Automatic Strategy in Computer Design of Synthesis: An Example in Organophosphorus Chemistry

M. H. ZIMMER, F. CHOPLIN, P. BONNET, and G. KAUFMANN*

Laboratoire de Modèles Informatiques Appliqués à la Synthèse, Institut Le Bel, 67000 Strasbourg, France

Received April 5, 1979

Complex synthetic trees are automatically generated by strategy automata which have been developed for the PASCOP system. The implementation and general features of these automata are described; they are independent of the chemical file, they can access any transform, and they can be based on very general goals. A currently running example is presented—the ARBUSOV automaton, used to disconnect the phosphorus carbon bond in a P(O)–C moiety—and a sample of a synthetic tree produced by this program is presented and discussed.

INTRODUCTION

Much work has been done recently in the field of computer design of synthesis,[1] and from the user's standpoint it appears interesting to summarize the options offered by the existing programs to the chemist, who uses them to plan synthetic routes for a given target molecule. At present there are three main such options, described as follows. (i) The user can let the program run freely (no strategy option), and get all the immediate precursors for his target. No intervention is required from the user, except to select a next target among the already obtained structures. In this case, all the solutions contained in the program are displayed, but this can lead to a large number of synthetically and chemically irrelevant precursors, and does not solve the difficult problem of handling functional group interchanges and introductions in the retrosynthetic direction.[1,2] (ii) Another way is to guide the development of the synthetic tree by means of goals provided by the chemist himself (interactive strategy option). These strategies operate on one (possibly two or three) level of the tree, but are restricted to the target under scrutiny.[1b,3] This allows the user to direct the construction of the tree, according to his own chemical experience and intuition, but prevents full use of the possibilities of the program. (iii) A third option consists of relying entirely on the program, which builds automatically a tree in specific synthetic situations. The synthesis system is monitored by a strategy automaton which keeps entire control of the building of the tree; such are the supertransforms developed by Corey, used to disconnect strategic bonds in polycyclic structures,[4] or to perform other well-defined tasks.

Every option has its advantages and shortcomings, but the third one is very interesting from several standpoints: it does not require additional information from the user, it can treat automatically entire fields of chemistry, and it is very efficient since many irrelevant chemical reactions are ignored, which would otherwise have been used. In the course of our work to build a strategy module for the PASCOP system,[5] we have been interested in implementing strategy automata for the design of synthetic pathways in organophosphorus chemistry. However, the language and the programs developed to create and run these automata are very general and can be used in any other area of chemistry. This paper describes the logical structure and implementation of these automata, and provides an example of a presently running program in the field of organophosphorus chemistry.

ORGANIZATION OF THE SYSTEM

A block diagram given by Figure 1 shows the high level organization of the PASCOP system and the relationships between its modules. Since several of them are derived from the SECS program,[1c] they will be described here only briefly. The target molecule is entered by the chemist via a teletype and a graphics terminal (Tektronix 4012), and the INPUT module creates a connection table from the drawing.[1c] Then, the PERCEPTION module performs a recognition of the topological and chemical features of the target under scrutiny.[6] If no strategy is used, control goes to the TRANSFORM INTERPRETER, which accesses the files of chemical transforms,[7] interprets them, and builds a list of precursors. This list is treated by the OUTPUT module, which evaluates each precursor and displays the current synthetic tree. Control goes back to the INPUT module to allow the chemist to select a new target among the already obtained precursors.

When strategy is required, goals can be interactively expressed by the user, who types in these goals[3] by means of an English-like language (see later). Then, after perception of the target, control goes to the STRATEGY module, in which the on-line STRATCOMP compiler[8] (see Figure 2) translates into binary code the sentences describing the strategy entered by the chemist. Goals can also be provided by a strategy automaton: in this case, the binary code of an automaton, which is stored on a separate file, is directly accessed by the STRATEGY module, the task of which is to perform an a priori selection of the transforms[7] relevant to the goals, and transmit this set of selected transforms to the TRANSFORM INTERPRETER. When precursors are generated, the STRATEGY module checks if they satisfy the required goals, and deletes those which do not. More detailed description of the functions of this module is given in the paragraph dealing with the running of an automaton.

GENERAL ANALYSIS OF A STRATEGY AUTOMATON

A strategy automaton can be considered formally as a flowchart, which gathers all the heuristics used by a chemist to solve a synthetic problem. Its aim is to describe (and generate via the STRATEGY module) the pathways corresponding to the synthesis of a given substructure in a target molecule. This is realized by defining a final goal, such as the retrosynthetic disconnection (or connection) of one or several bonds in the substructure, by means of one or more transforms available in the chemistry file. These transforms can be hierarchically employed or not, according to their synthetic importance. However, since in many cases the final goal cannot be attained after one step (i.e., at the first level of precursors), removal of any obstacle is attempted by use of relevant transforms. Such unfavorable situations are recognized by careful analysis of the topological and chemical environment around the initial substructure, and can arise from the presence of an interfering functional group or pattern of atoms, or a substituent at a phosphorus atom, etc. In this case, the automaton provides a set of choices, each one being ex-

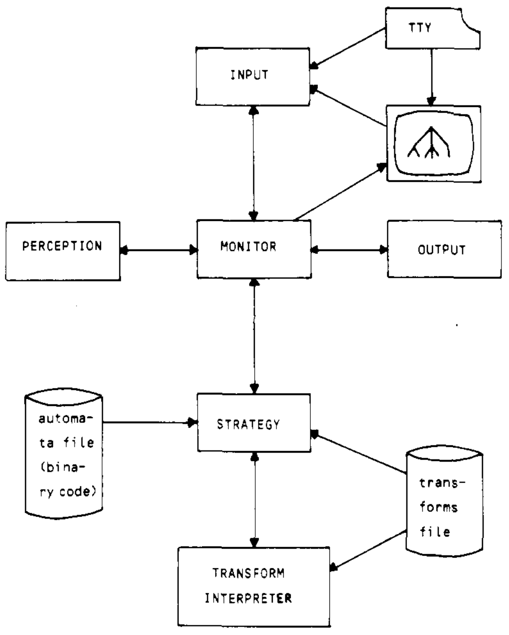


Figure 1. Block diagram of the organization of the PASCOP system.

plored in a depth-first fashion, until one or several valid precursors are obtained (valid here means that the precursor meets the final goal requirement). Although this general philosophy is analogous to that of Corey's supertransforms,[4] it must be noted that automata have specific characters which differentiate them from Corey's work: the synthetic tree obtained is of general type (and not only binary), the final goal can require application of a set of transforms (and not only a specific one), and an automaton is physically independent of the transforms file, which means that it can call any transform, regardless of its type. These differences are treated in the following paragraphs, which show how strategy automata are written and run.

COMPUTER IMPLEMENTATION OF AUTOMATA

Two steps are needed to write a strategy automaton: (i) the chemist has to define the general aim and the final goal of its automaton, and develop the flowchart of all the chemical obstacles and solutions encountered to reach the final goal; (ii) in a second step, he translates this flowchart into a computer-readable form, by means of the strategy language, and creates in this way a source automaton which is stored in a file. Again, two steps are necessary to run this automaton: (i) translation of the source into binary code is performed by the STRATCOMP compiler, and (ii) execution of this code is achieved via the STRATEGY module of the system.

Writing of a Source Automaton. Before describing how an automaton can be written by the chemist, it is necessary to give some details about the tool used for this purpose. The strategy language developed in our laboratory allows the chemist to clearly describe goals, and it possesses many common features with ALCHEM,[9] employed to write chemical transforms. Although it has been fully described earlier,[3,8] basic definitions can be recalled here. A strategy, i.e., an organized collection of goals, is expressed by a set of sentences, which are of three types: (i) directives, which clearly describe synthetic goals to be performed on the target; (ii) inquiries, used to study the topological and chemical environment of a substructure; (iii) instructions, to perform nonchemical operations, such as set manipulations, subroutine call, description of choices, etc. Directives can be linked by logical operators (AND, OR, NOT), but this possibility is especially useful in interactive strategy.[3] The source written by means of this language is translated into binary code by the STRATCOMP compiler[8] (used here as a stand-alone program).

The examples of source automaton given here proceed from the ARBUSOV automaton, the more important of several automata which have been used satisfactorily in our laboratory for many months. Its general aim is to generate a synthetic tree for any molecule containing a P(=O)–C moiety, the final goal required being the retrosynthetic cleavage of the PC bond. This automaton is especially useful since, inter alia, it has to plan pathways for phosphonates, a very important class of organophosphorus compounds needed in the Horner–Emmons reaction.[10] Complete chemical description of ARBUSOV is beyond the scope of this paper and will be reported later.[11] It must be noted, however, that the programs of the STRATEGY module and the strategy language have been developed in a general purpose, and allow the user to deal with any aspect of organic chemistry. The examples which are provided here illustrate the treatment of many possible situations encountered in the creation of a synthetic tree.

As shown by the listing in Table I, the first two instructions of the ARBUSOV automaton are:

```
(1)          ARBUSOV: P(=O)-C/
(2)                MINREQ 1-3
```

The first one gives, along with the mnemonic name, the linear description of the required substructure, in a way analogous to that used in transforms.[1b] Atoms are internally numbered according to their position in this description. The optional instruction MINREQ (MINimal REQuirement) indicates the numbering of the atoms and the bonds which must be present in a structure taken as intermediate target, this option being useful when fragments are produced. As an example, the following transform leads to two fragments:

$$(RO)_2-\underset{\underset{O}{\|}}{P}-CH_2-CH_2-NR'_2 \Rightarrow \underset{\mathbf{a}}{(RO)_2-\underset{\underset{O}{\|}}{P}-CH_2-CH_2-X} + \underset{\mathbf{b}}{H-NR'_2}$$

and the automaton can select fragment **a** as a next target, since the PC bond is not yet broken (which corresponds to the bond between atoms 1 and 3 in the substructure).

Then, structural inquiries analyze the different possible topological and chemical environments of the substructure (see Table I). As soon as a new situation is detected, several choices or branching to another part of the automaton are presented. When there is an obstacle which prevents the fulfillment of the final goal, such as the presence of an interfering functional group, or the presence of a substituent at a phosphorus atom, special treatment is required. Thus, no retrosynthetic cleavage of the PC bond, via the transform describing the Arbusov reaction, can be obtained if a primary or secondary amine is present,[5a] or, generally, use of the Arbusov or Michaelis–Becker reactions is not recommended to prepare β-ketophosphonates.[12] Protection of functional groups (retrosynthetic deprotection) is made by special routines, which provide a solution for any type of encountered group. As an example, the protection routine for an acid is written as follows:

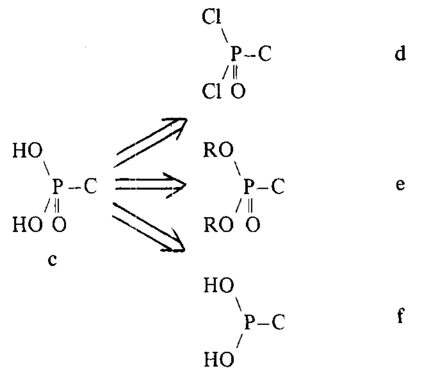
```
ENTRY PROTECACIDE
PRACID 1: IF ATOM IS ACID (1) THEN
              BEGIN
                CALL 7 ACIDE AT (1)
                GO TO PRACID 1
              DONE
END-ENTRY
```

where 7ACIDE is the mnemonic name of the transform which describes the retrosynthetic interchange of an acid into an ester. (*N*) represents the number of a set register where intermediate results can be stored. At any stage in the automaton, as soon as an interfering group is detected, protection can be required by the instruction:

```
CALLSP SUBROUTINE NAME
```

CALL being reserved for the access to a transform in the chemical file, as shown by the instruction CALL 7ACIDE.

To solve a given problem, several solutions can generally be proposed. In order to write an automaton as completely as possible, the writer must describe all these choices. Thus, several possibilities can be envisioned to modify the substituents at a phosphorus atom in **c**, to make feasible the retrosynthetic disconnection of the PC bond. When a P–OH group is encountered, one can think of the three following transforms (among others):



Such a set of solutions is described by means of the TRY CHOICES instruction, the syntax of which is indicated below:

```
TRY CHOICES
  CALL PCLOH AT ATOM 1   } step c⇒d
  GO TO PAMINE
-NEXT CHOICE:
  CALL 1 POROH AT ATOM 1 } step c⇒e
  GO TO PAMINE
-NEXT CHOICE
  CALL POXYD AT ATOM 1   } set c⇒f
  GO TO PHYDROGEN
ENDCHOICE
```

In these directives, PCLOH, 1POROH, POXYD are the mnemonic names of the transforms called to remove the obstacles; PAMINE and PHYDROGEN are labels which allow branching to another part of the automaton. Any available transform of any type can thus be accessed. In the above example, instead of calling specific transforms, the writer could also simply define the type of operation to be performed; thus a functional group interchange is required by the directive:

```
EXCHANGE AMINE INTO AMIDE AT ATOM 3
```

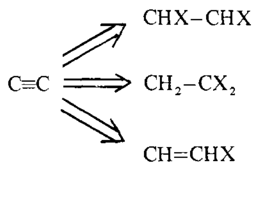
and a substitution at a phosphorus atom by:

```
MAKE SUBSTITUTION POH/POR AT ATOM 1
```

However, in this case, the program uses all the transforms satisfying this requirement, and a set of precursors can be obtained, especially for a functional group interchange which can be performed by several transforms. Thus, if a directive

```
EXCHANGE ACETYLENE INTO
                    HALIDE AT ATOM N
```

is employed, three precursors are obtained, corresponding to the transforms available in the chemistry file:



Since the writer must know exactly which precursor is generated, in order to continue writing the automaton (except for the last step where precursors have reached the final goal), it is preferable in many instances to call directly the transform which performs exactly the desired operation. Thus, the structure which is taken as next target is perfectly known, and no problem can arise at this point.

Inside a choice block, other choices can be described by use of the same instructions. A complex set of solutions can thus be written in the following way:

```
TRY CHOICES:
  choice 11
-NEXT CHOICE:
  CALL transform
  TRY CHOICES
    choice 21
  -NEXT CHOICE:
    choice 22
  -NEXT CHOICE:
  . . .
  ENDCHOICE
-NEXT CHOICE:
  choice 13
  . . .
ENDCHOICE
```

When an obstacle is cleared, next inquiries are processed in order to detect and resolve other situations. When all the obstacles are cleared, fulfillment of the final goal is required. One can indicate the transforms to be applied, with a TRY CHOICES instruction; it is better to describe clearly this goal by means of one or several directives. In the ARBUSOV automaton, a directive

```
BREAK BOND 1-3
```

is used, so that all the relevant transforms can be selected in the transform file. An END instruction indicates the physical end of the automaton. The binary code created from this source by the STRATCOMP compiler is stored on a file, and can be accessed by the STRATEGY module to generate a complete synthetic tree, for any target including the P(=O)–C structure, as shown in the next paragraphs.

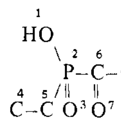
Running of an Automaton. The call to an automaton is interactively made by the chemist who types the instruction:

PERFORM automaton name [AT *N*1, *N*2, *N*3, . . .]

where the text between brackets is optional, and *N*1, *N*2, *N*3, . . . are the input numbers of the target atoms corresponding to the substructure atoms. Thus, the instruction:

```
PERFORM ARBUSOV AT 2, 3, 6
```

can be used to process a target such as **g**.


g

Let us recall that the ARBUSOV substructure is P(=O)–C. One could have also declared:

PERFORM ARBUSOV AT 2, or PERFORM ARBUSOV

If the user wants to try all the automata available in the library, he will type:

```
PERFORM ALL
```

At this point, the binary code of the automaton is loaded into a buffer of the STRATEGY module (a directory included in the automata file makes possible the direct access to any of them contained in this file). Actually, this module contains two main parts (as shown in Figure 2): the compiler, used

Table I. Partial Listing of the Source of the ARBUSOV Automation[a]

```
 1   ARBUSOV : P(=O) - C/
 2   MINREQ 1 - 3
 3   POXYHYDRO :
 4           IF OXYGEN IS ALPHA TO ATOM 1 OFFPATH (1)
 5           IF (1) IS ATTACHED TO 1 HYDROGEN (2) THEN GO TO POH
 6   PAMINE :
 7           IF NITROGEN IS ALPHA TO ATOM 1 OFFPATH (1) THEN GO TO PN
 8   PHALOGENE :
 9           IF HALOGEN IS ALPHA TO ATOM 1 (1) THEN GO TO PCL
             .........
10   PROTECTION :
             .........
11   HALOSURPC :
12           IF HALOGEN IS WITHIN EPSILON TO ATOM 3 OFFPATH THEN
13           GO TO CHALOGENE
             .........
14   CDOUBLIAISON :
15           IF OLEFIN IS ATOM 3 THEN GO TO CDLIAISON
16   OXYGENSURPC
             .........
17           GO TO FIN
18   POH :   ; phosphinic and phosphonic acids
19           IF ATOM 3 IS OLEFIN THEN
                    BEGIN TRY CHOICES
21                       CALL 3196KOSO AT ATOM 1
22                       STOP
23                     - NEXT CHOICE :
24                       MAKE SUBSTITUTION POH/POR  AT ATOM 1
25                       GO TO PAMINE
26                     - NEXT CHOICE :
27                       MAKE SUBSTITUTION POH/PCL AT ATOM 1
28                       GO TO PAMINE
29                    ENDCHOICE
30                 DONE
                   ........
31   PN :    ; N - P (=O) - C substructure
             .........
32   PCL :   ; CL - P(=O) - C substructure
33           IF ATOM 3 IS OLEFIN THEN
34                  BEGIN IF DOUBLE BOND IS ATOM 3 (2)
35                         IF CARBON OFFPATH IS (2)(1)
36                         CALL 4OLEFIN AT (1)
37                  DONE
                    ........
38                  TRY CHOICES
39                      GO TO FIN
40                    - NEXT CHOICE :
41                      CALL PXHYDROL AT ATOM 1
42                      GO TO FIN
43                    - NEXT CHOICE :
44                      MAKE SUBSTITUTION PCL/POR AT ATOM 1
45                      GO TO PROTECTION
46                    - NEXT CHOICE :
47                      CALL POXYD AT ATOM 1
48                      GO TO FIN
49                  ENDCHOICE
                    ........
50   CHALOGENE :
             ; an halide is on the carbon chain
             ..........
51           IF ATOM ALPHA TO ATOM 3 OFFPATH IS HALIDE THEN GO TO FIN
             ..........
52   CDLIAISON :
             ; carbon atom is on an olefin
53           IF DOUBLE BOND IS ATOM 3 (2)
54           IF CARBON OFFPATH IS (2) (1)
             ..........
55           IF (1) IS PRIMARY ATOM THEN
56                  BEGIN TRY CHOICES
57                      GO TO FIN
58                    - NEXT CHOICE :
59                      CALL 1PXPOR3 AT ATOM 1
60                      GO TO PHALOGENE
61                    - NEXT CHOICE :
62                      CALL WITHORNER AT (1)
63                      GO TO OXYGENSURPC
64                    - NEXT CHOICE :
65                      CALL 4OLEFIN AT (1)
66                      GO TO HALOSURPC
67                  ENDCHOICE
                    ........
68   FIN :
69           BREAK BOND 1-3
70           STOP
71           END
```

[a] Are given the directives and statements used to generate the synthetic tree shown by Figure 3. Numbers in parentheses represent set registers,[1c] where any intermediate variable can be stored and manipulated, up to seven registers being available. A semicolon indicates a comment.

on an interactive basis, and the STRATX routine (STRATegy eXecution), which scans the binary code of the automaton (or the code generated by the on-line compiler, on interactive strategy mode). When an instruction or an inquiry is met, control goes to the INTINS (interpretation of instructions), or INTINQ routines (interpretation of inquiries). A directive requires a chemical or topological operation to be performed on the target, by means of one or several transforms. In this case, the INTDIR routine (interpretation of directives) creates the set of transforms relevant to the goals described by the directives. To perform this operation, it needs additional information, which is available from the transform file, where an inverted subfile provides, for each type of directive, the set of transforms which can possibly satisfy this goal.[3,8] This information is stored as bit chains, making easier the internal manipulation of logical combinations of directives.

The set of transforms, generated by STRATX (via INTDIR), contains in general one transform (except when application of the final goal is required), which is transmitted to the TRANSFORM INTERPRETER. If the transform succeeds, a new precursor is created, and if it does not fulfill the final goal requirement, it will be taken as a next intermediate target, which is analyzed by the PERCEPTION module. A new node is thus added to the synthetic tree, which is redisplayed. Any node without precursors is deleted except those which satisfy the final goal requirement, or correspond to duplications (see later). Then, control is returned to the STRATX routine, and scanning of the binary code is resumed, until all the choices corresponding to the current problem have been explored.

OUTPUT EXAMPLE

Figure 3 shows straightforward synthetic sequences automatically generated by the ARBUSOV automaton for vinylphosphonic acid (**1**) as target. It must be emphasized that if **1** is processed without the use of any strategy, no interesting result is obtained, owing to the presence of the interfering phosphonic acid group. Before comparing with experimental results, let us examine how this synthetic tree is produced; Table I gathers all the statements and directives of ARBUSOV, used in this specific case. After perception of the target **1** (Figure 3), the program recognizes the presence of a P–OH moiety; starting from the beginning of the automaton, inquiries (lines 4–5) lead to POH (1.18) (see Table I), where several choices are described: choice 1 (transform whose mnemonic

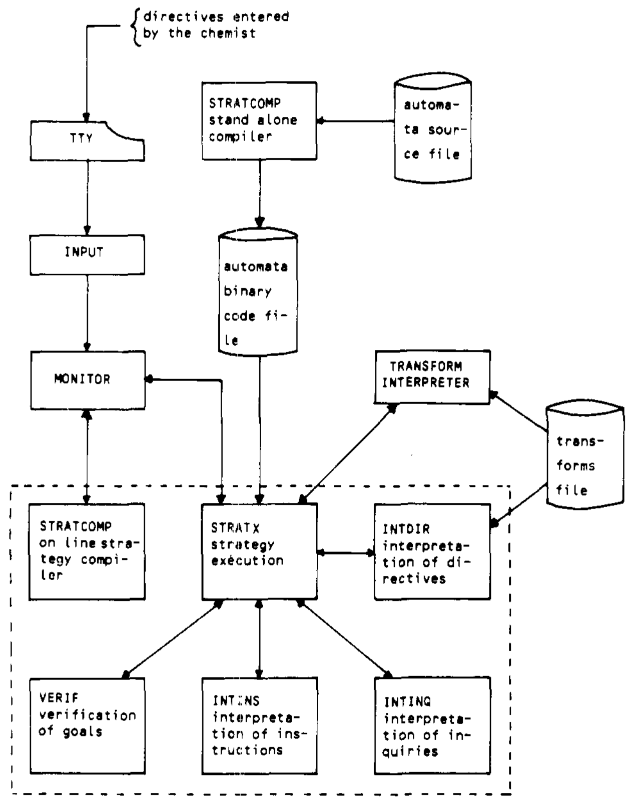

directives entered by the chemist
TTY
STRATCOMP stand alone compiler
automata source file
INPUT
automata binary code file
MONITOR
TRANSFORM INTERPRETER
transforms file
STRATCOMP on line strategy compiler
STRATX strategy execution
INTDIR interpretation of directives
VERIF verification of goals
INTINS interpretation of instructions
INTINQ interpretation of inquiries

Figure 2. Organization of the STRATEGY module. Routines which belong to this module are included in the dotted line box.

name is 3196KOSO, 1.21) produces precursors **2** + **3** which satisfy the final goal, choice 2 (1.24) is then processed and gives precursor **4**, which is perceived, and the analysis restarts at the PAMINE label (1.6). The first positive answer to an inquiry is met after CDOUBLIAISON (1.14), and leads to CDLIAISON (1.52). Four choices are then described: the first one (GO TO FIN) (1.57) corresponds to the final goal requirement, and precursors **6**–**16** reach this goal. The analysis does not restart from these structures, since a STOP instruction is met in the flowchart. Next choice corresponding to **4** is processed (1.59), and represents the retrosynthetic exchange P–OR ==> P–Cl (i.e., in the synthetic direction, the reaction of a PCl bond with an alcohol ROH). Precursors **17** + **18** are thus obtained.

The program recognizes that only **17** can be taken as next intermediate target, and starting at PHALOGENE (1.8), the analysis leads to PCL (1.32) where an exchange of the olefin into an halide is required (transform 4OLEFIN) (1.36). As soon as precursor **21** is created, the choices which follow are interpreted: structures **28**–**34** satisfy the final goals (1.39), and the next choice corresponds to the synthetic hydrolysis of a PCl bond (1.41) (precursor **35**). Starting from **35** as target, the program interprets the transforms which can disconnect the PC bond. Precursors **38**–**41** are thus obtained. Next choice related to **21** is processed (1.44) (a substitution at the phosphorus atom), giving **36**: through inquiries under labels PROTECTION (1.10), HALOSURPC (1.11), CHALOGENE (1.50), disconnection of the PC bond is required, and produces structures **42**–**50**. The last choice for **21** (1.47) corresponds to the phosphorus oxidation in **37**, which itself leads to **51**–**52** as final precursors.

All the choices have been considered for **21**, and the program goes back to **4**, where other choices are still to be processed. Structure **4** is then perceived again, since the perception result for only one structure can be present in the core at a given time. Let us remember that all the choices associated with **4** are gathered under label CDLIAISON (1.52), and the choice that is to be processed represents the application of the transform WITHORNER (1.62) (retrosynthetic description

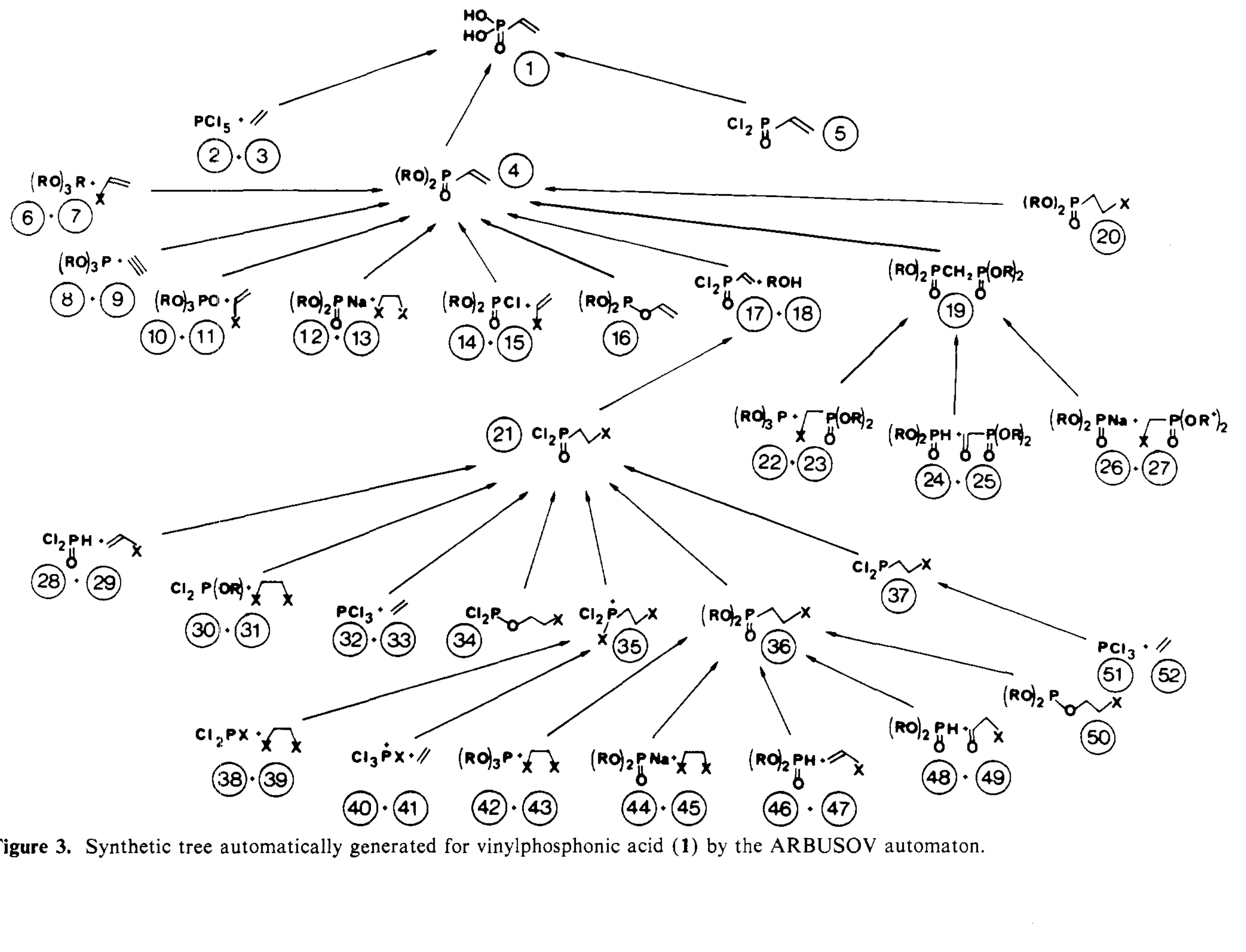
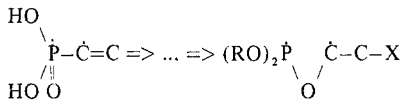


Figure 3. Synthetic tree automatically generated for vinylphosphonic acid (**1**) by the ARBUSOV automaton.

of the Horner–Emmons reaction). Precursor **19** is obtained and leads to **22–27**. Last choice for **4** requires the olefin interchange into a halide (1.65): at this point, the program recognizes that **20** and **36** are identical and does not process further structure **20**. In the same way, **5**, which is the precursor corresponding to the last choice associated to the initial target **1**, is found identical with **21**, and the analysis is ended. Fast recognition of these duplications is made by means of a canonical name generated for each structure,[5b–13] and these duplications are easily recognized by the user, since they are the only leaves in the tree which do not satisfy the final goal.

Comparison with experimental results indicates that sequence **50** → **36** → **21** → **17** (≡**5**) → **1** is the most common for the given target (with R = CH_2CH_2Cl).[14] However, preparation of 2-chloroethyl dialkyl phosphite (**50**) from $(RO)_2PCl$ and ethylene oxide is not shown by the program, since, in **50**, the final goal of the automaton has been satisfied, through disconnection of the PC bond:



Other sequences have been described in the literature, such as (**38** + **39**) → **35** → **21** → **17** (≡**5**) → **1**.[15] Step (**2** + **3**) → **1** has been realized with substituted olefins.[16] Arbusov reaction of trialkyl phosphite with vinyl chloride, in the presence of $NiCl_2$ as a catalyst, has been reported,[17] and corresponds to steps (**6** + **7**) → **4**, but Michaelis–Becker reaction involving 1,2-dichloroethane **13** and **12** (or **44** + **45**) leads to a mixture of products depending on the conditions.[18] Addition of triethyl phosphite **8** with acetylene **9** is not recommended in this case,[19] since a dienophile is required. This holds also for steps (**28** + **29**) → **21**, and (**46** + **47**) → **36**, in which free radical conditions are required if the olefin is not correctly activated.[20] Grignard reactions can also be considered [(**10** + **11**) → **4**, (**14** + **15**) → **4**]. The ARBUSOV rearrangement used in steps **16** → **4**, **34** → **21**, is almost always proposed by the program in such cases: this arises from the fact that we have little information about the synthetic possibilities of this reaction;[21] therefore, the corresponding transform in the chemistry file is poorly documented, and is called very often. Step (**42** + **43**) → **36** has been described,[22] but not in the case of an alkyl dichlorophosphite **30** (however, Arbusov reactions with compounds such as **30** have been reported[23]). Functional group interchanges (dehydrohalogenation of **20** and **21**) and substitutions at the phosphorus atom (**4** → **1**, **5** → **1**, **36** → **21**) do not involve any special problem. Preparation of **4** via a Horner–Emmons reaction has been reported[24] (steps (**22** + **23**) → **19** → **4**). Precursors (**24** + **25**), (**48** + **49**), correspond to the reaction of a PH bond with an aldehyde (via a tosylhydrazone), and which has been described in somewhat different situations.[25]

Actually, it is readily seen that relatively few transforms are involved in the building of the tree. As soon as obstacles are cleared, by functional group interchanges or substitutions at phosphorus, almost similar sets of transforms are used to disconnect the PC bond. Moreover, some sequences are redundant. Thus, (**32** + **33**) → **21** is analogous to (**51** + **52**) → **37** → **21**. This corresponds to the fact that the chemistry file includes global transforms (which involve several elementary steps, such as addition of a PCl bond to an olefin, and oxidation of a trivalent phosphorus, in (**32** + **33**) → **21**), and elementary transforms which must be present in order to have a file as general and complete as possible (thus, in (**51** + **52**) → **37** → **21**, PCl bond addition to an olefin and phosphorus oxidation are described successively).

Execution time of this automaton is rather short, since relatively few transforms are called, and duplications are easily and early recognized. The current example needed 45 s (CPU time), on an UNIVAC 1110 machine. This time should be shorter, however, since there is duplication of the inquiries made in the transforms and in the automaton, because the latter is physically independent of the transform file.

CONCLUSION

Strategy automata, which generate automatically a synthetic tree for a given target molecule, have been developed for the PASCOP system. Their specific features are as follows: they are physically independent of the chemical file, where they can access any transform, and they can build a synthetic tree based on a rather general goal. They are intended to describe well-defined areas of chemistry, and thus allow the user to rely on them to plan complex synthetic pathways. Work is currently under way to study their performances thoroughly, extend the possibilities of the existing automata, and develop new ones.

ACKNOWLEDGMENT

M. H. Zimmer thanks the Roussel-Uclaf Company for a grant. We are indebted to Dr. C. Laurenço for helpful discussions and comments.

REFERENCES AND NOTES

(1) (a) E. J. Corey and W. T. Wipke, *Science*, **166**, 178 (1969); (b) W. T. Wipke, G. I. Ouchi, and F. Krishnan, *Artif. Intell.*, **11**, 173 (1978); (c) W. T. Wipke, H. Braun, G. Smith, F. Choplin, and W. Sieber, "SECS-Simulation and Evaluation of Chemical Syntheses: Strategy and Planning" in "Computer Assisted Organic Synthesis", ACS Symposium Series, No. 61, American Chemical Society, Washington, D.C., 1977, p 97; (d) G. Moreau, *Nouv. J. Chim.*, **2**, 187 (1978); (e) R. Barone and M. Chanon, *ibid.*, **2**, 659 (1978); (f) J. Gasteiger and C. Jochum, *Top. Curr. Chem.*, **74**, 93 (1978); (g) E. J. Corey and A. K. Long, *J. Org. Chem.*, **43**, 2208 (1978); (h) H. L. Gelernter, A. F. Sanders, D. L. Larsen, K. K. Agarwal, R. H. Boivie, G. A. Spritzer, and J. E. Searleman, *Science*, **197**, 1041 (1977); (i) M. Bersohn and A. Esack, *Chem. Rev.*, **76**, 269 (1976).

(2) (a) H. W. Whitlock, *J. Am. Chem. Soc.*, **98**, 3225 (1976); (b) E. J. Corey and W. Jorgensen, *ibid.*, **98**, 203 (1976).

(3) F. Choplin, P. Bonnet, M. H. Zimmer, and G. Kaufmann, *Nouv. J. Chim.*, **3**, 223 (1979).

(4) (a) E. J. Corey, W. J. Howe, and D. A. Pensak, *J. Am. Chem. Soc.*, **96**, 7724 (1974); (b) E. J. Corey, W. J. Howe, H. W. Orf, D. A. Pensak, and G. Petersson, *ibid.*, **97**, 6116 (1975).

(5) PASCOP stands for Programme d'Aide a la Synthèse des Composês Organo Phosphorês. It is derived from the version 2.0 of SECS:[1c] (a) F. Choplin, C. Laurenço, R. Marc, G. Kaufmann, and W. T. Wipke, *Nouv. J. Chim.*, **2**, 285 (1978); (b) F. Choplin, R. Marc, G. Kaufmann, and W. T. Wipke, *J. Chem. Inf. Comput. Sci.*, **18**, 110 (1978).

(6) E. J. Corey, W. T. Wipke, R. D. Cramer, III, and W. J. Howe, *J. Am. Chem. Soc.*, **94**, 431 (1972).

(7) Since the program works in the retrosynthetic direction,[1a] transforms describe chemical reactions in this direction.

(8) (a) P. Bonnet, Thèse de 3ème cycle, University de Nancy, Février 1978; (b) P. Bonnet, J. C. Derniame, M. H. Zimmer, F. Choplin, and G. Kaufmann, accepted in Revue RAIRO de l'AFCET—Intelligence Artificielle.

(9) ALCHEM, Associative Language for CHEMistry: W. T. Wipke, T. M. Dyott, C. Still, P. Friedland, to be published.

(10) (a) R. Engel, *Chem. Rev.*, **77**, 349 (1977); (b) J. Boutagy and R. Thomas, *Chem. Rev.*, **74**, 87 (1974).

(11) M. H. Zimmer, to be published.

(12) F. Mathey and P. Savignac, *Tetrahedron*, **34**, 649 (1978).

(13) W. T. Wipke and T. M. Dyott, *J. Am. Chem. Soc.*, **96**, 4834 (1974).

(14) (a) M. I. Kabachnik and P. A. Rossiiskaya, *Izv. Akad. Nauk SSSR, Otd. Khim. Nauk*, 403 (1946); *Chem. Abstr.*, **42**, 7242c (1948); (b) M. I. Kabachnik and T. Ya Medved, *Izv. Akad. Nauk SSSR, Otd. Khim. Nauk*, 2142 (1959); *Chem. Abstr.*, **54**, 10834e (1960); (c) K. Schimmelschmidt and W. Denk, German Patent 1 023 033 (1958); *Chem. Abstr.*, **54**, 5466d (1960); (d) K. Schimmelschmidt and W. Denk, German Patent 1 023 034 (1958); *Chem. Abstr.*, **54**, 5466c (1960).

(15) J. M. Maynard and J. M. Swan, *Aust. J. Chem.*, **16**, 596 (1963).

(16) G. M. Kosolapoff and W. F. Huber, *J. Am. Chem. Soc.*, **68**, 2540 (1946).

(17) P. Tavs and M. Weitkamp, *Tetrahedron*, **26**, 5529 (1970).

(18) K. A. Petrov, F. L. Maklyaev, and N. K. Bliznyuk, *Zh. Obshch. Khim.*, **30**, 1608 (1960).

(19) R. G. Harvey and E. R. Desombre, *Top. Phosphorus Chem.*, **1**, 93 (1964).

(20) (a) A. R. Stiles, W. E. Vaughan, and F. F. Rust, *J. Am. Chem. Soc.*, **80**, 714 (1958); (b) A. R. Stiles and F. F. Rust, U.S. Patent 2 724 718

(1955); *Chem. Abstr.*, **50**, 10124d (1956).
(21) A. E. Arbusov and L. V. Nesterov, *Dokl. Akad. Nauk SSSR*, **92**, 57 (1953); *Chem. Abstr.*, **48**, 10538b (1954).
(22) (a) A. H. Ford-Moore and J. H. Williams, *J. Chem. Soc.*, 1465 (1947); (b) G. M. Kosolapoff, *J. Am. Chem. Soc.*, **70**, 1971 (1948).
(23) J. Kwiatek, U.S. Patent 2882311 (1959); *Chem. Abstr.*, **53**, 16964i (1959).
(24) R. M. Davidson and G. L. Kenyon, *J. Org. Chem.*, **42**, 1030 (1977).
(25) S. Inokawa, Y. Nakatsukasa, M. Horisaki, M. Yamashita, H. Yoshida, T. Ogata, and H. Inokawa, *Synthesis*, 179 (1977).

An Evaluation of the Leading Patent Equivalents Services[†]

TRISHA M. JOHNS*
G. D. Searle & Company, Skokie, Illinois

WILLIAM G. ANDRUS
The Upjohn Company, Kalamazoo, Michigan

STANLEY DE VOE
Medical Research Division, American Cyanamid Company, Pearl River, New York

JOHN MYERS
A. H. Robins Company, Richmond, Virginia

ROGER GRANT SMITH
Merck & Co., Inc., Rahway, New Jersey

OTOKAR UHLIR
Abbott Laboratories, North Chicago, Illinois

Received May 29, 1979

Patent equivalents services are useful tools for identifying patent families. A patent family consists of a group of patents in various countries which cover the same invention. The patent equivalents services provided by the three leading suppliers—Chemical Abstracts Service, Derwent Publications Ltd., and INPADOC—have been tested and evaluated for coverage, recall, and accuracy. Results are reported for a test sample consisting of 75 pharmaceutical patent families.

INTRODUCTION

Patent equivalents arise when a patentee (either the inventor or the inventor's assignee) applies for patent protection for an invention in a number of countries. The patentee may then be issued patents in several countries, all protecting the same invention. Since it would be misleading to treat each of the patents as if it pertained to a separate invention, it has become the practice of those who deal with patent information to label such patents as "equivalents", "counterparts", "cognates", "corresponding", or, collectively, a "patent family".[1]

Why is it important to know if a given patent has equivalents in other countries? Some of the main reasons are the following:

(1) Avoiding translation–Locating an equivalent in a familiar language is usually cheaper and faster than having a translation made of a patent in an unfamiliar language.

(2) Determining extent of patent protection—Knowledge of the countries in which patent protection for a competing product has been applied for is useful to the business executive in planning marketing strategy and in assessing the competitor's confidence in the product. Newman and Hoegberg[2] stress the importance of such knowledge in making long range decisions.

(3) Disclosure of patentability problems—A patent issued by an examining country[3] will sometimes be delayed if there are legal impediments (e.g., an interference) to the issuance of the patent. The patent attorney can often detect such patentability problems by analyzing the time lags between the filing dates and issue dates of the examined members of a patent family. The attorney may also glean valuable information by comparing the claims of equivalent patent documents.[4] Claims of questionable patentability will probably have been eliminated from the examined patents.

(4) Avoiding double abstracting and indexing—It is far more efficient to abstract and index only one member of a patent family and relate the others by cross-reference than it is to abstract and index every member of the patent family individually. The amount of work saved becomes apparent when one considers the huge volume of new patent documents: the 24 major patent-issuing countries currently generate about 9500 documents per week, of which 5200 are found to be equivalents.[5] By not having to abstract or index these equivalents, 50% of the effort required is saved. This is usually a concern of the large commercial services which abstract and index patents, although there are a number of private firms which do their own patent abstracting and indexing in specific areas.

(5) Avoiding confusion during patent searching–This use of equivalents has been discussed by Valicenti.[6] When more than one data base is used during a patent search, the same patent family may have been uncovered by each of the data bases, but because different members of the family are cited by each service, they superficially appear to be separate, distinct hits. The correlation of these equivalents will reduce the hit count and avoid confusing the recipient of the search.

[†] Presented at the PMA Science Information Subsection Annual Meeting, March 6, 1979.