List Operations on Chemical Graphs. 2. Combining Basic List Operations

R. Gautzsch and P. Zinn*

Lehrstuhl für Analytische Chemie, Ruhr-Universität Bochum, Postfach 102 148, D-4630 Bochum 1, Germany

Received May 27, 1992

List operations on chemical graphs are a concept of general interest for the manipulation of chemical structure information. Combining basic list operations is an efficient method for the calculation and generation of molecular descriptors in quantitative structure–property relationship studies. As examples, the rebuilding cycle for constitutions and the list-based implementation of topological indexes are discussed.

INTRODUCTION

In a previous paper,[1] we presented basic procedures of a developed list operating system. Combining these basic procedures is an efficient method to derive more complex procedures for the calculation of molecular descriptors in quantitative structure–property relationship (QSPR) studies. In this paper, we give some examples of combining basic list operations.

As a first example, the rebuilding cycle for constitutions will be discussed. This example will demonstrate that the representation of molecular structure in the form of a line notation or a connection table is equivalent. In further examples the use of molecular paths and molecular spheres is demonstrated for the calculation of topological indexes. The calculated indexes are well-known in the literature, but the way of implementing such calculations using list operations shows some concepts of general interest. The demonstration of combining basic list operations is of importance for the comprehension of applications in QSPR studies that will be discussed in subsequent papers.[2,3]

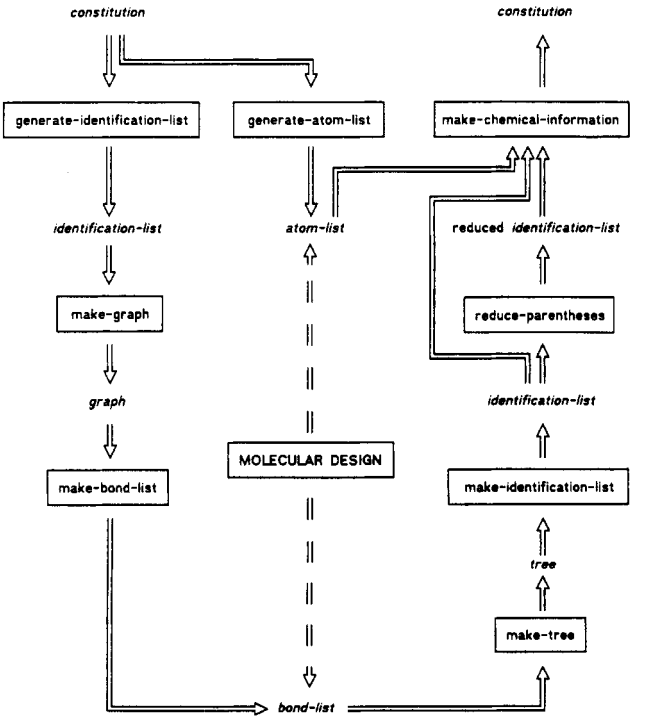

Figure 1. Rebuilding cycle for constitutions using combined basic list operations. Molecular design programs may support the list operating system via bond-lists and atom-lists.

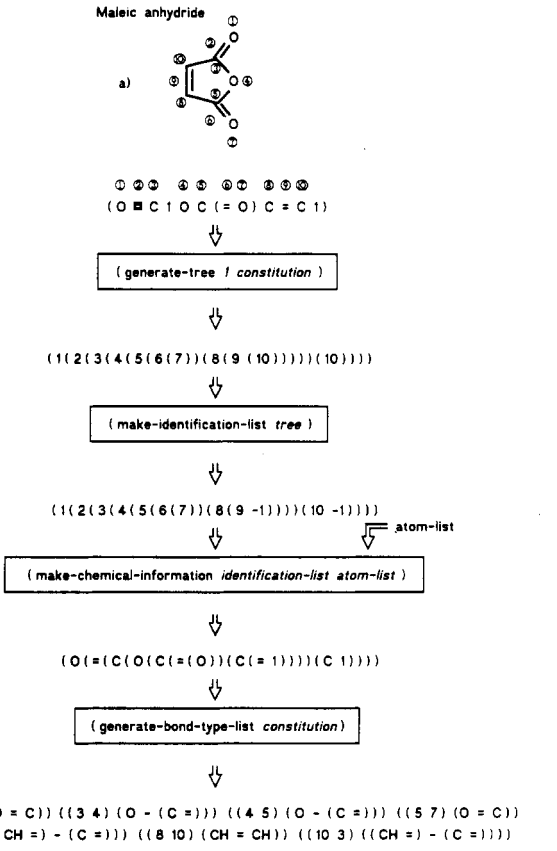

Figure 2. Important steps of a constitution rebuilding for maleic anhydride *without reducing parentheses*. As a proof, the resulting bond-types are identical with those in Figure 3.

REBUILDING CONSTITUTIONS

A typical example for the combination of basic list operations is the rebuilding of constitutions. This is of interest with respect to the development of an interface between the list operating system and the molecular design programs. In cases of handling large molecules, the verification of input constitutions may be simplified using a graphical molecular design editor instead of a constitution line notation. Several molecular design programs store molecular structure information in list-like data files similar to bond-lists and atom-lists.[1] As an example, the molecular design program ALCHEMY (Evans & Sutherland, St. Louis, MO) was taken, and an interface program to support the list operating system with molecular information was developed. The linkage between the list operating system and the molecular design program is shown in Figure 1 in the context of the total rebuilding cycle of constitutions.

0095-2338/92/1632-0551$03.00/0

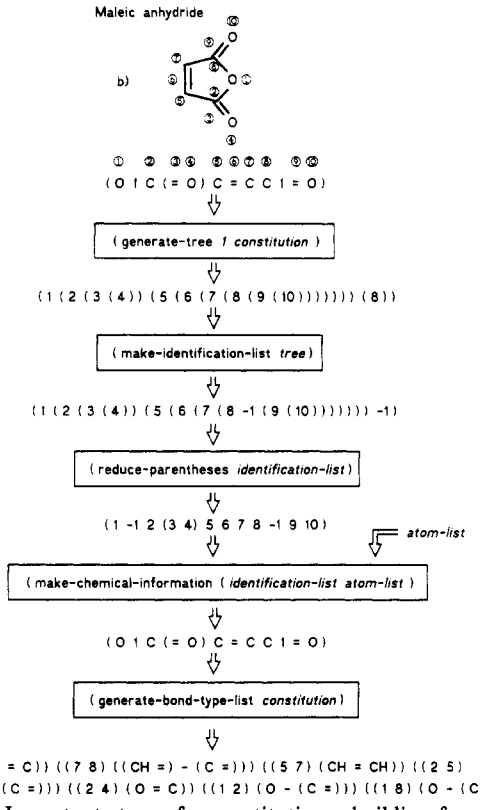

Figure 3. Important steps of a constitution rebuilding for maleic anhydride including reduction of parentheses. As a proof, the resulting bond-types are identical with those in Figure 2.

The rebuilding cycle starts with a constitution notation using basic list operations to build the bond-list stepwise. One possible way to rebuild a constitution from a bond-list is to generate a tree followed by operations reverse to bond-list building. This way of rebuilding does not generate a unique constitution. The resulting constitution depends mainly on the structure of the tree derived intermediately. In order to construct a tree-like data structure, a root element has to be determined. The chosen root element is the first atom of the resulting constitution. Additionally, the tree building depends on the sequence of the bonds in the required bond-list. For these reasons the final constitution notation is not unique.

The identification-list generated from a tree has the same nesting structure as the tree itself with the exception of removed parentheses for ring-bond indicators. To remove the nesting structure as far as possible from such identification-lists, the procedure reduce-parentheses eliminates all tree-characterizing parentheses and keeps all side chain indicating ones. Suppressing tree characterizing parentheses results in an essential acceleration of following list operations. For example, generating a bond-type-list from the input constitution in Figure 2 works about three times faster than starting with the tree-structured constitution from the rebuilding cycle. The speed of execution obviously depends on the nesting structure of lists, which may cause multiple entering of recursive procedures. The proliferation of procedure calls is illustrated by the traces in Figures 4 and 5. The make-chemical-information-procedure transforming identification-lists into constitutions serves as an example. The input identification-list of Figure 4 corresponds to a constitution notation of maleic anhydride without tree-characterizing parentheses. In opposition, the input identification-list of Figure 5 corresponds to a tree-like structured notation of the same compound. The frequency of entering the procedure increases dramatically with the complexity of the input structure. The consequence with respect to the development of a unique constitution notation is to minimize the number of parentheses. An algorithm minimizing the parentheses structure to form a

```
1 Enter MAKE-CHEMICAL-INFORMATION (1 -1 2 (3 4) 5 6 7 8 -1 9 10)
| 2 Enter MAKE-CHEMICAL-INFORMATION (-1 2 (3 4) 5 6 7 8 -1 9 10)
|   3 Enter MAKE-CHEMICAL-INFORMATION (2 (3 4) 5 6 7 8 -1 9 10)
|   | 4 Enter MAKE-CHEMICAL-INFORMATION ((3 4) 5 6 7 8 -1 9 10)
|   |   5 Enter MAKE-CHEMICAL-INFORMATION (3 4)
|   |   | 6 Enter MAKE-CHEMICAL-INFORMATION (4)
|   |   |   7 Enter MAKE-CHEMICAL-INFORMATION NIL
|   |   |   7 Exit MAKE-CHEMICAL-INFORMATION NIL
|   |   | 6 Exit  MAKE-CHEMICAL-INFORMATION (O
|   |   5 Exit MAKE-CHEMICAL-INFORMATION (= O)
|   |   5 Enter MAKE-CHEMICAL-INFORMATION (5 6 7 8 -1 9 10)
|   |   | 6 Enter MAKE-CHEMICAL-INFORMATION (6 7 8 -1 9 10)
|   |   |   7 Enter MAKE-CHEMICAL-INFORMATION (7 8 -1 9 10)
|   |   |   | 8 Enter MAKE-CHEMICAL-INFORMATION (8 -1 9 10)
|   |   |   |   9 Enter MAKE-CHEMICAL-INFORMATION (-1 9 10)
|   |   |   |   | 10 Enter MAKE-CHEMICAL-INFORMATION (9 10)
|   |   |   |   |   11 Enter MAKE-CHEMICAL-INFORMATION (10)
|   |   |   |   |   | 12 Enter MAKE-CHEMICAL-INFORMATION NIL
|   |   |   |   |   | 12 Exit MAKE-CHEMICAL-INFORMATION NIL
|   |   |   |   |   11 Exit MAKE-CHEMICAL-INFORMATION (O)
|   |   |   |   | 10 Exit MAKE-CHEMICAL-INFORMATION (= O)
|   |   |   |   9 Exit MAKE-CHEMICAL-INFORMATION (1 = O)
|   |   |   | 8 Exit MAKE-CHEMICAL-INFORMATION (C 1 = O)
|   |   |   7 Exit MAKE-CHEMICAL-INFORMATION (C C 1 = O)
|   |   | 6 Exit MAKE-CHEMICAL-INFORMATION (= C C 1 = O)
|   |   5 Exit MAKE-CHEMICAL-INFORMATION (C = C C 1 = O)
|   | 4 Exit MAKE-CHEMICAL-INFORMATION ((= O) C = C C 1 = O)
|   3 Exit MAKE-CHEMICAL-INFORMATION (C (= O) C = C C 1 = O)
| 2 Exit MAKE-CHEMICAL-INFORMATION (1 C (= O) C = C C 1 = O)
1 Exit MAKE-CHEMICAL-INFORMATION (O 1 C (= O) C = C C 1 = O)
(O 1 C (= O) C = C C 1 = O)
```

Figure 4. Trace of the make-chemical-information procedure. The identification-list generated from the input constitution of maleic anhydride in Figure 3 serves as input structure.

```
1 Enter MAKE-CHEMICAL-INFORMATION (1 (2 (3 (4)) (5 (6 (7 (8 -1 (9 (10)))))))) -1)
| 2 Enter MAKE-CHEMICAL-INFORMATION ((2 (3 (4)) (5 (6 (7 (8 -1 (9 (10)))))))) -1)
|   3 Enter MAKE-CHEMICAL-INFORMATION (2 (3 (4)) (5 (6 (7 (8 -1 (9 (10))))))))
|   | 4 Enter MAKE-CHEMICAL-INFORMATION ((3 (4)) (5 (6 (7 (8 -1 (9 (10))))))))
|   |   5 Enter MAKE-CHEMICAL-INFORMATION (3 (4))
|   |   | 6 Enter MAKE-CHEMICAL-INFORMATION ((4))
|   |   |   7 Enter MAKE-CHEMICAL-INFORMATION (4)
|   |   |   | 8 Enter MAKE-CHEMICAL-INFORMATION NIL
|   |   |   | 8 Exit MAKE-CHEMICAL-INFORMATION NIL
|   |   |   7 Exit MAKE-CHEMICAL-INFORMATION (O)
|   |   |   7 Enter MAKE-CHEMICAL-INFORMATION NIL
|   |   |   7 Exit MAKE-CHEMICAL-INFORMATION NIL
|   |   | 6 Exit MAKE-CHEMICAL-INFORMATION ((O))
|   |   5 Exit MAKE-CHEMICAL-INFORMATION (= (O))
|   |   5 Enter MAKE-CHEMICAL-INFORMATION ((5 (6 (7 (8 -1 (9 (10)))))))
|   |   | 6 Enter MAKE-CHEMICAL-INFORMATION (5 (6 (7 (8 -1 (9 (10))))))
|   |   |   7 Enter MAKE-CHEMICAL-INFORMATION ((6 (7 (8 -1 (9 (10))))))
|   |   |   | 8 Enter MAKE-CHEMICAL-INFORMATION (6 (7 (8 -1 (9 (10)))))
|   |   |   |   9 Enter MAKE-CHEMICAL-INFORMATION ((7 (8 -1 (9 (10)))))
|   |   |   |   | 10 Enter MAKE-CHEMICAL-INFORMATION (7 (8 -1 (9 (10))))
|   |   |   |   |   11 Enter MAKE-CHEMICAL-INFORMATION ((8 -1 (9 (10))))
|   |   |   |   |   | 12 Enter MAKE-CHEMICAL-INFORMATION (8 -1 (9 (10)))
|   |   |   |   |   |   13 Enter MAKE-CHEMICAL-INFORMATION (-1 (9 (10)))
|   |   |   |   |   |   | 14 Enter MAKE-CHEMICAL-INFORMATION ((9 (10)))
|   |   |   |   |   |   |   15 Enter MAKE-CHEMICAL-INFORMATION (9 (10))
|   |   |   |   |   |   |   | 16 Enter MAKE-CHEMICAL-INFORMATION ((10))
|   |   |   |   |   |   |   |   17 Enter MAKE-CHEMICAL-INFORMATION (10)
|   |   |   |   |   |   |   |   | 18 Enter MAKE-CHEMICAL-INFORMATION NIL
|   |   |   |   |   |   |   |   | 18 Exit MAKE-CHEMICAL-INFORMATION NIL
|   |   |   |   |   |   |   |   17 Exit MAKE-CHEMICAL-INFORMATION (O)
|   |   |   |   |   |   |   |   17 Enter MAKE-CHEMICAL-INFORMATION NIL
|   |   |   |   |   |   |   |   17 Exit MAKE-CHEMICAL-INFORMATION NIL
|   |   |   |   |   |   |   | 16 Exit MAKE-CHEMICAL-INFORMATION ((O))
|   |   |   |   |   |   |   15 Exit MAKE-CHEMICAL-INFORMATION (= (O))
|   |   |   |   |   |   |   15 Enter MAKE-CHEMICAL-INFORMATION NIL
|   |   |   |   |   |   |   15 Exit MAKE-CHEMICAL-INFORMATION NIL
|   |   |   |   |   |   | 14 Exit MAKE-CHEMICAL-INFORMATION ((= (O)))
|   |   |   |   |   |   13 Exit MAKE-CHEMICAL-INFORMATION (1 (= (O)))
|   |   |   |   |   | 12 Exit MAKE-CHEMICAL-INFORMATION (C 1 (= (O)))
|   |   |   |   |   | 12 Enter MAKE-CHEMICAL-INFORMATION NIL
|   |   |   |   |   | 12 Exit MAKE-CHEMICAL-INFORMATION NIL
|   |   |   |   |   11 Exit MAKE-CHEMICAL-INFORMATION ((C 1 (= (O))))
|   |   |   |   | 10 Exit MAKE-CHEMICAL-INFORMATION (C (C 1 (= (O))))
|   |   |   |   | 10 Enter MAKE-CHEMICAL-INFORMATION NIL
|   |   |   |   | 10 Exit MAKE-CHEMICAL-INFORMATION NIL
|   |   |   |   9 Exit MAKE-CHEMICAL-INFORMATION ((C (C 1 (= (O)))))
|   |   |   | 8 Exit MAKE-CHEMICAL-INFORMATION (= (C (C 1 (= (O)))))
|   |   |   | 8 Enter MAKE-CHEMICAL-INFORMATION NIL
|   |   |   | 8 Exit MAKE-CHEMICAL-INFORMATION NIL
|   |   |   7 Exit MAKE-CHEMICAL-INFORMATION ((= (C (C 1 (= (O))))))
|   |   | 6 Exit MAKE-CHEMICAL-INFORMATION (C (= (C (C 1 (= (O))))))
|   |   | 6 Enter MAKE-CHEMICAL-INFORMATION NIL
|   |   | 6 Exit MAKE-CHEMICAL-INFORMATION NIL
|   |   5 Exit MAKE-CHEMICAL-INFORMATION ((C (= (C (C 1 (= (O)))))))
|   | 4 Exit MAKE-CHEMICAL-INFORMATION ((= (O)) (C (= (C (C 1 (= (O))))))))
|   3 Exit MAKE-CHEMICAL-INFORMATION (C (= (O)) (C (= (C (C 1 (= (O))))))))
|   3 Enter MAKE-CHEMICAL-INFORMATION (-1)
|   | 4 Enter MAKE-CHEMICAL-INFORMATION NIL
|   | 4 Exit MAKE-CHEMICAL-INFORMATION NIL
|   3 Exit MAKE-CHEMICAL-INFORMATION (1)
| 2 Exit MAKE-CHEMICAL-INFORMATION ((C (= (O)) (C (= (C (C 1 (= (O))))))) 1)
1 Exit MAKE-CHEMICAL-INFORMATION (O (C (= (O)) (C (= (C (C 1 (= (O))))))) 1)
(O (C (= (O)) (C (= (C (C 1 (= (O))))))) 1)
```

Figure 5. Trace of the make-chemical-information procedure. As input the tree-structured identification-list of maleic anhydride of Figure 3 is transformed to a tree-structured output constitution.

unique constitution is the subject of a paper that is in preparation.[4]

As shown, rebuilding constitutions from different bond-lists and corresponding atom-lists may result in different constitution notations. For many applications of list operations on chemical graphs it is not necessary to use a unique constitution description, especially if an application is based on graph isomorphism or invariants. As an example, it is a trivial fact that different constitution notations for one compound must have identical numbers and types of bonds.

```
(defun generate-paths-count-atom-list (constitution)
  (let ((pure-bond-list                          ; generation of a
                                                 ; temporary
         (generate-pure-bond-list constitution))) ; pure-bond-list
    (mapcar #'(lambda (id-number)    ; loop generating paths-count-atom-list
                (list id-number                  ; association of id-
                                                 ; number
                      (length                    ; with length of
                        (make-atom-paths         ; all paths starting
                                                 ; with
                          id-number              ; id-number
                          pure-bond-list))))     ;
            (mapcar                  ; generation of the list
              'car                   ; of all id-numbers as
              (generate-pure-atom-list ; increments for the loop
                constitution)))))
```

Figure 6. Definition of the procedure generate-paths-count-atom-list as an example for the combination of the basic list operations generate-pure-bond-list, make-atom-paths, and generate-pure-atom-list within a frame of LISP source code.

Table I. Path Counts of Skeletal Atoms of Maleic Anhydride Corresponding to Notation in Figure 2

skeletal-atom	identification-number	path-count
O=	1	12
>C=	3	12
—O—	4	13
>C=	5	12
O=	7	12
—CH=	8	13
—CH=	10	13

This is shown in Figures 2 and 3. In both cases the bond-types in the generated bond-type-lists are identical. They do not depend on the input constitution notation.

In summary, the rebuilding of constitutions shows how to use basic list operations. On the other hand, this example illustrates that the representation of chemical constitutions by a line notation is not contrary to that by adjacency information like bond-lists or also list-based connection tables. Line notations and adjacency lists are two sides of the same coin that are transferable from one into the other.

USING MOLECULAR PATHS

A typical application of using molecular paths in chemistry is the calculation of some types of topological indexes in QSPR studies. An overview about applications and concepts of topological indexes is given by Hall and Kier.[5] The discussed molecular connectivity χ indexes, molecular shape κ indexes, and the topological state for the characterization of skeletal atoms are mostly based on molecular paths.

As a simple demonstration of using paths in a list operating system, the implementation of a paths counting procedure is shown in Figure 6. The procedure generate-paths-count-atom-list counts for each atom in a given constitution the number of paths starting with the specific atom. It internally uses the basic list operations generate-pure-bond-list, make-atom-paths, and generate-pure-atom-list to associate identification numbers with the length of the list of all paths starting with the corresponding atoms. The result is the paths-count-atom-list, which is analogous to an atom-list, where chemical information is substituted by paths-count information. Table I gives an example of atom paths-counts for maleic anhydride with respect to the notation in Figure 2.

```
(defun wiener-index (constitution) ;half the sum of all wiener-atom-
                                   ;indexes
  (/ (apply '+ (mapcar 'cadr (gen-wiener-atom-list constitution))) 2))

(defun generate-wiener-atom-list (constitution)
  (mapcar #'(lambda (id-number)    ; loop generating wiener-atom-list
              (list id-number      ; association of id-number
                    (wiener-atom-index ; with wiener-atom-index
                      id-number      ; (= sum of topological
                      1              ;    distances from one atom
                      constitution   ;    to all other atoms of
                      0)))           ;    the constitution)
          (mapcar                  ; generation of the list
            'car                   ; of all id-numbers as
            (generate-pure-atom-list ; increments for the loop
              constitution))))     ;

(defun wiener-atom-index (id-number distance constitution result)
  (let ((sphere (generate-sphere id-number      ; generation of a
                                 distance       ; temporary sphere to
                                 constitution))) ; derive a partial result
                                                 ; for wiener-atom-index
    (cond ((null sphere) result)                 ; all spheres considered,
          (t (wiener-atom-index                  ; consider next sphere
               id-number                         ; with
               (+ distance 1)                    ; incremented distance,
               constitution                      ;
               (+ result                         ; summarize result of former
                  (* distance                    ; spheres with multiple
                     (length sphere))))))))      ; distance to the elements in
                                                 ; the actual sphere
```

Figure 7. Implementation of a procedure to calculate Wiener topological indexes by using the basic list operations generate-sphere and generate-pure-atom-list. The intermediately generated wiener-atom-list makes atomic-level Wiener indexes available.

The association of skeletal atoms with topological information is a general concept, easy to verify in a list operating system. As another example, the implementation of molecular connectivity χ indexes bases on χ-atom-lists. In this case, skeletal atoms are associated with connectivity information on atomic level. The computation of χ indexes starts with the generation of paths of specified length. Intermediately atomic level ξ indexes of eq 1 are calculated summarizing atomic degrees D over all paths p of length n starting with atom i.

$$ {}^{n}\xi = \sum_{p} (D_i D_j \dots D_{n+1})^{-1/2} \qquad (1) $$

The ξ indexes are associated to the skeletal atoms in the χ-atom-list. The atomic level ξ indexes are useful in molecular structure–physical property relationship studies.[2,3] Comparable atomic indexes are also discussed in the literature.[6] Results of ξ indexes, again for maleic anhydride, are given in Table II. Summation of all atomic indexes results in twice the molecular index with the exception of ${}^{\circ}\chi$, that is directly equal to the sum of ${}^{\circ}\xi_i$.

USING MOLECULAR SPHERES

A sphere is a list of atoms surrounding a central atom in a given topological distance. Using spheres is advantageous in studies that investigate the influence of neighbor atoms on a specific property of a central atom. For example, Diudea, Minailiuc, and Balaban count vertexes in molecular spheres (layers) to introduce the so-called B-matrix for the development of new topological indexes.[7]

In the list operating system, spheres are generated from constitutions by the procedure generate-sphere. The generation of all spheres surrounding a central atom is supported by generate-all-spheres.

Table II. Atomic ξ and Molecular χ Connectivity Indexes of Order 0–5 of Maleic Anhydride Corresponding to Notation in Figure 2

skeletal-atom	identification-number	${}^{0}\xi$	${}^{1}\xi$	${}^{2}\xi$	${}^{3}\xi$	${}^{4}\xi$	${}^{5}\xi$
O=	1, 7	1.000	0.577	0.816	0.524	0.569	0.402
>C=	3, 5	0.577	1.394	0.524	0.569	0.402	0.0
—O—	4	0.707	0.816	1.394	0.408	0.236	0.236
—CH=	8, 10	0.707	0.908	0.986	0.659	0.402	0.118
χ		5.276	3.288	3.023	1.957	1.492	0.638

```
(defun generate-estate-atom-list (constitution)
  (mapcar #'(lambda (id-number)              ; loop generating estate-atom-
                                             ; list,
                    (list id-number          ; association of id-number
                          (estate            ; with corresponding E-state
                            id-number
                            constitution)))
          (mapcar 'car                       ; generation of the list
                  (generate-pure-atom-list   ; of all id-numbers as
                    constitution))))         ; increments for the loop

(defun estate (id-number constitution)       ; definition of estate-
                                             ; procedure,
  (let* ((atom-type-list                     ; generation of a temporary
           (generate-atom-type-list          ; atom-type-list
             constitution))
         (local-intrinsic-value              ; fetching the intrinsic value
           (get-intrinsic-value              ; for the actual id-number
             (cadr (assoc
                     id-number
                     atom-type-list)))))
    (+ local-intrinsic-value                 ; addition of the atom's intrinsic
       (apply '+                             ; value with the perturbation term,
              (make-deltai-list              ; deriving perturbation term's atom
                                             ; interaction list with actual
                                             ; parameters
                local-intrinsic-value
                2
                atom-type-list
                (generate-all-spheres
                  id-number
                  constitution)
                nil))))))

(defun make-deltai-list          ; definition of perturbation term's
                                 ; atom interaction list (deltai-list)
         (local-intrinsic-value
          distance               ; to the actual sphere
          atom-type-list
          all-spheres            ; recursively reduced,
          deltai-list)           ; starts with nil,
  (cond ((null all-spheres) deltai-list) ; all spheres done: give result,
        (t (make-deltai-list             ; else: do recursive operation
             local-intrinsic-value
             (+ distance 1)              ; increment distance to next
                                         ; sphere
             atom-type-list
             (cdr all-spheres)           ; list of remaining spheres,
             (append                     ; expansion of deltai-list
               deltai-list               ; with interactions between
                                         ; central atom and atoms of actual
                                         ; sphere.
               (mapcar                   ; loop building atom interaction
                 #'(lambda               ; list with actual sphere
                     (neighbour-atom)
                     (/ (- local-intrinsic-value
                           (get-intrinsic-value
                             (cadr (assoc neighbour-atom
                                          atom-type-list))))
                        (* distance distance)))
                 (car all-spheres)))))))) ; increments for the loop
```

Figure 8. Implementation of the procedure generate-estate-atom-list calculating the electrotopological indexes of a chemical compound. The procedure is supported by the basic list operations generate-pure-atom-list, generate-atom-type-list, and generate-all-spheres. The procedure get-intrinsic-value fetches the intrinsic value of a skeletal atom from a lookup table.

Table III. Atomic-Level Wiener Indexes for Maleic Anhydride Corresponding to Notation in Figure 2

skeletal-atom	identification-number	atomic-level Wiener index
O=	1, 7	15
>C=	3, 5	10
—O—	4	10
—CH=	8, 10	11
Wiener index		41

As an application of using spheres in the list operating system, the computation of the Wiener index[8] is demonstrated. The Wiener index is defined as half the sum of the elements of a molecule's distance matrix. The implementation using basic list operations starts with the computation of the sum of distances from one atom to all other atoms in the molecule. This is repeated for all atoms and gives atomic-level Wiener indexes that are associated in the Wiener-atom-list. Table III gives examples of atomic-level Wiener indexes again for maleic anhydride. The whole implementation of the Wiener index supported by the basic procedures generate-sphere and generate-pure-atom-list is commented on in Figure 7.

Another more complex example of using spheres is the calculation of the electrotopological index (E-state) described by Hall, Mohney, and Kier.[9] This is an atomic-level index describing molecular topology and electronic properties of the skeletal atoms. The implementation of the E-state calculation is performed in two stages. In a first step, the intrinsic value of the actual atom considering electronic properties is fetched from a lookup table. In a second step, the so-called perturbation term is calculated, that includes the influences of all other skeletal atoms graded to their topological distance to the actual atom. This step is supported by the generate-all-spheres procedure. The whole implementation is shown in Figure 8. As mentioned above the calculated E-states are again associated with the corresponding identification numbers in an estate-atom-list. An example for maleic anhydride is shown in Table IV.

Table IV. Electrotopological Indexes for Maleic Anhydride Corresponding to Notation in Figure 2

skeletal-atom	identification-number	electrotopological index
O=	1, 7	9.92
>C=	3, 5	–0.58
—O—	4	3.97
—CH=	8, 10	1.09

CONCLUSIONS

The discussed procedures demonstrate that combining basic list operations within a small frame of source code is an easy way to implement more complex algorithms and to extend the developed list operating system without great effort. The presented examples are only a small part of the list operating system. They should illustrate some concepts important for applications of the system in QSPR studies.[2,3] These applications showed that it is advantageous to embed the list operations into an object-oriented shell. Object orientation delivers a new perspective of data structure classification and inheritance of list operations from one class to another. Especially, handling a larger amount of compounds requires an efficient data organization and additional procedures such as statistical evaluation of whole data sets or substructure search. A discussion of our experience with object-oriented programming will be the subject of forthcoming papers.

REFERENCES AND NOTES

(1) Gautzsch, R.; Zinn, P. List Operations on Chemical Graphs. 1. Basic List Structures and Operations. *J. Chem. Inf. Comput. Sci.*, preceding paper in this issue.

(2) Duvenbeck, Chr.; Zinn, P. List Operations on Chemical Graphs. 3. Development of Vertex and Edge Models for Fitting Retention Index Data. *J. Chem. Inf. Comput. Sci.*, in press.

(3) Duvenbeck, Chr.; Zinn, P. List Operations on Chemical Graphs. 4. Using Edge Models for Prediction of Retention Index Data. In preparation.

(4) Gautzsch, R.; Zinn, P. List Operations on Chemical Graphs. 5. Unique Constitution Notation. In preparation.

(5) Hall, L. H.; Kier, L. B. The Molecular Connectivity Chi Indexes and Kappa Shape Indexes in Structure–Property Modeling. In *Reviews Computational Chemistry*; Lipkowitz, K. B., Boyd, D. B., Eds.; VCH Publishers: New York, 1991; p 367.

(6) Hall, L. H.; Kier, L. B. A Molecular Connectivity Study of Electron Density in Alkanes. *Tetrahedron* **1977**, *33*, 1953–1957.

(7) Diudea, M. V.; Minailiuc, O.; Balaban, A. T. Molecular Topology. IV. Regressive Vertex Degrees (New Graph Invariants) and Derived Topological Indices. *J. Comput. Chem.* **1991**, *12*, 527–535.

(8) Wiener, H. J. Correlation of Heats of Isomerization and Differences in Heat of Vaporization of Isomeres, among the Paraffin Hydrocarbons. *J. Am. Chem. Soc.* **1947**, *69*, 2636–2638.

(9) Hall, L. H.; Mohney, B.; Kier, L. B. The Electrotopological State: Structure Information at the Atomic Level for Molecular Graphs. *J. Chem. Inf. Comput. Sci.* **1991**, *31*, 76–82.