molecular formulas into existing systems without large manpower commitments. A subset of the techniques used for the much harder problems of structure drawing,[13] and the generation of WLN formulas from connection tables,[14] should meet the demands of structural molecular formula generation.

ACKNOWLEDGMENT

The author wishes to thank Professor F. W. McLafferty for stimulating his interest in the maximal substructural commonality problem. The author thanks Patrick Keogh for timing tests on the UNIVAC 1108. Thanks are also due to Carol Jacobs, Lorraine Scarr, and Glenda Gregor for their help in preparing the manuscript.

REFERENCES AND NOTES

(1) P. N. Craig and H. M. Ebert, "Eleven Years of Structure Searching Using the SK&F Fragmentation Codes", *J. Chem. Doc.*, **9**, 141 (1969).
(2) E. Meyer, "Superimposed Screens for the GREMAS System", Proceedings of the FID-IFIP Conference, Rome 1967, K. Samuelson, Ed., North-Holland, Amsterdam, 1968, p 280.
(3) G. W. Adamson, J. Cowell, M. F. Lynch, A. H. McLure, W. G. Town, and A. M. Yapp, "Strategic Considerations in the Design of a Screening System for Substructure Searches of Chemical Structure Files", *J. Chem. Doc.*, **13**, 153 (1973).
(4) M. Milne, D. Lefkovitz, H. Hill, and R. Powers, "Search of CA Registry (1.25 Million Compounds) with the Topological Screens System", *J. Chem. Doc.*, **12**, 183 (1972).
(5) C. E. Granito, G. T. Becker, S. Roberts, W. J. Wiswesser, and K. J. Windlinx, "Computer-Generated Substructure Codes (Bit Screens)", *J. Chem. Doc.*, **11**, 106 (1971).
(6) E. Hyde in "Computer Representation and Manipulation of Chemical Information", W. T. Wipke et al., Ed., Wiley, New York, N.Y., 1974.
(7) R. G. Dromey, "A Simple Tree-Structured Line Formula Notation for Representing Molecular Topology", *J. Chem. Inf. Comput. Sci.*, submitted for publication.
(8) B. G. Buchanan and J. Lederberg, "The Heuristic Dendral Program for Explaining Empirical Data", Stanford University Computer Science Dept., Report No. CS-203, 1971.
(9) R. G. Dromey, "A Highly Compressed Inverted File for Molecular Formula and Homologous Series Searching of Large Data Bases", *Anal. Chem.*, **49**, 1982 (1977).
(10) N. Wirth, "Algorithms + Data Structures = Programs", Prentice-Hall, Englewood Cliffs, N.J., 1976.
(11) F. W. McLafferty, private communication.
(12) D. G. Corneil and C. C. Gotlieb, "An Efficient Algorithm for Graph Isomorphism", *J. Assoc. Comput. Mach.*, **17**, 51 (1970).
(13) E. J. Corey and W. T. Wipke, "Computer Assisted Design of Complex Organic Syntheses", *Science*, **166** 178-192 (Oct 10, 1969).
(14) G. A. Miller, "Encoding and Decoding WLN", *J. Chem. Doc.*, **12**, 60 (1972).

Computer-Assisted Simulation of Chemical Reaction Sequences. Applications to Problems of Structure Elucidation[1,2]

TOMAS H. VARKONY, RAYMOND E. CARHART, DENNIS H. SMITH,* and CARL DJERASSI

Departments of Chemistry and Computer Science and Genetics, Stanford University, Stanford, California 94305

Received February 14, 1978

An interactive computer program (REACT) for simulation of chemical reaction sequences and its application to problems of structure elucidation are described. The program is supplied with information about laboratory operations such as chemical reactions, separations, and data about structural features of the products. The program applies this information to structures in the computer memory and allows a chemist to examine the results by displaying the reaction/separation sequence or drawings of structures. A sequence demonstrating the use of REACT is illustrated for the structure elucidation of palustrol (**1**). A detailed description of some of the algorithms is presented.

Chemical reactions are fundamental tools in the study of molecular structure far beyond obvious applications to synthetic organic chemistry. Other areas of research where chemical reactions play a crucial role include mechanistic studies, e.g., cyclizations and rearrangements, and a wide variety of reactions applied to unknown structures during the course of structure elucidation, e.g., functionalization, derivatization and degradation, or simplification reactions. Modern instrumentation, including X-ray crystallography, mass spectrometry, and ^{13}C NMR spectroscopy have dramatically altered the methodology of structural studies in both mechanistic organic chemistry and structure elucidation. There remain, however, many structural problems where knowledge gained from the results of applications of chemical reactions is crucial to determining mechanistic pathways or structural identity.

We have developed an interactive computer program, called REACT,[3,4,6] to help a chemist to explore certain aspects of chemical reactions applied to representations of molecular structure. REACT was designed to provide a general tool for studying the results and implications of chemical reactions. In structure elucidation problems such results can reduce further the number of possible structures for an unknown, thus helping the chemist to focus attention on the correct structure. In mechanistic studies it is particularly useful to determine exhaustively all possible interconversion pathways and intermediates, or to decipher all structural implications of measurements on products subsequent to an extended sequence of reactions.[3] The reactions typically utilized in such problems are relatively general (i.e., they apply in a wide variety of structural contexts) and well understood. Although many are relatively simple reactions (hydrogenation, hydrolysis, Wagner–Meerwein-type rearrangements), REACT provides the capability for definition of reactions of considerable complexity which can be constrained in several ways to express details of the structural context in which they apply.

In addition we can use REACT for structure generation. For example, if an unknown was obtained from a known starting material through a set of well-defined reactions, REACT can generate the products obeying the constraints of the reactions. These products then are candidate structures for the unknown. Thus REACT provides a "computer laboratory" in which representations of chemical reactions can be applied to structures in ways which parallel actual experimental studies or plans.

The REACT program is a logical extension[3] of the CONGEN program for computer-assisted structure elucidation.[5] CONGEN, by generating all possible structures consistent with given physical or chemical data, provides candidate structures for an unknown, or potential precursors for mechanistic studies. REACT provides a method for subjecting these structures (in the computer) to sequences of chemical reactions.

There are several other programs designed to represent chemical reactions in a computer. These programs were written to assist chemists in the planning of organic synthesis.[7,8] We have previously compared the similarities and differences of REACT as compared to programs for synthesis planning.[3]

The first version of REACT and its applications were described previously.[3] Subsequently, the structure of the program was revised significantly to include commands and internal operations which more closely parallel laboratory procedures. The new version has been described briefly[4] and some applications of REACT to mechanistic problems have been discussed.[6] In subsequent sections we describe the REACT program in detail, together with an example of the application of the program to a structural problem.

METHOD AND EXAMPLE

The discussion is divided into major sections corresponding to the sequence of obtaining starting materials, reaction definition, carrying out a reaction, maintaining a record of results (the "reaction tree"), separating products, and applying constraints based on measurements on products. The introduction to each section can be read alone for a summary of the program's function. Each section contains, in addition, a more detailed description of how REACT operates together with actual dialog with the program illustrating how the example was solved. Appendices I and II summarize the available commands in REACT. This summary illustrates the scope of the program and provides a more detailed description for interested readers. The necessity of interfacing REACT to CONGEN together with need for structural manipulations similar to those in CONGEN (e.g., graph matching, substructure definition) suggested use of the same structural representation. Thus, communication with CONGEN is achieved via files of structures which represent starting materials for REACT.

Starting Material. Candidate structures generated by CONGEN, and saved on a file, are used in REACT by requesting the file (see RESTORE and GET commands, Appendix I). If requisite structures are not available via CONGEN, they can be entered manually by a simple structure editor (PRECURSOR command). This structure editor (which is similar to the structure editor in CONGEN[5]) uses structural manipulation commands (Appendix II) which allow the chemist to define, interactively, structures of arbitrary complexity. This method is based on techniques developed by Feldmann.[9] In entering large number of similar structures the chemist can define for his convenience the largest common substructure (common skeleton) and modify it in a few steps for each individual starting material.

Reaction Definition. Unlike most synthesis programs[7,8] REACT allows interactive definition and immediate application of a reaction without intervening steps. Reactions can be tailor-made according to the wishes of the chemist using the program. Reactions are entered by a routine called EDITREACT. Our representation of reactions consists of (a) the name, (b) the reaction site, (c) the reaction transform, and (d) constraints on application of the reaction.

(a) Name. Reactions in REACT are referred by names. Originally there is no internal library. However, a chemist using REACT can easily define and maintain his private library of reactions (see SAVE and RESTORE commands, Appendix I).

(b) Reaction Site. The reaction site includes those atoms and their relationships which are required to be present in the structure before reaction can take place. In chemical terms sites are often referred to as functional groups or active sites. Obviously, the site usually consists of only those atoms participating in or affecting the course of a reaction.

Table I. Commands for Definition of the Site of the Dehydration Reaction

Command[a]	Comment
#EDITREACT	Enter EDITREACT
REACTION NAME:DEHYDRATION	Name the reaction
*SITE	Begin definition of the reaction site
>CHAIN 3	Create a chain of three carbon atoms
>ATNAME 3 O	Name one to be oxygen
>HRANGE 1 1 3 3 1 1	There must be at least one hydrogen on C-1 and on the oxygen
>ADRAW	Draw the site
DEHYDRATION:(HRANGES NOT INDICATED)	
C-C-O	
>DONE	Finish definition of site - return to EDITREACT
*	Ready for next command

[a] Underlined text is supplied by the chemist. The remainder comes from REACT.

For illustration, we present how a simple dehydration reaction, used in the example below, might be defined using EDITREACT. The sequence of commands used to specify the site is shown in Table I with annotations. These are the same commands which are used for defining substructures in CONGEN or starting materials in REACT (Appendix II). In addition to regular features such as rings, chains, and atom names, we can specify features like variable bond orders or atom names, the aromatic nature of atoms, charged or radical species, relative quantities of hydrogens or π electrons on atoms, and so forth (Appendix II).

The atom numbers associated with atoms in the site are arbitrary; what they are depends only on the order of commands used to construct the site. Once the association is made, however, it is fixed throughout the remainder of the reaction definition to provide a frame of reference for the atoms involved in the transform and constraints.

(c) Reaction Transform. The reaction transform consists of the structural manipulations which convert the precursor into a product. The chemist, on specifying the transform, begins with the entire site (Table I) as a starting point. The set of commands which transforms the site into the product (Table II) uses the atom numbers of the site as a reference numbering. On application of a reaction (see below), these commands are applied to atoms in the site in a complete structure to yield the product. Note that the transform specifies only atom and bond modifications which formally represent the reaction, independent of mechanistic or stereochemical considerations.

(d) Reaction Constraints. REACT allows specification of several types of constraints which restrict the structural context in which a reaction will apply. The constraints available include forbidden and allowed substructural features, ring sizes, and proton distributions.[11] The same structural manipulation commands (Appendix II) are used during definition of substructural constraints. The constraints, which may apply to structures or to site/transform substructures, and their interpretation are as follows:

1. Starting Material Constraints. These are structural features of the starting material which affect the reaction. In most cases these are features which are remote from the reaction site but which cause the reaction to fail or follow another course, e.g., competing functionalities. For example,

Table II. Commands for Definition of the Transform for the Dehydration Reaction

Command	Comment
*TRANSFORM	Begin definition of transform
>NDRAW	Draw the previously defined reaction site
DEHYDRATION:(HRANGES NOT INDICATED)	
NON-C ATOMS: 3->O	
1-2-3	
>UNJOIN 2 3	Break the C-O bond
>JOIN 1 2	Form C-C double bond
>DELATS 3	Remove the oxygen atom (effectively as water)
>NDRAW	Draw the transformed site
DEHYDRATION:(HRANGES NOT INDICATED) 1=2	
>DONE	Finish definition of transform - return to EDITREACT
*	Ready for next command

the presence of a free carboxylic acid functionality can interfere with $LiAlH_4$ reduction of ketones.

2. Site-Specific Constraints. These are structural features of the starting material involving atoms of the reaction site which affect the course of the reaction, usually by preventing it. For example, one could define a reaction of a double bond and include a site-specific constraint which forbade the reaction from occurring at a tetrasubstituted double bond.

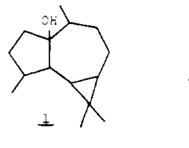
3. Transform-Specific Constraints. These are structural features of the product which involve atoms of the transformed site and which are usually undesired. Such constrains may be used, for example, to prevent introduction of undesired combinations of functionalities which only become obvious after the transform has been carried out. For example, an elimination reaction which produces a double bond can be prevented where unstable allenes would be formed.

4. Product Constraints. These are structural features of the product which are usually undesired. These constraints are reserved for reaction-specific constraints, such as forbidding introduction of a double bond at a bridgehead.

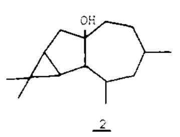
In practice, most constraints are negative statements, reflecting the fact that given the site the reaction will occur in all but a few, well-defined circumstances. In the example dehydration reaction (below), we used no constraints. This decision was partly motivated by knowledge that there were no competing functionalities,[10] other than the propensity for terpenoid systems (see following discussion) to undergo skeletal rearrangements. A more general definition of a dehydration reaction would include information on interfering functionalities.

Carrying Out a Reaction. Once a reaction has been defined it can be applied to a set of starting materials. Initially, the starting materials are placed automatically in a flask called STRUCS. These structures may be obtained as summarized under Starting Material, above.

To demonstrate the application of REACT we choose an example which illustrates some (but not all) aspects of the use of REACT in a structure elucidation problem. A contrived example might illustrate many of the other features and subtleties of the program, but would not be as meaningful chemically. The example involves a dehydration reaction (see reaction definition) applied during the course of elucidation of the structure of palustrol[10] (**1**). Consideration of all



1



2

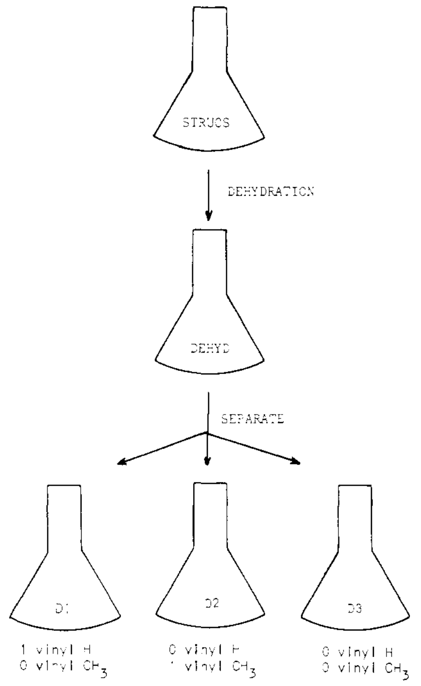


Figure 1. A summary of laboratory observations on the dehydration of palustrol (**1**) and subsequent separation of products and acquisition of strucural data.

Table III. Commands to Carry Out the Dehydration Reaction on Structures in the Flask STRUCS

Command	Comment
#REACT	Enter reaction mode
FLASK NAME OF STARTING MATERIAL:STRUCS	Reaction to be applied to flask STRUCS
REACTION NAME(S):DEHYDRATION	Select reaction(s)
NUMBER OF STEPS:1	Single step (see text for options)
FLASK NAME FOR PRODUCTS:DEHYD	Products will be placed in the new flask DEHYD
241 PRODUCTS WERE OBTAINED	Report of results
#	Return to REACT

available spectroscopic data had reduced the problem to a set of 88 candidates (which were previously generated by CONGEN[10]) prior to carrying out the dehydration reaction. Structural features of the products were powerful constraints on the identity of the compound. This example is of interest because it represents a case where direct translation of observations on products back to structural constraints on the starting materials is difficult.[3] Using REACT, expression of structural information is straightforward and logical. The laboratory reaction, separation, and key structural information are summarized in Figure 1. The starting materials, in a flask called STRUCS, are the candidate structures for palustrol (**1**). The contents of the flask STRUCS were dehydrated and the products placed in a flask called DEHYD. Separation of the reaction mixture yielded three products, placed in flasks D1, D2, and D3. The numbers of vinyl protons and vinyl methyl groups detected by ^1H NMR for each product are summarized in Figure 1.

The dialog for carrying out the example dehydration reaction (Table III) with REACT summarizes the important parameters. Reactions are applied to the contents of a specified flask (Table III). A flask can contain the primary structure list or any of the products lists which were obtained

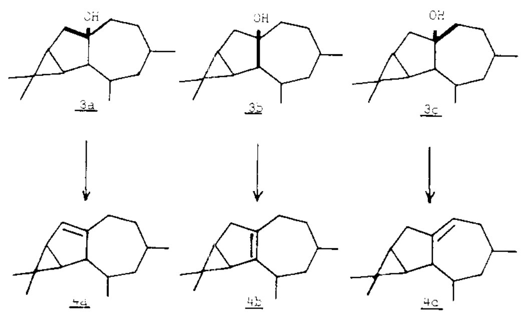
Table IV. Possible Specifications of Number of Steps for a Given Reaction(s) and Their Interpretation

Number of Steps	Interpretation
1	Single step reaction.
0-1	Incomplete reaction - include the starting material in the products flask.
EX	Exhaustive reaction - continue applying the reaction until all sites are consumed.
EQ	Equilibrium reaction - continue applying the the reaction until no new products are obtained.
0-EQ	Same as E, but include the starting material in the product flask.

by previously applied reactions or separations (see subsequent section).

The program applies any previously defined reaction or set of reactions. The interpretation of application of a set of reactions is to apply each of the reactions to each of the structures (according to the number of steps) and collect all products in the given flask. Thus, we can emulate competing reactions. The product flask will contain the products obtained from all of these competing reactions.

There are several possible ways to apply a reaction (or a set of competing reactions) to a set of structures (Table IV). There are several important aspects of the internal (to the computer) representation and application of reactions. REACT applies reactions in a way closely related to the method used in synthesis planning programs.[7,8] Structures are examined for the presence of the reaction site (by graph matching[11]). Single step reactions assume sufficient reagent to convert *one* reaction site per precursor into product. If there are multiple sites, the reaction is applied in turn to each site which obeys the constraints. For example, structure **2**, one of the 88 candidate structures for palustrol (**1**), is found to possess three reaction sites (see Table I) indicated in structures **3a**, **3b**, and **3c**. The reaction transform (see Table II) is applied to each site to yield the indicated three products, **4a**, **4b**, and **4c**.



This representation of single-step reactions predicts *n* products for *n* reaction sites (in case of symmetrical sites or symmetrical reacting molecules, the number of unique products is smaller than the number of sites). If additional reaction sites remain after one step (we assume that there is enough reagent to transform only one site), further steps are *not* carried out (see exhaustive reaction, below); the program is in the same quandry as the chemist who has supplied insufficient reagent to consume all reaction sites and is left with a mixture. In the example, **4a**–**4c** could not react further; no sites remain after one step. For a set of competing reactions, each reaction is selected in turn and the same procedure is repeated. Constraints are tested at several steps of the procedure.

Application of an exhaustive reaction(s) assumes the presence of sufficient reagent to transform all reaction sites.

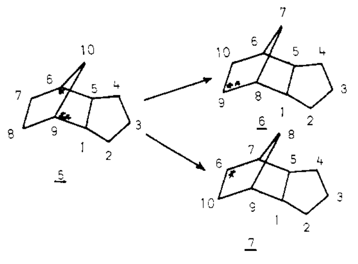


Figure 2. Some 1,2-alkyl shifts in tetrahydrodicyclopentadiene **5**.

A variant of the algorithm is used so that products from the single-step reaction are subjected to reaction again and the process continued until all reaction sites are consumed. Only final products (which contain no remaining active sites) are collected in the products flask.

Specification of an equilibration reaction(s) results in repeated, stepwise application of the set of reactions to the products of the previous step until no more new structures are obtained. REACT keeps a record of all structures encountered during this stepwise application, but not of all pathways between them because of the potentially large size of such record and the presence of many circular pathways. (A record of pathways can be reconstructed if needed by applying the same reaction one step at a time until no new structures are obtained.)

REACT modifies the structural representation of all products by converting them to a canonical[12] form so that duplicate structures can be detected and removed. The information about the origin of the removed duplicate structures is kept together with one selected canonical representation. This process conserves computer memory storage. The canonicalization process modifies the association of atoms and their respective numbers so that structures which are topologically equivalent but differ in the numbering of their atoms are converted to structures with the same numbering. This process is justified chemically because without isotopic labeling there is no way to differentiate between structures which are topologically equivalent (not considering stereochemistry).

For certain mechanistic studies, however, it is important to follow the fate of individual atoms in the structure during a reaction. One way to obtain this information in the laboratory and in REACT is to label the atoms (isotopic labeling). Another way (which enables us in REACT to follow the fate of *all* atoms at the same time) is to preserve the association of atoms and their respective numbers. MREACT (Appendix I) is a command similar to the command REACT, with the only difference being that products are preserved which are topologically equivalent to other structures but which differ in the numbering of atoms. Consider, as a brief example (a detailed example of application of MREACT is given in ref 3), possible 1,2-alkyl shifts in tetrahydrodicyclopentadiene **5**, under constraints forbidding formation of three and four-membered rings and methyl groups (Figure 2). The shift of bond 1–9 from carbon number 9 to carbon number 8 creates structure **6**, and the shift of bond 5–6 from carbon number 6 to carbon number 7 creates structure **7**. Structures **5**–**7** are topologically and chemically equivalent; however, by labeling carbons 6 and 9 they become distinguishable. Thus, using MREACT, one can follow the fate of every atom (e.g., to follow an isotopic label) in an extended sequence of reactions simply by following its number in the original structure.

The different ways of applying reactions (REACT or MREACT), the selection of different numbers of steps, together with the ability to apply one or more reactions at each step (above) cover all common applications of chemical reactions to structures.

The Reaction Tree. The reaction tree is a representation of the sequence of laboratory procedures (reactions and

```
STRUCS=88
|
*DEHYDRATION->DEHYD=241
```

Figure 3. Teletype output of the reaction tree (DISPLAY command, Appendix I) subsequent to dehydration of the 88 candidate structures for plaustrol (**1**).

separations) to which precursors and their products have been subjected. Formally, it consists of named flasks and their interrelationships in the form of reaction names and separation steps. If there are multiple precursors (i.e., more then one structure in a flask), as in the example, each is allowed to react, independently, resulting in a data structure internal to REACT which records the reactions of each structure separately. The chemical meaning of multiple structures in the starting material flask STRUCS is that the exact identity of the compound is not known; its structure is represented by one of all the possible structures in the flask. If the flask was created via a reaction(s), the structures represent the collection of all products from all precursors where, again, the identity of each of the products in the laboratory application of the reaction is not necessarily known. In our representation, an example of which is shown in Figure 3, flasks which could possess multiple structures, such as multiple candidates for an unknown, are depicted as containing all structures, and all possible products appear lumped together in a product flask. The dehydration reaction applied above (see Table III) is summarized in Figure 3. This figure is interpreted to mean that the 88 candidate structures, any one of which could be the true unknown in the flask STRUCS, yield a total of 241 possible products, all associated with the flask DEHYD. Remember that the internal representation is effectively *n* copies of the reaction tree where *n* is the number of precursors in the flask STRUCS, or 88 for the example of Figure 1. For example, one such copy encodes the information about the conversion of **3** to **4a**–**4c**.

In our example we discuss only a single reaction. In general, however, the reaction tree can be of arbitrary complexity. Several different reactions can be applied to aliquots of a precursor (whether it be an original starting material or a product of a previous reaction). In addition, an extended sequence of reactions can be carried out. Thus, the reaction tree can grow arbitrarily in width and depth.

Separation. A flask obtained by reaction can contain a mixture of products. A single precursor can yield multiple products in three ways in a reaction: (1) presence of multiple reaction sites, each yielding a different product; (2) multiple reactions; and (3) cleavage reactions where all fragments are isolable. The usual laboratory step subsequent to reaction is separation of the products. Thus, REACT has a SEPARATE command which allows the chemist to express to the program his laboratory observations on performing the separation. The number of products obtained on separation is a constraint on the identity of the starting material and is information useful in applications of REACT to structural problems. The separation requires placement of each separated product into a designated or named flask (Table V).

It is characteristic of many laboratory reactions that an unspecified, perhaps large, number of additional products are obtained, some legitimate, but at low concentration, others from side reactions which may not be incorporated in the definition of the reaction used in REACT. The chemist using REACT must base his use of the SEPARATE command on his own evaluation of the reaction applied in the laboratory. Selection of a named flask in which to place a separated product implies that the product so separated arose from the named reaction, and not from some other unspecified reaction. However, to accommodate the fact that the reaction may have been incomplete or side reactions may have occurred, additional products can be specified to be in a "tar" flask associated with each set of separated products. On separation, the new flasks each contain one unique product, whose identity is not known. The structure of the product must be one of the structural possibilities associated with the flask. However, the structures in the "tar" flask (or in any flask prior to separation) can be a mixture of products, where each product in the mixture may be represented by several structural possibilities.

```
STRUCS=72
|
*DEHYDRATION->DEHYD=210-s-|D3=210
                          |
                          |D2=210
                          |
                          |D1=210
```

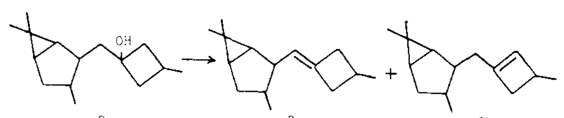
Figure 4. The reaction tree subsequent to separation of the reaction products in flask DEHYD into three flasks D1, D2, and D3.

Table V. Dialog with REACT on Separation of Contents of Product Flask DEHYD into Flasks D1, D2, and D3

Command	Comment
#SEPARATE	Enter separation mode
NAME OF FLASK TO BE SEPARATED:DEHYD	Select product flask
NEW FLASK NAME:D1	Select names for flasks
NEW FLASK NAME:D2	for three separated products
NEW FLASK NAME:D3	
NEW FLASK NAME:	No other flasks
MAXIMUM NUMBER OF ADDITIONAL PRODUCTS:0	No additional products in the tar flask
210 STRUCTURES SURVIVED SEPARATION	Results
BEGINNING RAMIFICATION...DONE	Implications of separation
#	Return to REACT

The dialog to establish separated products and a tar flask with REACT is summarized in Table V. In the laboratory, separation yielded three products[10] (Figure 1). In this example we choose to specify exactly three products by selection of three flasks to receive the products, D1, D2, and D3, and no other.

The fact that three products, all assumed to arise from the dehydration, were observed is a constraint on the identity of the starting material in the flask STRUCS (Figure 3). Those structural possibilities (according to CONGEN) for palustrol,[10] which would yield only two products (e.g., **8**, to yield **8a** and **8b**) can be rejected independently of the identity of the products, while those structures which yield three products on dehydration remain under consideration (e.g., **1** and **2**) until additional data on the identities of the products are gathered and specified to REACT (see subsequent section). The



8 8a 8b

reaction tree which results from the separation (Table V) is shown in Figure 4.

The reduction in the numbers of structures in flasks STRUCS and DEHYD (compare Figure 4 with Figure 3) results from the implications, or "ramifications", of the statement on separation. REACT has a record of how many products are obtained from each structure and the identities of each precursor and product. It can eliminate automatically (see Appendix III) from further consideration precursors which yield an undesired number of products. If three products are observed, as in the example, only 72 of the original 88 structures remain as candidates. Sixteen of the structures yielded, by the computer program, other than exactly three products and were therefore removed from consideration as

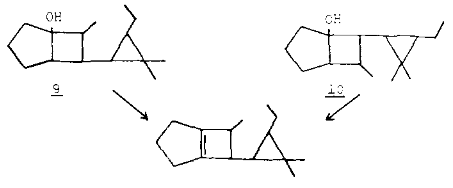
```
STRUCS=72
!
*DEHYDRATION->DEHYD=210-s-!D3=187
                          !
                          !D2=187
                          !
                          !D1=129
```

Figure 5. The reaction tree after constraining the contents of flask D1 to include those products possessing one vinyl proton and no vinyl methyls.

```
STRUCS=45
!
*DEHYDRATION->DEHYD=135-s-!D3=76
                          !
                          !D2=52
                          !
                          !D1=69
```

Figure 6. The reaction tree after constraining the contents of flask D2 to include those products possessing one vinyl methyl and no vinyl protons.

candidates. The products of these sixteen structures are also removed from the product flasks, resulting in a decrease in the number of structures in DEHYD from 241 to 210. The remaining 210 structures are not exactly three times 72 because several candidates yield equivalent products. For example, the dehydration of both **9** and **10** yields, among other products, **11**. As mentioned previously, duplicate structures



9 10 11

are detected and removed for efficiency, except in mechanistic reactions.

What of the contents of the flasks D1, D2, and D3? Up to this point, no statements about the structural identity of any product have been made, paralleling the laboratory events of, first, separation, and, later, gathering of data on the products. Thus, any of the 210 products in DEHYD might be in any of the flasks D1–D3 (see Figure 4, where all 210 products remain allocated to D1–D3). Stated at the level of internal representation in REACT (see also above discussion), where the original structures are represented individually, each structure (in STRUCS) yielded three products, any of which might be in any flask. Subsequent operations will perform the appropriate allocations of structures to flasks.

Details of the internal representation and the algorithm which performs ramification after SEPARATE and PRUNE (see below) are given in Appendix III. This algorithm is responsible for determining legal allocations for structures to flasks throughout the reaction tree whenever the tree is modified in any way.

PRUNE. Expression of Constraints on Products. In laboratory procedures, the next step would be to collect data on the product in each flask. Structural information gained represents constraints not only on the identity of the products, but also on the identity of the precursor and its precursor and so forth throughout an entire reaction sequence. REACT allows structural statements to be made as constraints on the contents of any flask in a reaction tree. The command to express constraints is PRUNE (a word which is jargon but does carry with it the concept of trimming the reaction tree and also corresponds to the same command in CONGEN[5]).

Substructural constraints can be obtained from a file or defined by the chemist as required, using EDITSTRUC (Appendix I). In our example, the product in one of the flasks (D1) was observed according to ^1H NMR analysis to possess one vinyl proton and no vinyl methyl groups. These substructures, PT1 (**12**) and VINM (**13**), respectively, were

H - C = C
12(PT1)

CH_3 - C = C
13(VINM)

Table VI. Use of PRUNE to Express the Constraint That Product Flask D1 Possesses One Vinyl Proton (PT1) and No Vinyl Methyls (VINM)

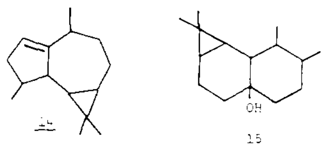
Command	Comments
#PRUNE	Enter pruning mode
NAME OF FLASK TO BE PRUNED:D1	Choose flask
SHALL I USE THE GLOBAL CONSTRAINT LIST:	Predefined constraints list? No
YOU MAY ENTER A TEMPORARY LIST BELOW:	Define a temporary list
>PROTON	Select PROTON constraints
PROTON CONSTRAINTS	
CONSTRAINT NAME:PT1	Select the substructure by name
MINIMUM NUMBER:1	Exactly one
MAXIMUM NUMBER:1	
CONSTRAINT NAME:	No others

>BADLIST	Select BADLIST constraints
BADLIST CONSTRAINTS	
CONSTRAINT NAME:VINM	Select substructure by name
CONSTRAINT NAME:	No others

>DONE	No more constraints
129 STRUCTURES SURVIVE PRUNING	Results
BEGINNING RAMIFICATION...DONE	Implications of pruning
#	Return to REACT

defined and the substructures supplied to PRUNE as shown in Table VI.

The reaction tree which results on application of PRUNE (Table VI) is shown in Figure 5. There remain, under the constraints of Table VI, 129 structures which could be in the flask D1. The number of structural candidates (72) has not been reduced, implying that *all* 72 can yield *at least* one structure possessing one vinyl proton and no vinyl methyls. Some candidate structures can yield more than one product which obeys these constraints and might therefore be in D1, resulting in 129 rather than only 72 structures in that flask. For example, **2** yields two products obeying the constraints, **4a** and **4c**; either could be the product observed in D1. However, for structure **1**, only one of the products (**14**) is a



14 15

legal structure under the constraints; that structure must be in flask D1.

If one product is forced to be in a certain flask, it can be in no other flask (Appendix III). Thus, the number of dehydration products which could be in D2 and D3 decreases from 210 to 187 (compare Figures 4 and 5). Obviously, with a more complex reaction tree, such logical decisions become complicated. Using the algorithm in Appendix III, REACT determines allowable allocations automatically.

a
```
STRUCS=14
|
*DEHYDRATION->DEHYD=42-s-|D3=14
                         |
                         |D2=14
                         |
                         |D1=14
```
b
```
STRUCS=12
|
*DEHYDRATION->DEHYD=36-s-|D3=12
                         |
                         |D2=12
                         |
                         |D1=12
```

Figure 7. (a) The reaction tree after constraining flask D3 to include those products possessing no vinyl methyl and no vinyl protons, (b) the reaction tree after testing the structures in STRUCS for exactly two secondary methyl groups.

Flask D2 contains a product which possesses no vinyl protons and one vinyl methyl group (Figure 1). Constraining the contents of D2 with this structural information results in the allocation summarized in Figure 6. Now the number of candidate structures in STRUCS is reduced to 45, implying that there are 72 −45 = 27 structures which cannot yield a product distribution which satisfies the structural constraints placed on *both* flasks D1 and D2. An example is **15**, which, although it yields at least one (two) product(s) satisfying the constraints on flask D1, yields no products satisfying the constraints on flask D2. It is therefore discarded as a candidate structure. At the same time, any products of discarded structures (and precursors in a more complex tree) are removed from DEHYD and flasks D1–D3.

Application of the constraints on flask D3 (Figure 1) that the product contained therein possesses neither a vinyl methyl nor a vinyl proton results in the reaction tree shown in Figure 7a. Now only 14 structural candidates remain, and from the allocation of products to flasks (Figure 7a) each yields three unique products. Following the scenario in ref 10, each of the structural candidates was tested for the presence of exactly two secondary methyl groups; the reaction tree of Figure 7b results.

Previously,[10] translation of the results of the dehydration into a substructure used to test the 88 candidates reduced the number of candidates to 22, rather than 14 (Figure 7a). The substructure[10] was correct, but incomplete in that eight structures which obeyed the substructural constraint could not yield the observed products. Through use of REACT, structural information can be applied directly to the structures of potential products without the necessity of translating observations back to the precursors.

Utilities. Appendix I summarizes important utilities available in REACT. We discuss them briefly here not because they are critical to understanding the method but because they are an essential part of the interactive utility of REACT.

(1) Displaying Reaction Tree. Examples of reaction trees in Figures 3–7 illustrate the format in which the reaction sequence can be observed. The DISPLAY1 command allows the chemist to view selected portions of the tree, i.e., one named flask together with any separations or reactions performed on that flask.

(2) Drawing Structures. The structures (or any subset) in any selected flask can be drawn.[13] To check numbering of atoms, particularly in the use of MREACT, structures can also be drawn with structure numbers (NDRAW).

(3) Determining Structural Relationships. Relationships between precursors and products can be obtained using the PARENTS and PRODUCTS commands. A report can be obtained for all or selected structures in a flask, either to summarize precursors which led to a structure (PARENTS reports flask and structure number of every parent of every structure) or products of all or selected structures (PRODUCTS reports flask and structure number of every product of every structure). The FLASKS and COMPARE commands have the function shown in Appendix I. These commands were used to examine the reaction tree in the example to determine relationships among structures presented in the text.

(4) File Manipulation and Other Commands. These utility commands (Appendix I) allow a chemist to save and restore problems or portions thereof at will, thereby maintaining a computer-based "lab notebook" of his operations. Other commands in Appendix I simplify the reporting of problems and subsequent improvements of REACT and correction of errors. CHECKPOINT and UNDO are useful when the chemist wants to explore the consequences of a separation or pruning and still return to his previous reaction tree if desired.

CONCLUSION

In this paper and in previous publications[3,4,6] we have illustrated that the REACT program can be used to assist in the study of chemical reactions in mechanistic and structural chemistry. REACT's basic operations and automatic ramification on SEPARATE and PRUNE represent parallels to some aspects of manual application of chemical reactions and interpretation of results. However, the combination of CONGEN and REACT by no means represents an accurate model of manual structure elucidation. The programs do not "bootstrap" themselves from an empirical formula plus structural inferences from both spectroscopic and chemical data to suggestion of candidate structures. Rather the programs must be used more methodically, which at least has the advantage that the exploration of plausible structures will be exhaustive.

The program has other limitations. The most important is the absence of stereochemistry in the representation of both structures and reactions. Another limitation concerns the algorithm for representing and manipulating the reaction tree. The algorithm (Appendix III) currently assumes the contents of separated flasks are pure compounds. Inseparable mixtures can be handled by REACT by artificially placing the components in separate flasks, but the lack of parallel with laboratory procedures is inelegant.

However, we achieved our goal of providing the chemist with a working tool in the form of an interactive program. This places additional demands on REACT in that the program must be robust, tolerant, and helpful in order that persons unfamiliar with the program are able to use it effectively. The advantage of an interactive program is that the chemist can control each procedural step in his computer "laboratory". To facilitate this control we provided simple mechanisms for definition of reactions and substructures, and for saving and restoring files of structures and previously defined reactions. We provided the capability for thorough investigation, at the computer terminal, of all structures and their interrelationships so that in a short session a chemist can define and carry out reactions and study the consequences of the reactions in depth.

EXPERIMENTAL

The REACT program is written in the programming language INTERLISP. It runs on a Digital Equipment Corp. PDP-10 at the SUMEX computer facility, Stanford University. The program is available (to the limit of available resources) on-line to interested persons over two nationwide computer networks. For information on access to the program please contact the authors.

Appendix I. Commands Available in REACT and Their Functions

category	commands	function
substructure, constraint and reaction definition	EDITSTRUC	for definition of substructures used as pruning constraints
	CONSTRAINT	for creating an internal constraints list
	PRECURSOR	for definition of one or more starting materials
	EDITREACT	for definition of reactions, including site, transform, and constraints
	ATOM	for defining new atoms (the program recognizes C, H, N, and X as atoms; X stands for any atom type)
	REACTIONS?	for reporting names of reactions currently in memory
	CONSTRAINTS?	for reporting what is available on the internal constraint list
reaction application	REACT	for carrying out one or more reactions on one or more structures in a given flask, through one or more steps
	MREACT	same as REACT, but preserves atom identities throughout reaction sequence
product testing	SEPARATE	for separating products of a reaction into named flasks; also determines ramifications of specified number of products
	PRUNE	for testing contents of a flask with structural constraints; also determines ramifications of test results
	MOLECULAR-WEIGHT	for testing contents of a flask when the molecular weight or the empirical formula of the structure known. Also determines ramifications of test results
studying reaction sequence[a]	DISPLAY	for showing the reaction sequence including all reactions and flasks
	DISPLAY1	for showing a selected portion of the reaction sequence
	DRAW	for drawing all or selected structures in a given flask
	NDRAW	for drawing structures with atom numbers
	PARENTS	for determining the parents (precursors) of all or selected products in a given flask
	PRODUCTS	for determining the products of all or selected precursors in a given flask
	FLASKS	for determining which flasks contain which products after separation or pruning
	COMPARE	for comparing the contents of two flasks for identical structures
	MOLECULAR-WEIGHT?	for calculating and displaying the molecular weight and the empirical formula of each of the structures in a flask
file manipulation	SAVE	for saving current reaction sequence, and all structures, reactions, and substructures
	RESTORE	for restoring a previously saved file
	GET	for selective restoring any or all portions of a previously saved file
utility	CHECKPOINT	for saving a copy of REACT in its current state prior to the next operation
	UNDO	for cancelling the last operation and restoring REACT to a previous checkpoint
	CLEAR	for reinitializing REACT
	GRIPE	for registering complaints
	BUGOUT	for reporting errors

[a] A more appropriate (but jargon) name is reaction "tree", that term being descriptive of the way in which the sequence tends to grow outward and downward from the precursor.

Appendix II. Structure Manipulation Commands Used to Define Structural Units for EDITSTRUC, EDITREACT and PRECURSOR

command	function
RING *n*	constructs a ring of *n* carbon atoms
CHAIN *n*	constructs a chain of *n* carbon atoms
JOIN *i j*	increases the bond order between atoms *i* and *j* by one
UNJOIN *i j*	decreases the bond order between atoms *i* and *i* by one
BORD *i j n*	sets the bond order between atoms *i* and *j* to *n* (*n* = ANY indicates a bond of unspecified multiplicity)
BRANCH *i n*	sprouts a branch of *n* carbon atoms from
LINK *i j n*	links atoms *i* and *j* with a chain of *n* carbon atoms
LNODE *i* min max	declares atom *i* to be a chain of atoms of length min to max
ATNAME *i* name	associates name with atom *i*
DELATS *i*	removes atom *i* from the substructure
RENUMBER *i* new	renumbers atom *i* to be atom number new
HRANGE *i* min max	associates a range of hydrogen atoms with atom *i*
TAG *i*	associates a "tag" (an asterisk) with atom *i*; used now only to represent the hydrogen atom for PROTON constraints
UNTAG *i*	removes the tag from atom *i*
ARTYPE *i* artype	declares the aromaticity of atom *i* to be that specified by artype; the artype may be either AROMATIC (AROM, A), NON-AROMATIC (N, NON-AR), or EITHER (E)
ADRAW	draws the current substructure with atom names
NDRAW	draws the current substructure with atom numbers
SHOW	prints the connection table of the substructure
CLEAR	erases the current substructure
RENAME name	associates a new name with the current substructure
RELS	defines relative quantities of hydrogens or π electrons attached to atoms, or the relative lengths of atoms which were declared to be LNODES
GRIPE	registers a complaint or comment about the program
BUGOUT	saves a copy of the program when an error is encountered
DONE	exits with modified substructure
HALT	exits with no substructure modifications

Appendix III. Internal Representation, Ramification and Allocation

As noted in the text, the reaction tree is a data structure internal to REACT which contains the detailed interrelationships between precursors and products at all levels of a problem. Though it is too complex to display in toto, the user of REACT may access as much of the stored information as is needed via the PARENTS and PRODUCTS commands (see Appendix I).

The compactness of the data structure allows storage of large amounts of information. For instance, in the generation of marine sterol side chains,[1] a reaction tree with about 34 flasks with an average of 20 structures per flask, with an average of 10 atoms per structure, was easily constructed.

The reaction tree is composed of two types of nodes which we term RTRXNs and RTSTRUCs (the RT prefix signifies "reaction tree") (Figure 8). These nodes contain "pointers", or links, to other nodes indicating various relationships. Conceptually, an RTRXN corresponds to the application of a reaction to a single precursor structure while an RTSTRUC represents a unique structure in one of the overall lists of structures (e.g., in STRUCS or in DEHYD in the text).

The information contained in an RTRXN (Figure 8) is summarized as follows:

FROMSTRUC	a pointer to the RTSTRUC which is the precursor of this reaction
FROMFLASK	the name of the flask (separated or "raw") to which the reaction was applied
RXNAME	the name of the reaction which was previously defined
PRODS	a list (possibly empty) of pointers to the RTSTRUCs produced by this reaction
RAWFLASK	the name of the flask containing the "raw" (unseparated) products
SEPFLASKS	(if the "raw" products have been separated) a list of flask-name sets, one for each product in PRODS; each set indicates the separation flasks which are allowed recipients of the corresponding product; data associated with the separation flasks (the associations are shown as lines or "edges" connecting PRODS and SEPFLASKS in the RTRXN of Figure 8) summarize both the pruning constraints and any implied constraints found by the allocator (see below)

The information contained in an RTSTRUC (Figure 8) is as follows:

FROMRXNS	a list of one or more pointers to the RTRXNs which produce this structure (i.e., which contain, in their PRODS lists, pointers to this RTSTRUC)
STRUC	a topological description of the structure itself
TORXNS	a list (possibly empty) of pointers to the RTRXNs for which this structure is the precursor

These basic units are created, and the tree extended, during each use of the REACT command. The RTSTRUCs are scanned to find all which can reside in the flask to which the reaction is being applied. For each of these, a reaction is applied, a new RTRXN is added to the tree, and for each product which has not previously been encountered a new RTSTRUC is added as well. Thus the REACT command has a purely constructive effect on the tree.

The SEPARATE and PRUNE commands are essentially destructive to the tree. During SEPARATE, all RTRXNs which do not yield the proper number of products are marked for deletion from the tree, while those that do are given initial SEPFLASKS entries indicating that any product may occupy any separation flask. During PRUNE, each RTRXN which gives products in the specified flask is examined, and each product which can reside in (or be allocated to) the flask undergoes testing against the given constraints. If a product fails the test, the effect upon the tree depends upon whether the flask being pruned is a "raw" products flask or a separation flask. In the former case, the RTSTRUC is simply marked for deletion from the tree (see "fail test, "delete" structure", Figure 8). In the latter case, the product is "blocked" from occupying the flask by removing the flask from the appropriate SEPFLASKS entry in each parent reaction (see "fail test, remove pointer", Figure 8). This blocking (removal of edges from PRODS to SEPFLASKS) may invalidate the RTRXN which is being examined because illegal allocations (see below) may result, so each of the examined RTRXNS is marked for checking.

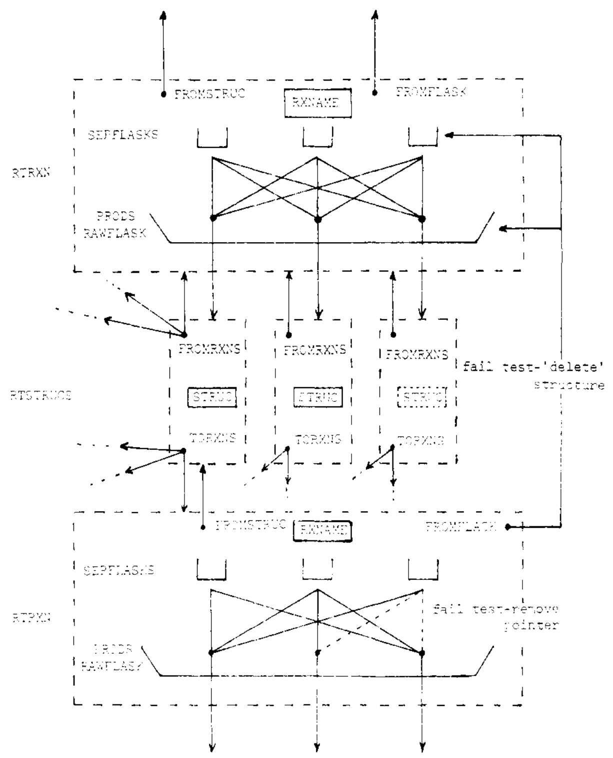


Figure 8. Representation of RTRXN and RTSTRUC nodes in a reaction tree.

The process of removal of RTRXN or RTSTRUC nodes from the tree may have a far-reaching effect on the entire reaction tree because each removal may have implications for higher or lower nodes (see below). The algorithm responsible for chasing down all such implications is called RAMIFY,[14] the principles of which rest upon some simple conditions which must exist in the tree: Each RTSTRUC (except at the top level) and each RTRXN must have a "parent"; the SEPFLASKS entry of each RTRXN must satisfy the conditions of the allocator (i.e., an allocation must exist and all implied restrictions must be reflected in SEPFLASKS; see below); the PRODS of an RTRXN are to be treated as an indivisible set (i.e., if one product is found to be illegal, all the others are, too); and an RTRXN remains in the tree if and only if its FROMSTRUC is capable of residing in the flask to which the RTRXN applies.

As noted above, nodes in the tree are capable of being "marked" either for deletion or for checking. RAMIFY first attempts to locate an RTRXN which has been marked for deletion. If one is found, then two basic operations take place. First, since the reaction may no longer serve as a parent for any product P in the PRODS list, the RTRXN is removed from the FROMRXNS of each such P and the product P is marked for checking. Second, the RTRXN is removed from the TORXNS of its precursor (if the FROMSTRUC pointer has not already been deleted; see below). An implication of this removal is that the precursor of the RTRXN cannot be allowed to occupy the flask to which the RTRXN applied. As with the PRUNE command discussed above, this may have one of two effects. If the flask is a raw products flask, then the precursor is marked for deletion. If the flask is a separation flask, then the precursor is "blocked" from occupying the flask and all RTRXNs leading to the precursor are marked for checking.

This search-deletion loop continues until there are no remaining RTRXNs to be deleted. RAMIFY then searches for an RTRSRUC which is marked for deletion. If one is found, the RTSTRUC is removed from the PRODS list of each FROMRXN and the FROMSTRUC entry of each TORXN is deleted. Then, each of the FROMRXNS and TORXNS of the RTSTRUC is marked for deletion. When an RTSTRUC has been processed in this way, RAMIFY returns to searching for RTRXNs which are marked for deletion.

When no nodes remain to be deleted, RAMIFY searches for an RTRXN which is marked for checking. If one is found, the allocator is called using the current SEPFLASKS entries. If an allocation is no longer possible, the RTRXN is marked for deletion. Otherwise, any product in the PRODS list is marked for checking if the corresponding SEPFLASKS entry was changed (i.e., if an implied restriction was found for the product) by the allocator. After an RTRXN has been checked in this way, RAMIFY once again begins the search for RTRXNs which are marked for deletion.

When no nodes are marked for deletion and no RTRXNs are marked for checking, RAMIFY searches for an RTSTRUC which is marked for checking. If the RTSTRUC has no FROMRXNS, it is marked for deletion. If any of the TORXNS are applied to separation flasks which the RTSTRUC can no longer occupy, those TORXNS are marked for deletion. When an RTSTRUC has been checked in this way, RAMIFY once again begins searching for RTRXNs which are marked for deletion.

Eventually, no nodes remain to be deleted or to be checked, and the tree is fully consistent (i.e., all ramifications of the original changes have been found). After reclaiming space freed by the deletions and updating various data structures in REACT which summarize aspects of the tree, RAMIFY terminates.

The Allocator. The task of the allocator is twofold. Given a list of products from the application of a particular reaction to a single precursor, and given an equal number of labeled separation flasks (assigned by the SEPARATE command) each of which can hold exactly one product, the first problem is to determine whether there is at least one allowable allocation of products to flasks. Initially, before any restrictions are placed on the separation flasks, this is a trivial problem; any product can go to any flask and any allocation is allowable. But as the user of REACT associates constraints with the flasks via the PRUNE command, or after an additional reaction-separation sequence, the set of flasks which can receive a given product (called the allowed recipients of the product) becomes smaller. At some point these restrictions may become so severe that there is no one-to-one assignment of products to flasks such that each product occupies one of its allowed recipients. The primary responsibility of the allocator is to detect such situations. Secondly, the allocator needs to perceive additional restrictions which are implied by a particular allocation problem.

REACT maintains, in the SEPFLASKS entries of the RTRXNs, a separate record of the separation and allocation

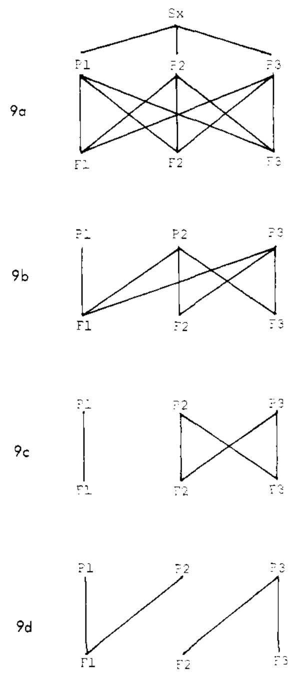


Figure 9. (a) A trivial allocation problem. (b) The allocation problem before and (c) after detection of implied restrictions. (d) Impossible allocation.

for the products of each precursor. As mentioned above, the allocator is applied whenever the chemist has placed restrictions on the distribution of products into flasks using PRUNE or SEPARATE, or whenever a RTRXN or a RTSTRUC has been eliminated from the reaction tree. Consider as an example the trivial allocation shown as a graph in Figure 9a subsequent to a reaction of Sx to yield three products. P1–P3 were allocated to three flasks F1–F3. The lines (or "edges") connecting products to flasks in Figure 9a represent possible association of particular products with flasks. Assume that a constraint allows product P1 to be only in flask F1. The PRUNE command removes the edges between P1 and F2, F3, resulting in a new graph, Figure 9b. There is still at least one way of allocating P1–P3 to F1–F3, but there are implied restrictions which are detected by the algorithm. Consider the product P1, which can be only in F1. Since F1 can only contain one product, P2 and P3 cannot be allocated to F1; therefore edges P2–F1 and P2–F1 can be removed from the graph to yield the new allocation, Figure 9c. It should be emphasized that this implied restriction is a property of the whole allocation problem rather than a conclusion about the structures P2 and P3. The actual bookkeeping problem is made more complex by the fact that there may be some other route to P2 or P3 starting from some other precursor (since REACT eliminates redundant structures, only one copy of P2 and P3 is retained) and the allocation of the products of that reaction may allow P2 or P3 to occupy the flask labeled F1.

The allocator accomplishes these tests using principles related to the graph-theoretical problem of finding complete matchings in bipartite graphs.[15] The algorithm considers all nonempty subsets of the set of products for a given RTRXN. For each such subset S it compares the size of S (say, N) with the number of flasks (say, M) which can contain at least one of the products in S. It is a well-known, graph-theoretical result[15] that an allocation is illegal if and only if some subset can be found with $N > M$. By this criterion, the graph in Figure 9b has a legal allocation because the possible subsets, {P1}, {P2}, {P3}, {P1,P2}, {P1,P3}, {P2,P3} and {P1,P2,P3} have N = 1, 1, 1, 2, 2, 2, 3, respectively, and M = 1, 3, 3, 3, 3, 3, 3, respectively, and in no case is $N > M$. Consider as an example the allocation shown in Figure 9d. For this graph, however, the subset {P1,P2} has $N = 2$ and $M = 1$, so no legal allocation exists.

The implied restrictions are detected using our own addition to the well-known algorithm, an addition which can be proven accurate. The rule is that whenever a subset S is encountered with $N = M$, all edges can be deleted which connect products *outside* of S to one of the M recipients of the products *inside* of S. Thus, in Figure 9b, if S = {P1} we have $N = M = 1$, and thus the edges connecting P2 and P3 (the products outside of S) to F1 (the recipient of P1) can be deleted. All other subsets for the graph in Figure 9b have $N < M$, so there are no other implied restrictions. The graph in Figure 9c results.

For N products, there are $2^N - 1$ nonempty subsets and thus the procedure described above requires an amount of time roughly proportional to 2^N in the worst case (i.e., when a legal allocation exists so that all subsets must be explored).

LITERATURE CITED

(1) Part XXVII of the series "Applications of Artificial Intelligence for Chemical Inference." For part XXVI, see T. H. Varkony, D. H. Smith, and C. Djerassi *Tetrahedron*, **34**, 841 (1978).
(2) We wish to thank the National Aeronautics and Space Administration (NGS-05-020-004) and the National Institutes of Health (RR-00612, GM20832, and GM06840) for their generous financial support for our research and for their support (RR-00785 SUMEX) of the SUMEX computer resource on which the REACT program is available via nationwide computer networks.
(3) T. H. Varkony, R. E. Carhart, and D. H. Smith in "Computer-Assisted Organic Synthesis Planning", W. T. Wipke and W. J. Howe, Ed., American Chemical Society, Washington, D.C., 1977, p 188.
(4) R. E. Carhart, T. H. Varkony, and D. H. Smith in "Computer-Assisted Structure Elucidation", D. H. Smith, Ed., American Chemical Society, Washington, D.C., 1977, p 126.
(5) R. E. Carhart, D. H. Smith, H. Brown, and C. Djerassi, *J. Am. Chem. Soc.*, **97**, 5755 (1975).
(6) T. H. Varkony, D. H. Smith, and C. Djerassi, *Tetrahedron*, **34**, 841 (1978).
(7) E. J. Corey and W. T. Wipke, *Science*, **166**, 178 (1969).
(8) W. T. Wipke, H. Brown, G. Smith, S. Choplin, and W. Sieder, "Computer-Assisted Organic Synthesis Planning", W. T. Wipke and W. J. Howe, Ed., American Chemical Society, Washington, D.C., 1977, p 97.
(9) R. J. Feldmann, "Computer Representation and Manipulation of Chemical Information", W. T. Wipke, S. R. Heller, R. J. Feldmann, and E. Hyde, Ed., Wiley-Interscience, New York, N.Y., 1974, p 55.
(10) C. Cheer, D. H. Smith, C. Djerassi, B. Tursch, J. C. Braekman, and D. Daloze, *Tetrahedron*, **32**, 1807 (1976).
(11) R. E. Carhart and D. H. Smith, *Computer Chem.*, **1**, 79 (1976).
(12) (a) H. L. Morgan, *J. Chem. Doc.*, **5**, 107 (1965); (b) W. T. Wipke and T. M. Dyott, *J. Am. Chem. Soc.*, **96**, 4834 (1974).
(13) R. E. Carhart, *J. Chem. Inf. Comput. Sci.*, **16**, 82 (1976).
(14) One way of visualizing the operation of RAMIFY is to actually work through an example of a reaction tree of moderate complexity. Contact the authors for further information.
(15) C. L. Liu, "Introduction to Combinatorial Mathematics", McGraw-Hill, New York, N.Y., 1968, p 280.

NEWS AND NOTES

Drexel Receives Two-Year NSF Grant

The Graduate School of Library Science, Drexel University, has been awarded a $229,000 grant from the National Science Foundation (NSF) to continue its Individualized Instruction in Data Access project. The project, which began in 1975 with a grant of $72,000, has been renewed for two more years to bring it to a close.

Professor Charles T. Meadow of the Library School faculty will be Principal Investigator. Working with him will be Dr. Thomas T. Hewett, of Drexel's Psychology and Sociology Department, and staff members of the Franklin Institute Research Laboratories.

The objective of the project is to develop a computer system which will assist scientist-users to search computer-based bibliographic files. In the Drexel system, the computer would work cooperatively and actively with the searchers. It would have the ability to detect and help overcome errors, as well as suggest new search approaches. A small intermediary computer, acting as a monitor, will be connected both to the user's data communications terminal and to the large, remotely located search computer. In this way, scientists and engineers, working in their own laboratories or offices, can make searches of the world's published literature.

CAS Registry System Contract

The Warner-Lambert/Parke-Davis Pharmaceutical Research Division has established a private satellite of the Chemical Abstracts Service Chemical Registry System. The division has contracted with CAS to maintain a special computer-based registry of chemical substances in which it is interested and feed information from this registry into the division's information retrieval system at Ann Arbor, Michigan.

Structures and names of chemical substances provided by the Warner-Lambert/Parke-Davis unit are recorded in private registry files at CAS using procedures and computer programs similar to those used to create the CAS registry files. Copies of the private name and structure files are delivered to Ann Arbor along with special files and programs that enable the division to display and print structure diagrams from the computer-readable structure records. CAS also determines which of the substances it registers for the division are among the more than 4.1 million substances in the CAS registry and supplies the CAS registry numbers, systematic names, and other names on file for them.

The special registry is the first to be established at CAS for an industrial contractor. CAS has operated a registry-based chemical information system under contract for the National Cancer Institute since 1965.

CA SEARCH and REG/CAN

Chemical Abstracts Service is offering two new computer-readable information files. One, *CA Search*, brings together in a single computer file the full identifying information and index entries for papers, patents, and other documents abstracted and indexed in *Chemical Abstracts*. The other, REG/CAN, links the unique Registry Numbers that identify chemical substances in CAS's computer-based Chemical Registry System with the CA abstract numbers for documents that contain information about the substances.

CA Search combines the content of CAS's widely used *CA Condensates* computer-readable file, which contains complete bibliographic citations and keyword index terms for all documents abstracted in CA, with that of the newer *CA Subject Index Alert* file, which contains the more comprehensive and precise CA general subject and chemical substance index entries for the same documents. The merger of these