Universidade do Minho

Licenciatura em Engenharia Informática

2011

SRCR

Grupo 8

Segmento 2

Ano Lectivo de 2010/2011

54745 **André Pimenta**
54808 **Cedric Pimenta**
54825 **Daniel Santos**
54738 **João Gomes**
54802 **Milton Nunes**

5 de Maio de 2011

Resumo

Neste relatório sobre o segmento 2 de Sistemas de Representação de Conhecimento e Raciocínio apresentaremos o trabalho desenvolvido nas várias funcionalidades que se pretendiam e a forma como chegamos à sua implementação. Explicaremos, também, os extras criados e as principais dificuldades que surgiram.

Conteúdo

Lista de Figuras

Capítulo 1

Introdução

1.1 Motivação e objectivos

Depois de efectuado com sucesso o segmento 1, pretendemos continuar neste segmento o trabalho e objectivos definidos no anterior. Pretendemos, portanto, desenvolver os conhecimentos leccionados nas aulas, de forma a concluir com sucesso os desafios propostos tendo sempre em atenção que o sistema de representação de conhecimento e raciocínio a desenvolver seja o mais aplicável à realidade possível.

1.2 Estrutura

O presente relatório é constituído por 5 capítulos. O primeiro, este, menciona a motivação e os objectivos do projecto, apresentando uma pequena introdução. No seguinte, apresentaremos os conhecimentos que achamos que o leitor deverá possuir para entender o projecto e a sua resolução, enquanto no **Capítulo III** explicaremos tudo o que foi desenvolvido durante o nosso trabalho. Por fim, no **Capítulo IV** apresentaremos a conclusão e interpretação dos resultados obtidos; e por fim, o **Capítulo V** conterá toda a bibliografia consultada.

1.3 Introdução

Neste segundo segmento de Sistemas de Representação de Conhecimento e Raciocínio pretende-se desenvolver um sistema que caracterize um universo de discurso farmacológico. Para tal, serão desenvolvidos uma série de funcionalidades, descritas e explicadas neste relatório, através do uso da programação em lógica. Neste segmento incide-se, sobretudo, no tratamento de informação incompleta. Isto é, no trabalho anterior existiam apenas duas respostas possíveis: verdadeiro e falso, tratava-se portanto de Programação Lógica, contudo agora lidamos com Programação Lógica Estendida. Em PLE existe para além do verdadeiro e falso, o desconhecido, ou seja, podemos estar perante uma situação em que a informação existente não permite que se possa ser conclusivo, e é essa possibilidade de resposta que a PLE oferece e que precisamos para realizar este segmento 2. Para além disto, teremos em atenção a manipulação da base de conhecimento restringindo a inserção e a remoção de conhecimento de forma a que esta se mantenha sempre coerente.

Capítulo 2

Preliminares

Neste capítulo vão ser apresentados alguns conceitos fundamentais para a elaboração deste trabalho e algumas das ferramentas fundamentais para a elaboração do mesmo.

2.1 Estudos Anteriores

Para a realização deste trabalho foram necessários e fundamentais vários conhecimentos sobre programação em lógica e PROLOG, conhecimentos estes que foram adquiridos ao longo das aulas teóricas e teorico-praticas da Unidade Curricular de Sistemas de Representação de Conhecimento e Raciocíocinio, e que já foram utilizados e expostos no trabalho prático anterior: *Segmento 1*.
Assim, a novidade neste trabalho prático passou a ser a compreensão e o uso da programação em lógica estendida no seguimento da representação de informação incompleta.
Para a melhor compreensão do trabalho que é apresentado neste relatório, vamos introduzir alguns conceitos novos que não foram expostos no trabalho anterior *Segmento 1* mas que são importantes para a compreensão deste.

2.2 Base de dados Vs Representação de Conhecimento

Os sistemas de Bases de Dados (BD's) e os sistemas de representação de conhecimento lidam com aspectos concretos do mundo real e podem-se comparar nos termos em que dividem a utilização da informação [Analide,2002].

Posto isto, apresentamos os pressupostos no qual se baseiam as linguagens de manipulação, tanto das bases de dados como dos sistemas de representação de conhecimento.

2.2.1 Base de Dados

Os pressupostos em que se baseiam as linguagens de manipulação de base de dados seguem o principio de que apenas existe e é válida a informação contida nesta. Assim sendo, apresentámos, então, os seguintes pressupostos:

- **Pressuposto do Mundo Fechado** – toda a informação que não mencionada na base de dados é considerada falsa;

- **Pressuposto dos Nomes Únicos** – duas constantes diferentes (que definam valores atómicos ou objectos) designam, necessariamente, duas entidades diferentes do universo de discurso;
- **Pressuposto do Domínio Fechado** – não existem mais objectos no universo de discurso para além daqueles designados por constantes na base de dados.

2.2.2 Sistema de Representação de Conhecimento

Porém, nem sempre se pretende assumir que a informação representada é a única que se pode considerar válida e que as entidades representadas sejam as únicas existentes. Posto isto, apresentámos, então, os seguintes pressupostos:

- **Pressuposto do Mundo Aberto** – podem existir outros factos ou conclusões verdadeiros para além daqueles representados na base de conhecimento;
- **Pressuposto dos Nomes Únicos** – duas constantes diferentes (que definam valores atómicos ou objectos) designam, necessariamente, duas entidades diferentes do universo de discurso;
- **Pressuposto do Domínio Aberto** – podem existir mais objectos do universo de discurso para além daqueles designados pelas constantes da base de conhecimento.

2.3 Representação de Informação Incompleta

Muitas vezes presenciámos situações onde a informação existente é insuficiente e/ou incompleta. Perante estas situações, é importante saber representá-las para além do verdadeiro ou falso; temos de ser capazes de demonstrar que a questão/situação em causa é incompleta/desconhecida e que o seu valor não pode ser definido no imediato, ao contrário do que acontece, na maior parte das situações, nas mais variadas linguagens de programação.
É aqui que a programação em lógica estendida se apresenta como capaz de resolver estas situações. Através da programação em lógica estendida seremos capazes de representar tais situações *desconhecidas*. A forma de representar o conhecimento é apresentada a seguir:

- **Verdadeiro**: quando for possível provar uma questão(Q) na base de conhecimento;
- **Falso**: quando for possível provar a falsidade de uma questão(-Q) na base de conhecimento;
- **Desconhecido**: quando não for possível provar a questãoQ nem a questão-Q.

De facto, para cumprir com os pressupostos definidos anteriormente, teremos de definir outro tipo de informação, além da informação positiva e a informação negativa. Esta pode ser representada da seguinte forma:

- **Negação por Falha**: quando não existe nenhuma prova. É representada pelo teorema *não* já usado, anteoriormente, no *Segmento 1*;
- **Negação Forte**: representada por (-) e que afirma que determinado predicado é falso.

Por fim, atendendo aos requisitos do trabalho, será ainda necessário obdecer às seguintes caracteristicas:

- Representar casos de conhecimento imperfeito, pela utilização de valores nulos de todos os tipos estudados.
- Manipular invariantes que designem restrições de inserção e remoção de conhecimento do sistema.
- Lidar com a problemática da evolução do conhecimento, criando os procedimentos adequados.
- Desenvolver um sistema de inferência capaz de implementar os mecanismos de raciocínio inerentes a estes sistemas.

Capítulo 3

Descrição do Trabalho e Análise de Resultados

Neste capítulo vamos apresentar os passos do desenvolvimento deste segmento, acompanhados pelas explicações dos mesmos. Faremos, também, uma breve análise dos resultados obtidos.

3.1 Base de Conhecimento

A Base de Conhecimento é constituída por um conjunto de axiomas acerca de um determinado tema. Durante a realização deste projecto, foi necessário desenvolver uma base de conhecimento, onde são apresentados várias localidades **localidade(nome)), hospitais hospital(nome)**, onde apresentámos as farmácias de uma localidade ou hospital **farmacia(nome,localidade/hospital)**, aplicações clínicas **appClinica(doenca)**, estatutos **estatuto(tipo)**, as datas dos medicamentos **data(medicamento, ano, mes, validade meses)**, os medicamentos **Medicamento(nome, doença, especialidade, principio activo)**, e a disponibilidade e preços dos mesmos **Disponibilidades(farmacia, medicamento, preco publico, preco reformado)**.
De seguida, apresentámos as funcionalidades que desenvolvemos para pesquisas na base do conhecimento.

3.2 Funcionalidades Básicas

Nesta secção serão apresentados e explicados os métodos usados para a resolução das funcionalidades básicas propostas.

3.2.1 Representação de Conhecimento Positivo e Negativo

A representação do conhecimento positivo é feito nos factos, por exemplo:

```
farmacia(central,'S. Marcos').
farmacia(oliveira,'perelhal').
farmacia(nova,'braga').
```

Figura 3.1: Conhecimento positivo

Isto significa que existe a farmácia oliveira na localidade do perelhal. O conhecimento negativo é representado por duas formas, ou utilizando a negação por falha ou a negação forte:

```
-farmacia(Nome,Sitio) :-
   nao(farmacia(Nome,Sitio)),
        nao(excepcaoFarm(Nome,Sitio)).
```

Figura 3.2: Negação por falha

```
-farmacia(peralhal,'braga').
```

Figura 3.3: Negação forte

3.2.2 Representação de Casos de Conhecimento Imperfeito, pela utilização de valores nulos de todos os tipos

Os valores nulos surgem como uma estratégia para a enumeração de casos, para os quais se pretende fazer a distinção entre situações em que as respostas a questões deverão ser concretizadas como conhecidas (verdadeiras ou falsas) ou desconhecidas.
Assim, existem 3 tipos de valores nulos: do tipo desconhecido, do tipo desconhecido de um conjunto dado de valores e do tipo não permitido.
Para o caso das aplicações clínicas usamos o axioma **farmacia(nome,localidade/hospital)**, no entanto, para a representação das farmácias cujo local não seja conhecido na base do conhecimento, utilizámos os valores nulos do tipo desconhecido:

```
farmacia(pimentel,desc).
excepcaoFarm(Nome,Sitio) :-
        farmacia(Nome,desc).

-farmacia(Nome,Sitio) :-
   nao(farmacia(Nome,Sitio)),
        nao(excepcaoFarm(Nome,Sitio)).
```

Figura 3.4: Valor Nulo tipo 1

Para o caso em que o medicamento possa ter a data *'10-2003'* ou a data *'10-2004'* usámos os valores nulos do tipo desconhecido de um conjunto de valores:

```
data(epizen, 1997, 01, 12).
data(relistor, 1997, 02, 18).
data(tobi, 2002, 10, 24).

        %VALOR NULO TIPO II
        %Nao se sabe se o medicamento rennan foi lançado em 2003 ou 2004
excepcaoDT(rennan, 2003, 10, algum).
excepcaoDT(rennan, 2004, 10, algum).

        %VALOR NULO TIPO II
        %Nao se sabe a data de lancamento do kineret. Apenas se conhece a decada
excepcaoDT(kineret,A,M,V) :-
        A >= 1980, A =< 1990, M >= 1, M =< 12.


-data( D,A,M,V ) :-
   nao(data( D,A,M,V )),
        nao(excepcaoDT( D,A,M,V )).
```

Figura 3.5: Valor Nulo tipo 2

Para o tipo não permitido, usámos o exemplo em que não se permite conhecer o medicamento que trate o cancro:

```
medicamento(algum,cancro,alguma,algum).
excepcaoMedic(M,D,E,PA) :-
        medicamento(algum,D,alguma,algum).
nulo(cancro).


-medicamento( N,A,S,PA ) :-
   nao(medicamento( N,A,S,PA )),
        nao(excepcaoMedic( N,A,S,PA )).
```

Figura 3.6: Valor Nulo tipo 3

3.2.3 Manipulação de Invariantes que designem restrições à inserção e à remoção de conhecimento

Quando queremos remover ou inserir novos factos na base de conhecimento temos de ter em atenção se nada depende deles, ou se eles já não se encontram presentes, respectivamente. No nosso caso em particular, antes de tentar remover um hospital, usámos o invariante abaixo apresentado que testa se não existe nenhuma farmácia associada a esse mesmo hospital:

```
%Só podemos remover um hospital se este não se encontrar associado
-hospital(Hosp) :: (nao(farmacia(_,Hosp))).
```

Figura 3.7: Remover Hospital

No caso de pretender-mos inserir um hospital, usámos o invariante abaixo para nos certificarmos que o hospital ainda não consta na base do conhecimento:

```
%Só podemos adicionar um hospital se este ainda não existe
+hospital(Hosp) :: (findall(Hosp, (hospital(Hosp)), S),
                                    comprimento(S,N), N==1).
```

Figura 3.8: Adicionar Hospital

Assim, este mesmo procedimento demonstrado para manter a base de conhecimento coerente, foi aplicado a todos os factos e axiomas presentes no projecto.

3.2.4 Tratamento da problemática da evolução do conhecimento, pela criação dos procedimentos adequados

O tratamento da problemática da evolução do conhecimento prende-se com o facto de ao inserir ou remover conhecimento na base do conhecimento, deixar essa base coesa e fidedigna. Para que isto aconteça, não podemos apagar informação que seja uma dependência de outra, ou inserir informação repetida; assim, aquando da alteração da informação da base de conhecimento, teremos que testar se não irá corromper a base do conhecimento. De forma a garantir esta coerência, usámos os invariantes para inserção e remoção já falados acima, nas funções de *evolução* e *remover*:

```
evolucao( Termo ) :-
   solucoes( Invariante,+Termo::Invariante,Lista ),
        insercao( Termo ),
        teste( Lista ).

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

Figura 3.9: Função de evolução

```
remover( Termo ):-
   findall( Inv,-Termo::Inv,LInv),
        remocao(Termo),
        teste(LInv).

remocao(Termo):-
   retract(Termo).
remocao(Termo):-
        assert(Termo),!,fail.
```

Figura 3.10: Função de remover

3.2.5 Desenvolvimento de um sistema de inferência capaz de implementar os mecanismos de raciocínio inerentes a estes sistemas

O pretendido neste ponto que implemente mos um mecanismo, interpretador de questões, que mediante uma questão obtenhamos uma de três respostas possíveis: Verdadeiro, Falso ou Desconhecida.
O conceito utilizado foi o do PMF (Principio do Mundo Fechado), ou seja, toda a informação que não exista mencionada na base de dados é considerada falsa.
O interpretador dada uma questão caso se possa concluir a veracidade da questão na base de dados a resposta será 'verdadeiro', caso esta esteja presente como uma negação forte na base de dados a resposta será 'falso', e para o caso não esteja presente na base de dados como verdadeira, nem esteja negada fortemente esta terá como resposta 'desconhecido'. Desta forma conseguimos um programa em lógica estendido.

```
demo(Questao, verdadeiro) :-
        Questao.
demo(Questao, falso) :-
        -Questao.
demo(Questao, desconhecido) :-
        nao( Questao ),
        nao( -Questao ).

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

Figura 3.11: Demo

3.3 Funcionalidades extra

Findo as funcionalidades básicas exigidas, decidimos criar extras que achamos que seriam importantes e que trariam funcionalidades úteis para o nosso programa.
Foram implementados quatro. O primeiro extra por nos criado tem como finalidade encontrar o medicamento para uma determinada doença, em seguida decidimos criar uma função para encontrar o medicamento com o maior prazo de validade para uma determinada doença.
Após a criação de dois extras referentes a medicamento para doenças, decidimos focar nos na

parte das farmácias, onde criamos mais dois extras, um que determina a farmácia que possui o medicamento ao preço mais baixo, e uma outra que lista as aplicações clínicas disponíveis numa determinada farmácia.

Capítulo 4

Conclusão

A utilização de Programação Lógica Estendida surgiu como o principal desafio neste segmento. Depois de estudarmos ao pormenor e praticar, visto que possuíamos apenas alguns conceitos leccionados nas aulas e não estavamos muito à vontade neste tipo de implementação, conseguimos com relativa facilidade perceber e utilizar a PLE para realizar os desafios propostos no enunciado desta etapa. Conseguimos implementar, com sucesso, todas as funcionalidades pretendidas e também alguns extras como era sugerido, contudo tivemos algumas dificuldades na criação de exemplos relacionados com a realidade para a criação da base de conhecimento. Para além disso, a outra grande dificuldade que enfrentamos baseou-se nos extras, não propriamente no seu desenvolvimento, mas na decisão do que seria necessário implementar de forma a tornar o nosso segmento ainda mais útil a um utilizador.

Bibliografia

4.1 Referências Bibliográficas

[Analide, 2011]ANALIDE, Cesar, NOVAIS, Paulo, NEVES, José,
"Sugestões para a Elaboração de Relatórios",
Relatório Técnico, Departamento de Informática, Universidade do Minho, Portugal, 2011.

[Bratko, xxxx]BRATKO, Ivan,
"PROLOG: Programming for Artificial Intelligence",
E.Kardelj University, J.Stefan Institute Yugoslavia, 1986.

[Analide, xxx]ANALIDE, Cesar, NEVES, José,
"Representação de Informação Incompleta"

[Analide, 2002]ANALIDE, Cesar, NOVAIS, Paulo, NEVES, José,
"Representação de Informação Incompleta (slides)",

4.2 Referências WWW

"http://pt.wikibooks.org/wiki/Prolog/Listas",
10 de Abril de 2011.

"http://pt.wikibooks.org/wiki/Prolog/Comando_Fail_ e _ a_ combinação_ Cut/Fail",
09 de Abril de 2011.

Capítulo 5

Anexos

5.1 Elementos do Grupo

Figura 5.1: André Pimenta

Figura 5.2: Cedric Pimenta

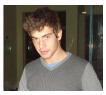

Figura 5.3: Daniel Santos

Figura 5.4: João Miguel

Figura 5.5: Milton Nunes

5.2 Codigo

```
:- set_prolog_flag( single_var_warnings,off ).
:- set_prolog_flag( discontiguous_warnings,off ).
:- op( 900,xfy,'::' ).

:- dynamic localidade/1.
:- dynamic hospital/1.
:- dynamic farmacia/2.
:- dynamic appClinica/1.
:- dynamic medicamento/4.
:- dynamic estatuto/1.
:- dynamic data/4.
:- dynamic medicDisponivel/4.
:- dynamic excepcaoAppClinica/1.
:- dynamic excepcaoFarm/2.
:- dynamic remove/2.

%---------------------------------
%               Hospital(nome)
%---------------------------------
hospital('S. Marcos').


%---------------------------------
%               Local(nome)
%---------------------------------
localidade('perelhal').
localidade('braga').
localidade('barcelos').


%---------------------------------
%               Farmacia(nome,localidade/hospital).
%---------------------------------
farmacia(central,'S. Marcos').
farmacia(oliveira,'perelhal').
farmacia(nova,'braga').
-farmacia(peralhal,'braga').

        %VALOR NULO TIPO I
        %Caso em que não se sabe em que sitio está a farmacia pimentel
farmacia(pimentel,desc).
excepcaoFarm(Nome,Sitio) :-
        farmacia(Nome,desc).

-farmacia(Nome,Sitio) :-
   nao(farmacia(Nome,Sitio)),
        nao(excepcaoFarm(Nome,Sitio)).


%---------------------------------
%               Aplicaçao Clinica
%---------------------------------
appClinica(constipacao).
appClinica(alergia).
appClinica(calvicie).

-appClinica( A ) :-
   nao(appClinica(A)),
   nao(excepcaoAppClinica(A)).


%---------------------------------
%       Estatuto(tipo)
%---------------------------------
estatuto(reformado).
estatuto(publico).
```

```
        %CONHECIMENTO NEGATIVO
-estatuto(jovem).
-estatuto(pensionista).

        %Representacao de informacao negativa pelo pressoposto do mundo fechado -- NEGACAO POR FALHA NA
PROVA
-estatuto(E) :-
        nao(estatuto(E)).


%---------------------------------
%       Data( medicamento, ano, mes, validade meses)
%---------------------------------
data(epizen, 1997, 01, 12).
data(relistor, 1997, 02, 18).
data(tobi, 2002, 10, 24).

        %VALOR NULO TIPO II
        %Nao se sabe se o medicamento rennan foi lançado em 2003 ou 2004
excepcaoDT(rennan, 2003, 10, algum).
excepcaoDT(rennan, 2004, 10, algum).

        %VALOR NULO TIPO II
        %Nao se sabe a data de lancamento do kineret. Apenas se conhece a decada
excepcaoDT(kineret,A,M,V) :-
        A >= 1980, A =< 1990, M >= 1, M =< 12.


-data( D,A,M,V ) :-
   nao(data( D,A,M,V )),
        nao(excepcaoDT( D,A,M,V )).


%---------------------------------
%     Medicamento(nome,doença,especialidade,principio activo)
%---------------------------------
medicamento(relistor,constipacao,hematologia,methylnaltrexone).
medicamento(epizen,alergia,pneumologia,ephinephrine).
medicamento(tobi,constipacao,hematologia,tobramycin).
medicamento(rennan,alergia,pneumologia,rennamycin).
medicamento(kineret,reumatismo,ortopedia,anakinra).
medicamento(cenas,alergia,cenas,cenas).

        %VALOR NULO TIPO 1
        %E desconhecido medicamento para a calvice
medicamento(algum,calvicie,algum,algum).
excepcaoMedic(M,D,E,PA) :-
        medicamento(algum,D,algum,algum).

        %VALOR NULO TIPO III
        %Caso em que não se permite conhecer o medicamento que trate o cancro
medicamento(algum,cancro,alguma,algum).
excepcaoMedic(M,D,E,PA) :-
        medicamento(algum,D,alguma,algum).
nulo(cancro).


-medicamento( N,A,S,PA ) :-
   nao(medicamento( N,A,S,PA )),
        nao(excepcaoMedic( N,A,S,PA )).


%EXTRA 1
%Encontra medicamentos para determinada doenca
medicParaDoenca(D,M) :-
        findall( (N,D,E,PA), (medicamento(N,D,E,PA), nao(excepcaoMedic(algum,D,algum,algum))), M ),
                        comprimento(M,T), T>=1.
```

```
-medicParaDoenca(D,M) :-
        nao(medicamento(N,D,E,PA)),
        nao(excepcaoMedic(algum,D,algum,algum)).


%EXTRA 2
%Encontra o medicamento com mais validade para uma doenca
medicDoencaMaiorValidade(D,Med) :-
        findall( (N,D,Validade),
                                (medicamento(N,D,E,PA), nao(excepcaoMedic(algum,D,algum,algum)),
                                 data(N,Ano,Mes,Validade), nao(excepcaoDT( N,_,_,_))),
                                R ),
        comprimento(R,T), T>=1,
        maiorValid(R,Med).

-medicDoencaMaiorValidade(D,Med) :-
        nao(medicDoencaMaiorValidade(D,Med)),
        nao(excepcaoMedic(_,D,_,_)),
        findall( (N,D), (medicamento(N,D,E,PA), excepcaoDT(rennan,Ano,Mes,Valid)), R),
        comprimento(R,T), T==0.


maiorV((N1,D1,V1),(N2,D2,V2),(N1,D1,V1)):- V1>=V2.
maiorV((N1,D1,V1),(N2,D2,V2),(N2,D2,V2)):- V2>V1.

maiorValid([R],R).
maiorValid([(N,D,Validade)|T],R):-
        maiorValid(T,R1),
        maiorV((N,D,Validade),R1,R).


%--------------------------------
%        Disponibilidades(farmacia, medicamento, preco publico, preco reformado)
%--------------------------------
medicDisponivel(central, relistor, 12.5, 3.5).
medicDisponivel(oliveira, tobi, 6.5, nulo).
medicDisponivel(central, tobi, 7.5, 2.3).
medicDisponivel(oliveira, kineret, 20.3, 15.3).
medicDisponivel(central, epizen, 10.3, 7.3).
medicDisponivel(central, kineret, 5.3, 2.3).

        %Desconhece-se se existe epizen na farmacia central ou na nova
excepcaoMDisp(central,epizen, algum, algum).
excepcaoMDisp(nova,epizen, algum, algum).
        %Desconhece-se se existe rennan na farmacia nova
excepcaoMDisp(nova,rennan, algum, algum).

        %A farmacia central nao disponibiliza medicamentos para a calvice --CONHECIMENTO NEGATIVO
-medicDisponivel(central,M,D,PP,PR) :-
        medicamento(M,calvice,E,PA).

-medicDisponivel( F,M,PP,PR ) :-
   nao(medicDisponivel( F,M,PP,PR )),
        nao(excepcaoMDisp(F,M,algum,algum)).

%EXTRA 3
%em que farmacia se encontra o medicamento pretendido ao melhor preco
medicMaisBaratoPublic(M,R) :-
        findall((M,F,PP), (medicDisponivel(F,M,PP,PR)), S),
                                                        comprimento(S,N), N>=1,
                                                        menorL(S,R).

-medicMaisBaratoPublic(M,R) :-
        nao(medicDisponivel( F,M,PP,PR )),
        nao(excepcaoMDisp(F,M,algum,algum)).
```

```
menor((F1,PP1),(F2,PP2),(F1,PP1)):- PP1=<PP2.
menor((F1,PP1),(F2,PP2),(F2,PP2)):- PP2<PP1.

menorL([(M,F,PP)],(F,PP)).
menorL([(M,F,PP)|T],R):- menorL(T,R1), menor((F,PP),R1,R).


%EXTRA 4
%Que aplicacoes clinicas sao disponibilizadas por uma farmacia
appClinicasFarmacia(F,R) :-
        findall( (D), (medicDisponivel(F,N,PP,PR), medicamento(N,D,E,PA)), R1),
        comprimento(R1,T), T>=1,
        removeRep(R1,R).

-appClinicasFarmacia(F,R) :-
        nao(appClinicasFarmacia(F,R)),
        nao(excepcaoMDisp(F,_,_,_)).

remove(_,[],[]).
remove(X,[X|T],R1) :-
        remove(X,T,R1).
remove(X,[Y|T],[Y|R1]) :-
        remove(X,T,R1).

removeRep([],[]).
removeRep([H|T],[H|R]) :-
        remove(H,T,R1),
        removeRep(R1,R).


%---------------------------------
%%Funcoes de insercao
%---------------------------------
evolucao( Termo ) :-
   solucoes( Invariante,+Termo::Invariante,Lista ),
        insercao( Termo ),
        teste( Lista ).

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

remover( Termo ):-
   findall( Inv,-Termo::Inv,LInv),
        remocao(Termo),
        teste(LInv).

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


teste( [] ).
teste( [R|LR] ) :-
   R,
   teste( LR ).

solucoes(X,Y,Z) :-
        findall(X,Y,Z).

comprimento([],0).
comprimento([X|XS],R) :-
        comprimento(XS,R1),
        R is R1+1.
```

```
%--------------------------------
%%Funções auxiliares utilizadas
%--------------------------------

demo(Questao, verdadeiro) :-
        Questao.
demo(Questao, falso) :-
        -Questao.
demo(Questao, desconhecido) :-
        nao( Questao ),
        nao( -Questao ).

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



%%Manipulação de invariantes que designem restrições à inserção e à remoção de conhecimento

%Só podemos remover um hospital se este não se encontrar associado
-hospital(Hosp) :: (nao(farmacia(_,Hosp))).

%Só podemos remover uma localidade se este não se encontrar associado
-localidade(Loc) :: (nao(farmacia(_,Loc))).

%Só podemos remover uma farmácia se este não se encontrar associado
-farmacia(Nome,_) :: (nao(medicDisponivel(Nome,_,_,_)),
                                        nao(excepcaoMDisp(Nome,_,_,_))).

%Só podemos remover uma aplicação clínica se este não se encontrar associado
-appClinica(Ac) :: (nao(medicamento(_,Ac,_,_))).

%Só podemos remover um medicamento se este não se encontrar associado
-medicamento(Med,_,_,_) :: (nao(data(Med,_,_,_)),
                                                        nao(excepcaoDT(Med,_,_,_)),
                                                        nao(medicDisponivel(_,Med,_,_)),
                                                        nao(excepcaoMDisp(_,Med,_,_))).

%Só podemos adicionar um hospital se este ainda não existe
+hospital(Hosp) :: (findall(Hosp, (hospital(Hosp)), S),
                                        comprimento(S,N), N==1).

%Só podemos adicionar uma localidade se este ainda não existe
+localidade(Loc) :: (findall(Loc, (localidade(Loc)), S),
                                    comprimento(S,N), N==1).

%Só podemos adicionar uma farmácia se o nome ainda não existe e o localidade/hospital existe
+farmacia(Nome,LH) :: (findall((Nome,LH), (farmacia(Nome,LH)), S),
                                            comprimento(S,N), N==1,
                                            localidade(LH);
                                            hospital(LH)).

%Só podemos adicionar uma aplicação clínica se esta ainda não existe
+appClinica(Ac) :: (findall(Ac, (appClinica(Ac)), S),
                                        comprimento(S,N), N==1).

%Só podemos adicionar um medicamento se o nome é único e a aplicação clínica existe
+medicamento(Nome,Ac,X,Z) :: (findall((Nome,Ac,_,_), (medicamento(Nome,Ac,_,_)), S),
                                                        comprimento(S,N), N==1,
                                                        appClinica(Ac),
                                                        nao(nulo(Ac))).
```

```
%Só podemos adicionar um estatuto se este ainda não existe
+estatuto(Tipo) :: (findall(Tipo, (estatuto(Tipo)), S),
                                        comprimento(S,N), N==1).

%Só podemos associar uma data a um medicamento se este existe e a data que se pretende associar, ainda
não está associada a este
+data(Med,A,M,MV) :: (findall((Med,A,M,MV), (data(Med,A,M,MV)), S),
                                            comprimento(S,N), N==1,
                                            medicamento(Med,_,_,_)).

%Só podemos associar um medicamento a um farmácia se ambos existem e ainda não se encontram associados
+medicDisponivel(Farm,Med,_,_) :: (findall((Farm,Med,_,_), (medicDisponivel(Farm,Med,_,_)), S),
                                                                        comprimento(S,N), N==1,
                                                                        farmacia(Farm,_),
                                                                        medicamento(Med,_,_,_)).
```