Exercise Session Informatik III

7. DCG and Sorting

Parser, Syntax in DCG

```
declExpr(Store, Store2)        -->
   decl, ident(Id), equal, bulk(B),
   { create(B,Id,Store,Store2), output(Id, Store2) }.
insExpr(Store, Store2)         -->
   insert, number(N), in, ident(Id),
   { insert(N,Id,Store,Store2), output(Id, Store2) }.
delExpr(Store, Store2)         -->
   delete, number(N), in, ident(Id),
   { delete(N,Id,Store,Store2), output(Id, Store2) }.
haltExpr                       -->
   halt.
identExpr(Id)                  -->
   ident(Id).
opExpr(Result, Store)          -->
   expr(X, Store), operator(Op), expr(Y, Store),
   { calc(Op,X,Y,Store,Result) }.
```

Parser, Syntax in DCG

```
brackExpr(Result, Store)       -->
   openBr, opExpr(Result, Store), closeBr.
expr(Id, Store)                -->
   existingIdent(Id,Store).
expr(Result, Store)            -->
   brackExpr(Result, Store).
existingIdent(Id, Store)       -->
   ident(Id), { id_exists(Id, Store) }.
ident(Id)                      -->
   [Id], { atom(Id) }.
number(N)                      -->
   [N], { number(N) }.
```

Parser, Syntax in DCG

```
bulk(set)                      -->   [set].
bulk(bag)                      -->   [bag].
operator(+)                    -->   [+].
operator(-)                    -->   [-].
operator(*)                    -->   [*].
decl                           -->   [val].
equal                          -->   [=].
insert                         -->   [insert].
delete                         -->   [delete].
```

Parser, Syntax in DCG

```
in                             -->   [in].
halt                           -->   [halt].
openBr                         -->   ['('].
closeBr                        -->   [')'].
```

... and so on!

Insertion Sort

```
insert_sort(List, Sorted) :-
   insert_sort(List, [], Sorted).

insert_sort([], Acc, Acc).
insert_sort([X|Xs], Acc, Sorted) :-
   ord_insert(X, Acc, Acc1),
   insert_sort(Xs, Acc1, Sorted).

ord_insert(X, [], [X]).
ord_insert(X, [Y|Ys], [X,Y|Ys]) :-
   X =< Y.
ord_insert(X, [Y|Ys], [Y|Zs]) :-
   X > Y, ord_insert(X, Ys, Zs).
```

It was Bubble Sort!

```
bubble_sort(List, Sorted) :-
   bubble_sort(List, [], Sorted).

bubble_sort([], Acc, Acc).
bubble_sort([X|Xs], Acc, Sorted) :-
   aux(X, Xs, Ys, Max),
   bubble_sort(Ys, [Max|Acc], Sorted).

aux(X, [], [], X).
aux(X, [Y|Ys], [Y|Zs], Max) :-
   X > Y,
   aux(X, Ys, Zs, Max).
aux(X, [Y|Ys], [X|Zs], Max) :-
   X =< Y,
   aux(Y, Ys, Zs, Max).
```

globis

What does aux/4?

```
| ?- aux(4, [3,5,6], Ys, Max).
      1  1  Call: aux(4,[3,5,6],_398,_422) ?
      2  2  Call: 4>3 ?
      2  2  Exit: 4>3 ?
      3  2  Call: aux(4,[5,6],_953,_422) ?
      4  3  Call: 4>5 ?
      4  3  Fail: 4>5 ?
      4  3  Call: 4=<5 ?
      4  3  Exit: 4=<5 ?
      5  3  Call: aux(5,[6],_2134,_422) ?
      6  4  Call: 5>6 ?
      6  4  Fail: 5>6 ?
      6  4  Call: 5=<6 ?
      6  4  Exit: 5=<6 ?
      7  4  Call: aux(6,[],_3314,_422) ?
?     7  4  Exit: aux(6,[],[],6) ?
?     5  3  Exit: aux(5,[6],[5],6) ?
?     3  2  Exit: aux(4,[5,6],[4,5],6) ?
?     1  1  Exit: aux(4,[3,5,6],[3,4,5],6) ?

Ys = [3,4,5],
Max = 6 ?
```

globis

QuickSort

```
quick_sort([X|Xs], Ys) :-
   partition(Xs, X, Littles, Bigs),
   quick_sort(Littles, Ls),
   quick_sort(Bigs, Bs),
   append(Ls, [X|Bs], Ys).
quick_sort([], []).

partition([X|Xs], Y, [X|Ls], Bs) :-
   X <= Y,
   partition(Xs, Y, Ls, Bs).
partition([X|Xs], Y, Ls, [X|Bs]) :-
   X > Y,
   partition(Xs, Y, Ls, Bs).
partition([], Y, [], []).
```

globis

That's all folks!

Hurra! Die neue Testat Übung ist da! Runterladen und lösen... Abgabetermin: 10. Januar 2002

globis