UNIVERSIDADE DO MINHO

Trabalho de Grupo – 2º Exercício

Licenciatura em Engenharia Informática

Sistemas de Representação de Conhecimento e Raciocínio

3º Ano, 2º Semestre

Ano letivo 2014/2015

67646 Bruno Barbosa

53690 João Monteiro

54811 Hélder Machado

67707 Tiago Cunha

Departamento de Informática

4710-057

Braga, Portugal

Março 2015

RESUMO

Neste relatório são apresentadas e descritas todos os procedimentos necessários à resolução do segundo exercício do trabalho prático.

As tarefas essenciais para esta primeira fase consistem em elaborar um sistema de representação de conhecimento e raciocínio com capacidade para especializar um universo de comércio de automóvel, num contexto como o que é sugerido no enunciado.

Para além das funcionalidades sugeridas, foram ainda introduzidas outras funcionalidades e características no sistema não só ao nível das capacidades de representação do conhecimento como também ao nível das faculdades de raciocínio.

CONTEÚDO

INTRODUÇÃO

Foi-nos proposto no âmbito da unidade curricular de Sistemas de Representação de Conhecimento e Raciocínio desenvolver um sistema de representação de conhecimento e raciocínio com capacidade para caracterizar um universo com o qual se pretende descrever o comércio automóvel.

Neste sentido, o sistema proposto é descrito por um cenário que caracteriza os seguintes dados:

- Matrícula
- Modelo
- Marca
- Cor
- Preço
- Ano
- Registo

O caso prático que apresentamos é capaz de demonstrar as funcionalidades subjacentes à programação em lógica estendida e à representação do conhecimento imperfeito, recorrendo à temática dos valores nulos. Ou seja, o caso em questão respeita as funcionalidades:

- Representação do conhecimento positivo e do conhecimento negativo.
- Representação de casos de conhecimento imperfeito, através de valores nulos de todos os tipos estudados.
- Manipulação de invariantes que restringem inserções e remoções de conhecimento.
- Criação de procedimentos capazes de tratar a problemática da evolução do conhecimento.
- Desenvolvimento do sistema de inferência capaz de implementar os mecanismos de raciocínio inerentes a estes sistemas.

PRELIMINARES

Um sistema computacional, seja ele qual for, precisa de armazenar e manipular informação. Quer os sistemas de Base de Dados quer os sistemas de representação de conhecimento, lidam com aspetos concretos do mundo real referentes a informação. Tanto um como outro, distinguem as funções de representação (explicação do esquema conceptual e dos dados) e de computação (elaboração de soluções a questões e manuseamento dos dados).

As linguagens responsáveis pela manipulação de informação num sistema de Base de Dados baseia-se nos seguintes pressupostos:

- Pressuposto do Mundo Fechado (PMF) – A informação inexistente na base de dados é considerada como sendo falsa.
- Pressuposto dos Nomes Únicos – Duas constantes diferentes definem, obrigatoriamente, duas entidades diferentes do universo.
- Pressuposto do Domínio Fechado – Não admite a existência de objetos no universo para além daqueles que são designados por constantes na base de dados.

Por sua vez, os princípios que sustentam um sistema de representação de conhecimento, resumem-se a:

- Pressuposto do Mundo Aberto (PMA) – Podem haver outros factos verdadeiros para além do que já pertencem à base de dados.
- Pressuposto dos Nomes Únicos – Duas constantes diferentes definem, obrigatoriamente, duas entidades diferentes do universo
- Pressuposto do Domínio Aberto – É possível que existam mais objetos do universo de discurso para além dos que já são vistos como constantes na base de conhecimento.

No entanto, estes pressupostos não permitem representar diretamente a informação. As linguagens de Programação em Lógica trazem mais poder na representação do conhecimento contudo, não admitem informação incompleta.

Com a introdução da informação incompleta, o sistema ganha a capacidade de representação e inferência para além daquilo que é verdade e falso, podendo representar assim o desconhecido.

Isto é, genericamente, a resposta a uma questão $q(X)$ é:

- Verdadeira se,

$$\exists X : q(X) \text{ é } verdade$$

- Falsa se,

$$\exists X : q(X) \text{ é } falso \equiv \neg q(X) \text{ é } verdade$$

- Desconhecida se não é verdadeira nem falsa, ou seja,

$$\nexists X : q(X) \text{ é } verdade \bigvee \neg q(X) \text{ é } verdade$$

Contudo, os problemas em programação em lógica podem ser abordados de uma forma muito mais natural se for declarada explicitamente informação falsa (negativa). Esta negação divide-se em dois tipos:

- Negação por falha, que é representada pelo termo *não*.
- Negação forte ou clássica, como maneira de reconhecer informação negativa, ou falsa, representada pela conectiva "¬".

A desigualdade entre o significado de $não\ P\ e\ \neg P$ é fundamental quando não se pretende (ou não se quer) considerar que a informação existente sobre P é completa. Ou seja, quando não se aplica o princípio do mundo fechado a P, não é possível inferir a sua falsidade a partir da ausência de informação sobre o mesmo. Não esquecendo também o facto de que a inclusão da negação explícita, combinada com a negação por falha, admite uma maior expressividade e clareza da própria linguagem.

Relativamente à identificação dos valores nulos, esta é uma estratégia utilizada na enumeração de casos, para os quais se quer fazer uma diferenciação entre as situações em que as respostas a questões são provadas como conhecidas, podendo ser verdadeiras ou falsas, ou desconhecidas.

DESCRIÇÃO DO TRABALHO E ANÁLISE DE RESULTADOS

Definição dos predicados

Os predicados que escolhemos para descrever e caracterizar o nosso universo de comércio automóvel são os que apresentamos na lista que se segue.

- Matrícula
- Modelo
- Marca
- Cor
- Preço
- Ano
- Registo

A matrícula é o nosso identificador único para qualquer automóvel. Ela também responsável por associar os restantes predicados que são responsáveis por caracterizar um determinado automóvel.

O predicado modelo surge relacionado com a matrícula e desta maneira conseguimos identificar um modelo de um automóvel através da sua matrícula.

Da mesma forma que o anterior, o predicado marca aparece relacionado com a matrícula correspondente.

O próximo predicado, que é o da cor, serve para caracterizar um automóvel tendo em conta a sua cor.

O predicado preço do automóvel é o único com três valores associados. Como habitual, a matrícula é um dos valores presentes nos predicados de modo a identificar o automóvel respetivo. Juntamente a estes, existe ainda uma descrição que serve para descrever o estado do automóvel.

O ano, tal como a palavra indica, coliga uma matrícula de um determinado automóvel a um ano.

Por último, temos o registo do automóvel, que para uma dada matrícula, diz-nos quem é o proprietário atual do mesmo.

Invariantes

Nesta secção apresentamos os invariantes que permitem correta inserção de conhecimento no sistema. Ora, começando pela matrícula, temos por exemplo, que não podem existir duas matrículas iguais, pelo que o seu invariante é o que mostramos a seguir. Note que este mesmo caso serve apenas como exemplo, pois existem outros predicados para os quais também foram precisos implementar os seus invariantes estruturais muito semelhantes a este.

$$+matricula(M) :: (solucoes(M, matricula(M), S), comprimento(S, N), N == 1).$$

No entanto, estes invariantes ainda não são suficientes para representar corretamente o conhecimento. Há assim, a necessidade de existirem outros tipos de invariantes para além daqueles que garantem a não inserção de conhecimento repetido. Por isso, retomando os predicados que já foram explicados na secção anterior, seguem-se os próximos invariantes. Considere apenas o invariante apresentado como exemplo geral para todos os outros casos.

$$+modelo(M, \_) :: \big(solucoes\big((M, -), modelo(M, -), S\big), comprimento(S, N), N == 1\big).$$

O invariante acima escrito, tem como funcionalidade assegurar que uma matrícula está associada a um e um só modelo. Identicamente a este invariante, existem outros que garantem a coerência e coesão do conhecimento. São eles, por exemplo.

- Uma matrícula está associada a uma e uma só marca.
- Uma matrícula está ligada a uma única cor.
- Uma matrícula está relacionada com apenas um preço.
- Uma matrícula tem um ano fixo de quando foi emitida.
- Uma matrícula tem única e exclusivamente um proprietário a ela associado.

Como suporte a estes invariantes, introduzimos mais um último tipo de invariante que restringe a inserção de conhecimento num caso de não haverem dados na base de conhecimento que o consigam relacionar. Do género, não é permitido adicionar um ano a uma matrícula sem que a matrícula exista. Estes invariantes seguem o seguinte formato.

$$+ano(M, \_) :: matricula(M).$$

Seguindo a mesma lógica de há bocado, foram feitos invariantes semelhantes a estes para os predicados que restam.

Conhecimento negativo

Como já foi referido nos Preliminares, o conhecimento negativo possibilita um ganho de a partir de informação a qual sabemos que é falsa. Por exemplo, no caso a seguir,

$$-matricula(M) : - nao\big(matricula(M)\big).$$

Estamos a dizer que uma matrícula não existe, se o seu predicado é falso. Note que através do predicado não, o resultado da questão é verdade. Assim sendo, foram definidos mais predicados de conhecimento negativo, contudo, com uma ligeira diferença em relação ao apresentado porque é imprescindível a verificação de exceções as quais iremos abordar mais à frente. Seguem-se então, os predicados de conhecimento negativo.

$$-marca(M, MA) : - nao(marca(M, MA)), nao(excepcao(marca(M, MA))).$$

$$-cor(M, C) : - \ nao\big(cor(M, C)\big), nao\left(excepcao\big(cor(M, C)\big)\right).$$

$$-modelo(M, MO) : - nao(modelo(M, MO)), nao(excepcao(modelo(M, MO))).$$

$$-preco(M, E, P) : - nao(preco(M, E, P)), nao(excepcao(preco(M, E, P))).$$

$$-ano(M, A) : - nao(ano(M, A)), nao(excepcao(ano(M, A))).$$

$$-registo(M, P) : - \ mao\big(registo(M, P)\big), nao\left(excepcao\big(registo(M, P)\big)\right).$$

Pelo que podemos observar por estes predicados, o que eles nos dizem é que não existe conhecimento sobre algo se não existe quaisquer informação sobre a sua existência e nem sequer uma exceção referente aos mesmos. Agora na próxima secção, iremos abordar as exceções para explicar em que contexto elas se inserem.

Exceções

As exceções surgem no âmbito de informação incompleta. Vamos supor que se sabe a matrícula de um automóvel mas não se conhece a sua cor. Nesta situação a cor do automóvel vai tomar o valor de *unkown* visto que é desconhecida. Veja o exemplo descrito a seguir.

$$excepcao\big(cor(M, -)\big) : - cor(M, unkwon).$$

Valores nulos

Os valores nulos por sua vez aparecem como uma estratégia para a enumeração de ocorrências onde se pretende distinguir as que são conhecidas (verdadeiras ou falsas) ou desconhecidas. Os valores nulos que escolhemos são os que apresentamos a seguir, cada um deles associado ao predicado que lhe corresponde.

$$nulo(modelo\_nulo).$$

$$nulo(marca\_nula).$$

$$nulo(descricao\_nula).$$

$$nulo(cor\_nula).$$

$$nulo(nenhum).$$

Meta-Predicados

Demo

Neste exercício o meta-predicado Demo sofreu algumas alterações relativamente ao exercício anterior. Antes o Demo permitia única e exclusivamente um termo, contudo neste segundo exercício é possível saber o resultado de vários termos em simultâneo. Denominamos como *demolista* e definimo-lo da seguinte maneira.

```
demolista([],verdadeiro).
demolista([Questao|ListaQuestoes], X ) :- Questao,
demolista(ListaQuestoes,X).
demolista([Questao|_],falso) :- -Questao.
demolista([Questao|_],falso) :- nao( Questao ), nao( -Questao ).
```

Inserir e remover conhecimento

O nosso meta-predicado evolução também sofreu alterações. Passou de um para dois, onde num caso temos o inserir conhecimento e noutro temos remover conhecimento. As suas definições são as que apresentamos de seguida.

Inserir conhecimento

```
inserirConhecimento(Termo) :-
        findall( Invariante, +Termo::Invariante, Lista),
        insercao(Termo),
        teste( Lista ).

insercao(Termo) :-
        assert(Termo) .
insercao(Termo) :-
        retract(Termo), !, fail .

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

Remover Conhecimento

```
removerConhecimento(Termo) :-
        findall( Invariante, -Termo::Invariante, Lista),
        teste( Lista ) ,
        remocao(Termo).

remocao(Termo) :-
        retract(Termo).
```

Não

O meta-predicado não permite verificar de um predicado é verdadeiro ou falso. A sua definição é bastante simples e apresentamo-la já de seguida.

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

CONCLUSÕES E SUGESTÕES

No final deste segundo exercício do trabalho de grupo proposto, consideramos que foram implementadas as funcionalidades fundamentais à resolução do mesmo.

Inicialmente surgiram dúvidas sobre a forma como o trabalho ia ser implementado, como seria o desenvolvimento usando JAVA recorrendo à biblioteca JASPER. Ultrapassado esse obstáculo, seguimos para a conceção da interface gráfica e à programação em lógica em PROLOG.

De seguida, iniciamos pelas definições dos predicados que caracterizam o universo de comércio automóvel, desde a matrícula, modelo, marca, cor, entre outros, e pela programação em PROLOG através das bases de conhecimento. O próximo passo foi garantir que o conhecimento era inserido de forma correta. Para tal, definimos os diversos invariantes estruturais e referenciais que garantiam essa propriedade. Uma vez definidos, o esqueleto já estava praticamente montado e portanto, passamos à definição das exceções do valores nulos que nos admitiram abordar casos particulares de informação incompleta. Antes de seguirmos para o JAVA tivemos ainda que redefinir os predicados Demo e Evolução de modo a que permitissem vários termos em simultâneo.

Não esquecendo a parte referente a JAVA, tivemos que desenhar a interface gráfica que ia suportar a nossa aplicação. Depois criamos uma classe responsável por conectar-se ao PROLOG através do JASPER, uma classe representadora dos automóveis e uma classe com o método *main*, responsável pela execução da aplicação.

Em suma, consideramos que o trabalho foi bem-sucedido e que a maioria dos objetivos do mesmo foram alcançados.

REFERÊNCIAS

[1] BRATKO, Ivan

"Prolog Programming for Artificial Intelligence, 4th Edition"

Addison-Wesley, University of Ljubljana, 24 de Agosto de 2011

[2] COELHO, Hélder

"A Inteligência Artificial em 25 Lições"

Fundação Calouste Gulbenkian, 1995

[3] ANALIDE, César, NEVES, José

"Representação de Informação Incompleta"

Universidade do Minho, Departamento de Informática

[4] ANALIDE, César, NEVES, José, NOVAIS, Paulo

"Representação de Conhecimento Imperfeito"

Universidade do Minho, Departamento de Informática, ISLab