

## Automatic Problem Solving Applied to Synthetic Chemistry

Malcolm BERSOHN

*Department of Chemistry, University of Toronto, Toronto 181, Ontario, Canada*

(Received September 24, 1970)

The first computer program that plans multi-step organic syntheses has been written. In the course of its operation the program generates partially complete synthetic pathways and ranks them as to efficiency. At each stage the program develops that partially complete route which it judges to be the most efficient, and extends it by one step. Partially complete routes which lose too much material are discarded. After each tolerably efficient pathway from available starting materials to the goal molecule is completed, the entire pathway is printed, together with the yield of each step and the overall yield.

### The Information Structure of a Synthesis Problem

The various alternative methods of synthesizing a substance can be represented by a diagram like that of Fig. 1. Note that the figure is by no means a complete statement of all of the possible ways of making benzene from cheaper or simpler substances. For clarity, small molecules like  $\text{CO}_2$ ,  $\text{O}_2$ ,  $\text{H}_2$ ,  $\text{H}_2\text{O}$  etc. are omitted from the diagram.

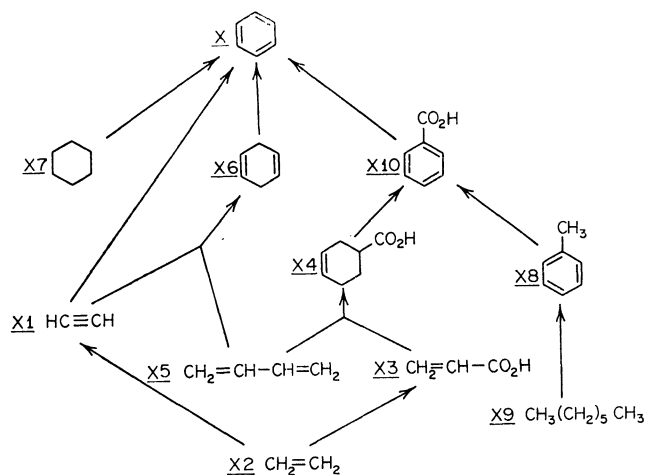


Fig. 1.

Such a two dimensional diagram can become very difficult to decipher because the more pathways that are included the more intersecting lines are possible. In addition, we will have more and more cases like X2, which is both four steps and two steps removed from the goal molecule. Hence there is no well defined level structure or generation structure. Also, evidently, we cannot describe the structure as a tree since there are cycles in it, such as X, X1, X2, X3, X4, and X10. A tree in a forest has no such rings and the tree of graph theory is also acyclic. Nonetheless, despite the unsuitability of two dimensional diagrams for representing it and the absence of a familiar name, the structure does have definite properties, as follows:

- (1) Every molecule except the goal molecule has at least one successor molecule.
- (2) Every molecule except the starting molecules has at least one parent molecule.
- (3) Some molecules have coreactants with which they react to form their common successor mole-

cule. For example, acetylene is the coreactant of butadiene and butadiene is the coreactant of acetylene in the problem of Fig. 1.

These three kinds of relationships link all the molecules of the problem together and these links must be suitably represented in the memory of a computer.

### The Strategy Used by Chemists for Finding Good Synthetic Routes

Suppose that we wish to synthesize a molecule named M, starting from readily available molecules. The usual way of attacking this problem is to consider all possible synthetic reactions which could directly produce M. Suppose, for example that L1 and L2 react to form M, L3 can be rearranged to give M, L4 and L5 also react to produce M, all with satisfactory yield. Then we must next consider ways to make the set of molecules {L1, L2, L3, L4, L5}. For example, L1 can be made from K1 and K2; it can also be prepared by isomerization of K3 or K4 etc. We keep on moving backwards in this fashion until available substances are reached, *e.g.* the set {C1, C2, C3, B1, B3, A1, ..., A12}. Some of the pathways found from available substances to M have higher yield and consequently are cheaper to execute than others; we select the most efficient synthetic route(s) for experimental trial.

There are two major difficulties in the carrying out of the above method for planning a synthesis. The first difficulty is that our memory may fail us when considering how to make an intermediate substance, *e.g.* K3; we may not recall a particularly good reaction that could produce K3 from, let us say, J27.

The second difficulty is much greater. Even when and if we remember all relevant synthetic reactions, the complete set of possible pathways is too enormous for human consideration. Suppose, for example, that we have a molecule as complicated as the anti-biotic aureomycin, and assume there exists at least "some" possible synthetic routes which can produce aureomycin in 20 steps. If we assume that at each step there are five possible ways to produce a given intermediate in one step, then there are not only "some" possible routes but actually there are something like  $5^{20}$  possible routes capable of being carried out experimentally. Some of these routes are grossly inefficient; neglecting then is no loss. But some that we do not consider may be much more efficient than the route that we choose to test experimentally.

Let us consider the first sublist: (C1 carbon 3 ((C2 1))). The first symbol, C1, is the name of an atom. It is convenient, but certainly not necessary to refer to carbon atoms with a name that begins with C. The second symbol, carbon, is the name of the element of which the atom C1 is an example. The third symbol, 3, is the number of hydrogen atoms bonded to C1. The last item in the sublist is a list of pairs. In this case there is only one such pair, (C2 1). This means that C2 is singly bonded to C1. The pair (C2 3) would mean that C2 is triply bonded to C1. An aro-

matic bond between C1 and C2 would be shown as (C2 0).

The reader will note various redundancies in this representation. It is obvious that since C1 is bonded to only one non-hydrogenic atom, and by a single bond, then C1 must have three attached hydrogen atoms. Hence, if we neglect stable radicals, the number of hydrogen atoms need not be entered. The program could compute this number when needed. I have temporarily taken the view that time is more important than space so the present program stores this redundant number for quick retrieval. Another redundancy may be noted: the sublist for C1 contains the pair (C1 1) which tells us that C2 is singly bonded to C1. The sublist for C2 contains the pair (C1 1), which conveys the same information. In fact all such bonding information is stored in two places. For another example, (O4 2) in the C3 sublist conveys the same information as (C3 2) in the O4 list. Here again, speed of retrieval has been the dominant consideration. Similarly, in a machine or assembly language program it would be simple to recognize the C in C2 and infer that it is a carbon atom. In a higher level language such as LISP, breaking off the alphabetic part of a name from the numerical part consumes computer time.

Let us call this kind of list of sublists a molecular structure list. Since each sublist refers to a particular non-hydrogenic atom, we can call the sublists atomic property lists. We mention in passing that *cis-trans*, chirality, and isotopic labelling information can be added to an atomic property list when necessary.

The reader will find it instructive to write down the Kekule structure, with appropriate labels, corresponding to the following molecular structure list:

```
((C 1 carbon 0 ((C 2 0)(C 6 0)(C 7 1)))
 (C 2 carbon 1 ((C 1 0)(C 3 0)   ))
 (C 3 carbon 1 ((C 2 0)(C 4 0)   ))
 (C 4 carbon 1 ((C 3 0)(C 5 0)   ))
 (C 5 carbon 1 ((C 4 0)(C 6 0)   ))
 (C 6 carbon 1 ((C 5 0)(C 1 0)   ))
 (C 7 carbon 0 ((C 1 1)(O 8 2)(O 9 1)))
 (O 8 oxygen 0 ((C 7 2)         ))
 (O 9 oxygen 1 ((C 7 1)         )))
```

### Representation of Chemical Reactions in a Computer

From the above discussion it is apparent that a molecular structure list, which is conveniently one dimensional, can be put into a computer to represent molecular structure unambiguously. Chemical reactions can then be represented in a computer as transformations of these molecular structure lists. The four basic transformations of the program are called ADDH (add a hydrogen atom), REMOVEH (remove a hydrogen atom), MAKEBOND, and BREAKBOND. In addition, changes of chirality or of *cis-trans* arrangement of allylic atoms or of ring junctions are made on the atomic property lists of the parent molecules whenever the reaction concerned requires this.

We define the programming term *subroutine*. A subroutine is a set of instructions inside a program which

form a unit and carry out a particular function. The subroutine is given one or more symbols, numbers or structures as input. It produces one or more symbols, numbers or structures as results. These results are then used by the section of the program which called upon the subroutine for answer(s). Typically a subroutine is called upon to produce an answer at several places in a program; hence, it is executed several or many times in the course of running the program, each time with different input. An example from this program is a subroutine called RINGCOUNT which, given the structure list of a molecule as input, returns an integer as answer, telling how many rings the molecule has. Another example is a subroutine CARBON which given the name of an atom as input, states as output (after searching the atomic property list for the second item, the element name) whether it is true or false that the given atom is a carbon atom.

Subroutines that represent chemical reactions always have as one of their inputs the name of a molecule. These subroutines usually begin by making a copy of the molecular structure list of the input molecule. The transformations are then performed on this copy. In this way the structure of the input molecule is always preserved in memory.

The subroutine HYDROGENATE is supplied with the names of two non-hydrogenic atoms as well as the name of the molecule to which the two atoms belong. For each of the two atoms the subroutine calls upon ADDH to increase the number of bonded hydrogens by one. It also changes the number 2 in the atomic property lists into the number 1, unless the two atoms are triply bonded to each other in which case it changes the number 3 to 2 in both of the atomic property lists. (Complete hydrogenation of a triple bond requires the use of this subroutine twice). If the two atoms are aromatically bonded to each other the subroutine HYDROGENATE makes them singly bonded to each other and changes the bonding of the other atoms in the same ring, which are not part of another aromatic ring, from aromatic to double bonding. If the two atoms are not bonded to each other the subroutine sends an error message.

Suppose the program's internal name for 1-propanal is X51. Then the command "HYDROGENATE C3 O4 X51" results in the following new molecular structure list:

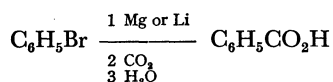
```
(C 1 carbon 3 ((C 2 1)   ))
 (C 2 carbon 2 ((C 3 1)(C 1 1)))
 (C 3 carbon 2 ((C 2 1)(O 4 1)))
 (O 4 oxygen 1 ((C 3 1)   ))
```

The program gives this new structure a unique but arbitrary name. DEHYDROGENATE is also given the names of two non-hydrogenic atoms and the name of the molecule to which they belong as input. Thus, if X51 is 1-propanal as before, then the command "DEHYDROGENATE C1 C2 X51" results in the acrolein molecular structure list:

```
(C 1 carbon 2 ((C 2 2)   ))
 (C 2 carbon 1 ((C 3 1)(C 1 2)))
 (C 3 carbon 1 ((C 2 1)(O 4 2)))
 (O 4 oxygen 0 ((C 3 2)   ))
```

This new substance is similarly given some temporary internal name by the program, not any chemical name.

To illustrate the operations MAKEBOND and BREAKBOND, let us consider the case of benzoic acid, whose molecular structure list was given above. We examine the reaction



The program allows this type of reaction only when the product is a monocarboxylic acid. Let us call benzoic acid B1. Let us further introduce a variable Q, which will exist only inside of this subroutine. Initially we will set Q equal to B1, that is Q will initially have the same structure as benzoic acid. The structure list of Q will be successively modified by four commands of the subroutine for this type of carboxylation. The end result will be to re-organize the final structure list of Q into the structure lists of the parents of B1.

The first of the four commands is "BREAKBOND C1 C7 Q." The result of this first operation is to produce the following modified structure list for Q:

```
(C1 carbon 0 ((C2 0)(C6 0)))
(C2 carbon 1 ((C1 0)(C3 0)))
(C3 carbon 1 ((C2 0)(C4 0)))
(C4 carbon 1 ((C3 0)(C5 0)))
(C5 carbon 1 ((C4 0)(C6 0)))
(C6 carbon 1 ((C5 0)(C1 0)))
(C7 carbon 0 ((O8 2)(O9 1)))
(O8 oxygen 0 ((C7 2) ))
(O9 oxygen 1 ((C7 1) ))
```

In the atomic property list of C7 the subroutine has removed any reference to C1; in the atomic property list of C1 the reference to C7 has also been removed. In the structure list of Q the bond has now been broken. At the next command in the subroutine "MAKEBOND C1 BR10 Q" the structure list of Q is modified again. The name BR10 is chosen by the program for convenience. After execution of this MAKEBOND command the structure list of Q is:

```
(C1 carbon 0 ((BR10 1)(C2 0)(C6 0)))
(C2 carbon 1 ((C1 0)(C3 0)))
(C3 carbon 1 ((C2 0)(C4 0)))
(C4 carbon 1 ((C3 0)(C5 0)))
(C5 carbon 1 ((C4 0)(C6 0)))
(C6 carbon 1 ((C5 0)(C1 0)))
(C7 carbon 0 ((O8 2)(O9 1)))
(O8 oxygen 0 ((C7 2) ))
(O9 oxygen 1 ((C7 1) ))
(BR10 bromine 0 ((C7 1) ))
```

We remark in passing that the input to the BREAKBOND subroutine is the names of two atoms followed by the name of the molecule to which they belong; the arguments of the MAKEBOND subroutine, however, are the names of two atoms followed by the name of the molecule to which the *first* mentioned atom belongs. Thus, in the command just discussed, MAKEBOND C1 BR10 Q is correct but MAKEBOND BR10 C1 Q is syntactically incorrect.

For the third command we introduce the variable K which refers to the hydroxyl oxygen of the carboxyl

group in question, namely O9, in this case. The execution of the command MAKEBOND C7 K Q results in the following structure list for Q:

```
(C1 carbon 0 ((BR10 1)(C2 0)(C6 0)))
(C2 carbon 1 ((C1 0)(C3 0)))
(C3 carbon 1 ((C2 0)(C4 0)))
(C4 carbon 1 ((C3 0)(C5 0)))
(C5 carbon 1 ((C4 0)(C6 0)))
(C6 carbon 1 ((C5 0)(C1 0)))
(C7 carbon 0 ((O8 2)(O9 2)))
(O8 oxygen 0 ((C7 2) ))
(O9 oxygen 1 ((C7 2) ))
(BR10 bromine 0 ((C1 1) ))
```

If the inputs to MAKEBOND are atoms which are singly bonded to each other, MAKEBOND adds one more bond, making them doubly bonded.

The last command is "REMOVEH K." This command results in the final structure list for Q:

```
(C1 carbon 0 ((BR10 1)(C2 0)(C6 0)))
(C2 carbon 1 ((C1 0)(C3 0)))
(C3 carbon 1 ((C2 0)(C4 0)))
(C4 carbon 1 ((C3 0)(C5 0)))
(C5 carbon 1 ((C4 0)(C6 0)))
(C6 carbon 1 ((C5 0)(C1 0)))
(C7 carbon 0 ((O8 2)(O9 2)))
(O8 carbon 0 ((C7 2) ))
(O9 carbon 0 ((C7 2) ))
(BR10 bromine 0 ((C1 1) ))
```

The generation is now complete. After each generation is complete, a subroutine called REORGANIZE is called in to form, out of the changed copy of the original structure list (*i.e.* out of the final structure list for Q), one or more parent structure lists consisting of connected atoms. (In some reactions, like the esterification of a dicarboxylic acid, there are more than two "parent" molecules.) Here the subroutine REORGANIZE finds one group of three connected atoms and another group of seven connected atoms. REORGANIZE then forms two new structure lists, as follows:

```
I (C7 carbon 0 ((O8 2)(O9 2)))
   (O8 oxygen 0 ((C7 2) ))
   (O9 oxygen 0 ((C7 2) ))
```

The program does not know that this molecule is called carbon dioxide; it gives it an arbitrary name such as B2. The program does know, as we shall see, that a molecule this small can be considered to be available.

```
II (C1 carbon 0 ((BR10 1)(C2 0)(C6 0)))
   (C2 carbon 1 ((C1 0)(C3 0)))
   (C3 carbon 1 ((C2 0)(C4 0)))
   (C4 carbon 1 ((C3 0)(C5 0)))
   (C5 carbon 1 ((C4 0)(C6 0)))
   (C6 carbon 1 ((C5 0)(C1 0)))
   (BR10 bromine 1 ((C1 1) ))
```

This parent, bromobenzene, is given an arbitrary name, such as B3. The names Q and K and L and their associated information are discarded as we exit from the reaction subroutine. REORGANIZE also attaches to the name B3 the names B1 as successor and B2 as coreactant. To the name B1 is attached the name B3 as parent. B2 is formally also a parent but it is not

marked as such by the program, because it is the smaller parent. B2 has attached to it the name B3 as coreactant. To the largest parent, B3, is also attached the name of the reaction, BROMIDECARBOXYLATION, which causes the relationships. Yield information is also inserted at this point. In the present program all the yields supplied by the reaction subroutines are average values from the literature. In future versions of the program subroutines like STERICINDRANCE (which calculates degree of branching near a site) will be called in to calculate the estimated deviation of the yield from the average yield, so that chemical experience can be fully utilized.

### A Generalized Form for Writing the Chemical Transformations of a Reaction Subroutine

In the previous discussion of the reaction BROMIDECARBOXYLATION, the first command given was BREAKBOND C1 C7 Q. Actually this is the content of the command at a comparatively low level. The initial, *i.e.* the highest level version of this command merely says something like BREAKBOND CARBON-NEXT-TO CARBOXYL-CARBON CARBOXYL-CARBON Q. It is left to the detailed operation of the subroutine as it calls in other subroutines to find that C7 is a carboxyl carbon and that C1 is its neighbouring carbon atom. Similarly "MAKEBOND C1 BR10 Q" is originally something equivalent to "MAKEBOND CARBON-NEXT-TO-CARBON-CARBON INVENTED-NAME-FOR-BROMINE-ATOM Q." The command MAKEBOND C7 O9 Q is at the higher level also essentially MAKEBOND CARBOXYL-CARBON HYDROXYL-OXYGEN-NEXT-TO-CARBON-CARBON Q. It is apparent from this that if we just provide a subroutine with the simple  $-C-CO_2H$  structure and give directions for converting it to  $-C-Br$  and  $CO_2$  then the reaction subroutine can be made more compact. The useful procedure GENREACTION takes as its arguments (inputs) a general "structure," such as  $-C-CO_2H$  Br, in the form of a structure list, a list of pairs of atoms in the general structure between which we should break a bond, a list of pairs of atoms of the general structure between which we should make a bond, a list of the atoms of the general structure to which we should add a hydrogen atom and a list of atoms of the general structure from which we should remove a hydrogen atom. Some of these lists may be empty. The output of GENREACTION will be the required predecessor(s).

Thus in the latest version of the program, inside of the BROMIDECARBOXYLATION subroutine is the command

```
GENREACTION (E1 C0 ((E2.1)))
              (E2 C0 ((E1.1)(E3.2)(E4.1)))
              (E3 O0 ((E2.2)))
              (E4 O1 ((E2.1)))
              (E5 Br 0 nil)
              ((E1.E2)) ((E2.E4) (E1.E5)) nil (E4).
```

In the future the program will be modified to "learn"

synthetic reactions at a rapid rate. All this means is that the program will accept input data in the form of: *name of the reaction, model structure list, lists of atom pairs from the model structure list for BREAKBOND and MAKEBOND, lists of atoms of the model structure for ADDH and REMOVEH, yield, structural conditions which decrease the yield, presence of acid, base, oxidizers or reducing agents among reagents of the reaction.* The program will then convert this input data into appropriately named subroutines and add the subroutines to itself permanently. (The new expanded program will be written onto a disk file or punched out onto cards.) The program must also insert the name of the new subroutine into the proper list, *e.g.* carbonyl reactions, so that it will be called when appropriate. It is evident that compact procedures like GENREACTION will be required for this "learning" of chemistry by the program.

### Minimizing of Repetition

Hydrogenation and dehydrogenation subroutines are available in the program. In order to prevent a long series consisting of useless alternate addition and removal of hydrogen the dehydrogenation subroutine is defined so that it cannot be used to cancel the effect of an immediately adjacent hydrogenation. At least one other chemical reaction must intervene between a hydrogenation and a dehydrogenation. Similarly hydrolysis may not immediately follow or precede ketal formation or esterification. Also, wherever possible, the temporary attachment and subsequent removal of a blocking, protective, activating or solubilizing group is treated together with the reaction that requires this as a single reaction.

The program indeed performs much useless hydrogenation and dehydrogenation but pathways along which this occurs automatically get a lower score than otherwise identical pathways along which the useless reactions did not occur.

### The Operation of the Program

The general operation of the program can be summarized in the following three steps.

1. Put goal molecule on NEWLIST. Examine goal molecule.
2. GENERATEPREDECESSORS of the first molecule on NEWLIST.
3. If NEWLIST is empty then STOP. Otherwise go to Step 2.

*Newlist and Oldlist.* A list named NEWLIST is maintained which consists of the names of all of the molecules whose predecessors we do not know. There is a corresponding list called OLDLIST which consists of the names of all of the molecules which are not available but for which we have found all possible immediate predecessors.

*Beginning of the Program.* The program begins by establishing NEWLIST, with a single item on it, *i.e.* the name of the goal molecule M. It then calls in the subroutine EXAMINE to survey the goal molecule,

noting all structural features and combinations of structural feature, *i.e.* relations of functional groups to one another and to rings, etc. A list of these features is attached to the name of the molecule.

*Main Part of the Program.* In the next step of the program the first item on NEWLIST is subjected to the action of a subroutine called GENERATEPREDECESSORS. This subroutine will be repeatedly called and does the main work of the program. Initially, of course, the first item on NEWLIST is the only item on NEWLIST, *i.e.* the name of the goal molecule. For generality let us call the input molecule to GENERATEPREDECESSORS by the name X. GENERATEPREDECESSORS first decides whether or not it has previously seen this molecule in the solution of this problem. It searches for an isomer of X on OLDLIST. If there are no such isomers we proceed to the next phase. All isomers of X on OLDLIST are subjected to a detailed examination to see if they have exactly the same structure as X. If there is a molecule, *e.g.* by the name of Y, which is on OLDLIST but which has the same structure as X then there must exist two separate pathways from this structure to the goal molecule. The pathway of higher score, *i.e.* greater efficiency, is chosen. The other one is erased from memory. If X is erased we go back to get the first item on NEWLIST and subject it in turn to GENERATEPREDECESSORS etc. If Y is erased we continue with the next phase of GENERATEPREDECESSORS.

*Generation Stage.* GENERATEPREDECESSORS now collects the list of reactions associated with each structural feature of X. (For example if X has the amide functional group, there are associated with this functional group the list of two reactions: aminolysis of methyl ester and aminolysis of acyl chloride.) For each reaction there is a reaction subroutine which is immediately used to generate the appropriate predecessor(s). An example is the generation of bromobenzene as a predecessor of benzoic acid by the reaction subroutine BROMIDECARBOXYLATION. To take another example, if a certain diene and a dienophile can react in a Diels-Alder reaction to give X then the DIELSALDER reaction subroutine generates the diene and a dienophile structures in response to the input of X. For this to happen the procedure EXAMINE must have found a cyclohexenyl group in X with a -M group just outside of the ring, three atoms away from the nearer of the two vinyl carbons in the ring.

*Case Where the Program is Baffled.* If no predecessor of X can be generated, *i.e.* the EXAMINE subroutine has failed to detect a feature familiar to it and the program is therefore unable to find an applicable reaction subroutine, then the structure and the arbitrary name of X are printed out together with the question, "How do you make this?"

*Disposition of the Predecessors.* As soon as each predecessor is generated it is disposed of in some way. Let us call the predecessor Z. There are four cases.

A. If the overall yield from Z to the goal molecule is less than the cutoff value discussed previously then the pathway and yield are printed out together with

the comment that the overall yield is inadequate to justify further development of this pathway. Then the pathway is removed from the computer memory, "erased," an absolutely necessary step to preserve unfilled memory space.

B. If Z is available *and* all necessary coreactants on the path to the goal molecule are also either available or can be synthesized by a method now known to the program, then the entire pathway from the available predecessor to the goal molecule is printed out together with the yield of each step, the name of each reaction and the overall yield. The program recognizes as available any molecule which satisfies any one of the following three criteria:

1. It has no more than eight non-hydrogenic atoms.
2. It is an ester or a carboxylic acid, or it has only one heteroatom, and it has nine non-hydrogenic atoms.
3. A molecule with identical structure appears on a list called AVAILABLELIST.

The last criterion is important in practice because very often we have a large number of byproducts of other processes available to us. If the program knows about them much computer time will be saved.

C. If Z is available but on the pathway to the goal molecule there are necessary coreactants which are neither available nor does the program yet have a synthesis for them, then no action is taken in regard to Z. The information that Z is a predecessor of X is permanently attached to the name of X. In this way information about Z can be easily retrieved when and if the entire synthetic route has been generated.

D. If Z is not available and the yield from Z to M is not less than the cutoff value, then Z is put onto NEWLIST. This is the most usual case.

*Scoring.* Just before being put on NEWLIST the predecessor Z must be scored. In the program at present the cost part of the scoring procedure has not yet been reduced to specific monetary units. It is assumed for the present that cost depends on the overall yield, the number of steps required and the amount of material carried through the route. To illustrate the last point we note that a reaction with a 50% yield followed by a reaction with a 100% yield is cheaper, other things being equal, than a two step sequence consisting of a reaction with a 100% yield followed by a reaction with a 50% yield. The overall yield is the same in both cases but in the first case there is less material carried into the second reaction.

A procedure called SCORE considers all these aspects in calculating a somewhat arbitrary cost. The program knows a yield for each reaction; in some cases this yield depends on the structure of the molecules involved. A monetary cost could also be easily inserted if the necessary chemical engineering data were available.


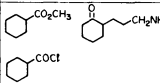
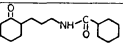
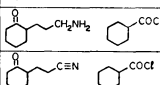
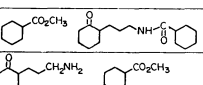
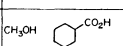
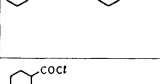
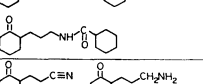

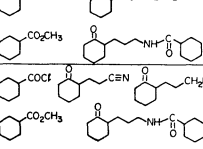
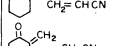
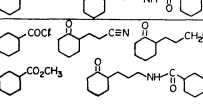
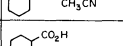
When there are two or more coreactants generated as predecessors SCORE gives all of them the same score which is that calculated for the largest of the molecules. SCORE calculates the cost expended for the amount of simplification achieved in the largest of the coreactants. The cost is the cost of going from Z to M and the "simplification" is really the number of atoms and

functional groups added in going from Z to M.

**Hierarchy of NEWLIST.** Each predecessor of X is placed on NEWLIST in a definite position which depends upon its score. All molecules closer to the front of the list than Z must have a score greater or equal to that of Z. All molecules which follow Z on NEWLIST must have a lower score than Z or the same score that Z has. In this way when the program takes the first molecule on NEWLIST for processing by GENERATEPREDECESSORS it is really considering the most promising incomplete pathway.

**End of GENERATEPREDECESSORS.** When, within the limits of the repertoire of reactions of the program, all possible predecessors of X have been generated and suitably placed, the name of X is taken off NEWLIST and placed on OLDLIST.

Table 1.

Cycle (a)	NEWLIST	OLDLIST	Available Molecules Generated
0			
1			
2			
3			
4			
5			

a) cycle: the number of times the sub-routine has been called

### Example of the Operation of the Program

In the example of Fig. 2 GENERATEPREDECESSORS was called five times. The result of each application of this subroutine is shown in Table 1. In this case EXAMINE found that M is an amide and hence GENERATEPREDECESSORS calls in the subroutines for acylating an amine with a methyl ester and with an acyl chloride and generates the corresponding amine and methyl ester and acyl chloride. In the next cycle the methyl ester stands at the head of NEWLIST. EXAMINE has found out that it is an ester, hence the reaction subroutine for esterification is called in, and it generates the carboxylic acid and methanol. The carboxylic acid is available hence it is not put on OLDLIST. The  $\text{CH}_2\text{NH}_2$  functional group is related to the reaction subroutine called HYDRO-

GENATENITRILE. This latter subroutine generates the corresponding nitrile, but recognizes the simultaneous presence of an active carbonyl group that needs to be protected, the ketone group, and a note is included for the final printout that the ketone group must be protected from hydrogenation by a ketal before the reaction is done and that this ketal is to be removed after the reaction is finished. EXAMINE has seen that in the nitrile there are exactly three saturated carbons between two -M groups and hence an alpha-beta addition is suggested. The corresponding subroutine is quite complicated since it generates cyclohexanone and acrylonitrile which can react *via* the Stork enamine reaction<sup>1)</sup> to obtain the nitrile. The subroutine also generates acetonitrile and  $\alpha$ -methylencyclohexanone. In one case a ketone adds across the double bond of an unsaturated nitrile; in the other case a nitrile adds across the double bond of an unsaturated ketone. A note is added in the case of the enamine reaction that the enamine must be prepared first. The yield is also different in the two similar reactions. Since all of the predecessors are available the synthesis of M can be effected in an overall yield of 58%, or 51% depending on the pathway. The successful pathways are printed out.

In the fifth and final cycle of GENERATEPREDECESSORS, cyclohexane carboxylic acid is generated, as the predecessor of the corresponding acyl chloride.

The reader will note that there was no attempt to generate a secondary alcohol, *i.e.* to synthesize the ketone from a secondary alcohol. The reason is that since we do not wish to oxidize alcohols indiscriminately, the exact conditions under which we want to do this should be specified in the routine. At the time of running of this example I had not yet thought out the matter to completion, hence the alcohol oxidation reaction subroutine had not yet been written. (The conclusion now reached is that it is perfectly satisfactory to oxidize any alcohol to the corresponding aldehyde or ketone, provided that the next reaction step is not hydrogenation of the aldehyde or ketone.)

### Conclusion

Computing synthetic pathways is now possible without human intervention at any intermediate stage. The process is expensive, however, and only when it is made cheap will a computer program become indispensable to the synthetic chemist.

The author gratefully acknowledges support of this work by the John Simon Guggenheim Memorial Foundation and by the National Research Council of Canada.

1) G. Stork, *J. Amer. Chem. Soc.*, **85**, 207 (1963).