



Universidade do Minho

# CONTRATAÇÃO PÚBLICA

MESTRADO INTEGRADO EM ENGENHARIA  
INFORMÁTICA

SISTEMAS DE REPRESENTAÇÃO DE CONHECIMENTO E  
RACIOCÍNIO

*84656 - Hugo Cunha*

*86268 - Maria Pires*

*84167 - Susana Marques*

3 DE MAIO DE 2020

## **Resumo**

O primeiro trabalho prático no âmbito da unidade curricular *Sistemas de Representação de Conhecimento e Raciocínio* resume-se na construção de um caso prático de aplicação dos conhecimentos, que seja capaz de demonstrar as funcionalidades subjacentes à programação em lógica estendida e à representação de conhecimento imperfeito, recorrendo à temática dos valores nulos.

Deste modo, primeiro será efetuada uma introdução ao projeto, e de seguida é explicado toda a base de conhecimento usada e os tipos de conhecimento representado.

São ainda descritos os invariantes universais, de contrato e das entidades adjudicatária e adjudicante criados, seguidos pela evolução e involução do conhecimento.

Serão também ilustrados predicados auxiliares que serviram de ajuda para a construção do caso prático e a representação do sistema de inferência. Finalmente, apresentaremos uma apreciação crítica ao trabalho desenvolvido.

# Conteúdo

<b>1</b>	<b>Introdução</b>	<b>5</b>
<b>2</b>	<b>Descrição do Trabalho e Análise de Resultados</b>	<b>6</b>
2.1	Base de conhecimento . . . . .	7
2.1.1	Conhecimento Perfeito . . . . .	7
2.1.2	Conhecimento Imperfeito Incerto . . . . .	8
2.1.3	Conhecimento Imperfeito Impreciso . . . . .	9
2.1.4	Conhecimento Imperfeito Interdito . . . . .	10
2.2	Invariante s . . . . .	12
2.2.1	Invariante s universais . . . . .	12
2.2.2	Invariante s de Contrato . . . . .	12
2.2.3	Invariante s da Entidade Adjudicante . . . . .	15
2.2.4	Invariante s da Entidade Adjudicatária . . . . .	16
2.3	Evolução e Involução do conhecimento . . . . .	18
2.3.1	Evolução do conhecimento perfeito positivo e negativo . .	18
2.3.2	Evolução do conhecimento imperfeito incerto . . . . .	19
2.3.3	Evolução do conhecimento imperfeito impreciso . . . . .	20
2.3.4	Evolução do conhecimento imperfeito interdito . . . . .	21
2.3.5	Involução do conhecimento perfeito positivo e negativo . .	22
2.3.6	Involução do conhecimento imperfeito incerto . . . . .	23
2.3.7	Involução do conhecimento imperfeito impreciso . . . . .	24
2.3.8	Involução do conhecimento imperfeito interdito . . . . .	25
2.4	Predicados Auxiliares . . . . .	26
2.5	Sistema de Inferência . . . . .	29
<b>3</b>	<b>Conclusão</b>	<b>31</b>
<b>4</b>	<b>Referências Bibliográficas e Electrónicas</b>	<b>32</b>

## **Lista de Figuras**

2.1 Exemplos de factos perfeitos positivos da base de conhecimento . . . . .	7
2.2 Exemplos de factos perfeitos negativos da base de conhecimento . . . . .	8
2.3 PMF para os predicados existentes . . . . .	8
2.4 Exemplos de conhecimento imperfeito incerto da base de conhecimento . . . . .	9
2.5 Exceções de conhecimento imperfeito incerto . . . . .	9
2.6 Exemplos de conhecimento imperfeito impreciso da base de conhecimento . . . . .	10
2.7 Exemplos de conhecimento imperfeito interditado da base de conhecimento . . . . .	11
2.8 Exceções ao conhecimento imperfeito interditado . . . . .	11
2.9 Invariante universais para cada tipo de conhecimento . . . . .	12
2.10 Invariante de contrato . . . . .	12
2.11 Invariante de contrato . . . . .	13
2.12 Invariante de contrato . . . . .	13
2.13 Invariante de contrato . . . . .	13
2.14 Invariante de contrato . . . . .	13
2.15 Invariante de contrato . . . . .	14
2.16 Invariante de contrato . . . . .	14
2.17 Invariante de contrato . . . . .	14
2.18 Invariante de contrato . . . . .	15
2.19 Invariante de contrato . . . . .	15
2.20 Invariante de contrato . . . . .	15
2.21 Invariante de uma entidade adjudicante . . . . .	15
2.22 Invariante de uma entidade adjudicante . . . . .	16
2.23 Invariante de uma entidade adjudicante . . . . .	16
2.24 Invariante de uma entidade adjudicante . . . . .	16
2.25 Invariante de uma entidade adjudicatária . . . . .	16
2.26 Invariante de uma entidade adjudicatária . . . . .	17
2.27 Invariante de uma entidade adjudicatária . . . . .	17
2.28 Invariante de uma entidade adjudicatária . . . . .	17
2.29 Evolução do conhecimento . . . . .	18
2.30 Exemplos de evolução de conhecimento perfeito positivo . . . . .	18
2.31 Exemplos de evolução de conhecimento perfeito negativo . . . . .	18
2.32 Exemplos de evolução de conhecimento perfeito positivo . . . . .	19
2.33 Predicado de evolução do conhecimento imperfeito incerto de contrato . . . . .	19
2.34 Predicado de evolução do conhecimento imperfeito incerto de uma entidade adjudicante . . . . .	19

2.35	Predicado de evolução do conhecimento imperfeito incerto de uma entidade adjudicatária . . . . .	19
2.36	Exemplos de evolução de conhecimento imperfeito incerto . . . . .	20
2.37	Predicado de evolução do conhecimento imperfeito impreciso para contrato . . . . .	20
2.38	Predicado de evolução do conhecimento imperfeito impreciso de uma entidade adjudicante . . . . .	20
2.39	Predicado de evolução do conhecimento imperfeito impreciso de uma entidade adjudicatária . . . . .	20
2.40	Exemplos de evolução de conhecimento imperfeito incerto . . . . .	21
2.41	Predicados de evolução de conhecimento imperfeito incerto para um intervalo de valores . . . . .	21
2.42	Exemplos de evolução de conhecimento imperfeito incerto para um intervalo de valores . . . . .	21
2.43	Predicado de evolução de conhecimento imperfeito interdito para um contrato . . . . .	21
2.44	Predicado de evolução de conhecimento imperfeito interdito para uma entidade adjudicante . . . . .	22
2.45	Predicado de evolução de conhecimento imperfeito interdito para uma entidade adjudicatária . . . . .	22
2.46	Exemplos de evolução de conhecimento imperfeito interdito . . . . .	22
2.47	Involução do conhecimento . . . . .	22
2.48	Exemplos de involução de conhecimento perfeito positivo . . . . .	23
2.49	Exemplos de involução de conhecimento perfeito positivo . . . . .	23
2.50	Predicado de involução de conhecimento imperfeito incerto . . . . .	23
2.51	Predicado de involução de conhecimento imperfeito incerto . . . . .	23
2.52	Predicado de involução de conhecimento imperfeito incerto . . . . .	23
2.53	Exemplo de involução de conhecimento imperfeito incerto . . . . .	24
2.54	Predicado de involução de conhecimento imperfeito impreciso . . . . .	24
2.55	Predicado de involução de conhecimento imperfeito impreciso . . . . .	24
2.56	Predicado de involução de conhecimento imperfeito impreciso . . . . .	24
2.57	Exemplo de involução de conhecimento imperfeito impreciso . . . . .	24
2.58	Predicado de involução de conhecimento imperfeito interdito . . . . .	25
2.59	Predicado de involução de conhecimento imperfeito interdito . . . . .	25
2.60	Predicado de involução de conhecimento imperfeito interdito . . . . .	25
2.61	Exemplo de involução de conhecimento imperfeito interdito . . . . .	25
2.62	Predicado 'nao' . . . . .	26
2.63	Predicado 'comprimento' . . . . .	26
2.64	Predicado 'soluções' . . . . .	26
2.65	Predicados de inserção e remoção de conhecimento . . . . .	27
2.66	Predicados 'teste' . . . . .	27
2.67	Predicado 'nifValido' . . . . .	27
2.68	Predicado 'valorValido' . . . . .	27
2.69	Predicados de validação de uma data . . . . .	28
2.70	Predicado 'menos3Anos' . . . . .	28
2.71	Predicado 'sumVals' . . . . .	28
2.72	Sistema de inferência simples . . . . .	29
2.73	Exemplificação da utilização do predicado demo . . . . .	29
2.74	Sistema de inferência com uma lista de questões . . . . .	29
2.75	Exemplificação da utilização do predicado demoList . . . . .	29

2.76 Predicado conjunção . . . . .	30
2.77 Predicado disjunção . . . . .	30
2.78 Conjunção/disjunção com uma lista de questões . . . . .	30
2.79 Exemplificação da utilização do predicado query . . . . .	30

# 1 | Introdução

Este primeiro trabalho consiste no desenvolvimento de um sistema para a caracterização de um universo de discurso na área da contratação pública através do paradigma da programação em lógica e o motor de inferência PROLOG. Este sistema tem como requisitos a implementação de várias funcionalidades, apresentadas ao longo deste relatório, que permitem representar conhecimento perfeito positivo, negativo e os vários tipos de conhecimento imperfeito, manipular invariantes que restringem quer a inserção quer a remoção de conhecimento no sistema, lidar com a problemática da evolução e involução do conhecimento e finalmente desenvolver um sistema de inferência que implementa os mecanismos de raciocínio necessários.

## 2 | Descrição do Trabalho e Análise de Resultados

O relatório do trabalho divide-se em 5 partes fundamentais. A primeira parte é a **Base de Conhecimento**, onde são descritos os predicados relativos ao conhecimento perfeito ou relativos a conhecimento imperfeito que constituem a base de conhecimento do nosso sistema. De seguida, são apresentados os **Invariante**s estruturais e referenciais que controlam a inserção e remoção de conhecimento de todos os tipos de predicados criados. Na secção seguinte, **Evolução e Involução de Conhecimento**, apresentamos os procedimentos construídos para adicionar e remover conhecimento no sistema, ilustrados com exemplos práticos de aplicação. Na quarta secção, apresentamos os **Predicados Auxiliares**, onde se encontram as funções que nos auxiliaram na construção dos restantes predicados mais elaborados. Finalmente, apresentamos o **Sistema de Inferência** desenvolvido para o nosso sistema.

## 2.1 Base de conhecimento

O nosso sistema de representação de conhecimento relativo à área da contratação pública contém as seguintes fontes de conhecimento:

O **contrato** é caracterizado pelo seu identificador único, o id, pelos nifs das entidades adjudicante e adjudicatária, pelo tipo de contrato, pelo tipo de procedimento (Consulta Previa, Ajuste Direto, Concurso publico), pelo valor, prazo local e data.

**Contrato:** IdC, IdAd, IdAda, TipoDeContrato, TipoDeProcedimento, Descrição, Valor, Prazo, Local, Data -> {V,F,D}

A entidade **adjudicante** é identificada pelo seu id, nome, nif e morada.

**Adjudicante:** IdAd, Nome, NIF, Morada -> {V,F,D}

A entidade **adjudicatária** é identificada pelo seu id, nome, nif e morada.

**Adjudicatária:** IdAda, Nome, NIF, Morada -> {V,F,D}

É de salientar que para que possamos representar informação incompleta o domínio das soluções é verdadeiro, falso ou desconhecido.

### 2.1.1 Conhecimento Perfeito

A construção da nossa base de conhecimento iniciou-se pela adição de factos perfeitos positivos relativos a todos os predicados mencionados anteriormente.

```
%Extensao do predicado Contrato: Id,IdAd, IdAda, TipodeContrato, TipoDeProcedimento, Descrição, Valor, Prazo, Local, Data -> {V,F,D}
contrato(1,600018709,502381973, "Aquisicao de servicos", "Ajuste Direto",
        "Aquisicao de servicos de acesso da Base de Dados Juridicos", 4000, 195, "Lisboa", data(19,03,2020)).
contrato(2,506696464,800589087, "Aquisicao de servicos", "Ajuste Direto",
        "Reparacao e Conservacao de Escolas - Fornecimento de aluminos", 2800, 30, "Vila Flor", data(16,02,2020)).

%Extensao do predicado adjudicante: IdAd, Nome, NIF, Morada -> {V,F,D}
adjudicante(1, "Direcao-Geral do Tribunal de Contas", 600018709, "Lisboa").
adjudicante(2, "Municipio de Vila Flor", 506696464, "Vila Flor").

%Extensao do predicado adjudicataria: IdAda, Nome, NIF, Morada -> {V,F,D}
adjudicataria(1, "Data Juris - Direito e Informatica Lda", 502381973, "Portugal").
adjudicataria(4, "OLYMPUS IBERIA SAU", 980474710, "Portugal").
```

Figura 2.1: Exemplos de factos perfeitos positivos da base de conhecimento

No caso de conhecimento perfeito negativo, adicionamos apenas factos com negação forte à base de conhecimento.

```

-contrato(9,501128840,502605731, "Aquisicao de servicos", "Ajuste Direto",
    |   |   | "Prestacao de Servicos de Informatica para a Gestao dos Clientes", 4800.50, 365, data(17,03,2020)).
-adjudicante(9, "Municipio de Braganca", 501128840, "Braganca").
-adjudicataria(13, "CGITI Portugal", 502605731, "Portugal").

```

Figura 2.2: Exemplos de factos perfeitos negativos da base de conhecimento

De modo a construir uma base de conhecimento generalizada, para além da informação negativa explícita admitimos também o pressuposto do mundo fechado, PMF, para os três predicados. Posto isto, caso um contrato, entidade adjudicante ou adjudicatária não estejam definidos na base de conhecimento nem que contenham alguma exceção associada são considerados conhecimento falso. As vantagens desta decisão encontram-se, por exemplo, no caso da representação de conhecimento imperfeito impreciso, visto que se não soubermos exatamente o valor do prazo do contrato mas se soubermos o intervalo de valores em que se situa podemos afirmar com certezas que um contrato com um prazo não contido entre esses valores é falso, apesar dessa informação não se encontrar explicitamente na nossa base de conhecimento.

```

-contrato(IdC,IdAd,IdAda,TContrato,TProcedimento,D,V,P,L,DT) :-  

    nao(contrato(IdC,IdAd,IdAda,TContrato,TProcedimento,D,V,P,L,DT)),  

    nao(excecao(contrato(IdC,IdAd,IdAda,TContrato,TProcedimento,D,V,P,L,DT))).  

  

-adjudicante(IdAd,Name,Nif,Morada) :-  

    nao(adjudicante(IdAd,Name,Nif,Morada)),  

    nao(excecao(adjudicante(IdAd,Name,Nif,Morada))).  

  

-adjudicataria(IdAd,Name,Nif,Morada) :-  

    nao(adjudicataria(IdAd,Name,Nif,Morada)),  

    nao(excecao(adjudicataria(IdAd,Name,Nif,Morada))).  


```

Figura 2.3: PMF para os predicados existentes

### 2.1.2 Conhecimento Imperfeito Incerto

O conhecimento imperfeito incerto diz respeito a conhecimento desconhecido de um determinado campo do predicado, dentro de um conjunto ilimitado de hipóteses.

Como se pode observar pelo exemplo apresentado na figura 2.4 o contrato de locação de bens móveis por contrato público com id 10 possui um prazo desconhecido. Este prazo desconhecido é tomado como conhecimento incerto pois é-nos totalmente desconhecido qual poderá ser o valor que este toma.

```
%Não se sabe o prazo do contrato
contrato(10,506346773, 503504564, "Locacao de bens moveis", "Concurso Publico",
| | | "Fornecimento de energia eletrica em BTN e MT para as instalacoes da GESAMB", 1780.22, prazo_desconhecido, data(18,01,2019)).
```

```
%Não se sabe a morada da entidade adjudicante
adjudicante(8, "Supremo Tribunal Administrativo", 600006638, morada_desconhecida).
```

```
%Não se sabe a morada da entidade adjudicatária
adjudicataria(3, "Plateia de Calculos Unipessoal Lda", 515204463, morada_desconhecida).
```

Figura 2.4: Exemplos de conhecimento imperfeito incerto da base de conhecimento

Para representar este tipo de conhecimento, é necessário criar o contrato, entidade adjudicante ou adjudicatária mencionando que o campo desconhecido é "desconhecido". Foram criadas exceções na base de conhecimento para que ao serem feitas questões relativamente a estes o resultado seja indicado corretamente como desconhecido.

```
%----- - - - - -
% Exceções para conhecimento imperfeito incerto
%----- - - - - -
```

```
% ----- Exceções do predicado Contrato
excecao(contrato(Id,IdA,IdAda,TC,TP,Desc,Val,P,Local,Data)) :- contrato(Id,IdA,IdAda,desconhecido,TP,Desc,Val,P,Local,Data).
excecao(contrato(Id,IdA,IdAda,TC,TP,Desc,Val,P,Local,Data)) :- contrato(Id,IdA,IdAda,TC,desconhecido,Desc,Val,P,Local,Data).
excecao(contrato(Id,IdA,IdAda,TC,TP,Desc,Val,P,Local,Data)) :- contrato(Id,IdA,IdAda,TC,TP,desconhecido,Val,P,Local,Data).
excecao(contrato(Id,IdA,IdAda,TC,TP,Desc,Val,P,Local,Data)) :- contrato(Id,IdA,IdAda,TC,TP,Desc,desconhecido,P,Local,Data).
excecao(contrato(Id,IdA,IdAda,TC,TP,Desc,Val,P,Local,Data)) :- contrato(Id,IdA,IdAda,TC,TP,Desc,Val,desconhecido,Local,Data).
excecao(contrato(Id,IdA,IdAda,TC,TP,Desc,Val,P,Local,Data)) :- contrato(Id,IdA,IdAda,TC,TP,Desc,Val,P,desconhecido,Data).
excecao(contrato(Id,IdA,IdAda,TC,TP,Desc,Val,P,Local,Data)) :- contrato(Id,IdA,IdAda,TC,TP,Desc,Val,P,Local,desconhecido).

% ----- Exceções do predicado Adjudicante
excecao(adjudicante(Id,Nome,Nif,Morada)) :- adjudicante(Id,desconhecido,Nif,Morada).
excecao(adjudicante(Id,Nome,Nif,Morada)) :- adjudicante(Id,Nome,Nif,desconhecido).

% ----- Exceções do predicado Adjudicatária
excecao(adjudicataria(Id,Nome,Nif,Morada)) :- adjudicante(Id,desconhecido,Nif,Morada).
excecao(adjudicataria(Id,Nome,Nif,Morada)) :- adjudicante(Id,Nome,Nif,desconhecido).
```

Figura 2.5: Exceções de conhecimento imperfeito incerto

### 2.1.3 Conhecimento Imperfeito Impreciso

O conhecimento imperfeito impreciso diz respeito a conhecimento desconhecido de um determinado campo do predicado, dentro de um conjunto limitado de hipóteses.

Como se pode observar pelo exemplo apresentado na figura 2.6 para entidade adjudicante "Universidade do Minho" com id igual a 4 não se sabe ao certo qual a morada, no entanto sabe-se que pode ser Braga ou Guimarães. Assim, implementamos a exceções de modo a que quando a base de conhecimento for interrogada relativamente à entidade adjudicante "Universidade do Minho" a resposta seja falso caso a morada não seja "Braga" ou "Guimarães" e a resposta seja desconhecido caso a morada seja uma das duas mencionadas anteriormente.

```

%Não se sabe se o valor do contrato é 500€ ou 4700€
excecao(contrato(11,600086992, 503188620, "Locacao de bens moveis", "Concurso Publico",
    "Contratacao de 2 veiculos eletricos em regime de aluguer operacional de veiculos", 5000, 140,
    "Portugal", data(16,01,2020))).
excecao(contrato(11,600086992, 503188620, "Locacao de bens moveis", "Concurso Publico",
    "Contratacao de 2 veiculos eletricos em regime de aluguer operacional de veiculos", 4700, 140,
    "Portugal", data(16,01,2020))).

%Não se sabe se a morada da entidade é Braga ou Guimarães
excecao(adjudicante(4, "Universidade do Minho", 502011378, "Braga")).
excecao(adjudicante(4, "Universidade do Minho", 502011378, "Guimaraes")).

%Não se sabe se a morada da entidade é Vila Flor ou Mirandela
excecao(adjudicataria(2, "Carlos Manuel Pires", 809589087, "Vila Flor")).
excecao(adjudicataria(2, "Carlos Manuel Pires", 809589087, "Mirandela")).

```

Figura 2.6: Exemplos de conhecimento imperfeito impreciso da base de conhecimento

É de notar que como todos os predicados assentam no PMF o sistema de inferência dá falso como resultado caso a base de conhecimento não seja questionada com as alternativas existentes nas exceções construídas, e dá desconhecido caso seja questionada com uma das alternativas. Caso tivéssemos decidido excluir algum dos predicados do PMF a resposta a esse seria sempre desconhecido visto que tudo o que não está explicitamente definido como positivo/negativo é considerado desconhecido.

#### 2.1.4 Conhecimento Imperfeito Interdito

O conhecimento imperfeito interdito diz respeito a conhecimento desconhecido que não é passível de ser conhecido.

No caso do conhecimento imperfeito interdito, como podemos observar pela figura 2.7, para o exemplo da entidade adjudicatária com id igual a 5, cuja morada é impossível saber, começamos por criá-la com as suas informações e com o campo "morada\_interdita" e é necessário criar também uma exceção relativa ao campo interdito, para que não seja considerado conhecimento positivo/negativo. É também preciso criar uma instância de nulo interdito com "morada\_interdita" para que ao construirmos o invariante que impeça a evolução de conhecimento relativo a este predicado possamos verificar que se trata de uma entidade adjudicatária com informação impossível de conhecer e, consequentemente, impossível de ser adicionada.

```

%É impossivel saber o valor do contrato
contrato(12,508481287,508592909, "Aquisicao de bens moveis", "Consulta Previa", "Aquisicao de seringas",
         valor_interdito, 352, data(14,01,2020)).
excecao(contrato(Id,IdA,IdAda,Tc,TP,Desc,Val,Pr,Local,Data)) :-
    contrato(contrato(Id,IdA,IdAda,Tc,TP,Desc,valor_interdito,Pr,Local,Data)).


+contrato(Id,IdA,IdAda,Tc,TP,Desc,Val,Pr,Local,Data) :: (solucoes((Id,IdA,IdAda,Tc,TP,Desc,Val,Pr,Local,Data),
    contrato(12,508481287,508592909, "Aquisicao de bens moveis",
        "Consulta Previa", "Aquisicao de seringas", valor_interdito, 352,
        data(14,01,2020)),
    nao(nulointerdito(valor_interdito))),R), comprimento(R,0)).

%É impossivel saber a nome da entidade
adjudicante(6, nome_interdito, 501413197, "Porto").
excecao(adjudicante(Id,Nome,Nif,Morada)) :- adjudicataria(Id,nome_interdito,Nif,Morada).

+adjudicante(Id,Nome,Nif,Morada) :: (solucoes((Id,Nome,Nif,Morada),
    (adjudicante(6,Nom, 501413197, "Porto"),
        nao(nulointerdito(Nom))), R), comprimento(R,0)).


%É impossivel saber a morada da entidade
adjudicataria(5, "Manuel Rui Azinhais Nabeiro Lda", 500853975, morada_interdita).

+adjudicataria(Id,Nome,Nif,Morada) :: (solucoes((Id,Nome,Nif,Morada),
    (adjudicataria(5,"Manuel Rui Azinhais Nabeiro Lda", 500853975, morada_interdita),
        nao(nulointerdito(morada_interdita))), R), comprimento(R,0)).

```

Figura 2.7: Exemplos de conhecimento imperfeito interdito da base de conhecimento

```

%----- %
% Exceções para conhecimento imperfeito interdito
%----- %

% ----- Exceções do predicado Contrato
excecao(contrato(Id,IdA,IdAda,TC,TP,Desc,Val,P,Local,Data)) :- contrato(Id,IdA,IdAda,tc_interdito,TP,Desc,Val,P,Local,Data).
excecao(contrato(Id,IdA,IdAda,TC,TP,Desc,Val,P,Local,Data)) :- contrato(Id,IdA,IdAda,TC,tp_interdito,Desc,Val,P,Local,Data).
excecao(contrato(Id,IdA,IdAda,TC,TP,Desc,Val,P,Local,Data)) :- contrato(Id,IdA,IdAda,TC,TP,descricao_interdita,Val,P,Local,Data).
excecao(contrato(Id,IdA,IdAda,TC,TP,Desc,Val,P,Local,Data)) :- contrato(Id,IdA,IdAda,TC,TP,desc,valor_interdito,P,Local,Data).
excecao(contrato(Id,IdA,IdAda,TC,TP,Desc,Val,P,Local,Data)) :- contrato(Id,IdA,IdAda,TC,TP,desc,Val,prazo_interdito,Local,Data).
excecao(contrato(Id,IdA,IdAda,TC,TP,Desc,Val,P,Local,Data)) :- contrato(Id,IdA,IdAda,TC,TP,desc,Val,local_interdito,Data).
excecao(contrato(Id,IdA,IdAda,TC,TP,Desc,Val,P,Local,Data)) :- contrato(Id,IdA,IdAda,TC,TP,desc,Val,P,local,data_interdita).

+contrato(Id,IdA,IdAda,TC,TP,Desc,Val,P,Local,Data) :: (solucoes((Id,IdA,IdAda,TC,TP,Desc,Val,P,Local,Data),
    (contrato(Id,_,_,TC,TP,Desc,Val,P,Local,Data), (nulointerdito(TC);
        nulointerdito(P);
        nulointerdito(TP);
        nulointerdito(Desc);
        nulointerdito(Val);
        nulointerdito(Local));
        R),
    comprimento(R,0)).

% ----- Exceções do predicado Adjudicante
excecao(adjudicante(Id,Nome,Nif,Morada)) :- adjudicante(Id,nome_interdito,Nif,Morada).
excecao(adjudicante(Id,Nome,Nif,Morada)) :- adjudicante(Id,Nome,Nif,morada_interdita).

% ----- Exceções do predicado Adjudicataria
excecao(adjudicataria(Id,Nome,Nif,Morada)) :- adjudicataria(Id,nome_interdito,Nif,Morada).
excecao(adjudicataria(Id,Nome,Nif,Morada)) :- adjudicataria(Id,Nome,Nif,morada_interdita).

```

Figura 2.8: Exceções ao conhecimento imperfeito interdito

## 2.2 Invariantes

### 2.2.1 Invariantes universais

Existem alguns invariantes que podem ser generalizados e, desta forma, todos os predicados que serão inseridos na nossa base de conhecimento terão que obedecer a esses invariantes. Em primeiro lugar, construímos invariantes que garantem que não existe conhecimento redundante, e criamos invariantes que não permitem que seja adicionado conhecimento perfeito positivo que contradiz conhecimento perfeito negativo presente na base de conhecimento e vice-versa como se pode ver na figura seguinte.

```
% Invariante que garante que não existe conhecimento perfeito positivo repetido
+contrato(A,B,C,D,E,F,G,H,I,J)      :: (solucoes(A, contrato(A,B,C,D,E,F,G,H,I,J),L),comprimento(L,1)).
+adjudicante(A,B,C,D)                :: (solucoes(A, adjudicante(A,B,C,D),L),comprimento(L,1)).
+adjudicataria(A,B,C,D)              :: (solucoes(A, adjudicataria(A,B,C,D),L),comprimento(L,1)).

% Invariante que garante que não existe conhecimento perfeito negativo repetido
+(-contrato(A,B,C,D,E,F,G,H,I,J))   :: (solucoes(A, -contrato(A,B,C,D,E,F,G,H,I,J),L),comprimento(L,1)).
+(-adjudicante(A,B,C,D))            :: (solucoes(A, -adjudicante(A,B,C,D),L),comprimento(L,1)).
+(-adjudicataria(A,B,C,D))          :: (solucoes(A, -adjudicataria(A,B,C,D),L),comprimento(L,1)).

% Invariante que garante que não existe conhecimento imperfeito generico repetido
+imperfeito(CII)                   :: (solucoes(CII, excecao(CII), R),comprimento(R, 1)).
% Invariante que garante que não existe conhecimento imperfeito incerto repetido
+incerto(CII)                      :: (solucoes(CII, excecao(CII), R),comprimento(R, 1)).
% Invariante que garante que não existe conhecimento imperfeito impreciso repetido
+impreciso(CII)                    :: (solucoes(CII, excecao(CII), R),comprimento(R, 1)).
% Invariante que garante que não existe conhecimento imperfeito interdito repetido
+interdito(CII)                    :: (solucoes(CII, excecao(CII), R),comprimento(R, 1)).
% Invariante que garante que não existem excecoes repetidas
+(excecao(E)) :: (solucoes(E, excecao(E), R),comprimento(R, 1)).

% Invariante que não permite adicionar conhecimento perfeito positivo que contradiz conhecimento perfeito negativo
+contrato(A,B,C,D,E,F,G,H,I,J)      :: nao((solucoes(A, -contrato(A,B,C,D,E,F,G,H,I,J),L),comprimento(L,1))).
+adjudicante(A,B,C,D)                :: nao((solucoes(A, -adjudicante(A,B,C,D),L),comprimento(L,1))).
+adjudicataria(A,B,C,D)              :: nao((solucoes(A, -adjudicataria(A,B,C,D),L),comprimento(L,1))).

% Invariante que não permite adicionar conhecimento perfeito negativo que contradiz conhecimento perfeito positivo
+(-contrato(A,B,C,D,E,F,G,H,I,J))   :: nao((solucoes(A, contrato(A,B,C,D,E,F,G,H,I,J),L),comprimento(L,1))).
+(-adjudicante(A,B,C,D))            :: nao((solucoes(A, adjudicante(A,B,C,D),L),comprimento(L,1))).
+(-adjudicataria(A,B,C,D))          :: nao((solucoes(A, adjudicataria(A,B,C,D),L),comprimento(L,1))).
```

Figura 2.9: Invariantes universais para cada tipo de conhecimento

### 2.2.2 Invariantes de Contrato

O invariante apresentado de seguida garante que conhecimento perfeito não é conhecimento imperfeito

```
+contrato(IdC,_,_,_,_,_,_,_,_) :: (solucoes(IdC, excecao(contrato(IdC,_,_,_,_,_,_,_)),R),comprimento(R,0)).
```

Figura 2.10: Invariante de contrato

É necessário garantir que o id de cada contrato é único para conhecimento perfeito positivo, imperfeito incerto e imperfeito interdito. No caso do conhecimento imperfeito impreciso, uma vez que podem existir vários valores possíveis para os parâmetros é aceitável que haja mais que um contrato com o mesmo id.

```
%Garantir que não há 2 contratos com ids iguais
+contrato(IdC,_,-,-,-,-,-,-,-) :: (solucoes(IdC, contrato(IdC,_,-,-,-,-,-,-,-) ,R), comprimento(R,1)).
+incerto(contrato(IdC,_,-,-,-,-,-,-,-)) :: (solucoes(IdC, contrato(IdC,_,-,-,-,-,-,-,-) ,R), comprimento(R,1)).
+interdito(contrato(IdC,_,-,-,-,-,-,-,-)) :: (solucoes(IdC, contrato(IdC,_,-,-,-,-,-,-,-) ,R), comprimento(R,1)).
```

Figura 2.11: Invariante de contrato

É necessário assegurar que contratos com ids diferentes têm informação diferente. Tal como no invariante apresentado anteriormente, no caso do conhecimento imperfeito impreciso, devido à possibilidade de associar múltiplos valores a um campo aceitam-se ids repetidos.

```
% Garantir que contratos com ids diferentes têm diferente informação
+contrato(IdC,A,B,C,D,E,F,G,H,Data) :: (solucoes(IdC, contrato(_A,B,C,D,E,F,G,H,Data) ,R) ,comprimento(R,1)).
+incerto(contrato(IdC,A,B,C,D,E,F,G,H,Data)) :: (solucoes(IdC, contrato(_A,B,C,D,E,F,G,H,Data) ,R) ,comprimento(R,1)).
+interdito(contrato(IdC,A,B,C,D,E,F,G,H,Data)) :: (solucoes(IdC, contrato(_A,B,C,D,E,F,G,H,Data) ,R) ,comprimento(R,1)).
```

Figura 2.12: Invariante de contrato

Na base de conhecimento apenas são permitidos contratos cujo tipo de procedimento seja "Ajuste Direto", "Consulta Prévia" e "Concurso Público".

```
% Garantir que cada contrato só tem 1 dos 3 tipos de procedimentos possíveis
+contrato(IdC,IdAd,IdAda,TContrato,TProcedimento,D,V,P,L,DT) :: (contrato(IdC,IdAd,IdAda,TContrato,"Ajuste Direto",D,V,P,L,DT);
contrato(IdC,IdAd,IdAda,TContrato,"Consulta Previa",D,V,P,L,DT);
contrato(IdC,IdAd,IdAda,TContrato,"Concurso Publico",D,V,P,L,DT)).
+incerto(contrato(IdC,IdAd,IdAda,TContrato,TProcedimento,D,V,P,L,DT)) :: (excecao(contrato(IdC,IdAd,IdAda,TContrato,desconhecido,D,V,P,L,DT));
excecao(contrato(IdC,IdAd,IdAda,TContrato,"Ajuste Direto",D,V,P,L,DT));
excecao(contrato(IdC,IdAd,IdAda,TContrato,"Consulta Previa",D,V,P,L,DT));
excecao(contrato(IdC,IdAd,IdAda,TContrato,"Concurso Publico",D,V,P,L,DT))).
+impreciso(contrato(IdC,IdAd,IdAda,TContrato,TProcedimento,D,V,P,L,DT)) :: (excecao(contrato(IdC,IdAd,IdAda,TContrato,"Ajuste Direto",D,V,P,L,DT));
excecao(contrato(IdC,IdAd,IdAda,TContrato,"Consulta Previa",D,V,P,L,DT));
excecao(contrato(IdC,IdAd,IdAda,TContrato,"Concurso Publico",D,V,P,L,DT))).
+interdito(contrato(IdC,IdAd,IdAda,TContrato,TProcedimento,D,V,P,L,DT)) :: (excecao(contrato(IdC,IdAd,IdAda,TContrato,procedimento_interdito,D,V,P,L,DT));
excecao(contrato(IdC,IdAd,IdAda,TContrato,"Ajuste Direto",D,V,P,L,DT));
excecao(contrato(IdC,IdAd,IdAda,TContrato,"Consulta Previa",D,V,P,L,DT));
excecao(contrato(IdC,IdAd,IdAda,TContrato,"Concurso Publico",D,V,P,L,DT))).
```

Figura 2.13: Invariante de contrato

Um contrato cujo tipo de procedimento seja "Ajuste Direto" não pode ter um valor superior a 5000 euros.

```
%Garantir que o valor do contrato por ajuste direto é igual ou inferior a 5000 euros
+contrato(IdC,_,-,-,"Ajuste Direto",_,V_,_,_,_) :: (V <= 5000).
+incerto(contrato(IdC,_,-,-,"Ajuste Direto",_,V_,_,_,_)) :: (excecao(contrato(IdC,_,-,-,"Ajuste Direto",_,desconhecido,_,-,_)) ; (V <= 5000)).
+impreciso(contrato(IdC,_,-,-,"Ajuste Direto",_,V_,_,_,_)) :: ((intervalo(V)) ; (excecao(contrato(IdC,_,-,-,"Ajuste Direto",_,V_,_,_,_)), V <= 5000)).
+interdito(contrato(IdC,_,-,-,"Ajuste Direto",_,V_,_,_,_)) :: (excecao(contrato(IdC,_,-,-,"Ajuste Direto",_,valor_interdito,_,-,_)) ; V <= 5000).
```

Figura 2.14: Invariante de contrato

Um contrato cujo tipo de procedimento seja "Ajuste Direto" apenas pode ser do tipo "Contrato de Aquisição", "Locação de Bens Móveis" ou "Aquisição de Serviços".

```
%Garantir que um contrato por ajuste direto é um dos seguintes: Contrato de aquisição ou locação de bens móveis ou aquisição de serviços
+contrato(IdC,...,TContrato,"Ajuste Direto",...,_,_,_) :: (contrato(IdC,...,"Aquisicao","Ajuste Direto",...,_,_,_);
    contrato(IdC,...,"Locacao de bens moveis","Ajuste Direto",...,_,_,_);
    contrato(IdC,...,"Aquisicao de servicos","Ajuste Direto",...,_,_,_));
+incerto(contrato(IdC,...,TContrato,"Ajuste Direto",...,_,_,_)) :: (execao(contrato(IdC,...,"Aquisicao","Ajuste Direto",...,_,_,_));
    execao(contrato(IdC,...,"Locacao de bens moveis","Ajuste Direto",...,_,_,_));
    execao(contrato(IdC,...,"Aquisicao de servicos","Ajuste Direto",...,_,_,_));
+impreciso(contrato(IdC,...,TContrato,"Ajuste Direto",...,_,_,_)) :: (execao(contrato(IdC,...,"Aquisicao","Ajuste Direto",...,_,_,_));
    execao(contrato(IdC,...,"Locacao de bens moveis","Ajuste Direto",...,_,_,_));
    execao(contrato(IdC,...,"Aquisicao de servicos","Ajuste Direto",...,_,_,_));
+interdito(contrato(IdC,...,TContrato,"Ajuste Direto",...,_,_,_)) :: (execao(contrato(IdC,...,"Aquisicao","Ajuste Direto",...,_,_,_));
    execao(contrato(IdC,...,"Locacao de bens moveis","Ajuste Direto",...,_,_,_));
    execao(contrato(IdC,...,"Aquisicao de servicos","Ajuste Direto",...,_,_,_));
    execao(contrato(IdC,...,tip_interdito,"Ajuste Direto",...,_,_,_));
    execao(contrato(IdC,...,"Locacao de bens moveis","Ajuste Direto",...,_,_,_));
    execao(contrato(IdC,...,"Aquisicao de servicos","Ajuste Direto",...,_,_,_)));

```

Figura 2.15: Invariante de contrato

Um contrato cujo tipo de procedimento seja "Ajuste Direto" pode ter um prazo de vigência até 1 ano, inclusive, a contar da decisão de adjudicação, este invariante verifica-se para qualquer tipo de conhecimento.

```
% Prazo de vigencia ate 1 anno
+contrato(IdC,...,"Ajuste Direto",...,Prazo,...) :: Prazo=<365.
+incerto(contrato(IdC,...,"Ajuste Direto",...,Prazo,...)) :: (execao(contrato(IdC,...,"Ajuste Direto",...,desconhecido,...)) ; Prazo=<365).
+impreciso(contrato(IdC,...,"Ajuste Direto",...,Prazo,...)) :: (Intervalo(Prazo);Prazo=<365).
+interdito(contrato(IdC,...,"Ajuste Direto",...,Prazo,...)) :: (execao(contrato(IdC,...,"Ajuste Direto",...,prazo_interdito,...));Prazo=<365).
```

Figura 2.16: Invariante de contrato

O invariante seguinte força a que uma entidade adjudicante possa 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.

```
%Regra dos 3 anos válida para todos os contratos
+contrato(IdC,IdAd,IdAda,TContrato,TProcedimento,Descricao,Val,Prazo,Local,Data) :: 
    (solucoes(VL, (contrato(_,IdAd,IdAda,...,VL,...,Dt),
        nao(nulointerdito(VL)),
        nao(anoinperfeito(Dt)),
        nao(contrato(_,IdAd,IdAda,...,VL,...,desconhecido)),
        menos3Anos(Dt,Data)),L),
     sumVals([-Val|L],Ret),Ret<75000).

+incerto(contrato(IdC,IdAd,IdAda,TContrato,TProcedimento,Descricao,Val,Prazo,Local,Data)) :: 
    (contrato(IdC,IdAd,IdAda,TContrato,TProcedimento,Descricao,desconhecido,Prazo,Local,Data); 
        anoImperfeito(Data),
        (solucoes(VL, (contrato(_,IdAd,IdAda,...,VL,...,Dt),
            nao(nulointerdito(VL)),
            nao(anoinperfeito(Dt)),
            nao(contrato(_,IdAd,IdAda,...,desconhecido,...,Dt)),
            menos3Anos(Dt,Data)),L),
         sumVals([-Val|L],Ret),
         Ret<75000)).

+interdito(contrato(IdC,IdAd,IdAda,TContrato,TProcedimento,Descricao,Val,Prazo,Local,Data)) :: 
    (contrato(IdC,IdAd,IdAda,TContrato,TProcedimento,Descricao,valor_interdito,Prazo,Local,Data); 
        anoImperfeito(Data),
        (solucoes(VL, (contrato(_,IdAd,IdAda,...,VL,...,Dt),
            nao(nulointerdito(VL)),
            nao(anoinperfeito(Dt)),
            nao(contrato(_,IdAd,IdAda,...,desconhecido,...,Dt)),
            menos3Anos(Dt,Data)),L),
         sumVals([-Val|L],Ret),
         Ret<75000)).
```

Figura 2.17: Invariante de contrato

É necessário garantir que para qualquer contrato, este está associado a um entidade adjudicante ou uma entidade adjudicatária que está presente na base de conhecimento, este invariante verifica-se para qualquer tipo de conhecimento.

```
% Garantir que o valor de cada contrato é válido (>= 0) para conhecimento perfeito positivo
+contrato(IdC,_,-,_,-,V,_,-,-) :: valorValido(V).
+incerto(contrato(IdC,_,-,_,-,V,_,-,-)) :: (contrato(IdC,_,-,_,-,desconhecido,_,-,-) ; valorValido(V)).
+impreciso(contrato(IdC,_,-,_,-,V,_,-,-)) :: (contrato(IdC,_,-,_,-,valor_interdito,_,-,-) ; valorValido(V)).
+interdito(contrato(IdC,_,-,_,-,V,_,-,-)) :: (contrato(IdC,_,-,_,-,valor_interdito,_,-,-) ; valorValido(V)).
```

  

```
% Garantir que o valor de cada contrato é válido (>= 0) para conhecimento perfeito negativo
+(-contrato(IdC,_,-,_,-,V,_,-,-)) :: valorValido(V).
```

Figura 2.18: Invariante de contrato

É necessário garantir que um contrato está associado a um nif de um entidade adjudicante e adjudicatária que já existe na base de conhecimento.

```
% Garantir que um contrato está associado a um nif existente, quer para uma entidade adjudicante, quer para uma entidade adjudicatária
+contrato(.,Ida,Idada,_,_,_) :: (solucoes(Ida, (adjudicante(.,_,Ida,_)),excecao(adjudicante(.,_,Ida,_))), R1), comprimento(R1,S1, S1 >= 1),
    (solucoes(Idada, (adjudicatária(.,_,Idada,_)),excecao(adjudicatária(.,_,Idada,_))), R2), comprimento(R2,S2, S2 >= 1)).
+incerto(contrato(.,Ida,Idada,_,_,_)) :: (solucoes(Ida, (adjudicante(.,_,Ida,_)),excecao(adjudicante(.,_,Ida,_))), R1), comprimento(R1,S1, S1 >= 1),
    (solucoes(Idada, (adjudicatária(.,_,Idada,_)),excecao(adjudicatária(.,_,Idada,_))), R2), comprimento(R2,S2, S2 >= 1)).
+impreciso(contrato(.,Ida,Idada,_,_,_)) :: (solucoes(Ida, (adjudicante(.,_,Ida,_)),excecao(adjudicante(.,_,Ida,_))), R1), comprimento(R1,S1, S1 >= 1),
    (solucoes(Idada, (adjudicatária(.,_,Idada,_)),excecao(adjudicatária(.,_,Idada,_))), R2), comprimento(R2,S2, S2 >= 1)).
+interdito(contrato(.,Ida,Idada,_,_,_)) :: (solucoes(Ida, (adjudicante(.,_,Ida,_)),excecao(adjudicante(.,_,Ida,_))), R1), comprimento(R1,S1, S1 >= 1),
    (solucoes(Idada, (adjudicatária(.,_,Idada,_)),excecao(adjudicatária(.,_,Idada,_))), R2), comprimento(R2,S2, S2 = 1)).
```

Figura 2.19: Invariante de contrato

A data de um contrato tem de ser válida, isto é, o dia tem de ser um número entre 1 e 31, o mês entre 1 e 12 e o ano não pode ser superior a 2020 uma vez que não pode ser adicionados contratos futuros.

```
%Data Valida
+contrato(_,_,_,_,_,_,_,_,_,_,Data):- dataValida(Data).
+incerto(contrato(_,_,_,_,_,_,_,_,_,_,D)):-dataValida(D).
+interdito(contrato(_,_,_,_,_,_,_,_,_,_,D)):-dataValida(D).
+imperfeito(contrato(_,_,_,_,_,_,_,_,_,_,D)):-dataValida(D).
```

Figura 2.20: Invariante de contrato

### 2.2.3 Invariantes da Entidade Adjudicante

É necessário garantir que o id e o nif de cada entidade adjudicante é único. Devido à possibilidade de existirem múltiplos valores para determinados parâmetros, no caso do conhecimento imperfeito impreciso, o invariante abaixo não se aplica.

```
%Garantir que o id e nif de cada entidade adjudicante é único para conhecimento perfeito positivo
+adjudicante(IdAd, Nome, Nif, Morada) :: (solucoes(IdAd, (adjudicante(IdAd,_,_,_)), R), comprimento(R,1)).
+adjudicante(IdAd, Nome, Nif, Morada) :: (solucoes(Nif, adjudicante(_,_,Nif,_), R), comprimento(R,1)).

%Garantir que o id e nif de cada entidade adjudicante é único para conhecimento imperfeito incerto
+incerto(adjudicante(IdAd, Nome, Nif, Morada)) :: (solucoes(IdAd, (adjudicante(IdAd,_,_,_)), R), comprimento(R,1)).
+incerto(adjudicante(IdAd, Nome, Nif, Morada)) :: (solucoes(Nif, adjudicante(_,_,Nif,_), R), comprimento(R,1)).

%Garantir que o id e nif de cada entidade adjudicante é único para conhecimento imperfeito interdito
+interdito(adjudicante(IdAd, Nome, Nif, Morada)) :: (solucoes(IdAd, (adjudicante(IdAd,_,_,_)), R), comprimento(R,1)).
+interdito(adjudicante(IdAd, Nome, Nif, Morada)) :: (solucoes(Nif, adjudicante(_,_,Nif,_), R), comprimento(R,1)).
```

Figura 2.21: Invariante de uma entidade adjudicante

O invariante apresentado de seguida garante que entidades adjudicantes com IDs diferentes têm diferente informação. Devido à possibilidade de existirem

múltiplos valores para determinados parâmetros, no caso do conhecimento imperfeito impreciso, o invariante abaixo não se aplica.

```
% Garantir que adjudicantes com ids diferentes têm diferente informação para conhecimento perfeito positivo
+adjudicante(IdAd, Nome, Nif, Morada) :: (solucoes((Nome, Nif, Morada), adjudicante(_, Nome, _, Morada), R), comprimento(R,1)).
+incerto(adjudicante(IdAd, Nome, Nif, Morada) :: (solucoes((Nome, Nif, Morada), adjudicante(_, Nome, _, Morada), R), comprimento(R,1)).
+interdito(adjudicante(IdAd, Nome, Nif, Morada) :: (solucoes((Nome, Nif, Morada), adjudicante(_, Nome, _, Morada), R), comprimento(R,1)).

% Garantir que adjudicantes com ids diferentes têm diferente informação para conhecimento perfeito negativo
+(-adjudicante(IdAd, Nome, Nif, Morada) :: (solucoes((Nome, Nif, Morada), -adjudicante(_, Nome, Nif, Morada), R),
comprimento(R,1)).
```

Figura 2.22: Invariante de uma entidade adjudicante

Caso uma entidade adjudicante esteja associada a algum contrato da base de conhecimento essa entidade não pode ser removida. O invariante da figura seguinte assegura-se disso.

```
% Garantir que não é possível remover uma entidade adjudicatária com um contrato
-adjudicante(_,_,Nif,_) :: (solucoes(Nif, contrato(IdC,Nif,_,_,_,_,_), R),comprimento(R, 0)).
+incerto(adjudicante(_,_,Nif,_) :: (solucoes(Nif, contrato(IdC,Nif,_,_,_,_,_), R),comprimento(R, 0)).
-interdito(adjudicante(_,_,Nif,_) :: (solucoes(Nif, contrato(IdC,Nif,_,_,_,_,_), R),comprimento(R, 0))).
```

Figura 2.23: Invariante de uma entidade adjudicante

O nif de uma entidade tem que ser um número de 9 dígitos para ser considerado válido.

```
% Garantir que o nif do adjudicante é válido para conhecimento positivo
+adjudicante(_,_,Nif,_) :: nifValido(Nif).
+(-adjudicante(_,_,Nif,_) :: nifValido(Nif).
+incerto(adjudicante(Id,_,Nif,_) ::(adjudicante(Id,_,desconhecido,_);nifValido(Nif)).
+interdito(adjudicante(Id,_,Nif,_) ::(adjudicante(Id,_,nif_interdito,_);nifValido(Nif))).
```

Figura 2.24: Invariante de uma entidade adjudicante

## 2.2.4 Invariantes da Entidade Adjudicatária

É necessário garantir que o id e o nif de cada entidade adjudicatária é único. Devido à possibilidade de existirem múltiplos valores para determinados parâmetros, no caso do conhecimento imperfeito impreciso, o invariante abaixo não se aplica.

```
%Garantir que o id e nif de cada entidade adjudicatária é único para conhecimento perfeito positivo
+adjudicataria(IdAd, Nome, Nif, Morada) :: (solucoes(IdAd, (adjudicataria(IdAd,_,_,_),R), comprimento(R,1)).
+adjudicataria(IdAd, Nome, Nif, Morada) :: (solucoes(Nif, adjudicataria(_,_,Nif,_,_), R), comprimento(R,1)).

%Garantir que o id e nif de cada entidade adjudicatária é único para conhecimento imperfeito incerto
+incerto(adjudicataria(IdAd, Nome, Nif, Morada) :: (solucoes(IdAd, (adjudicataria(IdAd,_,_,_),R), comprimento(R,1)).
+incerto(adjudicataria(IdAd, Nome, Nif, Morada) :: (solucoes(Nif, adjudicataria(_,_,Nif,_,_), R), comprimento(R,1)).

%Garantir que o id e nif de cada entidade adjudicatária é único para conhecimento imperfeito interdito
+interdito(adjudicataria(IdAd, Nome, Nif, Morada)) :: (solucoes(IdAd, (adjudicataria(IdAd,_,_,_),R), comprimento(R,1)).
+interdito(adjudicataria(IdAd, Nome, Nif, Morada)) :: (solucoes(Nif, adjudicataria(_,_,Nif,_,_), R), comprimento(R,1))).
```

Figura 2.25: Invariante de uma entidade adjudicatária

O invariante apresentado de seguida garante que entidades adjudicatárias com ids diferentes têm diferente informação. Devido à possibilidade de existirem múltiplos valores para determinados parâmetros, no caso do conhecimento imperfeito impreciso, o invariante abaixo não se aplica.

Figura 2.26: Invariante de uma entidade adjudicatária

Caso uma entidade adjudicataria esteja associada a algum contrato da base de conhecimento essa entidade não pode ser removida. O invariante da figura seguinte assegura-se disso.

Figura 2.27: Invariante de uma entidade adjudicatária

O nif de uma entidade tem que ser um número de 9 dígitos para ser considerado válido.

```
% indica que o nif do adjudicataria é válido para conhecimento positivo
+adjudicataria(Id,...,Nif,...) :: nifValido(Nif),
+adjudicataria(Id,...,Nif,...) :: not nifValido(Nif),
+adjudicataria(Id,...,Nif,...) :: incerto(adjudicataria(Id,...,desconhecido,...);nifValido(Nif)).
+interdito(adjudicataria(Id,...,Nif,...)) :: (adjudicataria(Id,...,nif_interditado,...);nifValido(Nif)).
```

Figura 2.28: Invariante de uma entidade adjudicatária

## 2.3 Evolução e Involução do conhecimento

Para tratar a evolução do conhecimento decidimos criar procedimentos específicos para cada um dos tipos de conhecimento existentes, tendo em consideração cada uma das implicações impostas por cada um deles. De seguida será explicado detalhadamente cada um dos procedimentos criados para o efeito bem como exemplos de aplicação dos mesmos.

### 2.3.1 Evolução do conhecimento perfeito positivo e negativo

O predicado evolução é responsável por adicionar novo conhecimento à base de conhecimento, verificando primeiro se todos os invariantes de adição de conhecimento relativos a este se mantêm verdadeiros após a inserção, caso isto se verifique o conhecimento é adicionado, caso contrário é removido. Para a evolução de conhecimento perfeito negativo criamos o terceiro predicado presente na figura seguinte, no qual é necessário especificar que o objetivo é adicionar conhecimento negativo. De seguida serão verificados os invariantes que dizem respeito à inserção de conhecimento negativo e, em seguida e em caso de sucesso, é inserido na base de conhecimento.

```
% Insere novo conhecimento na base de conhecimento
evolucao(T) :- evolucao(T, positivo).

% Insere conhecimento perfeito positivo na base de conhecimento
evolucao(T, positivo) :- solucoes(Inv, +T :: Inv, Linv),
|   |   |   |   |
|   |   |   |   insercao(T),
|   |   |   |   teste(Linv).

% Insere conhecimento perfeito negativo na base de conhecimento
evolucao(T, negativo) :- solucoes(Inv, +(T) :: Inv, Linv),
|   |   |   |   |
|   |   |   |   insercao(-T),
|   |   |   |   teste(Linv).
```

Figura 2.29: Evolução do conhecimento

```
| ?- evolucao(contrato(20,600018709,502381973, "Aquisicao","Ajuste Direto","desc", 4000, 195, "Lisboa", data(19,03,2020))).
yes
| ?- evolucao(adjudicante(21, nome, 123456789, morada)).
yes
| ?-
evolucao(adjudicataria(21,nome,987654321,morada)). | ?-
yes
```

Figura 2.30: Exemplos de evolução de conhecimento perfeito positivo

```
| ?- evolucao(~contrato(20,600018709,502381973, "Aquisicao","Ajuste Direto","desc", 4000, 195, "Lisboa", data(19,03,2020)), negativo).
yes
| ?- evolucao(~adjudicante(21,nome,123456789,morada), negativo).
yes
| ?-
evolucao(~adjudicataria(21,nome,123456789,morada), negativo).
yes
```

Figura 2.31: Exemplos de evolução de conhecimento perfeito negativo

Como se pode observar pela figura 2.30, foi adicionada a entidade com id igual a 21, contudo ao tentar adicionar um entidade com id igual a 22 mas com um nif igual é quebrado um invariante pelo que como se pode observar pela figura 2.31 o sistema responde "no" não fazendo a inserção desse. Nesta figura podemos também observar a tentativa de adição de um contrato com o nif de uma entidade adjudicatária que não se encontra na base conhecimento pelo que a inserção não é impedida.

```
[1 ?- evolucao(adjudicataria(22,nome,987654321,morada)).
no
[1 ?- evolucao(contrato(20,123456789,502381973, "Aquisicao","Ajuste Direto","desc", 4000, 195, "Lisboa", data(19,03,2020))).
no
```

Figura 2.32: Exemplos de evolução de conhecimento perfeito positivo

### 2.3.2 Evolução do conhecimento imperfeito incerto

Como já foi explicado em secções anteriores, para conhecimento imperfeito incerto é necessário adicionar o predicado pretendido com o campo em falta como “desconhecido” e é necessária uma exceção associada a esse mesmo campo. Para a criação de procedimentos adequados à evolução do conhecimento incerto foi necessário desenvolver vários predicados tendo em conta que um contrato, uma entidade adjudicante e uma entidade adjudicatária têm diferentes possibilidades de campos incertos.

Nas figuras seguintes apresentam-se apenas alguns exemplos de predicados desenvolvidos para lidar com este tipo de conhecimento, uma vez que são todos semelhantes.

```
% Insere conhecimento imperfeito incerto na base de conhecimento de um contrato com um tipo de contrato desconhecido
evolucao(contrato(Idc,IdAd,IdAdta,TC_desconhecido,TProcedimento,D,V,P,L,DT), incerto, tipocontrato) :-
    solucoes(Inv,+incerto(contrato(Idc,IdAd,IdAdta,desconhecido,TProcedimento,D,V,P,L,DT)) :: Inv, Linv),
    insercao(contrato(Idc,IdAd,IdAdta,desconhecido,TProcedimento,D,V,P,L,DT)),
    teste(Linv).
```

Figura 2.33: Predicado de evolução do conhecimento imperfeito incerto de contrato

```
% Insere conhecimento imperfeito incerto na base de conhecimento de um adjudicante com um nome desconhecido
evolucao(adjudicante(IdAd,Name,Nif,Morada), incerto, nome) :-
    solucoes(Inv,+incerto(adjudicante(IdAd,desconhecido,Nif,Morada)) :: Inv, Linv),
    insercao(adjudicante(IdAd,desconhecido,Nif,Morada)),
    teste(Linv).
```

Figura 2.34: Predicado de evolução do conhecimento imperfeito incerto de uma entidade adjudicante

```
% Insere conhecimento imperfeito incerto na base de conhecimento de um adjudicante com um nome desconhecido
evolucao(adjudicante(IdAd,Name,Nif,Morada), incerto, nome) :-
    solucoes(Inv,+incerto(adjudicante(IdAd,desconhecido,Nif,Morada)) :: Inv, Linv),
    insercao(adjudicante(IdAd,desconhecido,Nif,Morada)),
    teste(Linv).
```

Figura 2.35: Predicado de evolução do conhecimento imperfeito incerto de uma entidade adjudicatária

Na figura seguinte apresentam-se exemplos práticos da evolução deste tipo de conhecimento:

```

| ?- evolucao(contrato(31,600018709,502381973,desconhecido,"Ajuste Direto","desc",340,25,"local",data(1,1,2020)), incerto, tipocontrato).
yes
| ?- evolucao(adjudicante(20,desconhecido,987654321,morada), incerto, nome).
yes
| ?- evolucao(adjudicante(21,nome,123456789,desconhecido), incerto, morada).
yes

```

Figura 2.36: Exemplos de evolução de conhecimento imperfeito incerto

### 2.3.3 Evolução do conhecimento imperfeito impreciso

Para satisfazer os requisitos do conhecimento imperfeito impreciso é necessário adicionar exceções para cada uma das imprecisões possíveis em cada predicado. Este procedimento averigua se os invariantes relativos às exceções permanecem verdadeiros depois do conhecimento ser inserido. Em caso afirmativo, o conhecimento permanece na base de conhecimento.

Nas figuras seguintes apresentam-se apenas alguns exemplos de predicados desenvolvidos para lidar com este tipo de conhecimento, uma vez que são todos semelhantes.

```

% Insere conhecimento imperfeito impreciso na base de conhecimento de um contrato com um tipo de contrato impreciso
evolucao(contrato(IdC,IdAd,IdAda,TipoContrato,TProcedimento,D,Valor,P,L,DT),impreciso,tipocontrato);-
    solucoes(Inv, +impreciso(contrato(IdC,IdAd,IdAda,TipoContrato,TProcedimento,D,Valor,P,L,DT)) :: Inv,Linv),
    insercao(execcao(contrato(IdC,IdAd,IdAda,TipoContrato,TProcedimento,D,Valor,P,L,DT))) ,
    teste(Linv).

```

Figura 2.37: Predicado de evolução do conhecimento imperfeito impreciso para contrato

```

% Insere conhecimento imperfeito impreciso na base de conhecimento de um adjudicante com nome impreciso
evolucao(adjudicante(IdAd,Nome,Nif,M),impreciso,nome);-
    solucoes(Inv, +impreciso(adjudicante(IdAd,Nome,Nif,M)) :: Inv,Linv),
    insercao(execcao(adjudicante(IdAd,Nome,Nif,M))) ,
    teste(Linv).

```

Figura 2.38: Predicado de evolução do conhecimento imperfeito impreciso de uma entidade adjudicante

```

% Insere conhecimento imperfeito impreciso na base de conhecimento de uma entidade adjudicataria com nome impreciso
evolucao(adjudicataria(IdAd,Nome,Nif,M),impreciso,nome);-
    solucoes(Inv, +impreciso(adjudicataria(IdAd,Nome,Nif,M)) :: Inv,Linv),
    insercao(execcao(adjudicataria(IdAd,Nome,Nif,M))) ,
    teste(Linv).

```

Figura 2.39: Predicado de evolução do conhecimento imperfeito impreciso de uma entidade adjudicataria

Na figura seguinte apresentam-se exemplos práticos da evolução deste tipo de conhecimento:

```
% Insere conhecimento imperfeito impreciso na base de conhecimento de um contrato com um intervalo de valores
evolucao(contrato(IdC,IdAd,IdAda,TipoContrato,TProcedimento,D,Valor,P,L,DT), impreciso, valor, LimiteInferior, LimiteSuperior) :-
    insercao((execicao(contrato(IdC,IdAd,IdAda,TipoContrato,TProcedimento,D,Val,P,L,DT)) :- Val >= LimiteInferior, Val <= LimiteSuperior)).

% Insere conhecimento imperfeito impreciso na base de conhecimento de um contrato com um intervalo de valores para o prazo
evolucao(contrato(IdC,IdAd,IdAda,TipoContrato,TProcedimento,D,Valor,P,L,DT), impreciso, prazo, LimiteInferior, LimiteSuperior) :-
    insercao((execicao(contrato(IdC,IdAd,IdAda,TipoContrato,TProcedimento,D,Valor,Pr,L,DT)) :- Pr >= LimiteInferior, Pr <= LimiteSuperior)).

% Insere conhecimento imperfeito impreciso na base de conhecimento de um contrato com um intervalo de datas
evolucao(contrato(IdC,IdAd,IdAda,TipoContrato,TProcedimento,D,Valor,P,L,DT), impreciso, data, data(Di,Mi,Ai), data(Ds,Ms,As)) :-
    insercao((execicao(contrato(IdC,IdAd,IdAda,TipoContrato,TProcedimento,D,Valor,P,L,data(D,M,A))) :- ((A > Ai);(A==Ai; (M>Mi; (M==Mi, D>Di))),((A < As);(A==As, (M<Ms; (M==Ms, D=<Ds)))))).
```

Figura 2.40: Exemplos de evolução de conhecimento imperfeito incerto

No caso de conhecimento impreciso, existe também a possibilidade de adicionar conhecimento com um campo desconhecido entre um intervalo finito de valores, por exemplo, no caso do prazo de um contrato se situar entre 200 a 300 dias. Para tal, criamos predicados de evolução adicionais que se encontram na figura seguinte.

```
% Insere conhecimento imperfeito impreciso na base de conhecimento de um contrato com um intervalo de valores
evolucao(contrato(IdC,IdAd,IdAda,TipoContrato,TProcedimento,D,Valor,P,L,DT), impreciso, valor, LimiteInferior, LimiteSuperior) :-
    insercao((execicao(contrato(IdC,IdAd,IdAda,TipoContrato,TProcedimento,D,Val,P,L,DT)) :- Val >= LimiteInferior, Val <= LimiteSuperior)).

% Insere conhecimento imperfeito impreciso na base de conhecimento de um contrato com um intervalo de valores para o prazo
evolucao(contrato(IdC,IdAd,IdAda,TipoContrato,TProcedimento,D,Valor,P,L,DT), impreciso, prazo, LimiteInferior, LimiteSuperior) :-
    insercao((execicao(contrato(IdC,IdAd,IdAda,TipoContrato,TProcedimento,D,Valor,Pr,L,DT)) :- Pr >= LimiteInferior, Pr <= LimiteSuperior)).

% Insere conhecimento imperfeito impreciso na base de conhecimento de um contrato com um intervalo de datas
evolucao(contrato(IdC,IdAd,IdAda,TipoContrato,TProcedimento,D,Valor,P,L,DT), impreciso, data, data(Di,Mi,Ai), data(Ds,Ms,As)) :-
    insercao((execicao(contrato(IdC,IdAd,IdAda,TipoContrato,TProcedimento,D,Valor,P,L,data(D,M,A))) :- ((A > Ai);(A==Ai; (M>Mi; (M==Mi, D>Di))),((A < As);(A==As, (M<Ms; (M==Ms, D=<Ds)))))).
```

Figura 2.41: Predicados de evolução de conhecimento imperfeito incerto para um intervalo de valores

Para a utilização destes procedimentos, é necessário indicar o termo T que se pretende adicionar, o tipo de predicado em questão, impreciso, o campo desconhecido e o limite inferior e superior, respetivamente, do intervalo de valores.

```
| ?- evolucao(contrato(34,600018709,502381973,"Aquisicao","Ajuste Direto",desc,34,340,local,data(idk,idk,idk),impreciso, data,data(1,1,2020),data(2,2,2020)).
yes
| ?- evolucao(contrato(20,600018709,502381973, "Aquisicao","Ajuste Direto","desc", 3000, 195, "Lisboa", data(19,03,2020)), impreciso, prazo, 200, 300).
yes
| ?- evolucao(contrato(25,600018709,502381973, "Aquisicao","Ajuste Direto","desc", idk, 195, "Lisboa", data(19,03,2020)), impreciso, valor, 200, 300).
yes _
```

Figura 2.42: Exemplos de evolução de conhecimento imperfeito incerto para um intervalo de valores

### 2.3.4 Evolução do conhecimento imperfeito interdito

Para a evolução deste conhecimento foram necessárias criar as exceções presentes na base de conhecimento e ter em atenção os invariantes que precisam de existir previamente e a que estes têm que obedecer.

```
% Insere conhecimento imperfeito interdito na base de conhecimento de um contrato com um tipo de contrato interdito
evolucao(contrato(IdC,IdAd,IdAda,TipoContrato,TProcedimento,D,Valor,P,L,Data),interdito,tipocontrato) :-
    solucoes(Inv, +interdito(contrato(IdC,IdAd,IdAda,tc_interdito,TProcedimento,D,Valor,P,L,Data)) :: Inv,Linv),
    insercao(contrato(IdC,IdAd,IdAda,tc_interdito,TProcedimento,D,Valor,P,L,Data)),
    teste(Linv).
```

Figura 2.43: Predicado de evolução de conhecimento imperfeito interdito para um contrato

```

% Insere conhecimento imperfeito interdito na base de conhecimento de uma entidade adjudicante com morada imprecisa
evolucao(adjudicante(Id, Nome, Nif, Morada), interdito, morada ):-  

    solucoes(Inv, +interdito(adjudicante(Id, Nome, Nif, morada_interdita)) :: Inv, Linv),  

    insercao(adjudicante(Id, Nome, Nif, morada_interdita)),  

    teste(Linv).

```

Figura 2.44: Predicado de evolução de conhecimento imperfeito interdito para uma entidade adjudicante

```

% Insere conhecimento imperfeito interdito na base de conhecimento de uma entidade adjudicatária com morada imprecisa
evolucao(adjudicatária(Id, Nome, Nif, Morada), interdito, morada ):-  

    solucoes(Inv, +interdito(adjudicatária(Id, Nome, Nif, morada_interdita)) :: Inv, Linv),  

    insercao(adjudicatária(Id, Nome, Nif, morada_interdita)),  

    teste(Linv).

```

Figura 2.45: Predicado de evolução de conhecimento imperfeito interdito para uma entidade adjudicatária

```

| ?- evolucao(adjudicante(20, desconhecido, 987654321, morada), interdito, nome).  

yes  

| ?- evolucao(adjudicante(21, nome, 123456789, desconhecido), interdito, morada).  

yes  

| ?- evolucao(contrato(31, 600018709, 502381973, "Aquisição", "Ajuste Direto", desc, 1240, 340, local, data(desconhecido, desconhecido, desconhecido)), interdito, data).  

yes

```

Figura 2.46: Exemplos de evolução de conhecimento imperfeito interdito

### 2.3.5 Involução do conhecimento perfeito positivo e negativo

```

%-----  

involucao(I):-involucao(I,positivo).  

  

% Remove conhecimento perfeito positivo na base de conhecimento  

involucao(T, positivo) :- solucoes(Inv, -T :: Inv, Linv),  

    | T,  

    | remocao(T),  

    | teste(Linv).  

  

% Remove conhecimento perfeito negativo na base de conhecimento  

involucao(T, negativo) :- solucoes(Inv, -(~T) :: Inv, Linv),  

    | ~T,  

    | remocao(~T),  

    | teste(Linv).  

%-----  


```

Figura 2.47: Involução do conhecimento

Como se pode observar, após a inserção de novo conhecimento este é removido através do predicado involução, e quando invocado para conhecimento que já não existe o sistema responde "no" à impossibilidade de remover conhecimento que já não existe.

```

| ?- evolucao(adjudicante(21,nome,123456789,morada)).
yes
| ?- involucao(adjudicante(21,nome,123456789,morada)).
yes
| ?- involucao(adjudicante(21,nome,123456789,morada)).
no

```

Figura 2.48: Exemplos de involução de conhecimento perfeito positivo

A título de exemplo, a entidade adjudicante com id igual a 1 existe na base de conhecimento e tem um contrato associado a ela, assim ao tentar remove-la da base de conhecimento é quebrado o invariante que garante os contratos existentes estão associados a uma entidade existente pelo que é impossível remover esta entidade enquanto o contrato existir na base de conhecimento, como podemos observar pela figura abaixo.

```

| ?- adjudicante(I, "Direcao-Geral do Tribunal de Contas", 600018709, "Lisboa").
yes
| ?- contrato(I, 600018709, 502381973, "Aquisicao de servicos", "Ajuste Direto", "Aquisicao de servicos de acesso da Base de Dados Juridicos", 4000, 195, "Lisboa", d
ata(19,03,2020)).
yes
| ?- involucao(adjudicante(I, "Direcao-Geral do Tribunal de Contas", 600018709, "Lisboa")).


```

Figura 2.49: Exemplos de involução de conhecimento perfeito positivo

### 2.3.6 Involução do conhecimento imperfeito incerto

Para a involução de conhecimento imperfeito foram criados diferentes procedimentos para cada um dos tipos de predicados existentes, na figura seguinte apresentamos apenas um exemplo uma vez que a representação dos restantes é análoga.

```

% Remove conhecimento imperfeito incerto na base de conhecimento de um contrato com um tipo de contrato desconhecido
involucao(contrato(IdC,IdAd,IdAda,TC_desconhecido,TProcedimento,D,V,P,L,DT), incerto, tipocontrato) :-
    solucoes(Inv,-incerto(contrato(IdC,IdAd,IdAda,desconhecido,TProcedimento,D,V,P,L,DT)) :: Inv, LInv),
    contrato(IdC,IdAd,IdAda,desconhecido,TProcedimento,D,V,P,L,DT),
    remocao(contrato(IdC,IdAd,IdAda,desconhecido,TProcedimento,D,V,P,L,DT)),
    teste(LInv).

```

Figura 2.50: Predicado de involução de conhecimento imperfeito incerto

```

% Remove conhecimento imperfeito incerto na base de conhecimento de um adjudicante com uma morada desconhecida
involucao(adjudicante(IdAd,Name,Nif,Morada), incerto, morada) :-
    solucoes(Inv,-incerto(adjudicante(IdAd,Name,Nif,desconhecido)) :: Inv, LInv),
    adjudicante(IdAd,Name,Nif,desconhecido),
    remocao(adjudicante(IdAd,Name,Nif,desconhecido)),
    teste(LInv).


```

Figura 2.51: Predicado de involução de conhecimento imperfeito incerto

```

% Remove conhecimento imperfeito incerto na base de conhecimento de um adjudicataria com um nome desconhecido
involucao(adjudicataria(IdAd,Name,Nif,Morada), incerto, nome) :-
    solucoes(Inv,-incerto(adjudicataria(IdAd,desconhecido,Nif,Morada)) :: Inv, LInv),
    adjudicataria(IdAd,desconhecido,Nif,Morada),
    remocao(adjudicataria(IdAd,desconhecido,Nif,Morada)),
    teste(LInv).


```

Figura 2.52: Predicado de involução de conhecimento imperfeito incerto

Na figura seguinte encontra-se um exemplo de involução de conhecimento para cada tipo de predicado presente no sistema.

```
| ?- involucao(contrato(33,600018709,502381973,"Aquisicao de servicos","Ajuste Direto",desc,-1,25,local,data(1,1,2020)),incerto, valor).
yes
| ?- involucao(contrato(33,600018709,502381973,"Aquisicao de servicos","Ajuste Direto",desc,-1,25,local,data(1,1,2020)), incerto, valor).
yes
| ?- involucao(adjudicante(20,desconhecido,987654321,morada), incerto, nome).
yes
| ?- involucao(adjudicante(20,desconhecido,987654321,morada), incerto, nome).
yes
| ?- involucao(adjudicante(21,nome,123456789,desconhecido), incerto, morada).
yes
| ?- involucao(adjudicante(21,nome,123456789,desconhecido), incerto, morada).
yes
```

Figura 2.53: Exemplo de involução de conhecimento imperfeito incerto

### 2.3.7 Involução do conhecimento imperfeito impreciso

Para o caso da involução de conhecimento imperfeito impreciso encontra-se um exemplo dos procedimentos criados na figura seguinte, apenas apresentamos um deles, uma vez que a representação dos restantes é análoga.

```
involucao(contrato(IdC,IdAd,IdAda,TipoContrato,TProcedimento,D,Valor,P,L,DT),impreciso,descricao):-
    solucoes(Inv, -impreciso(contrato(IdC,IdAd,IdAda,TipoContrato,TProcedimento,D,Valor,P,L,DT)) :: Inv,Linv),
    excecao(contrato(IdC,IdAd,IdAda,TipoContrato,TProcedimento,D,Valor,P,L,DT)),
    remocao(excecao(contrato(IdC,IdAd,IdAda,TipoContrato,TProcedimento,D,Valor,P,L,DT))) ,
    teste(Linv).
```

Figura 2.54: Predicado de involução de conhecimento imperfeito impreciso

```
involucao(adjudicante(IdAd,Nome,Nif,M),impreciso,morada):-
    solucoes(Inv, -impreciso(adjudicante(IdAd,Nome,Nif,M)) :: Inv,Linv),
    excecao(adjudicante(IdAd,Nome,Nif,M)),
    remocao(excecao(adjudicante(IdAd,Nome,Nif,M))) ,
    teste(Linv).
```

Figura 2.55: Predicado de involução de conhecimento imperfeito impreciso

```
involucao(adjudicataria(IdAd,Nome,Nif,M),impreciso,nome):-
    solucoes(Inv, -impreciso(adjudicataria(IdAd,Nome,Nif,M)) :: Inv,Linv),
    excecao(adjudicataria(IdAd,Nome,Nif,M)),
    remocao(excecao(adjudicataria(IdAd,Nome,Nif,M))) ,
    teste(Linv).
```

Figura 2.56: Predicado de involução de conhecimento imperfeito impreciso

Na figura seguinte encontra-se um exemplo de involução de conhecimento para cada tipo de predicado presente no sistema.

```
| ?- involucao(contrato(36,600018709,502381973,"Aquisicao de servicos","Ajuste Direto",desc,340,25,local,data(1,1,2020)),impreciso, descricao).
yes
| ?- involucao(contrato(36,600018709,502381973,"Aquisicao de servicos","Ajuste Direto",desc,340,25,local,data(1,1,2020)), impreciso, descricao).
yes
| ?- involucao(adjudicante(20,impreciso,987654321,morada), impreciso, nome).
yes
| ?- involucao(adjudicante(20,impreciso,987654321,morada), impreciso, nome).
yes
| ?- involucao(adjudicataria(22,impreciso,987654321,morada), impreciso, nome).
yes
| ?- involucao(adjudicataria(22,impreciso,987654321,morada), impreciso, nome).
yes
```

Figura 2.57: Exemplo de involução de conhecimento imperfeito impreciso

### 2.3.8 Involução do conhecimento imperfeito interdito

Para o caso da involução de conhecimento imperfeito interdito encontra-se um exemplo dos procedimentos criados na figura seguinte, apenas apresentamos um deles, uma vez que a representação dos restantes é análoga.

```
involucao(contrato(IdC,IdAd,IdAda,TipoContrato,TP,Valor,P,L,Data),interdito,tipoprocedimento) :-
    solucoes(Inv, -interdito(contrato(Idc,IdAd,IdAda,tp_interdito,TP,Valor,P,L,Data)) :: Inv,Linv),
    contrato(Idc,IdAd,IdAda,TipoContrato,tp_interdito,D,Valor,P,L,Data),
    remocao(contrato(Idc,IdAd,IdAda,TipoContrato,tp_interdito,D,Valor,P,L,Data)),
    teste(Linv).
```

Figura 2.58: Predicado de involução de conhecimento imperfeito interdito

```
%% Adjudicante Interdita
involucao(adjudicante(Id, Nome, Nif, Morada), interdito, nome ):- 
    solucoes(Inv, -interdito(adjudicante(Id, nome_interdito, Nif, Morada)) :: Inv,Linv),
    adjudicante(Id, nome_interdito, Nif, Morada),
    remocao(adjudicante(Id, nome_interdito, Nif, Morada)),
    teste(Linv).
```

Figura 2.59: Predicado de involução de conhecimento imperfeito interdito

```
%% Adjudicataria Interdita
involucao(adjudicataria(Id, Nome, Nif, Morada), interdito, nome ):- 
    solucoes(Inv, -interdito(adjudicataria(Id, nome_interdito, Nif, Morada)) :: Inv,Linv),
    adjudicataria(Id, nome_interdito, Nif, Morada),
    remocao(adjudicataria(Id, nome_interdito, Nif, Morada)),
    teste(Linv).
```

Figura 2.60: Predicado de involução de conhecimento imperfeito interdito

Na figura seguinte encontra-se um exemplo de involução de conhecimento para cada tipo de predicado presente no sistema.

```
| ?- evolucao(contrato(25,600018709,502381973,"Aquisicao de servicos","Ajuste Direto",desc,-1,25,local,data(1,1,2020)),interdito, valor).
yes
| ?- involucao(contrato(25,600018709,502381973,"Aquisicao de servicos","Ajuste Direto",desc,-1,25,local,data(1,1,2020)),interdito, valor).
yes
| ?- evolucao(adjudicante(20,desconhecido,987654321,morada), interdito, nome).
yes
| ?- involucao(adjudicante(20,desconhecido,987654321,morada), interdito, nome).
yes
| ?- evolucao(adjudicataria(21,nome,123456789,desconhecido), interdito, morada).
yes
| ?- involucao(adjudicataria(21,nome,123456789,desconhecido), interdito, morada).
yes
```

Figura 2.61: Exemplo de involução de conhecimento imperfeito interdito

## 2.4 Predicados Auxiliares

Para o desenvolvimento dos procedimentos do sistema de representação de conhecimento e raciocínio, utilizamos diversos predicados auxiliares ao longo de todo o desenvolvimento que apresentaremos de seguida.

O predicado *não*, devolve o valor de verdade contrário ao termo Q passado como parâmetro, caso exista uma prova afirmativa ou negativa explícita de Q na base de conhecimento dá no como resposta e na ausência de prova, dá yes.

```
% Extensao do meta-predicado não: Q -> {V,F}

% Devolve o valor de verdade contrário a Q (negação fraca)
não(Q) :- Q, !, fail.
não(Q).
```

Figura 2.62: Predicado 'não'

O predicado *soluções* coloca em R a lista com as soluções do tipo X para todo o XS que encontrar.

```
% Extensao do predicado soluções: X,XS,R -> {V,F}

% Encontra todas as soluções
solucoes(X, XS, R) :- findall(X, XS, R).
```

Figura 2.63: Predicado 'comprimento'

O predicado *comprimento* calcula o tamanho da lista passada como argumento.

```
% Extensao do predicado comprimento: X,R -> {V,F}
% Devolve o comprimento de uma lista
comprimento([],0).
comprimento([_|XS],R) :- comprimento(XS,S), R is 1 + S.
```

Figura 2.64: Predicado 'soluções'

O predicado *inserção* insere Q na base de conhecimento no caso de sucesso e retorna yes. No caso de insucesso retorna no após a retração do mesmo. O predicado *remoção*, faz o oposto do predicado mencionado anteriormente.

```

% Inserção de conhecimento
insercao(Q) :- assert(Q).
insercao(Q) :- retract(Q), !, fail.

% Remoção de conhecimento
remocao(Q) :- retract(Q).
remocao(Q) :- assert(Q), !, fail.

```

Figura 2.65: Predicados de inserção e remoção de conhecimento

O predicado *teste*, que testa se todos os predicados passados como parâmetro são verdadeiros.

```

% Testa se todos os predicados são verdadeiros
teste([]).
teste([I|L]) :- I, teste(L).

```

Figura 2.66: Predicados 'teste'

O predicado *nifValido* garante que o nif de qualquer entidade é válido, isto é, é um número com 9 dígitos.

```

% Nif válido
nifValido(Nif) :- Nif >= 100000000, Nif <= 999999999.

```

Figura 2.67: Predicado 'nifValido'

O predicado *valorValido* garante que o valor inserido para um contrato não é negativo.

```

% Valor válido (>= 0)
valorValido(V) :- V >= 0.

```

Figura 2.68: Predicado 'valorValido'

Os predicados apresentados na figura seguinte garantem que os valores introduzidos numa data são válidos.

```

% Data válida
dataValida(data(D,M,A)):-((flagImperfeito(A);anoValido(A)),
                           (flagImperfeito(M); mesValido(M)),
                           (flagImperfeito(D);diaValido(D))).
```

```

% Dia válido
diaValido(Dia) :- Dia >= 1, Dia <= 31.
```

```

% Mes válido
mesValido(M) :- M >= 1, M <= 12.
```

```

% Ano válido
anoValido(A) :- A >= 0, A <= 2020.
```

```

flagImperfeito(desconhecido).
flagImperfeito(F) :- nuloInterdito(F).
anoImperfeito(A) :- nuloInterdito(A).
anoImperfeito(desconhecido).
anoImperfeito(data(_,_,A)):-nuloInterdito(A).
anoImperfeito(data(_,_,desconhecido)).
```

Figura 2.69: Predicados de validação de uma data

O predicado *menos3Anos* calcula a diferença entre duas datas.

```

% Dadas duas datas calcula se estão separadas por 3 anos
menos3Anos(data(_,_,NYDt),data(_,_,NYData)):- SubY is NYData - NYDt,!, SubY <= 3, SubY >= -3.
```

Figura 2.70: Predicado 'menos3Anos'

O predicado *sumVals* calcula o valor total de uma lista.

```

% Soma de valores
sumVals([V],V).
sumVals([V1|T],Ret):- sumVals(T,Ret2), Ret is Ret2+V1.
```

Figura 2.71: Predicado 'sumVals'

## 2.5 Sistema de Inferência

Por último, foi necessário construir um sistema de inferência capaz de implementar os mecanismos de raciocínio adequados. Posto isto, construímos o predicado demo, cuja resposta é verdadeiro se existir uma prova explícita na base de conhecimento de que a questão é verdadeira, falso, se existir na base de conhecimento a negação forte da mesma, e desconhecido, caso não existam provas sobre Q na base de conhecimento.

```
% Extensao do meta-predicado demo: Questao,Resposta -> {"verdadeiro","falso","desconhecido"}
% capaz de responder a uma única questão
demo(Q,verdadeiro) :- Q.
demo(Q,falso) :- not(Q).
demo(Q,desconhecido) :- nao(Q), nao(not(Q)).
```

Figura 2.72: Sistema de inferência simples

Como podemos observar na figura seguinte, o predicado de adjudicante passado como argumento consta na base de conhecimento pelo que inquirindo o sistema sobre a sua veracidade a resposta é "yes". O segundo, sendo igual ao primeiro mas com um id diferente, uma vez que não consta na base de conhecimento quando inquirido sobre ser falso a resposta é também "yes". Em último temos uma entidade adjudicante presente na base de conhecimento cujo morada é imprecisa, contudo sabe-se que poderá ser "Braga"ou "Guimarães pelo que quando o sistema é questionado sobre a morada ser "Braga"e este facto ser desconhecido a resposta é "yes".

```
| ?- demo(adjudicante(1, "Direcao-Geral do Tribunal de Contas", 600018709, "Lisboa"),verdadeiro).
yes
| ?- demo(adjudicante(2, "Direcao-Geral do Tribunal de Contas", 600018709, "Lisboa"),falso).
yes
| ?- demo(adjudicante(4, "Universidade do Minho", 502011378, "Braga"), desconhecido).
yes _
```

Figura 2.73: Exemplificação da utilização do predicado demo

Criamos também um predicado que permite responder a várias questões em simultâneo. Desta forma, basta utilizar predicado da figura 6.2 de modo a obter resposta todas elas de uma só vez.

```
% Extensao do meta-predicado demoList: [Questao], [Resposta] -> {V,F,D}
% Capaz de responder a várias questões em simultâneo
demoList([Q],[R]) :- demo(Q,R).
demoList([Q|Qs],[R|Rs]) :- demo(Q,R), demoList(Qs,Rs).
```

Figura 2.74: Sistema de inferência com uma lista de questões

```
| ?- demoList([adjudicante(1, "Direcao-Geral do Tribunal de Contas", 600018709, "Lisboa"),adjudicante(2, "Direcao-Geral do Tribunal de Contas", 600018709, "Lisboa")]), X;
X = [verdadeiro,falso] ?
```

Figura 2.75: Exemplificação da utilização do predicado demoList

Uma vez que existe a possibilidade do utilizador se interrogar sobre conjunção e/ou disjunção de uma lista de várias questões desenvolvemos os predicados de conjunção e disjunção que tomam os valores diferentes dependendo do que está a ser analisado.

```
% Extensao do predicado conjuncao: X,Y -> {V,F,D}
conjuncao(verdadeiro,verdadeiro,verdadeiro).
conjuncao(verdadeiro,desconhecido,desconhecido).
conjuncao(desconhecido,verdadeiro,desconhecido).
conjuncao(desconhecido,desconhecido,desconhecido).
conjuncao(falso,_,falso).
conjuncao(_,falso,falso).
```

Figura 2.76: Predicado conjunção

```
% Extensao do predicado disjuncao: X,Y -> {V,F,D}
disjuncao(verdadeiro,_,verdadeiro).
disjuncao(_,verdadeiro,verdadeiro).
disjuncao(falso,falso,falso).
disjuncao(falso,desconhecido,desconhecido).
disjuncao(desconhecido,falso,desconhecido).
disjuncao(desconhecido,desconhecido,desconhecido).
```

Figura 2.77: Predicado disjunção

Finalmente, utilizando os predicados apresentados anteriormente, foi criado o procedimento da figura seguinte com o intuito de responder a uma lista de questões intercaladas por conjunções e/ou disjunções, obtendo um único resultado final.

```
% Extensao do meta-predicado query: [Questao], Resposta -> {V,F,D}
% capaz de fazer a conjunção e/ou disjunção de uma lista de questões
query([],verdadeiro).
query([Q],R) :- demo(Q,R).
query([Q1,'E'|Qs],R) :- demo(Q1,R1), query(Qs,R2), conjuncao(R1,R2,R).
query([Q1,'OU'|Qs],R) :- demo(Q1,R1), query(Qs,R2), disjuncao(R1,R2,R).
```

Figura 2.78: Conjunção/disjunção com uma lista de questões

```
?- demoList(adjudicante1, "Direcao-Geral do Tribunal de Contas", 600018709, "Lisboa"),adjudicante2, "Direcao-Geral do Tribunal de Contas", 600018709, "Lisboa").
[], X),
X = {verdadeiro,falso} ?
yes
?- query([adjudicante1, "Direcao-Geral do Tribunal de Contas", 600018709, "Lisboa"], 'E', adjudicante2, "Direcao-Geral do Tribunal de Contas", 600018709, "Lisboa"),X .
X = falso ?
yes
?- query([adjudicante1, "Direcao-Geral do Tribunal de Contas", 600018709, "Lisboa"], 'OU', adjudicante2, "Direcao-Geral do Tribunal de Contas", 600018709, "Lisboa"),X .
X = verdadeiro ?
yes
```

Figura 2.79: Exemplificação da utilização do predicado query

## 3 | Conclusão

No final deste exercício, foram implementadas todas as funcionalidades pedidas que permitem uma representação de todo o tipo de conhecimento estudado, perfeito (positivo, negativo) e imperfeito(incerto, impreciso e interdito), com vários exemplos e demonstrações de cada um deles.

Foram pensados ao pormenor os invariantes necessários para uma correta manipulação da base do conhecimento, tanto para o caso da adição como para o caso da remoção de conhecimento.

A problemática da evolução e da involução de conhecimento foi analisada ao detalhe para que todos os casos fossem abrangidos como forma de conseguirmos construir um sistema que insere e remove qualquer tipo de conhecimento.

Foi ainda construído um sistema de inferência, possibilitando a resposta a conjunções e disjunções de várias questões, tendo sempre em atenção que estamos perante uma extensão à programação em lógica.

A realização deste trabalho foi bastante importante e contribuiu imenso para entender o funcionamento da linguagem de programação lógica PROLOG e que problemas podem ser resolvidos e analisados com ela. Conseguimos, desta forma, consolidar não só todos os conhecimentos adquiridos nas aulas práticas, como também vários outros que foram sendo necessários adquirir durante a realização deste exercício.

## 4 | Referências Bibliográficas e Electrónicas

FAQ do código dos Contratos Públícos, disponível em:  
<http://www.base.gov.pt/Base/pt/PerguntasFrequentes>

Base de dados de contratos públicos, disponível em:  
<http://www.base.gov.pt>

[Analide, 2011] ANALIDE, Cesar, NOVAIS, Paulo, Neves, José, “Sugestões para a Redacção de Relatórios Técnicos”, Relatório Técnico, Departamento de Informática, Universidade do Minho, Portugal, 2011.

[Analide, 1996] ANALIDE, Cesar, Neves, José, “Representação de Informação Incompleta”, Texto Pedagógico, Grupo de Inteligência Artificial, Centro de Ciências e Tecnologias da Computação, Portugal, 1996.