ListSort.mesa

The ListSort interface contains a single procedure, Sort, that efficiently sorts a LIST of a specific type. The package's implementation uses the "online merge sort" algorithm to sort an n-item list in time O(n log n).

ListSort sorts lists of Item, where Item (and a procedure for comparing two Items) is defined in a definitions module that parameterizes the interface. A user of this package creates a suitable definitions module to parameterize ListSort, then compiles ListSort and OnlineMergeSortImpl.

Last edited by:

MBrown on August 26, 1982 5:29 pm

```
DIRECTORY
  Environment USING [Comparison],
  ParticularList USING [Item, nullItem, Compare];
ListSort: CEDAR DEFINITIONS IMPORTS ParticularList = BEGIN
  Item: TYPE = ParticularList.Item;
  nullItem: Item = ParticularList.nullItem;
  Compare: PRIVATE PROC [l1, l2: LIST OF Item] RETURNS [Environment.Comparison]
    = INLINE { RETURN [ParticularList.Compare[l1, l2]] };
  Sort: PROC [l: LIST OF Item] RETURNS [LIST OF Item];
    Destructive sort of l; returns sorted list containing same items.  Order of equal items is not
        preserved.  Each call to Sort does one CONS from the default zone.
  END.
```

How to use

Compiling the package

This package is compile-time tailorable to a particular application. This tailoring is done without editing the source code of the package's interface or implementation. This is easy if only one version of the package is to be part of the application, and somewhat more involved if two or more versions of the package are to be part of the application. In the former case the procedure is:

(1) create a ParticularList definitions module, which must define Item, nullItem, and Compare as follows:

```
DIRECTORY Environment USING [Comparison], ... ;
ParticularList: DEFINITIONS = BEGIN
  Item: TYPE = <any type specification>;
  nullItem: Item = <any constant Item value;  if Item is a REF type, use NIL >;
  Compare: PROC [l1, l2: LIST OF Item] RETURNS [Environment.Comparison] ... ;
    Compares the two Items contained in l1.first and l2.first (l1, l2 are never NIL)
    Result = less means l1.first < l2.first, etc.  May be defined inline.
  END.
```

(2) Compile the ParticularList module created in step 1.

(3) Compile ListSort (this module).

(4) Compile OnlineMergeSortImpl (the implementation of this module).

(5) Clients of the package use the ListSort.bcd created in step 3, and the application binds in the OnlineMergeSortImpl.bcd created in step 4.

In case of multiple versions of the package within a single application, the different versions of modules

ParticularList, ListSort, and OnlineMergeSortImpl must have distinct bcd names. The different ParticularList source files must also have distinct names. Since in this case the module name <->file name correspondence is not one-to-one, compiler command-line parameterization controls the different versions, as in:

(3') xxxListSort _ ListSort[ParticularList: xxxParticularList]

(4') xxxOnlineMergeSortImpl _ OnlineMergeSortImpl[ListSort: xxxListSort]

A version of this package that sorts LIST s of REF ANY and accepts a comparison procedure at runtime is available through the List interface.

Concurrency

The implementation of this package uses no mutable global data; hence there are no restrictions on concurrent use of the Sort procedure. Naturally it does not work for two processes to attempt to sort the same list at the same time.

Change Log

Created by MBrown on 19-Aug-81 16:13:51

Changed by MBrown on March 10, 1982 3:01 pm

Make interface parameterized by importing the ParticularList interface, instead of by hand-editing.

Changed by MBrown on June 28, 1982 10:22 am

Make interface CEDAR.

Changed by MBrown on August 26, 1982 5:50 pm

Use Environment.Comparison.