UNIVERSIDADE DO MINHO

Health Care System

Mestrado Integrado de Engenharia Informática

Sistemas de Representação de Conhecimento e Raciocínio
(2ºSemestre/2017-2018)

Número	Nome do(s) Autor(es)
a78468	João Vieira
a78821	José Martins
a77049	Miguel Quaresma
a77689	Simão Barbosa

Braga, Portugal
17 de Março de 2018

1 Resumo

O objetivo deste projeto é criar um sistema de representação de conhecimento e raciocínio na caraterização de um universo na área da prestação de cuidados de saúde, em que as medidas de sucesso mínimas são as propostas pelo enunciado. Contudo foram desenvolvidas algumas funcionalidades extras para enriquecer a base de conhecimento.

Conteúdo

2 Introdução

Health Care System é um Sistema de Representação de Conhecimento e Raciocínio usado na caraterização de um universo na área da prestação de cuidados de saúde desenvolvido usando a linguagem de programação em lógica PROLOG. O projeto desenvolvido apresenta funcionalidades básicas associadas ao sistema em questão tendo ainda sido complementado com outras (funcionalidades) que complementam o universo em causa.

3 Descrição do Trabalho e Análise de Resultados

3.1 Base de Conhecimento

A base de conhecimento do sistema desenvolvido é essencial à representação do conhecimento e raciocínio, como tal, tendo em conta o sistema em questão, foram desenvolvidas as seguintes entidades:

- utente: #IdUt, Nome, Idade, Morada → {V,F}
- prestador: #IdPrest, Nome, Especialidade, Instituição → {V,F}
- cuidado: Data, #IdUt, #IdPrest, Descrição, Custo → {V,F}
- instituicao: #IdIt, Nome, Tipo, Cidade → {V,F}

São estas as entidades que servirão de suporte ao sistema desenvolvido.

3.2 Funcionalidades

As regras de uma base de conhecimento conferem-lhe a utilidade necessária ao seu funcionamento visto que, se não for possível efetuar questões e obter respostas às mesmas, a utilidade da base de conhecimento é nula. Em PROLOG estas funcionalidades/regras são implementadas tendo por base as caraterísticas desta linguagem:

- Algoritmo de Resolução: mais propriamente o *Modus Tollens* ({A se B,¬A}⊢ ¬B) no qual uma questão é verdade se adicionando a negação da mesma à base de conhecimento origina uma inconsistência
- Clausulado de Horn: todas as regras são uma cláusula de Horn **i.e.** o qual admite apenas 1 termo positivo(conclusão), as fórmulas são bem formadas e fechadas, quantificadas universalmente (**e.g**. $\forall A, B$ avo(A,B)⊢neto(B,A))
- Mecanismo de *backtracking*: na presença de uma solução inválida(falsa) esta meta-heurística continua à procura de "outro caminho"de modo a que a regra seja verdadeira.

Adicionalmente assumem-se vários pressupostos de entre os quais:

- Pressuposto dos Nomes Únicos: duas constantes designam duas entidades diferentes
- Pressuposto do Mundo Fechado: tudo o que não existe mencionado é falso
- Pressuposto do Domínio Fechado: não há mais objetos no universo para além dos designados por constantes

Por fim o sistema de representação de conhecimento e raciocínio desenvolvido deve respeitar as "leis"da lógica recorrendo para isso a invariantes que garantem que certas propriedades são respeitadas. Estes (invariantes) garantem a inexistência de inconsistências bem como a preservação do "significado do conhecimento"e são usados na evolução e involução da base de conhecimento.
Foram assim, implementadas as seguintes funcionalidades:

- Registar utentes, prestadores e cuidados de saúde,instituicoes
  registarUtente: Id, Nome, Idade, Morada → {V,F}
  registarPrestador: Id, Nome, Especialidade, Instituição → {V,F}
  registarCuidado: Data, IdUtente, IdPrestador, Descrição, Custo → {V,F}
  registarInstituicao: Id, Nome, Tipo, Cidade → {V,F}
- Remover utentes, prestadores e cuidados de saúde, instituicoes
  removerUtente: Id, Nome, Idade, Morada → {V,F}
  removerPrestador: Id, Nome, Especialidade, Instituição → {V,F}
  removerCuidado: Data, IdUtente, IdPrestador, Descrição, Custo → {V,F}
  removerInstituicao: Id, Nome, Tipo, Cidade → {V,F}

- Identificar utentes por critérios de seleção
  identificaUtente : nome, NomeUtente, Solução → {V,F}
  identificaUtente : idade, IdadeUtente, Solução → {V,F}
  identificaUtente : morada, MoradaUtente, Solução → {V,F}

- Identificar as instituições prestadoras de cuidados de saúde
  identificaInstituicoes : Solução → {V,F}

- Identificar cuidados de saúde prestados por instituição/cidade/datas
  identCuidPrest : instituicao, Instituicao, Resultado → {V,F}
  identCuidPrest : cidade, Cidade, Resultado → {V,F}
  identCuidPrest : datas, Data, Data, Resultado → {V,F}

- Identificar os utentes de um prestador/especialidade/instituição
  identUtentes : prestador, IdPrestador, Resultado → {V,F}
  identUtentes : especialidade, Especialidade, Resultado → {V,F}
  identUtentes : instituicao, Instituicao, Resultado → {V,F}

- Identificar cuidados de saúde realizados por utente/instituição/prestador
  identificaCuidadosRealizados: utente, IdUtente, Resultado → {V,F}
  identificaCuidadosRealizados: instituicao, Instituição, Resultado → {V,F}
  identificaCuidadosRealizados: prestador, IdPrestador, Resultado → {V,F}

- Determinar todas as instituições/prestadores a que um utente já recorreu
  porUtente: instituicao, IdUtente, Resultado → {V,F}
  porUtente: prestador, IdUtente, Resultado → {V,F}

- Calcular o custo total dos cuidados de saúde por utente/especialidade/prestador/datas
  custoTotal: utente, IdUtente, Resultado → {V,F}
  custoTotal: especialidade, Especialidade, Resultado → {V,F}
  custoTotal: prestador, IdPrestador, Resultado → {V,F}
  custoTotal: datas, Data, Data, Resultado → {V,F}

3.2.1 Funcionalidades Extra

Por forma a apresentar uma descrição mais fiel do sistema em causa foram desenvolvidas/implementadas funcionalidades extra que permitem uma interação mais complexa com a base de conhecimento.
Tendo em conta as entidades sugeridas no enunciado (utente, prestador e cuidado) foi considerado pelo grupo como pertinente adicionar funcionalidades que permitam obter as seguintes informações:

- Determinar as especialidades com que um utente esteve relacionado, devolvendo a data das mesmas
  especialidadesDeUtente : IdUtente, Solução → {V,F}

- Determinar os prestadores que cuidaram de um utente numa dada instituição
  prestadoresDeUtenteEmInstituicao : IdUtente, Solução → {V,F}

- Que utente tem mais cuidados realizados
  utenteComMaisCuidados: Resultado → {V,F}

- Que prestador prestou mais cuidados
  prestadorComMaisCuidados: Resultado → {V,F}

Para além disto, a nova entidade sugerida pelo grupo (instituição) permitiu aumentar o leque de funcionalidades e de informação a ser possível recolher da base de conhecimento em causa.
Tendo isto em conta, foram implementadas as seguintes funcionalidades:

- Determinar as instituições existentes numa cidade
  instituicoesDeCidade: Cidade, Solução → {V,F}

- Determinar os tipos de instituições existentes numa cidade
  tiposDeInstituicoesDeCidade: Cidade, Solução → {V,F}

- Determinar o tipo de instituições que um utente já visitou
  tiposInstituicoesVisitadasPorUtente: IdUtente, Solução → {V,F}
- Que instituições são hospitais/clínicas/centros de saúde
  instituicoesDoTipo: Tipo, Solução → {V,F}
- Em que cidade foram mais cuidados realizados
  cidadeComMaisCuidados: Resultado → {V,F}
- Qual a instituição com mais cuidados realizados
  instituicaoComMaisCuidados: Resultado → {V,F}
- Em que instituição o utente realizou mais cuidados
  instituicaoMaisFrequentadaPorUtente: IdUtente, Resultado → {V,F}

3.2.2 Invariantes

Com a implementação de certos invariantes considerados como importantes para o contexto em causa, a base de dados de conhecimento em PROLOG fica assim mais coerente e menos suscetível a falhas. Tendo isto em conta, foram implementados 10 invariantes que permitem que:

- não existam utentes com *id's* repetidos
- não existam prestadores com *id's* repetidos
- não existam instituições com *id's* repetidos
- um prestador tenha que pertencer obrigatoriamente a uma instituição da base de conhecimento
- não existam cuidados repetidos
- seja garantida a existência do utente do cuidado a adicionar
- seja garantida a existência do prestador do cuidado a adicionar
- não pode ser removido um utente com cuidados associados
- não pode ser removido um prestador com cuidados associados
- não pode ser removida uma instituição se tiver associada a prestadores

3.2.3 Funções Auxiliares

O desenvolvimento do sistema em causa envolveu, por vezes, o uso de regras que partilhavam certas operações entre si, como tal estas operações foram degeneradas em regras individuais por forma a reduzir a quantidade de código necessária. As operações referidas encontram-se descritas de seguida:

- Encontra todos os predicados(Questão) que sejam satisfeitos ao efetuar o *backtracking* tendo Formato em conta
  solucoes : Formato, Questao, Soluçoes → {V,F}
- Função para somar uma lista
  somaLista: Lista,Solucao → {V,F}
- Conta o número de ocorrências de um elemento numa lista
  contaOcorrencias: Elemento,Lista,Solucao → {V,F}
- Calcula o elemento mais frequente de um par
  maxFreqPair: Elemento1,Frequencia,Elemento2,Frequencia,Solucao → {V,F}
- Calcula o elemento mais frequente de uma lista
  maxRepeticoes: Lista,Solucao → {V,F}
- Inserir conhecimento
  inserir: Termo → {V,F}

- Remover conhecimento
  remover: Termo → {V,F}
- Regra de teste dos invariantes correspondentes
  test: Lista → {V,F}

4 Conclusões e Sugestões

Para concluir, os objetivos propostos pelo enunciado foram cumpridos mas também foram adicionadas novas funcionalidades com o propósito de fundamentar a base de conhecimento.Ainda assim é possível adicionar novas funcionalidades extras ao trabalho apresentado para enriquecer ainda mais a *Health Care System*.

Referências

[1] MARTINS, José Carlos Lima, *Notas Teóricas*, José Carlos Lima Martins, 2018.

[2] VIEIRA, João Pedro Ferreira, *Notas Teóricas*, João Pedro Ferreira Vieira, 2018.

[3] QUARESMA, Miguel Miranda, *Notas Teóricas*, Miguel Miranda Quaresma, 2018.

[4] BARBOSA, Simão Paulo Leal, *Notas Teóricas*, Simão Paulo Leal Barbosa, 2018.