Universidade Federal de Santa Catarina
Campus Araranguá
Engenharia de Computação
ARA7502 – Lógica Aplicada a Computação
Prof. Gustavo Mello Machado

Trabalho Prolog 05 - Aula 10/11/2016

Orientações preliminares.

- É permitida a realização deste trabalho individualmente ou em duplas.
- As entregas serão aceitas exclusivamente via Moodle
- Este trabalho comporá a nota da avaliação E1 como previsto no plano de ensino.

LISTAS

Em Prolog, uma lista é uma sequência linear de itens separados por vírgulas e delimitados por colchetes. Por exemplo, a lista composta pelos iten `a`, `b`, `c` e `d` é representada por

```
[a, b, c, d]
```

Uma lista vazia é dada por `[]` e uma lista com pelo menos um item é representada por `[X|Y]`, onde `X` é o primeiro item da lista e `Y` representa a sublista composta pelos demais itens da lista. Por exemplo

```
?- [X|Y] = [terra, sol, lua].
X = terra
Y = [sol, lua]
Yes

?- [X|Y] = [estrela].
X = estrela
Y = []
Yes

?- [X|Y] = [].
No
```

Para selecionar o terceiro item de uma determinada lista podemos fazer o seguinte:

```
?- [_,_,X|_] = [a,b,c,d,e].
X = c
Yes
```

Aplicando Recursividade em Listas

A recursividade é uma ferramenta muito poderosa para tratar listas em Prolog. Por exemplo, podemos imprimir listas da seguinte maneira

Exemplo 7.1. Imprime lista

```
% imprime(L)
imprime([]) :- nl.
imprime([X|Y]) :- write(X), imprime(Y).
```

Com isto temos um predicado que imprime o primeiro elemento da lista e chama recursivamente a impressão dos elementos subsequentes na sublista restante. A condição de parada acontece quando a lista está vazia. Neste caso é impresso apenas a quebra de linha.

Também podemos aplicar recursividade para identificar se uma determinada lista contém um determinado elemento. Por exemplo

Exemplo 7.2. Lista contém elemento?

```
% contem(X,L)
contem(X,[X|_]).
contem(X,[_|Y]) :- contem(X,Y).
```

Com isto podemos realizar as seguintes consultas:

```
?- contem(c,[a,b,c,d]).
Yes
```

```
?- contem(e,[a,b,c,d]).
No
```

Note que este mesmo predicado poderá ser usado para acessar os elementos de uma lista. Por exemplo

```
?- contem(X,[a,b,c,d]).
X = a ;
X = b ;
X = c ;
X = d
```

Também podemos aplicar recursividade para concatenar duas listas como segue

Exemplo 7.3. Concatenação

```
% concatena(A,B,C): concatena B ao fim de A e coloca resultado em C
concatena([],B,B).
concatena([X|A],B,[X|C]) :- concatena(A,B,C).
```

Neste caso estamos determinando que o primeiro elemento de `A`, será recursivamente o primeiro elemento de `C`, a menos que `A` esteja vazio, neste caso, `C` recebe todo o conteúdo de `B`. Também podemos usar o mesmo predicado concatena para outras finalidades interessantes, por exemplo:

```
?- concatena([a,b],[c,d],L).
L = [a,b,c,d]
```

```
?- concatena([a,b],L,[a,b,c,d]).
L = [c,d]
```

```
?- concatena(X,Y,[a,b,c]).
X = []
```

```
Y = [a,b,c] ;
X = [a]
Y = [b,c] ;
X = [a,b]
Y = [c] ;
X = [a,b,c]
Y = []
```

Ordenamento de Listas

O algoritmo de ordenamento de listas, chamado mergesort, aplica a estratégia de solução de problemas conhecida como divisão e conquista. No caso do mergesort isto se dá da seguinte maneira

1. Divide a lista em duas sublistas de tamanhos aproximados
2. Ordena cada uma das sublistas de maneira independente
3. Combina de maneira eficiente duas sublistas ordenadas em uma lista ordenada

Exemplo 7.4. Mergesort

```
% divisao(L,A,B) : distribui os itens de L em A e B
divisao([],[],[]).
divisao([X],[X],[]).
divisao([X,Y|Z],[X,A],[Y|B]) :- divisao(Z,A,B).

% combina(A,B,L) : combina sublistas ordenadas A e B em L
combina([],B,B).
combina(A,[],A).
combina([X|A],[Y|B],[X|C]) :- X<=Y, combina(A,[Y|B],C).
combina([X|A],[Y|B],[Y|C]) :- X>Y, combina([X|A],B,C).

% ordena(L,S) : ordena lista L colocando resultado em S
ordena([],[]).
ordena([X],[X]).
ordena([X,Y|Z],S) :-
                    divisao([X,Y|Z],A,B),
                    ordena(A,As),
                    ordena(B,Bs),
                    combina(As,Bs,S).
```

Com isto, podemos realizar a seguinte consulta:

```
?- ordena([3,5,0,4,1,2],S).
S = [0,1,2,3,4,5]
```

Primeiramente Prolog separa a lista `[3,5,0,4,1,2]`, que tem pelo menos dois elementos, em duas sublistas `A = [3,5,0]` e `B = [4,1,2]`. A chamada recursiva por ordena vai nos dar `As = [0,3,5]` e `Bs = [1,2,4]`. Por fim Prolog combina o resultado de `As` e `Bs`, comparando sempre o primeiro elemento da lista para determinar a ordem correta de `S`.

EXERCÍCIOS

1) Defina os seguintes predicados
    a) Para se obter o último elemento de uma lista: `ultimo(L,S)`
    b) Para se obter o número de itens em uma lista: `tam(L,N)`
    c) Para se obter a soma dos valores dos elementos de uma lista: `soma(L,S)`
    d) Para se obter o valor do item com valor máximo em uma lista: `max(L,M)`
    e) Para se obter uma lista invertida: `inv(L,R)`
    f) Para se verificar se uma lista é simétrica: `sim(L)`