Universidade do Minho

Mestrado Integrado em Engenharia Informática

Sistemas de Representação de Conhecimento e Raciocínio

Autores:

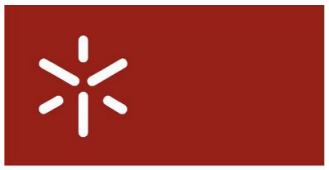
A75625 André Almeida Gonçalves

A75428 Bruno Miguel Sousa Cancelinha

A74576 José António Dantas Silva

A74817 Marcelo António Caridade Miranda

19 de Março de 2017



Universidade do Minho

Resumo

O projeto apresentado consiste no desenvolvimento de um sistema em PROLOG capaz de representar o conhecimento sobre um sistema de saúde básico. Para que a base de conhecimento seja representada de forma coerente foi feito uso de mecanismos como invariantes. Embora o sistema seja admissível, é ainda possível acrescentar várias funcionalidades de forma a possibilitar uma melhor representação da realidade.

Conteúdo

1 Introdução **3**

2 Descrição do Sistema **4**

3 Descrição do trabalho e análise de resultados **6**

3.1 Base de conhecimento 6

3.2 Invariantes 7

3.3 Predicados 8

4 Conclusão e Sugestões **10**

1 Introdução

Este projeto foi proposto no âmbito da unidade curricular Sistemas de Representação de Conhecimento e Raciocínio com o objetivo de fundamentar os conhecimentos lecionados até então. O projeto consiste na implementação de um sistema capaz de caracterizar uma gama de conhecimentos no domínio dos cuidados de saúde, com um foco na gestão de utentes e na realização de serviços e atos médicos.

Este relatório começará por contextualizar o leitor através da descrição e especificação das capacidades do sistema desenvolvido. Seguidamente é feita uma análise técnica do trabalho realizado, na qual é descrito os aspetos teóricos que fundamentam o trabalho, assim como uma análise dos resultados obtidos. O relatório termina com uma análise crítica acerca do projeto.

2 Descrição do Sistema

O sistema aqui descrito tem como objetivo a representação de conhecimento e raciocínio no domínio dos cuidados de saúde. Como tal, este deve ser capaz de descrever diferentes aspetos do domínio como, por exemplo, utentes, médicos, serviços prestados e atos médicos realizados. Para além disso, deve ainda ser capaz de refletir sobre os conhecimentos adquiridos, sendo capaz de identificar os atos médicos realizados num dado utente ou os serviços fornecidos por uma dada instituição.

Prosseguiremos agora à caracterização detalhada de cada um dos aspetos que o sistema deve ser capaz de representar.

- **Utente**

O utente é o indivíduo sobre o qual é realizado um ou mais atos médicos. Este é descrito pelo seu nome, idade e local de residência. De forma a que um utente possa ser facilmente identificados é-lhe ainda atribuído um número de identificação único.

- **Médico**

O médico é um indivíduo especializado num dado serviço e é responsável pela realização dos atos médicos. Um médico é caraterizado pelo seu nome, a sua especialização e o conjunto de instituições nas quais presta serviços. Tal como o utente, este possui também um número de identificação único.

- **Serviço**

Um serviço descreve um conjunto de atos médicos que podem ser realizados numa dada instituição. Estes são descritos por um número de identificação único, uma designação, a instituição em que o serviço é realizado e a cidade em que esta instituição se situa.

- **Ato Médico**

Um ato médico descreve a prestação de um serviço por parte de um médico a um utente. Assim, um ato médico deve sempre referir o número de identificação do utente ao qual foi prestado auxílio, o número de identificação do médico que prestou o serviço e o número de identificação do serviço prestado. Este deve especificar a data em que o serviço ocorreu e custo imposto ao utente.

Para além da capacidade de suportar a representação do conhecimento acima descrito, o sistema deve ainda suportar as seguintes funcionalidades:

√ Registar utentes, médicos, serviços prestados e atos médicos.

√ Identificar utentes/médicos a partir de vários critérios de seleção distintos.

√ Identificar quais as instituições que prestam um determinado serviço de saúde.

√ Identificar os serviços prestados por uma instituição/cidade.

√ Identificar os utentes de um médico/serviço/instituição.

√ Identificar os atos médicos realizados por utente/médico/serviço/instituição.

√ Determinar todos os médicos/serviços/instituições a que um utente já recorreu.

√ Calcular o custo total dos atos médicos por utente/serviço/instituição/data.

√ Remover utentes, médicos, serviços prestados e atos médicos.

3 Descrição do trabalho e análise de resultados

Como foi explicado acima, o objetivo deste trabalho consiste em implementar um sistema de representação de conhecimento. Assim o trabalho divide-se em três partes, Base de conhecimento, Invariantes e Predicados.

3.1 Base de conhecimento

Tal como foi explicado no capítulo anterior, o projeto caracteriza o conhecimento da seguinte forma:

Utente: #IdUt, Nome, Idade, Morada $\rightarrow \{\mathbb{V}, \mathbb{F}\}$

O #IdUt é um número que identifica, dentro dos utentes, este utente em particular. A Idade é também um número, enquanto que o Nome e a Morada são simples strings.

Servico: #IdServ, Descrição, Instituição, Cidade $\rightarrow \{\mathbb{V}, \mathbb{F}\}$

O #IdServ é o identificador deste serviço. A Descrição, Instituição e Cidade são, todas elas, strings. Uma instituição pode existir em várias cidades diferentes.

Medico: #IdMed, Nome, #IdServ, [Instituições] $\rightarrow \{\mathbb{V}, \mathbb{F}\}$

O #IdMed é o número correspondente a este médico, O Nome e as Instituições são ambas strings. A especialização do médico é indicada pelo id do Serviço que ele efetua.

Ato: Data, #IdUt, #IdServ, #IdMed, Custo $\rightarrow \{\mathbb{V}, \mathbb{F}\}$

O ato não contém Identificador próprio, mas contém o identificador do Utente, Serviço e Médico, para além do Custo e da Data. Todos eles, excepto a Data, são números. A Data é uma string com o formato "AAAA-MM-DD" para que seja fácil na eventualidade de ordenar os atos pela data.

3.2 Invariantes

Para que seja possível desfrutar do conhecimento, é necessário garantir que este é coeso, ou seja, que é regido segundo certas regras. Essa é a função dos invariantes, garantir a consistência do conhecimento. Há dois tipos de invariantes presentes no nosso trabalho, invariantes de inserção, que garantem a consistência aquando da inserção, e os de remoção que, como é de esperar, garantem-na aquando da remoção. Para diferenciar os dois, os invariantes de inserção têm um prefixo '+', enquanto que os de remoção têm o prefixo '-'. Assim é possível procurar, durante qualquer das duas operações, quais os invariantes associados a cada uma das operações. Vamos observar um caso concreto. Primeiro é descrito um dos invariantes, neste exemplo temos a implementação do invariante que garante que o id de um utente tem de ser único.

```
+utente(Id, _, _, _) :: (
    findall( Id, (utente(Id, _, _, _)), S),
    length(S, 1) ).
```

Note-se o uso do prefixo '+' para esclarecer que este invariante terá que ser validado depois da inserção de um utente. O símbolo '::' foi definido como um operador com argumentos à esquerda e à direita, à esquerda a identificação do invariante, à direita o invariante em si. O invariante garante, com auxílio do predicado findall/3, que a lista de todos os utentes com id Id deve ter um só elemento, ou seja, que existe apenas um utente com esse id.

```
registar(Q) :-
    evolucao(Q).

evolucao(Q) :-
    findall(I, +Q::I, L),
    insere(Q),
    testar(L).

insere(Q) :- assert(Q).
insere(Q) :- retract(Q), !, fail.
```

O registar é apenas um *wrapper* do predicado evolucao/1. Este predicado garante que os invariantes relativos ao conhecimento que será inserido não é quebrado. Para isso, recolhe para uma lista todos os invariantes com ajuda do predicado findall/3, tomando partido do prefixo '+' e do operador '::'.

Insere o conhecimento e testa a lista de invariantes. No acaso da inserção não respeitar os invariantes, o predicado testar/1 falha e o insere/1 procede à sua remoção falhando também.

```
-? utente_id(1, U).
U = utente(1, "Maria Goncalves", 12, "Guimaraes").

-? registar(utente(1, "Marcelo Sousa", 68, "Belem").
false.
```

Listing 1: Não foi possível registar o utente por violar o invariante.

Quando falamos da remoção, os invariantes devem apresentar o contexto em que é válida a remoção. Por exemplo, a remoção de um utente é válida se não houver nenhum ato médico que referencia esse utente.

```
-utente(Id, _, _, _) :: (
    findall(Id, ato(_, Id, _, _, _), []) ).

involucao(Q) :-
     findall(I, -Q::I, L),
     testar(L),
     remove(Q).
```

Assim, o involucao/1 lista todos os invariantes relevantes (note-se o uso do prefixo '-') e testa-os para se certificar que estamos num contexto válido para a remoção desse conhecimento.

```
?- atos_utente(1, A).
A = [ato("2015-01-01", 1, 1, 1, 0), ato("2015-01-01", 1, 1, 1,
    10), ato("2015-01-11", 1, 10, 7, 10)].

?- remover(utente(1, "Maria Goncalves", 12, "Guimaraes")).
false.
```

Listing 2: Não foi possível remover o utente por violar o invariante.

3.3 Predicados

Os predicados manipulam o conhecimento, ou seja, podem inserir/remover conhecimento ou até aplicar mecanismos de raciocínio sobre esse mesmo

conhecimento. Todas as funcionalidades listadas no capítulo anterior foram implementadas através de predicados.

```
?- utentes_idade(12, Us).
Us = [utente(1, "Maria Goncalves", 12, "Guimaraes"), utente(11,
    "Vitoria Alves", 12, "Guimaraes")].
```

Listing 3: Utentes com idade 12.

Tal como foi esperado, o sistema consegui apresentar uma lista com todos os utentes de idade 12.

```
?- medicos_especialidade(2, Ms).
Ms = [medico(2, "Antonio Egas Moniz", 2, ["Sao Cruz", "Santa
    Maria"]), medico(4, "Maria Fernandes", 2, ["Santa Maria"])].
```

Listing 4: Médicos com especialidade 2

Tal como foi esperado, o sistema consegui apresentar uma lista com todos os medicos com especialidade 2.

```
?- instituicoes(I).
I = ["HES", "Santa Casa da Misericordia", "IPO", "Sao Cruz",
    "Santa Maria", "Senhora da Oliveira"] .
```

Listing 5: Todas as instituições

Tal como foi esperado, o sistema consegui apresentar uma lista com todas as instituições.

```
?- servicos_instituicao("Santa Casa da Misericordia", S).
S = [servico(7, "Imuno-hemoterapia", "Santa Casa da Misericordia",
    "Riba de ave")].
```

Listing 6: Serviços da instituição "Santa Casa da Misericodia".

Tal como foi esperado, o sistema consegui apresentar uma lista dos serviços da instiuição dada.

```
?- atos_utente(9, As).
As = [ato("2015-01-05", 9, 6, 15, 25), ato("2015-01-19", 9, 4, 3,
    3)].
```

Listing 7: Atos do utente 9.

Tal como foi esperado, o sistema consegui apresentar uma lista dos atos do utente 9.

4 Conclusão e Sugestões

Terminamos o projeto com um sistema capaz de representar o conhecimento pedido, assim como de aplicar os mecanismos de raciocínio especificados. Embora tenham sido ainda acrescentadas algumas funcionalidades, o sistema desenvolvido constitui uma simples aproximação à realidade, pelo que, no futuro, seria possível estender a capacidade do mesmo. Para tal poderíamos, por exemplo, acrescentar predicados capazes de descrever os quartos disponíveis ou até os enfermeiros em serviço.

Assim, concluímos que os objetivos propostos foram cumpridos. Concordamos também que este primeiro trabalho constituiu uma etapa importante na fundamentação do material lecionado até então, nomeadamente a representação e manipulação de conhecimento no âmbito da programação em lógica.