Universidade do Minho
Escola de Engenharia

Sistemas de Representação Conhecimento e Raciocínio

João Pedro da Santa Guedes A89588
Luís Pedro Oliveira de Castro Vieira A89601
Carlos Miguel Luzia Carvalho A89605
Bárbara Ferreira Teixeira A89610


A89588


A89601


A89605


A89610

9 de abril de 2021

Índice

1 Resumo

Este relatório tem como objetivo documentar o trabalho de grupo proposto na Unidade Curricular de Sistemas Representação de Conhecimento e Raciocínio usando a linguagem PROLOG. Numa fase inicial é apresentada uma introdução do projeto. Posteriormente, é explicado o raciocínio do grupo para cada uma das alíneas do trabalho assim como o conhecimento extra adicionado. Por fim, é feita uma apreciação crítica do trabalho pelo grupo.

2 Introdução

Este relatório tem como objetivo documentar o trabalho de grupo proposto na Unidade Curricular de Sistemas Representação de Conhecimento e Raciocínio usando a linguagem PROLOG. O objetivo pretendido é desenvolver um sistema de representação de conhecimento e raciocínio capaz de caracterizar um universo de discurso na área da vacinação global da população portuguesa no texto da Covid-19.

O sistema desenvolvido deverá ser capaz de gerir a base de conhecimento existente, isto é, permitir a definição de fases de vacinação, definindo critérios de inclusão de utentes nas diferentes fases (e.g., doenças crónicas, idade, profissão), identificar pessoas não vacinadas, vacinadas, vacinadas indevidamente, pessoas candidatas a ser vacinadas e pessoas a quem falta a segunda dose da vacina.

O conhecimento base é composto por utente, centro_saude, staff e vacinacao_Covid.

3 Preliminares

O sistema que foi desenvolvido ao longo deste trabalho pretende caracterizar o universo de discurso na área da vacinação global da população portuguesa no contexto da Covid-19. Para que este projeto pudesse ser iniciado, foi necessária a obtenção de conhecimento base na área. Para tal, este foi adquirido através das aulas frequentadas e resolução das fichas práticas dadas nas mesmas, permitindo um maior contacto com a linguagem utilizada. Foram realizadas pesquisas neste âmbito e ,após a leitura detalhada do enunciado, foi possível a distribuição do trabalho a realizar. Isto possibilitou uma maior organização do grupo e que o trabalho fosse concluído na data de limite proposta.

4 Descrição do Trabalho e Análise de Requisitos

4.1 Base de conhecimento

Neste projeto caracterizamos o conhecimento da seguinte forma:

- utente: Idutente,Nºsegsocial, Nome, Data-Nasc, Email, Telefone, Morada, Profissão, $[Doencas - Crónicas]$, IdCentroSaúde $-> \{V, F\}$

  O Idutente, será um inteiro que identificará este utilizador assim como o número de segurança social; o nome, data de nascimento, email e telefone são também informações respetivas a cada utente. Além disso cada utente terá também uma lista de possíveis doenças crónicas e ainda o IdCentroSaúde, um outro inteiro que identificará o centro de saúde em que o utente esteja registado.

```
1    utente(1,123123123,"Luis",02-02-2000,"luis@gmail.com"
     ,911222333,"braga","estudante",[cego,coxo],1).
2    utente(5,123456789,"Marta",02-02-1993,"marta@gmail.com"
     ,912345677,"algarve","esteticista",[],3).
```

- centro_saude: Idcentro, Nome, Morada, Telefone, Email $-> \{V, F\}$

  O Idcentro, como já referido em cima, é um inteiro que identificará o centro de saúde em causa; o nome, morada, telefone e email são também informações respetivas ao centro de saúde.

```
1    centro_saude(1,"Hospital de Braga","Braga",253027000,"
     hospital_braga@sns.pt").
2    centro_saude(2,"Hospital da Trofa Braga Norte","Braga"
     ,253027002,"hospital_trofa_braga@sns.pt").
3    centro_saude(3,"Hospital de S o Jo o","Porto",225512100,"
     hospital_sao_joao@sns.pt").
```

- staff: Idstaff, Idcentro, Nome, email $-> \{V, F\}$

  O Idstaff e o Idcentro, são inteiros relativos ao membro do staff em causa e ao centro de saúde onde o mesmo atua; o nome e o email são informações adicionais deste membro.

```
staff(1,1,"Jorge","jorge_andrade@gmail.com").
staff(2,2,"Liliana","liliana_albernaz@gmail.com").
staff(3,3,"Andre","martinsdr@gmail.com").
staff(4,1,"Antonio","antonioalves@gmail.com").
staff(5,2,"Ruben","rubenpteixeira@gmail.com").
staff(6,3,"Julian","julianaribeiro@gmail.com").
```

- vacinação_covid: Staff, utente, Data, Vacina, Toma $-> \{V, F\}$

  Contém o ID do staff que deu ou dará a vacina e o do utente que a recebeu ou receberá, a data da toma, a marca da vacina e a toma, sendo a primeira ou a segunda toma.

```
vacinacao_covid(1,1,23-03-2021,pfizer,1).
vacinacao_covid(1,1,31-03-2021,pfizer,2).
```

4.2 Invariantes

Para que a inserção e remoção de conhecimento seja feita devidamente, foram utilizados invariantes para fazer a manutenção da base de conhecimento. Criamos, portanto, invariantes de inserção e de remoção.

Para definir estes invariantes, utilizámos o predicado *solucoes* que devolve todas as soluções que cumprem com determinadas condições que especificamos.

4.2.1 Invariantes de Inserção - Utente

- Verificar se já existe um utente com este ID.

```
+utente( ID,_,_,_,_,_,_,_,_,_ ) :: (solucoes( ID,
      (utente( ID,_,_,_,_,_,_,_,_,_ )), S ),
          comprimento( S,NS ),
      NS == 1
                  ).

```

- Verificar se já existe um utente com este número de segurança social.

```
+utente( _,NSS,_,_,_,_,_,_,_,_ ) :: (solucoes( NSS,
      (utente( _,NSS,_,_,_,_,_,_,_,_ )), S ),
          comprimento( S,NS ),
      NS == 1
                  ).

```

- Verificar se já existe um utente inserido com este email.

```
+utente( _,_,_,_,E,_,_,_,_,_ ) :: (solucoes( E,
      (utente( _,_,_,_,E,_,_,_,_,_ )), S ),
          comprimento( S,NS ),
      NS == 1
                  ).

```

- Verificar se já existe um utente inserido com este número de telefone.

```
+utente( _,_,_,_,_,T,_,_,_,_ ) :: (solucoes( T,
      (utente( _,_,_,_,_,T,_,_,_,_ )), S ),
          comprimento( S,NS ),
      NS == 1
                  ).

```

- Verificar se o centro de saúde inserido existe.

```
+utente( _,_,_,_,_,_,_,_,_,_,IDCS ) :: (solucoes( IDCS,
      (centro_saude( IDCS,_,_,_,_ )), S ),
          comprimento( S,NS ),
      NS == 1
                  ).

```

4.2.2 Invariantes de Inserção - Staff

- Verificar se já existe um staff inserido com este ID.

```
+staff( IDS,_,_,_ ) :: (solucoes( IDS,
      (staff( IDS,_,_,_ )), S ),
          comprimento( S,NS ),
      NS == 1
                  ).

```

- Verificar se o centro de saúde inserido existe.

```
+staff( _,IDCS,_,_ ) :: (solucoes( IDCS,
      (centro_saude( IDCS,_,_,_,_ )), S ),
          comprimento( S,NS ),
      NS == 1
                  ).

```

- Verificar se já existe um staff inserido com este email.

```
+staff( _,_,_,E ) :: (solucoes( E,
      (staff( _,_,_,E )), S ),
          comprimento( S,NS ),
      NS == 1
                  ).

```

4.2.3 Invariantes de Inserção - Centro Saúde

- Verificar se já existe um centro de saúde inserido com este ID.

```
+centro_saude( IDC,_,_,_,_ ) :: (solucoes( IDC,
      (centro_saude( IDC,_,_,_,_ )), S ),
                  comprimento( S,NS ),
      NS == 1
                  ).

```

- Verificar se já existe um centro de saúde inserido com este número de telefone.

```
+centro_saude( _,_,_,T,_ ) :: (solucoes( T,
      (centro_saude( _,_,_,T,_ )), S ),
                  comprimento( S,NS ),
      NS == 1
                  ).

```

- Verificar se já existe um centro de saúde inserido com este email.

```
+centro_saude( _,_,_,_,E ) :: (solucoes( E,
      (centro_saude( _,_,_,_,E )), S ),
                  comprimento( S,NS ),
      NS == 1
                  ).

```

4.2.4 Invariantes de Inserção - Vacinação Covid

- Verificar se existe staff inserido.

```
+vacinacao_covid( IDS,_,_,_,_ ) :: (solucoes( IDS,
      (staff(IDS,_,_,_)), S ),
                  comprimento( S,NS ),
      NS == 1
                  ).

```

- Verificar se existe utente inserido.

```
+vacinacao_covid( _,IDU,_,_,_ ) :: (solucoes( IDU,
      (utente(IDU,_,_,_,_,_,_,_,_,_,_)), S ),
                  comprimento( S,NS ),
      NS == 1
                  ).

```

- Verifica se é a toma correta, isto é, se já tomou a primeira só pode tomar a segunda e se ainda não tomou nenhuma, só pode tomar a primeira.

```
+vacinacao_covid(_,IDU,_,_,T):: (
            (utente(ID,_,_,_,_,_,_,_,_,_)),
            numVacinasUtentes(ID,TS),T=<2,
            T =:= TS
            ).

```

- Verifica se está a tomar a mesma vacina no caso de já ter tomado uma primeira dose.

```
+vacinacao_Covid(_,IDU,_,V,T) :: (
            (T == 1;
            (solucoes(IDU,vacinacao_Covid(_,IDU,_,_,_,V,_),R),
            comprimento(R,N),
            N == 2)
            )).

```

- Verifica se está a tomar a segunda dose depois da primeira.

```
+vacinacao_covid(_,IDU,D-M-A,_,T) ::(
            (T == 1;
            (vacinacao_covid(_,IDU,D2-M2-A2,_,1)),
            T==2,
            (D>D2,M==M2,A==A2;M>M2,A==A2;A>A2)
            )).

```

4.2.5 Invariantes de Remoção - Utente

- Verifica se existe o utente, para não permitir a remoção de conhecimento inexistente.

```
-utente( ID,NSS,N,DN,E,T,M,P,LDC,IDCS ) :: (solucoes( (ID,NSS,N,
   DN,E,T,M,P,LDC,IDCS),
      (utente( ID,NSS,N,DN,E,T,M,P,LDC,IDCS )), S ),
          comprimento( S,NS ),
      NS == 1
                  ).

```

4.2.6 Invariantes de Remoção - Staff

- Verifica se existe o staff, para não permitir a remoção de conhecimento inexistente.

```
-staff( IDS,IDC,N,E ) :: (solucoes( (IDS,IDC,N,E),
      (staff( IDS,IDC,N,E )), S ),
          comprimento( S,NS ),
      NS == 1
                  ).

```

4.2.7 Invariantes de Remoção - Centro Saúde

- Verifica se existe o centro de saúde, para não permitir a remoção de conhecimento inexistente.

```
-centro_saude( IDC,N,M,T,E ) :: (solucoes( (IDC,N,M,T,E),
      (centro_saude( IDC,N,M,T,E )), S ),
          comprimento( S,NS ),
      NS == 1
                  ).

```

4.2.8 Invariantes de Remoção - Vacinação Covid

- Verifica se existe a vacinação, para não permitir a remoção de conhecimento inexistente.

```
-vacinacao_covid( IDS,IDU,D,V,T ) :: (solucoes( (IDS,IDU,D,V,T),
      (vacinacao_covid( IDS,IDU,D,V,T )), S ),
          comprimento( S,NS ),
      NS =< 2
                  )..

```

4.3 Predicados auxiliares

Nesta secção serão apresentados os vários predicados auxiliares aos requisitados neste projeto e auxiliares aos predicados extra que o grupo acrescentou.

- **comprimento** - Este predicado recebe uma lista e devolve o seu comprimento.

```
comprimento( S,N ) :- length( S,N ).

```

- **remove_dups** - Este predicado elimina todos os elementos repetidos de uma lista e devolve uma lista sem repetições.

```
remove_dups([], []).
remove_dups([First | Rest], NewRest) :- member(First, Rest),
    remove_dups(Rest, NewRest).
remove_dups([First | Rest], [First | NewRest]) :- nao(member(
    First, Rest)), remove_dups(Rest, NewRest).

```

- **pertence** - Este predicado verifica o valor de verdade da existência de um elemento numa lista.

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

```

- **adicionar** - Este predicado adiciona um dado elemento a uma dada lista apenas se este elemento não pertencer a essa lista.

```
adicionar(X,L,L):-pertence(X,L).
adicionar(X,L,[X|L]).

```

- **concatenar** - Este predicado concatena duas listas.

```
concatenar(T,[],T).
concatenar([],T,T).
concatenar([H|T],L,F):- adicionar(H,N,F), concatenar(T,L,N).

```

- **intersection** - Este predicado devolve os elementos comuns de duas listas.

```
intersection(A,B,AnB):- subtract(A,B,AminusB),subtract(A,AminusB
    ,K),sort(K,AnB).

```

- **remocao** - Este predicado remove um determinado conhecimento.

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

```

- **teste** - Este predicado testa os invariantes.

```
teste( [] ).
teste( [R|LR] ) :-
    R,
    teste( LR ).

```

- **insercao** - Este predicado insere conhecimento na base de conhecimento.

```
insercao( Termo ) :- assert( Termo ).
insercao( Termo ) :- retract( Termo ),!,fail.

```

4.4 Predicados Essenciais

Nesta secção serão apresentados os vários predicados requisitados para este projeto.

4.4.1 Fases de Vacinação

- **Primeira fase**

  A partir de dezembro de 2020

  - Profissionais de saúde envolvidos na prestação de cuidados a doentes;
  - Profissionais das forças armadas, forças de segurança e serviços críticos;
  - Profissionais e residentes em Estruturas Residenciais para;
  - Pessoas Idosas (ERPI) e instituições similares;
  - Profissionais e utentes da Rede Nacional de Cuidados Continuados Integrados (RNCCI).

  A partir de fevereiro de 2021

  - Pessoas de idade $\geq$ 50 anos, com pelo menos uma das seguintes patologias:
    - Insuficiência cardíaca;
    - Doença coronária;
    - Insuficiência renal (Taxa de Filtração Glomerular ¡ 60ml/min);
    - (DPOC) ou doença respiratória crónica sob suporte ventilatório e/ou oxigenoterapia de longa duração.
  - Pessoas com 80 ou mais anos de idade.

  Predicados

  - **Predicado que define as profissões prioritárias**:

```
1     profissional_prioritario("enfermeiro").
2     profissional_prioritario("enfermeira").
3     profissional_prioritario("medico").
4     profissional_prioritario("medica").
5     profissional_prioritario("militar").
6
```

  - **Predicado que averigua os utentes candidatos à primeira fase de acordo com a sua profissão**:

```
profissional(R):- solucoes((ID,N),(utente(ID,_,N,_,_,_,_,P,_,_),profissional_prioritario(P)),R).


```

– **Predicado que averigua se uma doença de um utente é uma das referidas nos critérios da primeira fase**

```
doenca_1afase([]):-!,fail.
doenca_1afase(["Insuficiencia_cardiaca"|_]).
doenca_1afase(["Insuficiencia_renal"|_]).
doenca_1afase(["Doenca_respiratoria"|_]).
doenca_1afase([H|T]):- doenca_1afase(T).

```

– **Predicado que averigua os utentes candidatos à primeira fase de acordo com a sua idade e a lista de doenças crónicas**

```
morethan50_diseased(RS):- solucoes((ID,N),(utente(ID,_,N,D-M-A,_,_,_,_,_,LDC,_),doenca_1afase(LDC),K is 2021, K-A >= 50),R),remove_dups(R,RS),!.

```

– **Predicado que averigua os utentes candidatos à primeira fase de acordo com a sua idade**

```
morethan80(R):- solucoes((ID,N),(utente(ID,_,N,D-M-A,_,_,_,_,_,LDC,_),K is 2021, K-A >= 80),R).

```

– **Predicado que averigua os utentes candidatos à primeira fase**

```
primeira_fase(RS):- morethan80(X),morethan50_diseased(Y),concatenar(X,Y,L),profissional(Z),concatenar(L,Z,R),remove_dups(R,RS),!.

```

- **Segunda fase**

  A partir de abril de 2021

  – Pessoas de idade $\geq$ 65 anos (que não tenham sido vacinadas previamente);

  – Pessoas entre os 50 e os 64 anos de idade, inclusive, com pelo menos uma das seguintes patologias:

    * Diabetes;
    * Neoplasia maligna ativa;
    * Doença renal crónica (Taxa de Filtração Glomerular ¿ 60ml/min);

* Insuficiência hepática;
* Hipertensão arterial;
* Obesidade.

Predicados

- **Predicado que averigua se uma doença de um utente é uma das referidas nos critérios da segunda fase**

```
doenca_2afase([]):-!,fail.
doenca_2afase(["Diabetes"|_]).
doenca_2afase(["Neoplasia_maligna"|_]).
doenca_2afase(["Doenca_renal_cronica"|_]).
doenca_2afase(["Insuficiencia_hepatica"|_]).
doenca_2afase(["Hipertensao_arterial"|_]).
doenca_2afase(["Obesidade"|_]).
doenca_2afase([H|T]):- doenca_2afase(T).

```

- **Predicado que averigua os utentes candidatos à primeira fase de acordo com a sua idade e a lista de doenças crónicas**

```
between50and64_diseaded(RS):- solucoes((ID,N),(utente(ID
,_,N,D-M-A,_,_,_,_,_,LDC,_),doenca_2afase(LDC),
between50and64(A)),R),remove_dups(R,RS),!.

```

- **Predicado que averigua se a idade de um utente se encontra entre os 50 e os 64**

```
between50and64(A):- K is 2021, K-A > 50, K-A =< 64.

```

- **Predicado que averigua os utentes candidatos à segunda fase que têm mais de 65 anos e ainda não foram vacinados**

```
morethan65_notvacinated(RS):- solucoes((ID,N),(utente(ID
,_,N,D-M-A,_,_,_,_,_,_,_),nao_vacinado(X),pertence((ID,N),
X),K is 2021, K-A >= 65),R),remove_dups(R,RS).

```

- **Terceira Fase**
  - Todos os restantes que não obedecem às condições impostas anteriormente.

Predicados

- **Predicado que averigua os utentes candidatos à terceira fase**

```
1    terceira_fase(RS) :- solucoes((ID,N),(utente(ID,_,N,_,_,
     _,_,_,_,_),primeira_fase(X),nao(pertence((ID,N),X)),
     segunda_fase(Y),nao(pertence((ID,N),Y))),R),remove_dups(
     R,RS).
2
```

4.4.2 Evolução e Involução

Para que seja possível inserir e remover conhecimento da base de conhecimento, utilizamos os predicados **Evolucao** e **Involucao** que utilizam certos predicados auxiliares para garantirem que a inserção e remoção de conhecimento seja feita corretamente.

- **evolucao** - Predicado que, utilizando os predicados auxiliares *solucoes*, *teste* e *insercao*, insere conhecimento na base de conhecimento corretamente.

```
1    evolucao( Termo ) :-
2        solucoes( Invariante,+Termo::Invariante,Lista ),
3        insercao( Termo ),
4        teste( Lista ).
5
```

- **involucao** - Predicado que, utilizando os predicados auxiliares *solucoes*, *teste* e *remocao*, remove conhecimento da base de conhecimento corretamente.

```
1    involucao(Termo):-
       solucoes(Invariante,-Termo::Invariante,Lista),
2        teste(Lista),
3    remocao(Termo).
4
```

4.4.3 Funcionalidades do Programa

Apresentamos, de seguida, os predicados requisitados.

- **registaUtente** - Extensão do predicado registaUtente que permite o registo de um utente

```
registaUtente(ID,NSS,N,DT,E,T,M,P,LDC,IDCS):- evolucao(utente(ID
    ,NSS,N,DT,E,T,M,P,LDC,IDCS)).

```

```
utente(1, 123123123, "Luis", 2-2-2000, "luis@gmail.com", 911222333, "braga"
, "estudante", ["Diabetes"], 1).
utente(5, 123456789, "Marta", 2-2-1993, "marta@gmail.com", 912345677, "alga
rve", "esteticista", [], 3).
utente(2, 123123124, "Joao", 24-4-2000, "joao@gmail.com", 911222334, "braga
", "estudante", ["Insuficiencia_renal"], 2).
utente(3, 124123124, "Manuel", 2-2-1969, "manuel@gmail.com", 911232334, "br
aga", "arquiteto", ["Insuficiencia_cardiaca", "Insuficiencia_renal", "Insuf
iciencia_hepatica"], 1).
utente(4, 124123125, "Xavier", 2-2-2000, "xavier@gmail.com", 911232355, "al
entejo", "enfermeiro", [], 1).
utente(6, 124123129, "Oliver", 2-2-1958, "oliver@gmail.com", 911232634, "br
aga", "professor", ["Insuficiencia_hepatica"], 2).

true.

?- registaUtente(7,125126127,"Nuno",23-4-1999,"nuno@gmail.com",966125127,"v
iseu","enfermeiro",[],3).
true .

?- showAllUtentes().
:- dynamic utente/10.

utente(1, 123123123, "Luis", 2-2-2000, "luis@gmail.com", 911222333, "braga"
, "estudante", ["Diabetes"], 1).
utente(5, 123456789, "Marta", 2-2-1993, "marta@gmail.com", 912345677, "alga
rve", "esteticista", [], 3).
utente(2, 123123124, "Joao", 24-4-2000, "joao@gmail.com", 911222334, "braga
", "estudante", ["Insuficiencia_renal"], 2).
utente(3, 124123124, "Manuel", 2-2-1969, "manuel@gmail.com", 911232334, "br
aga", "arquiteto", ["Insuficiencia_cardiaca", "Insuficiencia_renal", "Insuf
iciencia_hepatica"], 1).
utente(4, 124123125, "Xavier", 2-2-2000, "xavier@gmail.com", 911232355, "al
entejo", "enfermeiro", [], 1).
utente(6, 124123129, "Oliver", 2-2-1958, "oliver@gmail.com", 911232634, "br
aga", "professor", ["Insuficiencia_hepatica"], 2).
utente(7, 125126127, "Nuno", 23-4-1999, "nuno@gmail.com", 966125127, "viseu
", "enfermeiro", [], 3).

true.
```

Figure 1: registarUtente

- **registaCentroSaude** - Extensão do predicado registaCentroSaude que permite o registo de um centro de saude

```
1 registaCentroSaude(IDCS,N,M,T,E):-
      evolucao(centro_saude(IDCS,N,M,T,E)).
2
```

```
?- showAllCentroSaude().
:- dynamic centro_saude/5.

centro_saude(1, "Hospital de Braga", "Braga", 253027000, "hospital_braga@sn
s.pt").
centro_saude(2, "Hospital da Trofa Braga Norte", "Braga", 253027002, "hospi
tal_trofa_braga@sns.pt").
centro_saude(3, "Hospital de SÃ£o JoÃ£o", "Porto", 225512100, "hospital_sao
_joao@sns.pt").

true.

?- registaCentroSaude(4,"Hospital de Viseu","Viseu",232000551,"hospital_vis
eu@sns.pt").
true .

?- showAllCentroSaude().
:- dynamic centro_saude/5.

centro_saude(1, "Hospital de Braga", "Braga", 253027000, "hospital_braga@sn
s.pt").
centro_saude(2, "Hospital da Trofa Braga Norte", "Braga", 253027002, "hospi
tal_trofa_braga@sns.pt").
centro_saude(3, "Hospital de SÃ£o JoÃ£o", "Porto", 225512100, "hospital_sao
_joao@sns.pt").
centro_saude(4, "Hospital de Viseu", "Viseu", 232000551, "hospital_viseu@sn
s.pt").

true.
```

Figure 2: registaCentroSaude

- **registaStaff** - Extensão do predicado registaStaff que permite o registo de um staff

```
registaStaff(IDS,IDC,N,E):- evolucao(staff(IDS,IDC,N,E)).

```

```
?- showAllStaff().
:- dynamic staff/4.

staff(1, 1, "Jorge", "jorge_andrade@gmail.com").
staff(2, 2, "Liliana", "liliana_albernaz@gmail.com").
staff(3, 3, "Andre", "martinsdr@gmail.com").
staff(4, 1, "Antonio", "antonioalves@gmail.com").
staff(5, 2, "Ruben", "rubenpteixeira@gmail.com").
staff(6, 3, "Julian", "julianaribeiro@gmail.com").

true.

?- registaStaff(7,1,"Romeu","romeuteatro@gmail.com").
true .

?- showAllStaff().
:- dynamic staff/4.

staff(1, 1, "Jorge", "jorge_andrade@gmail.com").
staff(2, 2, "Liliana", "liliana_albernaz@gmail.com").
staff(3, 3, "Andre", "martinsdr@gmail.com").
staff(4, 1, "Antonio", "antonioalves@gmail.com").
staff(5, 2, "Ruben", "rubenpteixeira@gmail.com").
staff(6, 3, "Julian", "julianaribeiro@gmail.com").
staff(7, 1, "Romeu", "romeuteatro@gmail.com").
```

Figure 3: registaStaff

- **registaVacinacaoCovid** - Extensão do predicado registaVacinacaoCovid que permite o registo de uma vacinação feita

```
registaVacinacaoCovid(IDS,IDU,D,V,T):-
        evolucao(vacinacao_covid(IDS,IDU,D,V,T)).

```

```
?- showAllVacinacao().
:- dynamic vacinacao_covid/5.

vacinacao_covid(1, 1, 23-3-2021, "pfizer", 1).
vacinacao_covid(1, 1, 31-3-2021, "pfizer", 2).
vacinacao_covid(3, 5, 23-3-2021, "pfizer", 1).
vacinacao_covid(3, 2, 12-12-2020, "pfizer", 1).
vacinacao_covid(2, 3, 7-2-2021, "astrazeneca", 1).
vacinacao_covid(3, 2, 12-2-2021, "pfizer", 2).

true.

?- registaVacinacaoCovid(3,5,7-4-2021,"pfizer",2).
true .

?- showAllVacinacao().
:- dynamic vacinacao_covid/5.

vacinacao_covid(1, 1, 23-3-2021, "pfizer", 1).
vacinacao_covid(1, 1, 31-3-2021, "pfizer", 2).
vacinacao_covid(3, 5, 23-3-2021, "pfizer", 1).
vacinacao_covid(3, 2, 12-12-2020, "pfizer", 1).
vacinacao_covid(2, 3, 7-2-2021, "astrazeneca", 1).
vacinacao_covid(3, 2, 12-2-2021, "pfizer", 2).
vacinacao_covid(3, 5, 7-4-2021, "pfizer", 2).

true.
```

Figure 4: registaVacinacaoCovid

- **falta_segunda_vacina** - Lista com IDs e nomes de utentes a quem falta a segunda vacina.

```
falta_segunda_vacina(R):-solucoes( (IDU,N),
      (utente(IDU,_,N,_,_,_,_,_,_,_),
      vacinacao_covid(_,IDU,_,_,1),
      nao(vacinacao_covid(_,IDU,_,_,2))), R ).

```

  Neste predicado procuramos pelos IDs dos utentes que possuem a primeira dose tomada, mas não possuem uma segunda, juntando os resultados obtidos numa lista, juntamente com o nome de cada utente.

- **nao_vacinado** - Lista com IDs e nomes de utentes não vacinados.

```
nao_vacinado(R):- solucoes((ID,N),
        (utente(ID,_,N,_,_,_,_,_,_,_),
        nao(vacinacao_covid(_,ID,_,_,_))),R).

```

  Como não vacinado decidimos considerar os utentes que ainda não possuem as duas vacinas tomadas na totalidade, portanto procuramos pelos IDs dos utentes para os quais não existe qualquer conhecimento sobre vacinacao_covid, guardando numa lista o seu ID e nome.

- **vacinado** - Lista com IDs e nomes de utentes vacinados completamente.

```
vacinado(R):- solucoes((ID,N),(utente(ID,_,N,_,_,_,_,_,_,_),
              vacinacao_covid(_,ID,_,_,2)),R).

```

  Procurando por todas as existências sobre as vacinações de um utente que façam referência a uma segunda toma, guardamos numa lista o seu ID e nome, estando salvaguardados pelos invariantes de inserção de que não existe uma segunda dose sem a primeira.

- **vacinado_indevidamente** - Extensão do predicado vacinado_indevidamente que apresenta uma lista com os utentes que foram mal vacinados de acordo com os critérios estabelecidos.

```
vacinado_indevidamente(RS) :- solucoes((ID,N),
                (utente(ID,_,N,_,_,_,_,_,_,_),
                vacinado_indevidamente_1afase(X),
                vacinado_indevidamente_2afase(Z),concatenar(X,Z,
   R),pertence((ID,N),R)),R),
                remove_dups(R,RS).

```

Tendo em conta os critérios estabelecidos previamente, este predicado irá procurar por todos os utentes que tenham sido vacinados indevidamente ou na primeira ou na segunda fase, sendo posteriormente apresentado numa lista o seu ID e nome. Para tal verificação, fazemos uso dos predicados auxiliares *vacinado_indevidamente_1afase* e *vacinado_indevidamente_2afase*.

- **vacinado_indevidamente_1afase** - Extensão do predicado vacinado_indevidamente_1afase que apresenta uma lista com os utentes que foram mal vacinados de acordo com os critérios estabelecidos para a primeira fase.

```
vacinado_indevidamente_1afase(RS):- solucoes((ID,N),
        (utente(ID,_,N,_,_,_,_,_,_,_,_),
        atleast_one_vacina(ID),
        (vacinacao_covid(_,ID,D-M-A,_,_),(M\=12,A==2020;M>3,A
   ==2021);
        nao(check_1afase((ID,N)))),R),
        remove_dups(R,RS).

```

Para averiguarmos os utentes que foram vacinados indevidamente na primeira fase, vamos averiguar se o mesmo possui registo de pelo menos uma vacinação na base de conhecimentos, se alguma dessas vacinações foi realizada ou está agendada fora do período relativo à primeira fase e se esse utente não é candidato à primeira fase, sendo depois guardados numa lista o seu ID e nome.

- **vacinado_indevidamente_2afase** - Extensão do predicado vacinado_indevidamente_2afase que apresenta uma lista com os utentes que foram mal vacinados de acordo com os critérios estabelecidos para a segunda fase.

```
vacinado_indevidamente_2afase(RS):- solucoes((ID,N), (utente(ID,
   _,N,_,_,_,_,_,_,_,_),
        atleast_one_vacina(ID),
        (vacinacao_covid(_,ID,D-M-A,_,_),((M<4;M>8),A\=2021);
        nao(check_2afase((ID,N)))),R),
        remove_dups(R,RS).

```

Para averiguarmos os utentes que foram vacinados indevidamente na segunda fase, vamos averiguar se o mesmo possui registo de pelo menos uma vacinação na base de conhecimentos, se alguma dessas vacinações foi realizada ou está agendada fora do período relativo à segunda fase e se esse utente não é candidato à segunda fase, sendo depois guardados numa lista o seu ID e nome.

- **atleast_one_vacina** - Extensão do predicado atleast_one_vacina que indica se um utente já tem pelo menos uma vacina.

```
atleast_one_vacina(ID):- solucoes((ID,N),
    (utente(ID,_,N,_,_,_,_,_,_,_,_),
    vacinacao_covid(_,ID,_,_,_)),R),
    comprimento(R,N), N>=1.

```

O presente predicado serve como auxiliar para a verificação da presença de pelo menos uma vacinação para um utente na base de conhecimentos.

- **check_1afase** - Extensão do predicado check_1afase que indica se um utente pertence aos utentes aptos à primeira fase de vacinação.

```
check_1afase((ID,N)):- primeira_fase(Y),pertence((ID,N),Y).

```

O presente predicado verifica se para um dado tuplo, que contém o ID e nome do utente, este se encontra na lista dos utentes candidatos à primeira fase.

- **check_2afase** - Extensão do predicado check_2afase que indica se um utente pertence aos utentes aptos à segunda fase de vacinação.

```
check_2afase((ID,N)):- segunda_fase(Y),pertence((ID,N),Y).

```

O presente predicado verifica se para um dado tuplo, que contém o ID e nome do utente, este se encontra na lista dos utentes candidatos à segunda fase.

- **candidato_nao_vacinado** - Extensão do predicado candidato_nao_vacinado que apresenta uma lista com os utentes que ainda não foram vacinados e são candidatos.

```
candidato_nao_vacinado(RS):- solucoes((ID,N),
        (utente(ID,_,N,_,_,_,_,_,_,_,_),
        nao_vacinado(X),pertence((ID,N),X),
        candidato(Z),pertence((ID,N),Z)),R),
        remove_dups(R,RS).

```

Como dito previamente, consideramos utentes não vacinados, aqueles que ainda não foram totalmente vacinados, ou seja, não tomaram as duas doses. Assim, para averiguarmos os utentes que são candidatos mas ainda não foram vacinados, vamos procurar por aqueles que cumpram com esse requisitos fazendo uso dos predicados nao_vacinado e candidato, averiguando se um tuplo com o seu ID e nome, cumprem com tais requisitos, sendo depois guardados numa lista.

- **candidato** - Extensão do predicado candidato que indica se um utente é candidato a ser vacinado.

```
candidato(RS):- primeira_fase(X),
        segunda_fase(Y),
        concatenar(X,Y,R),
        remove_dups(R,RS).

```

O presente predicado serve como auxiliar na inferência dos utentes que são candidatos a serem vacinados, aqueles que cumprem com os critérios da primeira ou da segunda fase.

4.4.4 Sistema de Inferência

De modo a cumprir com o requisitado no enunciado do trabalho, deveríamos ser capazes de desenvolver um sistema de inferência capaz de implementar os mecanismos de raciocínio inerentes a estes sistemas.

- Extensão do meta-predicado demo: $Questao, Resposta -> \{V, F\}$
  $Resposta = \{verdadeiro, falso\}$

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

```

Com isto somos capazes de averiguar se, para uma dada questão, existe conhecimento na nossa base de conhecimento sobre a questão. Caso exista, iremos obter como resposta ***verdadeiro***; caso exista a negação dessa questão, iremos obter a resposta ***falso***; por fim, e não sendo necessário mas achando que faria sentido existir, se não houver informação sobre a questão, nem a sua negação, iremos obter a resposta ***desconhecido***.

4.5 Funcionalidades Extra

4.5.1 Predicados

Para além dos predicados requisitados no enunciado na secção anterior e respetivos predicados auxiliares a esses, decidimos adicionar novos predicados que complementam a nossa base de conhecimento com novas funcionalidades.

4.5.2 Listar base de conhecimento - Predicados show

- **showAllUtentes** - Extensão do predicado showAllUtentes que apresenta ao utilizador todos os utentes.

```
showAllUtentes():- listing(utente).

```

- **showAllCentroSaude** - Extensão do predicado showAllCentroSaude que apresenta ao utilizador todos os centros de saude.

```
showAllCentroSaude():- listing(centro_saude).

```

- **showStaffAtCentroSaude** - Extensão do predicado showStaffAtCentroSaude que apresenta uma lista de staff que está designado para um dado centro de saúde.

```
showStaffAtCentroSaude(IDCS,RS):- solucoes((IDS,N),(staff(IDS,
    IDCS,N,_)),R),remove_dups(R,RS).

```

- **showAllStaff** - Extensão do predicado showAllStaff que apresenta ao utilizador todo o staff.

```
showAllStaff():- listing(staff).

```

- **showVacinacaoDoneByStaff** - Extensão do predicado showVacinacaoDoneByStaff que apresenta uma lista de vacinações feitas por um determinado staff.

```
showVacinacaoDoneByStaff(IDS,RS):-solucoes(
                (vacinacao_covid(IDS,IDU,DT,V,T)),
                (staff(IDS,_,_,_),
                vacinacao_covid(IDS,IDU,DT,V,T)),R),
                remove_dups(R,RS).

```

- **showAllVacinacao** - Extensão do predicado showAllVacinacao que apresenta ao utilizador todas as vacinações.

```
showAllVacinacao():- listing(vacinacao_covid).

```

- **showVacinacaoAtCentroSaude** - Extensão do predicado showVacinacaoAtCentroSaude que apresenta uma lista de vacinações realizadas num dado centro de saúde.

```
showVacinacaoAtCentroSaude(IDCS,RS):-solucoes(
                (vacinacao_covid(IDS,IDU,D,V,T)),
                (utente(IDU,_,_,_,_,_,_,_,_,_,IDCS),
                vacinacao_covid(IDS,IDU,D,V,T),
                staff(IDS,_,_,_),
                centro_saude(IDCS,NCS,_,_,_)),R),
                remove_dups(R,RS).

```

4.5.3 Remoção de conhecimento à base de conhecimento - Remove

- **removeUtenteID** - Extensão do predicado removeUtenteID que permite a remoção de um utente pelo seu ID

```
removeUtenteID(ID):- involucao(utente(ID,_,_,_,_,_,_,_,_,_,_)).

```

- **removeUtenteNSS** - Extensão do predicado removeUtenteNSS que permite a remoção de um utente pelo seu numero de segurança social

```
removeUtenteNSS(NSS):- involucao(utente(_,NSS,_,_,_,_,_,_,_,_,_)).

```

- **removeUtenteAllInfoByID** - Extensão do predicado removeUtenteAllInfoByID que permite a remoção de toda a informação sobre um utente pelo seu ID

```
removeUtenteAllInfoByID(ID):- solucoes(ID,(vacinacao_covid(_,ID,
   _,_,_)),R),
                    removeInfoVacinacao(R),
                    removeUtenteID(ID).

```

- **removeInfoVacinacao** - Extensão do predicado removeInfoVacinacao que permite a remoção das vacinações de um utente pelo seu ID

```
removeInfoVacinacao([]).
removeInfoVacinacao([IDU|Tail]):-
                    removeVacinacaoCovid(IDU),
                    removeInfoVacinacao(Tail).

```

- **removeCentroSaudeID** - Extensão do predicado removeCentroSaudeID que permite a remoção de um centro de saude pelo seu ID

```
removeCentroSaudeID(IDCS):- involucao(centro_saude(IDCS,_,_,_,_)
    ).

```

- **removeCentroSaudeAndStaff** - Extensão do predicado removeCentroSaude-AndStaff que permite a remoção de um centro de saude pelo seu ID e o staff associado

```
removeCentroSaudeAndStaff(IDCS):-
                solucoes(IDS,(staff(IDS,IDCS,_,_)),R),
                removeStaffAtCentroSaude(R),
                involucao(centro_saude(IDCS,_,_,_,_)).


removeStaffAtCentroSaude([]).
removeStaffAtCentroSaude([IDS|T]):-
                involucao(staff(IDS,_,_,_)),
                removeStaffAtCentroSaude(T).

```

- **removeStaffID** - Extensão do predicado removeStaffID que permite a remoção de um staff pelo seu ID

```
removeStaffID(ID):- involucao(staff(ID,_,_,_)).

```

- **removeVacinacaoCovid** - Extensão do predicado removeVacinacaoCovid que permite a remoção de uma vacinação a partir do ID de um utente

```
removeVacinacaoCovid(IDU):- involucao(vacinacao_covid(_,IDU,_,_,
    _)).

```

5 Conclusão

A resolução deste projeto permitiu a consolidação dos conhecimentos adquiridos até aqui nas aulas relativamente à linguagem de programação em lógica o PROLOG. O grupo considera que conseguiu representar o conhecimento pedido. Para além disto tal como sugerido, optamos por adicionar novas funcionalidades porém ainda consideramos ser possível expandir ainda mais a base de conhecimento elaborada.

Neste aspeto, poderíamos ter elaborado mais a base de conhecimento com informação sobre utentes, staff, centros de saúde e vacinações realizadas ou agendadas, poderíamos ter adicionados mais funcionalidades extra que nos permitissem inferir mais pormenores como os utentes que existem num dado centro de saúde, ou ainda melhorar invariantes como o da vacinação relativamente a verificar se o staff que está a vacinar um dado utente, trabalha no centro de saúde onde o mesmo utente está registado.

Em suma, os objetivos estabelecidos foram cumpridos e o grupo conseguiu cumprir todos os requisitos e ainda aprimorar o projeto relativamente ao que era pedido, podendo então considerar que este trabalho foi realizado com sucesso.

References

[1] Critérios Vacinação: https://covid19.min-saude.pt/vacinacao/

[2] SWI Prolog Manual : https://www.swi-prolog.org/pldoc/man?section=builtin