UNIVERSIDADE DO MINHO

Trabalho Prático

Mestrado Integrado em Engenharia Informática

Sistemas de Representação de Conhecimento e Raciocínio
(2ºSemestre/2019-2020)

Grupo 36
Gonçalo Esteves A85731
Mário Real A72620
Rui Oliveira A83610

Braga
3 de Maio de 2020

Resumo

Este relatório visa a documentação do exercício proposto na unidade curricular de Sistemas de Representação de Conhecimento e Raciocínio que se baseia na 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 à criação de mecanismos de raciocínio adequados.

Serão abordadas as formas de conhecimento implementadas, bem como as funcionalidades do sistema desenvolvido, sendo que, paralelamente, serão também descritas as estratégias adotadas para a elaboração das mesmas.

Conteúdo

Lista de Figuras

1 Introdução

Integrado na cadeira de Sistemas de Representação de Conhecimento e Raciocínios, foi-nos proposto a demonstração de funcionalidades subjacentes à programação em lógica estendida e à representação de conhecimento imperfeito, num sistema com capacidade para caracterizar um universo de discurso, na área da contratação pública para a realização de contratos para a prestação de serviços.

O sistema deverá permitir:

1. Representar conhecimento positivo e negativo;

2. Representar casos de conhecimento imperfeito, pela utilização de valores nulos de todos os tipos estudados;

3. Manipular invariantes que designem restrições à inserção e à remoção de conhecimento do sistema;

4. Lidar com a problemática da evolução do conhecimento, criando os procedimentos adequados;

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

2 Preliminares

Como já foi referido, o programa que será utilizado neste exercício é o PROLOG, que consiste num conjunto de teoremas e regras que descrevem um dado problema. De modo a compreender o funcionamento desta linguagem, é necessário reunir alguns conceitos e significados.

Em primeiro lugar, compreender a distinção entre a *Teoria dos Modelos* e a *Teoria da Prova*. A *Teoria dos Modelos* examina a interpretação de relações lógicas através da associação de valores de verdade, enquanto que a *Teoria da Prova* examina a derivabilidade a partir de outras fórmulas e pela aplicação de regras de derivação.

Em segundo lugar, conhecer os conceitos *modus ponens* e *modus tollens*, que representam o cojunto de regras de derivação abordadas na *Lógica Preposicional*. No campo da *Lógica Matemática*, entender a importância das *Cláusulas de Horn*, que permitem representar os factos, as regras e as questões em PROLOG. Por um lado, os factos e as regras constituem a base de conhecimento do sistema e, por outro lado, as questões representam consultas à mesma, que são respondidas por dedução lógica, recorrendo a um *Algoritmo de Resolução*.

De modo a melhor compreender o programa, é também necessário conhecer os conceitos de *negação por falha na prova* e do comando *cut*, que impede que o mecanismo de *backtracking* seja possível.

Em PROLOG, é necessária a utilização sintática exemplo/? para representar um predicado com ? argumentos, sendo ? um número inteiro. De maneira a trabalhar os predicados definidos recorremos à manipulação de *Invariantes*, quer estruturais quer referenciais, que desempenham uma função importante relativamente à inserção/remoção de conhecimento.

No âmbito da programação em lógica existem três pressupostos fundamentais:

- Pressuposto dos Nomes Únicos: duas constantes diferentes designam duas entidades diferentes.
- Pressuposto do Mundo Fechado: toda a informação que não existe mencionada é falsa.
- Pressuposto do Domínio Fechado: não há mais objetos no universo para além dos designados por constantes.

No entanto, dado que iremos abordar a extensão à programação em lógica, abandonaremos o Pressuposto do Mundo Fechado. Existe, portanto, o conhecimento verdadeiro, falso e desconhecido.

No que diz respeito a conhecimento imperfeito, este poderá ser representado de três formas diferentes:

- Incerto: conhecimento acerca do qual não possuímos qualquer informação.
- Impreciso: conhecimento acerca do qual sabemos um conjunto/intervalo de valores.
- Interdito: conhecimento que não deve ser representado nem conhecido.

Para poder testar o que aqui abordamos, mais especificamente o que o programa irá desenvolver é necessário um interpretador(SICStus Prolog). Para colocar qualquer tipo de questão ao programa PROLOG, começamos por consultar o ficheiro relativo a este exercício (em Windows: abrir o SICTUS e correr consult('(...)tp1.pl').).

3 Descrição do Trabalho e Análise de Resultados

Nesta secção iremos descrever de forma concisa as características e funcionamento do sistema, apresentar o conhecimento representado e explicar a construção da extensão de predicados e os meta predicados desenvolvidos. Todas estas componentes serão auxiliadas por valores experimentais de modo a comprovar a veracidade dos valores obtidos.

Relativamante à construção de algumas extensões de predicados e à aplicação de invariantes, foi necessária a utilização dos seguintes predicados:

- teste/1 que testa uma lista;
- insercao/1 que, caso corra tudo bem, insere um termo na base de conhecimento;
- remocao/1 que, caso corra tudo bem, remove um termo da base de conhecimento;
- remocaoLista/1 que, caso corra tudo bem, remove uma lista de termos da base de conhecimento;
- solucoes/3 que devolve as soluções em formato de lista de termos segundo uma determinada questão;
- solucoesSemRep/3 que devolve as soluções em formato de lista de termos segundo uma determinada questão, sem elementos repetidos;
- comprimento/2 que calcula o tamanho de uma lista;
- soma/2 que calcula a soma dos elementos de uma lista;
- apagaUltimo/2 que remove o último elemento de uma lista.
- apagaPrimeiro/2 que remove o primeiro elemento de uma lista.
- concatenar/3 que concatena duas listas.

3.1 Descrição do Sistema

Considere-se que o panorama será caracterizado por conhecimento dado na forma que se segue:

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

Perspetivando o funcionamento do sistema, foram implementadas as seguintes funcionalidades:

- Registar adjudicantes, adjudicatárias e contratos;
- Remover adjudicantes, adjudicatárias e contratos;
- Calcular o valor total dos contratos celebrados por um/uma adjudicante/adjudicatária;
- Alterar a morada de uma adjudicataria;
- Alterar o nome de uma adjudicataria;
- Determinar lista de ID's/Nomes/NIFs/Moradas dos/das adjudicantes/adjudicatárias conforme opção;
- Identificar os contratos realizados por um/uma adjudicante/adjudicatária;
- Identificar o número total de contratos celebrados entre um adjudicante e uma adjudicatária;
- Identifcar os tipos de contrato dos contratos celebrados por um/uma adjudicante/adjudicatária;
- Identificar os contratos realizados num determinado local;
- Identifcar os contratos com prazo igual/superior/inferior a uma referência;
- Identifcar os contratos que aconteceram numa determinada data.

De modo a respeitar as indicações do documento de apoio a este projeto e, além disso, a não comprometer a coerência e veracidade do que se encontra na base de conhecimento, foi necessário ter em conta:

- Não podem existir adjudicantes/adjudicatárias/contratos com ID's iguais na base de conhecimento;
- Para inserir um contrato, as entidades adjudicante e adjudicatária do contrato têm que se encontrar na base de conhecimento;
- Não podem ser retirados adjudicantes ou adjudicatárias da base de conhecimento caso tenham, pelo menos, um contrato associado;
- Os tipos de procedimento admissíveis de um contrato são ajuste direto, consulta prévia e concurso público;
- Os contratos por ajuste direto têm certas restrições: valor igual ou inferior a 5000 euros; tipo de contrato como contrato de aquisição, locação de bens móveis ou aquisição de serviços; e o prazo de vigência até 1 ano, inclusive, a contar da decisão de adjudicação;
- Todos os contratos têm de respeitar a regra dos 3 anos, ou seja, 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.

3.2 Predicados

Nesta secção será explicitada e comprovada a construção da extensão dos predicados utilizados, essencialmente, para a representação do conhecimento positivo neste exercício.

3.2.1 adjudicante/4

O predicado *adjudicante: #IdAd, Nome, NIF, Morada* $\rightsquigarrow$ { $\mathbb{V}$, $\mathbb{F}$, $\mathbb{D}$ } tem como objetivo caracterizar um adjudicante existente na base de conhecimento através do seu identificador, nome, NIF e morada.

Com o objetivo de consultar o conhecimento adquirido sobre adjudicante/4, podemos recorrer ao predicado 'listing(adjudicante).'.

```
| ?- listing(adjudicante).
adjudicante(1, municipio_altodebasto, 705330336, portugal_braga_altodebasto).
adjudicante(2, municipio_guarda, 700000000, portugal_guarda).
adjudicante(3, municipio_vilaverde, 700000001, portugal_braga_vilaverde).
adjudicante(4, municipio_viseu, xpto1, portugal_viseu).
adjudicante(7, municipio_lisboa, interdito, portugal_lisboa).

yes
```

Figura 1: Apresentação dos conhecimentos adquiridos relativamente ao adjudicante

3.2.2 adjudicataria/4

O predicado *adjudicataria: #IdAd, Nome, NIF, Morada* $\rightsquigarrow$ { $\mathbb{V}$, $\mathbb{F}$, $\mathbb{D}$ } tem como objetivo caracterizar uma adjudicatária existente na base de conhecimento através do seu identificador, nome, NIF e morada.

Com o objetivo de consultar o conhecimento adquirido sobre adjudicataria/4, podemos recorrer ao predicado 'listing(adjudicataria).'.

```
| ?- listing(adjudicataria).
adjudicataria(1, xxx_associados_sociedadedeadvogados_sp_rl, 702675112, portugal).
adjudicataria(2, mm_lda, 711111111, portugal).
adjudicataria(3, rollido_companhia, 711111112, portugal).
adjudicataria(4, xpto2, 711111121, portugal).
adjudicataria(6, interdito2, 711111141, portugal).

yes
```

Figura 2: Apresentação dos conhecimentos adquiridos relativamente à adjudicatária

3.2.3 contrato/12

O predicado *contrato: #IdAd, #IdAda, IdC, TipoDeContrato, TipoDeProcedimento, Descrição, Custo, Prazo, Local, Data* $\rightsquigarrow$ {$\mathbb{V}$, $\mathbb{F}$, $\mathbb{D}$} tem como objetivo caracterizar um contrato existente na base de conhecimento através do identificador do adjudicante, identifcador da adjucicatária, seu identificador, tipo de contrato, tipo de procedimento, descrição, custo, prazo, local e data.

Com o objetivo de consultar o conhecimento adquirido sobre contrato/10, podemos recorrer ao predicado 'listing(contrato).'.

```
| ?- listing(contrato).
contrato(1, 1, 1, aquisicao_de_servicos, consulta_previa, assessoriajuridica, 15000, 547, altodebasto, 1, 1, 2010).
contrato(1, 2, 2, locacao_de_bens_moveis, ajuste_direto, assessoriajuridica, 2000, 240, altodebasto, 2, 2, 2000).
contrato(2, 3, 3, contrato_de_aquisicao, ajuste_direto, assessoriajuridica, 4000, 360, braga, 3, 3, 2010).
contrato(2, 1, 4, locacao_de_bens_moveis, ajuste_direto, assessoriajuridica, 4000, xpto3, braga, 14, 4, 2000).
contrato(1, 2, 6, locacao_de_bens_moveis, consulta_previa, assessoriajuridica, 5000, interdito3, porto, 16, 6, 2017).

yes
```

Figura 3: Apresentação dos conhecimentos adquiridos relativamente ao contrato

3.2.4 evolucao/1

O predicado evolucao: Termo $\rightsquigarrow$ {$\mathbb{V}$, $\mathbb{F}$} é o predicado responsável por adicionar informação à base de conhecimento, mais concretamente, registar adjudicantes, adjudicatárias e contratos, respeitando sempre os invariantes de inserção desenvolvidos.

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

3.2.5 involucao/1

O predicado involucao: Termo $\rightsquigarrow$ {$\mathbb{V}$, $\mathbb{F}$} é o predicado responsável por retirar informação à base de conhecimento, mais concretamente, remover adjudicantes, adjudicatárias e contratos, respeitando sempre os invariantes de remoção desenvolvidos.

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

3.2.6 contratosAdjudicante/2

O contratosAdjudicante: Id, Resultado $\rightsquigarrow \{\mathbb{V}, \mathbb{F}\}$ identifica a lista dos contratos de um determinado adjudicante. De seguida, apresentamos um exemplo para os resultados obtidos:

```
| ?- contratosAdjudicante(2,R).
R = [contrato(_A,3,3,contrato_de_aquisicao,ajuste_direto,assessoriajuridica,400
0,360,braga,3,3,2010),contrato(_B,1,4,locacao_de_bens_moveis,ajuste_direto,asse
ssoriajuridica,4000,xpto3,braga,14,4,2000)] ? y
yes
```

Figura 4: Lista dos contratos de um adjudicante

Para o predicado contratosAdjudicataria/2 o raciocínio é o mesmo, sendo que este identifica a lista dos contratos de uma determinada adjudicatária.

3.2.7 quantosContratos/3

O quantosContratos: Id1, Id2, Resultado $\rightsquigarrow \{\mathbb{V}, \mathbb{F}\}$ identifica o total de contratos realizados entre um adjudicante e uma adjudicataria. De seguida, apresentamos um exemplo para os resultados obtidos:

```
| ?- quantosContratos(1,1,R).
R = 1 ? y
yes
```

Figura 5: Número total de contratos entre um adjudicante e uma adjudicataria

3.2.8 valorAcumAdjudicante/2

O valorAcumAdjudicante: Id, Resultado $\rightsquigarrow \{\mathbb{V}, \mathbb{F}\}$ identifica o valor total dos contratos celebrados por um adjudicante. De seguida, apresentamos um exemplo para os resultados obtidos:

```
| ?- valorAcumAdjudicante(2,R).
R = 8000 ? y
yes
```

Figura 6: Valor total dos contratos celebrados por um adjudicante

Para o predicado valorAcumAdjudicataria/2 o raciocínio é o mesmo, sendo que este identifica o valor total dos contratos celebrados por uma adjudicatária.

3.2.9 tipoContAdjudicataria/2

O tipoContAdjudicataria: Id, Resultado $\rightsquigarrow$ {$\mathbb{V}$, $\mathbb{F}$} identifica os tipos de contrato dos contratos celebrados por uma adjudicatária. De seguida, apresentamos um exemplo para os resultados obtidos:

```
| ?- tipoContAdjudicataria(1,R).
R = [aquisicao_de_servicos,locacao_de_bens_moveis] ?
yes
```

Figura 7: Tipos de contrato dos contratos celebrados por uma adjudicataria

Para o predicado tipoContAdjudicante/2 o raciocínio é o mesmo, sendo que este identifica os tipos de contrato dos contratos celebrados por um adjudicante.

3.2.10 atualizaMorada/2

O atualizaMorada: Morada, Resultado $\rightsquigarrow$ {$\mathbb{V}$, $\mathbb{F}$} atualiza a morada de uma adjudicatária. De seguida, apresentamos um exemplo para os resultados obtidos:

```
| ?- atualizaMorada(2, portugal_madeira).
yes
| ?- listing(adjudicataria).
adjudicataria(1, xxx_associados_sociedadedeadvogados_sp_rl, 702675112, portugal).
adjudicataria(3, rollido_companhia, 711111112, portugal).
adjudicataria(4, xpto2, 711111121, portugal).
adjudicataria(6, interdito2, 711111141, portugal).
adjudicataria(2, mm_lda, 711111111, portugal_madeira).

yes
```

Figura 8: Atualização da morada de uma adjudicataria

3.2.11 atualizaNome/2

O atualizaNome: Nome, Resultado $\rightsquigarrow$ {$\mathbb{V}$, $\mathbb{F}$} atualiza o nome de uma adjudicatária. De seguida, apresentamos um exemplo para os resultados obtidos:

```
| ?- atualizaNome(3, vidraria_pereira).
yes
| ?- listing(adjudicataria).
adjudicataria(1, xxx_associados_sociedadedeadvogados_sp_rl, 702675112, portugal).
adjudicataria(4, xpto2, 711111121, portugal).
adjudicataria(6, interdito2, 711111141, portugal).
adjudicataria(2, mm_lda, 711111111, portugal_madeira).
adjudicataria(3, vidraria_pereira, 711111112, portugal).

yes
```

Figura 9: Atualização do nome de uma adjudicataria

3.2.12 contratosLocal/2

O contratosLocal: Local, Resultado $\rightsquigarrow \{\mathbb{V}, \mathbb{F}\}$ identifica a lista dos contratos realizados em determinado local. De seguida, apresentamos um exemplo para os resultados obtidos:

```
| ?- contratosLocal(braga,R).
R = [contrato(2,3,3,contrato_de_aquisicao,ajuste_direto,assessoriajuridica,4000,360,_A,3,3,2010),
contrato(2,1,4,locacao_de_bens_moveis,ajuste_direto,assessoriajuridica,4000,xpto3,_B,14,4,2000)]
? y
yes
```

Figura 10: Lista dos contratos realizados em Braga

3.2.13 contratosValor/3

O contratosValor: Valor, Operacao, Resultado $\rightsquigarrow \{\mathbb{V}, \mathbb{F}\}$ identifica a lista dos contratos com valor superior/inferior a determinada referência. De seguida, apresentamos um exemplo para os resultados obtidos:

```
| ?- contratosValor(4000, >, R).
R = [contrato(1,1,1,aquisicao_de_servicos,consulta_previa,assessoriajuridica,15000,547,altodebasto,1,1,2010),
contrato(1,2,6,locacao_de_bens_moveis,consulta_previa,assessoriajuridica,5000,interdito3,porto,16,6,2017)] ?
y
yes
```

Figura 11: Lista dos contratos com valor superior a 4000

3.2.14 contratosData/4

O contratosData: Dia, Mes, Ano, Resultado $\rightsquigarrow \{\mathbb{V}, \mathbb{F}\}$ identifica a lista dos contratos que aconteceram numa determinada data. De seguida, apresentamos um exemplo para os resultados obtidos:

```
| ?- contratosData(16,6,2017,R).
R = [contrato(1,2,6,locacao_de_bens_moveis,consulta_previa,assessoriajuridica,5000,interdito3,porto,_A,_B,_C)] ? y
yes
```

Figura 12: Lista dos contratos que aconteceram no dia 16-06-2017

3.2.15 opcoesAdjudicataria/2

O opcoesAdjudicataria: Opcao, Resultado $\rightsquigarrow \{\mathbb{V}, \mathbb{F}\}$ identifica a lista dos argumentos das adjudicatarias conforme opção. De seguida, apresentamos um exemplo para os resultados obtidos:

```
| ?- opcoesAdjudicataria(3,R).
R = [702675112,711111111,711111112,711111121,711111141] ? y
yes
```

Figura 13: Lista dos NIF's das adjudicatárias

Para o predicado opcoesAdjudicante/2 o raciocínio é o mesmo, sendo que este identifica a lista dos argumentos dos adjudicantes conforme opção.

3.3 Manipulação de Invariantes

Nesta secção será apresentada a construção dos invariantes desenvolvidos neste exercício numa fase inicial(estruturais e referenciais), sendo que mais alguns surgirão, posteriormente, ao abordarmos o conhecimento negativo e imperfeito. Na construção destes, são duas as componentes a ter em conta, a estrutura do conhecimento e a referência a informação contida na base de conhecimento.

3.3.1 Estruturais

Os invariantes estruturais de inserção que se apresentam de seguida permitem que seja adicionado à base de conhecimento um adjudicante/adjudicatária/contrato cujo ID seja único, ou seja, não permite adjudicantes/adjudicatárias/contratos com ID's repetidos.

1. Não podem existir adjudicantes com o mesmo ID

```
+adjudicante(Id, _, _, _) ::
        (solucoes((Id),(adjudicante(Id, Nome, NIF, M)),S),
        comprimento(S,N), N =< 1).
```

Como podemos ver na figura abaixo, ao tentar inserir um adjudicante com o ID 1, que já se encontra atribuído, este não será adicionado à base de conhecimento.

```
| ?- evolucao(adjudicante(1, municipio_braganca, 744444422, portugal_braganca)).
no
| ?- listing(adjudicante).
adjudicante(1, municipio_altodebasto, 705330336, portugal_braga_altodebasto).
adjudicante(2, municipio_guarda, 700000000, portugal_guarda).
adjudicante(3, municipio_vilaverde, 700000001, portugal_braga_vilaverde).
adjudicante(4, municipio_viseu, xpto1, portugal_viseu).
adjudicante(7, municipio_lisboa, interdito, portugal_lisboa).

yes
```

Figura 14: Invariante 1

2. Não podem existir adjudicatárias com o mesmo ID

```
+adjudicataria(Id, _, _, _) ::
        (solucoes((Id),(adjudicataria(Id, Nome, NIF, M)),S),
        comprimento(S,N), N =< 1).
```

Como podemos ver na figura abaixo, ao tentar inserir uma adjudicataria com o ID 1, que já se encontra atribuído, este não será adicionado à base de conhecimento.

```
| ?- evolucao(adjudicataria(1, serralharia_oliveira, 744444522, portugal_acores)).
no
| ?- listing(adjudicataria).
adjudicataria(1, xxx_associados_sociedadedeadvogados_sp_rl, 702675112, portugal).
adjudicataria(2, mm_lda, 711111111, portugal).
adjudicataria(3, rollido_companhia, 711111112, portugal).
adjudicataria(4, xpto2, 711111121, portugal).
adjudicataria(6, interdito2, 711111141, portugal).

yes
```

Figura 15: Invariante 2

3. Não podem existir contratos com o mesmo ID

```
+contrato(_, _, IdC, _, _, _, _, _, _, _, _, _) ::
        (solucoes((IdC),(contrato(IdAd, IdAda, IdC, TC, TP, D, V, P, L,
        DI, M, A)),S),
        comprimento(S,N),
        N =< 1).
```

Como podemos ver na figura abaixo, ao tentar inserir um contrato com o ID 1, que já se encontra atribuído, este não será adicionado à base de conhecimento.

```
| ?- listing(contrato).
contrato(1, 1, 1, aquisicao_de_servicos, consulta_previa, assessoriajuridica, 15000, 547, altodebasto, 1, 1, 2010).
contrato(1, 2, 2, locacao_de_bens_moveis, ajuste_direto, assessoriajuridica, 2000, 240, altodebasto, 2, 2, 2000).
contrato(2, 3, 3, contrato_de_aquisicao, ajuste_direto, assessoriajuridica, 4000, 360, braga, 3, 3, 2010).
contrato(2, 1, 4, locacao_de_bens_moveis, ajuste_direto, assessoriajuridica, 4000, xpto3, braga, 14, 4, 2000).
contrato(1, 2, 6, locacao_de_bens_moveis, consulta_previa, assessoriajuridica, 5000, interdito3, porto, 16, 6, 2017).

yes
```

Figura 16: Invariante 3

3.3.2 Referenciais

Os invariantes referenciais de remoção 1. e 2. apenas permitem remover adjudicantes/adjudicatárias, casos estes não tenham um contrato associado. O invariante 3. permite apenas que um contrato seja adicionado, caso o adjudicante e a adjudicatária nele contidos estiverem na base de conhecimento. Os invariantes 4., 5. e 6. foram os indicados no documento de apoio a este exercício.

1. Não permitir remover um adjudicante caso tenha contratos associados

```
-adjudicante(Id, _, _, _) ::
        (solucoes((Id),(contrato(Id, IdAda, IdC, TC, TP, D, V, P, L, DT)),S),
        comprimento(S,N), N == 0).
```

Como podemos ver na figura abaixo, ao tentar remover o adjudicante 2 que contém associado a si os contratos 3 e 4, este não será removido na base de conhecimento.

```
| ?- involucao(adjudicante(2, municipio_guarda, 700000000, portugal_guarda)).
no
| ?- listing(adjudicante).
adjudicante(1, municipio_altodebasto, 705330336, portugal_braga_altodebasto).
adjudicante(3, municipio_vilaverde, 700000001, portugal_braga_vilaverde).
adjudicante(4, municipio_viseu, xpto1, portugal_viseu).
adjudicante(7, municipio_lisboa, interdito, portugal_lisboa).
adjudicante(2, municipio_guarda, 700000000, portugal_guarda).

yes
| ?- listing(contrato).
contrato(1, 1, 1, aquisicao_de_servicos, consulta_previa, assessoriajuridica, 15000, 547, altodebasto, 1, 1, 2010).
contrato(1, 2, 2, locacao_de_bens_moveis, ajuste_direto, assessoriajuridica, 2000, 240, altodebasto, 2, 2, 2000).
contrato(2, 3, 3, contrato_de_aquisicao, ajuste_direto, assessoriajuridica, 4000, 360, braga, 3, 3, 2010).
contrato(2, 1, 4, locacao_de_bens_moveis, ajuste_direto, assessoriajuridica, 4000, xpto3, braga, 14, 4, 2000).
contrato(1, 2, 6, locacao_de_bens_moveis, consulta_previa, assessoriajuridica, 5000, interdito3, porto, 16, 6, 2017).

yes
```

Figura 17: Invariante 4

2. Não permitir remover uma adjudicatária caso tenha contratos associados

```
-adjudicataria(Id, _, _, _) ::
        (solucoes((Id),(contrato(IdAd, Id, IdC, TC, TP, D, V, P, L, DT)),S),
        comprimento(S,N), N == 0).
```

Como podemos ver na figura abaixo, ao tentar remover a adjudicataria 2 que contém associado a si os contratos 2 e 6, esta não será removido na base de conhecimento.

```
| ?- involucao(adjudicataria(2, mm_lda, 711111111, portugal)).
no
| ?- listing(adjudicataria).
adjudicataria(1, xxx_associados_sociedadedeadvogados_sp_rl, 702675112, portugal).
adjudicataria(3, rollido_companhia, 711111112, portugal).
adjudicataria(4, xpto2, 711111121, portugal).
adjudicataria(6, interdito2, 711111141, portugal).
adjudicataria(2, mm_lda, 711111111, portugal).

yes
| ?- listing(contrato).
contrato(1, 1, 1, aquisicao_de_servicos, consulta_previa, assessoriajuridica, 15000, 547, altodebasto, 1, 1, 2010).
contrato(1, 2, 2, locacao_de_bens_moveis, ajuste_direto, assessoriajuridica, 2000, 240, altodebasto, 2, 2, 2000).
contrato(2, 3, 3, contrato_de_aquisicao, ajuste_direto, assessoriajuridica, 4000, 360, braga, 3, 3, 2010).
contrato(2, 1, 4, locacao_de_bens_moveis, ajuste_direto, assessoriajuridica, 4000, xpto3, braga, 14, 4, 2000).
contrato(1, 2, 6, locacao_de_bens_moveis, consulta_previa, assessoriajuridica, 5000, interdito3, porto, 16, 6, 2017).

yes
```

Figura 18: Invariante 5

3. **Não permitir inserir um contrato caso o adjudicante e/ou adjudicatária correspondentes não estejam na base de conhecimento**

```
+contrato(IdAd, IdAda, _, _, _, _, _, _, _, _, _, _) ::
        (solucoes((IdAd),(adjudicante(IdAd, Nome, NIF, M)),S),
        comprimento(S,N), N == 1,
        solucoes((IdAda),(adjudicataria(IdAda, Nome1, NIF1, M1)),S1),
        comprimento(S1,N1), N1 == 1).
```

Como podemos ver na figura abaixo, ao tentar inserir o contrato 8 que contém associado a si os adjudicante com ID 5 e adjudicataria com ID 5, como estas entidades não estão na base de conhecimento, este não será inserido na base de conhecimento.

```
| ?- evolucao(contrato(5,5,8, aquisicao_de_servicos,consulta_previa, assessoriajuridica, 14000, 100, aveiro, 1.5, 2014)).
no
| ?- listing(contrato).
contrato(1, 1, 1, aquisicao_de_servicos, consulta_previa, assessoriajuridica, 15000, 547, altodebasto, 1, 1, 2010).
contrato(1, 2, 2, locacao_de_bens_moveis, ajuste_direto, assessoriajuridica, 2000, 240, altodebasto, 2, 2, 2000).
contrato(2, 3, 3, contrato_de_aquisicao, ajuste_direto, assessoriajuridica, 4000, 360, braga, 3, 3, 2010).
contrato(2, 1, 4, locacao_de_bens_moveis, ajuste_direto, assessoriajuridica, 4000, xpto3, braga, 14, 4, 2000).
contrato(1, 2, 6, locacao_de_bens_moveis, consulta_previa, assessoriajuridica, 5000, interdito3, porto, 16, 6, 2017).

yes
```

Figura 19: Invariante 6

4. **Não permitir inserir um contrato que não tenha o tipo de procedimento admissível**

```
+contrato(_, _, _, _, TP, _, _, _, _, _, _, _) ::
        (member(TP, [ajuste_direto, consulta_previa, concurso_publico])).
```

Como podemos ver na figura abaixo, ao tentar inserir o contrato 8 que contém associado a si o tipo de procedimento *servicos*, como este não é um tipo admissível, o contrato não será inserido na base de conhecimento.

```
| ?- evolucao(contrato(1,1,8, aquisicao_de_servicos, servicos, assessoriajurid
ca, 14000, 100, aveiro, 1.5, 2014)).
no
```

Figura 20: Invariante 7

5. **Não permitir inserir um contrato caso seja por ajuste direto e não respeite as condições obrigatórias do mesmo**

```
+contrato(_, _, _, TC, TP, _, V, P, _, _, _, _) ::
        ((member(TP, [consulta_previa, concurso_publico]));
        (TP == ajuste_direto,
                member(TC, [contrato_de_aquisicao, locacao_de_bens_moveis, aquisicao_de_servi
                V =< 5000,
                P =< 365)).
```

Como podemos ver na figura abaixo, ao tentar inserir o contrato 8 que contém associado a si o tipo de procedimento *ajuste_direto*, como este tem um valor que ultrapassa os 5000(valor mãximo obrigatório para este tipo de procedimento), o contrato não será inserido na base de conhecimento.

```
| ?- evolucao(contrato(1,1,8, aquisicao_de_servicos, ajuste_direto, assessoriaj
uridica, 14000, 100, aveiro, 1.5, 2014)).
no
```

Figura 21: Invariante 8

6. Não permitir inserir um contrato que não respeite a regra dos 3 anos

```
+contrato(_, _, _, TC, TP, _, V, P, _, _, _, _) ::
        ((member(TP, [consulta_previa, concurso_publico]));
        (TP == ajuste_direto,
                member(TC, [contrato_de_aquisicao, locacao_de_bens_moveis,
                aquisicao_de_servicos]),
                V =< 5000,
                P =< 365)).
```

Como podemos ver na figura abaixo, ao tentar inserir o contrato 8 que contém associado a si o tipo de procedimento *ajuste_direto* e o tipo de contrato *aquisicao_de_servicos*, como os contratos já existentes com esses tipos de procedimento e/ou contrato nos últimos dois anos, o atual inclusive, somam uma quantia de valores superior a 75000, o contrato não será inserido na base de conhecimento, pois não respeita a regra dos 3 anos.

```
| ?- evolucao(contrato(1,1,8, aquisicao_de_servicos, ajuste_direto, assessoriajuridica, 4000, 100, aveiro, 1,5, 2012)).
no
| ?- listing(contrato).
contrato(1, 1, 1, aquisicao_de_servicos, consulta_previa, assessoriajuridica, 72000, 547, altodebasto, 1, 1, 2010).
contrato(1, 2, 2, locacao_de_bens_moveis, ajuste_direto, assessoriajuridica, 2000, 240, altodebasto, 2, 2, 2000).
contrato(2, 3, 3, contrato_de_aquisicao, ajuste_direto, assessoriajuridica, 4000, 360, braga, 3, 3, 2010).
contrato(2, 1, 4, locacao_de_bens_moveis, ajuste_direto, assessoriajuridica, 4000, xpto3, braga, 14, 4, 2010).
contrato(1, 2, 6, locacao_de_bens_moveis, consulta_previa, assessoriajuridica, 5000, interdito3, porto, 16, 6, 2017).

yes
```

Figura 22: Invariante 9

3.4 Conhecimento Negativo

Após abordado e representado o conhecimento positivo, nesta secção, vamos trabalhar o conhecimento negativo.

A representação de conhecimento negativo tem por base a negação forte, que indica que um facto é considerado falso caso não exista na base de conhecimento e não tenha uma exceção associada, ou seja, a informação que não seja verdadeira nem desconhecida é considerada falsa.

Usamos, portanto, o **nao** como meta-predicado responsável por provar a veracidade da negação de uma questão na base de conhecimento, através da negação por falha na prova.

```
nao(Questao) :- Questao, !, fail.
nao(Questao).
```

De seguida, exemplificamos a extensão da negação forte do predicado adjudicante, sendo que foi utilizado o mesmo raciocínio para os predicados adjudicataria e contrato.

```
-adjudicante(Id, Nome, NIF, M) :-
    nao(adjudicante(Id, Nome, NIF, M)),
    nao(excecaoIncerta(adjudicante(Id, Nome, NIF, M))),
    nao(excecaoImprecisa(adjudicante(Id, Nome, NIF, M))),
    nao(excecaoImprecisaIntervalo(adjudicante(Id, Nome, NIF, M),_,_)),
    nao(excecaoInterdita(adjudicante(Id, Nome, NIF, M))).
```

Com o objetivo de impedir incoerência na base de conhecimento, provocada pela inserção/remoção de conhecimento, foram criados novos invariantes:

1. Invariante que impede a inserção de conhecimento negativo repetido

```
+(-X) :: (solucoes(X, clause(-X, true), S),
    comprimento(S,N),
    N =< 1).
```

2. **Invariantes que não permitam a existência do mesmo conhecimento negativo e positivo em simultâneo:**

```
+X :: nao(-X).
+(-X) :: nao(X).
```

3.5 Conhecimento Imperfeito

Como referimos anteriormente, neste exercício foram representadas três formas distintas de conhecimento imperfeito: incerto, impreciso e nulo.

Usamos, portanto, de modo a representar conhecimento desconhecido, o meta-predicado **excecao**. No entanto, este tipo de conhecimento tanto poderá estar associado a uma **excecao**, como a um **valor nulo**.

Com o objetivo de saber o tipo de conhecimento representado por uma questão, utilizamos o meta-predicado **si** que nos diz se o conhecimento é verdadeiro, falso ou desconhecido.

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

3.5.1 Conhecimento Incerto

Neste caso, não se possui qualquer informação relativamente à informação desconhecida. De maneira a caracterizar o desconhecido recorremos a um valor nulo(*xpto*) e associamos o mesmo a uma **excecao**. Vejamos então o exemplo de conhecimento incerto para o predicado adjudicante:

```
adjudicante(4, municipio_altodebasto, xpto1, portugal_braga_altodebasto).
excecao(adjudicante(Id, Nome, NIF, M)) :- adjudicante(Id, Nome, xpto1, M).
```

Este exemplo acima indica que para o adjudicante 4, o NIF é desconhecido. Além disso, podemos comprovar o mesmo na figura abaixo, em que o NIF para este adjudicante, independentemente de qualquer o valor questionado, é desconhecido.

```
| ?- si(adjducicante(4, municipio_altodebasto, 70000099, portugal_braga_altodebasto),R).
R = desconhecido ?
yes
```

Figura 23: Conhecimento incerto relativamente ao NIF do adjudicante com ID 4

Para os restantes predicados adjudicataria e contrato, o conhecimento incerto foi representado com o mesmo raciocínio.

3.6 Conhecimento Impreciso

Neste segundo caso, o conhecimento é desconhecido dentro de ou entre um conjunto de valores, ou seja, temos dois casos distintos e, portanto, para um melhor funcionamento do sistema, consideramo-los como tal com dois tipos de exceções diferentes. Portanto, qualquer valor que não esteja nesse conjunto de valores e não esteja na base de conhecimento é considerado falso. Recorremos, então, às exceções para representar os casos que abrangem esse conjunto de valores.

Vejamos então os dois casos de conhecimento impreciso para o predicado adjudicante:

```
excecaoImprecisa(adjudicante(5, municipio_braga, 700000010, portugal_braga)).
excecaoImprecisa(adjudicante(5, municipio_braga, 700000020, portugal_braga)).

excecaoImprecisaIntervalo(adjudicante(6, municipio_porto, NIF, portugal_porto),
        700000030, 700000050) :- NIF >= 700000030, NIF =< 700000050.
```

Observamos que, no primeiro caso, o NIF do adjudicante encontra-se em dúvida entre *700000010* e *700000020*.

Relativamente ao segundo caso, podemos comprovar a sua veracidade na figura abaixo, através do meta-predicado **si**.

```
| ?- si(adjudicante(6, municipio_porto, 700000020, portugal_porto),R).
R = falso ?
yes
| ?- si(adjudicante(6, municipio_porto, 700000040, portugal_porto),R).
R = desconhecido ?
yes
```

Figura 24: Conhecimento impreciso relativamente ao NIF do adjudicante com ID 4 entre 700000030 e 70000050

Para os restantes predicados adjudicataria e contrato, o conhecimento impreciso foi representado seguindo o mesmo raciocínio, sendo que para a adjudicataria não foi considerado conhecimento impreciso com intervalos.

3.7 Conhecimento Interdito

No último caso, o conhecimento é interdito, uma vez que este não deve ser conhecido nem especificado. De modo a interditar este conhecimento, recorremos a um valor nulo(*interdito*) e associamos o mesmo a uma **excecao**. Além disso, como não deve ser permitido incluir informação relativamente a este conhecimento, foi utilizado o meta-predicado **nuloInterdito** visando a caracterização destes valores nulos.

Vejamos então o seguinte exemplo da representação de conhecimento interdito para o predicado contrato:

```
contrato(1, 2, 5, interdito3, consulta_previa, assessoriajuridica, 5000, 400,
    porto, 16-06-2010).
excecao(contrato(IdAd, IdAda, IdC, TC, TP, D, V, P, L, DT)) :-
        contrato(IdAd, IdAda, IdC, interdito3, TP, D, V, P, L, DT).
nuloInterdito(interdito3).
```

Como vemos na figura, foi utilizado o valor nulo interdito3 ao parâmetro *prazo*, de modo a tornar essa informação interdita. Além disto, foi necessária a criação de um invariante que impedisse a inserção de factos relativamente ao *prazo* deste contrato.

Conseguimos comprovar isto referido na figura abaixo.

```
| ?- listing(contrato).
contrato(1, 1, 1, aquisicao_de_servicos, consulta_previa, assessoriajuridica, 72000, 547, altodebasto, 1, 1, 2010).
contrato(1, 2, 2, locacao_de_bens_moveis, ajuste_direto, assessoriajuridica, 2000, 240, altodebasto, 2, 2, 2000).
contrato(2, 3, 3, contrato_de_aquisicao, ajuste_direto, assessoriajuridica, 4000, 360, braga, 3, 3, 2010).
contrato(2, 1, 4, locacao_de_bens_moveis, ajuste_direto, assessoriajuridica, 4000, xpto3, braga, 14, 4, 2010).
contrato(1, 2, 6, locacao_de_bens_moveis, consulta_previa, assessoriajuridica, 5000, interdito3, porto, 16, 6, 2017).

yes
| ?- evolucao(contrato(1,2,6, locacao_de_bens_moveis, consulta_previa, assessoriajuridica, 5000, 30, porto, 16,6, 2017)).
no
| ?- evolucao(contrato(1,2,9, locacao_de_bens_moveis, consulta_previa, assessoriajuridica, 5000, 30, porto, 16,6, 2017)).
yes
| ?- listing(contrato).
contrato(1, 1, 1, aquisicao_de_servicos, consulta_previa, assessoriajuridica, 72000, 547, altodebasto, 1, 1, 2010).
contrato(1, 2, 2, locacao_de_bens_moveis, ajuste_direto, assessoriajuridica, 2000, 240, altodebasto, 2, 2, 2000).
contrato(2, 3, 3, contrato_de_aquisicao, ajuste_direto, assessoriajuridica, 4000, 360, braga, 3, 3, 2010).
contrato(2, 1, 4, locacao_de_bens_moveis, ajuste_direto, assessoriajuridica, 4000, xpto3, braga, 14, 4, 2010).
contrato(1, 2, 6, locacao_de_bens_moveis, consulta_previa, assessoriajuridica, 5000, interdito3, porto, 16, 6, 2017).
contrato(1, 2, 9, locacao_de_bens_moveis, consulta_previa, assessoriajuridica, 5000, 30, porto, 16, 6, 2017).

yes
```

Figura 25: Conhecimento interdito relativamente ao Prazo do contrato com ID 6

Para a representação de conhecimento interdito para os predicados adjudicante e adjudicataria foi utilizado o mesmo raciocínio.

Com o objetivo de impedir incoerência na base de conhecimento, provocada pela inserção/remoção de conhecimento, foram criados novos invariantes:

1. Invariante relativo a conhecimento nulo interdito

```
+contrato(IdAd, IdAda, IdC, TC, TP, D, V, P, L, DI, M, A) ::
        (solucoes(Px,(contrato(1, 2, 6, locacao_de_bens_moveis,
        consulta_previa, assessoriajuridica, 5000, Px, porto, 16, 06, 2017),
                nao(nuloInterdito(Px))), S),
                comprimento(S,N), N == 0).
```

3.8 Evolução do Conhecimento

Nesta secção, vamos abordar o problema da evolução do conhecimento e a sua resolução. Recorremos, portanto, à criação de meta-predicados que permitissem a atualização do conhecimento já existente na base de conhecimento, bem como a inserção de novo de conhecimento, positivo, negativo ou imperfeito. Para realizar esta tarefa tivemos que ter em conta vários fatores, que serão ao longo desta secção apresentados e explicados.

Para as provas de resultados obtidos que iremos apresentar, a constituição da base de conhecimento que tinhamos inicialmente foi alargada com conhecimento perfeito e servirá como referência.

```
| ?- listing(excecaoIncerta).
excecaoIncerta(adjudicante(A,B,_,C)) :-
        adjudicante(A, B, xpto1, C).
excecaoIncerta(adjudicataria(A,_,B,C)) :-
        adjudicataria(A, xpto2, B, C).
excecaoIncerta(contrato(A,B,C,D,E,F,G,_,H,I,J,K)) :-
        contrato(A, B, C, D, E, F, G, xpto3, H, I, J, K).

yes
| ?- listing(excecaoImprecisa).
excecaoImprecisa(adjudicante(5,municipio_braga,700000010,portugal_braga)).
excecaoImprecisa(adjudicante(5,municipio_braga,700000020,portugal_braga)).
excecaoImprecisa(adjudicataria(5,peugaspinto,711111131,portugal)).
excecaoImprecisa(adjudicataria(5,peugastinto,711111131,portugal)).
excecaoImprecisa(contrato(1,2,5,contrato_de_aquisicao,ajuste_direto,assessoriajuridica,5000,280,lisboa,15,5,2005)).
excecaoImprecisa(contrato(1,2,5,contrato_de_aquisicao,ajuste_direto,assessoriajuridica,5000,170,lisboa,15,5,2005)).

yes
| ?- listing(excecaoImprecisaIntervalo).
excecaoImprecisaIntervalo(adjudicante(6,municipio_porto,A,portugal_porto), 700000030, 700000050) :-
        A>=700000030,
        A=<700000050.
excecaoImprecisaIntervalo(contrato(3,2,7,urgente,concurso_publico,assessoriajuridica,200000,A,coimbra,15,5,2018), 100, 400) :-
        A>=100,
        A=<400.

yes
| ?- listing(excecaoInterdita).
excecaoInterdita(adjudicante(A,B,_,C)) :-
        adjudicante(A, B, interdito, C).
excecaoInterdita(adjudicataria(A,_,B,C)) :-
        adjudicataria(A, interdito2, B, C).
excecaoInterdita(contrato(A,B,C,D,E,F,G,_,H,I,J,K)) :-
        contrato(A, B, C, D, E, F, G, interdito3, H, I, J, K).

yes
```

Figura 26: Consituição da base de conhecimento(relativamente a conhecimento imperfeito)

3.8.1 Inserção de conhecimento perfeito

Consideramos que todo o conhecimento que não se encontre como positivo podia ser alterado, sendo que, passando este exercício a uma situação real, não faria sentido atualizar os dados dos adjudicantes/adjudicatarias/contratos que já se encontram como verdadeiros, a não ser o nome e morada das adjudicatárias, porém, esta componente, já foi abordada nas funcionalidades do sistema(através dos predicados *atualizaMorada* e *atualizaNome*). No entanto, o restante conhecimento pode ser atualizado e, portanto, foi desenvolvido o meta-predicado *atualizacao* que se certifica que sejam feitas as atualizações necessárias à base de conhecimento, antes de registar informação perfeita.

Um aspeto importante foi considerar o facto do conhecimento, na evolução do conhecimento, ser primeiro atualizado e só depois inserido, daí o motivo do nosso meta-predicado *evolucao* invocar primeiro o meta-predicado *atualizacao* e ser este depois o responsável por fazer a inserção respetiva.

1. Atualização de conhecimento incerto ou impreciso para positivo

Apesar de estar registado na base de conhecimento o desconhecimento acerca de determinado predicado, este pode ser atualizado. Sendo assim, considerou-se que qualquer conhecimento positivo inserido relavimanente a conhecimento incerto ou impreciso(com ou sem intervalos) seria correto e, consequentemente, as exceções deveriam ser, primeiramente, removidas da base de conhecimento e a inserção feita.

Este raciocínio, usado também para os predicados *adjudicataria* e *contrato* com o mesmmo formato, foi aplicado através da cláusula abaixo.

```
atualizacao(adjudicante(Id, Nome, NIF, Morada)) :-
        si(excecaoIncerta(adjudicante(Id, Nome, NIF, Morada)), verdadeiro),
        remocao(adjudicante(Id, Nome, xpto1, Morada)),
        remocao((excecaoIncerta(adjudicante(I, N, NI, M)) :-
                adjudicante(I, N, xpto1, M))),
        insercao(adjudicante(Id, Nome, NIF, Morada));
        si(excecaoImprecisa(adjudicante(Id, Nome, NIF, Morada)), verdadeiro),
        solucoes((excecaoImprecisa(adjudicante(_,_,_,_))),
                excecaoImprecisa(adjudicante(Id, Nome, _, Morada)), L),
        remocaoLista(L),
        insercao(adjudicante(Id, Nome, NIF, Morada));
        clause(excecaoImprecisaIntervalo(adjudicante(Id,_,_,_),_,_), R),
        remocao((excecaoImprecisaIntervalo(adjudicante(Id,_,N,_), X, Y) :-
                N >= X, N =<Y )),
        insercao(adjudicante(Id, Nome, NIF, Morada)).
```

De seguida, apresentamos um exemplo dos resultados obtidos a partir desta cláusula.

```
| ?- evolucao(adjudicante(4, municipio_viseu, 700009901, portugal_viseu)).
yes
| ?- listing(excecaoIncerta).
excecaoIncerta(adjudicataria(A,_,B,C)) :-
        adjudicataria(A, xpto2, B, C).
excecaoIncerta(contrato(A,B,C,D,E,F,G,_,H,I,J,K)) :-
        contrato(A, B, C, D, E, F, G, xpto3, H, I, J, K).

yes
| ?- listing(adjudicante).
adjudicante(1, municipio_altodebasto, 705330336, portugal_braga_altodebasto).
adjudicante(2, municipio_guarda, 700000000, portugal_guarda).
adjudicante(3, municipio_vilaverde, 700000001, portugal_braga_vilaverde).
adjudicante(7, municipio_lisboa, interdito, portugal_lisboa).
adjudicante(4, municipio_viseu, 700009901, portugal_viseu).

yes
```

Figura 27: Atualização(positiva) do ajudicante com ID 4

2. Inserção de conhecimento negativo relativamente a conhecimento incerto ou impreciso

Em primeiro lugar, em relação ao conhecimento incerto, este pode ser atualizado com conhecimento negativo, não sendo preciso realizar qualquer tipo de remoção, uma vez que podemos considerar que dentro do desconhecido não sabemos a informação verdadeira, mas podemos considerar que há valores que sabemos que são falsos. O mesmo acontece para o conhecimento impreciso especificado por um conjunto de valores. Porém, quando se trata de conhecimento impreciso definido por valores pontuais, é necessário efetuar primeiro uma remoção da excecão respetiva, visando a consistência da base de conhecimento.

Para tal, foi criada a cláusula abaixo apresentada.

```
atualizacao(-X) :- si(excecaoIncerta(X), verdadeiro),
                   insercao(-X);
                   si(excecaoImprecisa(X), verdadeiro),
                   remocao(excecaoImprecisa(X)),
                   insercao(-X);
                   clause(excecaoImprecisaIntervalo(X,_,_), R),
                   insercao(-X).
```

De seguida, apresentamos um exemplo dos resultados obtidos a partir desta cláusula.

```
| ?- evolucao(-adjudicante(4, municipio_viseu, 700009901, portugal_viseu)).
yes
| ?- listing(-).
-adjudicante(A,B,C,D) :-
        nao(adjudicante(A,B,C,D)),
        nao(excecaoIncerta(adjudicante(A,B,C,D))),
        nao(excecaoImprecisa(adjudicante(A,B,C,D))),
        nao(excecaoImprecisaIntervalo(adjudicante(A,B,C,D),_,_)),
        nao(excecaoInterdita(adjudicante(A,B,C,D))).
-adjudicataria(A,B,C,D) :-
        nao(adjudicataria(A,B,C,D)),
        nao(excecaoIncerta(adjudicataria(A,B,C,D))),
        nao(excecaoImprecisa(adjudicataria(A,B,C,D))),
        nao(excecaoImprecisaIntervalo(adjudicataria(A,B,C,D),_,_)),
        nao(excecaoInterdita(adjudicataria(A,B,C,D))).
-contrato(A,B,C,D,E,F,G,H,I,J,K,L) :-
        nao(contrato(A,B,C,D,E,F,G,H,I,J,K,L)),
        nao(excecaoIncerta(contrato(A,B,C,D,E,F,G,H,I,J,K,L))),
        nao(excecaoImprecisa(contrato(A,B,C,D,E,F,G,H,I,J,K,L))),
        nao(excecaoImprecisa(contrato(A,B,C,D,E,F,G,H,I,J,K,L)), _, _),
        nao(excecaoInterdita(contrato(A,B,C,D,E,F,G,H,I,J,K,L))).
-adjudicante(4,municipio_viseu,700009901,portugal_viseu).

yes
| ?- listing(excecaoIncerta).
excecaoIncerta(adjudicante(A,B,_,C)) :-
        adjudicante(A, B, xpto1, C).
excecaoIncerta(adjudicataria(A,_,B,C)) :-
        adjudicataria(A, xpto2, B, C).
excecaoIncerta(contrato(A,B,C,D,E,F,G,_,H,I,J,K)) :-
        contrato(A, B, C, D, E, F, G, xpto3, H, I, J, K).

yes
| ?- listing(adjudicante).
adjudicante(1, municipio_altodebasto, 705330336, portugal_braga_altodebasto).
adjudicante(2, municipio_guarda, 700000000, portugal_guarda).
adjudicante(3, municipio_vilaverde, 700000001, portugal_braga_vilaverde).
adjudicante(4, municipio_viseu, xpto1, portugal_viseu).
adjudicante(7, municipio_lisboa, interdito, portugal_lisboa).

yes
```

Figura 28: Atualização(negativa) do ajudicante com ID 4

3. Atualização de conhecimento negativo para positivo

Relativamente à informação negativa na base de conhecimento, esta pode ser, a qualquer momento, considerada positiva, desde que o negativo seja removido da base de conhecimento, de forma a não gerar contradições.

Este raciocínio, usado também para os predicados *adjudicataria* e *contrato* com o mesmmo formato, foi aplicado através da cláusula abaixo.

```
atualizacao(adjudicante(Id, Nome, NIF, Morada)) :-
        clause(excecaoIncerta(adjudicante(Id, _, _, _)), Q),
        si(adjudicante(Id,_, xpto1,_), verdadeiro),
        clause(-adjudicante(Id, Nome, NIF, Morada), true),
        solucoes((-adjudicante(Id, NO, N, M)), (-adjudicante(Id, _, _, _)), L1),
        remocaoLista(L1),
        remocao(adjudicante(Id, _, xpto1, _)),
        remocao((excecaoIncerta(adjudicante(I, N, NI, M)) :-
                adjudicante(I, N, xpto1, M))),
        insercao(adjudicante(Id, Nome, NIF, Morada));
        clause(excecaoImprecisaIntervalo(adjudicante(Id,_,_,_),_,_), R),
        clause(-adjudicante(Id, Nome, NIF, Morada), true),
        solucoes((-adjudicante(Id, NO, N, M)), (-adjudicante(Id, _, _, _)), L2),
        remocaoLista(L2),
        remocao((excecaoImprecisaIntervalo(adjudicante(Id,_,N,_), X, Y) :-
                N >= X, N =<Y )),
        insercao(adjudicante(Id, Nome, NIF, Morada));
        clause(-adjudicante(Id, Nome, NIF, Morada), true),
        solucoes((-adjudicante(Id, NO, N, M)), (-adjudicante(Id, _, _, _)), L3),
        remocaoLista(L3),
        solucoes((excecaoImprecisa(adjudicante(_,_,_,_))),
        excecaoImprecisa(adjudicante(Id, _, _, _)), L),
        remocaoLista(L),
        insercao(adjudicante(Id, Nome, NIF, Morada)).
```

4. Atualização de conhecimento positivo para negativo

Assim como a informação negativa na base de conhecimento pode dar lugar a conhecimento positiva, o contrário também acontece, desde que a informação desatualizada seja removida da base de conhecimento. Um aspeto importante a ter em conta é caso o facto a negar coincida com conhecimento incerto, sendo que este não poderá ser negado.

Portanto, foi criada a cláusula abaixo.

```
atualizacao(-X) :- clause(X, true),
                   si(excecaoIncerta(X), desconhecido),
                   remocao(X),
                   insercao(-X).
```

De seguida, apresentamos um exemplo dos resultados obtidos a partir destas duas últimas cláusulas.

```
| ?- evolucao(-adjudicante(3, municipio_vilaverde, 700000001, portugal_braga_vilaverde)).
yes
| ?- listing(-).
-adjudicante(A,B,C,D) :-
        nao(adjudicante(A,B,C,D)),
        nao(excecaoIncerta(adjudicante(A,B,C,D))),
        nao(excecaoImprecisa(adjudicante(A,B,C,D))),
        nao(excecaoImprecisaIntervalo(adjudicante(A,B,C,D),_,_)),
        nao(excecaoInterdita(adjudicante(A,B,C,D))).
-adjudicataria(A,B,C,D) :-
        nao(adjudicataria(A,B,C,D)),
        nao(excecaoIncerta(adjudicataria(A,B,C,D))),
        nao(excecaoImprecisa(adjudicataria(A,B,C,D))),
        nao(excecaoImprecisaIntervalo(adjudicataria(A,B,C,D),_,_)),
        nao(excecaoInterdita(adjudicataria(A,B,C,D))).
-contrato(A,B,C,D,E,F,G,H,I,J,K,L) :-
        nao(contrato(A,B,C,D,E,F,G,H,I,J,K,L)),
        nao(excecaoIncerta(contrato(A,B,C,D,E,F,G,H,I,J,K,L))),
        nao(excecaoImprecisa(contrato(A,B,C,D,E,F,G,H,I,J,K,L))),
        nao(excecaoImprecisa(contrato(A,B,C,D,E,F,G,H,I,J,K,L)), _, _),
        nao(excecaoInterdita(contrato(A,B,C,D,E,F,G,H,I,J,K,L))).
-adjudicante(3,municipio_vilaverde,700000001,portugal_braga_vilaverde).

yes
| ?- listing(adjudicante).
adjudicante(1, municipio_altodebasto, 705330336, portugal_braga_altodebasto).
adjudicante(2, municipio_guarda, 700000000, portugal_guarda).
adjudicante(4, municipio_viseu, xpto1, portugal_viseu).
adjudicante(7, municipio_lisboa, interdito, portugal_lisboa).

yes
| ?- evolucao(adjudicante(3, municipio_vilaverde, 700000001, portugal_braga_vilaverde)).
yes
| ?- listing(adjudicante).
adjudicante(1, municipio_altodebasto, 705330336, portugal_braga_altodebasto).
adjudicante(2, municipio_guarda, 700000000, portugal_guarda).
adjudicante(4, municipio_viseu, xpto1, portugal_viseu).
adjudicante(7, municipio_lisboa, interdito, portugal_lisboa).
adjudicante(3, municipio_vilaverde, 700000001, portugal_braga_vilaverde).

yes
```

Figura 29: Atualização(negativa) do ajudicante com ID 4 e, depois atualização(positiva) do mesmo

5. Inserção de conhecimento positivo novo

Caso a inserção de conhecimento positivo não seja relativa a informação já existente na base de conhecimento, temos que ter, porém, noção relativamente ao que estamos a inserir para não surgirem conflitos de existência de informação positiva e imprecisa com entidades com ID's iguais e informação diferente. Desta forma, antes de inserirmos conhecimento positivo novo, verificamos se existe conhecimento impreciso com ID igual ao que prentendemos inserir e, após isso, procedemos à inserção.

Posto isto, foi criada a cláusula apresentada de seguida, sendo que o mesmo raciocínio foi aplicada às entidades *contrato* e *adjudicataria*.

```
atualizacao(adjudicante(Id, Nome, NIF, Morada)) :-
        clause(excecaoImprecisaIntervalo(adjudicante(Id,_,_,_),_,_), R),
        remocao((excecaoImprecisaIntervalo(adjudicante(Id,_,N,_), X, Y) :-
                N >= X, N =<Y )),
        insercao(adjudicante(Id, Nome, NIF, Morada));
        solucoes((excecaoImprecisa(adjudicante(_,_,_,_))),
        excecaoImprecisa(adjudicante(Id, _, _, _)), L),
        remocaoLista(L),
        insercao(adjudicante(Id, Nome, NIF, Morada)).
```

De seguida, apresentamos um exemplo dos resultados obtidos a partir desta cláusula.

```
| ?- evolucao(adjudicante(10, municipio_guarda, 710000001, portugal_guarda)).
yes
| ?- listing(adjudicante).
adjudicante(1, municipio_altodebasto, 705330336, portugal_braga_altodebasto).
adjudicante(2, municipio_guarda, 700000000, portugal_guarda).
adjudicante(3, municipio_vilaverde, 700000001, portugal_braga_vilaverde).
adjudicante(4, municipio_viseu, xpto1, portugal_viseu).
adjudicante(7, municipio_lisboa, interdito, portugal_lisboa).
adjudicante(10, municipio_guarda, 710000001, portugal_guarda).

yes
```

Figura 30: Inserção positiva do ajudicante com ID 10

6. Inserção de conhecimento negativo novo

Caso a inserção de conhecimento negativo não seja relativa a informação já existente na base de conhecimento, então não e necessário atualizar a base de conhecimento. Desta forma a única tarefa necessária será a verificação dos invariantes, que é feita pelo meta-predicado *evolucao*, sendo depois a inserção feita pelo meta-predicado *atualizacao*.

Posto isto, foi desenvolvida a seguinte cláusula.

```
atualizacao(X) :- insercao(X).
```

De seguida, apresentamos um exemplo dos resultados obtidos a partir desta cláusula.

```
| ?- evolucao(-adjudicante(10, municipio_guarda, 710000001, portugal_guarda)).
yes
| ?- listing(-).
-adjudicante(A,B,C,D) :-
        nao(adjudicante(A,B,C,D)),
        nao(excecaoIncerta(adjudicante(A,B,C,D))),
        nao(excecaoImprecisa(adjudicante(A,B,C,D))),
        nao(excecaoImprecisaIntervalo(adjudicante(A,B,C,D),_,_)),
        nao(excecaoInterdita(adjudicante(A,B,C,D))).
-adjudicataria(A,B,C,D) :-
        nao(adjudicataria(A,B,C,D)),
        nao(excecaoIncerta(adjudicataria(A,B,C,D))),
        nao(excecaoImprecisa(adjudicataria(A,B,C,D))),
        nao(excecaoImprecisaIntervalo(adjudicataria(A,B,C,D),_,_)),
        nao(excecaoInterdita(adjudicataria(A,B,C,D))).
-contrato(A,B,C,D,E,F,G,H,I,J,K,L) :-
        nao(contrato(A,B,C,D,E,F,G,H,I,J,K,L)),
        nao(excecaoIncerta(contrato(A,B,C,D,E,F,G,H,I,J,K,L))),
        nao(excecaoImprecisa(contrato(A,B,C,D,E,F,G,H,I,J,K,L))),
        nao(excecaoImprecisa(contrato(A,B,C,D,E,F,G,H,I,J,K,L)), _, _),
        nao(excecaoInterdita(contrato(A,B,C,D,E,F,G,H,I,J,K,L))).
-adjudicante(10,municipio_guarda,710000001,portugal_guarda).

yes
```

Figura 31: Inserção negativa do ajudicante com ID 10

3.8.2 Inserção de conhecimento imperfeito incerto

Para o caso de estudo abordado, vamos apenas desenvolver as cláusulas do meta-predicado atualizarIncerto de forma a este inserir informação incerta relativamente ao NIF dos adjudicantes.Além disso, iremos mostrar alguns exemplos dos resultados obtidos através das cláusulas desenvolvidas.

1. Atualização de conhecimento positivo para incerto

Inicialmente, verificamos se existe informação positiva sobre o que queremos adicionar na base de conhecimento, sendo esta depois eliminada e é inserida a exceção com a informação incerta especificada, bem como o predicado *adjudicante* com o *xpto1* associado.

```
atualizacaoIncerto(adjudicante(Id, Nome, NIF, Morada), nif) :-
        si(adjudicante(Id, _, _, _),verdadeiro),
        nao(clause(-adjudicante(Id, Nome, NIF, Morada), true)),
        nao(clause(excecaoImprecisa(adjudicante(Id,_,_,_)), R)),
        nao(clause(excecaoImprecisaIntervalo(adjudicante(Id,_,_,_),_,_), R)),
        solucoes((adjudicante(Id, NO, N, M)), adjudicante(Id, _, _, _), L),
        comprimento(L,R), R == 1,
        remocaoLista(L),
        insercao((excecaoIncerta(adjudicante(I, NO, N, M)) :- adjudicante(I,NO, xpto1, M))),
        insercao(adjudicante(Id, Nome, xpto1, Morada)).
```

De seguida, apresentamos um exemplo dos resultados obtidos a partir desta cláusula.

```
| ?- atualizacaoIncerto(adjudicante(2, municipio_guarda, xpto1, portugal_guarda), nif).
yes
| ?- listing(adjudicante).
adjudicante(1, municipio_altodebasto, 705330336, portugal_braga_altodebasto).
adjudicante(3, municipio_vilaverde, 700000001, portugal_braga_vilaverde).
adjudicante(4, municipio_viseu, xpto1, portugal_viseu).
adjudicante(7, municipio_lisboa, interdito, portugal_lisboa).
adjudicante(2, municipio_guarda, xpto1, portugal_guarda).

yes
```

Figura 32: Atualização(incerta) do adjudicante com ID 2 que se encontrava como informação positiva

2. Atualização de conhecimento negativo para incerto

Inicialmente, verificamos se existe informação negativa sobre o que queremos adicionar na base de conhecimento, sendo esta depois eliminada e é inserida a exceção com a informação incerta especificada, bem como o predicado *adjudicante* com o *xpto1* associado.

```
atualizacaoIncerto(adjudicante(Id, Nome, NIF, Morada), nif) :-
        clause(-adjudicante(Id, _, _, _),true),
        nao(adjudicante(Id, _, _, _)),
        nao(clause(excecaoImprecisa(adjudicante(Id,_,_,_)), R)),
        nao(clause(excecaoImprecisaIntervalo(adjudicante(Id,_,_,_),_,_), R)),
        solucoes((-adjudicante(Id, NO, N, M)), (-adjudicante(Id, _, _, _)), L),
        comprimento(L,R), R >= 1,
        remocaoLista(L),
        insercao((excecaoIncerta(adjudicante(I, NO, N, M)) :-
                adjudicante(I,NO, xpto1, M))),
        insercao(adjudicante(Id, Nome, xpto1, Morada)).
```

De seguida, apresentamos um exemplo dos resultados obtidos a partir desta cláusula.

```
| ?- evolucao(-adjudicante(10, municipio_vilareal, 788888888, portugal_vilareal)).
yes
| ?- listing(-).
-adjudicante(A,B,C,D) :-
        nao(adjudicante(A,B,C,D)),
        nao(excecaoIncerta(adjudicante(A,B,C,D))),
        nao(excecaoImprecisa(adjudicante(A,B,C,D))),
        nao(excecaoImprecisaIntervalo(adjudicante(A,B,C,D),_,_)),
        nao(excecaoInterdita(adjudicante(A,B,C,D))).
-adjudicataria(A,B,C,D) :-
        nao(adjudicataria(A,B,C,D)),
        nao(excecaoIncerta(adjudicataria(A,B,C,D))),
        nao(excecaoImprecisa(adjudicataria(A,B,C,D))),
        nao(excecaoImprecisaIntervalo(adjudicataria(A,B,C,D),_,_)),
        nao(excecaoInterdita(adjudicataria(A,B,C,D))).
-contrato(A,B,C,D,E,F,G,H,I,J,K,L) :-
        nao(contrato(A,B,C,D,E,F,G,H,I,J,K,L)),
        nao(excecaoIncerta(contrato(A,B,C,D,E,F,G,H,I,J,K,L))),
        nao(excecaoImprecisa(contrato(A,B,C,D,E,F,G,H,I,J,K,L))),
        nao(excecaoImprecisa(contrato(A,B,C,D,E,F,G,H,I,J,K,L)), _, _),
        nao(excecaoInterdita(contrato(A,B,C,D,E,F,G,H,I,J,K,L))).
-adjudicante(10,municipio_vilareal,788888888,portugal_vilareal).

yes
| ?- atualizacaoIncerto(adjudicante(10, municipio_vilareal, 788888888, portugal_vilareal), nif).
yes
| ?- listing(-).
-adjudicante(A,B,C,D) :-
        nao(adjudicante(A,B,C,D)),
        nao(excecaoIncerta(adjudicante(A,B,C,D))),
        nao(excecaoImprecisa(adjudicante(A,B,C,D))),
        nao(excecaoImprecisaIntervalo(adjudicante(A,B,C,D),_,_)),
        nao(excecaoInterdita(adjudicante(A,B,C,D))).
-adjudicataria(A,B,C,D) :-
        nao(adjudicataria(A,B,C,D)),
        nao(excecaoIncerta(adjudicataria(A,B,C,D))),
        nao(excecaoImprecisa(adjudicataria(A,B,C,D))),
        nao(excecaoImprecisaIntervalo(adjudicataria(A,B,C,D),_,_)),
        nao(excecaoInterdita(adjudicataria(A,B,C,D))).
-contrato(A,B,C,D,E,F,G,H,I,J,K,L) :-
        nao(contrato(A,B,C,D,E,F,G,H,I,J,K,L)),
        nao(excecaoIncerta(contrato(A,B,C,D,E,F,G,H,I,J,K,L))),
        nao(excecaoImprecisa(contrato(A,B,C,D,E,F,G,H,I,J,K,L))),
        nao(excecaoImprecisa(contrato(A,B,C,D,E,F,G,H,I,J,K,L)), _, _),
        nao(excecaoInterdita(contrato(A,B,C,D,E,F,G,H,I,J,K,L))).

yes
| ?- listing(adjudicante).
adjudicante(1, municipio_altodebasto, 705330336, portugal_braga_altodebasto).
adjudicante(2, municipio_guarda, 700000000, portugal_guarda).
adjudicante(3, municipio_vilaverde, 700000001, portugal_braga_vilaverde).
adjudicante(4, municipio_viseu, xpto1, portugal_viseu).
adjudicante(7, municipio_lisboa, interdito, portugal_lisboa).
adjudicante(10, municipio_vilareal, xpto1, portugal_vilareal).

yes
```

Figura 33: Atualização(incerta) do adjudicante com ID 10 que que se encontrava como informação negativa

3. Atualização de conhecimento impreciso para incerto

Inicialmente, verificamos se existe informação imprecisa sobre o que queremos adicionar na base de conhecimento, sendo esta depois eliminada e é inserida a exceção com a informação incerta especificada, bem como o predicado *adjudicante* com o *xpto1* associado.

```
atualizacaoIncerto(adjudicante(Id, Nome, NIF, Morada), nif) :-
        nao(adjudicante(Id, _, _, _)),
        nao(clause(-adjudicante(Id, Nome, NIF, Morada), true)),
        si(excecaoImprecisa(adjudicante(Id, Nome, NIF, Morada)), verdadeiro),
        solucoes((excecaoImprecisa(adjudicante(_,_,_,_))),
                excecaoImprecisa(adjudicante(Id, Nome, _, Morada)), L),
        remocaoLista(L),
        remocao((excecaoImprecisa(adjudicante(Id,_,N,_)))),
        insercao((excecaoIncerta(adjudicante(I, NO, N, M)) :-
                adjudicante(I,NO, xpto1, M))),
        insercao(adjudicante(Id, Nome, xpto1, Morada)).
atualizacaoIncerto(adjudicante(Id, Nome, NIF, Morada), nif, NIF, SUP) :-
        nao(adjudicante(Id, _, _, _)),
        nao(clause(-adjudicante(Id, Nome, NIF, Morada), true)),
        nao(clause(excecaoImprecisa(adjudicante(Id,_,_,_)), R)),
        clause(excecaoImprecisaIntervalo(adjudicante(Id,_,_,_),_,_), R),
        remocao((excecaoImprecisaIntervalo(adjudicante(Id,_,N,_), X, Y) :-
                N >= X, N =<Y )),
        insercao((excecaoIncerta(adjudicante(I, NO, N, M)) :-
                adjudicante(I,NO, xpto1, M))),
        insercao(adjudicante(Id, Nome, xpto1, Morada)).
```

De seguida, apresentamos um exemplo dos resultados obtidos a partir desta cláusula.

```
| ?- atualizacaoIncerto(adjudicante(5, municipio_braga, 700000010, portugal_braga), nif).
yes
| ?- listing(excecaoImprecisa).
excecaoImprecisa(adjudicataria(5,peugaspinto,711111131,portugal)).
excecaoImprecisa(adjudicataria(5,peugastinto,711111131,portugal)).
excecaoImprecisa(contrato(1,2,5,contrato_de_aquisicao,ajuste_direto,assessoriajuridica,5000,280,lisboa,15,5,2005)).
excecaoImprecisa(contrato(1,2,5,contrato_de_aquisicao,ajuste_direto,assessoriajuridica,5000,170,lisboa,15,5,2005)).

yes
| ?- listing(adjudicante).
adjudicante(1, municipio_altodebasto, 705330336, portugal_braga_altodebasto).
adjudicante(2, municipio_guarda, 700000000, portugal_guarda).
adjudicante(3, municipio_vilaverde, 700000001, portugal_braga_vilaverde).
adjudicante(4, municipio_viseu, xpto1, portugal_viseu).
adjudicante(7, municipio_lisboa, interdito, portugal_lisboa).
adjudicante(5, municipio_braga, xpto1, portugal_braga).

yes
```

Figura 34: Atualização(incerta) do adjudicante com ID 5 que que se encontrava como informação imprecisa

4. Inserção de conhecimento incerto novo

Quando não há referência da informação que pretendemos inserir, é possível registar conhecimento incerto. Após verificada a inexistência da informação que pretendemos inserir, procedemos à inserção da excecão, bem como o predicado *adjudicante* com o *xpto1* associado.

```
atualizacaoIncerto(adjudicante(Id, Nome, NIF, Morada), nif) :-
        nao(adjudicante(Id, _, _, _)),
        nao(clause(-adjudicante(Id, Nome, NIF, Morada), true)),
        nao(clause(excecaoImprecisa(adjudicataria(Id,_,_,_)), R)),
        nao(clause(excecaoImprecisaIntervalo(adjudicataria(Id,_,_,_),_,_), R)),
        insercao((excecaoIncerta(adjudicante(I, NO, N, M)) :-
                adjudicante(I,NO, xpto1, M))),
        insercao(adjudicante(Id, Nome, xpto1, Morada)).
```

De seguida, apresentamos um exemplo dos resultados obtidos a partir desta cláusula.

```
| ?- atualizacaoIncerto(adjudicante(10, municipio_vilareal, 788888888, portugal_vilareal), nif).
yes
| ?- listing(adjudicante).
adjudicante(1, municipio_altodebasto, 705330336, portugal_braga_altodebasto).
adjudicante(2, municipio_guarda, 700000000, portugal_guarda).
adjudicante(3, municipio_vilaverde, 700000001, portugal_braga_vilaverde).
adjudicante(4, municipio_viseu, xpto1, portugal_viseu).
adjudicante(7, municipio_lisboa, interdito, portugal_lisboa).
adjudicante(10, municipio_vilareal, xpto1, portugal_vilareal).

yes
```

Figura 35: Inserção da informação incerta em relação ao adjudicante com ID 10

3.8.3 Inserção de conhecimento imperfeito impreciso

Para o conhecimento imperfeito impreciso, vamos apenas desenvolver apenas as cláusulas do meta-predicado *atualizarImpreciso* de forma a este inserir informação imprecisa relativamente ao NIF dos adjudicantes. Para esta componente tivemos que ter atenção dois casos, se pretendemos inserir imprecisão com um intervalo de valores ou com valores pontuais. Além disso, iremos mostrar alguns exemplos dos resultados obtidos através das cláusulas desenvolvidas.

1. Atualização de conhecimento positivo para impreciso

Inicialmente, verificamos se existe informação positiva sobre o que queremos adicionar na base de conhecimento, sendo esta depois eliminada e são inseridas a exceções com a informação imprecisa especificada.

```
atualizacaoImpreciso(adjudicante(Id, Nome, NIF, Morada), nif) :-
        si(excecaoIncerta(adjudicante(Id, Nome, NIF, Morada)), desconhecido),
        si(adjudicante(Id, _, _, _),verdadeiro),
        nao(clause(-adjudicante(Id, Nome, NIF, Morada), true)),
        solucoes((adjudicante(Id, NO, N, M)), adjudicante(Id, _, _, _), L),
        comprimento(L,R), R == 1,
        remocaoLista(L),
        insercao((excecaoImprecisa(adjudicante(Id, Nome, NIF, Morada)))).
atualizacaoImpreciso(adjudicante(Id, Nome, NIF, Morada), nif, INF, SUP) :-
        si(adjudicante(Id, _, _, _),verdadeiro),
        nao(clause(-adjudicante(Id, Nome, NIF, Morada), true)),
        solucoes((adjudicante(Id, NO, N, M)), adjudicante(Id, _, _, _), L),
        comprimento(L,R), R == 1,
        remocaoLista(L),
        insercao((excecaoImprecisaIntervalo(adjudicante(Id, Nome, NI,
                Morada), INF, SUP) :- NI >= INF, NI =< SUP)).
```

De seguida, apresentamos um exemplo dos resultados obtidos a partir de uma cláusula.

```
| ?- atualizacaoImpreciso(adjudicante(1, municipio_altodebasto, 70533036, portugal_braga_altodebasto),nif, 700000111, 700000999).
yes
| ?- listing(adjudicante).
adjudicante(2, municipio_guarda, 700000000, portugal_guarda).
adjudicante(3, municipio_vilaverde, 700000001, portugal_braga_vilaverde).
adjudicante(4, municipio_viseu, xpto1, portugal_viseu).
adjudicante(7, municipio_lisboa, interdito, portugal_lisboa).

yes
| ?- listing(excecaoImprecisaIntervalo).
excecaoImprecisaIntervalo(adjudicante(6,municipio_porto,A,portugal_porto), 700000030, 700000050) :-
        A>=700000030,
        A=<700000050.
excecaoImprecisaIntervalo(contrato(3,2,7,urgente,concurso_publico,assessoriajuridica,200000,A,coimbra,15,5,2018), 100, 400) :-
        A>=100,
        A=<400.
excecaoImprecisaIntervalo(adjudicante(1,municipio_altodebasto,A,portugal_braga_altodebasto), 700000111, 700000999) :-
        A>=700000111,
        A=<700000999.

yes
```

Figura 36: Atualização(imprecisa) do adjudicante com ID 1 que se encontrava como informação positiva

2. Atualização de conhecimento negativo para impreciso

Inicialmente, verificamos se existe informação negativa sobre o que queremos adicionar na base de conhecimento, sendo esta depois eliminada e são inseridas as exceções com a informação imprecisa especificada.

```
atualizacaoImpreciso(adjudicante(Id, Nome, NIF, Morada), nif) :-
        clause(-adjudicante(Id, _, _, _),true),
        nao(adjudicante(Id, _, _, _)),
        solucoes((-adjudicante(Id, NO, N, M)), (-adjudicante(Id, _, _, _)), L),
        comprimento(L,R), R >= 1,
        remocaoLista(L),
        insercao((excecaoImprecisa(adjudicante(Id, Nome, NIF, Morada)))).
atualizacaoImpreciso(adjudicante(Id, Nome, NIF, Morada), nif, INF, SUP)  :-
        clause(-adjudicante(Id, _, _, _),true),
        nao(adjudicante(Id, _, _, _)),
        solucoes((-adjudicante(Id, NO, N, M)), (-adjudicante(Id, _, _, _)), L),
        comprimento(L,R), R >= 1,
        remocaoLista(L),
        insercao((excecaoImprecisaIntervalo(adjudicante(Id, Nome, NI, Morada),
                INF, SUP) :- NI >= INF, NI =< SUP)).
```

De seguida, apresentamos um exemplo dos resultados obtidos a partir de uma das cláusulas.

```
| ?- evolucao(-adjudicante(10, municipio_faro, 787654321, portugal_faro)).
yes
| ?- atualizacaoImpreciso(adjudicante(10, municipio_faro, 787654321,portugal_faro ),nif, 712000000, 780777777).
yes
| ?- listing(excecaoImprecisaIntervalo).
excecaoImprecisaIntervalo(adjudicante(6,municipio_porto,A,portugal_porto), 700000030, 700000050) :-
        A>=700000030,
        A=<700000050.
excecaoImprecisaIntervalo(contrato(3,2,7,urgente,concurso_publico,assessoriajuridica,200000,A,coimbra,15,5,2018), 100, 400) :-
        A>=100,
        A=<400.
excecaoImprecisaIntervalo(adjudicante(10,municipio_faro,A,portugal_faro), 712000000, 780777777) :-
        A>=712000000,
        A=<780777777.

yes
| ?- listing(-).
-adjudicante(A,B,C,D) :-
        nao(adjudicante(A,B,C,D)),
        nao(excecaoIncerta(adjudicante(A,B,C,D))),
        nao(excecaoImprecisa(adjudicante(A,B,C,D))),
        nao(excecaoImprecisaIntervalo(adjudicante(A,B,C,D),_,_)),
        nao(excecaoInterdita(adjudicante(A,B,C,D))).
-adjudicataria(A,B,C,D) :-
        nao(adjudicataria(A,B,C,D)),
        nao(excecaoIncerta(adjudicataria(A,B,C,D))),
        nao(excecaoImprecisa(adjudicataria(A,B,C,D))),
        nao(excecaoImprecisaIntervalo(adjudicataria(A,B,C,D),_,_)),
        nao(excecaoInterdita(adjudicataria(A,B,C,D))).
-contrato(A,B,C,D,E,F,G,H,I,J,K,L) :-
        nao(contrato(A,B,C,D,E,F,G,H,I,J,K,L)),
        nao(excecaoIncerta(contrato(A,B,C,D,E,F,G,H,I,J,K,L))),
        nao(excecaoImprecisa(contrato(A,B,C,D,E,F,G,H,I,J,K,L))),
        nao(excecaoImprecisa(contrato(A,B,C,D,E,F,G,H,I,J,K,L)), _, _),
        nao(excecaoInterdita(contrato(A,B,C,D,E,F,G,H,I,J,K,L))).

yes
```

Figura 37: Atualização(imprecisa) do adjudicante com ID 10 que que se encontrava como informação negativa

3. Atualização de conhecimento incerto para impreciso

Inicialmente, verificamos se existe informação imprecisa sobre o que queremos adicionar na base de conhecimento, sendo esta depois eliminada e é inserida a exceção com a informação incerta especificada, bem como o predicado *adjudicante* com o *xpto1* associado.

```
atualizacaoImpreciso(adjudicante(Id, Nome, NIF, Morada), nif) :-
        nao(clause(-adjudicante(Id, Nome, NIF, Morada), true)),
        clause(excecaoIncerta(adjudicante(Id, _, _, _)), Q),
        si(adjudicante(Id,_, xpto1,_), verdadeiro),
        remocao(adjudicante(Id, _, xpto1, _)),
        remocao((excecaoIncerta(adjudicante(I, N, NI, M)) :-
                adjudicante(I, N, xpto1, M))),
        insercao((excecaoImprecisa(adjudicante(Id, Nome, NIF, Morada)))).
atualizacaoImpreciso(adjudicante(Id, Nome, NIF, Morada), nif, INF, SUP) :-
        nao(adjudicante(Id, _, _, _)),
        nao(clause(-adjudicante(Id, Nome, NIF, Morada), true)),
        clause(excecaoIncerta(adjudicante(Id, _, _, _)), Q),
        si(adjudicante(Id,_, xpto1,_), verdadeiro),
        remocao(adjudicante(Id, _, xpto1, _)),
        remocao((excecaoIncerta(adjudicante(I, N, NI, M)) :-
                adjudicante(I, N, xpto1, M))),
        insercao((excecaoImprecisaIntervalo(adjudicante(Id, Nome, NI, Morada),
                INF, SUP) :- NI >= INF, NI =< SUP)).
```

De seguida, apresentamos um exemplo dos resultados obtidos a partir desta cláusula.

```
| ?- atualizacaoIncerto(adjudicante(5, municipio_braga, 700000010, portugal_braga), nif).
yes
| ?- listing(excecaoImprecisa).
excecaoImprecisa(adjudicataria(5,peugaspinto,711111131,portugal)).
excecaoImprecisa(adjudicataria(5,peugastinto,711111131,portugal)).
excecaoImprecisa(contrato(1,2,5,contrato_de_aquisicao,ajuste_direto,assessoriajuridica,5000,280,lisboa,15,5,2005)).
excecaoImprecisa(contrato(1,2,5,contrato_de_aquisicao,ajuste_direto,assessoriajuridica,5000,170,lisboa,15,5,2005)).

yes
| ?- listing(adjudicante).
adjudicante(1, municipio_altodebasto, 705330336, portugal_braga_altodebasto).
adjudicante(2, municipio_guarda, 700000000, portugal_guarda).
adjudicante(3, municipio_vilaverde, 700000001, portugal_braga_vilaverde).
adjudicante(4, municipio_viseu, xpto1, portugal_viseu).
adjudicante(7, municipio_lisboa, interdito, portugal_lisboa).
adjudicante(5, municipio_braga, xpto1, portugal_braga).

yes
```

Figura 38: Atualização(imprecisa) do adjudicante com ID 5 que que se encontrava como informação incerta

4. Inserção de conhecimento impreciso novo

Quando não há referência da informação que pretendemos inserir, é possível registar conhecimento impreciso. Após verificada a inexistência da informação que pretendemos inserir, procedemos à inserção das exceções.

```
atualizacaoImpreciso(adjudicante(Id, Nome, NIF, Morada), nif) :-
        nao(adjudicante(Id, Nome, NIF, Morada)),
        nao(clause(-adjudicante(Id, Nome, NIF, Morada), true)),
        insercao((excecaoImprecisa(adjudicante(Id, Nome, NIF, Morada)))).
atualizacaoImpreciso(adjudicante(Id, Nome, NIF, Morada), nif, INF, SUP) :-
        nao(adjudicante(Id, Nome, NIF, Morada)),
        nao(clause(-adjudicante(Id, Nome, NIF, Morada), true)),
        nao(predImperfeito(adjudicante(Id, Nome, NIF, Morada), X)),
        insercao((excecaoImprecisa(adjudicante(Id, Nome, N, Morada)) :-
                N >= INF, N =< SUP)).
```

De seguida, apresentamos um exemplo dos resultados obtidos a partir desta cláusula.

```
| ?- atualizacaoImpreciso(adjudicante(9, municipio_viana, 76533036, portugal_viana),nif).
yes
| ?- listing(excecaoImprecisa).
excecaoImprecisa(adjudicante(5,municipio_braga,700000010,portugal_braga)).
excecaoImprecisa(adjudicante(5,municipio_braga,700000020,portugal_braga)).
excecaoImprecisa(adjudicataria(5,peugaspinto,711111131,portugal)).
excecaoImprecisa(adjudicataria(5,peugastinto,711111131,portugal)).
excecaoImprecisa(contrato(1,2,5,contrato_de_aquisicao,ajuste_direto,assessoriajuridica,5000,280,lisboa,15,5,2005)).
excecaoImprecisa(contrato(1,2,5,contrato_de_aquisicao,ajuste_direto,assessoriajuridica,5000,170,lisboa,15,5,2005)).
excecaoImprecisa(adjudicante(9,municipio_viana,76533036,portugal_viana)).

yes
```

Figura 39: Inserção da informação imprecisa em relação ao adjudicante com ID 10

3.8.4 Inserção de conhecimento imperfeito interdito

Na inserção de conhecimento interdito foram tidos em o conhecimento inexistente, conhecimento positivo, conhecimento negativo e conhecimento impreciso ou incerto. Desta forma, podemos transformar qualquer tipo de conhecimento existente ou não na base de conhecimento em conhecimento interdito, desde que este não esteja já reconhecido como interdito.

Ao registar conhecimento interdito na base de conhecimento, tal como nos casos anteriores, e necesário indicar qual o paâmetro imperfeito, para que se possam gerar as devidas exceções. Como este meta-predicado *atualizaInterdito* insere conhecimento na base de conhecimento, para qualquer uma das cláusulas utilizadas, devem ser verificados os invariantes de inserção.

Como anteriormente, consideramos apenas o NIF dos adjudicantes para a inserção de conhecimento incerto.

1. Atualização de conhecimento positivo para interdito

Inicialmente, verificamos se existe informação positiva sobre o que queremos adicionar na base de conhecimento. Como este pode estar associado a uma exceção de conhecimento de incerto, é também necessário verificarmos. Depois disto basta eliminar da base de conhecimento o conhecimento desatualizado, e inserir os factos e cláusulas necessárias à existência de conhecimento interdito.

```
atualizacaoInterdito(adjudicante(Id, Nome, NIF, Morada), nif) :-
        solucoes(Inv, +adjudicante(Id,Nome,NIF,Morada) :: Inv, Sol),
        nao(nuloInterdito(NIF)),
        si(excecaoIncerta(adjudicante(Id, Nome, NIF, Morada)), desconhecido),
        si(adjudicante(Id, _, _, _),verdadeiro),
        nao(clause(-adjudicante(Id, _, _, _), true)),
        nao(clause(excecaoImprecisa(adjudicante(Id,_,_,_)), R)),
        nao(clause(excecaoImprecisaIntervalo(adjudicante(Id,_,_,_),_,_), R)),
        solucoes((adjudicante(Id, NO, N, M)), adjudicante(Id, _, _, _), L),
        comprimento(L,R), R == 1,
        remocaoLista(L),
        insercao((adjudicante(Id, Nome, interdito, Morada))),
        insercao((excecaoInterdita(adjudicante(A,B,C,D)) :-
                adjudicante(A,B,interdito,D))),
        insercao((nuloInterdito(interdito))),
        insercao((+adjudicante(I, N, NI, M) :: (solucoes(Nif,
                (adjudicante(Id, Nome, Nif, Morada), nao(nuloInterdito(Nif))),S),
                comprimento(S,N), N==0))),
        teste(Sol).
```

De seguida, apresentamos um exemplo dos resultados obtidos a partir da cláusula.

```
| ?- atualizacaoInterdito(adjudicante(2, municipio_guarda, 700000000, portugal_guarda), nif).
yes
| ?- listing(adjudicante).
adjudicante(1, municipio_altodebasto, 705330336, portugal_braga_altodebasto).
adjudicante(3, municipio_vilaverde, 700000001, portugal_braga_vilaverde).
adjudicante(4, municipio_viseu, xpto1, portugal_viseu).
adjudicante(7, municipio_lisboa, interdito, portugal_lisboa).
adjudicante(2, municipio_guarda, interdito, portugal_guarda).

yes
```

Figura 40: Atualização(interdita) do adjudicante com ID 2 que se encontrava como informação positiva

2. Atualização de conhecimento negativo para interdito

Assim como acontecia com o conhecimento positivo, neste caso também devemos sverificar se o conhecimento existente na base de conhecimento é realmente falso. Caso aconteça, o conhecimento desatualizado deve ser eliminado da base de conhecimento e inserir os factos e cláusulas necessárias à existência de conhecimento interdito.

```
atualizacaoInterdito(adjudicante(Id, Nome, NIF, Morada), nif) :-
        solucoes(Inv, +adjudicante(Id,Nome,NIF,Morada) :: Inv, Sol),
        nao(nuloInterdito(NIF)),
        clause(-adjudicante(Id, _, _, _), true),
        nao(clause(excecaoImprecisa(adjudicante(Id,_,_,_)), R)),
        nao(clause(excecaoImprecisaIntervalo(adjudicante(Id,_,_,_),_,_), R)),
        solucoes((-adjudicante(Id, NO, N, M)), (-adjudicante(Id, _, _, _)), L),
        comprimento(L,R), R >= 1,
        remocaoLista(L),
        insercao((adjudicante(Id, Nome, interdito, Morada))),
        insercao((excecaoInterdita(adjudicante(A,B,C,D)) :-
                adjudicante(A,B,interdito,D))),
        insercao((nuloInterdito(interdito))),
        insercao((+adjudicante(I, N, NI, M) :: (solucoes(Nif,
                (adjudicante(Id, Nome, Nif, Morada), nao(nuloInterdito(Nif))),S),
                comprimento(S,N), N==0))),
        teste(Sol).
```

De seguida, apresentamos um exemplo dos resultados obtidos a partir da cláusula.

```
| ?- evolucao(-adjudicante(2, municipio_guarda, 700000000, portugal_guarda)).
yes
| ?- atualizacaoInterdito(adjudicante(2, municipio_guarda, 700000000, portugal_guarda), nif).
yes
| ?- listing(-).
-adjudicante(A,B,C,D) :-
        nao(adjudicante(A,B,C,D)),
        nao(excecaoIncerta(adjudicante(A,B,C,D))),
        nao(excecaoImprecisa(adjudicante(A,B,C,D))),
        nao(excecaoImprecisaIntervalo(adjudicante(A,B,C,D),_,_)),
        nao(excecaoInterdita(adjudicante(A,B,C,D))).
-adjudicataria(A,B,C,D) :-
        nao(adjudicataria(A,B,C,D)),
        nao(excecaoIncerta(adjudicataria(A,B,C,D))),
        nao(excecaoImprecisa(adjudicataria(A,B,C,D))),
        nao(excecaoImprecisaIntervalo(adjudicataria(A,B,C,D),_,_)),
        nao(excecaoInterdita(adjudicataria(A,B,C,D))).
-contrato(A,B,C,D,E,F,G,H,I,J,K,L) :-
        nao(contrato(A,B,C,D,E,F,G,H,I,J,K,L)),
        nao(excecaoIncerta(contrato(A,B,C,D,E,F,G,H,I,J,K,L))),
        nao(excecaoImprecisa(contrato(A,B,C,D,E,F,G,H,I,J,K,L))),
        nao(excecaoImprecisa(contrato(A,B,C,D,E,F,G,H,I,J,K,L)), _, _),
        nao(excecaoInterdita(contrato(A,B,C,D,E,F,G,H,I,J,K,L))).

yes
| ?- listing(adjudicante).
adjudicante(1, municipio_altodebasto, 705330336, portugal_braga_altodebasto).
adjudicante(3, municipio_vilaverde, 700000001, portugal_braga_vilaverde).
adjudicante(4, municipio_viseu, xpto1, portugal_viseu).
adjudicante(7, municipio_lisboa, interdito, portugal_lisboa).
adjudicante(2, municipio_guarda, interdito, portugal_guarda).

yes
```

Figura 41: Atualização(interdita) do adjudicante com ID 2 que que se encontrava como informação negativa

3. Atualização de conhecimento incerto ou impreciso para interdito

O conhecimento impreciso ou incerto pode ser também alterado para conhecimento interdito, sendo que neste caso é necessário assegurar que o conhecimento impreciso ou incerto atualizado seja removido.

Para isto, são eliminadas as exceções relativas aos tipos de conhecimento anteriores e inseridas os factos e cláusulas necessárias à existência de conhecimento interdito.

```
atualizacaoInterdito(adjudicante(Id, Nome, NIF, Morada), nif) :-
        solucoes(Inv, +adjudicante(Id,Nome,NIF,Morada) :: Inv, Sol),
        nao(nuloInterdito(NIF)),
        nao(clause(-adjudicante(Id, Nome, NIF, Morada), true)),
        clause(excecaoIncerta(adjudicante(Id, _, _, _)), Q),
        si(adjudicante(Id,_, xpto1,_), verdadeiro),
        remocao(adjudicante(Id, _, xpto1, _)),
        remocao((excecaoIncerta(adjudicante(I, N, NI, M)) :-
                adjudicante(I, N, xpto1, M))),
        insercao((adjudicante(Id, Nome, interdito, Morada))),
        insercao((excecaoInterdita(adjudicante(A,B,C,D)) :-
                adjudicante(A,B,interdito,D))),
        insercao((nuloInterdito(interdito))),
        insercao((+adjudicante(I, N, NI, M) :: (solucoes(Nif,
                (adjudicante(Id, Nome, Nif, Morada), nao(nuloInterdito(Nif))),S),
                comprimento(S,N), N==0))),
        teste(Sol);
        solucoes(Inv, +adjudicante(Id,Nome,NIF,Morada) :: Inv, Sol),
        nao(nuloInterdito(NIF)),
        nao(adjudicante(Id, _, _, _)),
        nao(clause(-adjudicante(Id, Nome, NIF, Morada), true)),
        si(excecaoImprecisa(adjudicante(Id, Nome, NIF, Morada)), verdadeiro),
        solucoes((excecaoImprecisa(adjudicante(_,_,_,_))),
                excecaoImprecisa(adjudicante(Id, Nome, _, Morada)), L),
        remocaoLista(L),
        remocao((excecaoImprecisa(adjudicante(Id,_,N,_)))),
        insercao((adjudicante(Id, Nome, interdito, Morada))),
        insercao((excecaoInterdita(adjudicante(A,B,C,D)) :-
                adjudicante(A,B,interdito,D))),
        insercao((nuloInterdito(interdito))),
        insercao((+adjudicante(I, N, NI, M) :: (solucoes(Nif,
                (adjudicante(Id, Nome, Nif, Morada), nao(nuloInterdito(Nif))),S),
                comprimento(S,N), N==0))),
        teste(Sol);
        solucoes(Inv, +adjudicante(Id,Nome,NIF,Morada) :: Inv, Sol),
        nao(nuloInterdito(NIF)),
        nao(adjudicante(Id, _, _, _)),
        nao(clause(-adjudicante(Id, Nome, NIF, Morada), true)),
        nao(clause(excecaoImprecisa(adjudicante(Id,_,_,_)), R)),
        clause(excecaoImprecisaIntervalo(adjudicante(Id,_,_,_),_,_), R),
        remocao((excecaoImprecisaIntervalo(adjudicante(Id,_,N,_), X, Y) :-
                N >= X, N =<Y )),
        insercao((adjudicante(Id, Nome, interdito, Morada))),
        insercao((excecaoInterdita(adjudicante(A,B,C,D)) :-
                adjudicante(A,B,interdito,D))),
        insercao((nuloInterdito(interdito))),
        insercao((+adjudicante(I, N, NI, M) :: (solucoes(Nif,
                (adjudicante(Id, Nome, Nif, Morada), nao(nuloInterdito(Nif))),S),
                comprimento(S,N), N==0))),
        teste(Sol).
```

De seguida, apresentamos um exemplo dos resultados obtidos a partir desta cláusula.

```
| ?- atualizacaoInterdito(adjudicante(4, municipio_viseu, xpto1, portugal_viseu), nif).
yes
| ?- listing(excecaoIncerta).
excecaoIncerta(adjudicataria(A,_,B,C)) :-
        adjudicataria(A, xpto2, B, C).
excecaoIncerta(contrato(A,B,C,D,E,F,G,_,H,I,J,K)) :-
        contrato(A, B, C, D, E, F, G, xpto3, H, I, J, K).

yes
| ?- listing(adjudicante).
adjudicante(1, municipio_altodebasto, 705330336, portugal_braga_altodebasto).
adjudicante(2, municipio_guarda, 700000000, portugal_guarda).
adjudicante(3, municipio_vilaverde, 700000001, portugal_braga_vilaverde).
adjudicante(7, municipio_lisboa, interdito, portugal_lisboa).
adjudicante(4, municipio_viseu, interdito, portugal_viseu).

yes
```

Figura 42: Atualização(interdita) do adjudicante com ID 4 que que se encontrava como informação incerta

4. Inserção de conhecimento interdito novo

Ao inserir conhecimento interdito a partir do qual nãao existe qualquer informação, e necesario verificar se esse conhecimento e, de facto, inexistente na base de conhecimento. Depois de se assegurar isso, são inseridas na base de conhecimento os factos e cláusulas necessários para que esse conhecimento seja tido como interdito.

```
atualizacaoInterdito(adjudicante(Id, Nome, NIF, Morada), nif) :-
        solucoes(Inv, +adjudicante(Id,Nome,NIF,Morada) :: Inv, Sol),
        nao(nuloInterdito(NIF)),
        nao(adjudicante(Id, _, _, _)),
        nao(clause(-adjudicante(Id, Nome, NIF, Morada), true)),
        nao(clause(excecaoImprecisa(adjudicante(Id,_,_,_)), R)),
        nao(clause(excecaoImprecisaIntervalo(adjudicante(Id,_,_,_),_,_), R)),
        insercao((adjudicante(Id, Nome, interdito, Morada))),
        insercao((excecaoInterdita(adjudicante(A,B,C,D)) :-
                adjudicante(A,B,interdito,D))),
        insercao((nuloInterdito(interdito))),
        insercao((+adjudicante(I, N, NI, M) :: (solucoes(Nif,
                (adjudicante(Id, Nome, Nif, Morada), nao(nuloInterdito(Nif))),S),
                comprimento(S,N), N==0))),
        teste(Sol).
```

De seguida, apresentamos um exemplo dos resultados obtidos a partir desta cláusula.

```
| ?- atualizacaoInterdito(adjudicante(10, municipio_portalegre, 77777754, portugal_portalegre), nif).
yes
| ?- listing(adjudicante).
adjudicante(1, municipio_altodebasto, 705330336, portugal_braga_altodebasto).
adjudicante(2, municipio_guarda, 700000000, portugal_guarda).
adjudicante(3, municipio_vilaverde, 700000001, portugal_braga_vilaverde).
adjudicante(4, municipio_viseu, xpto1, portugal_viseu).
adjudicante(7, municipio_lisboa, interdito, portugal_lisboa).
adjudicante(10, municipio_portalegre, interdito, portugal_portalegre).

yes
```

Figura 43: Inserção da informação imprecisa em relação ao adjudicante com ID 10

3.9 Sistema de Inferência

Para o sistema de inferência foi criado o meta-predicado *si*, que infere a veracidade de uma questão. Apresentamos então abaixo o meta-predicado referido.

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

4 Conclusões e Sugestões

Chegando à fase final deste relatório, consideramos que a realização deste exercício foi bem sucedida, uma vez que acreditamos ter cumprido todos os requisios pedidos, sendo que tentámos também explorar ao máximo a criatividade e dar a maior complexidade ao sistema, estendo o conhecimento do mesmo.

A maior dificuldade foi mesmo a problemática da evolução do conhecimento, ou seja, eliminar o conhecimento descontextualizado com a realidade que visamos e, posteriormente, inserir o mais recente. E, apesar de termos considerado apenas o NIF dos adjudicantes para registo de conhecimento impergeito, o raciocínio a aplicar a *adjudicataria* e *contrato* era exatamente o mesmo.

Em suma, este exercício permitiu uma maior consolidação dos conceitos de conhecimento(perfeito e imperfeito), bem como a sua evolução e manipulação através do uso de invariantes e cláusulas apropriadas, tendo como auxílio a linguagem de programação em lógica PROLOG.