Universidade do Minho

Mestrado Integrado em Engenharia Biomédica

Programação em Lógica, Conhecimento e Raciocínio

Trabalho realizado por:

Fábio França 61785

Filipe Fernandes 61754

Docente:

Professor José Maia Neves

Professor César Rodrigues

Novembro de 2013

Resumo

Este trabalho encontra-se englobado no âmbito da representação de conhecimento imperfeito, recorrendo à utilização de valores nulos e da criação de mecanismos de raciocínio adequados.Para tal foi desenvolvido um sistema de representação de conhecimento e raciocínio com capacidade para caracterizar um universo de discurso de âmbito farmacêutico.

Conteúdo

1 Introdução

No âmbito da Unidade Curricular Programação em Lógica, Conhecimento e Raciocínio foi incentivada a representação de conhecimento imperfeito, recorrendo à utilização de valores nulos e da criação de mecanismos de raciocínio adequados. Por forma a demonstrar tais pressupostos, foi desenvolvido um sistema de representação de conhecimento e raciocínio com capacidade para caracterizar um universo de discurso de âmbito farmacêutico.

2 Preliminares

Considerando a Programação em Lógica, uma qustão ?P feita à base de dados pode ser provada ou não (sendo assim verdadeira ou falsa). No entanto, a programação em lógica clássica é baseada nalguns pressupostos que impoêm limitações ao tipo de processamento necessário para tratar informação incompleta. Alguns desses pressupostos são:

- Mundo Fechado - toda a informação que não se encontra na base de dados é falsa;
- Dominio Fechado - todos os objectos no universo de discurso estão representados na base de dados.

Sistemas reais, no entanto, não trabalham com estes pressupostos, podendo beneficiar largamente de abordagens que evitem estas limitações. Ao adicionar capacidade para representação e raciocínio sobre informação incompleta a um sistema, a sua base de dados passa a poder descrever o mundo real de forma muito mais correcta.

Um outro ponto de acção é a representação de informação negativa explicita. Ao contrário de assumir que o que não se conhece é falso (i.e., mundo fechado), representa-se na base de dados o conhecimento de algo que sabemos ser falso. sendo assim, a base de dados tem duas partes: o que sabemos ser verdadeiro, e o que sabemos ser falso. tudo o resto é desconhecido.

Em termos de programação em lógica isso pode ser visto por:

demo(P, true) ¬ P
demo(P, false) ¬ ØP
demo(P, unknown)

Onde P é o predicado a provar, e o simbolo Ø representa 'negação explicita'. Há no entanto situações onde o mundo fechado faz sentido. Uma vez que não se pretende assumir que a informação representada é a única que é válida, e, muito menos, que as entidades representadas sejam as únicas existentes no mundo exterior, surgem novos pressupostos ao qual se baseia um sistema de representação de conhecimento.

- Mundo Aberto – podem existir outros factos ou conclusões verdadeiros para além daqueles representados na base de conhecimento;
- Domínio Aberto – podem existir mais objectos do universo de discurso para além daqueles designados pelas constantes da base de conhecimento.

A generalidade dos programas escritos em lógica representa implicitamente a informação negativa, assumindo a aplicação do raciocínio segundo o PMF(Pressuposto do Mundo Fechado). A extensão de um programa em lógica passa a contar com dois tipos de negação: a negação por falha, característica dos programas em lógica tradicionais, representada pelo termo não, e a negação forte ou clássica, como forma de identificar informação negativa, ou falsa, representada pela conectiva '¬'.

Podemos distinguir os dois tipos de negação a partir de um exemplo oportuno apresentado por *John McCarthy*(pioneiro de Inteligência Artificial:

- **Negação por falha na prova**

$$\text{atravessar} \leftarrow \text{não(automóvel)}.$$

- **Negação forte**

$$\text{atravessar} \leftarrow \neg \text{ automóvel}.$$

No primeiro caso, o peão atravessará a estrada se não existir prova de que o automóvel vai passar. Ao invés, no caso da negação forte o peão apenas atravessará na existência de prova de que o veículo não vai passar. Neste contexto, passa a ser possível distinguir situações que são falsas de situações para as quais não existe prova de que sejam verdadeiras.

De seguida faz-se uma abordagem a diferentes tipos de valores que podem estar presentes em sistemas onde se verifiquem situações de informação incompleta, designados por valores nulos. São considerados três géneros de valores nulos: do tipo desconhecido, do tipo desconhecido mas de um conjunto determinado de valores e do tipo não permitido(interdito).

Para um predicado p, representa-se um valor nulo vi do tipo desconhecido, como uma excepção a ¬p:

$$\neg \text{p(X)} \leftarrow \text{não excepçãop(X)}$$

,concretizando, de seguida, as excepções que se pretendem definir como valores nulos:

$$\begin{gathered}\text{excepçãop(X)} \leftarrow \text{p(v1)} \\ \text{excepçãop(X)} \leftarrow \text{p(v2)} \\ \ldots\end{gathered}$$

Considera-se então, que a identificação de valores nulos do tipo desconhecido será feita através da introdução de uma situação de excepção correspondendo a uma condição anómala, na cláusula de formalização do PMF.

Assim inclui-se a excepção no corpo da cláusula de formalização do PMF que já fazia parte do programa em lógica, correspondendo à forma genérica:

$$\neg \text{p(X)} \leftarrow \text{não p(X)} \wedge \text{não excepçãop(X)}$$

No caso de um valor nulo do tipo Desconhecido, de um Conjunto Dado de Valores, os *vi* representam os valores nulos do conjunto de valores possíveis existentes, sendo valores atómicos que concretizam explicitamente todas as possibilidades de representação dos valores nulos.

Um valor do tipo Não permitido ou Interdito, além de verificar um valor desconhecido, representa um valor que não é permitido especificar ou conhecer, e qualquer tentativa posterior de concretizar esse valor deverá ser rejeitada como sendo provocadora de inconsistência na informação constante da base de conhecimento.

3 Descrição do Trabalho e Análise de Resultados

3.1 Descrição do Trabalho

Para a realização do trabalho,pretende-se construir um caso prático de aplicação dos conhecimentos, capz de demonstrar as funcionalidades subjacentes à programação em lógica estendida e à representação de conhecimento imperfeito, recorrendo à temática dos valores nulos.

Desta forma no desenvolvimento de um sistema de representação de conhecimento e raciocínio com capacidade para caracterizar um universo de discurso de âmbito farmacêutico, o principal objetivo centrou-se no cumprimento dos requesitos impostos. No entanto, procurou-se gerar um universo de âmbito farmacêutico com informação relevante e consistência.

3.1.1 Descrição do caso prático

Tal como referido pretende-se implementar um sistema de representação de conhecimento e raciocínio com capacidade de caracterizar um universo de discurso de âmbito farmacêutico. De tal forma, são várias as funcionalidades que o programa em Prolog consegue descrever:

- Inserção na base de medicamentos;
- Obter o Tipo de medicamento dado o seu nome, e.g., o medicamento Ben-u-ron pertence aos analgésicos;
- Aplicação ou aplicações clínicas de um medicamento, e.g. Cegripe é apropriado para Síndromes gripais e Constipações;
- Príncipio activo do Medicamento;
- Apresentação ou apresentações farmacêuticas de um medicamento, e.g Ben-u-ron pode ser vendido em comprimidos ou xarope;
- Preço de um medicamento, com distinção entre o preço recomendado, público e para regime especial;
- Contra Indicações e problemas de sáude específicos que podem advir da utilização de um medicamento;
- Data de validade e de introdução de um medicamento no mercado;
- Armazém de medicamentos, sendo cada um destes específico de um tipo de medicamento;

Como expectável, para além dos predicados necessários para descrever o universo de âmbito farmacêutico e , recorreu-se a predicados auxiliares tais como: meta-predicado

Demo, soluções, comprimento, remoção ou inserção de conhecimento. Outro aspecto de especial importância é o desenvolvimento de invariantes por forma a manter a consistência da base de conhecimento, designando restrições à inserção e à remoção de conhecimento do sistema. Pretende-se assim representar conhecimento positivo e negativo bem como casos de conhecimento imperfeito, pela utilização de valores nulos dos 3 tipos estudados.

4 Representação do Conhecimento e Mecanismos de Raciocínio

Como ponto de partida para o desenvolvimento e representação do conhecimento em Prolog, considera-se relevante a explicação do meta predicado Demo.

4.1 Meta-Predicado Demo

```
%-------------------------------- - - - - - - - - - - -  -  -  -  -   -
% Extensao do meta-predicado demo: Questao,Resposta -> {V,F,D}

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

O meta predicado demo faz parte de um sistema de inferência capaz de implementar um mecanismo de raciocínio para a programação em lógica estendida. Este mecanismo suporta como enunciado três tipos de resposta: verdadeiro, falso e desconhecido. Este é constituído por 2 argumentos, sendo que o primeiro a questão colocada ao sistema e o segundo a resposta que pode tomar os três valores {V,F,D}.A primeira cláusula indica que a resposta a uma questão (Questão) tem valor "verdadeiro"se for possível desenvolver uma prova de (Questão) a partir da informação positiva existente na Base de conhecimento e não sendo esta uma excepção. A segunda cláusula do meta predicado Demo define que a resposta a uma Questão toma o valor "falso"se for possível provar ¬(Questa˜o). A terceira cláusula do predicado define que a resposta a uma questão apresenta o valor "desconhecido"se não existir prova de (Questão) nem de ¬(Questa˜o), desta forma se a (Questão) não estiver presente na Base de conhecimento do Sistema, tal como ¬(Questa˜o). Assim sendo o valor de (Questão) será "desconhecido"se esta for uma excepção.

4.2 Predicado medicamento

Após esclarecido o predicado capaz de implementar o mecanismo de raciocínio para a programação em lógica estendida, desenvolveu-se a base de suporte para o universo farmacêutico. Desta forma, inseriu-se à base de conhecimento um conjunto de medicamentos, tal como apresentado:

```
medicamento('ben-u-ron').
medicamento('buscopan').
```

```
medicamento('brufen').
medicamento('cegripe').
medicamento('neosoro').
medicamento('microvlar').
medicamento('plavix').
```

Importa referir, que sendo este o sustento da Base de Conhecimento apenas se considerou relevante a inserção de conhecimento positivo não incorporando exceções para Medicamentos. Por forma a manter a consistência da Base de Conhecimento, criou-se invariantes para a inserção e remoção de conhecimento do sistema, tal como apresentado:

```
%---------------Invariantes

%---------------  Não permitir a inserção de conhecimento repetido

+medicamento(M) :: (solucoes( (M),(medicamento(M)),S),
comprimento(S,N), N==1).


%--------------- Não permitir a remoção de um medicamento se existirem dependências
sobre este

-medicamento(M) :: (-tipo_medicamento(M, T), -aplicacao_clinica(M, A),
 -principio_activo(M,P),
-apresentacao_farmaceutica(M,Ap), -preco_recomendado(M,A,P), -preco_publico(M,A,P),
-regime_especial(M,A,E,P), -contra_indicacao(M,C), -data_validade(M,Ap,Dia,Mes,Ano),
 -data_introducao(M,Ap,Dia,Mes,Ano)).


% Conhecimento Negativo

-medicamento(M) :- nao(medicamento(M)), nao(excepcao(medicamento(M))).

%-------------------------------- - - - - - - - - - - -  -  -  -  -   -
```

4.3 Tipo_medicamento

Após a inserção de medicamentos na Base de conhecimento, considerou-se relevante a designação do tipo de medicamento. Desta forma, implementou-se o predicado tipo_medicamento composto por dois argumentos, o primeiro contendo o nome do medicamento e o segundo o respetivo tipo. De referir que foi utilizado um valor nulo do

tipo interdito para o medicamento Cegripe, tendo sido criada a correspondente exceção a baixo apresentada.

```
% Extensao do predicado tipo_medicamento: Medicamento,Tipo-> {V,F,D}

tipo_medicamento('ben-u-ron','analgesico').
tipo_medicamento('buscopan','analgesico').
tipo_medicamento('brufen','anti-inflamatorio').
tipo_medicamento('cegripe','tipo_nulo').
tipo_medicamento('neosoro','descongestionante').
tipo_medicamento('microvlar','contraceptivo').
tipo_medicamento('plavix','antiplaquetar').

% Excepções

excepcao(tipo_medicamento(M,T)) :- tipo_medicamento(cegripe, tipo_nulo).
```

Como observável, apenas foi considerada a utilização do valor nulo interdito para o medicamento cegripe, uma vez que o tipo de medicamento é informação considerada relevante e indubitável na definição do mesmo.

Como necessário em todos os predicados diretamente relacionados com a definicção do universo farmacêutico, definiu-se um conjunto de invariantes por forma a manter a consistência da Base de Conhecimento. Assim sendo, utilizou-se os seguintes invariantes, bem como a definição de conhecimento negativo:

```
% Invariantes

%---------------Medicamento tem de existir

+tipo_medicamento(M,T) :: medicamento(M).

%---------------  Não permitir a inserção de conhecimento repetido

+tipo_medicamento(M,T) :: (solucoes( (M,T),(tipo_medicamento(M,T)),S),
comprimento(S,N), N==1).

%---------------     Não permitir a inserção de conhecimento relativo ao cegripe

+tipo_medicamento(M, T) :: (solucoes(X, (tipo_medicamento(cegripe, X), nao(nulo(X))))

% Conhecimento Negativo

-tipo_medicamento(M,T) :- nao(tipo_medicamento(M,T)), nao(excepcao(tipo_medicamento(
```

4.4 Aplicação_clínica

De seguida, procorou-se dotar a base de conhecimento acerca das possíveis aplicações clínicas de cada medicamento. Para tal foi utilizado o predicado aplicacao_clinica composto por dois argumento, Medicamento e respetiva aplicação. Além da inserção de conhecimento positivo recorreu-se à utlização do valor nulo Desconhecido (Incerto) para o medicamento Neosoro bem como conhecimento negativo relativo a este medicamento. Com base nestas definições não se sabe qual ou quais as aplicações clínicas do medicamento neosoro, apenas se sabendo que este não é adequado para Traumatismos, Febre ou Síndromes gripais.

```
% Extensao do predicado aplicacao_clinica: Medicamento, aplicação -> {V,F,D}

aplicacao_clinica('ben-u-ron','dores de cabeca').
aplicacao_clinica('ben-u-ron','sindromes gripais').
aplicacao_clinica('ben-u-ron','febre').
aplicacao_clinica('ben-u-ron','dores de dentes').
aplicacao_clinica('buscopan','dores abdominais').
aplicacao_clinica('brufen','febre').
aplicacao_clinica('brufen','osteoartrose').
aplicacao_clinica('brufen','traumatismos').
aplicacao_clinica('cegripe','sindromes gripais').
aplicacao_clinica('cegripe','constipacoes').
aplicacao_clinica('cegripe','perturbacoes alergicas da nasofaringe').
aplicacao_clinica('neosoro','aplicacao_incerta').
aplicacao_clinica('microvlar','anti-concepcional').
aplicacao_clinica('plavix','prevencao da trombose arterial').

-aplicacao_clinica('neosoro','traumatismos').
-aplicacao_clinica('neosoro','febre').
-aplicacao_clinica('neosoro','Sindromes gripais').

%Excepções

excepcao(aplicacao_clinica('neosoro',C)) :- aplicacao_clinica('neosoro',aplicacao_in
```

Como ponto indiscutível na realização do trabalho prático, mais uma vez foram implementados invariantes:

```
% Invariantes

%---------------Medicamento tem de existir
```

```
+aplicacao_clinica(M,A) :: medicamento(M).

%---------------  Não permitir a inserção de conhecimento repetido

+tipo_medicamento(M,A) :: (solucoes( (M,A),(tipo_medicamento(M,A)),S),
comprimento(S,N), N==1).

% Conhecimento Negativo

-aplicacao_clinica(M,A) :- nao(aplicacao_clinica(M,A)), nao(excepcao(aplicacao_clini
```

4.5 Princípio_Activo

Continuou-se a definicão do universo farmacêutico e desta feita, a explicitação do princípio activo. Para tal foi criado o predicado principio_activo composto por 2 argumentos(Medicamento e respectivo Princípio Activo). Neste predicado foram utilizados dois valores de tipo Nulo Desconhecido (Incerto e Impreciso).

Assim sendo, o é nos desconhecido o princípio activo do medicamento neosoro sabendo-se apenas que não o é o paracetamol ou ibuprofen. Para o valor nulo "Impreciso" utilizou-se o medicamento Microvlar, e como abaixo descrito, não se sabe qual o princípio activo do Microvlar sabendo-se no entanto que será o levonorgestrel ou o etinilestradiol.

```
 %Extensao do predicado principio_activo: Medicamento,
% Princípio Ativo -> {V,F,D}

principio_activo('ben-u-ron','paracetamol').
principio_activo('buscopan','duboisia').
principio_activo('brufen','ibuprofen').
principio_activo('ben-u-ron','paracetamol').
principio_activo('cegripe','paracetamol').
principio_activo('neosoro',principio_incerto).
principio_activo('plavix','clopidrogel').

-principio_activo('neosoro','paracetamol').
-principio_activo('neosoro','ibuprofen').

% Excepções

excepcao(principio_activo('neosoro',P)) :-
 principio_activo('neosoro',principio_incerto).
```

```
excepcao(principio_activo('microvlar','levonorgestrel')).
excepcao(principio_activo('microvlar','etinilestradiol')).
```

Novamente se impõe a utilização de Invariantes:

```
% Invariantes

%---------------Medicamento tem de existir

+principio_activo(M,P) :: medicamento(M).

%---------------  Não permitir a inserção de conhecimento repetido

+principio_activo(M,P) :: (solucoes( (M,P),(principio_activo(M,P)),S),
comprimento(S,N), N==1).

% Conhecimento Negativo

-principio_activo(M,P) :- nao(principio_activo(M,P)), nao(excepcao(principio_activo(
```

4.6 Apresentação_farmacêutica

Posteriormente, implementou-se no programa Prolog o predicado apresentacao_farmaceutica. Este, tal como o nome indica, permite estabelecer a apresentação ou apresentações farmaçêuticas de um dado medicamento. Assim sendo este predicado é composto por dois argumentos(Medicamento e respectiva apresentação). Face às implicações deste predicado no conjunto do programa, considerou-se favorável apenas a inserção de conhecimento positivo sem introdução de valores nulos e conhecimento negatvo. Desde já se apresenta o conjunto de invariantes utilizados.

```
% Extensao do predicado apresentacao_farmaceutica: Medicamento,
%Apresentacao -> {V,F,D}


apresentacao_farmaceutica('ben-u-ron','comprimidos').
apresentacao_farmaceutica('ben-u-ron','xarope').
apresentacao_farmaceutica('buscopan','ampolas').
apresentacao_farmaceutica('brufen','comprimidos').
apresentacao_farmaceutica('brufen','xarope').
```

```
apresentacao_farmaceutica('cegripe','comprimidos').
apresentacao_farmaceutica('neosoro','solucao nasal').
apresentacao_farmaceutica('microvlar','comprimidos').
apresentacao_farmaceutica('plavix','comprimidos').

% Invariantes

%---------------Medicamento tem de existir

+apresentacao_farmaceutica(M,A) :: medicamento(M).

%---------------  Não permitir a inserção de conhecimento repetido

+apresentacao_farmaceutica(M,A) :: (solucoes( (M,A),
(apresentacao_farmaceutica(M,A)),S),
comprimento(S,N), N==1).
```

4.7 Predicados relativos a Preços

4.7.1 Preco_Recomendado

Após a definição da apresentação farmaçêutica procedeu-se à implementação de preços para cada medicamento, dependendo da sua apresentação farmaçêutica. Desta forma foram criados 3 predicados: preco_recomendado, preco_publico e regime_especial. O predicado preco_recomendado é composto por 3 argumentos(Medicamento, Apresentação farmaçêutica e respetivo preço). Na implementação deste predicado utilizou-se os 3 diferentes tipos de valor Nulo (Incerto, Imprevisto e Interdito). Como observável o preço recomendado para o medicamento Buscopan em ampolas é desconhecido(Incerto), tendo sido criada a respectiva excepção. Para o medicamento Neosoro em solução nasal o preço é interdito sendo mais uma vez criada a correspondente excepção e posterior invariante por forma a não permitir a inserção de preço para este caso específico. Por último, para o Medicamento Brufen em comprimidos foi utilizado valor nulo Imprevisto, sendo que o preço recomendado se encontra entre 6 e 9 inclusivé.

```
% Extensao do predicado preco_recomendado: Medicamento, Apresentacao,
% Preco -> {V,F,D}

preco_recomendado('ben-u-ron','comprimidos',9).
preco_recomendado('ben-u-ron','xarope',11).
preco_recomendado('buscopan','ampolas',precor_incerto).
preco_recomendado('brufen','xarope',10).
preco_recomendado('neosoro','solucao nasal',preco_nulo).
preco_recomendado('microvlar','comprimidos',5).
preco_recomendado('plavix','comprimidos',25).
```

```
excepcao(preco_recomendado('brufen', 'comprimidos', P)) :- P>=6, P=<9 .
excepcao(preco_recomendado('neosoro','solucao nasal', P))
:- preco_recomendado('neosoro','solucao nasal',preco_nulo).

excepcao(preco_recomendado('buscopan','ampolas',P)) :-
preco_recomendado('buscopan','ampolas',precor_incerto).
```

Novamente se introduziu ao programa invariantes necessários por forma a manter a consistência da base de conhecimento, Importa referir que para os predicados envolvendo preço, é necessário que exista Medicamento e correspondente Apresentação farmaçêutica, ao invés dos predicados anteriormente apresentados cuja inserção apenas requeria a existência de Medicamento.

```
% Invariantes

%---------------Medicamento e correspondente apresentação farmaçêutica
%têm de existir

+preco_recomendado(M,A,P) :: (medicamento(M), apresentacao_farmaceutica(M,A)).

%---------------  Não permitir a inserção de conhecimento repetido

+preco_recomendado(M,A,P) :: (solucoes( (M,A,P),(preco_recomendado(M,A,P)),S),
comprimento(S,N), N==1).

%---------------  Não permitir a inserção de preço recomendado para o Medicamento
% neosoro

+preco_recomendado(M, A, P) :: (solucoes(Ps, (preco_recomendado('neosoro',
'solucao nasal', Ps),
 nao(nulo(Ps))), [])).


% Conhecimento Negativo

-preco_recomendado(M,A,P) :- nao(preco_recomendado(M,A,P)),
nao(excepcao(preco_recomendado(M,A,P))).
```

4.7.2 Preco_publico

O predicado preco_publico é em tudo semelhante ao predicao predicado preco_recomendado. Este é composto pelos mesmos 3 argumentos e similarmente faz uso dos 3 tipos de valor Nulo. Desta forma o medicamento Ben-u-ron em comprimido apresenta valor Incerto ao invés do medicamento Brufen em xarope cujo preço é interdito. Existem ainda dois medicamentos cujo preço é imprevisto: Ben-u-ron em xarope cujo preço pode variar entre os 12 e 13, e Microvlar em comprimidos em que o intervalo de valores se encontra entre 5 e 7, ambos inclusivé.

```
% Estensao do predicado preco_publico: Medicamento,
% Apresentacao, Preco -> {V,F,D}

preco_publico('ben-u-ron','comprimidos',preco_incerto).
preco_publico('buscopan','ampolas',4).
preco_publico('brufen','xarope',preco_nulo).
preco_publico('brufen','comprimidos',10).
preco_publico('neosoro','solucao nasal',4).
preco_publico('plavix','comprimidos',27).

% Excepções

excepcao(preco_publico('ben-u-ron', 'xarope', P)) :- P>=12, P=<13.
excepcao(preco_publico('microvlar', 'comprimidos', P)) :- P>=5, P=<7.

excepcao(preco_publico('brufen','xarope',P)) :-
 preco_publico('brufen','xarope', preco_nulo).
excepcao(preco_publico('ben-u-ron','comprimidos',P)) :-
 preco_publico('ben-u-ron','comprimidos',preco_incerto).
```

A criação de invariantes é novamente necessária, sendo estes em tudo idênticos aos utilizados no predicado preco_recomendado.

```
% Invariantes

%---------------Medicamento e correspondente apresentação farmaçêutica têm
de existir

+preco_publico(M,A,P) :: (medicamento(M), apresentacao_farmaceutica(M,A)).

%---------------  Não permitir a inserção de conhecimento repetido
```

```
+preco_publico(M,A,P) :: (solucoes( (M,A,P),(preco_publico(M,A,P)),S),
comprimento(S,N), N==1).

%---------------  Não permitir a inserção de preço recomendado para o Medicamento
neosoro

+preco_publico(M, A, P) :: (solucoes(Ps, (preco_publico('brufen', 'xarope',
Ps),
nao(nulo(Ps))), [])).


% Conhecimento Negativo

-preco_publico(M,A,P) :- nao(preco_publico(M,A,P)), nao(excepcao(preco_publico(M,A,P
```

4.7.3 Regime_especial

O predicado regime_especial é idêntico aos dois predicados anteriores, no entanto este é composto por mais um argumento (Escalão) que definirá o preço de um medicamento consuante o Escalão que um utente possui. Desta forma o predicado regime_especial é composto por 4 argumentos(Medicamento, Apresentação farmacêutica, Escalão e respectivo preço). Neste sentido considerou-se a existência de 3 escalões(A,B,C) com o maior desconto a ser aplicado para o escalão A sendo reduzido no sentido A a C.

Novamente se cosiderou relevante a utilização dos 3 tipos de valor nulo. Desta forma Medicamentos como o Ben-u-ron em comprimidos para escalão B apresentam preço desconhecido(Incerto), tal como e.g. Plavix em comprimidos para escalão B. O medicamento Brufen em xarope para escalão C apresenta preço interdito. Com preço Imprevisto implementou-se 3 casos mais especificamente, Ben-u-ron em xarope para escalão C, Buscopan em ampolas para escalão A e Plavix em comprimidos para escalão C. Estes medicamentos apresentam um preço respectivamente compreendido entre, 7 e 9, 1 e 3 e 3 e 17.

```
% Estensao do predicado regime_especial: Medicamento, Apresentacao, Escalao,
Preco -> {V,F,D}


regime_especial('ben-u-ron','comprimidos',a, 2).
regime_especial('ben-u-ron','comprimidos',b, regime_incerto).
regime_especial('ben-u-ron','xarope',b, regime_incerto).
regime_especial('brufen','comprimidos',b, 3).
```

```
regime_especial('brufen','xarope',c, preco_nulo).

regime_especial('buscopan','ampolas',c, 2).


% não existe regime especial para o medicamento neosoro

regime_especial('microvlar','comprimidos',b,3).
regime_especial('plavix','comprimidos',a,3).
regime_especial('plavix','comprimidos',b,regime_incerto).

% Excepções

excepcao(regime_especial('ben-u-ron', 'xarope', c, P)) :- P>=7, P=<9.
excepcao(regime_especial('buscopan', 'ampolas', b, P)) :- P>=1, P=<3.
excepcao(regime_especial('plavix', 'comprimidos', c, P)) :- P>=17, P=<20.


excepcao(regime_especial('brufen', 'xarope',c,P)) :-
regime_especial('brufen', 'xarope', c, preco_nulo).

excepcao(regime_especial('plavix', 'comprimidos', b,P)) :-
 regime_especial('plavix', 'comprimidos', b, regime_incerto).
excepcao(regime_especial('ben-u-ron','comprimidos',b,P)) :-
 regime_especial('ben-u-ron','comprimidos',b, regime_incerto).
excepcao(regime_especial('ben-u-ron','xarope',b,P)) :-
regime_especial('ben-u-ron','xarope',b, regime_incerto).
```

De seguida se apresentam os invariantes implementados para o predicado regime_especial.

```
% Invariantes

%---------------Medicamento e correspondente apresentação farmaçêutica têm
de existir

+regime_especial(M,A,E,P) :: (medicamento(M), apresentacao_farmaceutica(M,A)).

%---------------  Não permitir a inserção de conhecimento repetido

+regime_especial(M,A,E,P) :: (solucoes( (M,A,E,P),(regime_especial(M,A,E,P)),S),
comprimento(S,N), N==1).

%---------------  Não permitir a inserção de regime especial com Escalão
C para o Medicamento Brufen em Xarope
```

```
+regime_especial(M,A,E,P) :: (solucoes(Ps, (regime_especial('brufen', 'xarope',
c, Ps), nao(nulo(Ps))), [])).


% Conhecimento Negativo

-regime_especial(M,A,E,P) :- nao(regime_especial(M,A,E,P)), nao(excepcao(regime_espe
```

Predicado Soluções

5 Predicados relativos à Data de Introdução e de Validade

5.1 Data_validade

Após a introdução de informação relativa ao preço dos medicamentos, pretendeu-se adicionar ao programa dados sobre a data de validade e introdução de um dado medicamento. O predicado data_validade e data_introducao são compostos por 5 agumentos(Medicamento, Apresentação farmacêutica, Dia, Mês e Ano de validade).

No desenvolvimento deste predicado foram utilizados os 3 tipos de valores nulos. Para o medicamento Brufen em xarope a data de validade é interdida apresentando assim o dia, mês e ano valor nulo. O medicamento brufen em comprimidos possui dia e mês de validade desconhecido(Incerto), enquanto que o Microvlar em comprimidos apenas possui o dia incerto. Para exemplo de conhecimento impreciso encontra-se o medicamento Plavix em comprimidos cujo dia de validade se encontra entre o dia 9 e 21.

```
%------------------------------- - - - - - - - - - -  -  -  -  -   -
% Estensao do predicado data_validade: Medicamento, Apresentacao,
% Dia, Mes, Ano -> {V,F,D}

data_validade('ben-u-ron','comprimidos',16,10,2015).
data_validade('ben-u-ron','xarope',12,8,2015).
data_validade('brufen','comprimidos',dia_incerto,mes_incerto,2016).
data_validade('brufen','xarope',dia_nulo,mes_nulo,ano_nulo).
data_validade('microvlar','comprimidos',dia_incerto,3,2016).
data_validade('neosoro','solucao nasal',4,5,2018).

excepcao(data_validade('plavix', 'comprimidos', D, 7, 2017)) :- D=<21, D>=9.

excepcao(data_validade('brufen','comprimidos',D,M,2016)):-
```

```
data_validade('brufen','comprimidos',dia_incerto,mes_incerto,2016).

excepcao(data_validade('microvlar','comprimidos',D,3,2016)):-
data_validade('microvlar','comprimidos',dia_incerto,3,2016).

excepcao(data_validade('brufen','xarope',D,M,A)) :-
data_validade('brufen','xarope',dia_nulo,mes_nulo,ano_nulo).
```

O predicado data_validade apresenta no seu conjunto de invariantes um até aqui não descrito. Esse invariante garante que a data de validade de um medicamento é superior à correspondente data de introdução no mercado.

```
%Invariantes

% Não permitir a inserção de conhecimento repetido

+data_validade(X,Apr,D,M,A) :: (solucoes((X,Apr,D,M,A),data_validade( X,Apr,D,M,A
), S),comprimento( S,N ), N == 1).

% Medicamento e apresentação farmacêutica têm de existir

+data_validade(X,Apr,D,M,A) :: (medicamento(X), apresentacao_farmaceutica(X,
Apr)).

% Não permir a iserção de data de validade para o Medicamento Brufen em Xarope

+data_validade(X,Apr,D,M,A) :: (findall((D_X, M_X, A_X),
(data_validade('brufen', 'xarope', D_X, M_X, A_X),
(nao(nulo(D_X)); nao(nulo(M_X)); nao(nulo(A_X)))), [])).

%Data de validade tem de ser superior à data de introdução

+data_validade( X,Apr,D,M,A ) :: (
( nao(data_introducao(X,Apr,D2,M2,A2)) );
( A>A2 );
( A==A2, M>M2 );
( A==A2, M==M2, D>D2 )).

%Conhecimento Negativo

-data_validade(A,B,C,D,E) :- nao(data_validade(A,B,C,D,E)), nao(excepcao(data_valida
```

5.2 Data_introducao

O predicado data_introducao apresenta total semelhança no que concerne a argumentos e tipos de valores nulos utilizados. Tendo isso em consideração apresenta-se a implementação utilizada.

```
% Estensao do predicado data_introducao: Medicamento, Apresentacao, Dia,
Mes, Ano -> {V,F,D}

data_introducao('ben-u-ron','comprimidos',7,8,2013).
data_introducao('ben-u-ron','xarope',dia_incerto,mes_incerto,2014).
data_introducao('brufen','comprimidos',dia_incerto,4,2014).
data_introducao('brufen','xarope',dia_nulo,mes_nulo,ano_nulo).
data_introducao('microvlar','comprimidos',dia_incerto,mes_incerto,2014).
data_introducao('neosoro','solucao nasal', 17,12,2015).

%Excepções

excepcao(data_introducao('plavix', 'comprimidos', D2, 9, 2015)) :- D2=<29,
D2>=8.

excepcao(data_introducao('ben-u-ron','xarope',D,M,2014)):-
data_introducao('ben-u-ron','xarope',dia_incerto,mes_incerto,2014).

excepcao(data_introducao('brufen','comprimidos',D,4,2014)):-
 data_introducao('brufen','comprimidos',dia_incerto,4,2014).

excepcao(data_introducao('microvlar','comprimidos',D,M,2014)):-
data_introducao('microvlar','comprimidos',dia_incerto,mes_incerto,2014).

excepcao(data_validade('brufen','xarope',D,M,A)) :-
data_introducao('brufen','xarope',dia_nulo,mes_nulo,ano_nulo).

% Invariantes

%---------------Medicamento e correspondente apresentação farmaçêutica têm
de existir

+data_introducao(X,Apr,D,M,A) :: ( medicamento(X), apresentacao_farmaceutica(X,Apr)
).


% Não permitir conhecimento repetido
```

```
+data_introducao(X,Apr,D,M,A) :: (solucoes((X,Apr,D,M,A),data_introducao(
X,Apr,D,M,A ), S),
comprimento( S,N ), N == 1
).

% Data de introdução menor do que data de validade
+data_introducao( X,Apr,D,M,A ) :: (
( nao(data_validade(X,Apr,D2,M2,A2)) );
( A<A2 );
( A==A2, M<M2 );
( A==A2, M==M2, D<D2 )).

% Não permir a iserção de data de validade para o Medicamento Brufen em Xarope

+data_introducao(X,Apr,D,M,A) :: (solucoes((D_X, M_X, A_X),
 (data_introducao('brufen', 'xarope', D_X, M_X, A_X),
(nao(nulo(D_X)); nao(nulo(M_X)); nao(nulo(A_X)))), [])).

%Conhecimento Negativo

-data_introducao(A,B,C,D,E) :- nao(data_introducao(A,B,C,D,E)), nao(excepcao(data_ir
```

6 Contra_Indicacao

O predicado contra_indicacao foi criado com o intuito de definir as contra indicações ou problemas/estados de saúde de um utente que o impeça de utilizar determinado medicamento. Este predicado é constituído por 2 argumentos(Medicamento e Contra Indicação) e utilizou-se novamente valores nulos na definição deste predicado. Utilizaram-se mais especificamente valores nulos do tipo Incerto e Interdito. Para o primeiro caso indica-se que o medicamento buscopan possui contra indicação incerta, no entanto através da utilização de conhecimento negativo informa-se o sistema que a insuficiência cardíaca grave e a diabetes mellitus grave não fazem parte das contra indicações deste.

```
% Estensao do predicado contra_indicacao: Medicamento, Contra Indicação ou
Problemas do paciente -> {V,F,D}

contra_indicacao('ben-u-ron','doenca hepatica grave').
contra_indicacao('ben-u-ron','hipersensibilidade ao paracetamol').
contra_indicacao('brufen','hipersensibilidade ao ibuprofeno').
contra_indicacao('brufen','insuficiencia cardiaca grave').
contra_indicacao('brufen','alteracoes de coagulacao').
contra_indicacao('brufen','hemorrogia gastrointestinal').
```

```
contra_indicacao('buscopan',cindicacao_incerta).
contra_indicacao('microvlar','insuficiencia hepatica grave').
contra_indicacao('microvlar','antecedentes de ictericia').
contra_indicacao('microvlar','carcinoma de mama').
contra_indicacao('microvlar','diabetes mellitus grave').
contra_indicacao('neosoro',c_indicacao_nula).
contra_indicacao('plavix','intolerancia a galactose').
contra_indicacao('plavix','hipersensibilidade ao clopidrogel').

-contra_indicacao('buscopan','inufuciencia cardiaca grave').
-contra_indicacao('buscopan','diabetes mellitus grave').

% Excepções

excepcao(contra_indicacao('neosoro',C)) :- contra_indicacao('neosoro',c_indicacao_nu

excepcao(contra_indicacao('buscopan',C)) :- contra_indicacao('buscopan',cindicacao_i

% Invariantes

%---------------Medicamento tem de existir

+contra_indicacao(M,C) :: medicamento(M).

%---------------   Não permitir a inserção de conhecimento repetido

+contra_indicacao(M,C) :: (solucoes( (M,C),(contra_indicacao(M,C)),S),
comprimento(S,N), N==1).

%---------------  Não permitir a inserção de contra indicações para o medicamento
neosoro

+contra_indicacao(M, C) :: (solucoes(Cs, (contra_indicacao('neosoro', Cs),
nao(nulo(Cs))), [])).

% Conhecimento Negativo

-contra_indicacao(M,C) :- nao(contra_indicacao(M,C)), nao(excepcao(contra_indicacao(
```

7 Armazem

Por último foi criado o predicado armazem que pretende organizar a localização de cada medicamento segundo o tipo de medicamento. Este predicado é composto por dois argumentos(Número do armazém e tipo de Medicamento destinado ao mesmo). Desta forma implementou-se neste predicado valores nulos. No caso específico do armazém 3 o tipo de medicamento alocado é interdito, enquanto que para o armazém 8 este apenas é desconhecido(Incerto). Também para este predicado se recorreu a invariantes por forma a manter a consistência da base de conhecimento. Um novo invariante foi introduzido para este predicado por forma a impedir a introdução de mais do que um tipo de medicamento por armazém.

```
% Extensao do predicado armazem:  Armazem, Tipo de medicamento -> {V,F,D}

armazem(1,'analgesico').
armazem(2,'anti-inflamatorio').
armazem(3,tipo_nulo).
armazem(4,'antiplaquetar').
armazem(5,'contraceptivo').
armazem(6,'antibiotico').
armazem(7,'antidepressivo').
armazem(8,incerto).
armazem(9,'placebo').

% Excepções

excepcao(armazem(8,T)) :- armazem(8,incerto).
excepcao(armazem(3,T)) :- armazem(3,tipo_nulo).

% Invariantes

%---------------   Não permitir a inserção de conhecimento repetido

+armazem(Arm,T) :: (solucoes( (Arm,T),(armazem(Arm,T)),S),
comprimento(S,N), N==1).

%---------------  Não permitir a inserção de Tipo de medicamento para o armazém
3

+armazem(Arm, T) :: (solucoes(Ts, (armazem(3, Ts), nao(nulo(Ts))), [])).


%---------------  Não permitir mais do que um tipo de medicamento por armazem
```

```
+armazem(Arm,T) :: (solucoes( Ts, armazem(Arm, Ts),S),
comprimento(S,N), N==1).

% Conhecimento Negativo

-armazem(Arm,T) :- nao(armazem(Arm,T)), nao(excepcao(armazem(Arm,T))).
```

8 Declarações Iniciais e definições Iniciais

Declarações Iniciais

```
% SICStus PROLOG: Declaracoes iniciais

:- set_prolog_flag( discontiguous_warnings,off ).
:- set_prolog_flag( single_var_warnings,off ).
:- set_prolog_flag( unknown,fail ).
```

A implementação de flags permite a desactivação de "warnings", avisos de erros simples para o programador:

- A primeira flag permite desactivar mensagens de aviso sobre predicados extensos e que utilizam outros predicados nas suas definições;
- A segunda permite desactivar mensagens de aviso sobre variáveis que não comecem por _ e que são utilizadas apenas uma vez;
- A última declaração força que a consulta de predicados desconhecidos falhe;

Definições Iniciais

```
:- op( 900,xfy,'::' ).
:- dynamic '-'/1.
:- dynamic medicamento/1.
:- dynamic tipo_medicamento/2.
:- dynamic aplicacao_clinica/2.
:- dynamic principio_activo/2.
:- dynamic apresentacao_farmaceutica/2.
:- dynamic preco_recomendado/3.
:- dynamic preco_publico/3.
:- dynamic regime_especial/4.
:- dynamic data_validade/5.
:- dynamic data_introducao/5.
:- dynamic contra_indicacao/2.
:- dynamic armazem/2.
```

O predicado pré-definido *op* permite a definição do operador "::" utilizado na estrutura de invariantes. As cláusulas "*dynamic*", informam que os predicados em referência podem ser alterados durante a execução.

9 Predicados Auxiliares

Predicado Solucoes

```
%Extensão do predicado solucoes: X,Y,Z -> {V,F}
solucoes( X,Y,Z ) :-
findall( X,Y,Z ).
```

O predicado soluções possui como objetivo encontrar todas as soluções na Base de Conhecimento para resolução de determinado problema. Este predicado é constituído por 3 argumentos: (X) representa o termo que se pretende procurar na cláusula, na forma de consulta (Y). O argumento (Z) é uma lista de instâncias de (X) que tornam verdadeiro (Y). Este predicado recorre ao predicado findall, um predicado definido no interpretador SICStus que possui a mesma finalidade do predicado soluções.

Predicado Comprimento

```
%Extensao do predicado comprimento: S,N ->{V,F}
comprimento( S,N ) :-
length( S,N ).
```

O predicado comprimento permite conhecer o número de elementos de uma determinada lista. Este é constituído por apenas uma cláusula e 2 argumentos:

- (S) representa a lista cujo número de elementos se pretende determinar;
- (N) o número de elementos dessa lista. Tal como o predicado soluções o predicado comprimento recorre a um predicado definido no interpretador SICstus, neste caso *lenght*.

Predicado Evolução

```
% Extensão do predicado que permite a evolucao do conhecimento

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

insercao( Termo ) :-
assert( Termo ).
```

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

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

Este predicado foi criado para podermos adicionar novos factos à nossa base de conhecimento. Para isso, o argumento Termo passa por uma fase de validação de modo a manter a integridade da Base de conhecimento. Para tal , o predicado evolução precisa de invocar outros dois predicados : o **inserção** e o **teste**.

O predicado **inserção** tem apenas a função de adicionar , ou em caso de falha , remover informação da base de conhecimento. Isto é feito pelos dois predicados pré-definidos: o **assert** para inserção e o **retract** para a remoção.

O predicado **teste** verifica se todas as condições na lista, neste caso os invariantes, são verdadeiros . Isto vai impedir que informação comprometedora da base de conhecimento seja adicionado.

O predicado **evolução** usufrui das funcionalidades dos predicados acima referidos. Este contém uma cláusula que retorna uma lista de todos os invariantes estabelecidos pelo termo atribuído. De seguida , o predicado **insercao** adiciona o termo à base de conhecimento.

Por último , é realizado um teste à base de conhecimento de modo a verificar a validade dos invariantes definidos. Caso o termo seja válido o predicado textbfinsercao insere o termo à base de conhecimento, caso contrário, o programa retrocede à segunda cláusula deste predicado e o termo é removido.

Predicado Remoção remover(Termo) :- solucoes(Invariante, -Termo::Invariante, Lista), teste(Lista) , remocao(Termo).

remocao(Termo) :- retract(Termo).

Este predicado tem um funcionamento semelhante ao **evolucao** mas com a diferença essencial de remoção de informação da base de conhecimento.

Este é constituído por duas cláusulas: inicialmente o **remocao** remove informação da base de conhecimento ou insere em caso do primeiro falhar, e o **teste** que à semelhança do que foi descrito em anteriormente verifica a validade dos invariantes face a esta remoção.

O predicado **retirar** cria então uma lista com todos os invariantes face ao termo a ser adicionado, retirando pelo predicado **remocao** o termo pretendido da base de conhecimento. Por último o predicado **teste** verifica se a base de conhecimento se mantém consistente aos invariantes definidos anteriormente. Caso este teste falhe, o programa retrocede e executa a segunda cláusula do predicado **remocao**, inserindo o termo que tinha sido eliminado de modo a não remover informação que danifique a integridade da base de conhecimento.

10 Conclusões

A elaboração do trabalho prático em questão permitiu aprofundar conhecimentos na utilização da linguagem de programação Prolog e adjacente desenvolvimento de mecanismos de raciocínio para resolução de problemas.Em particular a representação de conhecimento imperfeito através de três tipos de valores nulos:incertos,imprecisos e interditos, foi alvo de maior estudo.

Uma vez que se pretendia demonstrar as funcionalidades subjacentes à programação em lógica estendida e à representação de conhecimento imperfeito, foi de prima importância a criação de um sistema de inferência capaz de implementar um mecanismo de raciocínio adequado.

Desta forma abandonou-se o pressuposto do mundo fechado que apresenta que tudo o que não está definido na base de conhecimento é tido como falso, podendo agora considerar respostas "desconhecidas".

Uma vez que foi possível cumprir com as necessidades de demonstração propostas bem como introdução de novas funcionalidades considera-se favorável a realização do Trabalho prático enunciado. Além dos requesitos necessários procorou-se construir um universo farmacêutico consistente e com informação relevante.

Como sugestão futura, considera-se interessante a extensão de mais predicados bem como a adição de mais produtos e consequentes relações por forma a melhor descrever o universo de âmbito farmacêutico.

11 Bibliografia

[1] Bratko, I. Prolog Programming for Artificial Intelligence. 3a Edição. Addison-Wesley Publishers Limited, 2001.

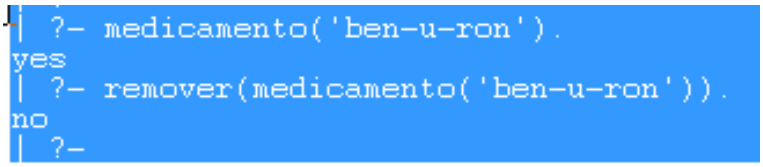
[2] SICStus Prolog User's Manual.

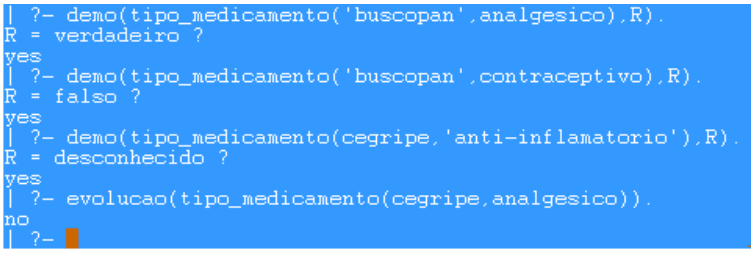
[3] Analide, César; Maia Neves, José. "Apontamentos das aulas teóricas da unidade curricular Programação em Lógica e Representação do Conhecimento", 2012.

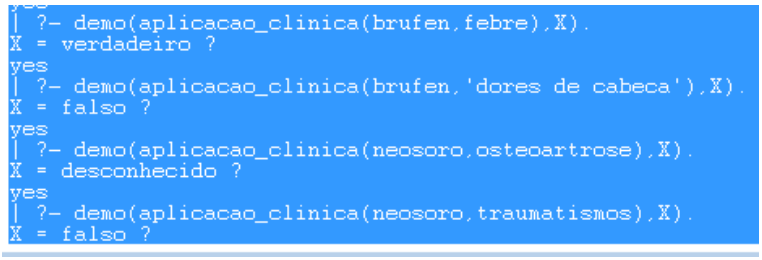
[4] Analide, César; Maia Neves, José. "Representação de Informação Incompleta"

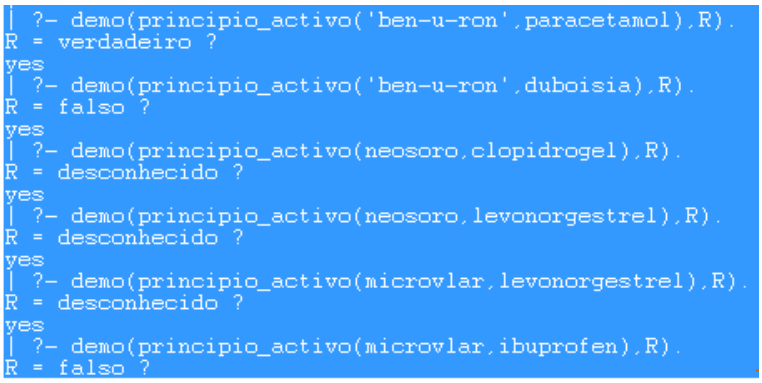
12 Apendices

De seguida se apresentam exemplos de utilização do programa em Prolog desenvolvido.









```
| ?- demo(preco_recomendado('ben-u-ron',comprimidos,9),R).
R = verdadeiro ?
yes
| ?- demo(preco_recomendado('ben-u-ron',comprimidos,15),R).
R = falso ?
yes
| ?- demo(preco_recomendado(buscopan,ampolas,12),R).
R = desconhecido ?
yes
| ?- demo(preco_recomendado(brufen,comprimidos,7),R).
R = desconhecido ?
yes
| ?- demo(preco_recomendado(brufen,comprimidos,12),R).
R = falso ?
```

```
| ?- demo(preco_publico(brufen,comprimidos,10),R).
R = verdadeiro ?
yes
| ?- demo(preco_publico(brufen,comprimidos,12),R).
R = falso ?
yes
| ?- demo(preco_publico(microvlar,comprimidos,7),R).
R = desconhecido ?
yes
| ?- demo(preco_publico(microvlar,comprimidos,2),R).
R = falso ?
yes
| ?- evolucao(preco_publico(microvlar,comprimidos,2)).
yes
| ?- evolucao(preco_publico(brufen,xarope,10)).
no
| ?-
```

```
| ?- demo(regime_especial(buscopan,ampolas,2),R).
R = desconhecido ?
yes
| ?- demo(regime_especial(buscopan,ampolas,15),R).
R = desconhecido ?
yes
| ?- demo(regime_especial(brufen,comprimidos,b,3),R).
R = verdadeiro ?
yes
| ?- demo(regime_especial(brufen,comprimidos,b,10),R).
R = falso ?
yes
| ?- demo(regime_especial(buscopan,ampolas,b,2),R).
R = desconhecido ?
yes
| ?- demo(regime_especial(buscopan,ampolas,b,15),R).
R = falso ?
```

```
| ?- demo(contra_indicacao('ben-u-ron','doenca hepatica grave'),R).
R = verdadeiro ?
yes
| ?- demo(contra_indicacao('ben-u-ron','hemorrogia gastrointestinal'),R).
R = falso ?
yes
| ?- demo(contra_indicacao(buscopan,'alteracoes de coagulacao'),R).
R = desconhecido ?
yes
| ?- demo(contra_indicacao(buscopan,'diabetes mellitus grave'),R).
R = falso ?
```

```
| ?- demo(data_validade('ben-u-ron',comprimidos,16,10,2015),R).
R = verdadeiro ?
yes
| ?- demo(data_validade('ben-u-ron',comprimidos,16,10,2017),R).
R = falso ?
yes
| ?- demo(data_validade(plavix,comprimidos,15,7,2017),R).
R = desconhecido ?
yes
| ?- demo(data_validade(plavix,comprimidos,29,7,2017),R).
R = falso ?
yes
| ?- demo(data_introducao(neosoro,'solucao nasal',17,12,2015),R).
R = verdadeiro ?
yes
| ?- demo(data_introducao(neosoro,'solucao nasal',31,12,2015),R).
R = falso ?
yes
| ?- demo(data_introducao(plavix,comprimidos,17,9,2015),R).
R = desconhecido ?
yes
| ?- demo(data_introducao(plavix,comprimidos,1,9,2015),R).
R = falso ?
```

```
| ?- demo(armazem(1,antibiotico),R).
R = falso ?
yes
| ?- demo(armazem(1,analgesico),R).
R = verdadeiro ?
yes
| ?- evolucao(armazem(3,analgesico)).
no
| ?- demo(armazem(3,analgesico),R).
R = desconhecido ?
yes
| ?- demo(armazem(1,analgesico),R).
R = verdadeiro ?
yes
| ?- evolucao(armazem(1,antibiotico)).
no
| ?- evolucao(armazem(10,antibiotico)).
yes
```