disjuncao(R1,R2,R).
```

Figura 9: Predicado que conjuga disjunções e conjunções de Questões

```
% Extensao do predicado conjuncao: X,Y -> {V,F,D}
conjuncao(verdadeiro,verdadeiro,verdadeiro).
conjuncao(verdadeiro,desconhecido,desconhecido).
conjuncao(desconhecido,verdadeiro,desconhecido).
conjuncao(desconhecido,desconhecido,desconhecido).
conjuncao(falso,_,falso).
conjuncao(_,falso,falso).

% Extensao do predicado disjuncao: X,Y -> {V,F,D}
disjuncao(verdadeiro,_,verdadeiro).
disjuncao(_,verdadeiro,verdadeiro).
disjuncao(falso,falso,falso).
disjuncao(falso,desconhecido,desconhecido).
disjuncao(desconhecido,falso,desconhecido).
disjuncao(desconhecido,desconhecido,desconhecido).
```

Figura 10: Predicado novo da conjunção e disjunção

4.3 Conhecimento Imperfeito

Neste capitulo vamos abordar conceito de conhecimento imperfeito, bem como a definição das regras para a inserção e remoção de elementos na base de conhecimento. Quanto a este tipo de conhecimento podemos destacar os tipos incerto, impreciso e interdito.

Cada tipo de conhecimento imperfeito tem regras que são precisas definir para a inserção deste tipo de informação na base de conhecimento. No nosso projeto vamos focar-nos mais no tipo incerto sendo que vamos explicar mais a frente no que consiste e abordar superficialmente o que os outros tipos são.

4.3.1 Tipo Incerto

Este tipo de conhecimento caracteriza-se por definir casos que são considerados exceções à realização da informação negativa. Nestas situações, há um argumento de um predicado que não se conhece, logo não pertence a um determinado conjunto de valores. Assim sendo, esse valor será definido como uma exeção. Um exemplo desse tipo de conhecimento pode ser visualizado na imagem abaixo:

```
(excecao(utente(IDU,NSEG,NOME,DNASC,EMAIL,TEL,MOR,PROF,DOEN,IDCENTRO)) :- utente(IDU,NSEG_DESC,NOME,DNASC,EMAIL,TEL,MOR,PROF,DOEN,IDCENTRO)
```

O exemplo acima diz respeito a uma exceção para um utente com NSEG (número da segurança social) desconhecido.

4.3.2 Tipo Impreciso

Este tipo de conhecimento caracteriza-se por um conjunto de valores bem definidos, isto é, existe um determinado conjunto de valores possíveis para um dos atributos de uma instância de um predicado, e desconhece-se quais desses valores realiza a questão.

4.3.3 Tipo Interdito

Este terceiro tipo de conhecimento caracteriza-se por um tipo de dados que para além de não se conhecerem, não é suposto que apareçam na base de conhecimento. Para além disso, é necessário que não seja possível haver nenhuma inclusão de informação positiva e negativa na base de conhecimento que vá contra a condição importa pelo valor nulo não permitido.

4.4 Evolução do Conhecimento

Com a finalidade de estender o nosso programa em lógica, é necessário haver a inserção de conhecimento imperfeito, o qual tem três formas: conhecimento incerto, conhecimento impreciso e conhecimento interdito.

4.4.1 Conhecimento Incerto

Este tipo de conhecimento imperfeito consiste em conhecimento que é desconhecido e pertence a um conjunto indeterminado de hipóteses.

4.4.2 Conhecimento Impreciso

O conhecimento imperfeito impreciso refere-se a um conhecimento desconhecido, mas dentro de um conjunto determinado de hipóteses. A diferença entre este tipo de conhecimento para o anterior é que o seu valor nulo, apesar de ser desconhecido, está dentro de um conjunto de valores, daí este não ser um conhecimento que tenha rigor. Apesar disto, como na nossa base de conhecimento não temos nenhuma informação imprecisa, não houve necessidade de adicionar nenhum predicado para a inserção deste tipo de informações.

4.4.3 Conhecimento interdito

Além dos tipo de conhecimento acima falados, temos também o conhecimento imperfeito interdito, que é aquele onde existe um conhecimento desconhecido, mas, tal como o nome diz, interdito, ou seja, que não é possível conhecer. No nosso caso, a nossa base de conhecimento também não tem informação com conhecimento interdito, daí também não haver necessidade de criar nenhum predicado de inserção deste tipo de conhecimento.

5 *Conclusões e Sugestões*

Durante a elaboração de todo o trabalho pratico da unidade curricular de Sistemas de Representação de Conhecimento e Raciocínio, foram vários os desafios com que nos deparamos na tentativa de obter um resultado final o mais fidedigno e consistente possível. Encontramos um problema no código que não conseguimos resolver até à data e que, consequentemente, nos impediu de testar o nosso programa, mas, apesar disso, achamos que o nosso desempenho foi positivo, pois conseguimos ultrapassar quase todos os desafios propostos para esta fase. Nesta fase, o principal foco do grupo foi alterar todo o pressuposto de mundo fechado que tínhamos desenvolvido na primeira parte, para o pressuposto de mundo aberto.

Este trabalho permitiu ainda aperfeiçoar e aprofundar os conhecimentos obtidos ao longo desta unidade curricular, bem como uma maior consolidação de conhecimentos no que diz respeito a linguagem de programação lógica *PROLOG*.

6 *Referências*

Para o desenvolvimento deste trabalho utilizamos como referência os conhecimentos adquiridos durante as aulas teóricas e praticas da UC, bem como todos os materiais disponibilizados na blackboard tal como slides e apontamentos.

7 *Anexos*

Ao longo do desenvolvimento desta fase do projeto, foi necessária a criação de funções auxiliares de forma a simplificar a abordagem a tomar e assim obtermos mais facilmente os resultados que nos foram propostos.

```
%Extensão do predicado que permite a evolucao do conhecimento

% Evolução - insercao de novo Termo
evolucao(Termo):- findall(Invariante,+Termo::Invariante,Lista),
    insercao(Termo),teste(Lista).


insercao(Termo):- assert(Termo).
insercao(Termo):- retract(Termo), !, fail. %cut operator para

%Involucao - remocao de Termo

involucao(Termo):- findall(Invariante,-Termo::Invariante,Lista),
    remocao(Termo),
    teste(Lista).

remocao(Termo):-retract(Termo).
remocao(Termo):-retract(Termo), !,fail.
```

```
% Insere novo conhecimento na base de conhecimento
evolucao(T) :-
    findall(I, +T::I, Linv),
    insercao(T),
    teste(Linv).

% Insere conhecimento perfeito positivo na base de conhecimento
evolucao(T, positivo) :-
    findall(I, +T::I, Linv),
    insercao(T),
    teste(Linv).

% Insere conhecimento perfeito negativo na base de conhecimento
evolucao(T, negativo) :-
  findall(I, +(-T)::I, Linv),
    insercao(-T),
    teste(Linv).
```