Sistemas de Representação de Conhecimento e Raciocínio

Exercício 1

Relatório de Desenvolvimento

Grupo 18

Cesário Miguel Pereira Perneta - A73883
Luís Miguel Bravo Ferraz - A70824
Rui Pedro Barbosa Rodrigues - A74572
Tiago Miguel Fraga Santos - A74092

19 de Abril de 2018

Resumo

Neste relatório irá ser abordado a resolução e verificação do segundo trabalho prático da Unidade Curricular de Sistemas de Representação de Conhecimento e Raciocínio. Será possível encontrar uma detalhada explicação dos predicados elaborados, bem como a apresentação do código usado na representação dos predicados fundamentais, que visam dar resposta aos requisitos propostos neste trabalho prático.

Conteúdo

1 Introdução **2**

2 Análise do Problema **3**

3 Enunciado **4**

3.1 Base de Conhecimento 4

3.1.1 Enunciado 4

3.1.2 Extra - Enuncidado 6

3.2 Representar conhecimento Positivo e Negativo 7

3.3 Representar casos de conhecimento imperfeito, pela utilização de valores nulos de todos os tipos estudados 7

3.3.1 Conhecimento Imperfeito - Incerto 7

3.3.2 Conhecimento Imperfeito - Impreciso 9

3.3.3 Conhecimento Imperfeito - Interdito 9

3.4 Manipular invariantes que designem restricões à inserção e à remoção de conhecimento do sistema . . 10

3.4.1 Invariantes - Conhecimento Imperfeito Incerto 10

3.5 Lidar com a problemática da evolução do conhecimento, criando os procedimentos adequados 11

3.6 Desenvolver um sistema de inferência capaz de implementar os mecanismos de raciocínio inerentes a estes sistemas 11

4 Conclusão **12**

Capítulo 1

Introdução

O segundo exercício proposto tem como objetivo utilizarmos a extensão à programação em lógica usando a linguagem PROLOG, bem como a representação de conhecimento imperfeito, recorrendo a utilização de valores nulos e da criação de mecanismos de raciocínio adequados. Tal como no primeiro exercício foi nos pedido para elaborarmos um sistema de representação de conhecimento e raciocíno com capacidade para caracterizar um universo de discurso na área da prestação de cuidados de saúde. Nos capítuos seguintes vamos descrever a forma como abordamos o problema, bem como a solução implementada para resolver o problema que nos foi dado.

Capítulo 2

Análise do Problema

Na elaboração do nosso caso prático, começamos por definir a nossa base de conhecimento, dividida em utentes, prestadores e cuidados médicos - que eram os predicados minimos obrigatórios - e como extra acrescentamos tambem, o predicado data, instituição e morada. Assim como foi proposto, usamos valores nulos para representar situações de conhecimento imperfeito sendo apresentado com mais detalhe ao longo do relatório. Assim que definida a base de conhecimento, começamos por implementar os predicados necessários para respeitar os requisitos deste exercício. Por fim elaboramos o sistema de inferência capaz de implementar os mecanismos de raciocínio deste sistema. Na criação do sistema de inferência tivemos como base o que foi lecionado nas aulas práticas, idealizando mais 3 sistemas de inferência que achamos necessários para o problema em questão, um que calcula a lista de respostas sobre uma lista de questões, outro que calcula a resposta consuante a disjunção de uma lista de perguntas, enquanto que outro calcula a resposta consuante a conjunção de uma lista de perguntas.

Capítulo 3

Enunciado

3.1 Base de Conhecimento

3.1.1 Enunciado

Utentes

```
utente: #IdUt, Nome, Idade, Morada -> {V,F}

utente(1,'Tiago',23,morada('Rua 1','Esposende')).
utente(2,'Cesario',21,morada('Rua 2','Valença')).
utente(3,'Luis',22,morada('Rua 3','Viana')).
utente(4,'Rui',21,morada('Rua 4','Braga')).
utente(5,'Marega',27,morada('Rua 5','Porto')).
utente(6,'Tiquinho',25,morada('Rua 6','Porto')).
utente(7,'Brahimi',26,morada('Rua 7','Olival')).
utente(8,'Aboubakar',26,morada('Rua 8','Olival')).
utente(9,'Danilo',25,morada('Rua 9','Porto')).
utente(10,'Herrera',25,morada('Rua 10','Porto')).

excecao(utente(20,'Deco',40, morada('Rua da Magia','Porto'))).
excecao(utente(99, 'Vitor Baia', 45, morada('Rua dos Titulos','Porto'))).
```

Prestadores:

```
prestador: #IdPrest, Nome, Especialidade,Instituição -> {V,F}

prestador(1,'Nelon Puga','Desporto',
    instituicao('Clinica do Dragao', morada('Rua das Antas','Porto'))).
prestador(2,'Joaquim','Cardiologia',
    instituicao('Clinica do Dragao', morada('Rua das Antas','Porto'))).
prestador(3,'Antonio','Fisioterapia',
    instituicao('Clinica do Dragao', morada('Rua das Antas','Porto'))).
prestador(4,'Pedro','Radiologia',
    instituicao('Clinica do Dragao', morada('Rua das Antas','Porto'))).
prestador(5,'Manuel','Oftalmogia',
    instituicao('Clinica do Dragao', morada('Rua das Antas','Porto'))).

prestador(6,'Ines','Cardiologia',
    instituicao('Hospital Braga', morada('Rua do Hospital','Braga'))).
```

```
prestador(7,'Joana','Cirurgia',
    instituicao('Hospital Braga', morada('Rua do Hospital','Braga'))).
prestador(8,'Maria','Radiologia',
    instituicao('Hospital Braga', morada('Rua do Hospital','Braga'))).
prestador(9,'Diana','Oftalmogia',
    instituicao('Hospital Braga', morada('Rua do Hospital','Braga'))).
prestador(10,'Ana','Fisioterapia',
    instituicao('Hospital Braga', morada('Rua do Hospital','Braga'))).

prestador(11,'Carlos','Cardiologia',
    instituicao('Hospital Sao Joao', morada('Rua da Circunvalacao','Porto'))).
prestador(12,'Tomas','Cirurgia',
    instituicao('Hospital Sao Joao', morada('Rua da Circunvalacao','Porto'))).
prestador(13,'Raquel','Radiologia',
    instituicao('Hospital Sao Joao', morada('Rua da Circunvalacao','Porto'))).
prestador(14,'Luisa','Oftalmogia',
    instituicao('Hospital Sao Joao', morada('Rua da Circunvalacao','Porto'))).
prestador(15,'Teresa','Fisioterapia',
    instituicao('Hospital Sao Joao', morada('Rua da Circunvalacao','Porto'))).

excecao(prestador(70,'Pinto da Costa','Presidente',
    instituicao('Clinica do Dragao',morada('Rua das Antas','Porto')))).
excecao(prestador(71,'Reinaldo Teles','Vice-Presidente',
    instituicao('Clinica do Dragao',morada('Rua das Antas','Porto')))).
```

Cuidados Médicos:

```
cuidado: Data, #IdUt, #IdPrest, Descrição, Custo -> {V,F}

cuidado(data(10,02,2018),1,9,'Consulta Rotina',30).
cuidado(data(10,02,2018),2,9,'Consulta Rotina',30).
cuidado(data(10,02,2018),3,9,'Consulta Rotina',30).
cuidado(data(10,02,2018),4,9,'Consulta Rotina',30).
cuidado(data(11,02,2018),5,9,'Consulta Rotina',30).

cuidado(data(15,02,2018),5,1,'Consulta Pos-Match',10).
cuidado(data(15,02,2018),6,1,'Consulta Pos-Match',10).
cuidado(data(15,02,2018),7,1,'Consulta Pos-Match',10).
cuidado(data(15,02,2018),8,1,'Consulta Pos-Match',10).
cuidado(data(15,02,2018),9,1,'Consulta Pos-Match',10).

cuidado(data(16,02,2018),5,11,'Eletrocardiograma',20).
cuidado(data(16,02,2018),6,11,'Eletrocardiograma',20).
cuidado(data(16,02,2018),7,11,'Eletrocardiograma',20).
cuidado(data(17,02,2018),10,15,'Fisio - Coxa Direita',10).
cuidado(data(17,02,2018),8,15,'Fisio - Gemeo Esq',10).

cuidado(data(18,03,2018),10,13,'Raio X',50).
cuidado(data(10,03,2018),5,1,'Consulta Pos-Match',10).
cuidado(data(10,03,2018),7,1,'Consulta Pos-Match',10).
cuidado(data(10,03,2018),8,1,'Consulta Pos-Match',10).
cuidado(data(10,03,2018),9,1,'Consulta Pos-Match',10).

cuidado(data(12,03,2018),6,15,'Fisio - Coxa Esq',10).
```

```
cuidado(data(12,03,2018),7,15,'Fisio - Joelho Dir',10).
cuidado(data(12,03,2018),8,15,'Fisio - Costas',10).
cuidado(data(12,03,2018),9,15,'Fisio - Virilha',10).
cuidado(data(12,03,2018),10,15,'Fisio - Coxa Dir',10).

cuidado(data(20,02,2018),1,9,'Consulta',30).
cuidado(data(20,02,2018),2,6,'Eletrocardiograma',25).
cuidado(data(20,02,2018),3,7,'Intervenção',30).
cuidado(data(20,02,2018),4,8,'Consulta Rotina',30).
cuidado(data(20,02,2018),5,5,'Consulta Rotina',30).

excecao(cuidado(data(18,04,2018),99,70,'Reuniao',10)).
excecao(cuidado(data(18,04,2018),20,70,'Reuniao',10)).
```

3.1.2 Extra - Enuncidado

Além do enunciado proposto, decidimos considerear estes dois predicados extra, de forma a oferecer mais dinamismo e, principalmente, organização na nossa base de conhecimento de forma a que a seleção de informação fosse feita de uma forma mais rica e sistemática.

Instituições:

```
instituição: Data, #IdUt, #IdPrest, Descrição, Custo -> {V,F}

instituicao('Clinica do Dragao', morada('Rua das Antas','Porto')).
instituicao('Hospital Sao Joao', morada('Rua da Circunvalacao','Porto')).

instituicao('Hospital Braga', morada('Rua do Hospital','Braga')).
instituicao('Centro Saude', morada('Rua Centro','Braga')).

instituicao('Hospital Viana', morada('Rua do Hospital','Viana')).
instituicao('Hospital Esposende', morada('Rua do Hospital', 'Esposende')).

excecao(instituicao('Campo do Olival', morada('Rua do Olival','Gaia'))).
excecao(instituicao('Estado do Dragao', morada('Rua das Antas','Porto'))).
```

Moradas:

```
morada: Rua,Localidade -> {V,F}
```

Datas:

```
data: Dia, Mes, Ano -> {V,F}

data(D, M, A) :- pertence(M, [1,3,5,7,8,10,12]), D >= 1, D =< 31.
data(D, M, A) :- pertence(M, [4,6,9,11]), D >= 1, D =< 30.
data(D, 2, A) :- A mod 4 =\= 0, D >= 1, D =< 28.
data(D, 2, A) :- A mod 4 =:= 0, D >= 1, D =< 29.
```

Como se pode verificar encontra-se acima representado conhecimento imperfeito atravées de valores nulos. Mais tarde será explicado os tipos de conhecimento imperfeito representados e o tratamento desse tipo de conhecimento.

3.2 Representar conhecimento Positivo e Negativo

Neste exercício ao contrário do anterior além de representarmos conhecimento positivo, também tivemos de representar conhecimento negativo.

```
%---------------------------------
% Conhecimento positivo
%---------------------------------


% Já está presente na base de conhecimento.



%---------------------------------
% Conhecimento negativo
%---------------------------------



-utente(ID,N,I,M) :-
    nao(utente(ID,N,I,M)),
    nao(excecao(utente(ID,N,I,M))).


-prestador(ID,N,E,I) :-
    nao(prestador(ID,N,E,I)),
    nao(excecao(prestador(ID,N,E,I))).


-cuidado(Dt,IDU,IDP,D,C) :-
    nao(cuidado(Dt,IDU,IDP,D,C)),
    nao(excecao(cuidado(Dt,IDU,IDP,D,C))).
```

Desta forma conseguimos representar conhecimento negativo acerca dos utentes, prestadores, cuidados médicos e instituições. Para a implementação utilizamos o predicado **nao** representando assim a negação forte.

3.3 Representar casos de conhecimento imperfeito, pela utilização de valores nulos de todos os tipos estudados

Nesta secção será explicada os casos de conhecimento imperfeito utilizados, bem como o tratamento desses casos.

3.3.1 Conhecimento Imperfeito - Incerto

```
%---------------------------------
% Utentes -> Incerto
%---------------------------------

utente(11,'Salah',27,incerto1).
```

```
excecao(utente(ID,N,I,M)) :- utente(ID,N,I,incerto1).

utente(12,'Firmino',incerto2,morada('Rua 4','Liverpool')).
excecao(utente(ID,N,I,M)) :- utente(ID,N,incerto2,M).

desc(incerto1).
desc(incerto2).
```

Este caso apresentado é conhecimento imperfeito incerto. Como podemos verificar o valor da morada do utente **Salah** é desconhecido e representado com o valor **incerto1**. No segundo caso, desconhece-se a idade do utente **Firmino**, sendo que esta é caracterizada pelo valor **incerto2**. Ambas as informações são definidas como exceções para serem itroduzidas na base de conhecimento.

```
%--------------------------------
% Prestadores -> Incerto
%--------------------------------

prestador(16,'Klopp','Radiologia',incerto3).
excecao(prestador(ID,N,E,I)) :- excecao(prestador(ID,N,E,incerto3)).

prestador(17,'Daniela',incerto4,instituicao('Hospital Braga', morada('Rua do Hospital','Braga'))).
excecao(prestador(ID,N,E,I)) :- excecao(prestador(ID,N,incerto4,I)).

desc(incerto3).
desc(incerto4).
```

Nestes dois casos desconhece-se igualmente a morada e a idade para os utentes **Klopp** e **Daniela**, respetivamente. Entes dois valores sao caracterizados sao caracterizados pelos valores **incerto3** e **incerto4**.

```
%--------------------------------
% Cuidado -> Incerto
%--------------------------------

cuidado(data(15,04,2018),9,1,'Consulta Pos-Match',incerto5).
excecao(cuidado(Dt,IDU,IDP,D,C)) :- cuidado(Dt,IDU,IDP,D,incerto5).

cuidado(data(15,04,2018),10,13,incerto6,50).
excecao(cuidado(Dt,IDU,IDP,D,C)) :- cuidado(Dt,IDU,IDP,incerto6,C).

desc(incerto5).
desc(incerto6).
```

Na caracterização destes dois cuidados médicos foram considerados incertos os campos **custo**, e **descrição**, sendo que foram caracterizados pelos valores **incerto5** e **incerto6**.

```
%--------------------------------
% Instituição -> Incerto
%--------------------------------

instituicao('Hospital Madrid', incerto7).
excecao(instituicao(N,M)) :- instituicao(N,incerto7).

desc(incerto7).
```

Por fim, no ultimo caso de conhecimento imperfeito incerto, na caracterização do predicado instituição foi considerado como valor incerto o campo **morada**, tendo sido caracterizado pelo valor **incerto7**.

3.3.2 Conhecimento Imperfeito - Impreciso

```
%---------------------------------
% Utentes -> Impreciso
%---------------------------------

excecao(utente(13,'Mane',40,morada('Rua 1','Liverpool'))).
excecao(utente(13,'Mane',27,morada('Rua 1','Liverpool'))).
```

Este caso trata-se de conhecimento imperfeito impreciso, visto o valor ser desconhecido, mas de um conjunto determinado de hipóteses. Como se pode verificar nãoo se sabe se a idade do utente **Mane** é **27** ou **40**.

```
%---------------------------------
% Prestadores -> Impreciso
%---------------------------------

prestador(18,'Gomes','Cirurgia',instituicao('Hospital Sao Joao', morada('Rua da Circunvalacao','Por
prestador(18,'Gomes','Radiologia',instituicao('Hospital Sao Joao', morada('Rua da Circunvalacao','P
```

Neste caso, não se sabe se a especialidade do prestador **Gomes** é **Radiologia** ou **Cirurgia**.

```
%---------------------------------
% Cuidado -> Impreciso
%---------------------------------

excecao(cuidado(data(16,02,2018),10,6,'Eletrocardiograma',27)).
excecao(cuidado(data(16,02,2018),10,6,'Eletrocardiograma',38)).
```

Na caracterização deste caso, nao se sabe se o valor do cuidado médico prestado na data **16 de Fevereiro de 2018**, ao utente com id número **10**, executado pelo prestador com o id número **6**, com a descrição de **'Eletrocardiograma'**, teve um custo de **27 euros** ou um custo de **38 euros**.

```
%---------------------------------
% Instituição -> Impreciso
%---------------------------------

excecao(instituicao('Hospital de Barcelona', morada('Rua Direita','Barcelona'))).
excecao(instituicao('Hospital de Barcelona', morada('Rua Esquerda','Barcelona'))).
```

Por fim, no último caso de conhecimento imperfeito impreciso, nao se sabe se a instituicao **Hospital de Barcelona**, é na **Rua Direita** ou na **Rua Esquerda** de Barcelona.

3.3.3 Conhecimento Imperfeito - Interdito

```
%---------------------------------
% Utentes -> Interdito
%---------------------------------

utente(14,'Henderson',interdito1,morada('Rua 7','Liverpool')).
excecao(utente(ID,N,I,M)) :- utente(ID,N,interdito1,M).
nulo(interdito1).
+utente(ID,N,I,M) :: solucoes((ID,N,Is,M), utente(14,'Henderson',interdito1,morada('Rua 7','Liverpo
```

Este tipo de conhecimento imperfeito caracteriza-se por nao se conhecer um dos valores da informação de um predicado e em momento algum se puderá vir a conhecer. Deste modo, definimos este tipo de conhecimento atribuindo ao utente

Henderson, que a sua idade fosse interdita, portanto, para o fazer definimos este campo como valor nulo designando-se **interdito1**. Além disto criamos o invariante que impede que seja introduzida a informação da idade para este utente.

```
%---------------------------------
% Prestadores -> Interdito
%---------------------------------

prestador(19,'Mafalda','Fisioterapia',interdito2).
excecao(prestador(ID,N,E,I)) :- excecao(prestador(ID,N,E,interdito2)).
nulo(interdito2).
+prestador(ID,N,E,I) :: solucoes((ID,N,E,Is), prestador(19,'Mafalda','Fisioterapia',interdito2), na
```

Neste caso, a prestadora **Mafalda** tinha como campo interdito a sua instituição, que foi declara como valor nulo designado por **interdito2**. Assim como no caso de cima, criamos também o invariante que não permite que seja introduzido o campo da instituição para esta prestadora.

```
%---------------------------------
% Cuidado -> Interdito
%---------------------------------


cuidado(interdito3,9,15,'Fisio - Virilha',10).
excecao(cuidado(Dt,IDU,IDP,D,C)) :- cuidado(interdito3,IDU,IDP,D,C).
nulo(interdito3).
+cuidado(Dt,IDU,IDP,D,C) :: solucoes((Dts,IDU,IDP,D,C), cuidado(interdito3,9,15,'Fisio - Virilha',1
```

Para o cuidado definido em cima, decidimos que seria ocultada a sua data. Desta forma, o campo data para este cuidado passou a ser designado pelo valor nulo **interdito3**. Como nos casos anteriores criamos tambem o invariante que não permite que seja introduzida a data para este cuidado médico.

```
%---------------------------------
% Instituição -> Interdito
%---------------------------------

instituicao('Hospital Manchester', interdito4).
excecao(instituicao(N,M)) :- instituicao(N,interdito4).
nulo(interdito4).
+instituicao(N,M) :: solucoes((N,Ms),instituicao('Hospital Manchester', interdito4),nao(nulo(Ms)),S
```

Por fim, no último caso do conhecimento imperfeito interdito, definimos que a instituição **Hospital Manchester** teria como valor nulo interdito a sua morada, deste modo, a morada é definida pelo valor **interdito4**. Foi criado o invariante que proibe a inserção do campo morada para esta instituição.

3.4 Manipular invariantes que designem restricões à inserção e à remoção de conhecimento do sistema

Nesta secção vamos apresentar os invariantes criados de forma a impedir a evolução do conhecimento imperfeito na base de conhecimento, garantindo, assim, a sua consistência. Estes invariantes sao criados com o auxilio dos predicados **desc**, **excecao** e **nulo**, nao sendo possivel fazer evoluir a base de conhecimento com estes predicados.

3.4.1 Invariantes - Conhecimento Imperfeito Incerto

```
+utente(ID,N,I,M) :: (solucoes((ID,N,I,M),
                        (utente(ID,N,I,M),
```

```
                                nao(desc(ID)),nao(desc(N)),nao(desc(I)),nao(desc(M))),S),
                                    comprimento(S,N),
                                        N == 0).


+prestador(ID,N,E,I) :: (solucoes((ID,N,E,I),
                                (prestador(ID,N,E,I),
                                    nao(desc(ID)),nao(desc(N)),nao(desc(E)),nao(desc(I))),S),
                                        comprimento(S,N),
                                            N == 0).


+cuidado(Dt,IDU,IDP,D,C) :: (solucoes((Dt,IDU,IDP,D,C),
                                    (cuidado(Dt,IDU,IDP,D,C),
                                        nao(desc(Dt)),nao(desc(IDU)),
                                            nao(desc(IDP)),nao(desc(D)),nao(desc(C))),S),
                                                comprimento(S,N),
                                                    N == 0).


+instituicao(N,M) :: (solucoes((N,M),
                           instituicao(S,M),
                               nao(desc(N)),nao(desc(M)),S),
                                   comprimento(S,T),
                                       T==0).
```

3.4.2 Invariantes - Conhecimento Imperfeito Impreciso

```
+excecao(utente(ID,N,I,M)) :: (solucoes((ID,N,I,M),
                                    (utente(ID,N,I,M),
                                        nao(excecao(utente(ID,N,I,M)))),S),
                                            comprimento(S,N),
                                                N == 0).

+excecao(prestador(ID,N,E,I)) :: (solucoes((ID,N,E,I),(
                                        prestador(ID,N,E,I),
                                            nao(excecao(prestador(ID,N,E,I)))),S),
                                               comprimento(S,N),
                                                    N == 0).

+excecao(cuidado(Dt,IDU,IDP,D,C)) :: (solucoes((Dt,IDU,IDP,D,C),
                                            (cuidado(Dt,IDU,IDP,D,C),
                                                nao(excecao(cuidado(Dt,IDU,IDP,D,C)))),S),
                                                    comprimento(S,N),
                                                        N == 0).

+excecao(instituicao(N,M)) :: (solucoes((N,M),
                                        instituicao(N,M),
                                            nao(excecao(N,M)),S),
                                                comprimento(S,T),T==0).
```

3.4.3 Invariantes - Conhecimento Imperfeito Interdito

```
+utente(ID,N,I,M) :: (solucoes((ID,N,I,M),
                        (utente(ID,N,I,M),
                            nao(nulo(ID)),nao(nulo(N)),nao(nulo(I)),nao(nulo(M))),S),
                                comprimento(S,N),
                                    N == 0).

+prestador(ID,N,E,I) :: (solucoes((ID,N,E,I),
                            (prestador(ID,N,E,I),
                                nao(nulo(ID)),nao(nulo(N)),nao(nulo(E)),nao(nulo(I))),S),
                                   comprimento(S,N),
                                        N == 0).

+cuidado(Dt,IDU,IDP,D,C) :: (solucoes((Dt,IDU,IDP,D,C),
                                (cuidado(Dt,IDU,IDP,D,C),
                                    nao(nulo(Dt)),nao(nulo(IDU)),
                                        nao(nulo(IDP)),nao(nulo(D)),nao(nulo(C))),S),
                                            comprimento(S,N),
                                                N == 0).

+instituicao(N,M) :: (solucoes((N,M),
                        instituicao(N,M),
                            nao(nulo(N)),nao(nulo(M)),S),
                                comprimento(S,T),T==0).
```

3.5 Lidar com a problemática da evolução do conhecimento, criando os procedimentos adequados

Ao longo deste trabalho tivemos que lidar com o problema da representação de conhecimento imperfeito, um dos problemas que tivemos foi se devíamos permitir a evolução de conhecimento imperfeito, ou se devíamos impedir que tal acontece-se. A solução que optamos por desenvolver foi a de impedir que o predicado evolucao - que na nossa base de conhecimento está definido como **registar** - conseguisse evoluir conhecimento imperfeito, para tal desenvolvemos doze invariantes com esse efeito já apresentados anteriormente, salientando o predicado criado para que este método fosse possivel de implementar é o predicado **desc**, que tem uma função igual ao predicado **nulo**, no entanto, serve para indicar quais os valores que representam conhecimento imperfeito incerto. Depois de implementada esta solução conseguimos que o nosso programa lidasse com a problemática da evolução, não permitindo a inserção de conhecimento imperfeito.

3.6 Desenvolver um sistema de inferência capaz de implementar os mecanismos de raciocínio inerentes a estes sistemas

Nesta secção apresentamos os sistemas de inferência criados.

3.6.1 Demo - Apresentado nas Aulas

```
% Extensao do meta-predicado demo: Questao,Resposta -> {V,F,D}

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

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

Este é o demo apresentado nas aulas, este sistema de inferência vai servir de base para os restantes.

3.6.2 Demos - Desenvolvidos para o trabalho prático

```
% Extensao do meta-predicado demoLista: ListQuestao,ListResposta -> {V,F,D}

demoLista( [],[] ).
demoLista( [Questao|L],[Resposta|S] ) :- demo( Questao,Resposta ), demoLista(L,S).


%-------------------------------- - - - - - - - - - -  -  -  -  -   -
% Extensao do meta-predicado demoConjL: [Questao],Resposta -> {V,F,D}

demoConjL([],verdadeiro).
demoConjL([Q|L],verdadeiro) :- demo(Q,verdadeiro), demoConjL(L,verdadeiro).
demoConjL([Q|L],falso) :- demo(Q,falso), demoConjL(L,_).
demoConjL([Q|L],falso) :- demo(Q,_), demoConjL(L,falso).
demoConjL([Q|L],desconhecido) :- demo(Q,desconhecido), demoConjL(L,verdadeiro).
demoConjL([Q|L],desconhecido) :- demo(Q,verdadeiro), demoConjL(L,desconhecido).
demoConjL([Q|L],desconhecido) :- demo(Q,desconhecido), demoConjL(L,desconhecido).

%-------------------------------- - - - - - - - - - -  -  -  -  -   -
% Extensao do meta-predicado demoDisjL: [Questao],Resposta -> {V,F,D}

demoDisjL([],falso).
demoDisjL([Q|L],verdadeiro) :- demo(Q,verdadeiro), demoDisjL(L,_).
demoDisjL([Q|L],verdadeiro) :- demo(Q,_), demoDisjL(L,verdadeiro).
demoDisjL([Q|L],falso) :- demo(Q,falso), demoDisjL(L,falso).
demoDisjL([Q|L],desconhecido) :- demo(Q,falso), demoDisjL(L,desconhecido).
demoDisjL([Q|L],desconhecido) :- demo(Q,desconhecido), demoDisjL(L,falso).
demoDisjL([Q|L],desconhecido) :- demo(Q,desconhecido),demoDisjL(L,desconhecido).
```

Aqui apresentamos os três *demos* desenvolvidos pelo grupo com o objetivo de satisfazer as necessidades do trabalho prático. O primeiro sistema de inferência denominado como **demoLista** foi criado com o intuito de apresentar uma **lista de respostas** de uma lista de questões passada como argumento do sistema. Ao criar este sistema, em vez de apresentar as questões uma a uma ao *demo*, é possivel passa-las todas como uma lista, e obtemos de seguida a resposta a todas essas questões. O segundo sistema de inferência denominado como **demoConjL** foi criado com o intuito de apresentar **uma resposta** com a **conjunção** - regra da Lógica - de todas as questões de uma lista passada como argumento ao sistema de inferência. O terceiro sistema de inferência denominado como **demoDisjL** foi criado com o intuito de apresentar **uma resposta** com a **disjunção** - regra da Lógica - de todas as questões de uma lista passada como argumento ao sistema de inferência. Em todos os casos usamos como predicado auxiliar o sistema de inferência criado nas aulas - *demo*.

Capítulo 4

Conclusão

Dado por terminado o segundo exercício desta unidade curricular podemos concluir que foi um bom modo de aprofundarmos os conhecimentos obtidos nas aulas acerca da programção em lógica estendida, bem como a representação de conhecimento imperfeito, com a utilização de valores nulos. As principais dificuldades encontradas foram, sobretudo, na forma como estruturar o nosso pensamento, de modo a abandonarmos o que antes considerávamos como certo. Assim que ultrapassado este obstáculo, a resolução do problema proposto tornou-se trivial. O trabalho desenvolvido é completo, competente e funcional, apresentando algumas funções extra implementadas que foram a evolução das funções extra do exercício um. No futuro este trabalho deixa em aberto um mar de oportunidades para o desenvolvimento de um sistema capaz de representar o conhecimento de um sistema de saúde real.