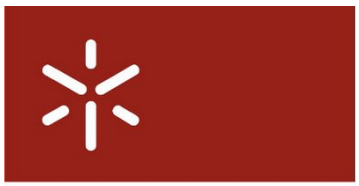

Universidade do Minho

Sistemas de Representação de Conhecimento e Raciocínio

MIEI - 3º ano - 2º semestre
Universidade do Minho

Trabalho Prático


Sara Pereira
A73700


Rui Vieira
A74658


Filipe Fortunato
A75008


Frederico Pinto
A73639

y

1. *Resumo*

Este documento representa o resultado obtido após a realização do primeiro trabalho prático da unidade UC Sistema de Representação de Conhecimento e Raciocínio com o tema *Programação lógica e Invariantes*. Com a realização deste exercício pretende-se motivar os alunos para a utilização da linguagem de programação em lógica PROLOG, no âmbito da representação de conhecimento e construção de mecanismos de raciocínio para a resolução de problemas.

Conteúdo

2. *Introdução*

Este relatório surge para concluir a elaboração do primeiro trabalho prático, proposto na unidade curricular de Sistemas de Representação de Conhecimento e Raciocínio.

Este primeiro trabalho prático consiste no 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. Para concretizar esse objetivo iremos utilizar o *PROLOG*.

PROLOG é uma linguagem de programação em lógica que está associada à inteligência artificial e linguística computacional.

3. *Preliminares*

Este sistema tem o intuito de representar a prestação de serviços médicos, contendo três fontes de conhecimento. Como principal interveniente temos o utente, que está associado um ID, Nome, Idade e Morada, utente esse ao qual é prestado um cuidado que tem a si associado uma Data, um Id de um utente, um Id de um prestador, uma Prioridade, uma Descrição e um Custo. O prestador do cuidado guarda informação como o seu Id, Nome, Especialidade e Id da instituição. É de notar que, por razões óbvias, o ID é único tendo em conta no entanto que os cuidados estão associados a utentes e prestadores podendo o mesmo cliente pode ser recebido várias vezes pelo mesmo prestador. Posto isto encontramos-nos aptos a prosseguir com a elaboração do trabalho em si.

4. *Descrição do Trabalho e Análise de Resultados*

4.1 Predicados de Evolução/Retrocesso de Conhecimento

Para o correto funcionamento da nossa base de conhecimento inserimos alguns invariantes de maneira a não existir qualquer problema com a informação contida na base de conhecimento. Desta forma é necessário utilizar invariantes para garantir que, no momento da inserção e remoção de conhecimento, haja controlo dessa informação. Para tal criamos os predicados *evolucao* e *involucao* para controlarmos a inserção e remoção, respetivamente.

```
remove(T):- retract(T).


insere(T):- assert(T).
insere(T):- retract(T),!,fail.



evolucao(T):- solucoes(Inv,+T::Inv,Lista),
              inserir(T),
              teste(Lista).


involucao(T):- solucoes(I,-T::I,Lista),
               teste(Lista),
               remove(T).
```

Como podemos ver, estes predicados impedem qualquer ação, no momento de inserção e remoção de informação, caso um invariante não se verifique, garantindo assim a consistência da nossa base de conhecimento.

4.2 Invariantes

Como já referido anteriormente, é necessário implementar invariantes de maneira a podermos controlar a informação na nossa base de conhecimento, garantindo assim o correto funcionamento da mesma. Criamos dois tipos de invariantes, uns associados à inserção e outros associados à remoção. Para diferenciar os dois, os invariantes de inserção têm como prefixo um '+' enquanto que os de remoção têm um '-' como prefixo. Assim é possível saber. Para os invariantes associados à inserção, criamos um conjunto que impede a repetição de conhecimento (utente, prestador,cuidado,instituição), isto é, após a inserção, é verificado se o ID do predicado dado é único (para o cuidado é verificada toda a informação sobre o predicado), caso contrário remove o mesmo da base de conhecimento. Para além destes, é também assegurado que não é possível inserir um cuidado médico cujos **id de utente** e **id de prestador** não existam na base de conhecimento. Quanto à remoção é necessário haver um controlo, garantindo que apenas um elemento é removido nesse momento.

```
+utente(Idu,N,Idd,M)::((solucoes(Idu, utente(IdU,N,Idd,M), U),
                        comprimento(U,N), N==1)).



+prestador(IdP,N,E,Itt)::((solucoes(Idp, prestador(Idp,N,E,Itt), P),
                           comprimento(P,N), N==1)).



+cuidado(D,IdU,IdP,Desc,C) :: (solucoes((D,IdU,IdP,Desc,C),
                                cuidado(D,IdU,IdP,Desc,C), L),
                                comprimento(L,N), N ==1).



+inst(Id,Nome,Cid) :: (solucoes(Id, inst(Id,Nome,Cid),S), comprimento(S,N), N ==1).



+cuidado(Data,IdU,IdPrest,Prio,Desc,Custo):: (utente(IdU,_,_,_),
                                               prestador(IdPrest,_,_,_)).


-utente(Idu,N,Idd,M):: (solucoes(Idu, utente(Idu,N,Idd,M), L),
                        comprimento(L,R), R==1).

-prestador(Idp,N,Esp,Inst) :: (solucoes(Idp, prestador(Idp,N,Esp,Inst), L),
                                comprimento(L,R), R==1).
```

```
-cuidado(Data,IdU,IdPrest,Prio,Desc,Custo) ::
(solucoes((Data,IdU,IdPrest,Prio,Desc,Custo),
cuidado(Data,IdU,IdPrest,Prio,Desc,Custo), L), comprimento(L,R), R==1).

-inst(Id,N,C) :: (solucoes(Id, inst(Id,N,C), L), comprimento(L,R), R ==1).
```

4.3 Predicados gerais/auxiliares

Nesta secção é feita a referência e explicação de todos os predicados auxiliares aos requisitados neste projeto, assim como os auxiliares aos predicados extra acrescentados pelo grupo.

- **comprimento**
  Este predicado, a partir de uma lista, devolve o comprimento da mesma

```
comprimento([],0).
comprimento([X|Y],R):- comprimento(Y,Z),
                                R is Z+1.
```

- **concat**
  Predicado que concatena duas listas

```
concat([], Y, Y).
concat([X|Y], Z, [X|L]) :- concat(Y,Z,L).
```

- **apaga1**
  Predicado auxiliar ao predicado apagaRep. Este recebe um elemento e uma lista e devolve a lista sem todas as ocorrências desse elemento

```
apaga1(_,[],[]).
apaga1(X,[X|Y],T):- apaga1(X,Y,T).
apaga1(X,[H|Y],[H|R]) :- apaga1(X,Y,R).
```

- **apagaRep**
  Predicado que apaga os elementos repetidos de uma lista

```
apagaRep([],[]).
apagaRep([X|Y],[X|L1]) :- apaga1(X,Y,L), apagaRep(L,L1).
```

- **pertence1**
  Predicado auxiliar ao predicado pertence. Este verifica o valor de verdade da existência de um elemento numa lista

```
pertence1(X,[X|Y]).
pertence1(X,[Y|Z]) :- pertence1(X,Z).
```

- **pertence**
  Predicado que, dadas duas listas retorna lista dos elementos da primeira lista que não existem na segunda lista

```
pertence([],_,[]).
pertence([X|Y],Z,[X|R]) :- not(pertence1(X,Z)), pertence(Y,Z,R).
pertence([X|Y],Z,R) :- pertence1(X,Z), pertence(Y,Z,R).
```

- **maxLista**
  Predicado que calcula o máximo de uma lista

```
maxLista([H],R):- R is H.
maxLista([X|L],R) :- maxLista(L,N), X>N, R is X.
maxLista([X|L],R) :- maxLista(L,N), X=<N, R is N.
```

- **iSort**
  Predicado que insere um cuidado num lista ordenadamente. Este é um predicado auxiliar do predicado ordena.

```
iSort((A,B,C,D,E,X),[],[(A,B,C,D,E,X)]).
iSort((A,B,C,D,E,X),[(F,G,H,I,J,Y)|Z],[(A,B,C,D,E,X),(F,G,H,I,J,Y)|Z]) :- X\=<
iSort((A,B,C,D,E,X),[(F,G,H,I,J,Y)|Z],[(F,G,H,I,J,Y)|R]) :- X>Y,
                                iSort((A,B,C,D,E,X),Z,R).
```

- ordena
  Predicado que ordena uma lista de cuidados

```
ordena([],[]).
ordena([(A,B,C,D,E,X)|Y],R) :- ordena(Y,R1), iSort((A,B,C,D,E,X),R1,R)
```

4.4 Conhecimento extra - Instituição e Prioridade

De maneira a acrescentar conhecimento ao mínimo requisitado pelo enunciado do projeto, o grupo decidiu por optar acrescentar informação sobre a prioridade 'Baixa','Media','Alta' de um cuidado de saúde. Assim como, também foi decidido acrescentar dados sobre uma instituição tais como um identificador único de cada uma, o nome e a cidade. Em conclusão, o panorama poderá ser caraterizado por conhecimento dado na forma:

- *cuidado: Data, #IdU, #IdPrest, Prioridade, Descrição, Custo -> V,F*
- instituição: #IdInst, Nome, Cidade -> V, F
- *utente: #IdUt, Nome, Idade, Morada ->*V, F
- *prestador: #IdPrest, Nome, Especialidade, Instituição-> V,F*

```
:- dynamic utente/4.
:- dynamic prestador/4.
:- dynamic cuidado/6.
:- dynamic inst/3.
```

4.5 Base de Conhecimento Inicial

Para a realização deste trabalho, é necessário que o programa já possua uma base de conhecimento inicial, de maneira a que possamos testar a nossa informação, sem termos de inserir conhecimento. Criamos dois utentes com a mesma informação de maneira a podermos abranger alguns casos específicos.

```
utente(1,'Pascoal',38,'Rua Limpa').
utente(1,'Pascoal',38,'Rua Limpa').
utente(2,'Zeca',20,'Rua da Capa').
utente(3,'Anibal',59,'Rua do Gota').
utente(4,'Maria',42,'Rua dos Peoes').
utente(5,'Carlota',22,'Rua do Speedy').
utente(6,'Brito',65,'Rua do Colombo').
utente(7,'Micaela',8,'Rua dos Olivais').
utente(8,'Julio',36,'Rua do Campo').
utente(9,'Dinis',48,'Rua da Cruz').
utente(10,'Rita',88,'Rua das Flores').
utente(11,'Mariana',6,'Rua do Carmo').
utente(12,'Sergio',26,'Rua dos Limoes').
utente(13,'Lucifer',14,'Rua do Palacio').
utente(14,'Miguel',49,'Rua do Pinheiro').
utente(15,'Joana',70,'Rua da Maria').
```

```
prestador(1,'Ze','Otorrino',1).
prestador(2,'Andreia','Dentaria',1).
prestador(3,'Guilherme','Dermatologia',1).
prestador(4,'Manuel','Oncologia',2).
prestador(5,'Elso','Ortopedia',3).
prestador(6,'Bino','Ginecologia', 3).
prestador(7,'Telmo','Radiologia',2).
prestador(8,'Miquelina','Cardiologia',4).
prestador(9,'Armando','Neurologia',4).
prestador(10,'Firmino','Endocrinologia',5).
prestador(11,'Horacio','Otorrino',5).
prestador(12,'Carlos','Dentaria',1).
prestador(13,'Maria','Dermatologia',3).
prestador(14,'Narcisa','Ginecologia',2).
prestador(15,'Adelaide','Psiquiatria',4).
prestador(16,'Teresa','Nutricao',1).
prestador(17,'Ambrosio','Podologia',5).
prestador(18,'Nuno','Radiologia',2).
prestador(19,'Marta','Cardiologia',4).
prestador(20,'David','Oncologia',5).


cuidado('2018-1-1',1,1,'Media','Amigdalite',10).
cuidado('2018-1-1',2,1,'Alta','Carie',26).
cuidado('2018-1-1',3,3,'Baixa','Acne',15).
cuidado('2018-1-2',4,4,'Alta','Cancro',32).
cuidado('2018-1-2',5,5,'Media','Fratura do pulso',19).
cuidado('2018-1-3',4,6,'Baixa','Papa Nicolau', 100).
cuidado('2018-1-3',7,20,'Alta','Cancro da Mama',20).
cuidado('2018-1-3',8,19,'Alta','Enfarte',198).
cuidado('2018-1-4',9,18,'Baixa','Tirar Raio-X',3).
cuidado('2018-1-4',10,17,'Media','Unha encravada',37).
cuidado('2018-1-5',11,16,'Baixa','Plano Alimentar',12).
cuidado('2018-2-5',12,15,'Media','Consulta rotina',90).
cuidado('2018-2-6',13,14,'Media','Ecografia aos ovarios',58).
cuidado('2018-2-6',14,13,'Alta','Urticaria',5).
cuidado('2018-2-7',15,12,'Baixa','Por aparelho',2000).

inst(1,'Hospital Privado de Braga', 'Braga').
inst(2,'IPO','Porto').
inst(3,'Hospital S.Joao','Porto').
inst(4,'Hospital de Felgueiras','Felgueiras').
inst(5,'Hospital dos Bonecos','Lisboa').
```

5. *Funcionalidades do Programa*

Neste tópico será apresentado e explicado cada uma das funcionalidades e quais os resultados obtidos para as soluções a apresentar

5.1 Registar utentes, prestadores e cuidados de saúde

Para que seja possível a inserção de nova informação na base de conhecimento usamos o teorema *evolucao* o qual insere na base de conhecimento de forma controlada.

```
registaUtente(Id,N,Idd,M) :- evolucao(utente(Id,N,Idd,M)).

registaPrest(Idp,N,E,I) :- evolucao(prestador(Idp,N,E,I)).

registaCuidado(D,Idu,Idp,P,Desc,C) :- evolucao(cuidado(D,Idu,Idp,P,Desc,C)).

registaInst(Id,N,C) :- evolucao(inst(Id,N,C)).
```

5.2 Remover utentes, prestadores e cuidados de saúde

Para que seja possível a remoção de informação da base de conhecimento usamos o teorema *involucao* o qual remove da base de conhecimento de forma controlada.

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

removePrestador(ID) :- involucao(prestador(ID,_,_,_)).

removeCuidado(Dt,IdU,IdP,P,Desc,C) :- involucao(cuidado(Dt,IdU,IdP,P,Desc,C)).

removeInst(IdInst) :- involucao(inst(IdInst,_,_)).
```

5.3 Identificar utentes por critérios de seleção

Para podermos identificar os utentes, são usados 4 critérios de seleção : ID, Nome, Idade e Morada. Para representarmos o conjunto de Utentes com os critérios pretendidos, usamos o predicado *solucoes*, que recolhe todos os utentes com os parâmetros (ID, Nome, Idade, Morada) que têm em comum o parâmetro dado inicialmente.

```
utenteID(ID,R) :- solucoes((ID,N,I,M), utente(ID,N,I,M), R).

utenteNome(Nome,R) :- solucoes((ID,Nome,I,M), utente(ID,Nome,I,M), R).

utenteIdade(Idade,R) :- solucoes((ID,N,Idade,M),utente(ID,N,Idade,M),R).

utenteMor(M,R) :- solucoes((ID,N,I,M),utente(ID,N,I,M),R).
```

Exemplo de output:

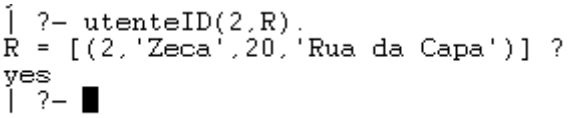

Figura 5.1: Exemplo de output do predicado utenteID

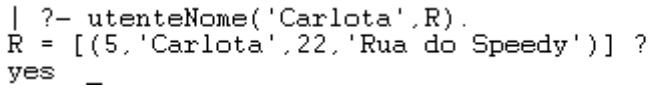

Figura 5.2: Exemplo de output do predicado utenteNome

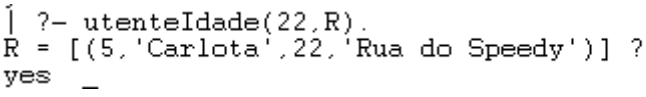

Figura 5.3: Exemplo de output do predicado utenteIdade

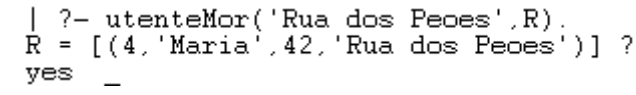

Figura 5.4: Exemplo de output do predicado utenteMor

5.4 Identificar as instituições prestadoras de cuidados de saúde

Da mesma maneira usada anteriormente, também, aqui é usado o predicado **solucoes** para recolher todas as instituições prestadoras de cuidados.Assim, no predicado é necessário cruzar o conhecimento das instituições com o dos cuidados de saúde e, de seguida apagar os elementos repetidos (apenas para questão de apresentação e perceção da solução) de maneira a obter o resultado pretendido. Assim, **R** representa a lista de **inst(Id,N,C)** devolvida pelo **solucoes** através do cruzamento de informação referido acima, em que os parâmetros em comum são **Idp - id de um prestador**.

```
inst_cuidados(R1) :- solucoes(inst(Id,N,C), (inst(Id,N,C),
prestador(Idp,_,_,Id),
```

```
    cuidado(_,Idp,_,_,_,_)), R),
    apagaRep(R,R1).
```

Exemplo de Output:

```
| ?- inst_cuidados(R).
R = [inst(1,'Hospital Privado de Braga','Braga'),inst(2,'IPO','Porto'),inst(3,'
Hospital S.Joao','Porto'),inst(4,'Hospital de Felgueiras','Felgueiras'),inst(5,
'Hospital dos Bonecos','Lisboa')] ?
yes
| ?-
```

Figura 5.5: Exemplo de output do predicado inst$_c$*uidados*

5.5 Identificar cuidados de saúde prestados por instituição/cidade/datas

Instituição

O raciocínio aplicado no cálculo deste predicado foi semelhante ao anterior, havendo também a união da informação devolvida pelo predicado **solucoes**, com os parâmetros **inst(Id,N,_)**, **prestador(Idp,_,_,Id)**, **cuidado(D,Idu,Idp,P,Desc,Custo)**.

```
cuidados_I(N,R) :- solucoes(cuidado(D,Idu,Idp,P,Desc,Custo),
                   (inst(Id,N,_),prestador(Idp,_,_,Id),
                   cuidado(D,Idu,Idp,P,Desc,Custo)),R).
```

Cidade

O raciocínio aplicado no cálculo deste predicado foi semelhante ao anterior, havendo também a união da informação devolvida pelo predicado **solucoes**, com os parâmetros **inst(Id,_,C)**, **prestador(Idp,_,_,Id)**, **cuidado(D,Idu,Idp,P,Desc,Custo)**.

```
cuidados_C(C,R) :- solucoes(cuidado(D,Idu,Idp,P,Desc,Custo),
                   (inst(ID,_,C), prestador(Idp,_,_,ID),
                   cuidado(D,Idu,Idp,P,Desc,Custo)),R).
```

Data

O raciocínio aplicado no cálculo deste predicado foi semelhante ao anterior, havendo também a união da informação devolvida pelo predicado **solucoes**, com os parâmetros **cuidado(D,Idu,Idp,P,Desc,Custo)**.

```
cuidados_D(D,R1) :- solucoes((D,Idu,Idp,P,Desc,C),
                    cuidado(D,Idu,Idp,P,Desc,C),R1).
```

Exemplo de Output:

```
| ?- cuidados_I('IPO',R).
R = [cuidado('2018-1-2',4,4,'Alta','Cancro',32),cuidado('2018-2-6',13,14,'Media
','Ecografia aos ovarios',58),cuidado('2018-1-4',9,18,'Baixa','Tirar Raio-X',3)
] ?
yes
```

Figura 5.6: Exemplo de output do predicado cuidado_I

```
| ?- cuidados_C('Braga',R).
R = [cuidado('2018-1-1',1,1,'Media','Amigdalite',10),cuidado('2018-1-1',2,1,'Al
ta','Carie',26),cuidado('2018-1-1',3,3,'Baixa','Acne',15),cuidado('2018-2-7',15
,12,'Baixa','Por aparelho',2000),cuidado('2018-1-5',11,16,'Baixa','Plano Alimen
tar',12)] ?
yes
```

Figura 5.7: Exemplo de output do predicado cuidado_C

```
| ?- cuidados_D('2018-1-1',R).
R = [('2018-1-1',1,1,'Media','Amigdalite',10),('2018-1-1',2,1,'Alta','Carie',26
),('2018-1-1',3,3,'Baixa','Acne',15)] ?
yes
```

Figura 5.8: Exemplo de output do predicado cuidados_D

5.6 Identificar os utentes de um prestador, especialidade,instituição

Prestador

Para este tópico, a abordagem continua a mesma, através do **solucoes** é devolvida a lista com os utentes de um dado prestador a partir do id do mesmo. Assim, é feito o cruzamento da informação através dos predicados **cuidado(_,Idu,Idp,_,_,_)**, **prestador(Idp,_,_,_)**, **utente(Idu,N,Idd,M)**. Desta maneira, os parâmetros em comum são **Idp - id do prestador**

```
utentes_de_prest(Idp,R) :- solucoes(utente(Idu,N,Idd,M),
                           (cuidado(_,Idu,Idp,_,_,_), prestador(Idp,_,_,_),
                            utente(Idu,N,Idd,M)),R1),apagaRep(R1,R).
```

Especialidade

Para este tópico, a abordagem continua a mesma, através do **solucoes** é devolvida a lista com os utentes de uma dada especialidade. Assim, é feito o cruzamento da informação através dos predicados **cuidado(_,Idu,Idp,_,_,_)**, **prestador(Idp,_,Esp,_)**, **utente(Idu,N,Idd,M)**. Assim, os parâmetros em comum são **Idp - id prestador** e **Esp - especialidade**.

```
utentes_de_esp(Esp,R) :- solucoes(utente(Id,N,Idd,M),
                         (cuidado(_,Id,Idp,_,_,_), prestador(Idp,_,Esp,_),
                         utente(Id,N,Idd,M)), R1), apagaRep(R1,R).
```

Instituição

Para este tópico, a abordagem continua a mesma, através do **solucoes** é devolvida a lista com os utentes de um dado prestador a partir do id do mesmo. Assim, é feito o cruzamento da informação através dos predicados **cuidado(_,Idu,Idp,_,_,_)**, **prestador(Idp,_,_,Idinst)**, **utente(Idu,N,Idd,M)**, **inst(Idinst,,NomeI,_**. Assim os parâmetros em comum são **Id - id de um utente**, **Idp - id de um prestador**, **Idinst - id instituição**.

```
utentes_de_instNome(NomeI,R) :- solucoes(utente(Id,N,Idd,M),
    (cuidado(_,Id,Idp,_,_,_), prestador(Idp,_,_,Idinst),
    inst(Idinst,NomeI,_),
    utente(Id,N,Idd,M)), R).
```

Exemplo de output:

```
| ?- utentes_de_prest(1,R).
R = [utente(1,'Pascoal',38,'Rua Limpa'),utente(2,'Zeca',20,'Rua da Capa')] ?
yes
| ?-
```

Figura 5.9: Exemplo de output do predicado utentes_de_prest

```
| ?- utentes_de_esp('Dermatologia',R).
R = [utente(3,'Anibal',59,'Rua do Gota'),utente(14,'Miguel',49,'Rua do Pinheiro
')] ?
yes
```

Figura 5.10: Exemplo de output do predicado utentes_de_esp

```
| ?- utentes_de_instNome('Hospital dos Bonecos',R).
R = [utente(7,'Micaela',8,'Rua dos Olivais'),utente(10,'Rita',88,'Rua das Flore
s')] ?
yes
| ?-
```

Figura 5.11: Exemplo de output do predicado utentes_de_instNome

5.7 Identificar cuidados de saúde realizados por utente/ prestador

Utente

Este requisito segue a linha de pensamento dos anteriores, neste caso devolvendo os cuidados de saúde realizados a um utente. Mais uma vez, o predicado **solucoes** devolve

a lista com a informação pretendida a partir da combinação de conhecimento entre os parâmetros do predicado **cuidado** e o identificador do utente para o qual se pretendem saber tais cuidados.

```
cuidados_por_utente(Idu,R):-solucoes(cuidado(D,Idu,Idp,P,Desc,Custo),
                            cuidado(D,Idu,Idp,P,Desc,Custo), R).
```

Prestador

```
cuidados_por_prest(Idp,R) :- solucoes(cuidado(D,Idu,Idp,P,Desc,Custo),
                           cuidado(D,Idu,Idp,P,Desc,Custo), R).
```

Exemplo output:

```
| ?- cuidados_por_utente(5,R).
R = [cuidado('2018-1-2',5,5,'Media','Fratura do pulso',19)] ?
yes
| ?-
```

Figura 5.12: Exemplo de output do predicado cuidados_por_utente

```
| ?- cuidados_por_prest(4,R).
R = [cuidado('2018-1-2',4,4,'Alta','Cancro',32)] ?
yes
| ?-
```

Figura 5.13: Exemplo de output do predicado cuidados_por_prest

5.8 Determinar todas as instituições/prestadores a que um utente já recorreu

Instituição

Para identificar todas as intituições a que um determinado utente recorreu, foi utilizado o predicado **solucoes** de maneira a ser devolvida a lista com o requisitado. Neste predicado, é necessária a combinação de conhecimento proveniente dos predicados **inst(Idi,N,C)**, **cuidado(_,Idu,Idp,_,_,_)**, **prestador(Idp,_,_,Idi)**, **utente(Idu,_,_,_)**. Assim, os parâmetros em comum são **Idi - id de instituição**, **Idu - id de um utente**, **Idp - id de um prestador**

```
todas_inst(Idu,R) :- solucoes(inst(Idi,N,C),(inst(Idi,N,C),
                     cuidado(_,Idu,Idp,_,_,_),prestador(Idp,_,_,Idi),
                     utente(Idu,_,_,_)), R1),apagaRep(R1,R).
```

Prestador

Para identificar todas os prestadores a que um determinado utente recorreu, foi utilizado o predicado **solucoes** de maneira a ser devolvida a lista com o requisitado. Neste predicado, é necessária a combinação de conhecimento proveniente dos predicados **cuidado(_,Idu,Idp,_,_,_)**, **prestador(Idp,N,Esp,Idi)**, comum são **Idu - id de um utente**, **Idp - id de um prestador**

```
todos_prest(Idu,R) :- solucoes(prestador(Idp,N,Esp,Idi),
                      (prestador(Idp,N,Esp,Idi),
                      cuidado(_,Idu,Idp,_,_,_)), R1), apagaRep(R1,R).
```

Exemplo output:

```
| ?- todas_inst(4,R).
R = [inst(2,'IPO','Porto'),inst(3,'Hospital S.Joao','Porto')] ?
yes
| ?-
```

Figura 5.14: Exemplo de output do predicado todos_inst

```
| ?- todos_prest(2,R).
R = [prestador(1,'Ze','Otorrino',1)] ?
yes
| ?-
```

Figura 5.15: Exemplo de output do predicado todos_prest

5.9 Calcular o custo total dos cuidados de saúde por utente/especialidade/prestador/datas

Para desenvolvimento de todos estes predicados foi usado outro auxiliar, o **custo_total**, que, basicamente faz o somatório de uma lista. À partida, esta lista será a lista de custos de uma instituição devolvida pelo **solucoes**.

Utente

Para complementar o que foi dita acima, o **solucoes** devolve a tal lista de custos através do predicado **cuidado(_,Idu,_,_,_,Custo)**.

```
custo_utente(Idu,R) :- solucoes(Custo, cuidado(_,Idu,_,_,_,Custo), R1),
                       custo_total(R1,R).
```

Especialidade Para complementar o que foi dita acima, o **solucoes** devolve a tal lista de custos através da união de conhecimento proveniente dos predicados **cuidado(_,_,Idp,_,_,Custo)**, **prestador(Idp,_,_,Esp,_)**.

```
custo_esp(Esp,R) :- solucoes(Custo, (cuidado(_,_,Idp,_,_,Custo),
                    prestador(Idp,_,Esp,_)), R1), custo_total(R1,R).
```

Prestador Para complementar o que foi dita acima, o **solucoes** devolve a tal lista de custos através da união de conhecimento proveniente dos predicados **cuidado(_,_,Idp,_,_,Custo)**, **prestador(Idp,_,_,_,_)**.

```
custo_prest(Idp,R) :- solucoes(Custo, (cuidado(_,_,Idp,_,_,Custo),
                      prestador(Idp,_,_,_)), R1), custo_total(R1,R).
```

Datas Para complementar o que foi dita acima, o **solucoes** devolve a tal lista de custos através do predicado **cuidado(D,_,_,_,_,Custo)**.

```
custo_data(D,R) :- solucoes(Custo, cuidado(D,_,_,_,_,Custo),R1),
   custo_total(R1,R).
```

Exemplo do output:

```
| ?- custo_utente(3,R).
R = 15 ?
yes
| ?-
```

Figura 5.16: Exemplo de output do predicado custo_utente

```
| ?- custo_esp('Cardiologia',R).
R = 198 ?
yes
| ?-
```

Figura 5.17: Exemplo de output do predicado custo_esp

```
| ?- custo_prest(4,R).
R = 32 ?
yes
| ?-
```

Figura 5.18: Exemplo de output do predicado custo_prest

```
| ?- custo_data('2018-2-7',R).
R = 2000 ?
yes
| ?-
```

Figura 5.19: Exemplo de output do predicado custo_data

6. *Funcionalidades extra*

6.1 Número total de utentes, prestadores e cuidados

6.1.1 Utente

Para podermos calcular o número total de utentes, recorremos ao predicado **solucoes** para recolhermos todos os utentes existentes na base de conhecimento e coloca-los numa lista. Depois disso é calculado o comprimento dessa lista que corresponde ao número total de utentes.

```
total_utentes(R) :- solucoes(Id, utente(Id,_,_,_), L), comprimento(L,R).
```

6.1.2 Prestadores

Este predicado segue o raciocínio do predicado anterior, usando o predicado **solucoes** para recolher todos os prestadores e de seguida é calculado o tamanho da lista resultante dessa recolha.

```
total_prestadores(R) :- solucoes(Idp, prestador(Idp,_,_,_), L),
                        comprimento(L,R).
```

6.1.3 Cuidados

Este predicado segue o raciocínio do predicado anterior, usando o predicado **solucoes** para recolher todos os cuidados e de seguida é calculado o tamanho da lista resultante dessa recolha.

```
total_cuidados(R) :- solucoes(Desc, cuidado(_,_,_,_,Desc,_), L),
                     comprimento(L,R).
```

Exemplo output:

```
| ?- total_utentes(R).
R = 16 ?
yes
```

Figura 6.1: Exemplo de output do predicado total_utentes

```
| ?- total_prestadores(R).
R = 20 ?
yes
| ?-
```

Figura 6.2: Exemplo de output do predicado total_prestadores

```
| ?- total_cuidados(R).
R = 15 ?
yes
| ?-
```

Figura 6.3: Exemplo de output do predicado total_cuidado

6.2 Numero de utentes de uma especialidade

Este predicado, através do **solucoes** e do **comprimento**, devolve o número de utentes de uma dada especialidade. Os parâmetros em comum para conseguirmos tal informação são **Idp - prestador** e **Esp - especialidade**

```
nr_ut_esp(Esp,R) :- solucoes(Idu, (prestador(Idp,_,Esp,_),
                    cuidado(_,Idu,Idp,_,_,_)), L), comprimento(L,R).
```

Exemplo de output:

```
| ?- nr_ut_esp('Psiquiatria',R).
R = 1 ?
yes
| ?-
```

Figura 6.4: Exemplo de output do predicado nt_ut_esp

6.3 Percentagem de utentes de uma especialidade

O resultado deste predicado é calculado através de dois auxiliares, um que devolve o número de utentes de uma dada especialidade **(Res)** e outro que devolve o total de utentes da nossa base de conhecimento **(X)**, tomando o valor **(Res*100)/X**.

```
percent_utentes_esp(Esp,R) :- nr_ut_esp(Esp,Res), total_utentes(X),
                              R is (Res*100)/X.
```

Exemplo de output:

```
| ?- percent_utentes_esp('Psiquiatria',R).
R = 6.25 ?
yes
```

Figura 6.5: Exemplo de output do predicado percent_utentes_esp

6.4 Prestadores sem cuidados

Este predicado foi criado com o intuito de perceber quais os prestadores da base de conhecimento que dão prejuízo, i.e, que não têm qualquer cuidado de saúde associado. O cálculo deste predicado foi um pouco mais complexo que os anteriores, pois para chegar ao resultado pretendido foi necessário cruzar a informação entre a lista de todos os prestadores da base de conhecimento com a lista dos prestadores que efetuaram cuidados. Assim, a partir do predicado **pertence**, conseguimos filtrar todos os **Idp - id de um prestador**, colocá-los numa lista e, a partir do predicado **prest**, criar a lista com os prestadores associado ao respetivo identificador. A extensões dos predicados auxiliares encontram-se no primeiro capítulo, secção **Predicados gerais/ auxiliares**.

```
prest_sem_cuidados(R) :- solucoes(Idp, prestador(Idp,_,_,_),L),
                         solucoes(Idp, (prestador(Idp,_,_,_),
                         cuidado(_,_,Idp,P,_,_)), L1),
                         pertence(L,L1,R1), prest(R1,R).
```

Exemplo de output:

```
sec, 0 clauses
?- prest_sem_cuidados(R).
R = [prestador(2, 'Andreia', 'Dentaria', 1), prestador(7, 'Telmo', 'Radiologia', 2), prestador(8,
'Miquelina', 'Cardiologia', 4), prestador(9, 'Armando', 'Neurologia', 4), prestador(10, 'Firmino',
'Endocrinologia', 5), prestador(11, 'Horacio', 'Otorrino', 5)]
```

Figura 6.6: Exemplo de output do predicado prest_sem_cuidados

6.5 Identificar utentes de uma Instituição através do id da mesma

Predicado complementar ao requisito número 6. Neste caso, em vez de se identificar a instituição por nome, identifica-se por id.

```
utentes_de_instID(Idi,R) :- solucoes(utente(Id,N,Idd,M),
(cuidado(_,Id,Idp,_,_,_), prestador(Idp,_,_,Idi),
inst(Idi,_,_), utente(Id,N,Idd,M)), R).
```

Exemplo de output:

```
| ?- utentes_de_instID(3,R).
R = [utente(5,'Carlota',22,'Rua do Speedy'),utente(4,'Maria',42,'Rua dos Peoes'
),utente(14,'Miguel',49,'Rua do Pinheiro')] ?
yes
```

Figura 6.7: Exemplo de output do predicado utentes_de_instID

6.6 Número de utentes de todas as Instituições

Para o cálculo deste predicado foi necessária a criação de um auxiliar- **nr_ut_inst**, que a partir uma lista de id's de instituições devolve a lista com os respetivos números de utente. Essa lista é obtida pelo **solucoes**.

```
nr_ut_todas_inst(R) :- solucoes(Idi, inst(Idi,_,C), L),
                       nr_ut_inst(L,R).


nr_ut_inst([],[]).
nr_ut_inst([Idi],[R]) :- utentes_de_instID(Idi,R1), comprimento(R1,R).
nr_ut_inst([Idi|Y],[R|L]) :- utentes_de_instID(Idi,R1),
                             comprimento(R1,R), nr_ut_inst(Y,L).
```

Exemplo de output:

```
| ?- nr_ut_todas_inst(R).
R = [6,3,3,2,2] ?
yes
```

Figura 6.8: Exemplo de output do predicado nt_ut_todas_inst

6.7 Cuidado mais caro registado até ao momento

Para o cálculo deste predicado, recorremos duas vezes ao **solucoes**. Primeiramente, para conseguir a lista com os custos de cada cuidado de saúde, que vai ser vir como parâmetro do predicado **maxLista**, de maneira a extrair o maior elemento.Seguidamente, o **solucoes** devolve o cuidado referente a esse preço.

```
cuidado_mais_caro(R) :- solucoes(C, cuidado(_,_,_,_,_,C), L),
                        maxLista(L,R1), solucoes((D,Idu,Idp,P,Desc,R1),
                        cuidado(D,Idu,Idp,P,Desc,R1), R).
```

Exemplo de output:

```
| ?- cuidado_mais_caro(R).
R = [('2018-2-7',15,12,'Baixa','Por aparelho',2000)] ?
yes
```

Figura 6.9: Exemplo de output do predicado cuidado_mais_caro

6.8 Ordenar lista de cuidados de uma instituição crescentemente por preço

Para a realização deste predicado foi necessária a criação de um predicado auxiliar, o **ordena**, de maneira a ser possível ordenar uma lista de cuidados através do seu

último parâmetro - o preço. Para obter a lista com os cuidados de uma certa instituição, é necessário cruzar a informação dada pelos predicados **inst(Idi,_,_)**, **prestador(Idp,_,_,Idi)** e **cuidado(D,Idu,Idp,P,Desc,R1)**, onde os parâmetros em comum são **Idp - id de um prestador**, **Idi - id de uma instituição**.

```
cuidados_por_preco(Idi,R) :- solucoes((D,Idu,Idp,P,Desc,R1),
                             (inst(Idi,_,_),prestador(Idp,_,_,Idi),
                             cuidado(D,Idu,Idp,P,Desc,R1)), L), ordena(L,R).
```

Exemplo de output:

```
| ?- cuidados_por_preco(3,R).
R = [('2018-2-6',14,13,'Alta','Urticaria',5),('2018-1-2',5,5,'Media','Fratura d
o pulso',19),('2018-1-3',4,6,'Baixa','Papa Nicolau',100)] ?
yes
```

Figura 6.10: Exemplo de output do predicado cuidados_por_preco

6.9 Cuidados de saúde por prioridade

O facto de ter sido acrescentada informação sobre a prioridade de um cuidado permitiu-nos desenvolver este predicado, que, através de uma prioridade **'Baixa','Media' ou 'Alta'**, devolve uma lista com todos os cuidados com uma dada prioridade. Lista essa que resulta da aplicação do predicado **solucoes**.

```
cuidados_prio(Prio,R) :-
solucoes(cuidado(D,Idu,Idp,Prio,Desc,C),
cuidado(D,Idu,Idp,Prio,Desc,C), R).
```

Exemplo de output:

```
| ?- cuidados_prio('Media',R).
R = [cuidado('2018-1-1',1,1,'Media','Amigdalite',10),cuidado('2018-1-2',5,5,'Me
dia','Fratura do pulso',19),cuidado('2018-1-4',10,17,'Media','Unha encravada',3
7),cuidado('2018-2-5',12,15,'Media','Consulta rotina',90),cuidado('2018-2-6',13
,14,'Media','Ecografia aos ovarios',58)] ?
yes
```

Figura 6.11: Exemplo de output do predicado cuidados_prio

6.10 Número de cuidados de uma prioridade

O facto de ter sido acrescentada informação sobre a prioridade de um cuidado permitiu-nos desenvolver este predicado, que, através de uma prioridade **'Baixa','Media' ou 'Alta'**, devolve o número de cuidados de saúde existentes com essa prioridade. Através do **solucoes** geramos a lista de cuidados e com o predicado **comprimento** aplicado ao resultado do anterior, é obtido o resultado pretendido.

```
nr_cuidado_prio(Prio,R) :-
solucoes(cuidado(D,Idu,Idp,Prio,Desc,C),
cuidado(D,Idu,Idp,Prio,Desc,C), R1),
comprimento(R1,R).
```

Exemplo de output:

```
| ?- nr_cuidado_prio('Alta',R).
R = 5 ?
yes
```

Figura 6.12: Exemplo de output do predicado nr_cuidados_prio

7. *Conclusão*

Sendo este o primeiro projeto realizado por nós em *PROLOG*, e sendo ele uma novidade recente, a realização deste trabalho prático permitiu-nos aprofundar e consolidar o conhecimento adquirido nas aulas da unidade curricular.

A maior dificuldade que encontramos durante a realização deste trabalho, foi descobrir qual a estratégia a implementar para concretizar o objetivo proposto da melhor maneira possível. Após vários dias de pesquisa essa dificuldade foi ultrapassada.

Por fim, podemos dizer que nos sentimos satisfeitos com o trabalho desenvolvido pois não só cumprimos o objetivo proposto mas também o conseguimos melhorar.