**Universidade do Minho**

**Exercício 1 – Registo de Eventos numa Instituição de Saúde**

Mestrado Integrado em Engenharia Informática - MiEI

Sistemas de Representação de Conhecimento e Raciocínio

(2º Semestre/2015-2016)

A72223 Gustavo da Costa Gomes

A71223 José Carlos da Silva Brandão Gonçalves

A70443 Tiago João Lopes Carvalhais

Local - Braga

Data - 22/03/2016

**Resumo**

Este documento serve de apoio ao primeiro exercício prático da unidade curricular de Sistemas de Representação de Conhecimento e Raciocínio, cujo objetivo principal é o de motivar para a utilização da linguagem de programação em lógica - *PROLOG*.

Numa primeira fase, serão explicadas as estratégias definidas para representar a informação pretendida, constituindo assim a base de conhecimento para este trabalho. De seguida, serão abordados todos os objetivos propostos no enunciado, bem como as técnicas utilizadas para os solucionar. Para a concretização dos mesmos, foram utilizados vários predicados auxiliares, pelo que também será efetuada uma explicação pormenorizada acerca dos mesmos. Para além disso, foi-nos proposta também a possibilidade de melhorar o trabalho proposto, e como tal, neste relatório serão apresentados todos os pontos realizados mediante esses objetivos, e que permitirão enriquecer mais este trabalho prático.

Num fase mais terminal deste relatório, serão ainda colocados exemplos de como questionar a base de conhecimento acerca das funcionalidades criadas, bem como os resultados produzidos pela mesma.

**Índice**

**Introdução**

O objetivo deste trabalho é a utilização da linguagem de programação em lógica –*PROLOG* - para representar conhecimento e criar mecanismos que atuem sobre o mesmo, de modo a resolver os problemas propostos.

O problema em si é bastante conhecido e, como tal, facilitou a fase inicial de análise dos requisitos necessários á sua elaboração, uma vez que os conhecimentos adquiridos através de experiências pessoais anteriores permitem compreender o assunto em questão, bem como as componentes que cada um dos predicados possam eventualmente possuir, por forma a cumprir os requisitos mínimos.

Para além destes, desenvolveram-se predicados um pouco diferentes no que diz respeito ao tipo de questões que possam ser colocadas, e que podem ser utilizadas como uma ferramenta de avaliação da qualidade dos diversos sistemas, ainda que, de uma forma subtil, visto que as interações entre estes sistemas não são, por vezes, assim tão lineares, pelo facto de existirem fatores externos a cada um deles, o que pode influenciar as suas decisões.

O sistema desenvolvido deve ser capaz de representar o tema do registo de eventos em instituições de saúde, bem como apresentar as várias funcionalidades que irão operar sobre a base de conhecimento, que é constituída por registos e fichas de utentes, serviços disponíveis, profissionais de saúde que atendem os pacientes e instituições de saúde onde estes recorrem.

**Preliminares**

Para o desenvolvimento dos predicados propostos foi necessário fazer uma pequena análise sobre o que era necessário para representar o conhecimento. O grupo começou por caracterizar o utente, isto é, definir todos os parâmetros que definem o que é ser um utente. Este terá um registo, ou seja, o conteúdo de todas as informações pessoais, sendo que neste caso apenas terá o nome, a idade e a localidade. Terá também uma ficha de utente, que é definida pelo nome, pelo serviço a que está a ser consultado, pelo profissional atribuído e pela instituição onde está a ser atendido.

Na realidade, o utente é composto por muitos outros fatores, tais como, número de utente, número de contribuinte, número de cartão de cidadão, afiliação, estado civil, tipo de sangue, historial clínico, entre muitos outros, mas que, para a resolução dos critérios designados como mínimos, apenas são requeridos aqueles que foram mencionados anteriormente, unicamente porque as questões não pretendem incidir sobre esses campos, sendo que estes apenas poderiam ser utilizados posteriormente na reavaliação do sistema para se produzirem predicados extras. Optámos por separar o utente no seu registo e na sua ficha de atendimento numa instituição, para que houvesse uma maior organização para uma posterior elaboração de predicados referentes ao mesmo, sendo que este pode ter uma ficha em mais do que uma instituição, mas o seu registo será único.

Um serviço é caracterizado pela sua designação, isto é, o nome, que terá de ser elucidativo, por exemplo, “cirurgia”, e pela instituição onde este é efetuado. Na realidade, um serviço é muito mais complexo e passa por ter chefes de equipa, e por vezes especificação detalhada, como por exemplo, “cirurgia cardíaca”, que é diferente de “cirurgia cerebral”. Isto é, dentro do serviço existem categorias e para cada categoria existem chefes de equipa e as respetivas equipas. Generalizando o exemplo, teríamos um chefe do serviço geral “cirurgia” que teria uma equipa de chefes de categorias desse serviço e cada categoria teria uma equipa de profissionais. Mas nada disto é utilizado para cumprir os requisitos mínimos propostos e, tal como nos casos anteriores, é sim algo que pode ser utilizado como uma extensão do projeto.

Um profissional é caracterizado pelo seu nome, pelo serviço em que está inserido e pela instituição onde labora, sendo que este pode trabalhar em mais do que um serviço e em mais do que uma instituição. Estes são apenas os requisitos mínimos que permitem ao sistema determinar as respostas a todas as questões, mas um profissional é algo mais que isto, pois tem também um chefe associado, uma especialidade, um identificador dentro da instituição, visto que, como já foi dito, um profissional pode exercer funções em mais que uma instituição, e em mais do que um serviço. Para além destes possui também informações semelhantes ás do utente e que dizem respeito ao facto de serem cidadãos, sendo características possíveis de utilizar em futuras extensões.

Uma instituição é caracterizada apenas pelo seu nome, por forma a simplificar este sistema, visto que os requisitos mínimos não pretendem questionar nada que implique que a mesma tenha de ter algo mais no seu predicado além do seu nome. Mas, como é óbvio, uma instituição é muito mais do que um nome, isto é, quando alguém pensa numa instituição de saúde não pensa unicamente no seu nome. Na realidade, esta é composta por vários departamentos, um dos quais é o departamento responsável pela aplicação dos diversos serviços e é nesse que se encontram os profissionais, possuindo ainda colaboradores, outros serviços, como cafetarias, salas de espera, balcões de atendimento ou de reclamações, entre muitos outros. No entanto, restringe-se esse conceito ao simples facto de ser um local onde profissionais prestam serviços a utentes.

**Descrição do Trabalho e Análise de Resultados**

**Representação de Conhecimento**

Para a representação dos vários tipos de conhecimento existentes neste trabalho prático, foram criados na base de conhecimento factos para registos de utentes, fichas de utentes, serviços, profissionais e instituições. Como já foi dito na secção anterior a instituição é caracterizada apenas pelo seu nome e está escrita na seguinte forma:

instituicao(hospital\_braga).

Desta forma podemos referir de forma muito simples que o hospital\_braga é uma instituição.

Já os serviços requerem, para além do seu nome, a instituição onde estão presentes, sendo isso exemplificado da seguinte forma:

servico(cirurgia, hospital\_braga).

Através da análise deste exemplo, podemos dizer que existe o serviço cirurgia na instituição hospital\_braga, pelo que, podemos, mais tarde, dizer quais os serviços presentes numa determinada instituição.

De seguida, vamos analisar a base de conhecimento sobre profissionais, sendo que este é caracterizado pelo seu nome, pelo serviço em que se especializa e pela instituição onde executa esse serviço. O exemplo abaixo demonstra isso mesmo:

profissional(pedro\_soares, cirurgia, hospital\_braga).

Neste caso, podemos referir que o profissional pedro\_soares é especialista no serviço cirurgia na instituição hospital\_braga, sendo que, há semelhança do caso dos serviços, podemos agora também construir predicados que nos digam quais os profissionais que efetuam um determinado serviço, ou então, quais os profissionais que estão registados numa determinada instituição, por exemplo.

Prosseguimos agora para o conhecimento acerca dos utentes. Este é representado por dois tipos, o seu registo e a sua ficha de utente numa instituição. O primeiro contém o seu nome, a sua idade e a sua localidade. Já a segunda contém o seu nome, mas também o serviço a que recorreu, assim como o profissional que o atendeu e a instituição que frequentou. Estes dois tipos de factos estão representados abaixo:

registo\_utente(samuel\_cunha, 39, esposende).

ficha\_utente(samuel\_cunha, cirurgia, pedro\_soares, hospital\_braga).

Através da análise destes factos, podemos referir que samuel\_cunha está registado como um utente com 39 anos de idade e como sendo natural de esposende. Para além disso, podemos afirmar que samuel\_cunha é utente do serviço de cirurgia, tendo sido atendido pelo profissional pedro\_soares na instituição hospital\_braga. Estamos, agora, em condições para construir predicados que nos possam dizer quais os utentes que recorreram a uma determinada instituição, ou então os serviços mais recorridos, bem como os profissionais que atenderam mais utentes.

Antes de terminar este ponto e, para que a análise de resultados possa ser explicada de forma a que o leitor reflita sobre a correção dos mesmos, segue em baixo toda a base de conhecimento inserida no programa e que foi utilizada pelo grupo para testar os predicados desenvolvidos:

instituicao(hospital\_braga).

instituicao(hospital\_lisboa).

instituicao(ipo\_porto).

servico(oncologia, ipo\_porto).

servico(cirurgia, hospital\_braga).

servico(clinica\_geral, hospital\_braga).

servico(cirurgia, hospital\_lisboa).

servico(psiquiatria, hospital\_braga).

profissional(antonio\_abreu, oncologia, ipo\_porto).

profissional(manuel\_pereira, clinica\_geral, hospital\_braga).

profissional(pedro\_soares, cirurgia, hospital\_lisboa).

profissional(joao\_pereira, cirurgia, hospital\_braga).

profissional(duarte\_gomes, psiquiatria, hospital\_braga).

registo\_utente(jose\_esteves, 45, matosinhos).

registo\_utente(miguel\_silva, 13, gualtar).

registo\_utente(carlos\_sousa, 25, oeiras).

registo\_utente(samuel\_cunha, 39, esposende).

registo\_utente(joana\_fernandes, 72, lomar).

ficha\_utente(jose\_esteves, oncologia, antonio\_abreu, ipo\_porto).

ficha\_utente(miguel\_silva, clinica\_geral, manuel\_pereira,

hospital\_braga).

ficha\_utente(carlos\_sousa, cirurgia, pedro\_soares, hospital\_lisboa).

ficha\_utente(samuel\_cunha, cirurgia, joao\_pereira, hospital\_braga).

ficha\_utente(joana\_fernandes, psiquiatria, duarte\_gomes,

hospital\_braga).

**Identificar os serviços existentes numa instituição**

Para identificar os serviços existentes numa instituição, foi criado um predicado servicos\_instituicao, que, recebendo o nome da instituição, irá retornar a lista dos serviços que lá são prestados:

servicos\_instituicao(I, S) :- findall(X, servico(X, I), S).

Como se pode verificar, no predicado acima é utilizado o *findall*, que irá procurar todos os serviços que são prestados na instituição I e guardá-los na lista de serviços S.

Segue em baixo um exemplo de uma questão proposta sobre este predicado:

?- servicos\_instituicao(hospital\_braga, S).

S = [cirurgia,clinica\_geral,psiquiatria] ?

yes

Como se pode verificar, o programa respondeu que na instituição hospital\_braga os serviços que são disponibilizados são cirurgia, clinica\_geral e psiquiatria. Ora, como referimos anteriormente, na base de conhecimento dissemos que, efetivamente, o serviço cirurgia era efetuado na instituição hospital\_braga, assim como os restantes, pelo que o resultado obtido foi o esperado, podendo confirmar-se que o programa funciona tal como era suposto.

**Identificar os utentes de uma instituição**

Para identificar os utentes que foram atendidos numa instituição, criámos uma predicado utentes\_instituicao, que, para uma dada instituição, irá devolver a lista dos seus utentes:

utentes\_instituicao(I, R) :- findall(U, ficha\_utente(U, S, P, I), R).

Como se pode verificar, no predicado acima é utilizado o *findall*, que irá procurar todos os utentes que foram atendidos na instituição I e guardá-los na lista de utentes R.

Segue em baixo um exemplo de uma questão proposta sobre este predicado:

?- utentes\_instituicao(hospital\_braga, R).

R = [miguel\_silva,samuel\_cunha,joana\_fernandes] ?

yes

Pela análise do resultado, podemos verificar que os utentes miguel\_silva, samuel\_cunha e joana\_fernandes foram todos atendidos na instituição hospital\_braga. Mais uma vez e, conforme o que foi escrito acerca da base de conhecimento num tópico anterior, o resultado da questão já era esperado, até porque, na base de conhecimento consta a ficha do utente miguel\_silva na instituição hospital\_braga, por exemplo.

**Identificar os utentes de um determinado serviço**

Para identificar os utentes de um determinado serviço em geral, foi criado um predicado utentes\_servico, que, recebendo o nome do serviço, retorna a lista dos utentes que recorreram ao mesmo:

utentes\_servico(S, R) :- findall(U, ficha\_utente(U, S, P, I), R).

Como se pode verificar, no predicado acima é utilizado o *findall*, que irá procurar todos os utentes que recorreram ao serviço S e guardá-los na lista de utentes R.

Segue em baixo um exemplo de uma questão proposta sobre este predicado:

?- utentes\_servico(cirurgia, R).

R = [carlos\_sousa,samuel\_cunha] ?

yes

Analisando o resultado, verificamos que os utentes carlos\_sousa e samuel\_cunha requisitaram o serviço de cirurgia. Mais uma vez, este resultado foi o esperado, já que, por exemplo, dissemos anteriormente na ficha de utente que samuel\_cunha requisitou o serviço de cirurgia.

**Identificar os utentes de um determinado serviço numa instituição**

Este predicado é bastante semelhante ao anterior, tendo também o mesmo nome, com a diferença de que agora o que se pretende é uma lista com os utentes de um serviço numa dada instituição:

utentes\_servico(S, I, R) :- findall(U, ficha\_utente(U, S, P, I), R).

Como se pode verificar, no predicado acima é utilizado o *findall*, que irá procurar todos os utentes que recorreram ao serviço S na instituição I e guardá-los na lista de utentes R. Convém notar que este predicado é diferente do anterior, pois, apesar de ter o mesmo nome, tem mais um argumento que o antecedente.

Segue em baixo um exemplo de uma questão proposta sobre este predicado:

?- utentes\_servico(cirurgia, hospital\_braga, R).

R = [samuel\_cunha] ?

yes

Podemos verificar que o utente samuel\_cunha foi o único utente que recorreu ao serviço cirurgia na instituição hospital\_braga. Novamente, o resultado é o esperado, e relembrando o resultado do predicado anterior, em que foi revelado que o utente samuel\_cunha foi um dos utentes que recorreram ao serviço de cirurgia, sendo que também referimos na base de conhecimento que esse utente recorreu a esse serviço na instituição hospital\_braga.

**Identificar as instituições onde seja prestado um serviço**

Para identificar as instituições onde seja prestado um determinado serviço, construímos um predicado instituicoes\_servico, que recebe o nome do serviço e retorna a lista das instituições onde o mesmo é prestado:

instituicoes\_servico(S, I) :- findall(N, servico(S, N), I).

Como se pode verificar, no predicado acima é utilizado o *findall*, que irá procurar todas as instituições que contêm o serviço S e guardá-las na lista de instituições I.

Segue em baixo um exemplo de uma questão proposta sobre este predicado:

?- instituicoes\_servico(cirurgia, I).

I = [hospital\_braga,hospital\_lisboa] ?

yes

De acordo com o resultado da questão, as instituições que possuem o serviço de cirurgia são as instituições hospital\_braga e hospital\_lisboa. O resultado é o esperado, tendo em conta o que está contido na base de conhecimento, uma vez que, por exemplo, é referido que o serviço cirurgia existe na instituição hospital\_braga.

**Identificar as instituições onde seja prestado um conjunto de serviços**

Este predicado é similar ao anterior, com a diferença de que este se chama instituicoes\_servicos e recebe um conjunto de serviços, em vez de um único serviço, retornando a lista das instituições onde esses serviços são prestados:

instituicoes\_servicos([], []).

instituicoes\_servicos([S | T], I) :- findall(N, servico(S, N), Li),

instituicoes\_servicos(T, Lt),

concatenar(Li, Lt, I).

Como se pode verificar, no predicado acima é utilizado o *findall*, que irá procurar todas as instituições que contêm o serviço S e guardá-las na lista de instituições I. Este corresponde à cabeça da lista, sendo que o programa irá chamar o predicado de forma recursiva para achar as instituições para os serviços que se encontram na cauda lista. Por fim, o resultado final é obtido através da concatenação, sem repetidos, da lista de instituições dos serviços da cauda com a lista de instituições do serviço que se encontra à cabeça, através do uso do predicado concatenar, que será explicado no anexo, conforme sugerido pela equipa docente no enunciado do projeto, visto que se trata de um predicado auxiliar. O caso de paragem ocorre quando já não existem mais serviços na lista, sendo que o programa, nesse caso, retorna uma lista também vazia.

Segue em baixo um exemplo de uma questão proposta sobre este predicado:

?- instituicoes\_servicos([cirurgia, oncologia], I).

I = [hospital\_braga,hospital\_lisboa,ipo\_porto] ?

yes

Verificamos que as instituições onde se efetuam os serviços de cirurgia e oncologia são o hospital\_braga, hospital\_lisboa e ipo\_porto. Mais uma vez, o resultado obtido está de acordo com o esperado, tendo em conta o resultado do predicado anterior e aquilo que se sabe acerca do conteúdo da base de conhecimento.

**Identificar os serviços que não se podem encontrar numa instituição**

O predicado de nome nao\_servicos\_instituicao recebe o nome de uma instituição e devolve a lista de serviços que não são efetuados na mesma:

nao\_servicos\_instituicao(I, S) :- findall(X, servico(X, Y), L1),

findall(X, servico(X, I), L2),

remover(L2, L1, R),

tira\_repetidos(R, S).

Desta vez, a função findall é chamada duas vezes. Da primeira vez irá determinar a lista de todos os serviços registados na base de conhecimento, guardando o resultado em L1, já da segunda vez irá determinar a lista de todos os serviços da instituição I, guardando o resultado em L2. De seguida, o programa irá encarregar-se de retirar de L1 todo o conteúdo presente em L2, através do predicado auxiliar remover, guardando o resultado em R, sendo que os repetidos serão retirados através da chamada de outro predicado auxiliar, o tira\_repetidos, produzindo o resultado final. De notar que os predicados auxiliares referidos serão posteriormente explicados no anexo deste relatório.

Segue em baixo um exemplo de uma questão proposta sobre este predicado:

?- nao\_servicos\_instituicao(hospital\_braga, S).

S = [oncologia] ?

yes

Relembrando o primeiro predicado abordado, ou seja, o que determina a lista de serviços de uma instituição, verificámos que os serviços que são prestado na instituição hospital\_braga são cirurgia, clinica\_geral e psiquiatria, sendo que o único dos serviços que não estava nessa lista e existe na base de conhecimento era oncologia, que é precisamente o resultado desta questão, pelo que podemos concluir que este predicado produziu o resultado correto.

**Determinar as instituições onde um profissional presta serviço**

O predicado instituicoes\_profissional tem como objetivo o de determinar a lista de instituições onde um determinado profissional exerce as suas funções:

instituicoes\_profissional(P, I) :- findall(N, profissional(P, S, N),

R),

tira\_repetidos(R, I).

O programa começa com mais uma chamada da função *findall*, que constrói uma lista com os profissionais que trabalham na instituição dada, sendo que, como estes podem exercer funções na mesma instituição em mais do que um serviço, então é necessário eliminar potenciais instituições repetidas, através do predicado auxiliar tira\_repetidos.

Segue em baixo um exemplo de uma questão proposta sobre este predicado:

?- instituicoes\_profissional(joao\_pereira, I).

I = [hospital\_braga] ?

yes

Podemos observar que o profissional joao\_pereira exerce funções apenas na instituição hospital\_braga, tal como era suposto ser, pois é o que está explícito na base de conhecimento.

**Determinar todas as instituições, serviços ou profissionais a que um utente já recorreu**

Este predicado, denominado info\_utente, irá determinar a lista de todas as instituições, serviços ou profissionais a que um utente já recorreu, recebendo o nome do mesmo e uma *string* com o que se quer determinar, ou seja, se é uma lista de instituições, serviços, ou profissionais:

info\_utente(U, instituicoes, L) :- findall(I, ficha\_utente(U, S, P,

I), L).

info\_utente(U, servicos, L) :- findall(S, ficha\_utente(U, S, P, I),

L).

info\_utente(U, profissionais, L) :- findall(P, ficha\_utente(U, S, P,

I), L).

Em qualquer uma das três situações, o *findall* irá procurar na base de conhecimento as fichas de utentes que contenham o nome desse utente e construir a lista com o que se pretende.

Segue em baixo um exemplo de questão para cada um dos tipos de resultados:

?- info\_utente(samuel\_cunha, instituicoes, I).

I = [hospital\_braga] ?

yes

?- info\_utente(samuel\_cunha, servicos, I).

I = [cirurgia] ?

yes

?- info\_utente(samuel\_cunha, profissionais, I).

I = [joao\_pereira] ?

yes

Como podemos verificar, o utente samuel\_cunha recorreu à instituição hospital\_braga, ao serviço de cirurgia e foi atendido pelo profissional joao\_pereira, sendo então o resultado, mais uma vez, o correto, pelos factos que já foram discutidos anteriormente.

**Registar utentes, profissionais, serviços ou instituições**

Este predicado é bastante distinto dos restantes já mencionados, uma vez que requer alterações na base de conhecimento, neste caso a inserção de informação na mesma, o que até seria simples, bastando para tal o uso da função *assert*. Contudo, esta função permite que possam ocorrer factos repetidos na base de conhecimento, o que se pretende evitar e, para além disso permite a inserção de conhecimento inconsistente, como inserir uma ficha de utente relativa a uma instituição inexistente, por exemplo. Para evitar estes problemas, optámos pela criação de um predicado registar, que se encarregará de inserir um novo facto na base de conhecimento, mas só o fará, se todas estas restrições forem respeitadas, pelo que iremos também proceder à definição de vários invariantes, que consistem nas restrições escritas “à moda” do *PROLOG*. Em baixo passamos a explicar este predicado:

registar(T) :- findall(I, +T :: I, L), inserir(T), testar(L).

O programa irá começar por usar o *findall* para encontrar todos os invariantes de inserção (representados com um sinal + atrás) relativos à questão passada como argumento e fará uma lista com os mesmos. De seguida, irá inserir o facto na base de conhecimento, através do predicado auxiliar inserir e, posteriormente, irá usar outro predicado auxiliar, o testar, que se encarregará de testar se, após a inserção de conhecimento, todos os invariantes são verificados e, caso pelo menos um deles falhe, volta a chamar a função de inserção, onde a linha de exceção fará com que o facto anteriormente inserido seja removido, evitando assim que a base de conhecimento fique inconsistente.

Seguem em baixo dois exemplos de inserção de conhecimento, um para um caso de sucesso no registo, o outro para uma situação de insucesso:

?- registar(instituicao(hospital\_faro)).

yes

?- registar(instituicao(hospital\_braga)).

no

No primeiro caso, decidimos registar uma nova instituição, o hospital\_faro e, como não existia qualquer facto que referisse a existência da mesma na base de conhecimento, o registo foi efetuado com sucesso. No segundo caso, como a instituição hospital\_braga já existia na base de conhecimento, o programa não permitiu a sua inserção, por se tratar de um facto repetido, pelo que o seu registo retornou insucesso.

Seguem, de seguida, todos os invariantes de inserção utilizados para a resolução deste predicado:

% Não podem existir registos de utentes repetidos na base de conhecimento

+registo\_utente(U, I, L) :: (findall((U, I, L),

registo\_utente(U, I, L), R),

length(R, T), T == 1).

% Não podem existir fichas de utentes repetidas na mesma instituição com serviços e profissionais repetidos

+ficha\_utente(U, S, P, I) :: (findall((U, S, P, I),

ficha\_utente(U, S, P, I), L),

length(L, R), R == 1).

% Só pode inserir uma ficha de utente se existir a instituição

+ficha\_utente(U, S, P, I) :: (findall(I, instituicao(I), L),

length(L, R), R == 1).

% Só pode inserir uma ficha de utente se existir o serviço na dada instituição

+ficha\_utente(U, S, P, I) :: (findall((S, I), servico(S, I), L),

length(L, R), R == 1).

% Só pode inserir uma ficha de utente se existir o profissional do serviço na instituição

+ficha\_utente(U, S, P, I) :: (findall((P, S, I),

profissional(P, S, I), L),

length(L, R), R == 1).

% Só pode inserir uma ficha de utente se este estiver registado na base de conhecimento

+ficha\_utente(U, S, P, I) :: (findall(U, registo\_utente(U, I, L), R),

length(R, T), T == 1).

% Não podem existir serviços repetidos na mesma instituição

+servico(S, I) :: (findall((S, I), servico(S, I), L), length(L, R),

R == 1).

% Só pode inserir um serviço numa instituição se esta existir

+servico(S, I) :: (findall(I, instituicao(I), L), length(L, R),

R == 1).

% Não podem existir profissionais repetidos na mesma instituição com o mesmo serviço

+profissional(P, S, I) :: (findall((P, S, I),

profissional(P, S, I), L), length(L, R),

R == 1).

% Só pode inserir um profissional numa instituição se esta existir

+profissional(P, S, I) :: (findall(I, instituicao(I), L),

length(L, R), R == 1).

% Só pode inserir um profissional de um determinado serviço numa instituição se esse serviço existir na mesma

+profissional(P, S, I) :: (findall((S, I), servico(S, I), L),

length(L, R), R == 1).

% Não podem existir instituições repetidas

+instituicao(I) :: (findall(I, instituicao(I), L), length(L, R),

R == 1).

Por exemplo, no primeiro invariante é dito que não queremos registos de utente repetidos, ou seja, que todos os campos sejam iguais e, como tal, o *findall* irá fazer uma lista de tuplos com tudo o que encontrar acerca de registos de utentes na base de conhecimento e, de seguida, é calculado o tamanho dessa lista, através do *length*, sendo que, se depois da inserção de conhecimento, a lista só tiver um elemento, então a base de conhecimento será consistente, caso contrário o invariante falha e o facto terá de ser removido. O mesmo raciocínio é feito para todos os restantes invariantes.