Universidade do Minho
Escola de Engenharia
Mestrado Integrado em Engenharia Informática

Unidade Curricula de Sistemas de Representação de Conhecimento e Raciocínio

Ano Lectivo de 2018/2019

Programação em Lógica Estendida

E Conhecimento Imperfeito

Henrique Faria, João Marques, José Pereira, Ricardo Petronilho

Abril, 2019

SRCR

Data de Receção	
Responsável	
Avaliação	
Observações	

Programação em Lógica Estendida

E Conhecimento Imperfeito

Henrique Faria, João Marques, José Pereira,

Ricardo Petronilho

Abril, 2019

Resumo

O presente trabalho tem como principal objetivo aprimorar a utilização da extensão à programação em lógica, bem como a representação de conhecimento imperfeito. Deste modo, pretendemos descrever os esforços efetuados no sentido a atingir o objetivo proposto.

Neste relatório será apresentado o projeto, seguindo-se a apresentação das soluções elaboradas para cada passo.

Por último é apresentada uma reflexão crítica sobre o trabalho que visa explanar as dificuldades encontradas, bem como os resultados obtidos.

Conteúdo

Índice de figuras

Índice de tabelas

1. Introdução

No âmbito da Unidade curricular de Sistemas de Representação de Conhecimento e Raciocínio do Mestrado integrado em Engenharia Informática da Universidade do Minho, foi-nos proposta a elaboração de um exercício para o desenvolvimento de uma base de conhecimento para utentes, serviços e consultas.

O principal objetivo da realização deste exercício é a motivação sobre os alunos para a utilização da extensão à programação em lógica, para a representação de conhecimento imperfeito.

Para realização da mesma utilizou-se a linguagem de programação **PROLOG**, utilizada na disciplina, que tem como base a lógica matemática. Nesse sentido, permite expressar programas na forma lógica e usar processos de inferência para produzir resultados.

Inicialmente, começou-se por representar conhecimento positivo e negativo, devido à possibilidade de se ter respostas *verdadeiras*, *falsas* e *desconhecidas,* sendo que o conhecimento positivo já tinha sido realizado no primeiro exercício.

De seguida, implementou-se a noção de conhecimento imperfeito, devido à presença da resposta como o desconhecido, existindo três tipos diferentes deste tipo de conhecimento incerto, impreciso e interdito.

2. Preliminares

Para uma correta resolução do enunciado proposto são necessários os seguintes requerimentos:

- Conhecimento da linguagem PROLOG;
- Compreender os conceitos de predicado e axioma;
- Compreender corretamente o enunciado;
- Distinguir os conceitos de negação explicita, e por falha;
- Compreender o conceito de conhecimento imperfeito e os seus três tipos (incerto, impreciso, interdito);
- Compreender o conceito de valor nulo;

Para a obtenção destes conhecimentos as aulas teóricas e as práticas foram cruciais.

3. Descrição do trabalho e Análise de resultados

Nesta secção será explicado e exemplificado todo o processo de resolução do trabalho prático realizado. Desde a inserção de conhecimento à remoção da mesma e os predicados realizados sobre a base de conhecimento, bem como a existência de conhecimento imperfeito.

3.1. Conhecimento negativo e positivo

A criação de predicados foi realizada no primeiro exercício, dos quais o *utente*, *servico*, *consulta* e *medico*. Neste mesmo exercício representou-se o conhecimento positivo, ou seja, informação verdadeira, tal como se pode observar na figura abaixo.

```
%Extenção do predicado utente: IdUt,Nome,Idade,Cidade -> {V,F,D}
utente(1,joao,22,lisboa).
utente(2,carlos,15,beja).
utente(3,alfredo,45,braga).
utente(4,dario,72,lisboa).
utente(5,joaquim,25,braga).
utente(6,henrique,20,braga).
utente(7,ricardo,20,braga).
utente(8,bruno,44,aveiro).
utente(9,nuno,33,porto).
utente(10,pedro,70,coimbra).
```

Figura 1: Representação do conhecimento positivo do predicado utente.

Por outro lado, a representação de conhecimento negativo, apenas foi realizada no exercício dois, porque neste existem três possíveis respostas: *verdadeiro*, *falso* e *desconhecido*, e para tal necessita-se da representação do mesmo, para que se possa definir a diferença do que é verdadeiro, falso e desconhecido.

```
% Conhecimento Negativo
%Extenção do predicado utente: IdUt,Nome,Idade,Cidade -> {V,F,D}
-utente(11,nuno,33,porto).
-utente(12,pedro,70,coimbra).

%Extenção do predicado servico: IdServ,Descrição,Instituição,Cidade -> {V,F,D}
-servico(15,dermatologia,hospital_coimbra,coimbra).
-servico(16,rinoplastia, hospital_lisboa, lisboa).

%Extenção do predicado consulta: Id,Data,IdUt,IdServ,IdMedico, Custo -> {V,F,D}
-consulta(7,data(15,8,2019),11,15,6,45).
-consulta(8,data(1,12,2017),12,16,5,50).

%Extenção do predicado medico: Id,Nome, Idade, Especialidade, Instituição -> {V,F,D}
-medico(5,joana,50,radiografia,hospital_trofa_1).
-medico(6,miguel, 44,cirurgia,hospital_porto).
```

Figura 2: Representação do conhecimento negativo dos predicados utente, servico, consulta e medico.

3.2. Conhecimento Imperfeito

Na programação em lógica estendida a resposta a uma questão pode ser: **verdadeira, falsa ou desconhecida**.

O conhecimento desconhecido ou imperfeito consiste na ausência de informação em relação à questão colocada. Em caso de informação incompleta, isto é, desconhecida, existem 3 casos distintos: **conhecimento incerto, impreciso ou interdito.**

Desta forma foi criado o meta-predicado *"nulo"* identificando que um determinado elemento atómico consiste em conhecimento imperfeito. Por exemplo, "*nulo(cidade)"* pode indicar que é impossível conhecer a cidade associada a outro predicado.

Assim falta estabelecer a associação do meta-predicado *"nulo"* a outro predicado para que seja possível identificar conhecimento imperfeito em certos campos desse predicado, consequentemente foi desenvolvido o predicado "excecao" para todos os predicados na base de conhecimento: **utente, servico, consulta e medico**.

Por exemplo, dado o seguinte predicado:

excecao(utente(Id, Nome, Idade, Morada)) :- utente(Id, Nome, X, Morada),
nulo(X).

É feita a verificação se o utente em causa tem uma idade desconhecida.

Tal como referido anteriormente, foi desenvolvido o predicado "excecao" para todos os predicados ilustrados de seguida.

```
% utente: IdUt,Nome,Idade,Cidade
% possíveis valores desconhecidos : Idade,Cidade
excecao(utente(A,B,C,D)) :- utente(A,B,X,D),
                            nulo(X).

excecao(utente(A,B,C,D)) :- utente(A,B,C,X),
                            nulo(X).

excecao(utente(A,B,C,D)) :- utente(A,B,X,Y),
                            nulo(X),
                            nulo(Y).
```

Figura 3: Predicado excecao para os utentes.

```
% servico: IdServ,Descrição,Instituição,Cidade
% possíveis valores desconhecidos : Descrição,Instituição
excecao(servico(A,B,C,D)) :- servico(A,X,C,D),
                             nulo(X).

excecao(servico(A,B,C,D)) :- servico(A,B,X,D),
                             nulo(X).

excecao(servico(A,B,C,D)) :- servico(A,X,Y,D),
                             nulo(X),
                             nulo(Y).
```

Figura 4: Predicado excecao para os servicos.

```
% consulta: Id,Data,IdUt,IdServ,IdMedico,Custo
% possíveis valores desconhecidos: Data,IdUt,IdMedico
excecao(consulta(A,B,C,D,E,F)) :- consulta(A,X,C,D,E,F),
                                  nulo(X).

excecao(consulta(A,B,C,D,E,F)) :- consulta(A,B,X,D,E,F),
                                  nulo(X).

excecao(consulta(A,B,C,D,E,F)) :- consulta(A,B,C,D,X,F),
                                  nulo(X).
```

Figura 5: Predicado excecao para as consultas.

```
% medico: Id,Nome, Idade, Especialidade, Instituição
%possíveis valores desconhecidos: Idade,Especialidade
excecao(medico(A,B,C,D,E)) :- medico(A,B,X,D,E),
                              nulo(X).

excecao(medico(A,B,C,D,E)) :- medico(A,B,C,X,E),
                              nulo(X).

excecao(medico(A,B,C,D,E)) :- medico(A,B,X,Y,E),
                              nulo(X),
                              nulo(Y).
```

Figura 6: Predicado excecao para os médicos.

3.2.1. Conhecimento Imperfeito incerto

Este tipo de conhecimento é desconhecido e não necessariamente de um conjunto determinado de valores, isto é, consiste em conhecimento sobre predicados, nos quais nem todos os campos são conhecidos.

De seguida é ilustrado conhecimento incerto associado a um utente.

```
utente(15, anabela, xIdade, guimaraes).
-utente(15, anabela, 20, guimaraes).

nulo(xIdade).
```

Figura 7: Conhecimento incerto associado a um utente.

A figura exemplifica que existe um utente com identificador 15, cuja idade é desconhecida, mas garantidamente não tem 20 anos.

Outros exemplos de conhecimento incerto foram desenvolvidos no projeto, mas não estão explicitados neste relatório.

3.2.2. Conhecimento Imperfeito impreciso

A diferença entre o conhecimento imperfeito impreciso e o explicado anteriormente é que o valor nulo do tipo desconhecido é referente a um conjunto finito e determinado de valores,

assumindo-se que caso a questão seja referente aos valores desse mesmo conjunto a resposta deverá ser sempre desconhecido.

A implementação deste conhecimento é feita com adição das excessões desse mesmo conjunto, tal como se pode observar na figura abaixo.

O conhecimento imperfeito impreciso, caracteriza-se por ter dados diferentes para a mesma entidade, como por exemplo, o utente com *id* 20, pode ter o nome de “mauricio” ou “anacleto”.

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

% não se sabe se o utente de id 20 se chama mauricio ou anacleto
excecao(utente(20,mauricio,20,braga)).
excecao(utente(20,anacleto,20,braga)).

% não se sabe se o servico de id 25 é disponibilizado no hospital_trofa_1 ou no hospital_trofa_2
excecao(servico(25,tumografia,hospital_trofa_1,braga)).
excecao(servico(25,tumografia,hospital_trofa_2,porto)).

% não se sabe se a consulta de id 15 foi feita ao paciente 1 ou 2
excecao(consulta(15,data(1,1,2019),1,15,1,30)).
excecao(consulta(15,data(1,1,2019),2,15,1,30)).

% não se sabe se o medico de id 15 tem 21 ou 22 anos
excecao(medico(15,goncalo,21,fisioterapia,hospital_trofa_2)).
excecao(medico(15,goncalo,22,fisioterapia,hospital_trofa_2)).
```

Figura 8: Conhecimento Imperfeito impreciso dos predicados utente, servico, consulta e medico.

3.2.3. Conhecimento Imperfeito interdito

Ao contrário dos 2 tipos de conhecimento imperfeito referidos anteriormente, este último é mais inflexível, uma vez que representa não só conhecimento desconhecido, mas também impossível, consequentemente torna-se impossível de adicionar este tipo de conhecimento à base de conhecimento. Para tal foram criados os respetivos invariantes estruturais para concretizar este impedimento.

A seguinte figura ilustra este tipo de conhecimento.

```
servico(25, tumografia, xInstituicao, braga).

+servico(A,B,C,D)::(solucoes(servico(A,B,C,D),servico(A,B,xInstituicao,D),S),
                    comprimento(S,N),
                    N == 0).
```

Figura 9: Conhecimento Imperfeito interdito do predicado servico.

A figura demonstra que é impossível saber qual a instituição que prestou o serviço com identificador 25 uma vez que é impossível adicionar qualquer predicado “servico” que tenha sido realizado na “xInstituicao” devido ao invariante estrutural definido.

Novamente, outros exemplos de conhecimento interdito foram desenvolvidos no projeto, mas não estão explicitados neste relatório.

3.3. Invariantes

Os invariantes estruturais relacionados com a adição de conhecimento Interdito já se encontram analisados em cima juntamente com a respetiva demonstração de tentativa de adição desse conhecimento, que deverá ser impossível nesse caso.

Para a adição ou remoção de conhecimento positivo ou perfeito os invariantes realizados são os mesmos do primeiro exercício.

Neste exercício foram ainda criados invariantes para o meta-predicado “excecao” para não ser possível adicionar exceções repetidas e também não ser possível remover exceções.

A seguinte figura mostra esses dois invariantes estruturais.

```
% nao permitir a insercao de uma excecao repetida
+excecao(T) :: ( solucoes(excecao(T),excecao(T),S),
                 comprimento(S,N),
                 N == 1 ).

% nao permitir a remoção de uma excecao
-excecao(T) :: ( no ).
```

Figura 10: Invariantes estruturais para o meta-predicado excecao.

3.4. Evolução do Conhecimento

Um dos problemas que se tem nas bases de conhecimento é a evolução do conhecimento, que se deve à presença de conhecimento perfeito e imperfeito, sendo que as mesmas têm que estar preparadas para tal.

Neste trabalho deparamo-nos com a necessidade de distinguir conhecimentos imperfeitos. Para isso escolhemos o predicado *nulo* para representação de conhecimento que possa ser *desconhecido* ou *interdito* e o predicado *excecao* que marca conhecimento imperfeito.

O predicado *evolucao* utiliza três cláusulas para cada um dos predicados da base de conhecimento que possuímos.

Assim, conseguiu-se distinguir o conhecimento imperfeito que necessita do auxílio do predicado excecao, do que não precisa e do conhecimento perfeito.

No início começou-se por fazer uso do predicado *demo* para filtrar conhecimento perfeito de imperfeito, de seguida com o auxílio do predicado *solucoes* fica-se a saber o tipo de conhecimento imperfeito a ser adicionado.

No caso em que o conhecimento faça uso do predicado excecao o conhecimento perfeito é transformado em imperfeito e reinserido na base de conhecimento, juntamente com o novo predicado, caso contrário a lista criada, contendo todos os predicados referentes ao conhecimento explicitado é eliminada, sendo adicionado o novo predicado como conhecimento perfeito.

Nota: Conhecimentos interditos nunca são removidos da base de conhecimento (por causa dos invariantes).

De seguida apresenta-se a *evolucao* criada para os utentes:

```
%Extensao do predicado evolucao: Termo -> {V, F}
evolucao( utente(A, B, C, D) ) :- demo(utente(A, B, C, _), R),
                                  R = desconhecido,
                                  solucoes(utente(A, E, F, G), utente(A, E, F, G), L),
                                  comprimento(L, S),
                                  S == 0,
                                  registar( excecao(utente(A, B, C, D)) ).
```

Figura 11: Evolução do conhecimento para o predicado utente.

```
% atualização dos dados
evolucao( utente(A, B, C, D ) ) :-  demo(utente(A, B, C, D), R),
                                    R = desconhecido,
                                    solucoes(utente(A, H, I, J), utente(A, H, I, J), L),
                                    comprimento(L, S),
                                    S > 0,
                                    registar(utente(A, B, C, D)),
                                    removeBC(L).
```

Figura 12: Evolução do conhecimento para atualização dos dados do predicado utente.

```
evolucao( utente(A, B, C, D) ) :-  demo(utente(A, B, C, D), R),
                                   R = falso,
                                   registar(utente(A, B, C, D)).
```

Figura 13: Evolução normal do conhecimento para o predicado utente.

3.5. Implementação de um sistema de inferência

A implementação do sistema de inferência, começou pela criação do meta-predicado *demo: Questao, Resposta -> {V,F}*, o qual tal como se pode observar pelo tipo, recebe uma questão e coloca na *Resposta*, um dos valores do conjunto de soluções, que são: *verdadeiro*, *falso* e *desconhecido.*

A determinação da resposta sobre a questão colocada no meta-predicado *demo* é verdadeira, caso exista conhecimento positivo sobre a mesma, falsa, caso exista conhecimento negativo, e desconhecido, caso não exista nenhum dos conhecimentos, tal como se pode observar na figura seguinte a definição do termo *demo*.

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

Figura 14: Definição do predicado demo.

Devido ao facto de o meta-predicado demo tratar apenas uma questão, definiu-se o meta-predicado *demoConjunto*, que trata um conjunto de questões que podem estar relacionadas como conjunções e disjunções, implicações, disjunções explícitas.

Deste modo, foi necessário definir também os meta-predicados *conjuncao, disjuncao, disjuncaoExplicita e implicacao*, pois existem três possíveis valores de resposta como já se viu anteriormente, logo é necessário saber avaliar o valor lógico da expressão, quer com valores verdeiro e falso, quer com o desconhecido.

Em seguida é ilustrado o predicado demoConjunto e o meta-predicado conjuncao, os outros três meta-predicados são idênticos mudando apenas as suas tabelas de verdade.

```
% Extensao do meta-predicado demoConjunto: Questao,Resposta -> {V,F}
demoConjunto(Q1/\Q2, R):- demoConjunto(Q2,R1),  % conjunção
            demoConjunto(Q1,R2),
            conjuncao(R1,R2,R).  % faz a conjunção das respostas anteriores

demoConjunto(Q1\/Q2, R):- demoConjunto(Q2,R1),  % disjunção
            demoConjunto(Q1,R2),
            disjuncao(R1,R2,R).  % faz a disjunção das respostas anteriores

demoConjunto(Q1+Q2, R):- demoConjunto(Q2,R1),  % disjunção explicíta
            demoConjunto(Q1,R2),
            disjuncaoExplicita(R1,R2,R).

demoConjunto(Q1\Q2, R):- demoConjunto(Q2,R1),  % implicacao
            demoConjunto(Q1,R2),
            implicacao(R1,R2,R).

demoConjunto(Q, R):- demo(Q,R).                %caso normal, uma questão
```

Figura 15: Definição do predicado demoConjunto.

```
% Extensao do meta-predicado conjuncao: A , B , Resposta -> {V,F}
% soluções possíveis entre o desconhecido e o verdadeiro
conjuncao(verdadeiro, desconhecido, desconhecido).
conjuncao(desconhecido, verdadeiro, desconhecido).

% soluções possíveis entre o desconhecido e o falso
conjuncao(falso, desconhecido, falso).
conjuncao(desconhecido, falso, falso).

% tabela de verdade tradicional para a conjunção
conjuncao(verdadeiro, verdadeiro, verdadeiro).
conjuncao(falso, falso, falso).
conjuncao(falso, verdadeiro, falso).
conjuncao(verdadeiro, falso, falso).

% conjuncao do desconhecido
conjuncao(desconhecido, desconhecido, desconhecido).
```

Figura 16: Definição do predicado conjuncao.

Tabelas de verdade para os quatro meta-predicados:

	V	D	F
V	V	D	F
D	D	D	F
F	F	F	F

Tabela 1: Tabela de verdade da conjuncao.

	V	D	F
V	V	V	V
D	V	D	D
F	V	D	F

Tabela 2: Tabela de verdade da disjuncao.

	V	D	F
V	F	D	V
D	D	D	D
F	V	D	F

Tabela 3: Tabela de verdade da disjuncaoExplicita.

	V	D	F
V	V	D	F
D	V	D	D
F	V	V	V

Tabela 4: Tabela de verdade da implicacao.

Conclusões e Sugestões

Em suma, os objetivos inicialmente propostos foram cumpridos pelo grupo, dos quais, implementação do conhecimento positivo e negativo, conhecimento imperfeito e um sistema de inferência.

Inicialmente, o grupo estava um pouco confuso acerca do conceito de conhecimento imperfeito, mas a ajuda dos professores e dos materiais de apoio disponibilizados por estes, foi determinante no esclarecimento de dúvidas para a devida resolução ser realizada com sucesso.

Assim, a base de conhecimento poderia ser mais estendida, mas para o trabalho proposto é suficiente para demonstrar todos os tipos de conhecimento imperfeito bem como os invariantes desenvolvidos para a gestão da base de conhecimento.