Universidade do Minho
Escola de Engenharia

TRABALHO PRÁTICO EXERCÍCIO 1

Mestrado Integrado em Engenharia Informática

Sistemas de Representação de Conhecimento e Raciocínio

(2º semestre, 3º ano)

A78322 André Filipe Ferreira de Mira Vieira

A77048 Eduardo Gil Ribeiro Rocha

A78764 Ricardo André Araújo Neves

Data:

18 de março de 2018

Índice

Índice de Figuras

Resumo

Este relatório serve como complemente ao primeiro exercício do trabalho prático da Unidade Curricular de Sistemas de Representação de Conhecimento e Raciocínio.

Aqui iremos expressar a nossa linha de pensamento e como implementamos cada um dos predicados pedidos no enunciado do exercício prático.

Para além disto, tomamos a iniciativa de acrescentar novos predicados, que pensamos que tornam o sistema mais completo.

Introdução

O objetivo desta primeira fase do trabalho prático é a utilização da **linguagem de programação em lógica PROLOG,** de modo a entender as práticas de programação em lógica e utilização de invariantes. Para tal desenvolvemos um sistema de representação de conhecimento e raciocínio para caracterizar uma área de cuidados de saúde.

Preliminares

Para realizar este trabalho foi preciso entender os básicos de como funciona a linguagem de programação PROLOG. Utilizamos os conhecimentos aprendidos nas aulas práticas como base para desenvolver este sistema.

É preciso entender como a linguagem PROLOG procura provas para como um fato é verdadeiro, logo para obter um resultado verdadeiro, devemos procurar a inclusão na base de conhecimentos do fato de que estamos à procura. Caso queremos provar que algo é falso, devemos então verificar que não existe esse fato na base de conhecimentos.

Representação das Entidades

Para representar o **Utente,** foi criada uma entidade com 4 atributos: **IdUt** (ID único que caracteriza um e um só utente)**, Nome**, **Idade** e **Morada**.

Na representação do **Prestador**, foi criada outra entidade com o mesmo número de atributos: **IdPrest** (ID único que caracteriza um e um só prestador de cuidados de saúde)**, Nome, Especialidade** e **Instituição** à qual está designado.

Por último, foi necessário criar a entidade **Cuidado**, desta vez, com 5 atributos: **Data** em que foi prestado o serviço**, IdUt** (ID do utente que foi atendido)**, IdPrest** (ID do prestador que prestou o serviço)**, Descrição** e **Custo**.

```
% utente(IdUt, Nome, Idade, Morada).
% prestador(IdPrest, Nome, Especialidade, Instituicao).
% cuidado(Data, IdUt, IdPrest, Descricao, Custo).
```

Figura 1 - Entidades usadas

Deste modo, para nos ajudar no desenvolvimento das funcionalidades a seguir definidas, criamos uma **situação de teste**, onde demos exemplos a cada um dos elementos acima. Assim, à medida que desenvolvíamos cada uma das seguintes "querys", fomos testando, de modo a corrigir possíveis erros de implementação.

```
utente(1, ricardo, 20, porto).
utente(2, andre, 20, porto).
utente(3, gil, 20, braga).
utente(4, joao, 14, braga).
utente(5, maria, 34, porto).
prestador(1, carlos, medicinaGeral, porto).
prestador(2, manuel, fisioterapia, braga).
prestador(3, david, medicinaGeral, porto).
prestador(4, paula, fisioterapia, porto).
prestador(5, luis, medicinaGeral, braga).
cuidado(16032018, 1, 1, consulta, 20).
cuidado(17032018, 1, 2, fisioterapia, 20).
cuidado(16032018, 5, 4, fisioterapia, 19).
cuidado(16032018, 2, 3, consulta, 15).
cuidado(18032018, 3, 4, fisioterapia, 50).
cuidado(20032018, 1, 5, consulta, 23).
```

Figura 2 - Exemplos de teste

Como podemos observar na imagem acima, podemos retirar uma série de factos:

- O Utente com ID número 1 chama-se Ricardo, tem 20 anos e é natural do Porto.
- O Prestador com ID número 5 é especialista em Medicina Geral, trabalha na Instituição de Braga, e chama-se Luís.
- No dia 18-03-2018, o prestador com ID 5 (Luís) prestou um cuidado de saúde do utente com ID 3 (Gil), com a descrição X e um custo de 50 euros.

Invariantes

Os Invariantes utilizados no desenvolvimento das funcionalidades do sistema foram:

Este invariante não permite que se insira conhecimento repetido relativamente ao utente. Ou seja, torna-se impossível inserir dois utentes exatamente iguais.

```
+utente(IdUt,Nome,Idade,Morada) :: findall((IdUt, Nome, Idade, Morada),(utente(IdUt, Nome, Idade, Morada)),S),
                                   comprimento(S,N),
                                   N == 1.
```

Figura 3 – Invariante do utente

Este invariante é repetido para as outras duas entidades do sistema (Prestador e Cuidado).

```
+prestador(IdPrest,Nome,Especialidade,Instituicao) :: findall((IdPrest, Nome, Especialidade, Instituicao),(prestador(IdPrest, Nome, Especialidade, Insituicao)),S),
                                                      comprimento(S,N),
                                                      N == 1.
```

Figura 4 - Invariante do Prestador

```
+cuidado(Data,IdUt,IdPrest,Descricao,Custo) :: findall((Data, IdUt, IdPrest, Descricao,Custo),(cuidado(Data, IdUt, IdPrest, Descricao, Custo)),S),
                                               comprimento(S,N),
                                               N == 1.
```

Figura 5 - Invariante do Cuidado

Como podemos observar, em cada um dos invariantes acima, constatamos a presença do predicado "comprimento", que calcula o número de elementos existentes numa lista.

```
comprimento([], 0).
comprimento([X|L], C) :- comprimento(L, N), C is 1+N.
```

Figura 6 - Comprimento

Registo de Utentes, Prestadores e Cuidados

A primeira funcionalidade desenvolvida foi a de registar um novo Utente, Prestador ou Cuidado de Saúde no sistema. Para isto, foi criado o predicado "evolução", que testa os Invariantes criados, insere o Termo no sistema, com o método "assert", e corre o teste final.

```
evolucao( Termo ) :- findall(I, +Termo::I, Li),
                     insercao(Termo),
                     teste(Li).

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

teste([]).
teste([H|S]) :- H, teste(S).
```

Figura 7 - Evolução do conhecimento

Exemplo:

```
| ?- evolucao(utente(6, Xavier, 8, aveiro)).
yes
```

Figura 8 - Exemplo 1

Foi inserido um novo Utente, cujo ID único é 6, chamado Xavier, com 8 anos de idade e natural da cidade de Aveiro.

Remoção de Utentes, Prestadores e Cuidados

Para ser possível remover registos de utentes, prestadores de serviços ou cuidados de saúde, foi também implementada essa funcionalidade. Assim, com o predicado "involução", remove o Termo do sistema, com o método "retract", correndo o teste final.

```
involucao( Termo ) :- Termo,
                      findall(I, -Termo::I, Li),
                      remocao(Termo),
                      teste(Li).

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

Figura 9 - Involução do conhecimento

Exemplo:

```
| ?- involucao(utente(6, Xavier, 8, aveiro)).
yes
```

Figura 10 - Exemplo 2

Foi removido o Utente, cujo ID único é 6, chamado Xavier, com 8 anos de idade e natural da cidade de Aveiro.

Identificação de Utentes por Critérios de Seleção

Os seguintes predicados permitem ao utilizador identificar todos os utentes que cumprem o requisito indicado.

```
idGetUtentes(Id, Li) :- findall((Id, Nome), utente(Id, Nome, Idade, Morada), Li).
nomeGetUtentes(Nome, Li) :- findall((Id, Nome), utente(Id, Nome, Idade, Morada), Li).
idadeGetUtentes(Idade, Li) :- findall((Id, Nome), utente(Id, Nome, Idade, Morada), Li).
moradaGetUtentes(Morada, Li) :- findall((Id, Nome), utente(Id, Nome, Idade, Morada), Li).
```

Figura 11 - Critérios de seleção

Por exemplo, ao utilizar o predicado "nomeGetUtentes" e ao indicar o nome "André", este irá apresentar, após percorrer todos os utentes inseridos no sistema, todos os utentes que cumprem o requisito Nome = "André".

Exemplo:

```
| ?- idadeGetUtentes(20, C).
C = [(1,ricardo),(2,andre),(3,gil)] ?
yes
```

Figura 12 - Exemplo 3

Os Utentes com 20 anos de idade são o Ricardo (ID 1), o André (ID 2) e o Gil (ID 3).

Identificação das Instituições Prestadores

Neste caso, o predicado implementado irá percorrer todos os Prestadores de Serviços presentes no sistema, e retirar a Instituição a que pertence.

Como é obvio, irão ser apresentados várias instâncias da mesma Instituição. No entanto, foi também implementada o método "eliminaRepetidos", que, como o nome indica, remove elementos duplicados de uma lista. Assim, evita-se que sejam apresentadas várias Instituições iguais.

```
getInstituicoes(Li2) :- findall(Instituicao, prestador(IdPrest, Nome, Especialidade, Instituicao), Li),
                        eliminaRepetidos(Li, Li2).
```

Figura 13 - Identificação das instituições

```
eliminaRepetidos([], []).

eliminaRepetidos([H|T], R) :- repetido(H,T),
                              eliminaRepetidos(T,R).

eliminaRepetidos([H|T], [H|TS]) :-
                              naoRepetido(H,T),
                              eliminaRepetidos(T, TS).
```

Figura 14 – Predicado "eliminaRepetidos"

```
repetido(X, [X|T]).
repetido(X, [H|T]) :- repetido(X,T).

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

Figura 15 - Predicados "repetido" e "naoRepetido"

Exemplo:

```
| ?- getInstituicoes(C).
C = [porto,braga] ?
yes
```

Figura 16 - Exemplo 4

As Instituições presentes no sistema são as do Porto e de Braga.

Identificação de Cuidados por Instituição

Foi implementada outra funcionalidade que permite ver todos os Cuidados prestados dada uma Instituição.

```
instituicaoGetCuidados(Inst, R) :- findall(IdPrest, prestador(IdPrest, Nome, Especialidade, Inst), Li),
                                   instituicaoGetCuidadosAux(Li, R).

instituicaoGetCuidadosAux([], []).
instituicaoGetCuidadosAux([H|T], [HS|TS]) :- findall((Data, IdUt, H, Descricao, Custo), cuidado(Data, IdUt, H, Descricao, Custo), HS),
                                             instituicaoGetCuidadosAux(T, TS).
```

Figura 17 - Identificação de cuidados por instituição

Como se pode ver na imagem acima, em primeiro lugar, é retirado todos os IDs dos prestadores que estão associados à Instituição indicada. Numa segunda fase, estes IDs são percorridos, sendo apresentado ao utilizador todos os cuidados prestados pelo Prestador.

Exemplo:

```
| ?- instituicaoGetCuidados(braga, R).
R = [[(17032018,1,2,fisioterapia,20)],[(20032018,1,5,consulta,23)]] ?
yes
```

Figura 18 - Exemplo 5

Na Instituição de Braga, foram prestados 2 serviços de saúde: um no dia 17 de março (fisioterapia) e outra no dia 20 de março (consulta).

Identificação de Cuidados por Cidade

Este caso é em tudo semelhante ao predicado anterior. No entanto, ao invés de ser indicada a Instituição, é dado como argumento o nome da Cidade do Utente.

```
cidadeGetCuidados(Cidade, R) :- findall(IdUt, utente(IdUt, Nome, Idade, Cidade), Li),
                                cidadeGetCuidadosAux(Li, R).

cidadeGetCuidadosAux([], []).
cidadeGetCuidadosAux([H|T], [HS|TS]) :- findall((Data, H, IdPrest, Descricao, Custo), cuidado(Data, H, IdPrest, Descricao, Custo), HS),
                                        cidadeGetCuidadosAux(T, TS).
```

Figura 19 - Identificação de cuidados por cidade

Neste predicado, é retirado todos os IDs dos Utentes que vivem na Cidade indicada. De seguida, com todos estes IDs, são apresentados todos os cuidados a que estes Utentes foram sujeitos.

Exemplo:

```
| ?- cidadeGetCuidados(porto, R).
R = [[(16032018,1,1,consulta,20),(17032018,1,2,fisioterapia,20),(20032018,1,5,consulta,23)],
[(16032018,2,3,consulta,15)],[(16032018,5,4,fisioterapia,19)]] ?
yes
```

Figura 20 - Exemplo 6

Os Utentes residentes no Porto sujeitaram-se, no total, a 5 cuidados de saúde.

Identificação de Cuidados por Datas

Este predicado retorna ao utilizador do sistema os Cuidados prestados numa série de Datas introduzidas pelo mesmo.

```
datasGetCuidados([], []).
datasGetCuidados([D|DS], [H|T]) :- findall((D, IdUt, IdPrest, Descricao, Custo), cuidado(D, IdUt, IdPrest, Descricao, Custo), H),
                                   datasGetCuidados(DS, T).
```

Figura 21 - Identificação de cuidados por datas

Este predicado, apesar de apresentar resultados semelhantes aos dois predicados anteriores, é um pouco mais simples. Isto deve-se ao facto de a Data ser um atributo da entidade Cuidado. Assim, o predicado "datasGetCuidados" percorre a lista de datas introduzidas como argumento, retornado os dados dos Cuidados prestados nesse dia.

Exemplo:

```
| ?- datasGetCuidados([19032018, 20032018], R).
R = [[],[(20032018,1,5,consulta,23)]] ?
yes
```

Figura 22 - Exemplo 7

No dia 19-03-2018, não houve qualquer serviço de saúde. No entanto, houve uma consulta no dia 20-03-2018, com um custo de 23 euros.

Identificação dos Utentes de um Prestador

Esta funcionalidade permite que sejam identificados todos os Utentes que foram acompanhados por um Prestador de cuidados de saúde.

```
prestadorGetUtentes(IdPrest, R) :- findall(IdUt, cuidado(Data, IdUt, IdPrest, Especialidade, Custo), Li),
                                   utentesGetUtentes(Li, R).

utentesGetUtentes([], []).
utentesGetUtentes([H|T], [HS|TS]) :- findall((H, Nome, Idade, Morada), utente(H, Nome, Idade, Morada), HS),
                                     utentesGetUtentes(T, TS).
```

Figura 23 - Identificação dos utentes de um prestador

Dado o ID do Prestador a que se quer obter informação, são retirados todos os Utentes a que o mesmo prestou um serviço de saúde. Por fim, percorrendo essa lista de IDs dos Utentes, é apresentado ao utilizador todos os dados de todos os Utentes (ID, nome, idade e morada).

Sentimos que, neste caso, seria muito mais beneficiário para o utilizador obter toda a informação dos Utentes que cumprem o requisito, ao invés de apenas ser retornado os IDs dos mesmos.

Exemplo:

```
| ?- prestadorGetUtentes(4, R).
R = [[(5,maria,34,porto)],[(3,gil,20,braga)]] ?
yes
```

Figura 24 - Exemplo 8

O Prestador com ID igual a 4 (corresponde à Prestadora Paula) atendeu 2 clientes, a Maria e o Gil.

Identificação dos Utentes de uma Especialidade

Como se pode observar na Figura 1, a Especialidade é um atributo da entidade Prestador. Ou seja, será necessário, dada uma Especialidade, retirar todos os Prestadores que são especialistas na mesma.

```
especialidadeGetUtentes(Especialidade, R) :- findall(IdPrest, prestador(IdPrest, Nome, Especialidade, Instituicao), Li),
                                             especialidadeGetUtentesAux(Li, R).

especialidadeGetUtentesAux([], []).
especialidadeGetUtentesAux([H|T], [HS|TS]) :- findall(IdUt, cuidado(Data, IdUt, H, Descricao, Custo), HS),
                                               especialidadeGetUtentesAux(T, TS).
```

Figura 25 - Identificação dos utentes por especialidade

Deste modo, com todos os IDs dos Prestadores, retira-se, mais uma vez, todos os IDs dos Utentes que recorreram a essa especialidade.

Exemplo:

```
| ?- especialidadeGetUtentes(medicinaGeral, R).
R = [[1],[2],[1]] ?
yes
```

Figura 26 - Exemplo 9

Os Prestadores especializados em Medicina Geral atenderam por duas vezes o Utente com ID igual 1, e uma vez o Utente com ID igual a 2.

Identificação dos Utentes de uma Instituição

Neste caso, é dado como argumento do predicado o nome da Instituição a que o Utente se dirigiu para lhe ser prestado um Cuidado de saúde. Deste modo, é apresentado os dados destes mesmos Utentes.

```
instituicaoGetUtentes(Instituicao, Ri) :- findall(IdPrest, prestador(IdPrest, Nome, Especialidade, Instituicao), Li),
                                          prestadoresGetUtentes(Li, R),
                                          utentesGetUtentes(R, Ri).

prestadoresGetUtentes([], []).
prestadoresGetUtentes([H|T], [HS|TS]) :- findall(IdUt, cuidado(Data, IdUt, H, Especialidade, Custo), HS),
                                         prestadoresGetUtentes(T, TS).
```

Figura 27 - Identificação dos utentes de uma instituição

Aqui, é de novo retirado os IDs dos Prestadores e, de seguida, com a lista desses IDs, apresentar os IDs dos Utentes que se deslocaram à Instituição designada.

Exemplo:

```
| ?- instituicaoGetUtentes(braga, R).
R = [[1],[1]] ?
yes
```

Figura 28 - Exemplo 10

A Instituição de Braga prestou dois serviços ao Utente com ID igual a 1.

Identificação de Cuidados de um Utente

Outra forma de apresentar os Cuidados de saúde, é se for indicado um Utente que realizou os mesmos.

```
utenteGetCuidados(IdUt, Li) :- findall((Data, IdUt, IdPrest, Especialidade, Custo), cuidado(Data, IdUt, IdPrest, Especialidade, Custo), Li).
```

Figura 29 - Identificação de cuidados de saúde por utente

Uma vez que o ID do Utente é um atributo da entidade Cuidado, este predicado é resumido em uma linha, onde para cada Cuidado inserido no sistema, verifica-se se o IdUt corresponde ao que foi inserido como argumento pelo utilizador. Em caso afirmativo, é devolvido ao utilizador do programa.

Exemplo:

```
| ?- utenteGetCuidados(2, R).
R = [(16032018,2,3,consulta,15)] ?
yes
```

Figura 30 - Exemplo 11

O Utente com ID igual a 2 foi sujeito a um serviço de saúde no dia 16-03-2018, com o custo de 15 euros, tratando-se de uma consulta.

Identificação de Cuidados prestados por um Prestador

A única diferença entre este predicado e o anterior resume-se ao argumento introduzido pelo utilizador. Aqui, é pedido que sejam apresentados todos os Cuidados que foram prestados por um Prestador introduzido.

```
prestadorGetCuidados(IdPrest, Li) :- findall((Data, IdUt, IdPrest, Especialidade, Custo), cuidado(Data, IdUt, IdPrest, Especialidade, Custo), Li).
```

Figura 31 - Identificação de cuidados prestados por prestador

Novamente, este predicado é composto apenas por um "findall" que, dado o ID do Prestador, retorna a lista de Cuidados prestados pelo mesmo. Esta informação contém todos os atributos da entidade Cuidado, ou seja, o utilizador tem acesso à Data, ao ID do Utente, ID do Prestador, Descrição e Custo dos vários serviços.

Exemplo:

```
| ?- prestadorGetCuidados(4, R).
R = [(16032018,5,4,fisioterapia,19),(18032018,3,4,fisioterapia,50)] ?
yes
```

Figura 32 - Exemplo 12

O Prestador com ID igual a 4 realizou dois serviços, um no dia 16-03-2018, com um custo de 19 euros, e outro no dia 18-03-2018, com custo de 50 euros.

Determinação das Instituições a que um Utente recorreu

Desta vez, este predicado irá retornar a lista de Instituições a que um Utente já recorreu. Como já foi referido neste documento, existe um outro predicado que é o inverso deste, ou seja, dada uma Instituição, o utilizador recebe todos os Utentes que a frequentaram.

```
utenteGetInstituicoes(IdUt, R) :- findall(IdPrest, cuidado(Data, IdUt, IdPrest, Descricao, Custo), Li),
                                  utenteGetInstituicoesAux(Li, R).

utenteGetInstituicoesAux([], []).
utenteGetInstituicoesAux([H|T], [HS|TS]) :- findall(Instituicao, prestador(H, Nome, Especialidade, Instituicao), HS),
                                            utenteGetInstituicoesAux(T, TS).
```

Figura 33 - Determinação das instituições a que um utente já recorreu

Sendo a Instituição um atributo da entidade Prestador, é necessário obter uma lista com todos os Prestadores a que o Utente já recorreu para que lhe seja prestado um serviço de saúde. Com todos estes IDs dos Prestadores, são retiradas as Instituições a que estão associados.

Exemplo:

```
| ?- utenteGetInstituicoes(5, T).
T = [[porto]] ?
yes
```

Figura 34 - Exemplo 13

O Utente com ID igual a 5 apenas realizou um serviço, na Instituição do Porto.

Determinação dos Prestadores a que um Utente recorreu

Este predicado é outro caso em que o inverso já foi implementado. Isto é, existe um predicado que, dado um Prestador de serviços, retorna a lista dos Utentes que foram acompanhados pelo mesmo.

Aqui, dado um ID do Utente, determina-se a lista de Prestadores a que ele recorreu.

```
utenteGetPrestadores(IdUt, R) :- findall(IdPrest, cuidado(Data, IdUt, IdPrest, Especialidade, Custo), Li),
                                 eliminaRepetidos(Li, LiD),
                                 prestadorGetPrestador(LiD, R).

prestadorGetPrestador([], []).
prestadorGetPrestador([H|T], [HS|TS]) :- findall((H, Nome, Especialidade, Instituicao), prestador(H, Nome, Especialidade, Instituicao), HS),
                                         prestadorGetPrestador(T, TS).
```

Figura 35 - Determinação dos prestadores de um utente

Como se pode ver, são retirados todos os IDs dos Prestadores a que um Utente já recorreu, aplicando-se de seguida a função "eliminaRepetidos", que já foi explicada anteriormente, neste relatório.

Com todos os IDs dos Prestadores, decidimos apresentar mais informação ao utilizador do programa. Assim, o utilizador tem acesso a informação detalhada de cada Prestador (o seu ID, Nome, Especialidade e Instituição em que trabalha).

Exemplo:

```
| ?- utenteGetPrestadores(1, R).
R = [[(1,carlos,medicinaGeral,porto)],[(2,manuel,fisioterapia,braga)],[(5,luis,m
edicinaGeral,braga)]] ?
yes
```

Figura 36 - Exemplo 14

O Utente com ID igual a 1 recorreu a 3 Prestadores: ao Prestador Carlos, ao Prestador Manuel e ao Prestador Luís.

Cálculo do Custo Total dos Cuidados de um Utente

Como se sabe, a cada Cuidado de saúde está associado um Custo, que é suportado pelo Utente. Assim, é possível calcular o total de dinheiro gasto por um certo Utente em serviços de saúde.

```
utenteGetCustoTotal(IdUt, C) :- findall(Custo, cuidado(Data, IdUt, IdPrest, Descricao, Custo), Li),
                                calculaCusto(Li, C).

calculaCusto([], 0).
calculaCusto([H|T], C) :- calculaCusto(T,N), C is H+N.
```

Figura 37 - Cálculo do custo dos cuidados por utente

Deste modo, é retirado para uma lista todos os custos que foram pagos pelo Utente referido. O método "calculaCusto" encarrega-se de somar todos os custos presentes nessa lista, devolvendo o custo total dos serviços de saúde do Utente.

Exemplo:

```
| ?- utenteGetCustoTotal(1, C).
C = 63 ?
yes
```

Figura 38 - Exemplo 15

O Utente com ID igual a 1 já gastou, no total, 63 euros em serviços de saúde.

Cálculo do Custo Total dos Cuidados de um Prestador

Este predicado já foi implementado para os Utentes. Desta vez, é possível inserir o ID de um Prestador e determinar rapidamente todo o dinheiro que faturou aos seus clientes.

```
prestadorGetCustoTotal(IdPrest, C) :- findall(Custo, cuidado(Data, IdUt, IdPrest, Descricao, Custo), Li),
                                      calculaCusto(Li, C).
```

Figura 39 - Cálculo do custo total dos cuidados por prestador

Como se pode observar, à semelhança do predicado anterior, são copiados todos os custos para uma lista, que irá ser enviada para a função "calculaCusto", também mencionada atrás.

Exemplo:

```
| ?- prestadorGetCustoTotal(3, C).
C = 15 ?
yes
```

Figura 40 - Exemplo 16

O Prestador com ID igual a 3 faturou 15 euros em serviços de saúde.

Cálculo do Custo Total dos Cuidados de uma Especialidade

Este predicado é mais exigente do que os dois anteriores. Isto deve-se ao facto de o atributo Especialidade estar atribuído aos Prestadores. Aqui é apresentado o valor gasto pelos Utentes quando são atendidos por um Prestador com uma dada Especialidade.

```
especialidadeGetCustoTotal(Especialidade, R) :- findall(IdPrest, prestador(IdPrest, Nome, Especialidade, Instituicao), Li),
                                                eliminaRepetidos(Li, LiD),
                                                especialidadeGetCustoTotalAux(LiD, R).

especialidadeGetCustoTotalAux([], _).
especialidadeGetCustoTotalAux([H|T], R) :- especialidadeGetCustoTotalAux(T, N), X is prestadorGetCustoTotal(H, 0), R is X+N.
```

Figura 41 - Cálculo do custo total por especialidade

Aqui, são retirados todos os Prestadores que se dedicam à Especialidade introduzida. Com a lista dos seus IDs, estes são percorridos e, a cada um, é aplicado o predicado que, dado um ID do Prestador, informa o valor total faturado pelo mesmo.

Assim, estes valores são todos somados e apresentados ao utilizador.

Cálculo do Custo Total dos Cuidados numa Data

Esta é a última funcionalidade que foi pedida no enunciado do trabalho prático. É em tudo semelhante às três outras anteriores, mas, desta vez, o argumento é uma Data. Aqui, irá ser apresentado o valor de dinheiro gasto pelos Utentes num determinado dia do ano.

```
dataGetCustoTotal(Data, C) :- findall(Custo, cuidado(Data, IdUt, IdPrest, Descricao, Custo), Li),
                              calculaCusto(Li, C).
```

Figura 42 - Cálculo do custo total num dia do ano

Novamente, é armazenada e enviada uma lista com custos para "calculaCusto", que apresentará o total gasto num dia.

Exemplo:

```
| ?- dataGetCustoTotal(16032018, C).
C = 54 ?
yes
```

Figura 43 - Exemplo 17

No dia 16-03-2018 foi gasto, no total, 54 euros em prestações de serviços de saúde.

Determinação dos Utentes cuja Idade é maior que o número introduzido

Esta é a primeira das cinco funcionalidades extra que o grupo decidiu implementar.

Aqui, o utilizador introduz uma idade escolhida e é lhe passado todos os Utentes que tem uma idade superior à introduzida.

```
idadeMinGetUtentes(IdadeMin, Li) :- findall((IdUt, Nome, Idade, Morada), (utente(IdUt, Nome, Idade, Morada), Idade > IdadeMin), Li).
```

Figura 44 - Determinação dos utentes cuja idade é maior que o numero introduzido

Aqui, o segundo argumento da função "findall" apresenta uma condição adicional, que é a que a idade do utente tem que ser maior que a idade mínima introduzida pelo utilizador. Em caso afirmativo, esse Utente é apresentado pelo sistema.

Exemplo:

```
| ?- idadeMinGetUtentes(21, C).
C = [(5,maria,34,porto)] ?
yes
| ?-
```

Figura 45 - Exemplo 18

Apenas a Maria tem uma idade superior a 20 (tem 34 anos).

Determinação dos Utentes cuja Idade é maior que o número introduzido

Outra funcionalidade adicional é o inverso do último predicado.

Assim, o utilizador introduz uma idade escolhida e é lhe passado todos os Utentes que tem uma idade inferior à introduzida.

```
idadeMaxGetUtentes(IdadeMax, Li) :- findall((IdUt, Nome, Idade, Morada), (utente(IdUt, Nome, Idade, Morada), Idade < IdadeMax), Li).
```

Figura 46 - Determinação de todos os utentes cuja idade é menor que o numero introduzido

Aqui, o segundo argumento da função "findall" apresenta uma condição adicional, que é a que a idade do utente tem que ser menor que a idade mínima introduzida pelo utilizador. Em caso afirmativo, esse Utente é apresentado pelo sistema.

Exemplo:

```
| ?- idadeMaxGetUtentes(16, C).
C = [(4,joao,14,braga)] ?
yes
```

Figura 47 - Exemplo 19

Apenas o João tem uma idade inferior a 16 (tem 14 anos).

Apresentar todos os IDs e Nomes dos Utentes

O terceiro de cinco novos predicados é o de apresentar todos os IDs e Nomes dos Utentes registados no sistema.

```
getIdNomeUtentes(Li) :- findall((IdUt, Nome), utente(IdUt, Nome, Idade, Morada), Li).
```

Figura 48 - Apresenta todos os IDs e nomes dos utentes

Esta funcionalidade é importante para ser apresentado, de forma rápida e resumida, o número de ID único de cada Utente, juntamente com o seu Nome.

Exemplo:

```
| ?- getIdNomeUtentes(C).
C = [(1,ricardo),(2,andre),(3,gil),(4,joao),(5,maria)] ?
yes
```

Figura 49 - Exemplo 20

Existem 5 Utentes registados no sistema (Ricardo, André, Gil, João e Maria).

Apresentar todos os IDs e Nomes dos Prestadores

Uma vez que é possível obter uma relação de todos os utentes muito rapidamente, foi implementado um método que fizesse o mesmo com os Prestadores.

```
getIdNomePrestadores(Li) :- findall((IdPrest, Nome), prestador(IdPrest, Nome, Especialidade, Instituicao), Li).
```

Figura 50 - Apresenta todos os IDs e nomes dos prestadores

Esta funcionalidade é importante para ser apresentado, de forma rápida e resumida, o número de ID único de cada Prestador, juntamente com o seu Nome.

Exemplo:

```
| ?- getIdNomePrestadores(C).
C = [(1,carlos),(2,manuel),(3,david),(4,paula),(5,luis)] ?
yes
```

Figura 51 - Exemplo 21

Existem 5 Prestadores registados no sistema (Carlos, Manuel, David, Paula, Luís).

Determinar o ID e o Nome dos Utentes a partir da Data do Cuidado

Por fim, a última ferramenta que o grupo decidiu adicionar foi a de determinar o ID e o nome dos Utentes que obtiveram cuidados médicos num dia introduzido.

```
dataGetUtentes(Data, R) :- findall(IdUt, cuidado(Data, IdUt, IdPrest, Descricao, Custo), Li),
                           dataGetUtentesAux(Li, R).

dataGetUtentesAux([], []).
dataGetUtentesAux([H|T], [HS|TS]) :- findall((H, Nome), utente(H, Nome, Idade, Morada), HS),
                                     dataGetUtentesAux(T, TS).
```

Figura 52 - Apresentação do ID e nome dos utentes a partir de uma data

Dada a Data do Cuidado prestado, retira-se todos os IDs dos Utentes que obtiveram serviço nesse dia. Como o ID não é suficiente, decidimos adicionar o nome desse Utente, com a ajuda da função "dataGetUtentesAux".

Exemplo:

```
| ?- dataGetUtentes(16032018, R).
R = [[(1,ricardo)],[(5,maria)],[(2,andre)]] ?
yes
```

Figura 53 - Exemplo 22

No dia 16-03-2018, foram sujeitos a serviços de saúde os Utentes Ricardo, Maria e André.

Conclusões e Sugestões

Em conclusão, este trabalho permitiu aprofundar os nossos conhecimentos em relação à linguagem PROLOG e melhor entender conceitos como invariantes.

Durante a realização deste trabalho prático, não obtivemos dificuldades de maior. No entanto, houve alguns predicados que se tornaram um pouco mais exigentes, o que nos levou a uma pesquisa mais profunda.

De resto, concordamos que terminamos este trabalho com sucesso, estando preparados para as seguintes fases destes exercícios práticos.

Referências

[Leite, 1978] LEITE, J. A. Américo,

"Metodologia de Elaboração de Teses",

McGraw-Hill do Brasil, São Paulo, 1978.

[Analide, 2001] ANALIDE, Cesar, NOVAIS, Paulo, NEVES, José,

"Sugestões para a Elaboração de Relatórios",

Relatório Técnico, Departamento de Informática, Universidade do Minho, Portugal, 2001.