Universidade do Minho
Escola de Engenharia

MESTRADO INTEGRADO EM ENGENHARIA BIOMÉDICA

INFORMÁTICA MÉDICA 2020/2021

Programação em Lógica, Conhecimento e Raciocínio – Parte 2

Alunos:

João Miguel da Silva Alves (83624)

Paulo Jorge Alves (84480)

Docente:

César Analide

18 de dezembro 2020

RESUMO

Neste trabalho, faz-se uma abordagem a um sistema de representação de conhecimento na área da prestação de cuidados de saúde, onde a informação existente não é completa.

Apresenta-se a programação em lógica como ferramenta para a representação de conhecimento e estende-se esta representação para informação incompleta (através da negação por falha e da negação forte). Aborda-se ainda, o problema do pressuposto do mundo fechado na programação em lógica estendida.

Para caracterizar e comprovar que o sistema criado cumpre os requisitos estabelecidos recorreu-se à ferramenta SICStus.

ÍNDICE

1. INTRODUÇÃO 6

2. PRELIMINARES 7

2.1. Pressupostos Lógicos 7

2.2. Extensão à Programação em Lógica 8

2.3. Representação de Conhecimento Imperfeito 8

3. DESCRIÇÃO DO TRABALHO 9

3.1. Declarações Iniciais 9

3.2. Definições Iniciais 9

3.3. Entidades 9

3.4. Sistema de Inferência 10

3.5. Predicados Auxiliares 11

3.5.1. Predicado Soluções 11

3.5.2. Predicado Comprimento 11

3.5.3. Predicado Não 12

3.5.4. Predicado Evolução 12

3.5.5. Predicado Involução 13

3.6. Conhecimento perfeito Positivo e Negativo 13

3.6.1. Conhecimento positivo 13

3.6.1.1. Predicado Utente 13

3.6.1.2. Predicado Serviço 14

3.6.1.3. Predicado Consulta 14

3.6.1.4. Predicado Prestador 14

3.6.2. Conhecimento negativo 14

3.6.2.1. Predicado Utente 15

3.6.2.2. Predicado Serviço 15

3.6.2.3. Predicado Consulta 15

3.6.2.4. Predicado Prestador 15

3.7. Conhecimento Imperfeito 16
3.7.1. Conhecimento incerto 16
3.7.1.1. Predicado Utente 16
3.7.1.2. Predicado Serviço 16
3.7.1.3. Predicado Consulta 17
3.7.1.4. Predicado Prestador 17
3.7.2. Conhecimento impreciso 17
3.7.2.1. Predicado Utente 18
3.7.2.2. Predicado Serviço 18
3.7.2.3. Predicado Consulta 19
3.7.2.4. Predicado Prestador 19
3.7.3. Conhecimento interdito 19
3.7.3.1. Predicado Utente 20
3.7.3.2. Predicado Serviço 20
3.7.3.3. Predicado Consulta 20
3.7.3.4. Predicado Prestador 21
3.8. Invariantes 21
3.8.1. Invariantes associados ao predicado utente 22
3.8.2. Invariantes associados ao predicado serviço 22
3.8.3. Invariantes associados ao predicado consulta 23
3.8.4. Invariantes associados ao predicado prestador 24
3.9. Evolução do Conhecimento 25
3.9.1. Evolução do conhecimento positivo para positivo 25
3.9.2. Evolução do conhecimento negativo para positivo 25
3.9.3. Evolução do conhecimento positivo para negativo 26
3.9.4. Evolução do conhecimento impreciso para incerto 27
3.9.5. Evolução do conhecimento positivo para interdito 28
3.9.6. Evolução do conhecimento interdito para positivo 28

3.9.7. Evolução do conhecimento impreciso para positivo 29
3.9.8. Evolução do conhecimento incerto para positivo.....29
3.9.9. Evolução do conhecimento incerto para impreciso.....30
3.9.10. Evolução do conhecimento positivo para incerto 30
3.9.11. Evolução do conhecimento positivo para impreciso 31
3.9.12. Atualização do conhecimento 31
3.9.12.1. Atualizar utente 31
3.9.12.2. Atualizar serviço 32
3.9.12.3. Atualizar consulta 32
3.9.12.4. Atualizar prestador 32
3.9.13. Inserção de conhecimento 33
3.10. Novo sistema de Inferência 33
4. CONCLUSÃO 36
5. REFERÊNCIAS 37
6. ANEXOS 38
A. Conhecimento perfeito Positivo 38

1. INTRODUÇÃO

Esta segunda parte do trabalho, onde se aplica a extensão à programação em lógica, usando a linguagem de programação PROLOG, tem o objetivo de representar conhecimento imperfeito, recorrendo à utilização de valores nulos e à criação de mecanismos de raciocínio adequados.

De forma a cumprir este objetivo, foi desenvolvido um sistema de representação de conhecimento e raciocínio com capacidade para caracterizar um sistema de bases de dados na área da prestação de cuidados de saúde.

Esta base de conhecimento foi construída de modo a que permitisse responder a alguns pontos essenciais propostos:

- Representar conhecimento positivo e negativo;
- Representar casos de conhecimento imperfeito, pela utilização de valores nulos de todos os tipos estudados;
- Representar invariantes que designem restrições à inserção e à remoção de conhecimento do sistema;
- Lidar com a problemática da evolução do conhecimento, criando os procedimentos adequados;
- Desenvolver um sistema de inferência capaz de implementar os mecanismos de raciocínio inerentes a estes sistemas.

2. PRELIMINARES

Para uma melhor compreensão desta segunda parte do trabalho prático é necessário introduzir novos conceitos, tais como os pressupostos da programação lógica, a programação em lógica estendida e a representação de conhecimento imperfeito.

2.1. Pressupostos Lógicos

As linguagens de manipulação de informação num sistema de BD alicerçam-se nos seguintes pressupostos:

- **Pressuposto Mundo Fechado:** toda a informação que não existe mencionada na base de dados é considerada falsa;
- **Pressuposto dos Nomes Únicos:** duas constantes diferentes designam, necessariamente, duas entidades diferentes do universo de discurso;
- **Pressuposto do Domínio Fechado:** não existem mais objetos no universo de discurso para além daqueles designados por constantes na base de dados.
- **Pressuposto Mundo Aberto:** podem existir outros factos ou conclusões verdadeiras para além daqueles representados na base de conhecimento;
- **Pressuposto do Domínio Aberto:** podem existir mais objetos do universo de discurso para além daqueles designados pelas constantes da base de conhecimento.

Na parte 1 deste trabalho prático, no sentido de desenvolver o sistema de representação de conhecimento, foi abordado o Pressuposto do Mundo Fechado, no entanto este apenas testa a base de conhecimento em dois valores de verdade: verdadeiro ou falso. Isto vai implicar que ocorrem várias limitações em termos da representação de conhecimento uma vez que apenas só vai ser verdadeiro o que estiver representado como tal, sendo falso tudo o que não existir dito como verdadeiro.

Nesta segunda parte, foi abandonado o pressuposto do mundo fechado e adotou-se o pressuposto do mundo aberto, de modo a possibilitar a representação de conhecimento imperfeito. Consequentemente, torna-se possível distinguir três tipos de conclusões para uma questão: esta pode ser verdadeira, falsa ou, quando não existe informação que permita inferir uma ou outra das conclusões anteriores, a resposta à questão será desconhecida.

2.2. Extensão à Programação em Lógica

Um programa em lógica apresenta várias limitações em termos de representação de conhecimento.

O objetivo de estender a programação em lógica é o de que passe a permitir representar informação negativa explicitamente. A extensão de um programa em lógica passa a contar com dois tipos de negação:

- **Negação por falha:** representada pelo predicado não.
- **Negação forte:** identifica informação negativa/falsa, sendo representada pela conectiva lógica '¬ '.

A distinção entre estas duas negações é essencial quando não se pretende assumir que a informação existente sobre P é completa, ou seja, caso exista ausência de informação de P, não indica que P seja necessariamente falso, o que reforça o abandono do Pressuposto do Mundo Fechado.

2.3. Representação de Conhecimento Imperfeito

Em sistemas construídos com base nesta forma de extensão à programação em lógica, passa-se a ter uma terceira hipótese de responder às questões que porventura lhe sejam colocadas: para além de serem verdadeiras as conclusões obtidas a partir da informação positiva e falsas as obtidas em função da informação negativa, torna-se possível concluir que a resposta é desconhecida, se nenhuma das conclusões anteriores for possível.

Os valores nulos surgem em situações de informação incompleta, quando há necessidade de distinguir valores conhecidos e valores desconhecidos. Podem aparecer sob a forma de três tipos diferentes:

- **Incerto:** representação de valores desconhecidos e não necessariamente de um conjunto determinado de valores, permitindo ser descoberto o seu valor posteriormente.
- **Impreciso:** representação de valores desconhecidos que pertencem a um conjunto finito de hipóteses para o valor deste.
- **Interdito:** representação de valores desconhecidos nos quais não é permitida a sua alteração, nunca sendo possível saber o valor destes.

3. DESCRIÇÃO DO TRABALHO

3.1. Declarações Iniciais

De modo a modelar o programa ao contexto é necessário a implementação de flags na base de conhecimento, através de:

```
1.  :- set_prolog_flag( discontiguous_warnings,off ).
2.  :- set_prolog_flag( single_var_warnings,off ).
3.  :- set_prolog_flag( unknown,fail ).
```

3.2. Definições Iniciais

As declarações dynamic são utilizadas para predicados que se pretendem que sejam dinâmicos uma vez que, estes, quando são gerados, tomam o valor de estáticos, o que não permite ao programa alterar esses mesmos predicados. Desta forma, a implementação destas definições iniciais surge para permitir que haja inserção ou remoção de conhecimento dos predicados utente, prestador, consulta e serviço. É também aqui que são declarados operadores, sendo neste caso declarado o operador relacionado com os invariantes “::" e os operadores "e" e "ou", relacionados com o sistema de inferência que desenvolvemos, evitando assim erros ao longo do programa. Outras definições foram adicionadas para assegurar o correto funcionamento dos predicados existentes.

```
1.  :- dynamic utente/8.
2.  :- dynamic prestador/4.
3.  :- dynamic consulta/5.
4.  :- dynamic servico/5.
5.  :- op(900,xfy,'::').
6.  :- op(300,xfy,ou).
7.  :- op(300,xfy,e).
8.  :- dynamic '-'/1.
9.  :- dynamic excecao/1.
10. :- dynamic interdito/1.
11. :- dynamic (::)/2.
```

3.3. Entidades

A primeira parte do trabalho prático consistiu na construção da base de conhecimento, sendo que esta é resultado da declaração de utentes, serviços, consultas e prestadores.

Nesta segunda parte, usando agora uma programação em lógica estendida, serão usadas as entidades descritas infra.

```
1. %predicado utente: IdUt, Nome, Idade, Cidade, Pai, Mãe, NIF,doença ->{V,F,D}
2. %predicado servico: Idserv, IdPrest,Descrição, Instituição, Cidade ->{V,F,D}
3. %predicado consulta: IdCons, Data, IdUt, IdServ, Custo ->{V,F,D}
4. %predicado prestador: IdPrest, Nome, Especialidade, Instituição ->{V,F,D}
```

3.4. Sistema de Inferência

Uma vez que, nesta segunda parte do trabalho prático, o contradomínio dos predicados representados em 3.3. apresenta três possíveis valores de verdade: {V, F, D}, surgiu a necessidade de construir um meta-predicado, em Prolog, capaz de os resolver.

Para tal, criou-se o Sistema de Inferência, si, que foi implementado da seguinte forma:

```
1. % Extensao do meta-predicado si: Questao,Resposta ->{V,F}
2.
3. si( Questao,verdadeiro ) :-
4.     Questao.
5. si( Questao,falso ) :-
6.     -Questao.
7. si( Questao,desconhecido ) :-
8.     nao( Questao ),
9.     nao( -Questao ).
```

Quando se coloca neste meta-predicado uma única questão, existem três possíveis respostas:

- **Verdadeiro**: Quando estão presentes factos que afirmem que a questão é verdadeira.
- **Falso**: Quando for possível provar uma negação da questão (¬Questão é Verdadadeiro)
- **Desconhecido**: Quando não existem evidencias positivas nem negativas da questão colocada.

Alguns exemplos de aplicação deste si estão representados abaixo.

```
| ?- si(utente(101,josepereira,23,porto,henriquepereira,anapereira,456789123,bronquite),R).
R = verdadeiro ?
yes
```

Figura 1 - Utilização do meta-predicado si num utente que existe na base de dados

```
| ?- si(utente(111,isabelfreitas,22,porto,gabrielfreitas,anafreitas,888811345,asma),R).
R = desconhecido ?
yes
```

Figura 2 - Utilização do meta-predicado si num utente em que não se sabe a cidade.

```
| ?- si(utente(115,andrelopes,50,vilanovadegaia,carloslopes,veralopes,331115654,asma),R).
R = falso ?
yes
```

Figura 3 - Utilização do meta-predicado si num utente que não existe na base de dados

No final, será construído um novo sistema de inferência capaz de responder a uma composição de questões.

3.5. Predicados Auxiliares

3.5.1. Predicado Soluções

Este predicado é utilizado quando se pretende listar todas as soluções como resposta a uma dada questão.

```
1.  %Extensao do predicado solucoes: X,Y,Z --> {V,F}
2.
3.  solucoes(X,Y,Z):-findall(X,Y,Z).
```

O predicado acima contém 3 argumentos: X, que corresponde ao elemento que se pretende obter; Y que é o teorema que se pretende provar quando é feita uma certa pergunta; L que é a lista que contém o conjunto de soluções.

A ferramenta "*findall*" já se encontra no SICStus.

3.5.2. Predicado Comprimento

Este predicado recebe como argumento uma lista e determina o seu comprimento.

```
1.  %Extensao do predicado comprimento: Lista, Resultado --> {V,F}
2.
3.  comprimento([],0).
4.  comprimento([X|L],R):- comprimento(L,N), R is N+1.
```

O comprimento de uma lista com apenas um elemento é 1. Pela recursividade, tendo a lista, X como cabeça e L como cauda, e se R1 é o comprimento de L, então a lista tem comprimento R1+1.

3.5.3. Predicado Não

Este predicado tem como objetivo retornar o valor falso caso o valor de verdade da questão seja verdadeiro ou retornar o valor verdadeiro caso o valor de verdade da questão seja falso.

```
1.  % Extensao do meta-predicado nao: Questao -> {V,F}
2.
3.  nao( Questao ) :-
4.      Questao, !, fail.
5.  nao( Questao ).
```

O predicado acima descrito recebe como argumento uma questão. Se o valor da questão for verdadeiro, então através da implementação do cut e fail que significa insucesso, este não procurará a segunda cláusula e irá retornar falso. Caso o valor da questão seja falso, então passará para a próxima cláusula e, por isso, irá retornar verdadeiro.

3.5.4. Predicado Evolução

A extensão deste predicado recebe como atributo um termo. De seguida com o predicado soluções, iremos obter uma lista com todos os invariantes correspondentes a esse termo. Depois através da primeira cláusula do predicado inserir, irá inserir o termo na nossa base de conhecimento. Com o predicado teste, iremos testar a consistência da nossa base de conhecimento. Este irá testar todos os invariantes associados a esse termo. Se tudo estiver correto, então o termo é inserido. Caso contrário, irá se passar para a segunda cláusula do predicado inserir em que se irá remover esse termo antes inserido e não voltará atrás para testar, uma vez que se encontra o cut e o fail que significa insucesso.

```
1.  %extensão do predicado que permite a evolução do conhecimento: Termo ->{V,F}
2.
3.  evolucao(Termo) :- solucoes(Inv, +Termo::Inv, Linv), inserir(Termo),
    teste(Linv).
4.
5.  %extensão do predicado que permite a inserção do termo: Termo ->{V,F}
6.
7.  inserir(Termo) :- assert(Termo).
8.  inserir(Termo) :- retract(Termo), !, fail.
9.
10.
11. %extensão do predicado que permite testar os invariantes de uma lista Termo -> {V,F}
12.
13. teste([]).
14. teste([I|R]) :- I,teste(R).
```

3.5.5. Predicado Involução

A extensão do predicado involução recebe como atributo um termo tanto do predicado utente, prestador, serviço ou consulta. De seguida, através do predicado soluções, iremos obter uma lista com todos os invariantes correspondentes a esse termo. Depois através do predicado testar, irá testar se vai ser possível remover o nosso termo da base de conhecimento. Este irá testar todos os invariantes de remoção do respetivo termo. Caso derem todos corretos, então passará para o predicado remover que irá remover o termo passado como argumento. Caso contrário, o predicado remover não ocorrerá e o termo não será removido.

```
1.  %extensão do predicado que permite a involução do conhecimento: Termo ->{V,F}
2.
3.  involucao(Termo) :- solucoes(Inv, -Termo::Inv, Linv), testar(Linv),
    remover(Termo).
4.
5.  %extensão do predicado que permite a remoção do termo: Termo ->{V,F}
6.
7.  remover(Termo) :- retract(Termo).
8.  remover(Termo) :- assert(Termo), !, fail.
9.
10. %extensão do predicado que permite testar os invariantes de uma lista: Termo ->{V,F}
11.
12. testar([]).
13. testar([I|R]) :- I,testar(R).
```

3.6. Conhecimento perfeito Positivo e Negativo

Para a representação do conhecimento recorremos a factos que pretendem representar conhecimento perfeito, negativo e desconhecido, sendo este discutido posteriormente. O conhecimento perfeito positivo e negativo são implementados com recurso às entidades anteriormente mencionadas.

3.6.1. Conhecimento positivo

3.6.1.1. Predicado Utente

```
utente(100, margaridaalves, 19, braga, manuelalves,joanaalves, 123456789, asma).
```

O predicado acima é um exemplo do predicado “utente” usado na base de conhecimento construída. Tem como atributos o id utente, o nome, a idade, localidade, nome do pai, nome da mãe, NIF e por fim a doença. Todos os restantes predicados “utente” usados, encontram-se em anexo.

3.6.1.2. Predicado Serviço

```
servico(200, 1005, oscultar, hospitaldatrofa, trofa).
```

O predicado acima é um exemplo do predicado "servico" usado na base de conhecimento construída. Tem como atributos o id do serviço, o id do prestador que o realiza, a descrição, a instituição onde o serviço é prestado e por fim a respetiva cidade. Todos os predicados "servico" usados na base de conhecimento encontram-se em anexo.

3.6.1.3. Predicado Consulta

```
consulta(1, 06112020, 100, 200, 80).
```

O predicado acima é um exemplo do predicado "consulta" usado na base de conhecimento construída. Tem como atributos o id da consulta, a data de realização, o id do utente que realiza a consulta, o id do serviço que é prestado nessa consulta e por último o custo dessa mesma consulta. Todos os predicados "consulta" usados na base de conhecimento encontram-se em anexo.

3.6.1.4. Predicado Prestador

```
prestador(1000, joaovasquez, medicinageral, hospitalpublicodebraga).
```

O predicado acima é um exemplo do predicado "prestador" usado na base de conhecimento construída. Tem como atributos o id prestador, o nome, a especialidade e por último a instituição onde exerce. Todos os predicados "prestador" usados na base de conhecimento encontram-se em anexo.

3.6.2. Conhecimento negativo

Para conhecimento perfeito negativo usamos a denotação -utente(), que irá incluir como conhecimento negativo aquele que não se encontra como positivo, ou seja negação por falha na prova e aquele que não se encontra no predicado exceção que também será mencionada a seguir, no conhecimento imperfeito.

Além disso, o conhecimento negativo é também representado pelos factos -utente(A,B,C,D,E,F,G,H), -servico(I,J,K,L,M), -consulta(N,O,A,I,P) e -prestador(Q,R,S,T).

3.6.2.1. Predicado Utente

```
%predicado utente: IdUt, Nome, Idade, Cidade, Pai, Mãe, NIF,doenca  -> {V,F,D}

-utente(A,B,C,D,E,F,G,H):-
nao(utente(A,B,C,D,E,F,G,H)),
nao(excecao(utente(A,B,C,D,E,F,G,H))).

-utente(114,eduardoramalho,24,braga,manuelramalho,sararamalho,222123455,covid).
-utente(115,andrelopes,50,vilanovadegaia,carloslopes,veralopes,331115654,asma).
-utente(116,pedropastor,45,famalicao,henriquepastor,dianapastor,756832122,febre).
-utente(117,tiagomatos,20,braga,josematos,luisamatos,777123123,covid).
```

3.6.2.2. Predicado Serviço

```
-servico(I,J,K,L,M):-
nao(servico(I,J,K,L,M)),
nao(excecao(servico(I,J,K,L,M))).

-servico(220,1008,oscultar,hospitalprivadodebraga,braga).
-servico(221,1002,medirtensao,hospitalpublicodebraga,braga).
-servico(222,1001,oscultar,hospitalpublicodebraga,braga).
```

3.6.2.3. Predicado Consulta

```
-consulta(N,O,A,I,P):-
nao(consulta(N,O,A,I,P)),
nao(excecao(consulta(N,O,A,I,P))).

-consulta(20,12082020,100,201,30).
-consulta(21,20112020,101,202,80).
-consulta(22,22112020,102,203,20).
```

3.6.2.4. Predicado Prestador

```
-prestador(Q,R,S,T):-
nao(prestador(Q,R,S,T)),
nao(excecao(prestador(Q,R,S,T))).

-prestador(1020,fabiomarques,pneumologista,hospitalpublicodebraga).
-prestador(1021,marciahenriqueta,medicinageral,hospitalpublicodebraga).
-prestador(1022,eduardapauleta,pediatra,hospitalsantaluzia).
-prestador(1023,otaviosoares,endocrinologia,hospitaldatrofa).
```

3.7. Conhecimento Imperfeito

3.7.1. Conhecimento incerto

Para o conhecimento imperfeito do tipo incerto, o resultado que devemos obter ao colocar termos deste tipo, será desconhecido e para tal é necessário implementar o predicado exceção para cada um dos predicados já existentes.

3.7.1.1. Predicado Utente

Neste caso, foi criado conhecimento incerto para o utente Isabel Freitas em que se desconhece a sua cidade e para a utente Nuna Mário que se desconhece a sua idade.

```
1.  utente(111,isabelfreitas,22,cidade,gabrielfreitas,anafreitas,888811345,asma).
2.  excecao(utente(A,B,C,D,E,F,G,H)):- utente(A,B,C,cidade,E,F,G,H).
3.
4.  utente(112,nunamario,idade,porto,diogomario,isabelmario,543524153,covid).
5.  excecao(utente(A,B,C,D,E,F,G,H)):- utente(A,B,idade,D,E,F,G,H).
```

```
| ?- si(utente(111,isabelfreitas,22,porto,gabrielfreitas,anafreitas,888811345,asma),R).
R = desconhecido ?
yes
| ?- si(utente(112,nunamario,20,porto,diogomario,isabelmario,543524153,covid),R).
R = desconhecido ?
yes
```

Figura 4 – Exemplos testados com o meta-predicado “si”

3.7.1.2. Predicado Serviço

Neste caso, tanto no serviço com o id 209 como no serviço com o id 210, desconhece-se o prestador que o realiza.

```
1.  servico(209,prestador,oscultar,hospitaldatrofa,trofa).
2.  excecao(servico(I,J,K,L,M)):- servico(I,prestador,K,L,M).
3.
4.  servico(210,prestador1,oscultar,hospitalprivadodebraga,braga).
5.  excecao(servico(I,J,K,L,M)):- servico(I,prestador1,K,L,M).
```

```
| ?- si(servico(209,mario,oscultar,hospitaldatrofa,trofa),R).
R = desconhecido ?
yes
```

Figura 5 – Exemplo testado com o meta-predicado “si”

3.7.1.3. Predicado Consulta

Para o predicado consulta, desconhece-se a data para a consulta com o id 14 e desconhece-se o custo para a consulta com o id 15.

```
1.  consulta(14,data,107,206,45).
2.  excecao(consulta(N,O,A,I,P)):- consulta(N,data,A,I,P).
3.
4.  consulta(15,13092021,110,206,custo).
5.  excecao(consulta(N,O,A,I,P)):- consulta(N,O,A,I,custo).
```

```
| ?- si(consulta(14,13092021,107,206,45),R).
R = desconhecido ?
yes
| ?- si(consulta(15,13092021,110,206,70),R).
R = desconhecido ?
yes
```

Figura 6 – Exemplos testados com o meta-predicado "si"

3.7.1.4. Predicado Prestador

Neste caso, desconhece-se a especialidade dos prestadores de id 1011 e id 1012.

```
1.  prestador(1011,fabiotigre,especialidade,hospitalpublicodebraga).
2.  excecao(prestador(Q,R,S,L)):- prestador(Q,R,especialidade,L).
3.
4.  prestador(1012,franciscolopes,especialidade1,hospitalsantaluzia).
5.  excecao(prestador(Q,R,S,L)):- prestador(Q,R,especialidade1,L).
```

```
| ?- si(prestador(1011,fabiotigre,dermatologista,hospitalpublicodebraga),R).
R = desconhecido ?
yes
```

Figura 7 – Exemplo testado com o meta-predicado "si"

3.7.2. Conhecimento impreciso

Para o conhecimento imperfeito do tipo impreciso, o resultado que devemos obter ao colocar termos deste tipo, será também desconhecido e para tal é necessário implementar o predicado exceção para cada um dos predicados já existentes.

3.7.2.1. Predicado Utente

Para o predicado utente, não se sabe se a utente Cristina Ferreira possui covid ou asma. Além disso, desconhece-se a idade do utente Rodrigo Lima, mas sabe-se que está entre 20 e 30.

```
1. excecao(utente(120,cristinaferreira,25, barcelos,fabioferreira,joanaferreira,2
   22255556,asma)).
2. excecao(utente(120,cristinaferreira,25, barcelos,fabioferreira,joanaferreira,2
   22255556,covid)).
3.
4. excecao(utente(121,rodrigolima,PO,porto,joaolima,analima,535454545,covid)) :-
   PO>=20, PO =< 30.
```

```
| ?- si(utente(120,cristinaferreira,25, barcelos,fabioferreira,joanaferreira,222255556,asma),R).
R = desconhecido ?
yes
| ?- si(utente(120,cristinaferreira,25, barcelos,fabioferreira,joanaferreira,222255556,covid),R).
R = desconhecido ?
yes
| ?- si(utente(120,cristinaferreira,25, barcelos,fabioferreira,joanaferreira,222255556,febre),R).
R = falso ?
yes
| ?- si(utente(121,rodrigolima,25,porto,joaolima,analima,535454545,covid),R).
R = desconhecido ?
yes
| ?- si(utente(121,rodrigolima,32,porto,joaolima,analima,535454545,covid),R).
R = falso ?
yes
```

Figura 8 – Exemplos testados com o meta-predicado “si”

3.7.2.2. Predicado Serviço

Para o predicado serviço, desconhece-se se é o prestador Diogo Amaral ou o prestador Tiago Borges que exerce o serviço com o id 230.

```
1. excecao(servico(230,diogoamaral,medirtensao,hospitaldatrofa,trofa)).
2. excecao(servico(230,tiagoborges,medirtensao,hospitaldatrofa,trofa)).
```

```
| ?- si(servico(230,diogoamaral,medirtensao,hospitaldatrofa,trofa),R).
R = desconhecido ?
yes
| ?- si(servico(230,tiagoborges,medirtensao,hospitaldatrofa,trofa),R).
R = desconhecido ?
yes
| ?- si(servico(230,zealves,medirtensao,hospitaldatrofa,trofa),R).
R = falso ?
yes
```

Figura 9 – Exemplos testados com o meta-predicado “si”

3.7.2.3. Predicado Consulta

No caso do predicado consulta, desconhece-se que o custo da consulta de id 25 se encontra entre 70 e 80.

```
1. excecao(consulta(25,15062021,106,205,CO)):- CO>=70, CO=<80.
```

```
| ?- si(consulta(25,15062021,106,205,70),R).
R = desconhecido ?
yes
| ?- si(consulta(25,15062021,106,205,81),R).
R = falso ?
yes
```

Figura 10 – Exemplos testados com o meta-predicado "si"

3.7.2.4. Predicado Prestador

Neste caso, desconhece-se se o prestador Nuno Lopes com o id 1030 apresenta a especialidade de pneumologia ou pediatria.

```
1. excecao(prestador(1030,nunolopes,pneumologista,hospitalpublicodebraga)).
2. excecao(prestador(1030,nunolopes,pediatra,hospitalpublicodebraga)).
```

```
| ?- si(prestador(1030,nunolopes,pediatra,hospitalpublicodebraga),R).
R = desconhecido ?
yes
```

Figura 11 – Exemplo testado com o meta-predicado "si"

3.7.3. Conhecimento interdito

Para o conhecimento imperfeito do tipo interdito, o resultado que devemos obter ao colocar termos deste tipo, será também desconhecido e para tal é necessário implementar o predicado exceção para cada um dos predicados já existentes. Além disso, não poderemos permitir inserção de tais valores, uma vez que o conhecimento é interdito e, por isso, não poderá ser conhecido, sendo usado um invariante para garantir esta última ocorrência.

3.7.3.1. Predicado Utente

No caso do predicado utente, desconhece-se a cidade do utente com o id 130 e não é permitido a sua inserção na base de conhecimento.

```
1. utente(130,joseramalho,48,cidade1,inacioramalho,albertinaramalho,345343431,covid).
2. excecao(utente(A,B,C,D,E,F,G,H)):- utente(A,B,C,cidade1,E,F,G,H).
3. interdito(cidade1).
4. +utente(A,B,C,D,E,F,G,H)::(solucoes(D,(utente(130,joseramalho,48,D,inacioramalho,
   albertinaramalho,345343431,covid),nao(interdito(D))),T), comprimento(T,Z),Z==0).
```

```
| ?- si(utente(130,joseramalho,48,braga,inacioramalho,albertinaramalho,345343431,covid),R).
R = desconhecido ?
yes
| ?- evolucao(utente(130,joseramalho,48,braga,inacioramalho,albertinaramalho,345343431,covid)).
no
```

Figura 12 – Exemplo testado com o meta-predicado "si"

3.7.3.2. Predicado Serviço

No caso do predicado serviço, desconhece-se a instituição o qual o serviço de id 240 é realizado e não é permitido a sua inserção na base de conhecimento.

```
1. servico(240,eduardovieira,analisessecrecoes,instituicao,braga).
2. excecao(servico(I,J,K,L,M)):- servico(I,J,K,instituicao,M).
3. interdito(instituicao).
4. +servico(I,J,K,L,M)::(solucoes(L,(servico(240,eduardovieira,analisessecrecoes,L,braga),
   nao(interdito(L))),T),comprimento(T,Z),Z==0).
```

```
| ?- si(servico(240,eduardovieira,analisessecrecoes,hospitaldaluz,braga),R).
R = desconhecido ?
yes
| ?- evolucao(servico(240,eduardovieira,analisessecrecoes,hospitaldaluz,braga)).
no
```

Figura 13 – Exemplo testado com o meta-predicado "si"

3.7.3.3. Predicado Consulta

No caso do predicado consulta, desconhece-se o custo para a consulta com id 40 e não se permite a sua inserção na base de conhecimento.

```
1. consulta(40,31082021,105,201,custo3).
2. excecao(consulta(N,O,A,I,P)):-consulta(N,O,A,I,P,custo3).
3. interdito(custo3).
4. +consulta(N,O,A,I,P)::(solucoes(P,(consulta(40,31082021,105,201,P),
   nao(interdito(P))),T), comprimento(T,Z),Z==0).
```

```
| ?- si(consulta(40,31082021,105,201,40),R).
R = desconhecido ?
yes
| ?- evolucao(consulta(40,31082021,105,201,40)).
no
```

Figura 14 – Exemplo testado com o meta-predicado "si"

3.7.3.4. Predicado Prestador

No caso do predicado prestador, desconhece-se a especialidade exercida pelo prestador com o id 1040 e não se permite a sua inserção na base de conhecimento.

```
1. prestador(1040,gabrielmario,especialidade3,hospitalpublicodebraga).
2. excecao(prestador(Q,R,S,L)):-prestador(Q,R,especialidade3,L).
3. interdito(especialidade3).
4. +prestador(Q,R,S,L)::(solucoes(S,(prestador(1040,gabrielmario,S,
   hospitalpublicodebraga), nao(interdito(S))),T), comprimento(T,Z),Z==0).
```

```
| ?- si(prestador(1040,gabrielmario,pediatra,hospitalpublicodebraga),R).
R = desconhecido ?
yes
| ?- evolucao(prestador(1040,gabrielmario,pediatra,hospitalpublicodebraga)).
no
```

Figura 15 – Exemplo testado com o meta-predicado "si"

3.8. Invariantes

Tal como foi explicado na parte 1 deste trabalho prático, a manipulação da base de dados é possível recorrendo ao uso de invariantes. A utilização destes invariantes garante uma integridade estrutural e referencial da base de conhecimento. Estes impedem a inserção de conhecimento repetido (invariantes estruturais) e asseguram a consistência da informação existente na base de conhecimento (invariantes referenciais).

Visto que já se explicou o código usado em cada invariante na parte 1, aqui apenas serão enunciados os invariantes usados. De notar que, em relação à primeira parte, foram introduzidos invariantes referentes ao conhecimento negativo e exceções.

3.8.1. Invariantes associados ao predicado utente

```
%Invariante de Inserção Estrutural: não permite a inserção de um utente se este já existir na base de conhecimento
+utente(A,B,C,D,E,F,G,H) :: (solucoes((A,B,C,D,E,F,G,H),utente(A,B,C,D,E,F,G,H),S), comprimento(S,N), N==1).

%Invariante de Inserção Referencial: não permite a inserção de um utente com o mesmo id
+utente(A,_,_,_,_,_,_,_) :: (solucoes(A,utente(A,_,_,_,_,_,_,_),S), comprimento(S,N), N==1).

%Invariante de Inserção Referencial: não permite a inserção de um utente com o mesmo nif
+utente(_,_,_,_,_,_,N,_) :: (solucoes(N,utente(_,_,_,_,_,_,N,_),S), comprimento(S,R), R==1).

%invariante de Remoção Estrutural: permite a remoção de um utente se este existir na base de conhecimento
-utente(A,B,C,D,E,F,G,H) :: (solucoes((A,B,C,D,E,F,G,H),(utente(A,B,C,D,E,F,G,H)),S), comprimento(S,N), N == 1).

%Invariante de Remoção Referencial: permite a remoção de um utente se não estiver associado a nenhuma consulta
-utente(A,B,C,D,E,F,G,H) :: (solucoes(A,consulta(T,Y,A,V,Z,J),S), comprimento(S,N), N==0).

%invariante de Inserção Estrutural: não permite a inserção de conhecimento perfeito negativo que já se encontre na base de dados
+(-utente(A,B,C,D,E,F,G,H)) :: (solucoes((A,B,C,D,E,F,G,H), -utente(A,B,C,D,E,F,G,H),S), comprimento(S,N), N==1).

%Invariante de Inserção Referencial: não permite a inserção de conhecimento perfeito negativo de um utente com o mesmo id
+(-utente(A,_,_,_,_,_,_,_)) :: (solucoes(A,-utente(A,_,_,_,_,_,_,_),S), comprimento(S,N), N==1).

%Invariante de Inserção Referencial: não permite a inserção de conhecimento perfeito negativo de um utente com o mesmo nif
+(-utente(_,_,_,_,_,_,N,_)) :: (solucoes(N,-utente(_,_,_,_,_,_,N,_),S), comprimento(S,R), R==1).

%invariante de Inserção Estrutural: não permite a inserção de uma exceção de utente se esta já existir na base de conhecimento
+(excecao(utente(A,B,C,D,E,F,G,H))) :: (solucoes((A,B,C,D,E,F,G,H),excecao(utente(A,B,C,D,E,F,G,H)),S), comprimento(S,N), N==1).

%invariante de Remoção Estrutural: permite a remoção de conhecimento perfeito negativo de um utente se este existir na base de conhecimento
-(-utente(A,B,C,D,E,F,G,H)) :: (solucoes((A,B,C,D,E,F,G,H),-utente(A,B,C,D,E,F,G,H),S), comprimento(S,N), N == 1).

%invariante de Remoção Estrutural: permite a remoção de excecao de um utente se esta existir na base de conhecimento
-excecao(utente(A,B,C,D,E,F,G,H)) :: (solucoes((A,B,C,D,E,F,G,H), excecao(utente(A,B,C,D,E,F,G,H)),S), comprimento(S,N), N == 1).
```

3.8.2. Invariantes associados ao predicado serviço

```
%invariante de Inserção Estrutural: não permite a inserção de um prestador se este já existir na base de conhecimento
+prestador(A,B,C,D) :: (solucoes((A,B,C,D),prestador(A,B,C,D),S), comprimento(S,N), N==1).
```

```
3.
4.  %Invariante de Inserção Referencial: não permite a inserção de um prestador com mesmo id
5.  +prestador(A,_,_,_) :: (solucoes(A,prestador(A,_,_,_),S), comprimento(S,N), N==1).
6.
7.  %invariante de Remoção Estrutural: permite a remoção de um prestador se este existir na
    base de conhecimento
8.  -prestador(A,B,C,D) :: (solucoes((A,B,C,D),prestador(A,B,C,D),S),
    comprimento(S,N), N==1).
9.
10. %Invariante de Remoção Referencial: permite a remoção de um prestador se não estiver
    associado a nenhum serviço
11. -prestador(A,B,C,D) :: (solucoes(A,servico(_,A,_,_,_),S), comprimento(S,N), N==0).
12.
13. %invariante de Inserção Estrutural: não permite a inserção de conhecimento perfeito
    negativo se este já existir na base de conhecimento
14. +(-prestador(A,B,C,D)) :: (solucoes((A,B,C,D),-prestador(A,B,C,D),S),
    comprimento(S,N), N==1).
15.
16. %Invariante de Inserção Referencial: não permite a inserção de conhecimento perfeito
    negativo de um prestador com o mesmo id
17. +(-prestador(A,_,_,_)) :: (solucoes(A,-prestador(A,_,_,_),S), comprimento(S,N), N==1).
18.
19. %invariante de Inserção Estrutural: não permite a inserção de uma excecao de prestador
    se esta já existir na base de conhecimento
20. +excecao(prestador(A,B,C,D)) :: (solucoes((A,B,C,D),excecao(prestador(A,B,C,D)),S),
    comprimento(S,N), N==1).
21.
22. %invariante de Remoção Estrutural: permite a remoção de conhecimento perfeito negativo
    um prestador se este existir na base de conhecimento
23. -(-prestador(A,B,C,D)) :: (solucoes((A,B,C,D),-prestador(A,B,C,D),S),
    comprimento(S,N), N==1).
24.
25. %invariante de Remoção Estrutural: permite a remoção de uma excecao de um prestador se
    esta existir na base de conhecimento
26. -excecao(prestador(A,B,C,D)) :: (solucoes((A,B,C,D),excecao(prestador(A,B,C,D)),S),
    comprimento(S,N), N==1).
```

3.8.3. Invariantes associados ao predicado consulta

```
1.  %invariante de Inserção Estrutural: não permite a inserção de um servico se este já
    existir na base de conhecimento
2.  +servico(A,B,C,D,E) :: (solucoes((A,B,C,D,E),servico(A,B,C,D,E),S), comprimento(S,N),
    N==1).
3.
4.  %Invariante de Inserção Referencial: não permite a inserção de um servico com o mesmo id
5.  +servico(A,_,_,_,_) :: (solucoes(A,servico(A,_,_,_,_),S), comprimento(S,N), N==1).
6.
7.  %invariante de Remoção Estrutural: permite a remoção de um serviço se este existir na
    base de conhecimento
8.  -servico(A,B,C,D,E) :: (solucoes((A,B,C,D,E),servico(A,B,C,D,E),S),
    comprimento(S,N), N==1).
9.
10. %invariante de Inserção Estrutural: não permite a inserção de conhecimento perfeito
    negativo de um servico se este já existir na base de conhecimento
11. +(-servico(A,B,C,D,E)) :: (solucoes((A,B,C,D,E),-servico(A,B,C,D,E),S),
    comprimento(S,N), N==1).
12.
13. %Invariante de Inserção Referencial: não permite a inserção de um servico com o mesmo id

14. +(-servico(A,_,_,_,_)) :: (solucoes(A,-servico(A,_,_,_,_),S), comprimento(S,N), N==1).
15.
```

```
%invariante de Inserção Estrutural: não permite a inserção de conhecimento perfeito negativo de um servico se este já existir na base de conhecimento
+excecao(servico(A,B,C,D,E)) :: (solucoes((A,B,C,D,E),excecao(servico(A,B,C,D,E)),S), comprimento(S,N), N==1).

%invariante de Remoção Estrutural: permite a remoção de conhecimento perfeito negativo um serviço se este existir na base de conhecimento
-(-servico(A,B,C,D,E)) :: (solucoes((A,B,C,D,E),-servico(A,B,C,D,E),S), comprimento(S,N), N==1).

%invariante de Remoção Estrutural: permite a remoção de uma excecao de um serviço se este existir na base de conhecimento
-excecao(servico(A,B,C,D,E)) :: (solucoes((A,B,C,D,E),excecao(servico(A,B,C,D,E)),S), comprimento(S,N), N==1).
```

3.8.4. Invariantes associados ao predicado prestador

```
%invariante de Inserção Estrutural: não permite a inserção de uma consulta se esta já existir na base de conhecimento
+consulta(A,B,C,D,E) :: (solucoes((A,B,C,D,E),consulta(A,B,C,D,E),S), comprimento(S,N), N==1).

%Invariante de Inserção Referencial: não permite a inserção de uma consulta com o mesmo id
+consulta(A,_,_,_,_) :: (solucoes(A,consulta(A,_,_,_,_),S), comprimento(S,N), N==1).

%invariante de Remoção Estrutural: permite a remoção de uma consulta se esta existir na base de conhecimento
-consulta(A,B,C,D,E) :: (solucoes((A,B,C,D,E),consulta(A,B,C,D,E),S), comprimento(S,N), N==1).

%invariante de Inserção Estrutural: não permite a inserção de conhecimento perfeito negativo uma consulta se esta já existir na base de conhecimento
+(-consulta(A,B,C,D,E)) :: (solucoes((A,B,C,D,E),-consulta(A,B,C,D,E),S), comprimento(S,N), N==1).

%Invariante de Inserção Referencial: não permite a inserção de conhecimento perfeito negativo de uma consulta com o mesmo id
+(-consulta(A,_,_,_,_)) :: (solucoes(A,-consulta(A,_,_,_,_),S), comprimento(S,N), N==1).

%invariante de Inserção Estrutural: não permite a inserção de uma consulta se esta já existir na base de conhecimento
+excecao(consulta(A,B,C,D,E)) :: (solucoes((A,B,C,D,E),excecao(consulta(A,B,C,D,E)),S), comprimento(S,N), N==1).

%invariante de Remoção Estrutural: permite a remoção de conhecimento perfeito negativo de uma consulta se esta existir na base de conhecimento
-(-consulta(A,B,C,D,E)) :: (solucoes((A,B,C,D,E),-consulta(A,B,C,D,E),S), comprimento(S,N), N==1).

%invariante de Remoção Estrutural: permite a remoção de uma excecao de uma consulta se esta existir na base de conhecimento
-excecao(consulta(A,B,C,D,E)) :: (solucoes((A,B,C,D,E),excecao(consulta(A,B,C,D,E)),S), comprimento(S,N), N==1).
```

3.9. Evolução do Conhecimento

Uma vez que no presente trabalho prático se considera a representação de conhecimento perfeito e imperfeito, é necessário ter várias considerações aquando da evolução do conhecimento, sendo estas apresentadas de seguida.

3.9.1. Evolução do conhecimento positivo para positivo

A evolução de conhecimento positivo para positivo, permite que um utente, serviço, consulta ou prestador de um determinado id seja agora substituído por outro com diferentes atributos, contudo mantendo o id anterior. Para tal, é necessário que o facto passado tenho valor de verdade verdadeiro e, assim com recurso ao método involução, que irá verificar os invariantes relacionados com essa remoção, para remover esse facto e com recurso ao predicado evolução, que irá verificar os invariantes relacionados a essa inserção, para inserir o novo facto.

```
1. %Extensao do predicado que permite a evolucao do conhecimento positivo para
   outro positivo: Qnovo,Qpassado --> {V,F}
2.
3. evolucao_PP(Qnovo, Qpassado):-
4. si(Qpassado, verdadeiro),
5. involucao(Qpassado),
6. evolucao(Qnovo).
```

```
| ?- evolucao_PP(utente(100, mickaelferreira, 23, braga, zeferreira, anaferreira, 989898970, asma),
utente(100, margaridaalves, 19, braga, manuelalves, joanaalves, 123456789, asma)).
yes
| ?- listing(utente).
          utente(101, josepereira, 23, porto, henriquepereira, anapereira, 456789123, bronquite).
utente(102, luisdelgado, 33, famalicao, luisdelgado, dianadelgado, 345678912, covid).
utente(103, rodrigomendes, 25, barcelos, josemendes, luisamendes, 789123456, covid).
utente(104, claudiasoares, 20, famalicao, andresoares, joanasoares, 891234567, febre).
utente(105, ruigoncalves, 41, braga, sergiogoncalves, inesgoncalves, 912345678, diarreia).
utente(106, leonorgarcia, 9, guimaraes, diogogarcia, saragarcia, 657891234, covid).
utente(107, mariamagalhaes, 29, porto, ricardomagalhaes, soniamagalhaes, 567891234, bronquite).
utente(108, manuelsilva, 23, braga, joaosilva, teresasilva, 213456789, pneumonia).
utente(109, fernandocerqueira, 12, braga, nunocerqueira, joanacerqueira, 765891234, covid).
utente(110, ismaelbarbosa, 54, barcelos, jorgebarbosa, patriciabarbosa, 987654321, asma).
utente(111, isabelfreitas, 22, cidade, gabrielfreitas, anafreitas, 888811345, asma).
utente(112, nunamario, idade, porto, diogomario, isabelmario, 543524153, covid).
utente(130, joseramalho, 48, cidade1, inacioramalho, albertinaramalho, 345343431, covid).
utente(100, mickaelferreira, 23, braga, zeferreira, anaferreira, 989898970, asma).
```

Figura 16 – Exemplo de evolução do conhecimento positivo para positivo

3.9.2. Evolução do conhecimento negativo para positivo

A evolução de conhecimento negativo para positivo, permite que um utente, serviço, consulta ou prestador de um determinado id que pertence ao conhecimento perfeito negativo seja agora substituído pelo mesmo, mas agora para conhecimento perfeito positivo. Para tal, é

necessário que o facto passado tenho valor de verdade verdadeiro e, assim com recurso ao método involução, que irá verificar os invariantes relacionados com essa remoção, para remover esse facto e com recurso ao predicado evolução, que irá verificar os invariantes relacionados a essa inserção, para inserir o novo facto.

```
1.  % Extensao do predicado que permite a evolucao do conhecimento negativo para
    positivo: Qnovo --> {V,F}
2.
3.  evolucao_NP(Qnovo):-
4.  si(-Qnovo,verdadeiro),
5.  remover(-Qnovo),
6.  evolucao(Qnovo).
```

```
| ?- evolucao_NP(utente(114,eduardoramalho,24,braga,manuelramalho,sararamalho,222123455,covid)).
yes
| ?- listing(utente).
utente(101, josepereira, 23, porto, henriquepereira, anapereira, 456789123, bronquite).
utente(102, luisdelgado, 33, famalicao, luisdelgado, dianadelgado, 345678912, covid).
utente(103, rodrigomendes, 25, barcelos, josemendes, luisamendes, 789123456, covid).
utente(104, claudiasoares, 20, famalicao, andresoares, joanasoares, 891234567, febre).
utente(105, ruigoncalves, 41, braga, sergiogoncalves, inesgoncalves, 912345678, diarreia).
utente(106, leonorgarcia, 9, guimaraes, diogogarcia, saragarcia, 657891234, covid).
utente(107, mariamagalhaes, 29, porto, ricardomagalhaes, soniamagalhaes, 567891234, bronquite).
utente(108, manuelsilva, 23, braga, joaosilva, teresasilva, 213456789, pneumonia).
utente(109, fernandocerqueira, 12, braga, nunocerqueira, joanacerqueira, 765891234, covid).
utente(110, ismaelbarbosa, 54, barcelos, jorgebarbosa, patriciabarbosa, 987654321, asma).
utente(111, isabelfreitas, 22, cidade, gabrielfreitas, anafreitas, 888811345, asma).
utente(112, nunamario, idade, porto, diogomario, isabelmario, 543524153, covid).
utente(130, joseramalho, 48, cidade1, inacioramalho, albertinaramalho, 345343431, covid).
utente(100, mickaelferreira, 23, braga, zeferreira, anaferreira, 989898970, asma).
utente(114, eduardoramalho, 24, braga, manuelramalho, sararamalho, 222123455, covid).
```

Figura 17 – Exemplo de evolução do conhecimento negativo para positivo

3.9.3. Evolução do conhecimento positivo para negativo

A evolução de conhecimento positivo para negativo, permite que um utente, serviço, consulta ou prestador de um determinado id que pertence ao conhecimento perfeito positivo seja agora substituído pelo mesmo, mas agora para conhecimento perfeito negativo. Para tal, é necessário que o facto passado tenho valor de verdade verdadeiro e, assim com recurso ao método involução, que irá verificar os invariantes relacionados com essa remoção, para remover esse facto e com recurso ao predicado evolução, que irá verificar os invariantes relacionados a essa inserção, para inserir o novo facto.

```
1.  % Extensao do predicado que permite a evolucao do conhecimento positivo para
    negativo: Qnovo --> {V,F}
2.  evolucao_PN(Qnovo):-
3.  si(Qnovo,verdadeiro),
4.  involucao(Qnovo),
5.  inserir(-Qnovo).
```

```
| ?- evolucao_PN(utente(105,ruigoncalves,41,braga,sergiogoncalves,inesgoncalves,912345678,diarreia)).
yes
| ?- listing(-).
-utente(A,B,C,D,E,F,G,H) :-
        nao(utente(A,B,C,D,E,F,G,H)),
        nao(excecao(utente(A,B,C,D,E,F,G,H))).
-utente(115,andrelopes,50,vilanovadegaia,carloslopes,veralopes,331115654,asma).
-utente(116,pedropastor,45,famalicao,henriquepastor,dianapastor,756832122,febre).
-utente(117,tiagomatos,20,braga,josematos,luisamatos,777123123,covid).
-servico(A,B,C,D,E) :-
        nao(servico(A,B,C,D,E)),
        nao(excecao(servico(A,B,C,D,E))).
-servico(220,1008,oscultar,hospitalprivadodebraga,braga).
-servico(221,1002,medirtensao,hospitalpublicodebraga,braga).
-servico(222,1001,oscultar,hospitalpublicodebraga,braga).
-consulta(A,B,C,D,E) :-
        nao(consulta(A,B,C,D,E)),
        nao(excecao(consulta(A,B,C,D,E))).
-consulta(20,12082020,100,201,30).
-consulta(21,20112020,101,202,80).
-consulta(22,22112020,102,203,20).
-prestador(A,B,C,D) :-
        nao(prestador(A,B,C,D)),
        nao(excecao(prestador(A,B,C,D))).
-prestador(1020,fabiomarques,pneumologista,hospitalpublicodebraga).
-prestador(1021,marciahenriqueta,medicinageral,hospitalpublicodebraga).
-prestador(1022,eduardapauleta,pediatra,hospitalsantaluzia).
-prestador(1023,otaviosoares,endocrinologia,hospitaldatrofa).
-utente(105,ruigoncalves,41,braga,sergiogoncalves,inesgoncalves,912345678,diarreia).
```

Figura 18 – Exemplo de evolução do conhecimento positivo para negativo

3.9.4. Evolução do conhecimento impreciso para incerto

A evolução do conhecimento impreciso para incerto permite que algo que era considerado desconhecido para determinados valores passa a ser desconhecido para todos os valores. Numa situação em que alguém ache que é entre duas situações possíveis, pode na verdade perceber que estava errado e então será desconhecido para todos os valores. Para tal, remove as exceções passadas relativas ao conhecimento impreciso usando o predicado involução e insere-se os novos factos e exceções que caracterizam o conhecimento incerto.

```
1. % Extensao do predicado que permite a evolucao do conhecimento impreciso para
   incerto: Qnovo, Qnovo_ex, Qnovo_exx, Qpassado1, Qpassado2 --> {V,F}
2. evolucao_ImpInc(Qnovo,Qnovo_ex, Qnovo_exx, Qpassado1, Qpassado2):-
3. si(Qpassado1,desconhecido),
4. si(Qpassado2,desconhecido),
5. involucao(excecao(Qpassado1)),
6. involucao(excecao(Qpassado2)),
7. inserir(Qnovo),
8. inserir((excecao(Qnovo_ex) :- Qnovo_exx)).
```

```
| ?- evolucao_ImpInc(utente(120,cristinaferreira,25, barcelos,fabioferreira,joanaferreira,222255556,doenca),
utente(A,B,C,D,E,F,G,H), utente(A,B,C,D,E,F,G,doenca), utente(120,cristinaferreira,25, barcelos,fabioferreira
,joanaferreira,222255556,asma), utente(120,cristinaferreira,25, barcelos,fabioferreira,joanaferreira,22225555
6,covid)).
yes
```

Figura 19 – Exemplo de evolução do conhecimento impreciso para incerto

3.9.5. Evolução do conhecimento positivo para interdito

Nesta evolução, algo que se considerava como conhecimento perfeito positivo, passa agora a ser considerado como desconhecido e não se pode permitir a sua inserção na base de conhecimento posteriormente. Para tal, recorre-se ao predicado involução para permitir a remoção do facto passado e insere-se os novos factos específicos do conhecimento interdito.

```
% Extensao do predicado que permite a evolucao do conhecimento positivo para interdito: Qnovo, Qnovo_ex, Qnovo_exx, Qnovo2, Qnovo3,Qpassado --> {V,F}
evolucao_PInt(Qnovo, Qnovo_ex, Qnovo_exx, Qnovo2, Qnovo3,Qpassado):-
si(Qpassado,verdadeiro),
involucao(Qpassado),
inserir(Qnovo),
inserir((excecao(Qnovo_ex) :- Qnovo_exx)),
inserir(Qnovo2),
inserir(Qnovo3).
```

```
| ?- evolucao_PInt(utente(107,nome,29,porto,ricardomagalhaes,soniamagalhaes,567891234,bronquite), utente(A,B,C,D,E,F,G,H), utente(A, nome, C,D,E,F,G,H), interdito(nome), (+utente(A,B,C,D,E,F,G,H)::(solucoes(B,(utente(107,B,29,porto,ricardomagalhaes,soniamagalhaes,567891234,bronquite), nao(interdito(B))),T), comprimento(T,Z),Z==0)), utente(107,mariamagalhaes,29,porto,ricardomagalhaes,soniamagalhaes,567891234,bronquite)).
yes
| ?- si(utente(107,mariamagalhaes,29,porto,ricardomagalhaes,soniamagalhaes,567891234,bronquite),R).
R = desconhecido ?
yes
| ?- evolucao(utente(107,mariamagalhaes,29,porto,ricardomagalhaes,soniamagalhaes,567891234,bronquite)).
no
```

Figura 20 – Exemplo de evolução do conhecimento positivo para interdito

3.9.6. Evolução do conhecimento interdito para positivo

Nesta evolução, algo que se considerava como conhecimento desconhecido interdito e que não se pode permitir a sua inserção na base de conhecimento, passa agora a ser considerado perfeito positivo. Para tal, remove-se os factos específicos do conhecimento interdito e recorre-se ao predicado evolução para inserir o novo facto.

```
% Extensao do predicado que permite a evolucao do conhecimento interdito para positivo: Qnovo,Qpassado,Qpassado2,Qpassado3,Qpassado4,Qpassado5 --> {V,F}
evolucao_IntP(Qnovo, Qpassado,Qpassado2, Qpassado3,Qpassado4, Qpassado5):-
si(Qpassado,verdadeiro),
remover(Qpassado),
remover((excecao(Qpassado2) :- Qpassado3)),
remover(Qpassado4),
remover(Qpassado5),
evolucao(Qnovo).
```

```
| ?- evolucao_IntP(prestador(1040,gabrielmario,especialidade3,hospitalpublicodebraga),prestador(1040,gabrielmario,especialidade3,hospitalpublicodebraga), prestador(Q,R,S,L), prestador(Q,R,especialidade3,L), interdito(especialidade3), +prestador(Q,R,S,L)::(solucoes(S,(prestador(1040,gabrielmario,S,hospitalpublicodebraga),nao(interdito(S))),T),comprimento(T,Z),Z==0)).
yes
```

Figura 21 – Exemplo de evolução do conhecimento interdito para positivo

3.9.7. Evolução do conhecimento impreciso para positivo

A evolução do conhecimento impreciso para positivo permite que algo que era considerado como desconhecido entre determinados valores possa agora ser considerado conhecimento perfeito positivo na base de conhecimento. Para tal, recorre-se ao predicado involução para efetuar a remoção das exceções e ao predicado evolução para inserir o novo facto.

```
1. % Extensao do predicado que permite a evolucao do conhecimento impreciso para
   positivo: Qnovo, Qpassado1, Qpassado2 --> {V,F}
2. evolucao_ImpP(Qnovo,Qpassado1,Qpassado2):-
3. si(Qpassado1,desconhecido),
4. si(Qpassado2,desconhecido),
5. involucao(excecao(Qpassado1)),
6. involucao(excecao(Qpassado2)),
7. evolucao(Qnovo).
```

```
| ?- evolucao_ImpP(servico(230,diogoamaral,medirtensao,hospitaldatrofa,trofa),servico(230,diogoamaral,medirtensao
,hospitaldatrofa,trofa),servico(230,tiagoborges,medirtensao,hospitaldatrofa,trofa)).
yes
```

Figura 22 – Exemplo de evolução do conhecimento impreciso para positivo

3.9.8. Evolução do conhecimento incerto para positivo

A evolução do conhecimento incerto para positivo permite que algo que era considerado como desconhecido para todos os valores, passa a tornar-se conhecimento perfeito positivo na base de conhecimento. Para tal, remove-se os factos e exceções específicos do conhecimento incerto e recorre-se ao predicado evolução para inserir o novo facto positivo.

```
1. % Extensao do predicado que permite a evolucao do conhecimento incerto para
   positivo: Qnovo, Qpassado--> {V,F}
2. evolucao_IncP(Qnovo, Qpassado):-
3. si(Qpassado,verdadeiro),
4. remover(Qpassado),
5. evolucao(Qnovo).
```

```
| ?- evolucao_IncP(utente(112,nunamario,75,porto,diogomario,isabelmario,543524153,covid),utente(112,nunamario,ida
de,porto,diogomario,isabelmario,543524153,covid)).
yes
```

Figura 23 – Exemplo de evolução do conhecimento incerto para positivo

3.9.9. Evolução do conhecimento incerto para impreciso

A evolução do conhecimento incerto para impreciso permite tornar algo que era considerado desconhecido para todos os valores para algo que agora é desconhecido para determinados valores. Para tal, remove-se o facto específico do conhecimento incerto e com recurso ao predicado evolução insere-se as exceções respetivas do conhecimento impreciso.

```
1. % Extensao do predicado que permite a evolucao do conhecimento incerto para
   impreciso: Qnovo1, Qnovo2, Qpassado --> {V,F}
2. evolucao_IncImp(Qnovo1,Qnovo2, Qpassado):-
3. si(Qpassado,verdadeiro),
4. remover(Qpassado),
5. evolucao(excecao(Qnovo1)),
6. evolucao(excecao(Qnovo2)).
```

```
| ?- evolucao_IncImp(utente(111,isabelfreitas,22,braga,gabrielfreitas,anafreitas,888811345,asma), utente(111,
isabelfreitas,22,porto,gabrielfreitas,anafreitas,888811345,asma), utente(111,isabelfreitas,22,cidade,gabrielf
reitas,anafreitas,888811345,asma)).
yes
```

Figura 24 – Exemplo de evolução do conhecimento incerto para impreciso

3.9.10. Evolução do conhecimento positivo para incerto

A evolução do conhecimento positivo para incerto permite que tornar algo que era considerado como verdadeiro no passado e que agora passa a ser considerado como desconhecido para todos os valores no respetivo atributo. Para tal recorre-se ao predicado involução para remover o facto passado e inserem-se os factos relativos ao conhecimento incerto.

```
1. % Extensao do predicado que permite a evolucao do conhecimento positivo para
   incerto: Qnovo, Qnovo_ex, Qnovo_exx,Qpassado --> {V,F}
2. evolucao_PInc(Qnovo,Qnovo_ex, Qnovo_exx, Qpassado):-
3. si(Qpassado1,verdadeiro),
4. involucao(Qpassado),
5. inserir(Qnovo),
6. inserir((excecao(Qnovo_ex) :- Qnovo_exx)).
```

```
| ?- evolucao_PInc(utente(100,margaridaalves,19,braga,pai,joanaalves,123456789,asma),utente(A,B,C,D,E,F,G,H), utent
e(A,B,C,D,pai,F,G,H),utente(100,margaridaalves,19,braga,manuelalves,joanaalves,123456789,asma)).
yes
```

Figura 25 – Exemplo de evolução do conhecimento positivo para incerto

3.9.11. Evolução do conhecimento positivo para impreciso

A evolução do conhecimento positivo para impreciso permite tornar algo que era considerado como verdadeiro para algo que agora é desconhecido para determinados valores. Para tal, recorre-se ao predicado involução para remover o facto passado e ao predicado evolução para se inserir as exceções que caracterizam o conhecimento impreciso.

```
1. % Extensao do predicado que permite a evolucao do conhecimento positivo para i
   mpreciso: Qnovo1,Qnovo2, Qpassado --> {V,F}
2. evolucao_PImp(Qnovo1,Qnovo2, Qpassado):-
3. si(Qpassado,verdadeiro),
4. involucao(Qpassado),
5. evolucao(excecao(Qpassado1)),
6. evolucao(excecao(Qpassado2)).
```

```
| ?- evolucao_PImp(consulta(1,06112020,100,200,30), consulta(1,06112020,100,200,40),consulta(1,06112020,100,200,80)).
yes
```

Figura 26 – Exemplo de evolução do conhecimento positivo para impreciso

3.9.12. Atualização do conhecimento

Os predicados “atualizar_utente”, “atualizar_servico”, “atualizar_consulta” e “atualizar_prestador” permitem que ocorram atualizações nos atributos específicos para cada respetivo predicado: utente, serviço, consulta ou prestador. Para tal, recorre-se ao predicado involução para remover o facto anterior e ao predicado evolução que também irá testar os invariantes que dizem respeito à inserção de conhecimento e irá inserir o facto atualizado.

3.9.12.1. Atualizar utente

O utente pode atualizar dados como: a sua idade, a cidade onde vive atualmente e sua doença.

```
1. atualizar_utente(utente(A,B,C,D,E,F,G,H), utente(A,B,Cn,Dn,E,F,G,Hn)):-
2. involucao(utente(A,B,C,D,E,F,G,H)),
3. evolucao(utente(A,B,Cn,Dn,E,F,G,Hn)).
```

```
| ?- atualizar_utente(utente(104,claudiasoares,20,famalicao,andresoares,joanasoares,891234567,febre), utente(
104,claudiasoares,21,famalicao,andresoares,joanasoares,891234567,covid)).
yes
```

Figura 27 – Exemplo de atualização de idade e doença num utente

3.9.12.2. Atualizar serviço

Pode-se atualizar dados no predicado serviço, tal como: o id do prestador que exerce esse serviço.

```
1. atualizar_servico(servico(M,I,N,L,O), servico(M,In,N,L,O)):-
2. involucao(servico(M,I,N,L,O)),
3. evolucao(servico(M,In,N,L,O)).
```

```
| ?- atualizar_servico(servico(200,1005,oscultar,hospitaldatrofa,trofa), servico(200,1006,oscultar,hospitalda
trofa,trofa)).
yes
```

Figura 28 – Exemplo de atualização de idade e doença num utente

3.9.12.3. Atualizar consulta

Pode-se atualizar dados no predicado consulta, tais como: a data de realização, o id do serviço prestado e o custo dessa consulta.

```
1. atualizar_consulta(consulta(P,Q,A,M,R), consulta(P,Qn,A,Mn,Rn)):-
2. involucao(consulta(P,Q,A,M,R)),
3. evolucao(consulta(P,Qn,A,Mn,Rn)).
```

```
| ?- atualizar_consulta(consulta(4,25112020,109,208,25),consulta(4,25112020,109,208,50)).
yes
```

Figura 29 – Exemplo de atualização do custo de uma consulta

3.9.12.4. Atualizar prestador

Pode-se atualizar dados no predicado prestador, tais como: a sua especialidade e a instituição onde exerce os seus cuidados.

```
1. atualizar_prestador(prestador(I,J,K,L), prestador(I,J,Kn,Ln)):-
2. involucao(prestador(I,J,K,L)),
3. evolucao(prestador(I,J,Kn,Ln)).
```

```
| ?- atualizar_prestador(prestador(1007,danielnunes,endocrinologia,hospitalsantaluzia), prestador(1007,daniel
nunes,endocrinologia,hospitaldatrofa)).
yes
```

Figura 30 – Exemplo de atualização do local de trabalho do prestador

3.9.13. Inserção de conhecimento

Para inserir conhecimento positivo, usa-se o predicado auxiliar "Evolucao", tal como descrito no ponto 3.5.

Para inserir conhecimento negativo, começa-se por inserir conhecimento positivo (com valor de falso, isto é, sabe-se que é apenas para inserir conhecimento negativo) e evolui-se para conhecimento negativo.

Para inserir conhecimento desconhecido, usamos o mesmo processo. Introduz-se os dados como se se tratasse de conhecimento positivo (que na realidade não o é), e depois evolui-se para qualquer um dos conhecimentos pretendidos (incerto, impreciso ou interdito).

Por exemplo, querendo inserir conhecimento interdito na base de dados de um utente em que não se pode saber a sua idade. Começa-se por usar o predicado "evolução" para inserir conhecimento positivo. Como a idade é interdita, começa-se por preencher a idade com um valor aleatório. Desta forma, é assumido que de conhecimento positivo se trata (figura 31). De seguida, usando o predicado que evolui conhecimento positivo em interdito, "insere-se" o conhecimento pretendido – neste exemplo, interdito (figura 32).

```
| ?- evolucao(utente(200,paulofonseca,100,braga,zefonseca,sarafonseca, 000121003, covid)).
yes
```

Figura 31 – Apesar da idade do utente ser interdita, adiciono-o com idade 100 (valor aleatório)

```
| ?- evolucao_PInt(utente(200, paulofonseca, idade, braga, zefonseca, sarafonseca, 121003, covid), utente(A,B
,C,D,E,F,G,H), utente(A, B, idade,D,E,F,G,H), interdito(idade), (+utente(A,B,C,D,E,F,G,H)::(solucoes(C,(utent
e(200, paulofonseca, C, braga, zefonseca, sarafonseca, 121003, covid), nao(interdito(C))),T), comprimento(T,Z
),Z==0)), utente(200, paulofonseca, 100, braga, zefonseca, sarafonseca, 121003, covid)).
yes
```

Figura 32 – Evolução do conhecimento positivo para interdito (deixando a idade interdita)

3.10. Novo sistema de Inferência

Com este novo sistema de inferência conseguimos relacionar várias questões usando conjunções ou disjunções e, consequentemente, obter o valor de verdade final. Para tal, usamos os predicados siC e siD que poderão receber como argumentos dois termos. Esses dois termos poderão ser uma questão ou um siC ou siD dentro de um siC ou siD. Assim, é possível agrupar duas questões de cada vez e obter no final o valor de verdade para essa composição de questões.

```
% siC ou siD para composicao de valores de verdade --> {V,F,D}

siC( Q1 e Q2,R ) :- siand( Q1,R1 ), siand( Q2,R2 ), conjuncao(R1,R2,R).
siand(siC(Q1 e Q2),R):-siand(Q1,R1),siand(Q2,R2),conjuncao(R1,R2,R).
siand(siD(Q1 ou Q2),R):-siand(Q1,R1),siand(Q2,R2),disjuncao(R1,R2,R).
siand(A,B) :- si(A,B).

siD( Q3 ou Q4,S ) :- sior( Q3,R3 ), sior( Q4,R4 ), disjuncao(R3,R4,S).
sior(siD(Q3 ou Q4),S):-sior(Q3,R3),sior(Q4,R4),disjuncao(R3,R4,S).
sior(siC(Q3 e Q4),S):-sior(Q3,R3),sior(Q4,R4),conjuncao(R3,R4,S).
sior(A,B) :- si(A,B).

disjuncao(verdadeiro,verdadeiro,verdadeiro).
disjuncao(verdadeiro,desconhecido,desconhecido).
disjuncao(verdadeiro,falso,verdadeiro).
disjuncao(desconhecido,verdadeiro,desconhecido).
disjuncao(desconhecido,desconhecido,desconhecido).
disjuncao(desconhecido,falso,falso).
disjuncao(falso,verdadeiro,verdadeiro).
disjuncao(falso,desconhecido,falso).
disjuncao(falso,falso,falso).

conjncao(verdadeiro,verdadeiro,verdadeiro).
conjuncao(verdadeiro,desconhecido,verdadeiro).
conjuncao(verdadeiro,falso,falso).
conjuncao(desconhecido,verdadeiro,verdadeiro).
conjuncao(desconhecido,desconhecido,desconhecido).
conjuncao(desconhecido,falso,desconhecido).
conjuncao(falso,verdadeiro,falso).
conjuncao(falso,desconhecido,desconhecido).
conjuncao(falso,falso,falso).
```

A tabela 1 mostra as relações de conjunção e disjunção entre os diferentes valores de verdade.

Tabela 1 - Conjunção e Disjunção entre valores de verdade

Q1	Q2	Conjunção	Disjunção
V	V	V	V
V	D	V	D
V	F	F	V
D	V	V	D
D	D	D	D
D	F	D	F
F	V	F	V
F	D	D	F
F	F	F	F

As imagens abaixo mostram 2 exemplos da aplicação do Sistema de Inferência desenvolvido.

Neste primeiro exemplo tem-se:

$$(V\ e\ F)\ e\ (D\ ou\ F)\ \equiv\ (F\ e\ F)\ \equiv\ F$$

```
| ?- siC(siC(prestador(1000,joaovasquez,medicinageral,hospitalpublicodebraga) e utente(114,eduardoramalho,24,braga,
manuelramalho,sararamalho,222123455,covid)) e siD(utente(120,cristinaferreira,25, barcelos,fabioferreira,joanaferre
ira,222255556,covid) ou servico(220,1008,oscultar,hospitalprivadodebraga,braga)), R).
R = falso ?
yes
```

Figura 33 – Exemplo de aplicação dos meta-predicados siD e siC

Neste segundo exemplo tem-se:

$$(D\ e\ V)\ ou\ F\ \equiv\ (V\ ou\ F)\ \equiv\ V$$

```
| ?- siD(siC(utente(111,isabelfreitas,22,porto,gabrielfreitas,anafreitas,888811345,asma) e consulta(9,30112020,110,
206,30)) ou servico(222,1001,oscultar,hospitalpublicodebraga,braga), R).
R = verdadeiro ?
yes
```

Figura 34 – Exemplo de aplicação dos meta-predicados siD e siC

4. CONCLUSÃO

A elaboração do segundo trabalho prático permitiu aprofundar os conhecimentos da linguagem PROLOG, visto que podemos dar continuidade ao que foi apendido no primeiro trabalho prático e durantes as aulas práticas da Unidade Curricular.

Cumprimos com todos os requisitos pretendidos para este trabalho prático, uma vez que foram implementados mecanismos de representação de conhecimento perfeito positivo, negativo, bem como desconhecido. Representamos, também, os invariantes que consideramos mais relevantes para permitir a remoção e inserção de conhecimento do sistema. Além disso, conseguimos lidar com a problemática da evolução do conhecimento, implementando os predicados necessários para a sua realização e, por fim, elaboramos um novo sistema de inferência que permite, de forma elegante, obter o valor de verdade final relativamente a um conjunto de questões.

É de salientar algumas dificuldades resultantes tanto na problemática de evolução do conhecimento como na realização do novo sistema de inferência, já que correspondem aos dois pontos de maior dificuldade do trabalho. No entanto, essas dificuldades foram ultrapassadas, permitindo obter os resultados que, desde o início, pretendíamos.

Concluindo, os resultados obtidos correspondem, sem dúvida, tanto às expectativas como à exigência do trabalho prático.

5. REFERÊNCIAS

ANALIDE, César; NEVES, José. "Representação de Informação Incompleta", 1996.

ALVES, João; ALVES, Paulo. “Programação em Lógica, Conhecimento e Raciocínio – Relatório, Parte 1”, 2020.

6. ANEXOS

A. Conhecimento perfeito Positivo

```
%predicado utente: IdUt, Nome, Idade, Cidade, Pai, Mãe, NIF,doenca  -> {V,F,D}
utente(100,margaridaalves,19,braga,manuelalves,joanaalves,123456789,asma).
utente(101,josepereira,23,porto,henriquepereira,anapereira,456789123,bronquite).
utente(102,luisdelgado,33,famalicao,luisdelgado,dianadelgado,345678912,covid).
utente(103,rodrigomendes,25,barcelos,josemendes,luisamendes,789123456,covid).
utente(104,claudiasoares,20,famalicao,andresoares,joanasoares,891234567,febre).
utente(105,ruigoncalves,41,braga,sergiogoncalves,inesgoncalves,912345678,diarreia).
utente(106,leonorgarcia,9,guimaraes,diogogarcia,saragarcia,657891234,covid).
utente(107,mariamagalhaes,29,porto,ricardomagalhaes,soniamagalhaes,567891234,bronquite).
utente(108,manuelsilva,23,braga,joaosilva,teresasilva,213456789,pneumonia).
utente(109,fernandocerqueira,12,braga,nunocerqueira,joanacerqueira,765891234,covid).
utente(110,ismaelbarbosa,54,barcelos,jorgebarbosa,patriciabarbosa,987654321,asma).


%predicado servico: Idserv, IdPrest,Descrição, Instituição, Cidade -> {V,F,D}
servico(200,1005,oscultar,hospitaldatrofa,trofa).
servico(201,1003,medirtensao,hospitaldatrofa,trofa).
servico(202,1010,analises,hospitalsantaluzia,vianadocastelo).
servico(203,1004,medirfebre,hospitalpublicodebraga,braga).
servico(204,1006,analisessecrecoes,hospitalprivadodebraga,braga).
servico(205,1002,medirtensao,hospitalpublicodebraga,braga).
servico(206,1001,oscultar,hospitalpublicodebraga,braga).
servico(207,1008,oscultar,hospitalprivadodebraga,braga).
servico(208,1004,analises,hospitalpublicodebraga,braga).


%predicado consulta: IdCons, Data, IdUt, IdServ, Custo -> {V,F,D}
consulta(1,06112020,100,200,80).
consulta(2,10102020,105,204,110).
consulta(3,22122020,110,206,30).
consulta(4,25112020,109,208,25).
consulta(5,30112020,106,203,30).
consulta(6,10102020,109,202,60).
consulta(7,06112020,107,206,30).
consulta(8,25112020,102,203,35).
consulta(9,30112020,110,206,30).
consulta(10,10102020,103,203,40).
consulta(11,06112020,104,203,25).
consulta(12,30112020,109,208,35).


%predicado prestador: IdPrest, Nome, Especialidade, Instituição -> {V,F,D}
prestador(1000,joaovasquez,medicinageral,hospitalpublicodebraga).
prestador(1001,thomaskall,pneumologista,hospitalpublicodebraga).
prestador(1002,anasilva,medicinageral,hospitalprivadodebraga).
prestador(1003,josecorreia,pediatra,hospitaldatrofa).
prestador(1004,gabrielarodrigues,medicinageral,hospitalpublicodebraga).
prestador(1005,franciscomenezes,pneumologista,hospitaldatrofa).
prestador(1006,franciscaalves,endocrinologia,hospitalprivadodebraga).
prestador(1007,danielnunes,endocrinologia,hospitalsantaluzia).
prestador(1008,henriquelage,pneumologista,hospitalprivadodebraga).
prestador(1009,saralurdes,pediatra,hospitalsantaluzia).
prestador(1010,hugopereira,medicinageral,hospitalsantaluzia).
```