Universidade do Minho
Mestrado Integrado em Engenharia Informática

Unidade Curricular de Sistemas de Representação de Conhecimento e Raciocínio

Ano Letivo de 2017/2018

2.º Exercício de Grupo

(Programação em Lógica Estendida e Conhecimento Imperfeito)

Diogo Emanuel da Silva Nogueira (a78957)

Fábio Quintas Gonçalves (a78793)

Sarah Tifany da Silva (a76867)

Grupo 8

Abril, 2018

Resumo

O presente relatório elaborado no âmbito da unidade curricular Sistemas de Representação de Conhecimento e Raciocínio visa a fundamentação e posterior resolução do exercício proposto e cujo tema diz respeito à Programação em Lógica Estendida e Conhecimento Imperfeito.

Neste, iremos utilizar o Sistema de Representação de Conhecimento e Raciocínio e a Base de Conhecimento (relativos aos cuidados de saúde) desenvolvidas no exercício anterior, demonstrando todas as funcionalidades subjacentes à programação em lógica e à representação de conhecimento imperfeito por recorrência à temática dos valores nulos.

Assim, ao longo deste relatório apresentaremos todo o procedimento que levou à resolução deste exercício bem como as várias funcionalidades que tiveram de ser alteradas/reaproveitadas da fase anterior de modo a se cumprir todos os requisitos necessários desta etapa.

Índice

1. Introdução 4
2. Preliminares 5
2.1.1. Base de Dados 6
2.1.2. Sistema de Representação de Conhecimento 7
3. Descrição do Trabalho e Análise dos Resultados 9
3.1. Bases de Conhecimento 9
3.2. Representação de Conhecimento Perfeito 11
3.2.1. Conhecimento Positivo 11
3.2.2. Conhecimento Negativo 11
3.3. Representação de Conhecimento Imperfeito 12
3.3.1. Conhecimento Incerto 13
3.3.2. Conhecimento Impreciso 16
3.3.3. Conhecimento Interdito 18
3.4. Manipulação de Invariantes 19
3.5. Problemática da evolução do conhecimento 21
3.6. Sistema de Inferência 21
4. Conclusão e Sugestões 24
5. Anexos 25

1. Introdução

O objetivo principal deste exercício prático incide, como já foi referido anteriormente, na aplicação de todo o conhecimento adquirido relativo à Programação em Lógica Estendida e Conhecimento Imperfeito. Por se tratar de uma continuação do trabalho empregue na primeira fase, é imperativo continuar a utilizar a linguagem de programação em lógica *PROLOG*, aproveitando-se de igual modo para se aprofundar o conhecimento relativo a esta linguagem.

Enquanto que no exercício anterior apenas nos era pedido o desenvolvimento de um Sistema de Representação de Conhecimento e Raciocínio com aptidão para caracterizar todo um cenário focado na área da prestação dos cuidados de saúde, neste exercício é necessário ir mais além, criando-se um sistema que seja não só capaz de receber e armazenar toda a informação associada à nossa base de conhecimento mas ao mesmo tempo de lidar com determinadas restrições de conhecimento denominados de Conhecimento Imperfeito.

Assim sendo, vai ser feita toda uma distinção dos vários tipos de Conhecimento Imperfeito, explicando-se o que cada um deles é e como é que serão aplicados à nossa Base de Conhecimento.

2. Preliminares

Este segundo exercício prático acaba por ser uma continuação do primeiro exercício no sentido em que se cria uma "ponte" entre os conhecimentos adquiridos anteriormente e os conhecimentos que agora vão ser reaproveitados e devidamente aprofundados.

No primeiro exercício focamo-nos na criação de um Sistema de Conhecimento e Raciocínio que funcionava como uma espécie de "mundo fechado" uma vez que as respostas às questões só poderiam ser do tipo **verdadeiro** ou **falso**, consoante a existência ou não de informação no Sistema de Conhecimento capaz de provar o cariz da afirmação. Neste segundo exercício, vai surgir toda uma nova forma de se representar conhecimento com a possibilidade de se conseguir dar resposta a conhecimento que é desconhecido. Neste caso, além da resposta à questão poder ser **verdadeiro** ou **falso**, passa também a poder ser **desconhecido**, que se obtém quando não existe maneira de provar que uma informação possa ser verdadeira ou falsa.

Com isto, facilmente se compreende que este trabalho prático vai exigir dois parâmetros fundamentais para a sua realização:

- O Sistema de Representação de Conhecimento e Raciocínio esboçados no exercício de grupo anterior;
- A capacidade de se conseguir caracterizar Conhecimento Imperfeito.

Antes de se embarcar para a resolução propriamente dita, é importante fazer uma breve referência a conceitos como Base de Dados, Base de Conhecimento e Representação de Conhecimento Imperfeito.

2.1. Representação de Conhecimento

Os sistemas de Bases de Dados (BD's) e os sistemas de representação de conhecimento, lidam com aspectos concretos do mundo real e podem-se comparar nos termos em que dividem a utilização da informação. Ambos os sistemas distinguem as funções de representação – descrição do esquema conceptual e dos dados – e de computação – construção de respostas a questões e manipulação dos dados.

Partindo-se da citação acima transcrita, apresentaremos (distinguindo-se estes dois sistemas) os pressupostos pelos quais se baseiam estas duas linguagens de manipulação.

2.1.1. Base de Dados

As linguagens de manipulação de Base de Dados baseiam-se no princípio de que apenas existe e é valida a informação que se encontre contida na linguagem, sendo que a restante é considerada como falsa.

Eis os pressupostos nos quais as linguagens de manipulação de informação num Sistema de Base de Dados se baseiam:

- **Pressuposto do Mundo Fechado** - toda a informação que não se encontre contida na Base de Dados é dada como falsa;
- **Pressuposto dos Nomes Únicos** - duas constantes diferentes (que definam valores atómicos ou objetos) designam, inevitavelmente, duas entidades diferentes do universo de discurso;
- **Pressuposto do Domínio Fechado** - não existem mais objetos no universo do discurso para além daqueles designados por constantes na Base de Dados.

2.1.2. Sistema de Representação de Conhecimento

Por outro lado, os Sistemas de Representação de Conhecimento nem sempre assumem que a informação representada seja a única que pode ser válida e que as entidades representadas sejam as únicas existentes no mundo exterior.

Eis os pressupostos nos quais as linguagens de manipulação de informação num Sistema de Representação de Conhecimento se baseiam:

- **Pressuposto do Mundo Aberto** - podem existir outros factos/conclusões verdadeiras para além daqueles apresentados na Base de Conhecimento;
- **Pressuposto dos Nomes Únicos** - duas constantes diferentes (que definam valores atómicos ou objetos) designam, obrigatoriamente, duas entidades diferentes do universo de discurso;
- **Pressuposto do Domínio Aberto** - podem existir objetos do universo de discurso para além daqueles designados pelas constantes da Base de Conhecimento.

Facilmente se deduz e conclui que os requisitos das Base de Dados e dos Sistemas de Representação de Conhecimento para o tratamento de informação são diferentes. É perante esta análise e distinção que o nosso trabalho será desenvolvido. A ideia passa então por adotar um Sistema de Representação de Conhecimento de características capazes de fazer o nosso Sistema raciocinar segundo os três pressupostos em que este tipo de Sistema se baseia.

Assim, iremos representar informação incompleta, visto que é uma problemática do mundo real pois nem sempre é possível saber uma determinada informação ou por esta ser desconhecida ou incompleta.

Surge com isto a necessidade de alargar o nosso domínio de conhecimento que outrora se limitava ao **verdadeiro** e **falso**, acrescentado o **desconhecido** que representará a informação que de momento não pode ser definida, isto é, nem é falsa nem verdadeira. É com isto que surge a programação em lógica estendida capaz de resolver esta situações.

3. Descrição do Trabalho e Análise dos Resultados

Nesta secção irá ser discutida e devidamente fundamentada toda a resolução deste segundo exercício de grupo. O foco deste capítulo estará em respeitar as necessidades de demonstração das várias funcionalidades enumeradas no enunciado do trabalho prático, no sentido de se conseguir produzir uma resolução atenta, mas concisa e aceitável.

Para se poder ir ao encontro destas funcionalidades o mais direto e organizado possível, o grupo decidiu criar um tópico para cada uma destas funcionalidades pedidas:

- Representação de conhecimento positivo e negativo;
- Representação de conhecimento imperfeito, em que inevitavelmente se abordarão o conhecimento incerto, conhecimento impreciso e conhecimento interdito;
- Manipulação de invariantes;
- Problemática da evolução de conhecimento;
- Sistema de Inferência.

3.1. Bases de Conhecimento

Tendo em conta que o universo que se pretende representar continua a focar-se na prestação de cuidados de saúde, as bases de conhecimento que foram representadas desde o início do primeiro exercício de grupo, acabam por se manter praticamente iguais, existindo apenas mínimas alterações.

Como o objetivo passa por representar informação incompleta, é obrigatório se alterar o domínio de soluções passando a incluir a opção do **Desconhecido**. Este representa um valor que não pode ser definido de imediato e é com ele que surge o conceito de programação em lógica estendida. É através deste tipo de programação que seremos capazes de representar situações de conhecimento desconhecido ou incompleto.

Eis os 3 diferentes tipos de conclusões que passam a existir para uma questão:

- **Verdadeiro** - quando é possível provar uma questão(Q) na base de conhecimento;
- **Falso** - quando é possível provar a falsidade de uma questão (-Q) na base de conhecimento;
- **Desconhecido** - quando não é possível provar a questão(Q) nem a negar (-Q) na base de conhecimento.

Com esta alteração, a definição das bases de conhecimento passa a ser a seguinte:

- *utente: (#IDU, Nome, Idade, Cidade) -> {V,F,D}*
- *prestador: (#IDP, Nome, Instituicao) -> {V,F,D}*
- *cuidado: (Data, #IDU, #IDP, Custo) -> {V,F,D}*

3.2. Representação de Conhecimento Perfeito

3.2.1. Conhecimento Positivo

Como se tem vindo a deixar evidente ao longo deste relatório, a representação de conhecimento positivo já se encontra realizada no primeiro exercício prático. No entanto, para a realização deste exercício foi necessário acrescentar a parte da representação do conhecimento negativo.

3.2.2. Conhecimento Negativo

Este conhecimento negativo foi representado/produzido tendo-se em conta os seus dois diferentes tipos:

- **Negação por Falha** - quando não existe nenhuma prova aquando da negação do predicado. Esta negação é representada pelo predicado `nao`, que se encontra abaixo definido.

```
% Negação por Falha para um Utente.
-utente(IdU, Nome, Idade, Morada) :-
      nao(utente(IdU,Nome,Idade,Morada)),
      nao(excecao(utente(IdU, Nome, Idade, Morada))).

% Negação por Falha para um Prestador.
-prestador(IDP,Nome,Especialidade,Instituicao) :-
      nao(prestador(IDP,Nome,Especialidade,Instituicao)),
      nao(excecao(prestador(IDP,Nome,Especialidade,Instituicao))).

% Negação por Falha para um Cuidado.
-cuidado(Data,IDU,IDP,Custo) :-
      nao(cuidado(Data,IDU,IDP,Custo)),
      nao(excecao(cuidado(Data,IDU,IDP,Custo))).
```

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

Relativamente ao utente, para se provar que um predicado é falso recorreu-se à negação por falha. O processo consiste em dado um utente procurar se não existe na base de conhecimento este ou se não existe uma exceção para este. Caso isto aconteça o resultado será falso.

De forma análoga aplica-se a mesma explicação para os casos do prestador e do cuidado.

- **Negação Forte** - quando se afirma que um determinado predicado é falso. Esta negação é representada pelo teorema (-).

```
% Negação Forte para um Utente.
-utente(1000,'Joaquim Nogueira',30,'Fanalicao').

% Negação Forte para um Prestador.
-prestador(6666,'Joao Rodrigues','Agente','Hospital Privado de
Guimaraes').

% Negação Forte para um Cuidado.
-cuidado(date(2018,11,12),3333,5555,'Carie',12).
```

3.3. Representação de Conhecimento Imperfeito

Na problemática da representação de informação incompleta, os valores nulos surgem com o objetivo de distinguir situações em que as respostas às questões são conhecidas (verdadeiro ou falso) e as situações em que as respostas às questões são desconhecidas. É neste sentido que surgem os três tipos de conhecimento imperfeito que são essenciais para todo um alargar de possibilidades de respostas às questões com base nas fontes de conhecimento existentes.

Façamos então uma distinção entre os três tipos de conhecimento imperfeito a que nos estamos a referir:

- **Conhecimento Incerto** - permite representar valores desconhecidos;
- **Conhecimento Impreciso** – permite representar valores desconhecidos, mas de um determinado conjunto de valores;
- **Conhecimento Interdito** – permite representar valores não permitidos, definidos como interditos na Base de Conhecimento a considerar.

Partindo-se desta distinção, criar-se-á um tópico para cada um dos conhecimentos de modo a se puder explicar e demonstrar de forma mais arrumada em que cada um destes consiste.

3.3.1. Conhecimento Incerto

Este tipo de conhecimento corresponde ao conhecimento para o qual não existe nenhuma prova que comprove que seja verdadeiro ou falso. Trata-se, portanto, de um valor nulo do tipo desconhecido e não necessariamente de um conjunto de valores.

Caracterizamos conhecimento tanto para o Utente como para o Prestador sendo que a única diferença entre estes reside no parâmetro que se decidiu "avaliar":

- **Utente** – desconhecimento da sua morada, nome e da sua idade;
- **Prestador** – desconhecimento da sua instituicao (sabendo apenas que não é Classaude Porto), da sua morada e do seu nome.

Fazendo uma análise do caso abaixo transcrito que diz respeito ao desconhecimento da Morada de um determinado Utente:

- Começamos por definir que para o `utente` Rodrigo Pires, de 60 anos, com ID igual a 6000 não temos informação relativa à sua morada;
- De seguida, para garantir a consistência da nossa Base de Conhecimento, criamos uma exceção que permite que a nossa contenha um `utente` cuja sua morada não se conheça.
- Para permitir a atualização da morada para os utentes foi desenvolvido o predicado `evolucaoIncertoMorada`. Este, testa inicialmente através do predicado demo, se a morada passada como argumento tem como resultado desconhecido. Se assim for, armazena todas a exceções numa lista L que contenham exatamente o mesmo `IDU`, o mesmo `Nome` e a mesma `Idade` passados como argumento, garantindo através do predicado `nao` que não seja inserido conhecimento interdito (isto é, uma morada definida como interdita). De seguida, são removidas todas as exceções que satisfizeram as condições supracitadas e é inserido o novo conhecimento na nossa base.

```
% Desconhecimento da morada de um Utente.
utente(6000,'Rodrigo Pires',60,morada_desconhecida).

% Exceção associada.
excecao(utente(IdU,Nome,Idade,Morada)) :-
      utente(IdU,Nome,Idade,morada_desconhecida).

% Extensão do predicado que permite a evolução do conhecimento imperfeito
incerto: Termo -> {V,F}
evolucaoIncertoMorada(utente(IDU,Nome,Idade,Morada)) :-
      demo(utente(IDU,Nome,Idade,Morada), desconhecido),
      solucoes((excecao(utente(IDU,Nome,Idade,Morada))
      :- utente(IDU,Nome,Idade,X)),
      (utente(IDU,Nome,Idade,X), nao(nulo(X))) , L),
      removeLista(L), remocao(utente(IDU,Nome,Idade,X)),
      evolucao(utente(IDU,Nome,Idade,Morada)).
```

O caso abaixo transcrito difere dos restantes, pois precisamos de garantir que a instituição para o prestador com ID igual a 9002 nunca corresponde à Classaude Porto e para tal adicionamos uma negação forte:

- Começamos por definir que para o `prestador` `Rui Alberto`, `Dentista`, com ID igual a `9002` não temos informação relativa à instituição onde opera;
- De seguida, criamos uma exceção que permite que a nossa Base contenha um `prestador` cuja instituição onde trabalha não se conheça;
- Para permitir a atualização do nome dos prestadores foi desenvolvido o predicado `evolucaoIncertoNome` funcionará da mesma forma foi explicado anteriormente, sendo apenas importante referir que na situação em que se pretende adicionar Classaude Porto como instituição para este prestador, como o resultado do predicado `demo` será falso, não será permitida a inserção pois a execução será parada.

```
% Desconhecimento da instituicão do Prestador, sabendo que não é a
Classaude Porto.
prestador(9002,'Rui Alberto','Dentista',instituicao_desconhecida).

% Exceção associada.
excecao(prestador(IdP,Nome,Especialidade,Instituicao)) :-
      prestador(IdP,Nome,Especialidade,instituicao_desconhecida).

% Negação.
-prestador(9002,'Rui Alberto','Dentista','Classaude Porto').

% Extensão do predicado que permite a evolução do conhecimento imperfeito
incerto: Termo -> {V,F}
evolucaoIncertoNome(prestador(IDP,Nome,Especialidade,Instituicao)) :-
      demo(prestador(IDP,Nome,Especialidade,Instituicao), desconhecido),
      solucoes((excecao(prestador(IDP,Nome,Especialidade,Instituicao))
      :- prestador(IDP,X,Especialidade,Instituicao)),
      (prestador(IDP,X,Especialidade,Instituicao), nao(nulo(X))), L),
      removeLista(L),remocao(prestador(IDP,X,Especialidade,Instituicao),
      evolucao(prestador(IDP,Nome,Especialidade,Instituicao)).
```

De forma análoga é efetuado o mesmo procedimento para os restantes predicados tanto do Utente como do Prestador. Para se evitar uma sobrecarga de informação desnecessária, o grupo optou por colocar estes predicados na secção 5 (Anexos).

3.3.2. Conhecimento Impreciso

Tal como o conhecimento desconhecido, este tipo corresponde ao conhecimento para o qual não há provas que comprove que seja verdadeiro ou falso, mas, no entanto, insere-se num conjunto de valores conhecidos, sendo que apenas não se consegue provar a quais deles é que irá responder à questão em causa.

Assim, qualquer que seja a resposta ao predicado que utilize um valor fora desse conjunto, o resultado será desconhecido, uma vez que apenas se sabe que o valor se encontra contido dentro desse conjunto.

Caracterizamos conhecimento tanto para o Utente como para o Prestador e ainda Cuidado sendo que a única diferença entre estes reside no parâmetro que se decidiu incutir um conjunto de valores possíveis:

- **Utente** – desconhecimento do seu nome (sabendo apenas que pode ser Joaquim Nogueira ou João Nogueira);
- **Prestador** – desconhecimento da sua especialidade (sabendo que apenas pode ser Ortopedia, Nutrucionismo, Ortopedia ou Psicologia);
- **Cuidado** – desconhecimento do seu custo (sabendo que apenas pode ser 30, 40 ou 45).

Fazendo uma análise do caso abaixo transcrito que diz respeito ao desconhecimento da especialidade do Prestador:

- Cria-se inicialmente todo o conjunto de exceções que nos permitem estabelecer os valores entre os quais a especialidade se pode encontrar;
- Para atualizar a informação referente um prestador recorre-se ao predicado evolucaoImpreciso. Este começa por testar se o resultado do demo é igual a desconhecido, caso seja, coloca todas as exceções numa lista, remove a lista da base de conhecimento e por fim adiciona o novo prestador.

```
% Desconhecimento da especialidade exercida pelo Dr. Afonso Alves na
clínica Classaude Porto, sabendo-se que é Nutricionismo, Ortopedia ou
Psicologia.
excecao(prestador(9011,'Afonso Alves','Nutricionismo','Classaude
Porto')).
excecao(prestador(9011,'Afonso Alves','Ortopedia','Classaude Porto')).
excecao(prestador(9011,'Afonso Alves','Psicologia','Classaude Porto')).

% Extensão do predicado que permite a evolucao do conhecimento
imperfeito impreciso: Termo -> {V,F}
evolucaoImpreciso(prestador(IDP,Nome,Especialidade,Instituicao)) :-
      demo(prestador(IDP,Nome,Especialidade,Instituicao),desconhecido),
      solucoes(excecao(prestador(IDP,N,E,I)),
      excecao(prestador(IDP,N,E,I)),L),
      removeLista(L),
      evolucao(prestador(IDP,Nome,Especialidade,Instituicao)).
```

Para os Utentes e Cuidados segue-se exatamente o mesmo pensamento estabelecido acima. Tal como acontece no conhecimento incerto, tanto as exceções como os predicados de evolução referentes a estas outras duas bases de conhecimento, encontram-se na secção 5 (Anexos).

3.3.3. Conhecimento Interdito

Este último tipo de conhecimento, para além de caracterizar um tipo de dados desconhecido, caracteriza também, um tipo de dados que não se admite que surja na base de conhecimentos.

Nesta situação, o valor nulo para além de identificar um valor desconhecido, representará um valor que não é permitido especificar ou conhecer e qualquer tentativa para a concretizar será rejeitada como sendo provocadora de inconsistência na informação presente na base de conhecimento.

Caracterizamos conhecimento tanto para o Utente como para o Prestador sendo que a única diferença entre estes reside no parâmetro que se decide como impossível de existir:

- **Utente** – impossibilidade de se conhecer o nome de um determinado Utente;
- **Prestador** – impossibilidade de se conhecer a instituição onde um determinado Prestador opera.

Fazendo uma análise do caso abaixo transcrito que diz respeito ao à impossibilidade de:

- Para este caso foi definido que o nome do utente com ID 9999 tem como nome interdito nome_interdito.
- Para garantir que tal informação não aparece na Base de Conhecimento e para não permitir a evolução deste, desenvolvemos um invariante correspondente para tal. O que este invariante se encarrega de fazer é encontrar todos os utentes com os mesmos atributos (exceto nome) na nossa Base e após isso verificar se os nomes desses utentes não são nulos.

```
% Impossibilidade de se saber um determinado nome correspondente a um
determinado Utente.
utente(9999,nome_interdito, 19, 'Fafe').
excecao(utente(ID,Nome,Idade,Cidade)) :-
utente(ID,nome_interdito,Idade,Cidade).
nulo(nome_interdito).
```

3.4. Manipulação de Invariantes

Para que a Base de Conhecimento funcione corretamente é indispensável garantir certas condições responsáveis pelo controlo da inserção e remoção de conhecimento. Para esse efeito, foi desenvolvido já no exercício anterior uma série de invariantes responsáveis por este controlo que optaremos por não tornar a incluir neste relatório.

Contudo, para além deste controlo de inserção/remoção acaba por ser também imperativo garantir igualmente o controlo da:

- Inserção de conhecimento perfeito positivo e negativo;
- Inserção de conhecimento imperfeito interdito.

Desta forma, foi necessário desenvolver invariantes que permitissem o controlo de conhecimento perfeito positivo perante situações em que existe conhecimento negativo:

- Por conseguinte, foram elaborados três invariantes com a mesma lógica de procedimento para o utente, prestador e cuidado. Estes, garantem que só seja possível a inserção na base de conhecimentos quando não existe conhecimento negativo do que se pretende inserir.

```
% Não permitir adicionar conhecimento quando se tem o conhecimento perfeito
negativo oposto.
+utente(Id,N,I,M) :: nao(-utente(Id,N,I,M)).

+prestador(Id,N,E,I) :: nao(-prestador(Id,N,E,I)).

+cuidado(D,IU,IP,D,C) :: nao(-cuidado(D,IU,IP,D,C)).
```

- Foi necessário a garantir que, para que a base de conhecimentos permanecesse consistente, não fosse permitida a introdução de exceções de conhecimento perfeito positivo já existente. Sendo assim, o invariante apresentado de seguida, verifica aquando da evolução se não existe nenhum conhecimento perfeito positivo desse termo. Caso não exista, a inserção da exceção é executada.

```
% Impossivel adicionar exceções a conhecimento positivo.
+excecao(T) :: nao(T).
```

Relativamente às exceções, ainda foi necessário garantir que não existisse a repetição destas, isto é, duas exceções iguais. Para tal elaboramos os seguintes invariantes que garantem que não existam duas exceções iguais, tanto para o conhecimento negativo como para o positivo:

```
% Não permitir a adição de exceções que sejam repetidas.
+(excecao(T)) :: (findall(excecao(T),excecao(T),S),comprimento(S,N), N <
2).

+(excecao(-T)) :: (findall(excecao(T),excecao(-T),S),comprimento(S,N), N
< 2).
```

De forma a controlar a inserção de conhecimento interdito foi necessário elaborar um invariante para o caso apresentado que se assegurasse que tal inserção acontecia. Assim, apresentamos de seguida, o invariante que permite o controlo da inserção do conhecimento interdito relativo ao Nome de um determinado utente:

- Logo, para este caso específico este invariante garante que a lista que contem todos os conjuntos de soluções com o nulo ou nome interdito terá sempre comprimento igual a 0.
- Por conseguinte, se assegura que não seja permita a evolução para este utente, com o nome interdito.

```
+utente(Id,Nome,I,C) :: (solucoes(N, (utente(9999,N, 19, 'Fafe'),
nao(nulo(N))), L), comprimento(L,R), R==0).
```

3.5. Problemática da evolução do conhecimento

Relativamente à problemática da evolução do conhecimento, esta temática prende-se no facto de garantir que a cada inserção/remoção na Base de Conhecimento permaneça coesa e fiável.

Para tal, é necessário implementar uma série de medidas que não permita:

- Remover informação que seja dependente de outra;
- Inserção de informação repetida.

Deste modo, no momento em que se pretende alterar a Base de Conhecimento é imperativo testar se tal alteração não irá corromper a Base de Conhecimento. Toda esta problemática acaba por ser solucionada pela implementação dos invariantes que até então fomos abordando/explicitando.

3.6. Sistema de Inferência

Um sistema de inferência capaz de implementar os mecanismos de raciocínio inerentes aos sistemas que anteriormente foram representados é um passo

indispensável ser efetuado uma vez que é com este sistema que podemos "criar" uma espécie de interpretador de questões.

Este interpetador estará apto, tal como deixamos evidente até agora, para poder interpetar a veracidade de uma determinada questão, dando como resultado:

- **Verdadeiro** – caso o conhecimento em causa esteja presente na Base de Conhecimento;
- **Falso** – caso o conhecimento em causa não esteja presente na Base de Conhecimento;
- **Desconhecido** – caso não exista informação na Base de Conhecimento ao ponto de se poder concluir se é verdadeira ou falsa.

Desenvolvemos um predicado demo para se interpetar uma questão mas também um conjunto delas:

- **demo** – quando se pretende aplicar o demo a uma só questão;
- **demoComp** – quando se pretende aplicar o demo a uma lista de questões.

Fazendo uma análise do caso abaixo transcrito que diz respeito ao predicado demoComp:

- Começa-se por aplicar o demo à cabeça da lista, aplicando o demoComp à tail da lista;
- Por fim, dependendo se for uma disjunção ou conjunção foram definidos predicados – conjunção e disjunção -para obter o resultado final.

```
% Extensao do meta-predicado demo: Questao, Resposta -> {V,F}
demo(Q,verdadeiro) :- Q.
demo(Q,falso) :- -Q.
demo(Q,desconhecido) :- nao(Q), nao(-Q).


% Extenão do predicado demoComp: Lista, R -> {V,F,D}
demoComp([Q],R) :- demo(Q,R).
demoComp([Q1,ou,Q2|T],R) :-
    demo(Q1,R1),
    demoComp([Q2|T],R2),
    disjuncao(R1,R2,R).

demoComp([Q],R) :- demo(Q,R).
demoComp([Q1,e,Q2|T],R) :-
    demo(Q1,R1),
    demoComp([Q2|T],R2),
    conjuncao(R1,R2,R).


conjuncao(verdadeiro,verdadeiro,verdadeiro).
conjuncao(verdadeiro,desconhecido,desconhecido).
conjuncao(desconhecido,verdadeiro,desconhecido).
conjuncao(desconhecido,desconhecido,desconhecido).
conjuncao(falso,_,falso).
conjuncao(_,falso,falso).

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

4. Conclusão e Sugestões

A realização deste segundo exercício de grupo foi fundamental para permitir o aprofundar das capacidades no que diz respeito à programação em lógica estendida em *PROLOG*, linguagem já usada no exercício de grupo anterior. Para além disto, permitiu toda uma expansão de conhecimento/aprofundamento na representação do conhecimento imperfeito que inicialmente era visto como sendo algo confuso e difícil de se manipular.

Apesar de já existir uma familiarização por parte do grupo em representar conhecimento, foi inevitável o surgimento de dúvidas e dificuldades aquando da perceção daquilo que seriam as melhores escolhas para a implementação dos demais requisitos. A grande dificuldade deste exercício esteve sempre em implementar corretamente os três tipos de conhecimento imperfeito e ao mesmo tempo os invariantes que consideramos importantes criar para o controle dos mesmos.

Fomos capazes de solidificar conhecimento nesta linguagem de programação lógica, criar novos pensamentos e ainda expandir ideias de maneira a criar toda uma resolução sensata e isenta de todos os problemas que foram existindo. Faz-se assim um feedback positivo, esperançoso e motivador para a resolução do último exercício de grupo que se aproxima.

5. Anexos

```
%-------------------- Base de Conhecimento Inicial --------------------%

% Base de Conhecimento Utente (IDU,Nome,Idade,Cidade).
utente(6472,'Diogo Nogueira',20,'Fafe').
utente(1211,'Diogo Nogueira',21,'Porto').
utente(8374,'Tifany Silva',24,'Porto').
utente(3463,'Fabio Fontes',19,'Braga').
utente(2321,'Mariana Lino',20,'Guimaraes').
utente(9232,'Francisco Silva',21,'Braga').
utente(2372,'Cristina Lopes',22,'Fafe').
utente(7363,'Tiago Nogueira',18,'Porto').
utente(6252,'Sara Costa',18,'Porto').
utente(2373,'Joana Maria',28,'Braga').
utente(2324,'Joao Miguel',21,'Guimaraes').

% Base de Conhecimento Prestador (IDP,Nome,Especialidade,Instituicao).
prestador(4343,'Navarro Silva','Dermatologia','Hospital Privado de
Guimaraes').
prestador(3947,'Margarida Antunes','Dentista','Hospital Privado de
Guimaraes').
prestador(3643,'Nuno Costa','Nutricao','Classaude Porto').
prestador(7364,'Ana Maria','Hidroterapia','Clinica Fisiatrica de Fafe').
prestador(3642,'Tomas Casimiro','Clínica Geral','Classaude Porto').
prestador(8675,'Maria Joana','Psiquiatria','Clinica Praxis Porto').
prestador(3742,'Joana Silva','Psicologia','Clinica Praxis Porto').
prestador(7864,'Ana Luisa','Dermatologia','Clinica Fisiatrica de Fafe').
prestador(7324,'Mariana Costa','Dentista','Clinica Fisiatrica de Fafe').
prestador(3127,'Joao Araujo','Dentista','Hospital Privado de
Guimaraes').

% Base de Conhecimento Cuidado de Saúde (Data,IDU,IDP,Custo).
cuidado(date(2018,11,12),6472,4343,'Carie',12).
cuidado(date(2017,10,25),8374,7324,'Consulta de rotina aparelho',20).
cuidado(date(2016,09,15),3463,7864,'Queimadura na cara',15).
cuidado(date(2018,11,12),6472,3742,'Consulta',50).
cuidado(date(2015,01,25),9232,8675,'Consulta agendada/rotina',55).
cuidado(date(2017,08,30),2372,3642,'Paciente com febre',15).
cuidado(date(2018,04,21),7363,7364,'Tratamento rotineiro',25).
cuidado(date(2016,06,08),6252,3643,'Consulta de iniciacao de tratamento
dietetico',60).
cuidado(date(2017,12,24),2373,3947,'Remocao dos dentes do siso',55).
cuidado(date(2018,12,12),6472,4343,'Tratamento de eczema',60).
```

```
%----------------------- Conhecimento Incerto -----------------------%

% Desconhecimento da idade do Utente.
utente(6001,'Rodrigo Pires',idade_desconhecida,'Braga').

% Excecao associada.
excecao(utente(IdU,Nome,Idade,Morada))
      :- utente(IdU,Nome,idade_desconhecida,Morada).

% Desconhecimento da idade do Utente, sabendo que não é 50.
utente(7770,'Allan Xavier',idade_desco,'Braga').

% Exceção associada.
excecao(utente(IdU,Nome,Idade,Morada))
      :- utente(IdU,Nome,idade_desco,Morada).

% Negação.
-utente(7770,'Allan Xavier', 50,'Braga').

% Extensão do predicado que permite a evolução do conhecimento imperfeito
incerto: Termo -> {V,F}
evolucaoIncertoIdade(utente(IDU,Nome,Idade,Morada))
      :- demo(utente(IDU,Nome,Idade,Morada), desconhecido),
      solucoes((excecao(utente(IDU,Nome,Idade,Morada))
      :- utente(IDU,Nome,X,Morada)),
      (utente(IDU,Nome,X,Morada), nao(nulo(X)) ),L),
      removeLista(L),
      remocao(utente(IDU,Nome,X,Morada)),
      evolucao(utente(IDU,Nome,Idade,Morada)).

            ------------------------------------------------

% Desconhecimento do nome do Utente.
utente(6002,nome_desconhecido,30,'Braga').

% Exceção associada.
excecao(utente(IdU,Nome,Idade,Morada))                                  :-
utente(IdU,nome_desconhecido,Idade,Morada).

% Extensão do predicado que permite a evolução do conhecimento imperfeito
desconhecido incerto: Termo -> {V,F}
evolucaoIncertoNome(utente(IDU,Nome,Idade,Morada))
      :- demo(utente(IDU,Nome,Idade,Morada), desconhecido),
      solucoes((excecao(utente(IDU,Nome,Idade,Morada))
      :- utente(IDU,X,Idade,Morada)),
      (utente(IDU,X,Idade,Morada), nao(nulo(X)) ),L),
      removeLista(L),
      remocao(utente(IDU,X,Idade,Morada)),
```

```
      evolucao(utente(IDU,Nome,Idade,Morada)).

            ----------------------------------------------

% Desconhecimento do nome do Prestador.
prestador(9003,nome_nao_sei,'Dentista','Classaude Porto').

% Exceção associada.
excecao(prestador(IDP,Nome,Especialidade,Instituicao)) :-
prestador(IDP,nome_nao_sei,Especialidade,Instituicao).

% Extensão do predicado que permite a evolucao do conhecimento
imperfeito desconhecido: Termo -> {V,F}
evolucaoIncertoNome(prestador(IDP,Nome,Especialidade,Instituicao))
      :- demo(prestador(IDP,Nome,Especialidade,Instituicao),
desconhecido),
      solucoes((excecao(prestador(IDP,Nome,Especialidade,Instituicao))
      :- prestador(IDP,X,Especialidade,Instituicao)),
      (prestador(IDP,X,Especialidade,Instituicao), nao(nulo(X)) ),L),
      removeLista(L),
      remocao(prestador(IDP,X,Especialidade,Instituicao)),
      evolucao(prestador(IDP,Nome,Especialidade,Instituicao)).

            ----------------------------------------------

% Desconhecimento da especialidade do Prestador.
prestador(9001,'Rui Alberto',especialidade_desconhecida,'Hospital').

% Exceção associada.
excecao(prestador(IDP,Nome,Especialidade,Instituicao)) :-
prestador(IDP,Nome,especialidade_desconhecida,Instituicao).

% Extensão do predicado que permite a evolução do conhecimento
imperfeito incerto: Termo -> {V,F}
evolucaoIncertoEspecialidade(prestador(IDP,Nome,Especialidade,Instituica
o))
      :- demo(prestador(IDP,Nome,Especialidade,Instituicao),
desconhecido),
      solucoes((excecao(prestador(IDP,Nome,Especialidade,Instituicao))
      :- prestador(IDP,Nome,X,Instituicao)),
      (prestador(IDP,Nome,X,Instituicao), nao(nulo(X)) ),L),
      removeLista(L), remocao(prestador(IDP,Nome,X,Instituicao)),
      evolucao(prestador(IDP,Nome,Especialidade,Instituicao)).
```

```
%---------------------- Conhecimento Impreciso ------------------------%


% Desconhecimento do nome do Utente, sabendo-se que é Joaquim Nogueira ou
João Nogueira.
excecao(utente(1001,'Joaquim Nogueira',23,'Lisboa')).
excecao(utente(1001,'João Nogueira',23,'Lisboa')).

% Extensão do predicado que permite a evolucao do conhecimento imperfeito
impreciso: Termo -> {V,F}
evolucaoImpreciso(utente(IDU,Nome,Idade,Cidade))
      :- demo(utente(IDU,Nome,Idade,Cidade),desconhecido),
      solucoes(excecao(utente(IDU,N,I,C)), excecao(utente(IDU,N,I,C)),L),
      removeLista(L),
      evolucao(utente(IDU,Nome,Idade,Cidade)).

            ------------------------------------------------

% Desconhecimento da especialidade exercida pelo Dr. Afonso Alves na clínica
Classaude Porto, sabendo-se que é Nutricionismo, Ortopedia ou Psicologia.
excecao(prestador(9011,'Afonso Alves','Nutricionismo','Classaude Porto')).
excecao(prestador(9011,'Afonso Alves','Ortopedia','Classaude Porto')).
excecao(prestador(9011,'Afonso Alves','Psicologia','Classaude Porto')).

% Extensão do predicado que permite a evolucao do conhecimento imperfeito
impreciso: Termo -> {V,F}
evolucaoImpreciso(prestador(IDP,Nome,Especialidade,Instituicao))
      :- demo(prestador(IDP,Nome,Especialidade,Instituicao),desconhecido),
      solucoes(excecao(prestador(IDP,N,E,I)),
      excecao(prestador(IDP,N,E,I)),L),
      removeLista(L),
      evolucao(prestador(IDP,Nome,Especialidade,Instituicao)).

            ------------------------------------------------

% Desconhecimento do custo de uma consutla no nome de Sara Costa, ocorrida
no Hospital Privado de Guimarães, no dia 25 de Janeiro de 2018 pelo dentista
João Araújo, sabendo-se que este custo é 30, 40 ou 45.
excecao(cuidado(date(2018,01,25),6252,3127,'Carie',30)).
excecao(cuidado(date(2018,01,25),6252,3127,'Carie',40)).
excecao(cuidado(date(2018,01,25),6252,3127,'Carie',45)).

% Extensão do predicado que permite a evolucao do conhecimento imperfeito
impreciso: Termo -> {V,F}
evolucaoImpreciso(cuidado(Data,IDU,IDP,Descricao,Custo))
      :- demo(cuidado(Data,IDU,IDP,Descricao,Custo),desconhecido),
```

```
      solucoes(excecao(cuidado(D,IDU,IDP,Desc,C)),excecao(cuidado(D,IDU,IDP
,Desc,C)),L),
      removeLista(L),
      evolucao(cuidado(Data,IDU,IDP,Descricao,Custo)).


%---------------------- Conhecimento Interdito ------------------------%

% Impossibilidade de se saber uma determinada instituicão correspondente a
um determinado Prestador.
prestador(1005,'Ricardo Almeida','Psiquiatria',instituicao_interdita).
excecao(prestador(IDP,Nome,Especialidade,Instituicao)) :-
prestador(IDP,Nome,Especialidade,instituicao_interdita).
nulo(instituicao_interdita).

%---------------------- Predicados Auxiliares ------------------------%

% Extensão do predicado comprimento : L, N ->{V,F}
comprimento([],0).
comprimento([X|L],R) :- comprimento(L,N), R is 1+N.


% Extensão do predicado removeLista : L ->{V,F}
removeLista([]).
removeLista([X|L]) :- retract(X),
                      removeLista(L).

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