SCSK CORPORATION

SCSK

VDMTools

VDM-SL ソートアルゴリズム

How to contact SCSK:

http://www.vdmtools.jp/	VDM information web site(in Japanese)
http://www.vdmtools.jp/en/	VDM information web site(in English)
http://www.scsk.jp/	SCSK Corporation web site(in Japanese)
http://www.scsk.jp/index_en.html	SCSK Corporation web site(in English)
vdm.sp@scsk.jp	Mail

VDM-SL ソートアルゴリズム 2.0

— Revised for VDMTools v9.0.2

目 次

1 はじめに

本書は *VDM-SL Toolbox* で提供される例題の一部であり、 `vdmhome/examples` のディレクトリの中に置かれている。本書ではソートアルゴリズムの数々の仕様を例示するが、これは Toolbox の基本的な機能を紹介するために *User Manual* の *VDM Tools* ガイドツアー の中に用いられている。

2 仕様

最初の例題では、教科書などでよく知られている標準マージソートアルゴリズムを示す。

```
functions

  MergeSort: seq of real -> seq of real
  MergeSort(l) ==
    cases l:
      []      -> l,
      [e]     -> l,
      others  -> let l1^l2 in set {l} be st abs (len l1 - len l2) < 2
                 in
                   let l_l = MergeSort(l1),
                       l_r = MergeSort(l2) in
                    Merge(l_l, l_r)
    end;


  Merge: seq of int * seq of int -> seq of int
  Merge(l1,l2) ==
    cases mk_(l1,l2):
      mk_([],l),mk_(l,[]) -> l,
      others              -> if hd l1 <= hd l2 then
```

```
                              [hd l1] ^ Merge(tl l1, l2)
                            else
                              [hd l2] ^ Merge(l1, tl l2)
    end
  pre forall i in set inds l1 & l1(i) >= 0 and
      forall i in set inds l2 & l2(i) >= 0
```

次の例題では、ソートアルゴリズムの陰仕様を示す。ImplSort 関数を Toolbox においてここで述べるように処理することはできないが、latex ジェネレーター、型チェッカー、構文チェッカー、といった他の VDM-SL ツールならば全 VDM-SL 言語の処理が可能なため、この仕様も処理可能なものである。

```
types

  PosReal = real
  inv r == r >= 0


functions

  ImplSort(l: seq of PosReal) r: seq of PosReal
  post IsPermutation(r,l) and IsOrdered(r);

  IsPermutation: seq of real * seq of real -> bool
  IsPermutation(l1,l2) ==
    forall e in set (elems l1 union elems l2) &
      card {i | i in set inds l1 & l1(i) = e} =
      card {i | i in set inds l2 & l2(i) = e};

  IsOrdered: seq of real -> bool
  IsOrdered(l) ==
```

```
  forall i,j in set inds l & i > j => l(i) >= l(j);
```

以下の例題では、陰関数 ImplSort を陽関数である ExplSort に変更した。ここでは IsPermutation テストをジェネレーター関数に変更している。

```
ExplSort : seq of PosReal -> seq of PosReal
ExplSort (l) ==
  let r in set Permutations(l) be st IsOrdered(r) in r;

Permutations: seq of real -> set of seq of real
Permutations(l) ==
  cases l:
    [],[-] -> {l},
    others -> dunion {{[l(i)]^j |
                       j in set Permutations(RestSeq(l,i))} |
                       i in set inds l}
  end;

RestSeq: seq of real * nat -> seq of real
RestSeq(l,i) ==
  [l(j) | j in set (inds l \ {i})]
pre i in set inds l
post elems RESULT subset elems l and
     len RESULT = len l - 1;
```

最後の例もまた標準アルゴリズムのもので、挿入によるソートの原則に基づいたものである。

```
DoSort: seq of real -> seq of real
DoSort(l) ==
  if l = [] then
    []
  else
    let sorted = DoSort (tl l) in
      InsertSorted (hd l, sorted);

InsertSorted: PosReal * seq of PosReal -> seq of PosReal
InsertSorted(i,l) ==
  cases true :
    (l = [])    -> [i],
    (i <= hd l) -> [i] ^ l,
    others      -> [hd l] ^ InsertSorted(i,tl l)
  end
```