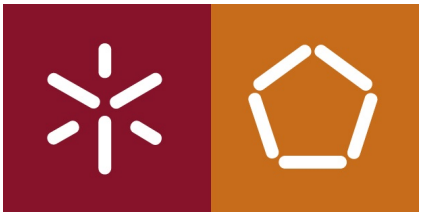
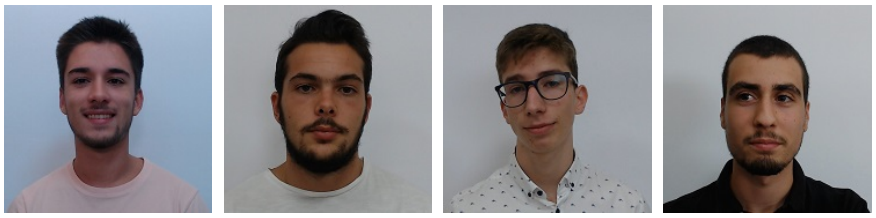


Universidade do Minho
Mestrado Integrado em Engenharia Informática
3ºano - 2º Semestre

Sistemas de Representação de Conhecimento e Raciocínio

Trabalho Prático Nº.1

Grupo 31



a85059 – Diogo Paulo Lopes de Vasconcelos
a83732 – Gonçalo Rodrigues Pinto
a84197 – João Pedro Araújo Parente
a84829 – José Nuno Martins da Costa

3 de Maio de 2020

Conteúdo

Lista de Figuras

1 Introdução

No 2º semestre do 3º ano do Curso de Engenharia Informática da Universidade do Minho, existe uma unidade curricular denominada por Sistemas de Representação de Conhecimento e Raciocínio, que tem como objectivo introduzir aos estudantes os paradigmas da representação do conhecimento e de raciocínio lógico, e sua aplicação na concepção e implementação de sistemas inteligentes ou de apoio à decisão, em que a qualidade da informação e o grau de confiança que é depositado na conjugação dos atributos de um predicado ou função lógica é fulcral como também tem como objectivo ajudar os estudantes a compreender a relação entre a complexidade de um modelo representação de conhecimento e as formas de raciocínio a ele associadas e o seu desempenho, utilizando esta informação na definição de uma estratégia para a sua optimização.

O presente trabalho teve como principal objectivo motivar os alunos para a utilização da extensão à programação em lógica, usando a linguagem de programação em lógica PROLOG, no âmbito da representação de conhecimento imperfeito, recorrendo à utilização de valores nulos e da criação de mecanismos de raciocínio adequados.

Neste trabalho pretendeu-se desenvolver um sistema de representação de conhecimento e raciocínio com capacidade para caracterizar um universo de discurso na área da contratação pública para a realização contratos para a prestação de serviços.

2 Descrição do problema

Neste trabalho considerou-se que o panorama é caracterizado por conhecimento, por exemplo, dado na forma que se segue:

- adjudicante: #IdAd, Nome, NIF, Morada $\rightsquigarrow \{ \mathbb{V}, \mathbb{F}, \mathbb{D} \}$
- adjudicatária: #IdAda, Nome, NIF, Morada $\rightsquigarrow \{ \mathbb{V}, \mathbb{F}, \mathbb{D} \}$
- contrato: #IdAd, #IdAda, TipoDeContrato, TipoDeProcedimento, Descrição, Valor, Prazo, Local, Data $\rightsquigarrow \{ \mathbb{V}, \mathbb{F}, \mathbb{D} \}$

Figura 1: Caracterização do conhecimento.

Os contratos públicos seguem os procedimentos e normas previstas no Código dos Contratos Públicos. Neste trabalho considerou-se apenas as seguintes indicações: haver 3 tipos de procedimento, existir condições obrigatórias do contrato por ajuste directo e ainda a regra dos 3 anos válida para todos os contratos.

A partir desta caracterização e para a realização do trabalho, construiu-se um caso prático de aplicação dos conhecimentos, que foi capaz de demonstrar as funcionalidades subjacentes à programação em lógica estendida e à representação de conhecimento imperfeito, recorrendo à temática dos valores nulos.

A elaboração do caso prático foi implementada de forma a respeitar as necessidades de demonstração das seguintes funcionalidades tais como a representação do conhecimento positivo e negativo, a representação de conhecimento imperfeito, pela utilização de valores nulos de todos os tipos estudados, a manipulação dos invariantes que designem restrições à inserção e à remoção de conhecimento do sistema, lidar com a problemática da evolução do conhecimento, criando os procedimentos adequados e por fim desenvolver um sistema de inferência capaz de implementar os mecanismos de raciocínio inerentes a estes sistemas.

3 Descrição do Trabalho

3.1 Base de Conhecimento

O sistema de representação de conhecimento e raciocínio desenvolvido desenvolvido caracteriza uma área da contratação pública para a realização contratos para a prestação de serviços.

3.1.1 Adjudicante

Um adjudicante é caracterizado por um identificador (número inteiro único para cada adjudicante), um nome, um NIF e uma morada. O seguintes predicados representam adjudicantes:

```
adjudicante(1,'Universidade do Minho',502011378,'Portugal,Braga').
adjudicante(2,'Servicos de Acao Social da Universidade do Minho',680047360,'Portugal,Braga').
adjudicante(3,'Hospital Dr. Francisco Zagalo - Ovar',501510150,'Portugal,Aveiro,Ovar').
adjudicante(4,'Municipio de Barcelos',505584760,'Portugal,Braga,Barcelos').
adjudicante(5,'Municipio de Caminha',500843139,'Portugal,Viana do Castelo,Caminha').
adjudicante(6,'DGAM',600012662,'Portugal,Setubal,Almada').
adjudicante(7,'Municipio de Alto de Basto',705330336,'Portugal,Braga,Alto de Basto').
```

Figura 2: Predicado adjudicante.

3.1.2 Adjudicatária

Uma adjudicatária é caracterizada por um identificador (tal como o adjudicante), um nome, um NIF e uma morada. Os seguintes predicados representam adjudicatárias:

```
adjudicataria(1,'IDUNA COMERCIO E INDUSTRIA DE MOBILIARIO,S.A',503263869,'Portugal').
adjudicataria(2,'Diversey Portugal-Sistemas de Higiene e Limpeza Unipessoal,Lda',500086753,'Portugal').
adjudicataria(3,'SIEMENS HEALTHCARE DIAGNOSTICO LDA',507925173,'Portugal').
adjudicataria(4,'Lugar do Plano, Gestao do Territorio e Cultura,Lda',506378802,'Portugal').
adjudicataria(5,'Herculano Filipe Marvao Franco de Almeida',228686750,'Portugal').
adjudicataria(6,'BASE2 - INFORMATICA E TELECOMUNICACOES,LDA',501333401,'Portugal').
adjudicataria(7,'XXX - Associados - Sociedade de Advogados,SP,RL',702675112,'Portugal').
adjudicataria(9,'Kone Portugal Elevadores,Lda',506682048,'Portugal').
adjudicataria(10,'Newcoffee - industria torrefatora de cafes,SA',508348684,'Portugal').
```

Figura 3: Predicado adjudicatária.

3.1.3 Contrato

Um contrato é definido à custa de um identificador (tal como a adjudicatária e adjudicante), pelos números identificadores do adjudicante e da adjudicatária envolvidos no contrato realizado, o tipo de contrato, tipo de procedimento, por uma descrição, por um valor (obviamente este deve ser positivo), por um prazo, um local e uma data (DD,MM,AAAA). Os seguintes predicados representam exemplos de contratos realizados:

```
contrato(1,1,1,'Aquisicao de bens moveis','Consulta Previa', 'Aquisicao de cadeiras de escritorio para os varios servicos administrativos',28600,30,'Portugal,Braga',data(24,02,2020)).
contrato(2,2,2,'Aquisicao de bens moveis','Concurso Publico','Fornecimento de preparacoes para lavagem, por lotes, para o ano de 2020',6065.50,365,'Portugal,Braga',data(01,04,2020)).
contrato(3,3,3,'Aquisicao de bens moveis','Ajuste Direto','Reagentes de laboratorio',1950.50,347,'Portugal,Aveiro,Ovar',data(14,01,2020)).
contrato(4,4,4,'Aquisicao de servicos','Ajuste Direto','Aquisicao de servicos para elaboracao da estrategia local de habitacao em Barcelos',19200.00,120,'Portugal,Braga,Barcelos',data(13,03,2020)).
contrato(5,5,5,'Aquisicao de servicos','Consulta Previa','Prestacao de servicos de Assessoria Juridica',54000.00,1095,'Portugal,Viana do Castelo,Caminha',data(30,03,2020)).
contrato(6,6,6,'Aquisicao de servicos','Ajuste Direto','Assistencia a multifuncoes Lexmark',2016.00,5,'Portugal',data(10,03,2020)).
contrato(7,1,6,'Aquisicao de bens moveis','Concurso Publico','Aquisicao de equipamento informatico - DTSI e SCOM',3976.00,30,'Portugal,Braga,Braga',data(21,02,2020)).
contrato(8,7,7,'Aquisicao de servicos','Consulta Previa','Assessoria juridica',13599,547,'Alto de Basto',data(11,02,2020)).
```

Figura 4: Predicado contrato.

3.2 Adição e remoção de conhecimento

A inserção e remoção de conhecimento é realizada à custa de invariantes que especificam um conjunto de restrições que devem ser verdadeiras após uma adição/remoção de conhecimento. A inserção de predicados pode ser vista como uma evolução do sistema em termos de conhecimento. De um modo análogo, uma remoção de um predicado pode ser vista como uma involução do sistema. De notar que, tanto as operações de inserção como de remoção são sempre efectuadas. No entanto, após cada uma dessas operações é verificada a consistência do sistema com recurso aos invariantes. Caso alguma operação provoque uma anomalia no sistema, esta perde o seu efeito e o sistema volta ao estado anterior à operação em questão (semelhante à operação de rollback numa base de dados).

Antes mesmo de se descrever, com detalhe, os processos de evolução e de involução do sistema, assim como todos os invariantes desenvolvidos, faz sentido descrever dois predicados que são absolutamente cruciais para o correto funcionamento destes processos. Um destes predicados denomina-se *solucoes* e é caracterizado da seguinte forma: **solucoes(X,Y,Z) :- findall(X,Y,Z).**. Este predicado procura todas as ocorrências de "X" que satisfaçam "Y" e os resultados são guardados numa lista "Z".

Por sua vez, será necessário para os processos de evolução e involução no sistema, determinar o tamanho da lista resultante após a execução do predicado *solucoes*. Este processo é descrito pelo predicado *comprimento* que é definido da seguinte forma: **comprimento(S,N) :- length(S,N).**. Este predicado que dada uma lista "S"determina o seu comprimento em "N".

Através destes dois predicados fundamentais, pode-se então descrever os processos de evolução e involução do sistema, assim como os invariantes desenvolvidos para cada um dos predicados apresentados na secção **3.1**.

3.2.1 Evolução do sistema

O processo de evolução do sistema é caracterizado por um aumento de conhecimento no sistema. Este processo é descrito por:

evolucao(Termo) :- solucoes(Invariante,+Termo::Invariante,Lista), insercao(Termo), teste(Lista).

Em que o predicado insercao caracteriza-se pela inserção de conhecimento no sistema (à custa do predicado assert) e o predicado teste pelo teste aos invariantes relativos a esse "Termo" armazenados na lista. De notar que, por conveniência, definiu-se que um invariante relativo a uma inserção de um termo qualquer é precedido pelo símbolo de "+" no nome desse mesmo termo.

3.2.2 Involução do sistema

O processo de involução do sistema é muito semelhante ao processo de evolução e é definido do seguinte modo:

involucao(Termo) :- solucoes(Invariante,-Termo::Invariante,Lista), remocao(Termo), teste(Lista).

Em que o predicado remocao caracteriza-se pela inserção de conhecimento no sistema (à custa do predicado retract) e o predicado teste pelo teste aos invariantes relativos a esse "Termo" armazenados na lista. De notar que, por conveniência, definiu-se que um invariante relativo a uma remoção de um termo qualquer é precedido pelo símbolo de "-" no nome desse mesmo termo.

3.2.3 Adjudicante

A inserção de conhecimento relativa a um adjudicante deve respeitar os seguintes critérios: o identificador de um adjudicante dever ser inteiro e maior que 0 e o NIF deve possuir um valor inteiro pertencente ao intervalo [100000000,999999999]. A inserção só é permitida se não existir conhecimento de uma adjudicante com o mesmo número de identificação.

```
+adjudicante( IdAd,Nome,NIF,Morada ) :: ((solucoes(IdAd, adjudicante( IdAd,Nome,NIF,Morada ), U), comprimento(U,N), N==1),
                                          integer(IdAd), integer(NIF), IdAd>0, NIF=<999999999, NIF>=100000000).
```

Por sua vez, a remoção de um adjudicante não deve ser permitida quando ainda existirem contratos realizados relativos ao mesmo:

```
-adjudicante( IdAd,Nome,NIF,Morada ) :: (solucoes( IdAd,contrato( _,IdAd,_,_,_,_,_,_,_ ),S1 ), comprimento( S1,0 )).
```

3.2.4 Adjudicatária

A inserção de conhecimento relativa a uma adjudicatária deve respeitar os seguintes critérios: o identificador de uma adjudicatária dever ser inteiro e maior que 0 e o NIF deve possuir um valor inteiro pertencente ao intervalo [100000000,999999999]. A inserção só é permitida se não existir conhecimento de uma adjudicatária com o mesmo número de identificação.

```
+adjudicataria( IdAda,Nome,NIF,Morada ) :: ((solucoes(IdAda, adjudicataria( IdAda,Nome,NIF,Morada ), U), comprimento(U,N), N==1),
                                            integer(IdAda), integer(NIF), IdAda>0, NIF=<999999999, NIF>=100000000).
```

Por sua vez, a remoção de uma adjudicatária não deve ser permitida quando ainda existirem contratos realizados relativos ao mesmo:

```
-adjudicataria( IdAda,Nome,NIF,Morada ) :: (solucoes( IdAda,contrato( _,_,IdAda,_,_,_,_,_,_ ),S1 ), comprimento( S1,0 )).
```

3.2.5 Contrato

Relativamente a um contrato por ajuste directo deve respeitar as seguintes regras: o valor deve ser um número pertencente ao intervalo [0,5000], o prazo deve também ser um número, o segundo argumento deve corresponder a um adjudicante já presente tal como o terceiro argumento deve corresponder a uma adjudicatária já presente. A inserção só é permitida se não existir conhecimento de um contrato com o mesmo número de identificação, o tipo de contrato for contrato de aquisição, locação de bens moveis ou aquisição de serviços e ainda o prazo de vigência ter uma duração de um ano, o equivalente a 364 ou 365 dias mediante o ano em causa.

```
validaContratoAjusteDireto( 'Contrato de aquisicao' ).
validaContratoAjusteDireto( 'Locacao de Bens moveis' ).
validaContratoAjusteDireto( 'Aquisicao de servicos' ).

validaPrazoVigencia( D,M,A,P ) :- A mod 4 =:= 0, P=<365. %>
validaPrazoVigencia( D,M,A,P ) :- A mod 4 =\= 0, P=<364. %>
validaPrazoVigencia( data( D,M,A ),P ) :- validaPrazoVigencia( D,M,A,P ).

+contrato(IdC,IdAd,IdAda,TipoDeContrato,'Ajuste Direto',Descricao,Valor,Prazo,Local,Data) :: (number(Valor), Valor =< 5000, Valor >= 0, number(Prazo),
                                                                                              solucoes( IdC, contrato( IdC,_,_,_,_,_,_,_,_,_ ), S0), comprimento(S0,N0), N0==1,
                                                                                              solucoes( IdAd, adjudicante( IdAd,_,_,_ ), S1), comprimento( S1,N1 ), N1==1,
                                                                                              solucoes( IdAda, adjudicataria( IdAda,_,_,_ ), S2), comprimento( S2,N2 ), N2==1,
                                                                                              validaContratoAjusteDireto( TipoDeContrato ), validaPrazoVigencia( Data,Prazo )).
```

Em relação à regra dos 3 anos válida para todos os contratos definiu-se o seguinte invariante que permite validar que uma entidade adjudicante não pode convidar a mesma empresa para celebrar um contrato com prestações de serviço do mesmo tipo ou idênticas às de contratos que já lhe foram atribuídos, no ano económico em curso e nos dois anos económicos anteriores, sempre que o preço contratual acumulado dos contratos já celebrados (não incluindo o contrato que se pretende celebrar) seja igual ou superior a 75.000 euros.

```
doisanteriores( A,B ) :- A>=B , B>=A-2.

+contrato( IdC,IdAd,IdAda,TipoDeContrato,TipoDeProcedimento,Descricao,Valor,Prazo,Local,data( D,M,A ) )  :: (
                                        solucoes( V,contrato( IdC,IdAd,IdAda,_,TipoDeProcedimento,_,V,_,_,data( D2,M2,A2 )), doisanteriores( A,A2 )),S1),
                                        somaN(S1,Total), Total-Valor =< 75000).
```

A inserção de conhecimento relativa a um contrato na regra geral deve respeitar os seguintes critérios: o valor deve ser um número maior que 0, a data fornecida deve ser válida, o prazo deve ser também um número, o segundo argumento deve corresponder a um adjudicante já presente tal como o terceiro argumento deve corresponder a uma adjudicatária já presente. A inserção só é permitida se não existir conhecimento de uma contrato com o mesmo número de contrato.

```
+contrato( IdC,IdAd,IdAda,TipoDeContrato,TipoDeProcedimento,Descricao,Valor,Prazo,Local,Data ) :: ( number(Valor), Valor >= 0, number(Prazo), data(Data),
                                                                                                    solucoes( IdAd, adjudicante( IdAd,_,_,_ ), S1 ), comprimento( S1,1 ),
                                                                                                    solucoes( IdAda, adjudicataria(IdAda,_,_,_ ), S2 ), comprimento( S2,1 ),
                                                                                                    solucoes( IdC, contrato( IdC,_,_,_,_,_,_,_,_,_ ), S3 ), comprimento( S3,0 )).
```

Após a remoção de um contrato deve ser garantido que não haja conhecimento do mesmo predicado, ou seja:

```
-contrato( IdC,IdAd,IdAda,TipoDeContrato,TipoDeProcedimento,Descricao,Valor,Prazo,Local,Data ) :: ( solucoes(IdC, contrato( IdC,_,_,_,_,_,_,_,_,_ ), S), comprimento( S,1 )).
```

3.3 Sistema de inferência

Por oposição ao conhecimento positivo, considerou-se a criação de um novo tipo de conhecimento, o conhecimento negativo. Os nomes dos predicados que representam este tipo de conhecimento foram precedidos pela etiqueta "-".

De modo a se poder demonstrar a veracidade de um determinado termo criou-se uma espécie de sistema de inferência que através de um determinado termo devolve o valor da sua veracidade. No entanto, apenas se considera um determinado termo como sendo falso quando se informa do mesmo à base de conhecimento através de um conhecimento negativo. Esta característica é traduzido pelo conhecimento do tipo desconhecido no qual não existe nenhuma informação.

O demonstrador base que permite avaliar um qualquer termo segundo as restrições impostas é o seguinte:

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

Este sistema apresenta no entanto algumas restrições, como por exemplo, não permitir avaliar um conjunto de questões. Deste modo, decidiu-se criar dois sistemas complementares. Um destes é o seguintes:

```
demo( Q1,e,Q2,F )   :- demo( Q1,F1 ), demo( Q2,F2 ), conjuncao( F1,F2,F ).
demo( Q1,ou,Q2,F )  :- demo( Q1,F1 ), demo( Q2,F2 ), disjuncao( F1,F2,F ).
demo( Q1,impl,Q2,F ) :- demo( Q1,F1 ), demo( Q2,F2 ), implicacao( F1,F2,F ).
demo( Q1,eq,Q2,F )  :- demo( Q1,F1 ), demo( Q2,F2 ), equivalencia( F1,F2,F ).


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

disjuncao( verdadeiro,X,verdadeiro ).
disjuncao( X,verdadeiro,verdadeiro ).
disjuncao( Y,desconhecido,desconhecido ) :- Y \= verdadeiro.
disjuncao( desconhecido,Y,desconhecido ) :- Y \= verdadeiro.
disjuncao( falso,falso,falso ).

implicacao( X,verdadeiro,verdadeiro ).
implicacao( falso,X,verdadeiro ).
implicacao( desconhecido,X,desconhecido ) :- X \= verdadeiro.
implicacao( verdadeiro,desconhecido,desconhecido ).
implicacao( verdadeiro,falso,falso ).

equivalencia( X,X,verdadeiro ).
equivalencia( X,desconhecido,desconhecido ).
equivalencia( desconhecido,Y,desconhecido ).
equivalencia( X,Y,falso ) :- X \= Y.
```

Este sistema recebe como argumentos duas questões (Q1 e Q2), uma flag F que representa o resultado e um tipo. Este tipo pode representar uma conjunção, uma disjunção, implicação ou equivalência. Para cada um dos destes tipos é usado o demonstrador base para determinar a veracidade das duas questões.

Além deste criou-se ainda um sistema que recebe como argumento um conjunto qualquer de questões e produz como resultado o conjunto de resultados obtidos Às mesmas questões.

```
demoList( [],[] ).

demoList( [Q|L1],[R|L2] ) :- demo( Q,R ), demoList( L1,L2 ).
```

3.4 Representação de conhecimento imperfeito

Aplicando o Pressuposto do Mundo Fechado (PMF), em que toda a informação que não existe mencionada na base de dados é considerada falsa, aos predicados previamente apresentados:

```
-adjudicante( IdAd,Nome,NIF,Morada ) :- nao( adjudicante( IdAd,Nome,NIF,Morada )), nao( excecao( adjudicante( IdAd,Nome,NIF,Morada ))).

-adjudicataria( IdAda,Nome,NIF,Morada ) :- nao( adjudicataria( IdAd,Nome,NIF,Morada )), nao( excecao( adjudicataria( IdAd,Nome,NIF,Morada ))).

-contrato( IdC,IdAd,IdAda,TipoDeContrato,TipoDeProcedimento,Descricao,Valor,Prazo,Local,Data ) :-
                    nao( contrato( IdC,IdAd,IdAda,TipoDeContrato,TipoDeProcedimento,Descricao,Valor,Prazo,Local,Data )),
                     nao( excecao( contrato( IdC,IdAd,IdAda,TipoDeContrato,TipoDeProcedimento,Descricao,Valor,Prazo,Local,Data ) )).
```

3.4.1 Tipo incerto

Este tipo de conhecimento diz respeito a situações em que não se conhece o valor de um determinado campo. Ou seja, casos em que parte da informação é desconhecida e está dentro de um conjunto indeterminado de hipóteses.

Por exemplo, se existir um adjudicante cuja morada é desconhecida, este conhecimento é facilmente descrito da seguinte forma, através do uso de uma variável que demarque o campo que não se conhece e da inserção de uma excepção associada.

```
adjudicante(9,'Instituto Nacional de Emergencia Medica, I.P.',501356126,morada_desconhecida).

excecao(adjudicante( IdAd,Nome,NIF,Morada )) :- adjudicante( IdAd,Nome,NIF,morada_desconhecida ).
```

No exemplo anterior, existe um adjudicante. cujo ID é 9, Nome é "Instituto Nacional de Emergencia Medica, I.P.", NIF é 501356126 mas desconhece-se a morada. Então, utilizou-se uma variável *morada_desconhecida* para representar o conhecimento imperfeito do tipo incerto associado ao campo "Morada". Posteriormente, bastou indicar que o adjudicante cujo campo "Morada"fosse *morada_desconhecida* correspondesse a uma excepção do próprio predicado adjudicante.

Um outro exemplo deste tipo de conhecimento é o desconhecimento do NIF mas sabendo que não é um determinado NIF, este conhecimento é descrito da seguinte forma:

```
adjudicante(8,'Santa Casa da Misericordia de Lisboa',nif_desconhecido,'Portugal,Lisboa').

-adjudicante(8,'Santa Casa da Misericordia de Lisboa',500745471,'Portugal,Lisboa').

excecao(adjudicante( IdAd,Nome,NIF,Morada )) :- adjudicante( IdAd,Nome,nif_desconhecido,Morada ).
```

No exemplo anterior, existe um adjudicante. cujo ID é 8, Nome é "Santa Casa da Misericordia de Lisboa"e Morada é "Portugal,Lisboa"mas desconhece-se o NIF. Então, utilizou-se uma variável *nif_desconhecida* para representar o conhecimento imperfeito do tipo incerto associado ao campo "NIF"tal como o exemplo anterior contudo acrescentou-se à base de dados que o adjudicante em causa o campo NIF não é 500745471 através da etiqueta -"associado ao predicado.

3.4.2 Tipo impreciso

Este tio de conhecimento imperfeito diz respeito a situações em que existe informação desconhecida que está dentro de um conjunto determinado de valores. Para este tipo, distinguimos duas variantes: uma variante correspondente a um valor do tipo impreciso cujo conjunto de hipóteses é totalmente discriminado; e outra variante também correspondente a um valor do tipo impreciso mas cujo conjunto de hipóteses é dado por uma gama de valores.

Conjunto de hipóteses discriminado: Esta variante corresponde, por exemplo; à existência de uma adjudicatária em que não haja certeza acerca da sua morada, sabendo-se no entanto que é uma de três possíveis. Esta situação está descrita no exemplo seguinte:

```
excecao( adjudicataria(8,'Mario Goncalves, Lda',500183872,'Portugal,Braga,Braga')).

excecao( adjudicataria(8,'Mario Goncalves, Lda',500183872,'Portugal,Porto,Porto')).

excecao( adjudicataria(8,'Mario Goncalves, Lda',500183872,'Portugal,Lisboa,Lisboa')).
```

Não se sabe se o adjudicatário "Mario Goncalves"tem sede em "Portugal,Braga,Braga"ou em "Portugal,Porto"ou ainda em "Portugal,Lisboa,Lisboa". Assim sendo, este conhecimento imperfeito do tipo impreciso e indicado com o auxílio de uma execução para cada umas das possibilidades.

Gama de valores: Uma situação em que, por exemplo, não se tenha a certeza acerca do valor de um contrato mas em que se saiba que este valor está situado entre um intervalo de valores, corresponde a este tipo de conhecimento imperfeito impreciso. Este conhecimento é denotado como uma excepção em que um dos campos é menor que um determinado valor e maior do que outro determinado valor.

```
contrato(10,2,10,'Aquisicao de bens moveis','Concurso Publico','Aquisicao de cafe, por lotes, para os anos de 2020 e 2021',valor_desconhecido,353,'Portugal,Braga',data(13,01,2020)).

excecao(contrato(10,2,10,'Aquisicao de bens moveis','Concurso Publico','Aquisicao de cafe, por lotes, para os anos de 2020 e 2021',Valor,353,'Portugal,Braga',data(13,01,2020))) :- aproximadamente(Valor,1000).

aproximadamente( X,Y ) :- W is 0.90 * Y, Z is 1.10 * Y, X =< Z, X >= W.
```

Por exemplo, no caso apresentado acima, não se conhece o valor do contrato cujo ID é 10, cujo identificador do adjudicante é o 2, o identificador da adjudicatária é o 10, o tipo de contrato é "Aquisicao de bens moveis", em que o procedimento é do tipo "Concurso Publico", referente a "Aquisicao de cafe, por lotes, para os anos de 2020 e 2021", com prazo de 353 dias, feito em "Portugal,Braga"no dia 13 de Janeiro de 2020 cujo valor é desconhecido. Com a introdução de uma nova excepção, pôde-se indicar o conhecimento perfeito relativo aos campos, e, como se sabe que a contrato teve um valor entre 900 e 1100 euros, pôde-se também limitar o seu campo Valor adequadamente.

3.4.3 Tipo interdito

Este tipo de conhecimento imperfeito diz respeito a situações em que existe informação desconhecida mas em que se sabe que é impossível vir a conhecer tal informação. Consequentemente, o conhecimento deste tipo nunca poderá ser evoluído. Assim, foi necessário a criação de um predicado *nulo*, que serviu para identificar informação deste tipo.

```
contrato(9,2,9,'Aquisicao de servicos','Concurso Publico',descricao_interdita,30024,1096,'Portugal,Braga',data(14,02,2020)).

excecao(contrato( IdC,IdAd,IdAda,TipoDeContrato,TipoDeProcedimento,Descricao,Valor,Prazo,Local,Data )) :-
                    contrato( IdC,IdAd,IdAda,TipoDeContrato,TipoDeProcedimento,descricao_interdita,Valor,Prazo,Local,Data).

nulo(descricao_interdita) :- contrato(9,2,9,'Aquisicao de servicos','Concurso Publico',Descricao,30024,1096,'Portugal,Braga',data(14,02,2020)), nao(nulo(Morada)).
```

3.5 Implementação dos predicados de consulta

Após a criação dos predicados que permitem a evolução e a involução de conhecimento do sistema, assim como o desenvolvimento dos demais invariantes, procedeu-se então à criação de predicados de consulta sobre a estrutura de conhecimento. Estes predicados foram maioritariamente desenvolvidos à custa do predicado *solucoes*, sendo que em alguns casos recorreu-se a alguns predicados auxiliares.

3.5.1 Registar adjudicantes, adjudicatárias ou contratos

Para que seja possível a inserção de nova informação na base de conhecimento usou-se o teorema evolucao o qual insere na base de conhecimento de forma controlada:

```
registaAdjudicante( IdAd,Nome,NIF,Morada ) :- evolucao( adjudicante( IdAd,Nome,NIF,Morada )).

registaAdjudicataria( IdAda,Nome,NIF,Morada ) :- evolucao( adjudicataria( IdAda,Nome,NIF,Morada )).

registaContrato( IdC,IdAd,IdAda,TipoDeContrato,TipoDeProcedimento,Descricao,Valor,Prazo,Local,Data ) :- evolucao( contrato( IdC,IdAd,IdAda,TipoDeContrato,TipoDeProcedimento,Descricao,Valor,Prazo,Local,Data )).
```

3.5.2 Remover adjudicantes, adjudicatárias ou contratos

Para que seja possível a remoção de nova informação na base de conhecimento usou-se o teorema involucao o qual insere na base de conhecimento de forma controlada:

```
removeAdjudicante( IdAd ) :- involucao( adjudicante( IdAd,_,_,_ )).

removeAdjudicataria( IdAda ) :- involucao( adjudicataria( IdAda,_,_,_ )).

removeContrato( IdC,IdAd,IdAda,TipoDeContrato,TipoDeProcedimento,Descricao,Valor,Prazo,Local,Data ) :- involucao( contrato( IdC,_,_,_,_,_,_,_,_,_ )).
```

3.5.3 Procurar adjudicantes ou adjudicatárias

Esta consulta permite obter adjudicantes ou adjudicatárias pelos seguintes critérios: identificador, nome, NIF ou morada. Os resultados são devolvidos numa lista. Os predicados que traduzem esta consulta são os seguintes:

```
adjudicanteID( IdAd,R ) :- solucoes( adjudicante( IdAd,Nome,NIF,Morada), adjudicante( IdAd,Nome,NIF,Morada),[R|_]).
adjudicanteNOME( Nome,R ) :- solucoes( adjudicante( IdAd,Nome,NIF,Morada), adjudicante( IdAd,Nome,NIF,Morada),R).
adjudicanteNIF( NIF,R ) :- solucoes( adjudicante( IdAd,Nome,NIF,Morada), adjudicante( IdAd,Nome,NIF,Morada),R).
adjudicanteMORADA( Morada,R ) :- solucoes( adjudicante( IdAd,Nome,NIF,Morada), adjudicante( IdAd,Nome,NIF,Morada),R).

adjudicatariaID( IdAda,R ) :- solucoes( adjudicataria( IdAda,Nome,NIF,Morada), adjudicataria( IdAda,Nome,NIF,Morada),[R|_]).
adjudicatariaNOME( Nome,R ) :- solucoes( adjudicataria( IdAda,Nome,NIF,Morada), adjudicataria( IdAda,Nome,NIF,Morada),R).
adjudicatariaNIF( NIF,R ) :- solucoes( adjudicataria( IdAda,Nome,NIF,Morada), adjudicataria( IdAda,Nome,NIF,Morada),R).
adjudicatariaMORADA( Morada,R ) :- solucoes( adjudicataria( IdAda,Nome,NIF,Morada), adjudicataria( IdAda,Nome,NIF,Morada),R).
```

3.5.4 Procurar contratos

O raciocínio aplicado na definição deste predicado foi semelhante ao anterior contudo apenas procuramos pelo identificador.

```
contratoID( IdC,R ) :- solucoes( contrato( IdC,IdAd,IdAda,TipoDeContrato,TipoDeProcedimento,Descricao,Valor,Prazo,Local,Data ),
                                 contrato( IdC,IdAd,IdAda,TipoDeContrato,TipoDeProcedimento,Descricao,Valor,Prazo,Local,Data),[R|_]).
```

3.5.5 Identificar adjudicantes ou adjudicatárias

Neste predicado apenas é fornecido todos adjudicantes ou adjudicatárias dependente do predicado que tenham contratos realizados, utilizou-se um predicado auxiliar que permite remover as entidades repetidas da lista devolvida pelo predicado solucoes:

```
adjudicante_contrato( R1 ) :- solucoes(adjudicante( IdAd,Nome,NIF,Morada ), (adjudicante( IdAd,Nome,NIF,Morada ),contrato( _,IdAd,_,_,_,_,_,_,_)),R ),
        apagaRep(R,R1).

adjudicataria_contrato( R1 ) :- solucoes(adjudicataria( IdAda,Nome,NIF,Morada ),
                                 (adjudicataria( IdAda,Nome,NIF,Morada ),contrato( _,_,IdAda,_,_,_,_,_,_)),R ), apagaRep(R,R1).
```

3.5.6 Identificar os contratos em que um adjudicante ou adjudicatária tenha realizado

Neste predicado identifica contratos que um adjudicante ou adjudicatária encontra-se envolvido:

```
contrato_adjudicante( IdAd,R ) :- solucoes( contrato( IdC,IdAd,IdAda,TipoDeContrato,TipoDeProcedimento,Descricao,Valor,Prazo,Local,Data ),
                (adjudicante(IdAd,Nome,NIF,Morada ), contrato( IdC,IdAd,IdAda,TipoDeContrato,TipoDeProcedimento,Descricao,Valor,Prazo,Local,Data)),R).

contrato_adjudicataria( IdAda,R ) :- solucoes( contrato( IdC,IdAd,IdAda,TipoDeContrato,TipoDeProcedimento,Descricao,Valor,Prazo,Local,Data ),
                (adjudicataria(IdAda,Nome,NIF,Morada ), contrato( IdC,IdAd,IdAda,TipoDeContrato,TipoDeProcedimento,Descricao,Valor,Prazo,Local,Data)),R).
```

3.5.7 Identificar os adjudicantes de uma adjudicatária, vice-versa

Neste predicado mostramos todos os adjudicantes de uma adjudicatária, isto é, dado uma determinado identificador de uma adjudicatária é apresentado todos os adjudicantes que fizeram contratos com esta adjudicatária, criou-se o predicado de ser fornecido o identificador de um adjudicante e retornar todos os adjudicatárias.

```
adjudicante_de_adjudicataria(IdAda,R) :- solucoes(adjudicante(IdAd,Nome,NIF,Morada), (contrato( _,IdAd,IdAda,_,_,_,_,_,_ ),
                                         adjudicataria( IdAda,_,_,_),adjudicante(IdAd,Nome,NIF,Morada)),R1), apagaRep(R1,R).

adjudicataria_de_adjudicante(IdAd,R) :- solucoes(adjudicataria(IdAda,Nome,NIF,Morada), (contrato( _,IdAd,IdAda,_,_,_,_,_,_ ),
                                         adjudicataria( IdAda,Nome,NIF,Morada ),adjudicante(IdAd,_,_,_)),R1), apagaRep(R1,R).
```

3.5.8 Identificar contratos pelos seus parâmetros

Nestes predicados mostramos os contratos que tenham sido realizados num determinado local, pelo tipo de contrato ou pelo tipo de procedimento, respectivamente:

```
contratos_por_local( Local,R ) :- solucoes(contrato( IdC,IdAd,IdAda,TipoDeContrato,TipoDeProcedimento,Descricao,Valor,Prazo,Local,Data),
                                    contrato(IdC,IdAd,IdAda,TipoDeContrato,TipoDeProcedimento,Descricao,Valor,Prazo,Local,Data ), R).

contratos_por_tipo( TipoDeContrato,R ) :- solucoes(contrato( IdC,IdAd,IdAda,TipoDeContrato,TipoDeProcedimento,Descricao,Valor,Prazo,Local,Data),
                                    contrato( IdC,IdAd,IdAda,TipoDeContrato,TipoDeProcedimento,Descricao,Valor,Prazo,Local,Data ), R).

contratos_por_procedimento( TipoDeProcedimento,R ) :- solucoes(contrato( IdC,IdAd,IdAda,TipoDeContrato,TipoDeProcedimento,Descricao,Valor,Prazo,Local,Data),
                                    contrato( IdC,IdAd,IdAda,TipoDeContrato,TipoDeProcedimento,Descricao,Valor,Prazo,Local,Data ), R).
```

3.5.9 Calcular o valor total pago/facturado por adjudicante ou adjudicatária

Nestes dois predicados apresentamos o valor total pago por um dado adjudicante ou facturado por uma determinada adjudicatária mediante o valor presente nos contratos que realizaram, este predicado necessitou de um outro predicado auxiliar que calcula o valor total da lista retornada pelo predicado solucoes, assim sendo é descrito da seguinte forma:

```
pago_adjudicante( IdAd,R ) :- solucoes(Valor, contrato( _,IdAd,_,_,_,_,Valor,_,_,_ ),R1), custo_total(R1,R).

facturado_adjudicataria( IdAda,R ) :- solucoes(Valor, contrato( _,_,IdAda,_,_,_,Valor,_,_,_ ),R1), custo_total(R1,R).
```

3.5.10 Numero total de entidades

Nestes predicados apresentamos o número total de adjudicantes, adjudicatárias ou contratos presentes na base de conhecimento, respectivamente, com o auxilio do predicado comprimento previamente apresentado.

```
total_adjudicante(R) :- solucoes(IdAd, adjudicante(IdAd,_,_,_), L), comprimento(L,R).

total_adjudicataria(R) :- solucoes(IdAda, adjudicataria(IdAda,_,_,_), L), comprimento(L,R).

total_contratos(R) :- solucoes(IdC,contrato( IdC,_,_,_,_,_,_,_,_,_ ),L), comprimento(L,R).
```

4 Conclusão

O presente relatório descreveu, de forma sucinta, o desenvolvimento do sistema de representação de conhecimento e raciocínio desenvolvido com capacidade para caracterizar um universo de discurso na área da contratação pública para a realização contratos para a prestação de serviços.

Após a realização deste trabalho, ficamos conscientes das potencialidades que a programação em lógica desenvolvida com PROLOG na medida que permite representar o conhecimento que um agente tem sobre o mundo de uma forma simples e directa, em uma linguagem de alto nível, tornando os programas mais compactos, flexíveis e inteligíveis e como também permite uma programação declarativa apenas basta especificar correctamente o problema que o motor de inferência encarrega-se de descobrir como obter sua solução.

Consideramos que os principais objectivos foram cumpridos, contudo temos consciência que poderíamos ter explorado melhor a problemática da evolução do conhecimento negativo.

Sentimos que a realização deste trabalho prático consolidou os nossos conhecimentos na utilização da extensão à programação em lógica, na linguagem de programação em lógica PROLOG como também permitiu utilizar valores nulos para representar conhecimento imperfeito e criar mecanismos de raciocínio adequados.