Programação Declarativa

Licenciatura em Engenharia Informática

2014–2015

Vitor Beires Nogueira

Escola de Ciências e Tecnologia
Universidade de Évora

Diferença de listas

- A sequencia de elementos `1,2,3` pode ser representada como a diferença entre pares de listas:
  - `[1,2,3]-[]`
  - `[1,2,3,4,5]-[4,5]`
  - `[1,2,3,8]-[8]`
- Vamos denotar `As-Bs` como sendo a estrutura que representa a diferença entre as listas `As` e `Bs`
  - `As` é a cabeça e `Bs` é a cauda
  - A escolha do functor (`-`) poderia ser outra, como por exemplo `\`

Append (diferença de listas)

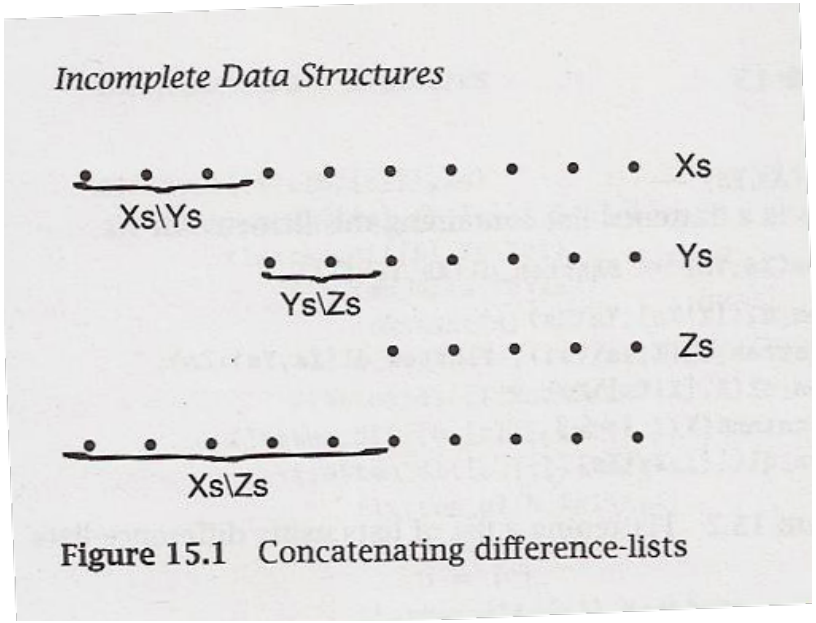

Figure 15.1 Concatenating difference-lists

Código

```
/* append_dl(As,Bs,Cs) :- The difference-list Cs
   is the result of appending Bs to As,where As
   and Bs are compatible difference-lists.
*/
append_dl(Xs-Ys,Ys-Zs,Xs-Zs).
```

Append (diferença de listas):exemplo

Exemplo

```
?- append_dl([a,b,c|Xs]-Xs,[1,2,3]-[],Ys).
Xs = [1, 2, 3],
Ys = [a, b, c, 1, 2, 3]-[]
```

- O append utilizando diferença de lista tem um tempo constante.

Flatten (diferença de listas)

Código

```
flatten(Xs,Ys) :-
  flatten_dl(Xs,Ys-[]).

flatten_dl([X|Xs],Ys-Zs) :-
  flatten_dl(X,Ys-Ys1),
  flatten_dl(Xs,Ys1-Zs).

flatten_dl(X,[X|Xs]-Xs) :-
  atomic(X),
  X \== [].

flatten_dl([],Xs-Xs).
```

Inverte (diferença de listas)

Exemplo (Inverte naive)

```
inverte([],[]).

inverte([X|Xs],Zs) :-
  inverte(Xs,Ys),
  concatena(Ys,[X],Zs).
```

Exemplo (Inverte com diferença de listas)

```
inverte(Xs,Ys) :- inverte_dl(Xs,Ys-[]).

inverte_dl([],Xs-Xs).

inverte_dl([X|Xs],Ys-Zs) :-
  inverte_dl(Xs,Ys-[X|Zs]).
```

Dicionários utilizando estruturas incompletas

Código

```
/* lookup(Key,Dictionary,Value) :- Dictionary
   contains Value indexed under key. Dictionary
   is represented as an incomplete list of pairs
   of the form (Key,Value).*/

lookup(Key,[(Key,Value)|Dict],Value).
lookup(Key,[(Key1,Value1)|Dict],Value) :-
    Key \== Key1, lookup(Key,Dict,Value).
```

- `?- lookup(pedro, [(pedro,123)],X).`
  - `X = 123`
- `?- lookup(pedro, X, 123), lookup(maria, X, 456), lookup(pedro,X,Y).`
  - `X = [ (pedro, 123), (maria, 456)|_G309], Y = 123`

Predicado `findall/3`

- Resolver uma query em Prolog envolve encontrar uma instância da query que é implicada pelo programa.
- Como é que podemos obter todas as instâncias da query que são implicadas pelo programa?
- Programação de segunda-ordem: pretendemos obter o conjunto de elementos com uma determinada propriedade.

Definição

`findall/3` O predicado `findall(+Template, +Goal, -Bag)` cria a lista das instanciações que `Template` obtém sucessivamente por backtracking sobre o `Goal` e unifica o resultado com `Bag`. Sucede com a lista vazia se o `Goal` não tiver soluções.

Exemplo de aplicação do `findall/3`

Código

```
father(terach,abraham).   father(haran,lot).
father(terach,nachor).    father(haran,milcah).
father(terach,haran).     father(haran,yiscah).
father(abraham,isaac).

male(abraham).   male(haran).   male(lot).
male(isaac).     male(nachor).

female(milcah).   female(yiscah).
```

- `children(X,Kids) :- findall(Kid,father(X,Kid),Kids).`
- `?- children(terach,Xs).`
  - `Xs = [abraham, nachor, haran]`

Definição (bagof/3)

O predicado `bagof(+Template, +Goal, -Bag)` unifica `Bag` com as alternativas de `Template`. Se `Goal` tem variáveis livres para além daquelas que partilha com `Template`, então o bagof irá fazer backtracking sobre as alternativas para tais variáveis livres, unificando `Bag` com as correspondentes alternativas para `Template`.
O construtor `+Var^Goal` indica ao bagof que não deve atribuir valor a `Var` no `Goal`.
O predicado `bagof/3` falha caso `Goal` não tenha soluções.

- O `findall/3` é equivalente ao `bagof/3` com todas as variáveis livres ligadas através do operador (`^`), com a excepção que `bagof/3` falha quando o goal não tem soluções.

bagof/3: exemplo

Exemplo

```
?- [user].
|: foo(a, b, c).
|: foo(a, b, d).
|: foo(b, c, e).
|: foo(b, c, f).
|: foo(c, c, g).

?- bagof(C, foo(A,B,C), Cs).
A = a, B = b, Cs = [c, d] ;
A = b, B = c, Cs = [e, f] ;
A = c, B = c, Cs = [g].

?- bagof(C, A^foo(A,B,C), Cs).
B = b, Cs = [c, d] ;
B = c, Cs = [e, f, g].

?- bagof(C, A^B^foo(A,B,C), Cs).
Cs = [c, d, e, f, g].
```

Definição

O predicado `setof(+Template, +Goal, -Set)` é equivalente ao `bagof/3`, com a diferença que ordena o resultado utilizando `sort/2` de modo a obter uma lista ordenada (sem duplicados).

Exemplo

- `?- setof(A, C^B^foo(A,B,C), As).`
  - `As = [a, b, c]`