Roteiro 7

Alexsandro Santos Soares
prof.asoares@gmail.com

Programação Lógica
Faculdade de Computação
Universidade Federal de Uberlândia

1 Introdução

Esta aula tem por finalidade :

- Praticar o uso de cortes e da negação como falha.
- Familiarizá-lo com os predicados do Prolog que colecionam todas as soluções de um problema em uma única lista.

2 Corte e negação como falha

E. 1 Faça experimentações com as três versões do predicado `max/3` definidas na aula teórica: a versão sem corte, a versão com corte verde e a versão com corte vermelho. Como usual, "experimentar" significa "executar o trace", assegurando-se que rastreie consultas na qual todos os três argumentos estão instanciados para inteiros e, também, consultas onde o terceiro argumento é uma variável.

E. 2 Experimente todos os métodos discutidos na aula para lidar com as preferências de Vicente. Isto é, faça experimentos com o programa que usa a combinação de corte com `fail`, com o programa que usa negação como falha corretamente e também com o programa que torna-se errôneo quando utiliza negação no lugar errado.

E. 3 Defina um predicado `nu/2` ("não unificável") que recebe dois termos como argumentos e sucede se os dois termos não unificam. Por exemplo:

```
?- nu(foo,foo).
false

?- nu(foo,blob).
true

?- nu(foo,X).
false
```

Você deve definir este predicado de três formas diferentes:

a) Primeiro (e mais fácil), escreva-o com a ajuda de `=` e `\+`.

b) Segundo, escreva-o com a ajuda de `=`, mas não use `\+`.

c) Terceiro, escreva usando uma combinação de corte e `fail`. Não use `=` e `\+`.

E. 4 Defina um predicado `unificável(Lista1,Termo,Lista2)` onde `Lista2` é a lista de todos os membros da `Lista1` que poderiam se unificar com `Termo`, mas *não* são instanciados pela unificação. Por exemplo,

```
?- unificável([X,b,t(Y)],t(a),Lista).
Lista = [X,t(Y)].
```

Note que `X` e `Y` ainda *não* estão instanciadas na resposta. Assim a parte complicada é: como verificar se elas unificam com `t(a)` sem instanciá-las? (Dica: considere usar o teste `\+ (termo1 = termo2)`. Por quê? Pense sobre isto. Talvez você deseje também pensar sobre o teste `\+(\+ (termo1 = termo2))`).

3 Predicados com coleta de todas as soluções

Suponha que tenhamos um predicado Prolog descrevendo informação sobre países. Os fatos têm o formato ***país(Nome, Continente, Populacão, Fronteiras)***, onde ***Nome*** é o nome do País, ***Continente*** é o continente a que o país pertence (África, América, Ásia, Europa ou Oceania), ***Populacão*** é um inteiro que representa o número de habitantes (em milhões) do país e ***Fronteiras*** é uma lista contendo os nomes dos países com os quais o país faz fronteira. Exemplo

```
país(alemanha, europa, 83, [frança, bélgica, holanda, suíça]).
país(austrália, oceania, 25, []).
país(bélgica, europa, 11, [frança, holanda, alemanha]).
país(espanha, europa, 47, [portugal, frança]).
país(frança, europa, 67, [espanha, suíça,  bélgica, alemanha,
     itália]).
país(holanda, europa, 17, [bélgica, alemanha]).
país(indonésia, oceania, 268, []).
país(itália, europa, 60, [frança, suíça]).
país(madagascar, África, 26, []).
país(portugal, europa, 10, [espanha]).
país(suíça, europa, 8, [frança, alemanha, itália]).
```

E. 5 Escreva um predicado `pop_elevada(Continente, Lista)` que calcule a lista de todos os países com mais de 15 milhões de habitantes de um dado continente, ordenada por ordem crescente de população, no formato indicado.

Exemplos de consulta:

```
?- pop_elevada(europa, Lista).
Lista = [47-espanha, 60-itália, 67-frança, 83-alemanha]

?- pop_elevada(África, Lista).
Lista = [26-madagascar]
```

E. 6 Escreva um predicado `isolados_grandes(Lista)` que calcule a lista, ordenada por ordem alfabética, de todos os continentes que possuem pelo menos dois países que tenham simultaneamente uma população superior a 15 milhões e duas ou menos fronteiras terrestres (com países conhecidos).

Exemplo:

```
?- isolados_grandes(Lista).
Lista = [europa, oceania]
% europa pois possui a espanha e a itália;
% oceania devido à austrália e indonésia.
```

4 Exercícios sobre conjuntos

E. 7 Conjuntos podem ser pensados como listas que não contenham elementos repetidos. Por exemplo, `[a,4,6]` é um conjunto, mas `[a,4,6,a]` não é, pois ele contém duas ocorrências de `a`.

Escreva um programa Prolog `subconjunto/2` que é satisfeito quando o primeiro argumento é um subconjunto do segundo argumento, isto é, quando qualquer elemento do primeiro argumento é um membro do segundo argumento. Por exemplo:

```
?- subconjunto([a,b],[a,b,c]).
true

?- subconjunto([c,b],[a,b,c])
true

?- subconjunto([],[a,b,c])
true
```

Seu programa deveria ser capaz de gerar todos os subconjuntos de um conjunto dado como entrada via retrocesso. Por exemplo, se você der como entrada

```
?- subconjunto(S,[a,b,c]).
```

ele deveria gerar sucessivamente todos os oitos subconjuntos de `[a,b,c]`.

E. 8 Usando o predicado `subconjunto` que acabou de escrever, e `findall`, escreva um predicado `conj_potência/2` que recebe um conjunto como seu primeiro argumento e retorna o conjunto potência deste conjunto como o segundo argumento. O conjunto potência de um conjunto é o conjunto de todos os seus subconjuntos. Por exemplo:

```
?- conj_potência([a,b,c],P).
P = [[],[a],[b],[c],[a,b],[a,c],[b,c],[a,b,c]]
```

Não importa se os conjuntos são retornados em uma ordem diferente da anterior. Por exemplo,

```
P = [[a],[b],[c],[a,b,c],[],[a,b],[a,c],[b,c]]
```

também é válido.

5 Exercícios sobre manipulação de bases de fatos

E. 9 Assuma que se inicie com uma base de dados vazia. Então, dado o comando:

```
?- assert(q(a,b)), assertz(q(1,2)), asserta(q(foo,blug)).
```

O que estará na base de dados agora?

Na sequência é dado o comando:

```
?- retract(q(1,2)), assertz( (p(X) :-  h(X)) ).
```

O que estará na base de dados agora?

Por fim, entre com o comando:

```
?- retract(q(_,_)),fail.
```

O que estará na base de dados agora?

6 Exercício sobre manipulação de bases de fatos

E. 10 Assuma que se tenha a seguinte base de dados:

```
q(blob,blug).
q(blob,blag).
q(blob,blig).
q(blaf,blag).
q(dang,dong).
q(dang,blug).
q(flab,blob).
```

Qual é a resposta do Prolog às seguintes consultas:

(a) `findall(X, q(blob,X), Lista).`

(b) `findall(X, q(X,blug), Lista).`

(c) `findall(X, q(X,Y), Lista).`

(d) `bagof(X, q(X,Y), Lista).`

(e) `setof(X, Y^q(X,Y), Lista).`

E. 11 Escreva um predicado `somatório/2` que recebe um inteiro $n > 0$ e calcula a soma de todos os inteiros entre 1 e n. Exemplo:

```
?- somatório(3,X).
X = 6

?- somatório(5,X).
X = 15
```

Escreva o predicado tal que os resultados sejam guardados na base de dados (é claro que não deveria haver mais que uma entrada por resultado na base de dados para cada valor) e reutilizados sempre que possível. Assim, por exemplo:

```
?- somatório(2,X).
X = 3

?- listing.
res_somatório(2,3).
```

Depois disto, quando realizarmos a consulta

```
?- somatório(3,X).
```

O Prolog não calculará tudo de novo, mas obterá o resultado `somatório(2,3)` da base de dados e somente somará 3 a este resultado. O Prolog então responderá:

```
X = 6

?- listing.
res_somatório(2,3).
res_somatório(3,6).
```

7 Outros exercícios

E. 12 Escreva um predicado `triplas/1` que unifique seu argumento, via retrocesso, com todas as triplas $[X, Y, Z]$ que satisfazem às seguintes condições:

(a) X, Y e Z são diferentes inteiros entre 0 e 9 (os limites estão incluídos)

(b) X, Y e Z satisfazem a equação $\frac{10 * X + Y}{10 * Y + Z} = \frac{X}{Z}$ com precisão infinita.

Por exemplo, suponha que $[3, 5, 9]$ e $[3, 1, 6]$ sejam as únicas soluções (de fato, elas não são!), então a saída deverá ser como segue:

```
?- triplas(Triplas).
Triplas = [3, 5, 9] ;
Triplas = [3, 1, 6] ;
false
```

A ordem das soluções não é importante.

8 Sugestões de leitura

- Leia os capítulos do livro de Eloi Favero referentes aos temas discutidos nesta prática.
- Leia o wikilivro sobre Prolog em `http://pt.wikibooks.org/wiki/Prolog`.