Universidade do Minho
Escola de Engenharia

Inteligência Artificial
Trabalho Prático 1

Grupo:
Eva Ferreira a103916
Filipa Pinto a103743
Guilherme Fernandes a103819

Docentes: César Analide Freitas Silva Costa Rodrigues
Inês Lucas Amorim Alves

2024/2025

Conteúdo

Lista de Figuras

1 | Introdução

Neste primeiro trabalho prático, será utilizada a Programação em Lógica com a linguagem PROLOG. PROLOG é uma linguagem declarativa que emprega lógica de primeira ordem (Cláusulas de Horn) para modelar o conhecimento de um determinado problema. Através de processos de inferência lógica, ela permite determinar o valor de verdade de uma proposição com base em sua "relação"com os fatos conhecidos.

Para representar Conhecimento Imperfeito, utilizaremos valores nulos e mecanismos específicos de raciocínio, onde os resultados possíveis são "Verdadeiro"(V), "Falso"(F) e "Desconhecido"(D).

O objetivo deste trabalho é desenvolver um sistema que seja capaz de representar e raciocinar sobre um universo de dados, aplicado à área de análise de saúde e exames de marcadores de saúde. As entidades principais a serem analisadas são: Paciente, Consulta, Condição e Marcação.

2 | Representação de Conhecimento e Raciocínio

2.1. Entidades da Base de Conhecimento

Este trabalho irá incidir numa análise à representação do conhecimento e do raciocínio sobre duas perspetivas: o conhecimento perfeito e o conhecimento imperfeito. Estas possuem representações similares, onde a principal diferença está associada aos pressupostos utilizados e, consequentemente, aos mecanismos de ilação e de prova que são utilizados para dar repostas às perguntas efetuadas.

2.1.1. Conhecimento Perfeito

Na representação do conhecimento perfeito, as cláusulas e os predicados pos suem exclusivamente como contradomínio dois valores de prova: "Verdadeiro" e "Falso". Desta forma, são considerados os seguintes pressupostos [1]:

- **Pressuposto do Mundo Fechado** - toda a informação que não se encontra mencionada na Base de Conhecimento é considerada falsa;
- **Pressuposto dos Nomes Únicos** - duas constantes diferentes designam, necessariamente, duas entidades diferentes na Base de Conhecimento;
- **Pressuposto do Domínio Fechado** - não há mais objetos no universo de discurso para além dos definidos na Base de Conhecimento.

2.1.2. Conhecimento Imperfeito

O Conhecimento Imperfeito deixa de assumir que a informação representada é a única válida e que as entidades representadas sejam as únicas existentes no mundo exterior. Deste modo, os pressupostos aceites podem ser diferentes daqueles que são a base da representação de conhecimento perfeito. Desta forma, são considerados os seguintes pressupostos [1]:

- **Pressuposto do 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;

- **Pressuposto dos Nomes Únicos**.

2.1.3. Representação de conhecimento

A Extensão da Programação em Lógica implementa mecanismos que funcionam adequadamente quando confrontados com informação incompleta ou em mudança. Um dos seus objetivos é permitir representar, explicitamente, informação falsa. Para tal, há a necessidade de considerar dois tipos de negação [1]:

- **Negação por falha na prova** - negação utilizada nos programas em Lógica Tradicional, representada pelo termo: "**não**";
- **Negação forte** (ou clássica)- negação utilizada como forma de identificar informação negativa, ou falsa, representada pela conetiva "¬".

Estes surgem da necessidade de distinguir dois tipos de situações que permitem concluir sobre o valor lógico de uma cláusula p:

- (*não(p)*) se aceitarmos que é falso por falta de prova;
- (-p) se é possível provar que é efetivamente falso pela negação forte.

Assim, o conjunto de respostas válidas às questões (q) sobre o programa, está definido para os valores de verdade "Verdadeiro", "Falso" e "Desconhecido", tal que:

- **Verdadeiro** (V): se $\exists x : q(x)$, então consegue-se provar a resposta através de factos positivos;
- **Falso** (F): se $\exists x : \neg q(x)$, então consegue-se provar a resposta através de factos negativos;
- **Desconhecido** (D): se $\exists x : \neg q(x) \wedge \neg(\neg q(x))$, então não é possível provar a resposta através dos factos e dos mecanismos de inferência.

3 Base de Conhecimento

3.1. Entidades da Base de Conhecimento

Nesta secção o principal objetivo é explicar o funcionamento da nossa Base de Conhecimento. A sua representação pode ser caracterizada por diferentes entidades:

- **Paciente**: #Idp, Nome, Data, Sexo, TipoSanguineo;
- **Consulta**: #Idc, Data, #Idp, Idade, Medicacao, Resultado;
- **Condição**: #Idcd, CondicaoMedica, #Idp, Gravidade;
- **Marcação**: #Idm,#Idp,Medico,Data,Hospital,Sala.

3.1.1. Paciente

Na Base de Conhecimento utilizada, o paciente é caracterizado pelo Id do paciente e o seu respetivo nome, data de nascimento, sexo e tipo sanguíneo. Cada paciente é identificado por um Id único de forma a evitar inconformidades e ambiguidade de informação. Desta forma, todos os utentes existentes na nossa Base de Conhecimento são do tipo:

```
119  % Base de Conhecimento
120  %------------------------------- - - - - - - - - - -  -  -  -  -  -
121  % Extensao do predicado paciente: Idpaciente,Nome,Data,Sexo,TipoSanguineo -> {V,F,D}
122
123  % ------- Conhecimento Perfeito Positivo -------%
124
125  paciente(p0,'Guilherme Fernandes', data(08,05,2004), masc,ts(a,+)).
126  paciente(p1,'Filipa Pinto',data(31,01,2004),fem,ts(b,+)).
127  paciente(p2,'Eva Ferreira',data(16,02,2004),fem,ts(o,-)).
128  paciente(p3,'Rita Alves',data(05,08,2002),fem,ts(b,+)).
129  paciente(p4,'Pedro Oliveira',data(07,06,2003),masc,ts(ab,+)).
```

Figura 3.1: Base de Conhecimento Paciente

3.1.2. Consulta

Uma condição é caracterizada pelo seu identificador (#Idc), pela sua data de realização, pelo identificador do utente (#Idp) a quem se refere, bem como a sua idade, medicação a tomar e o resultado da consulta. Para a implementação das datas das consultas, por uma questão de

conseguir satisfazer os requisitos do enunciado, foi considerado um predicado auxiliar data que segue a seguinte nomenclatura:

```
99   % Extensao dos predicados dia, mes e ano: Data, Dia/Mes/Ano -> {V,F}
100
101  dia( data( D,M,A ),D ).
102  mes( data( D,M,A ),M ).
103  ano( data( D,M,A ),A ).
```

Figura 3.2: Predicado auxiliar data

Os factos que representam uma consulta médica foram escritos da seguinte forma:

```
195  % Base de Conhecimento
196  %-------------------------------- - - - - - - - - - - - - - - - -
197  % Extensao do predicado consulta: consulta, data, Idpaciente,idade,medicação,resultado -> {V,F,D}
198
199  % ------- Conhecimento Perfeito Positivo -------%
200
201  consulta(c0, data(02,11,2024), p0, 20, 'Paracetamol','Inconclusivo').
202  consulta(c1, data(20,10,2024), p1, 20, 'Aspirina','Normal').
203  consulta(c2, data(06,11,2024), p2, 20, 'Ibuprofeno','Normal').
```

Figura 3.3: Base de Conhecimento Consulta

3.1.3. Condição

Uma condição caracteriza-se pelo seu próprio Id (#Idc) nome (CondicaoMedica), pelo identidicador do pacientge (#Idp) a quem pertence a condição e a gravidade da mesma. As bases de conhecimento são do tipo:

```
318  % Base de Conhecimento
319  %-------------------------------- - - - - - - - - - - - - - - - -
320  % Extensao do predicado condição: condição, condição médica, Idpaciente, gravidade -> {V,F,D
321
322  % ------- Conhecimento Perfeito Positivo -------%
323
324  condiçao(cd0,'Asma' , p0, 'Baixa' ).
325  condiçao(cd1,'Diabetes', p1, 'Média' ).
326  condiçao(cd2, 'Doença Celíaca', p2, 'Baixa' ).
```

Figura 3.4: Base de Conhecimento Condição

3.1.4. Marcação

Por fim, a marcação refere-se a informações sobre uma consulta. Esta recebe um Id próprio (#Idm) assim como o Id de um paciente (#Idp), recebe também o nome do médico responsável pela consulta, a data na qual esta irá ocorrer, o hospital e a sala da marcação.

```
% Base de Conhecimento
%--------------------------------- - - - - - - - - - -  -  -  -  -  -
% Extensao do predicado marcaçao:  marcaçao,  paciente, medico, data, hospital,sala --> {V,F,D}

% ------- Conhecimento Perfeito Positivo -------%
marcaçao(m0, p0, 'Pedro Silva', data(09,12,2024), 'Hospital de Braga', 328).
marcaçao(m1, p1, 'João Fernandes', data(03,01,2025), 'Hospital da Luz', 201).
marcaçao(m2, p2, 'Catarina Ferreira', data(16,12,2024), 'Hospital S.João', 156).
```

Figura 3.5: Base de Conhecimento Marcação

3.2. Integridade da Base de Conhecimento

De modo a garantir a integridade da Base de Conhecimento foram criados alguns predicados tendo sido necessário , ao longo do desenvolvimento deste trabalho, definir invariantes. Estes invariantes permitem-nos controlar em específico a inserção e a remoção de informações na nossa Base de Conhecimento sendo em Prolog representados da seguinte forma:

- + **Termo:** : Usada quando se pretende adicionar algo à base de conhecimento, que tem de ser verificada após o momento da inserção;
- **- Termo:** :Usada quando se pretende adicionar algo à base de conhecimento, que tem de ser verificada após o momento da remoção.

No Prolog, existem predicados já definidos que permitem adicionar e remover fatos da Base de Conhecimento: assert e retract, respectivamente. Contudo, o uso exclusivo desses predicados não assegura a consistência da base, uma vez que os invariantes definidos não são testados por si só e, assim sendo, é importante a criação de predicados auxiliares que nos tragam garantias que a inserção e a remoção de conhecimento não alterem a consistência da Base de Conhecimento.Esses predicados auxiliares serão discutidos posteriormente neste relatório.

3.2.1. Invariantes

- **Impede a inserção de pacientes com o mesmo Idp**:

```
+paciente(Idp, Nome, Data, Sexo, TipoSanguineo)::
```

```
            (findall(Idp, (paciente(Idp, Nome, Data, Sexo, TipoSanguineo)), S),
             comprimento(S, N), N == 1).
```

- **Impede a inserção de consultas com o mesmo Idc**:

```
 +consulta(Idc, Data, Idp, Idade, Medicacao, Resultado)::
            (findall(Idc, (consulta(Idc, Data, IdP, Idade, Medicacao, Resultado)),
             comprimento(S, N), N == 1).
```

- **Impede a inserção de marcações com o mesmo Idm**:

```
+marcacao(Idm, Idp, Medico, Data, Hospital, Sala)::
            (findall(Idm, (marcacao(Idm, Idp, Medico, Data, Hospital, Sala)), S),
             comprimento(S, N), N == 1).
```

- **Impede a inserção de condições com o mesmo Idcd**:

```
+condicao(Idcd, CondicaoMedica, Idp, Gravidade)::
            (findall(Idcd, (condicao(Idcd, CondicaoMedica, Idp, Gravidade)), S),
             comprimento(S, N), N == 1).
```

- **Impede a remoção de um paciente associado a uma consulta**:

```
 -paciente(Idp, Nome, Data, Sexo, TipoSanguineo)::
            (findall(Idp, consulta(Idc, Data, Idp, Idade, Medicacao, Resultado), S)
             comprimento(S, N), N == 0).
```

- **Impede a inserção de condições médicas duplicadas para o mesmo paciente**:

```
+condicao(Idcd, CondicaoMedica, Idp, Gravidade)::
            (findall((Idp, CondicaoMedica), (condicao(Idcd, CondicaoMedica, Idp, Gr
             comprimento(S, N), N == 1).
```

- **Impede a inserção de pacientes que ainda não nasceram (ano menor que 2025)**:

```
+paciente(Idp, Nome, Data, Sexo, TipoSanguineo)::
    (findall(Idp, (paciente(Idp, Nome, data(D, M, A), Sexo, TipoSanguineo),
        data(D, M, A) < data(09, 11, 2024)), S), comprimento(S, N), N == 0).
```

- **Impede que seja inserido conhecimento negativo de um paciente quando há conhecimento perfeito positivo para esse mesmo paciente**:

```
-paciente(Idp, _, _, _, _)::
            (findall(Idp, paciente(Idp, _, _, _, _), S), comprimento(S, N), N == 0)
```

- **Invariante para não inserir um paciente quando há conhecimento perfeito negativo para esse paciente**:

```
+paciente(Idp, _, _, _, _)::
            (findall(Idp, (-paciente(Idp, _, _, _, _)), S), comprimento(S, N), N ==
```

- **Não permitir inserir consultas com Idp não registados**:

```
+consulta(Idc, Data, Idp, Idade, Medicacao, Resultado)::
            (paciente(Idp, _, _, _, _)).
```

- **Garante que a gravidade da condição não seja nula ou vazia**:

```
+condicao(Idcd, CondicaoMedica, Idp, Gravidade)::
            (Gravidade \neq '').
```

- **Garante que o resultado da consulta não seja nulo ou vazio**:

```
+consulta(Idc, Data, Idp, Idade, Medicacao, Resultado)::
            (Resultado \neq '').
```

- **Garante que uma condição médica só pode ser registrada se o paciente correspondente existir**:

```
+(condicao(_, _, Idp, _))::
            (findall(Idp, paciente(Idp, _, _, _, _), S), comprimento(S, N), N > 0).
```

- **Garante que não sejam feitas marcações para pacientes que não estejam registados**:

```
+marcacao(Idm, Idp, Medico, Data, Hospital, Sala)::
            (paciente(Idp, _, _, _, _)).
```

- **Não permite a inserção de sexos sem ser masculino, feminino ou desconhecido**:

```
+paciente(Idp, Nome, Data, Sexo, TipoSanguineo)::
            (pertence(Sexo, [masc, fem, desconhecido])).
```

3.3. Valores Nulos

É possível, por vezes, encontrar informação incompleta, mas dentro desta ter-se diferentes tipos de desconhecimento. Esta identificação de valores é feita recorrendo ao conceito de valores nulos, que surgem como uma estratégia para fazer a distinção entre respostas a questões que devem ser concretizadas como "conhecidas" ("verdadeiras" ou "falsas') ou respostas concretizadas como "desconhecidas"[1].

As respostas podem ser desconhecidas por três razões: por serem relativas a uma questão de onde não se conhece efetivamente nenhum valor ; por se saber que é válida dentro de um conjunto determinado de valores (valores desconhecidos, mas de um conjunto finito de valores); por fim, por serem relativas a uma questão à qual não se permite haver resposta, ou seja, a resposta é interdita (valores não permitidos) [1].

Para conseguir a representação destes valores, apresentados de seguida, foram criados os

seguintes predicados:

- **excecao(Termo)**- para representar um caso de exceção;
- **nulo(Termo)**- para representar como nulo qualquer unificação com Termo.

Dado o contexto em que este problema está a ser desenvolvido, é necessário especificar em que medida se pode considerar um dado Termo como sendo efetivamente "falso", quando implementado em Prolog. A seguinte formalização permite identificar, para um dado predicado p (que pode representar qualquer uma das entidades da base de conhecimento adotada), em que situação se pode afirmar que é "falso" *p(x)*.

$$\neg p(x) \leftarrow \text{no}\,(p(x)) \wedge \text{no}\,(excecao\,(p(x))) \tag{3.1}$$

3.3.1. Tipo Desconhecido/Incerto

Os valores nulos do tipo desconhecido permitem representar valores desconhecidos sem que haja a especificação de um conjunto de valores. A identificaçãode valores deste tipo é feita através da introdução de uma situação de exceção correspondente a uma condição anómala.Se p é um predicado, x umargumento de p e v um valor nulo, tem-se a seguinte representação de valor nulo do tipo desconhecido:

$$excecao\,(p(x)) \leftarrow p(v) \tag{3.2}$$

3.3.2. TipoDesconhecidodeumdadoconjuntodevalores/Impreciso

A diferença entre o valor nulo do tipo desconhecido apresentado anteriormente e o valor nulo que se pretende apresentar agora, desconhecido mas de um conjunto finito e determinado de valores, está precisamente no facto de este último valor nulo representar um (ou mais) valores de um conjunto finito de valores bem determinados: só não é conhecido, especificamente, qual dos valores concretizará a questão [1].

Seja p um predicado, x um argumento de p e *v1* e *v2* valores nulos. Pode-se representar a possibilidade de p não ser "falso" para x $\in \{v1, v2\}$.

Para tal recorremos à sequência de exceções:

$$excecao\,(p(v1)) \tag{3.3}$$

$$excecao\,(p(v2)) \tag{3.4}$$

3.3.3. Tipo não permitido/Interdito

Este terceiro tipo de valores nulos caracterizam aqueles que não se pretende que surjam na base de conhecimento, o valor que representam não é permitido especificar ou conhecer (é interdito) [1]. Se textitp é um predicado, textitx um argumento de textitp e textitv um valor nulo, tem-se a seguinte representação de valor nulo do tipo interdito, com uma exceção que representa a situação anómala a representar:

$$excecao\,(p(x)) \leftarrow p(v) \tag{3.5}$$

$$nulo(v) \tag{3.6}$$

Deve ainda ser implementado um mecanismo que impeça a posterior inclusão de valores que violem a condição imposta pelo valor nulo não permitido. Neste exercício prático, este mecanismo foi implementado como um invariante, que se pode verificar em secções posteriores deste relatório.

3.4. Conhecimento Perfeito

O conhecimento perfeito, é aquele sobre o qual existe informação completa. Este pode ser positivo ou negativo.

3.4.1. Conhecimento Perfeito Positivo

O conhecimento perfeito positivo refere-se ao conjunto de factos que sabemos com certeza que são verdadeiros. Neste contexto, uma afirmação é dita de conhecimento perfeito positivo se a sua verdade é garantida, e não há incerteza associada a ela. O conhecimento perfeito positivo é importante em sistemas de Prolog porque ajuda a criar uma base de conhecimento sólida, onde todas as informações são verdadeiras sem espaço para ambiguidade ou incerteza.

No caso da representação do conhecimento perfeito positivo foi definida a informação da seguinte forma:

Paciente

A informação relativa aos pacientes é considerada tendo por base factos, tais como:

```
paciente(p0,'Guilherme Fernandes', data(08,05,2004), masc,ts(a,+)).
paciente(p1,'Filipa Pinto',data(31,01,2004),fem,ts(b,+)).
paciente(p2,'Eva Ferreira',data(16,02,2004),fem,ts(o,-)).
paciente(p3,'Rita Alves',data(05,08,2002),fem,ts(b,+)).
paciente(p4,'Pedro Oliveira',data(07,06,2003),masc,ts(ab,+)).
```

Figura 3.6: Conhecimento perfeito positivo paciente

Consulta

Recorreu-se ao predicado auxiliar data,apresentado na secção Predicados Auxiliares, para representar a data em que ocorreu uma consulta. A representação da informação relativa às consultas foi feita através de factos, tais como:

```
consulta(c0, data(02,11,2024), p0, 20, 'Paracetamol','Inconclusivo').
consulta(c1, data(20,10,2024), p1, 20, 'Aspirina','Normal').
consulta(c2, data(06,11,2024), p2, 20, 'Ibuprofeno','Normal').
```

Figura 3.7: Conhecimento perfeito positivo consulta

Condição

Para as condições temos a seguinte representação:

```
condiçao(cd0,'Asma' , p0, 'Baixa' ).
condiçao(cd1,'Diabetes', p1, 'Média' ).
condiçao(cd2, 'Doença Celíaca', p2, 'Baixa' ).
```

Figura 3.8: Conhecimento perfeito positivo condição

Marcação

A informação que representa uma marcação é a seguinte:

```
marcaçao(m0, p0, 'Pedro Silva', data(09,12,2024), 'Hospital de Braga', 328).
marcaçao(m1, p1, 'João Fernandes', data(03,01,2025), 'Hospital da Luz', 201).
marcaçao(m2, p2, 'Catarina Ferreira', data(16,12,2024), 'Hospital S.João', 156).
```

Figura 3.9: Conhecimento perfeito positivo marcação

3.4.2. Conhecimento Perfeito Negativo

O conhecimento perfeito negativo em Prolog refere-se ao conjunto de factos que sabemos com certeza que são falsos. Para representar o conhecimento negativo de forma explícita, foram definidos predicados auxiliares para cada entidade da Base de Conhecimento caracterizados pelo *-functor*, representando a negação forte do predicado.

Paciente

A informação negativa relativa aos pacientes pode ser apresentada da seguinte forma:

- O paciente Rui Soares com Idpaciente p5, nascido em 24/08/2001 nega ser do sexo masculino.

```
-paciente(p5,'Rui Soares',data(24,08,2001),masc,ts(b,+)).
```

Figura 3.10: Conhecimento perfeito negativo paciente 1

- O paciente com o Idpaciente p6 , chamada Leonor Vilas-Boas nega ter nascido no dia 26/08/1995

```
-paciente(p6,'Leonor Vilas-Boas',data(26,08,1995),fem,ts(a,+)).
```

Figura 3.11: Conhecimento perfeito negativo paciente 2

Consulta

A representação da informação negativa relativa às consultas médicos foi feita através da negação de factos, tais que podemos representar da seguinte forma:

- A paciente Rita Alves com Idpaciente p3 nega poder tomar Ibuprofeno.

```
-consulta(c3, data(01,11,2024), p3, 22, 'Ibuprofeno','Anormal').
```

Figura 3.12: Conhecimento perfeito negativo consulta 1

- O paciente com o Idpaciente p4 nega ter 23 anos.

```
-consulta(c4, data(08,11,2024), p4, 23,'Paracetamol','Normal').
```

Figura 3.13: Conhecimento perfeito negativo consulta 2

Condição

Para representarmos conhecimento negativo em relação ao marcador temos a seguinte representação:

- A paciente Rita Alves com Idpaciente p3 nega ter hipertensão.

```
-condiçao(cd3, 'Hipertensão', p3, 'Média' ).
```

Figura 3.14: Conhecimento perfeito negativo condição

Marcação

A representação da informação negativa relativa às marcações foi feita através da negação de factos, tais que podemos representar da seguinte forma:

- A marcação com Idm m3, com a data 30/11/2024 nega ser no Hospital de Braga.

```
-marcaçao(m3, p4, 'José Figueiredo', data(30,11,2024), 'Hospital de Braga', 123).
```

Figura 3.15: Conhecimento perfeito negativo marcação 1

- A marcação com o IdMarcaçao m4, com o médico Jorge Sousa nega ser na sala 205.

```
-marcaçao(m4, p3, 'Jorge Sousa', data(02,12,2024), 'Hospital da Luz', 205).
```

Figura 3.16: Conhecimento perfeito negativo marcação 2

3.5. Conhecimento Imperfeito

O conhecimento imperfeito pode ser de três tipos:

- **Incerto**

- **Impreciso**
- **Interdito**

3.5.1. Conhecimento Imperfeito Incerto

Este tipo de conhecimento será representado utilizando os valores nulos do tipo "desconhecido", ou seja, valores dos quais desconhecemos a sua concretização e não possuímos informação sobre o conjunto de valores.

Paciente

- Desconhece-se a data de nascimento do paciente com o Idpaciente p7, de nome Alberto Malheiro e do sexo masculino.

```
paciente(p7,'Alberto Malheiro',desconhecido,masc,ts(b,+)).
excecao(paciente( Idp,Nome,Data,Sexo,TipoSanguineo) ) :-
    paciente(Idp,Nome,desconhecido,Sexo,TipoSanguineo).
```

Figura 3.17: Conhecimento imperfeito incerto paciente 1

- Desconhece-se o sexo do paciente com o Idpaciente p11.

```
paciente(p11,'Antonio Rodrigues',data(16,07,1995),desconhecido,ts(o,+)).
excecao(paciente( Idp,Nome,Data,Sexo,TipoSanguineo) ) :-
    paciente(Idp,Nome,Data,desconhecido,TipoSanguineo).
```

Figura 3.18: Conhecimento imperfeito incerto paciente 2

Consulta

- Desconhece-se o resultado do paciente com o Idpaciente p5, com 23 anos.

```
consulta(c5,data(26,10,2024),p5, 23 ,'Paracetamol',desconhecido).
excecao(consulta( Idc,Data, Idp ,Idade, Medicacao,Resultado) ) :-
    consulta(Idc,Data, Idp ,Idade, Medicacao,Resultado).
```

Figura 3.19: Conhecimento imperfeito incerto consulta

Condição

- Desconhece-se a gravidade da condição do paciente com o Idpaciente 4, com Epilepsia.

```
condiçao(cd4, 'Epilepsia', p4, desconhecido).
excecao(condiçao( Idcd, CondicaoMedica, Idp , Gravidade) ) :-
    condiçao(Idcd, CondicaoMedica, Idp ,Gravidade).
```

Figura 3.20: Conhecimento imperfeito incerto condição

Marcação

- Desconhece-se a sala da marcação de Idmarcaçao m5, com o paciente p5 na data 31/01/2025.

```
marcaçao(m5, p5, 'Catarina Ferreira',data(31,01,2025),'Hospital de Braga', desconhecido ).
excecao(marcaçao( Idm,Idp,Medico,Data,Hospital,Sala) ) :-
   marcaçao( Idm,Idp,Medico,Data,Hospital,desconhecido).
```

Figura 3.21: Conhecimento imperfeito incerto marcação

3.5.2. Conhecimento Imperfeito Impreciso

Para representar o conhecimento imperfeito impreciso, serão utilizados valores nulos do tipo desconhecido, tendo por base um conjunto possíveis de valores.

Paciente

- A Judite Pereira, do sexo feminino, com Idpaciente p8 não se sabe se nasceu no dia 20/07/2009 ou no dia 25/07/2009.

```
excecao(paciente(p8,'Judite Pereira',data(20,07,2009),fem,ts(ab,+))).
excecao(paciente(p8,'Judite Pereira',data(25,07,2009),fem,ts(ab,+))).
```

Figura 3.22: Conhecimento imperfeito impreciso paciente 1

- Não se sabe ao certo o dia de nascimento do Antonio, com o Idp 9 e do sexo masculino. Apenas se sabe que nasceu entre o dia 7 e o dia 14 .

```
excecao(paciente(p9,'Antonio Vieira',data(D,09,1983),masc,(b,-))) :-
    D >= 07, D =< 14.
```

Figura 3.23: Conhecimento imperfeito impreciso paciente 2

Consulta

- O paciente com Idpaciente p6 não se sabe se tem de tomar Paracetamol ou Iboprufeno.

```
excecao( consulta(c6,data(17,10,2024),p6, 29 ,'Paracetamol','Anormal' ) ).
excecao( consulta(c6,data(17,10,2024),p6, 29 ,'Ibuprofeno','Anormal' ) ).
```

Figura 3.24: Conhecimento imperfeito impreciso consulta

- Não se sabe ao certo a idade do paciente com o Idp p7. Apenas se sabe que estará entre os 23 e os 26 anos.

```
excecao( consulta(c7,data(24,10,2024),p7, I ,'Ibuprofeno','Anormal' ) ) :-
    I >= 23, I=< 26.
```

Figura 3.25: Conhecimento imperfeito impreciso consulta 2

Condição

- O paciente com Idp 5 não se sabe se a gravidade da sua condição é 'Baixa ou 'Média'.

```
excecao( condiçao(cd5, 'Hipertensão', p5, 'Baixa' ) ).
excecao( condiçao(cd5, 'Hipertensão', p5, 'Média' ) ).
```

Figura 3.26: Conhecimento imperfeito impreciso condição

Marcação

- A marcação de Idmarcaçao m6, relativa ao paciente p6, no Hospital da Luz, não se sabe se será com a médica Ana Castro ou com o médico João Fernandes

```
excecao(marcaçao(m6, p6,'Ana Castro', data(23,02,2025), 'Hospital da Luz', 433 )).
excecao(marcaçao(m6, p6,'João Fernandes', data(23,02,2025), 'Hospital da Luz', 433 )).
```

Figura 3.27: Conhecimento imperfeito impreciso marcação

- Não se sabe ao certo o dia da marcaçao de Idmarcaçao m7, com o médico Francisco Cunha no Hospital S.João. Apenas se sabe que será entre o dia 8 e o dia 16.

```
excecao( marcaçao(m7, p7,'Francisco Cunha', data(D,01,2025), 'Hospital S.João', 120 ) ) :-
    D >= 08, D =< 16.
```

Figura 3.28: Conhecimento imperfeito impreciso marcação 2

3.5.3. Conhecimento Imperfeito Interdito

Conhecimento imperfeito interdito recorre a valores nulos do tipo não permitido,ou seja, valores dos quais desconhecemos a sua concretização e que se pretende que não surjam na base de conhecimento.

Paciente

- O utente com o Idp p10, de nome Alberto Costa, nascido no dia 18/06/1939 exige que não se saiba o seu sexo.

```
paciente( p10,'Alberto Costa',data(18,06,1939),desconhecido,ts(a,+)).
excecao( paciente( IDP,N,DN,S,TS) ) :-
    paciente( IDP,N,DN,desconhecido,TS ).
nulo( desconhecido ).
+paciente( IDP,N,DN,S,TS ) :: ( solucoes( X,( paciente( p10,_,_,X,_),nao( nulo( X ) ) ),S),
                                 comprimento( S,L ),
                                 L == 0 ).
```

Figura 3.29: Conhecimento imperfeito interdito paciente 1

Consulta

- O utente com o Idp p8 não quer que se saiba o seu resultado

```
consulta( c8,data(02,10,2024),p8, 15 ,'Ibuprofeno',desconhecido).
excecao( consulta( IDC,D,IDP,I,M,R ) ) :-
    consulta( IDC,D,IDP,I,M,desconhecido).
nulo(desconhecido).
+consulta( IDC,D,IDP,I,M,R ) :: ( solucoes( X,( consulta( c8,_,_,_,_,X ),nao( nulo( X ) ) ),R ),
                                  comprimento(R,L ),
                                  L == 0).
```

Figura 3.30: Conhecimento imperfeito interdito consulta

Condição

- O paciente com o Idp 6, exige que não se saiba a sua condição.

```
condiçao(cd6, desconhecido, p6, 'Baixa' ).
excecao( condiçao( IDCD, CM, IDP, G ) ) :-
    condiçao( IDCD, CM, IDP, desconhecido).
nulo( desconhecido ).
+condiçao( IDCD, CM, IDP, G ) :: ( solucoes( X,( condiçao( cd6,X,_,_ ),nao( nulo( X ) ) ),R ),
                                   comprimento(R,L ),
                                  L == 0)
```

Figura 3.31: Conhecimento imperfeito interdito condição

Marcação

- A marcaçao m8 com o paciente p8 no 'Hospital de Braga' no dia 08/12/2024, exige que não se saiba a sala onde se vai realizar.

```
marcaçao( m8, p8, 'Alberto Rodrigues',data(08,12,2024), 'Hospital de Braga', desconhecido).
excecao(marcaçao( IDM,IDP,N,DC,H,S)) :-
   marcaçao(IDM,IDP,N,DC,H,S,desconhecido).
nulo(desconhecido).
+paciente( IDM,IDP,N,DC,H,S) :: ( solucoes( X,( marcaçao( m8,_,_,_,_,X ),nao( nulo( X ) ) ),S ),
                                  comprimento( S,L ),
                                  L == 0 ).
```

Figura 3.32: Conhecimento imperfeito interdito marcação

4 | Predicados auxiliares

Nesta secção são apresentados alguns predicados que foram úteis para a imple mentação das funcionalidades do sistema. Teremos assim uma breve apresentação dos predicados usados, bem como, uma breve explicação do seu funcionamento.

Predicado "Data"

Este predicado surge de forma a facilitar a utilizacação e manipulção da data de um ato. Torna-se mais fácil pois conseguimos variar as variáveis em separado e não torna-las completamente dependentes umas das outras.

```
% Extensao dos predicados dia, mes e ano: Data, Dia/Mes/Ano -> {V,F}
dia( data( D,M,A ),D ).
mes( data( D,M,A ),M ).
ano( data( D,M,A ),A ).
```

Figura 4.1: Predicado Data

Predicado "Soluções"

O predicado solucões trata-se simplesmente de uma implementação que tem a mesma funcionalidade que o predicado *findall*, já existente no Prolog.Este é usado para obter todos os resultados que satisfazem uma dada condição. Em Z está a lista de resultados de X que satisfazem as condições em Y.

```
% Extensao do predicado solucoes
solucoes( X,Y,Z ) :-
    findall( X,Y,Z ).
```

Figura 4.2: Predicado Soluções

Predicado "Teste"

Tem-se também o predicado teste que recebe como argumento uma lista de in variantes e percorre toda essa lista testando cada um dos seus elementos, conjugando-os. Se algum falhar, o predicado teste falha no seu todo.

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

Figura 4.3: Predicado teste

Predicado "Não"

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

Figura 4.4: Predicado Não

Evolução de uma lista de termos

```
% Extensão do predicado que permite a evolução de uma lista de Termos

insertAll( [] ).
insertAll( [H|T] ) :-
    assert( H ),
    insertAll( T ).
```

Figura 4.5: Evolução lista de termos

Involução de uma lista de termos

```
% Extensão do predicado que permite a involução de uma lista de Termos
removeAll( [] ).
removeAll( [H|T] ) :-
    retract( H ),
    removeAll( T ).
```

Figura 4.6: Involução lista de termos

Predicado "Comprimento"

```
% Extensao do predicado comprimento: lista,N -> {V,F}
comprimento([],0).
comprimento([_|L],N):-
    comprimento(L,N1),
    N is N1 + 1.
```

Figura 4.7: Predicado comprimento

5 | Adição e Alteração de Conhecimento

Nesta secção o principal objetivo é mostrar como é adicionado, removido ou alterado conhecimento da nossa Base de Conhecimento.

Adição de conhecimento

No nosso projeto pode-se adicionar um novo paciente, uma consulta, uma nova condição ou uma nova marcação. Todas estas seguem o seguinte exemplo:

```
% Adicionar um novo paciente
evolucaoConhecimento(paciente(Idp, Nome, data(D, M, A), Sexo, TipoSanguineo)) :-
    \+ paciente(Idp, Nome, data(D, M, A), Sexo, TipoSanguineo),
    evolucao(paciente(Idp, Nome, data(D, M, A), Sexo, TipoSanguineo)).

evolucao(Paciente) :-
    assert(Paciente).
```

Figura 5.1: Adicionar novo paciente

Alteração de conhecimento

Foram definidas algumas alterações possíveis a serem feitas à base de conhecimento, entre estas:

- Alterar o sexo de um paciente
- Alterar a medicação da consulta de um dado paciente
- Alterar resultado da consulta de um dado paciente
- Alterar a idade de um paciente
- Alterar a gravidade da condição
- Alterar a condição médica de um paciente
- Alterar o médico da marcação
- Alterar o Hospital da marcação
- Alterar a Sala da marcação

Veremos o exemplo de alterar a condição médica de um paciente:

```
% Alterar condição médica de um paciente
alterar_condicao_medica(Idcd,Nova_condicao_medica):-condicao( Idcd,_, Idp , Gravidade),
                                                    involucao(condicao( Idcd,_, Idp , Gravidade)).
                                                    evolucao(condicao( Idcd, Nova_condicao_medica, Idp , Gravidade)).
```

Figura 5.2: Alterar condição médica

Remoção de conhecimento - da mesma forma que se pode adicionar um novo paciente, uma consulta, uma nova condição ou uma nova marcação, pode-se também remover, como mostra o exemplo:

```
% Remover uma consulta
remove_consulta(Idc, Data, Idp, Idade, Medicacao, Resultado) :-
    consulta(Idc, Data, Idp, Idade, Medicacao, Resultado),
    retract(consulta(Idc, Data, Idp, Idade, Medicacao, Resultado)).
```

Figura 5.3: Remoção de consulta

6 Inserção e Evolução do conhecimento

6.1. Evolução de conhecimento perfeito negativo para conhecimento perfeito positivo

```
% Transformacao de um não paciente para um paciente
passa_a_paciente( Idp,Nome,Data,Sexo,TipoSanguineo):-
    involucao(- paciente(Idp,Nome,Data,Sexo,TipoSanguineo)),
    evolucao( paciente(Idp,Nome,Data,Sexo,TipoSanguineo)).
```

Figura 6.1: Evolução do conhecimento

6.2. Evolução de conhecimento perfeito positivo para conhecimento perfeito negativo

```
% Transformacao de um paciente para um não paciente
negar_paciente( Idp,Nome,Data,Sexo,TipoSanguineo):-
    involucao( paciente(Idp,Nome,Data,Sexo,TipoSanguineo)),
    evolucao(-paciente(Idp,Nome,Data,Sexo,TipoSanguineo)).
```

Figura 6.2

6.3. Evolução de conhecimento imperfeito incerto para conhecimento perfeito

```
% Transformacao de um conhecimento Imperfeito Incerto para Prefeito
evolucao_IncertoPerf_paciente(Termo_atual,Termo_antigo):-
    si(Termo_antigo, verdadeiro),
    Termo_antigo = paciente( Idp,Nome,Data,Sexo,TipoSanguineo),
    Termo_atual = paciente( Idp,NovoNome,NovaData,NovoSexo,TipoSanguineo),
    (Nome == desconhecido ; Nome==NovoNome),
    (Data==desconhecido;Data==NovaData),
    (Sexo == desconhecido ; Sexo == NovoSexo),
    remocao(Termo_antigo),
    insercao(Termo_atual).
```

Figura 6.3: Conhecimento imperfeito incerto para perfeito

6.4. Evolução de conhecimento interdito incerto para conhecimento perfeito

```
% Transformacao de um conhecimento Imperfeito Interdito para Prefeito
evolucao_InterditoPerf_paciente(Termo_atual, Termo_antigo) :-
    si(Termo_antigo, verdadeiro),
    Termo_antigo = paciente( Idp,Nome,Data,Sexo,TipoSanguineo),
    Termo_atual = paciente( Idp,NovoNome,NovaData,NovoSexo,TipoSanguineo),
    (Nome == desconhecido, NovoNome \= desconhecido ; Nome == NovoNome),
    (Data == desconhecido, NovaData \= desconhecido ; Data == NovaData),
    (Sexo == desconhecido, NovoSexo \= desconhecido ; Sexo == NovoSexo),
    remocao(Termo_antigo),
    insercao(Termo_atual).
```

Figura 6.4: Conhecimento imperfeito interdito para perfeito

6.5. Evolução de conhecimento impreciso incerto para conhecimento perfeito

```
% Transformacao de um conhecimento Imprefeito Impreciso para Prefeito
evolucao_ImprecisoPerf_paciente(Termo_atual, Termo_antigo1, Termo_antigo2) :-
    si(Termo_antigo1, desconhecido),
    si(Termo_antigo2, desconhecido),
    Termo_antigo1 = paciente( Idp,Nome1,Data1,Sexo1,TipoSanguineo),
    Termo_antigo2 = paciente( Idp,Nome2,Data2,Sexo2,TipoSanguineo),
    Termo_atual = paciente( Idp,NovoNome,NovaData,NovoSexo,TipoSanguineo),
    ((Nome1 == Nome2, Nome1 == NovoNome) ; (Nome1 \= Nome2)),
    ((Data1 == Data2, Data1 == NovaData) ; (Data1 \= Data2)),
    ((Sexo1 == Sexo2, Sexo1 == NovoSexo) ; (Sexo1 \= Sexo2)),
    remocao(excecao(Termo_antigo1)),
    remocao(excecao(Termo_antigo2)),
    insercao(Termo_atual).
```

Figura 6.5: Conhecimento imperfeito impreciso para perfeito

7 | Interpretação de Valores Nulos

Após terem sido abordados os diferentes tipos de valores nulos e o respetivo conhecimento associado a cada um, procede-se à implementação do mecanismo de inferência destes valores.

Desta forma, criou-se um novo predicado: *si* habilitado a representar conhecimento no contexto de Programação em Lógica Estendida. Este é capaz de interpretar questões e responder com Verdadeiro (V), Falso (F) ou Desconhecido (D).

$$\textit{Extensão do predicado demo: Questao, Resposta} \rightarrow \{V, F\}$$

Tal como se encontra representado acima, este predicado recebe dois parâme tros como argumento: o primeiro é referente à questão a colocar ao sistema de inferência e o segundo representa a resposta a essa questão. O contradomínio do predicado demo terá como valores (V) ou (F) e terá como resposta os valores já apresentados: (V), (F) ou (D).

Assim, a extensão do predicado *si* na linguagem Prolog é a seguinte:

```
si( Q,verdadeiro ) :-
    Q.
si( Q,falso ) :-
    -Q.
si(Q, desconhecido) :-
    \+ Q,
    \+ -Q.
```

Figura 7.1: Si

A primeira cláusula indica que a resposta de uma determinada questão terá o valor de '**Verdadeiro**' se existirem provas na Base de Conhecimento que afirmam que a questão (Q) é verdadeira.

A segunda cláusula, pelo mesmo método, afirma que uma questão (Q) é falsa se existir informação explícita na Base de Conhecimento que mostre que a questão é falsa, ou quando não existir nenhuma exceção que prove que se trata de conhecimento imperfeito. Assim, admite-se que uma questão tomará valor de '**Falso**' se for possível encontrar uma prova de-Q.

Por fim, se não existir provas que a questão Q é verdadeira ou falsa, então a resposta terá o valor de '**Desconhecido**'.

Assim, o predicado *si* pode ser utilizado para dar resposta às questões colocadas ao sistema de inferência, tendo em consideração modificações da Base de Conhecimento, relacionadas com a existência de conhecimento imperfeito.

7.1. Adaptação do interpretador aos operadores lógicos

Dado ser necessário relacionar dois predicados diferentes, criou-se dois novos programas denominados *siC* e *siD*:

Extensao do predicado siC: Questao1 e Questao2, Resposta -> {V,F,D}

Extensao do predicado siD: Questao1 ou Questao2, Resposta -> {V,F,D}

Este recebe no primeiro argumento um tuplo de questões (Questão 1, Ques tão 2) separadas pelo operador lógico que lhes é aplicado, sendo que este pode ser uma conjunção (and) ou disjunção (or).

Em seguida é apresentada a tabela de valores lógicos que relaciona o par de questões efetuadas que o interpretador deve receber e a resposta que deve ser obtida.

7.1.1. Conjunção de predicados

$Q1$	$Q2$	$Q1 \wedge Q2$
$\mathbb{V}$	$\mathbb{V}$	$\mathbb{V}$
$\mathbb{V}$	$\mathbb{F}$	$\mathbb{F}$
$\mathbb{V}$	$\mathbb{D}$	$\mathbb{D}$
$\mathbb{F}$	$\mathbb{V}$	$\mathbb{F}$
$\mathbb{F}$	$\mathbb{F}$	$\mathbb{F}$
$\mathbb{F}$	$\mathbb{D}$	$\mathbb{F}$
$\mathbb{D}$	$\mathbb{V}$	$\mathbb{D}$
$\mathbb{D}$	$\mathbb{F}$	$\mathbb{F}$
$\mathbb{D}$	$\mathbb{D}$	$\mathbb{D}$

Figura 7.2: Conjunção de predicados

7.1.2. Disjunção de predicados

$Q1$	$Q2$	$Q1 \vee Q2$
$\mathbb{V}$	$\mathbb{V}$	$\mathbb{V}$
$\mathbb{V}$	$\mathbb{F}$	$\mathbb{V}$
$\mathbb{V}$	$\mathbb{D}$	$\mathbb{V}$
$\mathbb{F}$	$\mathbb{V}$	$\mathbb{V}$
$\mathbb{F}$	$\mathbb{F}$	$\mathbb{F}$
$\mathbb{F}$	$\mathbb{D}$	$\mathbb{D}$
$\mathbb{D}$	$\mathbb{V}$	$\mathbb{V}$
$\mathbb{D}$	$\mathbb{F}$	$\mathbb{D}$
$\mathbb{D}$	$\mathbb{D}$	$\mathbb{D}$

Figura 7.3: Disjunção de predicados

7.2. Adaptação do interpretador a uma lista de questões

De modo a responder em simultâneo a um conjunto de questões, desenvolveu se um programa que recebe como parâmetros duas listas: a lista de questões a testar e a lista onde serão guardadas as respostas às questões colocadas. Para tal, criou-se um predicado designado *siL* que, recorrendo ao predicado *si* ,testa cada uma das questões que constituem a primeira lista e coloca a respetiva resposta na lista criada para esse efeito. A resposta a cada questão será, como já visto, um dos seguintes valores: (V), (F) ou (D).

Extensao do meta-predicado siL: [],[] -> {V,F,D}

```
siL([],[]).
siL([Questao|L],[Resposta|S]):-
    siL(L,S),
    si(Questao,Resposta).
```

Figura 7.4: siL

8 | Apresentação dos dados da Base de Conhecimentos

Para a apresentação dos dados da Base de Conhecimento, foram criadas uma série de funções que permitem:

- Ver todos os pacientes/marcações/condições/consultas, como no exemplo:

  ```
  ver_todos_pacientes:-listing(paciente( Idp,Nome,Data,Sexo,TipoSanguineo)).
  ```

- Ver todos os pacientes/marcações/condições/consultas, como no exemplo:

  ```
  ver_todos_pacientes:-listing(paciente( Idp,Nome,Data,Sexo,TipoSanguineo)).
  ```

9 | Conclusão

Como já referido anteriormente, o objetivo deste trabalho prático consistiu no desenvolvimento de uma Base de Conhecimento, para que esta pudesse representar, não só o conhecimento perfeito, mas também o conhecimento imperfeito, podendo este ser impreciso, incerto ou interdito.

Ao longo da realização deste exercício, conseguiu-se compreender e organizar toda a informação presente na Base de Conhecimento, assim como gerar todos os predicados de forma natural.

Foram criados dois importantes predicados: o si, o siC e o siD que associa dois predicados, retornando um valor de verdade. Para tal, os operadores lógicos de junção de predicados que se utilizaram foram a conjunção e a disjunção.

Para inserir novos dados e evoluir a Base de Conhecimento, implementou-se alguns predicados como o evolucao e o evoluirConhecimento.

Desta forma, supõe-se que o objetivo do exercício foi cumprido, visto que foi desenvolvido um sistema de representação de conhecimento e raciocínio capaz de demonstrar as funcionalidades subjacentes à programação em lógica estendida e ao tratamento de conhecimento imperfeito.

10 | Bibliografia

[1] CÉSAR ANALIDE, J. N. Antropopatia em entidades virtuais. WORKSHOP OF THESIS AND DISSERTATIONS IN ARTIFICIAL INTELLIGENCE : proceedings, 1, Porto de Galinhas, Recife, Brazil (2002).