Synthesis of System Designs:
III. Toward a Process Concept Generator

J. J. SIIROLA, G. J. POWERS and D. F. RUDD
University of Wisconsin, Madison, Wisconsin

Principles are developed for the computer-aided generation of initial process flowsheets. The invention of a process concept is analogous to the proof of a multihypothesis/multiconclusion theorem, a mental activity which has been simulated on the computer. Techniques of artificial intelligence are combined with the concept of equivalence class matching and information flow logic to develop a working, computerized process synthesizer. This is part of a continuing project on the development of principles of process invention.

In this report we examine the use of the digital computer as a problem solver, rather than merely as a calculating machine. One of the most important and difficult areas of engineering problem solving is that of process design synthesis, in which ill-defined processing problems are transformed into systems specifications. Presently, this area of problem solving is solely the domain of the experienced engineer, and it is only in certain limited situations that advances have been made in computer-aided design synthesis (*1* to *5*, *17*, *18*).

We propose to examine certain methods of information handling required to extend the existing methods of system synthesis based on equipment matching to the point where the problem solving abilities of the computer can supplement effectively the engineer during the preliminary creative explorations which lead to process designs. This kind of activity is distinctly different from the present role of the computer in solving completely defined numerical problems associated with the detailed analysis and optimization of previously synthesized processes.

Here we give the computer a summary of the capability of existing process equipment and require the computer to select and assemble equipment into a variety of feasible processes. The nature of the process is the complete responsibility of the computer; we only require that it be consistent with a word statement of the processing need and with the physical conditions of the chemical reactions to be exploited.

The problem solving ability is that of circumventing the overwhelming combinatorial problems encountered during the trial synthesis of equipment interconnections. Previous work demonstrated that heuristic methods can be used effectively to guide equipment matching in certain relatively well-defined synthesis problems, such as heat exchange network design (*2*). However, it remains to be demonstrated that the computer can be programmed to select its own set of plausible equipment matches, upon which to base heuristic selection procedures, when the synthesis problem is as large and ill-defined as are most industrial design problems.

First we examine requirements of equipment matching system synthesizers which arise from conditions of process logic. These limit greatly the kinds of processing detail that can be part of a physically realizable process. Then we allude to means by which the computer can guide itself toward efficient system configurations. Finally, we discuss the means for programming a synthesizer and relate experiences with a first generation of synthesis programs capable of handling general industrial design problems.

In the concluding remarks we indicate how the development of a computerized process flowsheet generator fits in to a larger, long-range project on the development of principles of process invention.

LOGICAL STRUCTURE

A primary source of information useful in the synthesis of a process system from available unit processes is the design procedures which have come into being to describe the performance of the unit processes. This is the kind of information available in handbooks, equipment performance tables, empirical correlations, and the more sophisticated computer subroutines which are now used in process design.

In this section we examine the implications which can be drawn from the existence of such information at the unit process level, and examine the means by which this information can be used during system synthesis. The methods to be used are those of information flow analysis which have been developed for the analysis of existing designs (*6* to *9*).

The design procedures for unit process H may be thought of as employing M_H functional relationships and N_H variables. This may be expressed in the symbolic form of Equation (1), although the relationships need not be of the equation form one usually associates with this notation.

$$f_{iH}(X_{jH}) = 0 \quad \begin{array}{l}(i = 1, 2, \cdots, M_H \text{ functional relationships}) \\ (j = 1, 2, \cdots, N_H \text{ variables}) \\ (N_H > M_H)\end{array} \qquad (1)$$

Thus f_{iH} represents the i^{th} functional relationship used in the design of the unit process H, and X_{jH} represents the j^{th} variable which enters into the design procedures.

It is necessary to specify the numerical values of the excess variables, in order to determine the unit process completely. The numerical values of the other variables may then be found by the solution of the design Equations (1). The number of variables in excess is called the local degree of freedom for the unit process H.

$$D_H = N_H - M_H \qquad (2)$$

Local degrees of freedom are used in two ways: to link a unit process to its environment and to adjust the performance of the system of unit processes to the desired level of performance. The variables used for the first purpose here are referred to as input state variables and those used for the second purpose are called design variables. The variables solved are called output state variables.

An input state variable has the property that its numerical value either is given by the statement of the design problem, or that it is an output variable from some other unit process in the system. The problem of system synthesis by unit process matching is that of connecting output state variables to input state variables so as to generate a system with desired performance characteristics.

A necessary and mathematically sufficient condition for the existence of a process corresponding to a proposed arrangement of unit processes is that the design equations so generated have a meaningful solution. Unfortunately it is impossible to invoke this condition when in the midst of synthesis, since the complete mathematical problem is then yet to be generated. However a weaker condition is useful. The condition is that any subsystem of the system under synthesis must generate design equations which have the logical structure of a solvable set of equations. This is a statement of the diversity condition (*10*) which has been used in the development of methods for the decomposition of complex designs. This condition shall be exploited further in the context of process system synthesis.

AN EXTRACTOR SYSTEM

To illustrate these points we now consider the synthesis of a system of equilibrium stage extraction units using totally immiscible solvents, a problem chosen for its simplicity. The design equations for the extractor H are

$$F_H x_{1H} + W_H y_{1H} = F_H x_{2H} + W_H y_{2H} \quad \text{material balance}$$
$$\phi(x_{2H}, y_{2H}) = 0 \quad \text{equilibrium data}$$

where the notation is defined in Figure 1. The extractor H has four local degrees of freedom which can be consumed in a number of ways. For example, if the two flow rates (F_H and W_H) and the input concentrations (x_{1H} and y_{1H}) are specified, the two design equations can be solved for the effluent concentrations (x_{2H}, y_{2H}). However, if during the consumption of the local degrees of freedom the two effluent concentrations (x_{2H} and y_{2H}) were specified, there would result a set of design equations which cannot be solved. The first design equation would then have two unknowns and the second equation none. The two equations would not satisfy the diversity conditions of Hall, and therefore do not have the structure of a solvable set of equations.

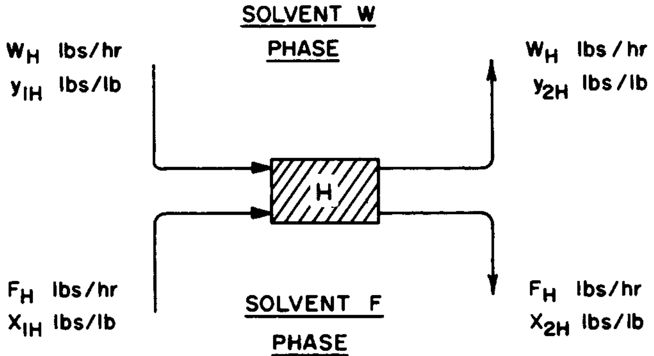


Fig. 1. An equilibrium stage extractor.

Figure 2 illustrates all nonfeasible directions of information flow through the two design equations, and thereby is a catalogue of possible ways in which the extractor H cannot enter into a larger system. Unfortunately any listing of feasible or infeasible information flow structures becomes overwhelming in larger systems, and any method of synthesis which relies on such listings will be inefficient. However, it is instructive to see how certain system structures are eliminated by these considerations well before the complete design problem is established. In practice the consistency tests to be performed now by inspection will be performed by the more powerful list processing methods.

Figure 3 shows the unsynthesized material flow diagram for a process in which a stream of the W phase solvent of specified solute concentration is to be processed with a stream of F phase solvent of known flow rate and specified solute concentration. Further, it is required that F and W phases of required solute concentration be manufactured. In Figure 4 are presented both the material and information flow diagrams for three proposed solutions to this synthe-

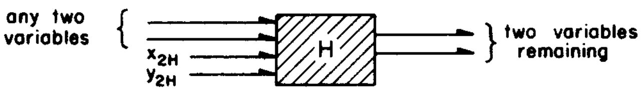
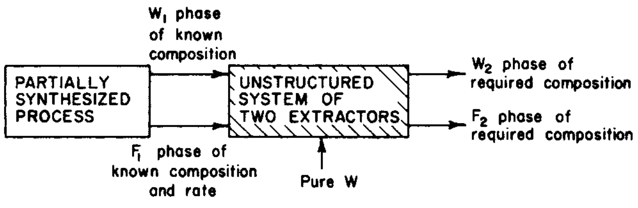


Fig. 2. Nonfeasible information flow for extractor.



W_1 phase of known composition
PARTIALLY SYNTHESIZED PROCESS
UNSTRUCTURED SYSTEM OF TWO EXTRACTORS
W_2 phase of required composition
F_2 phase of required composition
F_1 phase of known composition and rate
Pure W

Fig. 3. An extractor design problem.

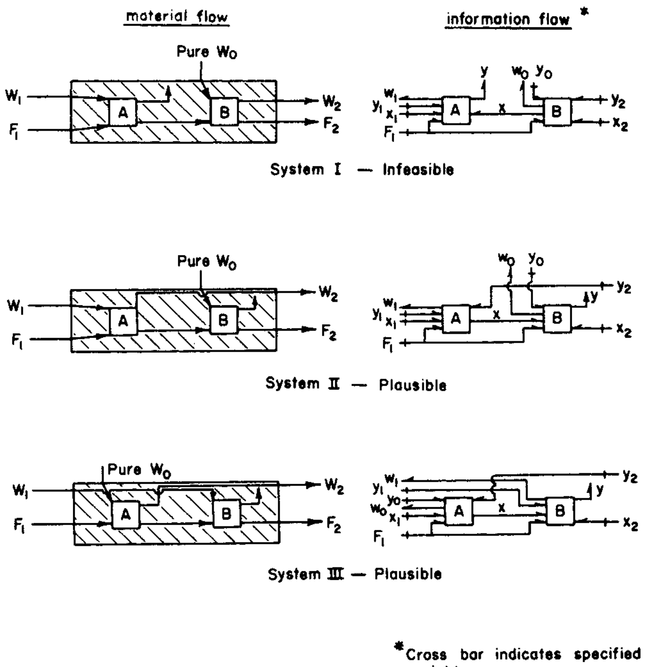


Fig. 4. Three proposed extractor systems.

sis problem. System I in Figure 4 is infeasible since the information flow structure imposed on extractor B is that of the illogical structure appearing in Figure 2. Systems II and III are plausible in that the information flow structures corresponding to these systems are not illogical. However, it remains to be seen if these latter systems are feasible, for the concentrations cited may be in ranges which cannot be reached by the adjustment of the system design variables, or the information flow reversal of rate W may not be allowed to penetrate the partially synthesized process as desired. In other words, it may be necessary to alter the system structure further to transform a plausible system into a feasible system. Information flow studies merely serve to detect illogical systems.

EQUIVALENCE CLASSES

Having introduced the concept of logical consistency as a necessary condition imposed on system synthesizers, we now lay the groundwork for the development of computer-aided process synthesizers. These will be implemented by list processing and string manipulation methods which have been developed for non-numeric computer applications. In this section two equivalence classes are defined into which the system variables must fall, and which impose constraints on the state variable matchings which can be proposed by the synthesizer.

In unit process matching, temperatures must be matched to temperatures, compositions to compositions, and so forth. It is convenient to define a set of matching equivalence classes E_k in such a way that the variables which fall in a matching equivalence class are carriers of the same kind of information and are eligible for state variable matching among themselves.

$$\{X\} \equiv E_k \tag{3}$$

A second useful equivalence class arises from the fact that a system must be realized physically, and that physical connections among unit processes usually involve the simultaneous transfer of mass, energy, and/or momentum. Thus when a composition-composition match is suggested, for example, this may imply a simultaneous flow rate-flow rate match and temperature-temperature match. These additional matches would be implied if the composition match is to be achieved by pumping a fluid between the two unit processes.

The physical connections among unit processes define information channels along which the state variable information carriers travel. Thus the set of variables defines the information channel equivalence class T_r, if the matching of one of the set of variables in T_r to a variable in a second equivalence class T_s implies the one-to-one matching of all members of T_r to members of T_s

$$\{X\} \equiv T_r \tag{4}$$

It must be noted that the existence of information channels implies nothing of the direction or nature of information flow through a channel. For example, information on temperature may be transferred from unit process A to B along a piping connection, but information on concentration may be transferred from B to A as a requirement of the feed composition for unit process B, and information on the flow rate may be transferred simultaneously to A and to B in the form of a system design variable which the design engineer adjusts.

During synthesis it is important to distinguish between these two classes E_k and T_r. The information channel classes define the ultimate physical connections which may exist among unit processes: however, to attempt to make such matches during synthesis requires the simultaneous matching of all of the kinds of information carried by each of two channels. This kind of a matching ability is excessively difficult to program. It is far simpler to propose a single match between E_k members, such as a temperature-temperature match, and then check the physical realizability later using the T_r classes. The selection of variables to enter the E_k matching equivalence classes then depends on the kinds of heuristics to be employed during synthesis, and the variables which enter the T_r classes are determined solely by physical considerations.

Thus, in the extractor systems alluded to above and shown in Figure 4, each extractor exhibits four information channel classes defined by the physical means of connection. For extractors A and B, the channel sets are

$$\left.\begin{array}{l}\{F_i, x_{1i}\} \equiv T_{i1}\\ \{F_i, x_{2i}\} \equiv T_{i2}\\ \{W_i, y_{1i}\} \equiv T_{i3}\\ \{W_i, y_{2i}\} \equiv T_{i4}\end{array}\right. \quad \text{for } i = A, B \tag{5}$$

The information in Equation (5) would be listed with the sources of design information for the unit processes, and may be thought of as a different kind of design information useful only during system synthesis.

The equivalence sets corresponding to Equation (3) for the two extractors might be cast as Equation (6), if a composition-composition matching heuristic is to be employed.

$\{x_{1i}, x_{2i}, y_{1i}, y_{2i}\} \equiv E_1$	information on solute presence
$\{W_i\} \equiv E_2$	information on solvent W presence
$\{F_i\} \equiv E_3$	information on solvent F presence

where $i = A, B$.

Now if in the midst of a synthesis program the candidate match $[x_{2A}, y_{1B}]$ is proposed, Equation (6) indicates that the match is feasible since both variables convey information on solute properties. A check of the physical realizability constraints of Equation (5) indicates that information channels

$$\{F_A, x_{2A}\} \equiv T_{A2}$$
$$\{W_B, y_{1B}\} \equiv T_{B3}$$

must be employed. This implies a matching $[F_A, W_B]$ which is not feasible since the two variables are not in the same equivalence class. The initial candidate match $[x_{2A}, y_{1B}]$ is not feasible since it cannot be realized by means of allowable physical connections. If the match were physically realizable, then a test of logical structure would be made.

SYNTHESIS HEURISTICS

The state variable matchings are to be made further to accomplish some system design goal. As was stated earlier, the criterion for selecting one state variable match as most desirable cannot be known during synthesis until considerable experience has been gained. In most practical problems, the optimal system must be synthesized before one can know for sure how to synthesize the optimal system: a disheartening fact. In this section we examine the role heuristics might play in synthesis.

Few fundamental principles exist leading to the development of an absolute criterion useful in determining the proper sequence of additions of unit processes during synthesis. Therefore, one must circumvent this obstacle by ruse; several methods have been examined previously

(*1*, *2*, *5*). Here we examine the role of heuristics in the context of unit process matching, for this approach is most generally applicable.

It is convenient to examine the demonstrative and the heuristic syllogism to determine the kind of information which can be gained during synthesis. We define a structure A as an incumbent structure of unit processes which solves the design task B. The incumbent structure is either the best structure found so far, or the optimal structure obtained by other means. The design task B is some related part of the larger design problem we must solve. Now suppose that we have progressed in the synthesis to the point where part of the structure purports to solve a design task B, one for which we have an incumbent structure. Two cases may be distinguished as follows.

Demonstrative Syllogism

1. If the synthesis program employed the proper selection criteria, a structure at least as good as the incumbent structure A would be generated for design task B.

2. The synthesis program generated a structure which is *poorer than the incumbent structure* A *to* solve a design task B.

3. Therefore the synthesis program *did not* employ the proper selection criteria.

Heuristic Syllogism

1. If the synthesis program employed the proper selection criteria, a structure at least as good as the incumbent structure A would be generated for design task B.

2. The synthesis program generated a structure which is at least as good as the incumbent structure A to solve a design task B.

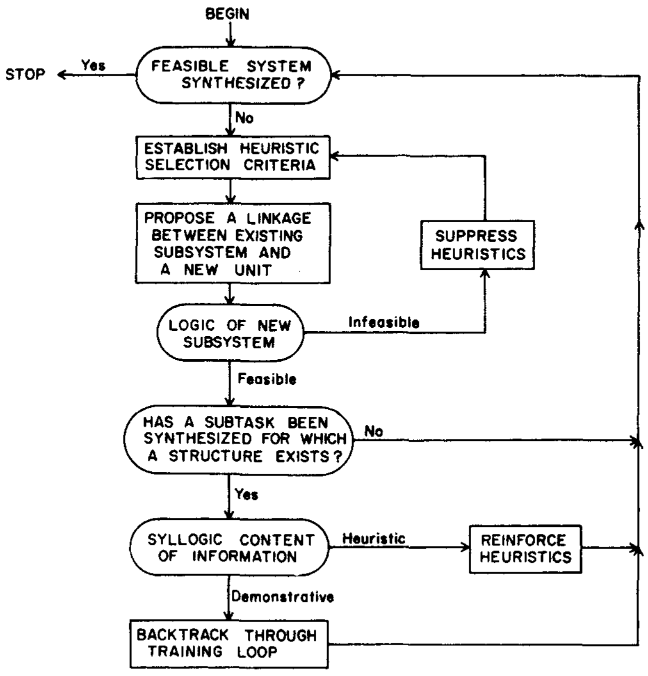
3. Therefore, it is more credible that the synthesis program did employ the proper selection criteria.

The demonstrative syllogism is a source of hard facts, whereas the heuristic syllogism proves nothing but suggests that the synthesis program is on the right path. The lack of an absolute selection criterion forces us to employ plausible criteria, which then leads to the need to gain some kind of information through the examination of the demonstrative and heuristic syllogism.

A plan of synthesis is obvious. First an attempt is made to encode previous design experience into a set of incumbent structures A for a variety of design tasks B. Then the synthesizer is supplied with a variety of plausible selection criteria $C = \{c_1, c_2, c_3, \cdots c_n\}$ which may be weighted by the set of selection weights $W = \{w_1, w_2, \cdots w_n\}$; the criterion c_i which seems to be the more useful having the largest weight w_i. Synthesis begins by the selection of a criterion from the set C, the selection being biased by the set of weights W. Unit matching proceeds until either a pattern is identified which purports to solve a subtask B or the synthesis is complete.

When a subtask B is identified, synthesis terminates temporarily as the success or failure of the synthesizer is examined at this subgoal level. This is shown in Figure 5. The examination of the success of the synthesizer in accomplishing subtasks is similar to the back track portion of Balas' method of integer programming (*11* to *13*). It is interesting that Balinski states "... clever enumeration schemes ... are the best known methods of solution [of integer programs]," and the term clever refers to methods for excluding large regions in the solution of the integer program. This is exactly what is required of an efficient process synthesizer.

The success of this kind of selection of equipment matches has been demonstrated in limited cases (*2*), and the practical extension of these ideas to full-scale process design by computer is feasible. As a step toward that end, we examine next some properties of an initial-design synthesizer.



BEGIN
STOP
Yes
FEASIBLE SYSTEM SYNTHESIZED ?
No
ESTABLISH HEURISTIC SELECTION CRITERIA
PROPOSE A LINKAGE BETWEEN EXISTING SUBSYSTEM AND A NEW UNIT
SUPPRESS HEURISTICS
LOGIC OF NEW SUBSYSTEM
Infeasible
Feasible
HAS A SUBTASK BEEN SYNTHESIZED FOR WHICH A STRUCTURE EXISTS ?
No
Yes
SYLLOGIC CONTENT OF INFORMATION
Heuristic
REINFORCE HEURISTICS
Demonstrative
BACKTRACK THROUGH TRAINING LOOP

Fig. 5. Logic of synthesizer.

IMPLEMENTATION OF AN INITIAL-DESIGN SYNTHESIZER

We now outline how the general principles of synthesis by unit matching are being implemented for the computer-aided creation of initial processing designs. Initial designs are the preliminary process flowsheets generated during the feasibility study phase of plant design which form the basis for later analysis, evaluation, and optimization. Naturally, it is desired to synthesize flowsheets with a high probability of being proved, when optimized, the best of all possible designs. However, the combinatorial problems associated with this, like other types of synthesis, preclude an exhausting search of all process arrangements. Heuristic approaches are used by design engineers, but are sufficiently complex that, despite the literature, industrial files, expert consultation, experience, and other sources of information available to the designer, the number of different flowsheets generated is limited. We are attempting to take advantage of the high-speed decision-making capabilities of the digital computer to automate this synthesis process and to explore a greater number of alternative processing concepts.

In comparison with the computer-aided synthesis of subsystems such as heat exchange and extractor networks, the generation of initial processing concepts is somewhat more complex. Final subsystems involve only the interconnections among specified kinds of processing equipment in an otherwise optimized design. Creating flowsheets involves not only interconnections, but also a determination of the tasks to be accomplished and the specification of the type of available technology to perform them in the conversion of some raw materials into the desired products. On the other hand, initial design synthesis is not tightly constrained by previous optimizations, but rather, only by the availability of raw materials, a reaction scheme to effect the necessary chemical transformations, the available technology to implement this reaction scheme, and the types of process logic previously discussed.

In this initial work, it is assumed that each step of the chemical reaction scheme occurs in a separate reactor.

Equivalence classes based on composition are then established as follows. From the list of desired products and from the reaction scheme, it is known that certain chemical species are required to be present at specific locations in the flowsheet. Similarly, again from the reaction scheme and the list of available raw materials, it is known that certain species can be found at other locations in the flowsheet. The synthesizer systematically processes "destinations" where species are required and searches for "sources" from which these species may be obtained.

Specifically, the first of two synthesis phases considers the required reactants of the first reaction in the scheme, and for each, searches over the list of available sources, which at first includes only the raw materials, and lists all component matches. Each match implies an information channel. The composition information at each end of the channel contains at least one common species. There is additional information at each end on the composition of other species, temperature, pressure, etc., which in general is not the same. Heuristics based on the severity of these differences, the availability of other sources of the reactant, and other factors are used to select only certain matches for inclusion in the preliminary design. Despite the obvious importance of concentrations and flow rates, no completely satisfactory manner has been devised to compute them for incomplete partially synthesized structures, and they were not used by the selection heuristics. This problem must be solved for future synthesizers since these types of information play an important role in process logic.

When a match is accepted, steps must be taken to eliminate differences which exist in the corresponding information channel. However, not all differences in composition information must be eliminated. There may be certain species which, although not specified in the reaction scheme, are still allowed in the reactor, as for example nitrogen, where air is used for oxidation although the scheme calls only for oxygen. The chemist is aware of the composition of the raw materials and as he creates the reaction scheme, he can state which species are allowed contaminants, and which for kinetics or other reasons are prohibited at each reactor. Although they may be eliminated in the next synthesis phase, all allowed contaminants from accepted sources are passed to the reactor effluent as new composition information different from what was specified by the reaction scheme. This dependence of effluent information on the previously synthesized structure makes the component matching phase nontrivial.

After all of the sources for a reactor have been accepted and the allowed contaminants transferred into the effluent, a slightly modified version of the component matching phase of the synthesizer considers the effluent as a new source and searches the previously considered reactor feeds for recycle destinations for each of the species. Information channels are again established for each accepted match. When this is completed, the effluent is appended to the list of available sources, and the component matching phase is applied to the requirements of the next reaction in the scheme, and if none exists, then to the desired products.

Although not all composition differences in the information channel must be eliminated, all other differences are. This is done by specifying a sequence of component separators, temperature and pressure changers, and other unit operations, during the second synthesis phase, using similar artificial intelligence techniques as are used in computerized theorem provers (*14*, *15*). The physical conditions at the source end of the information channel correspond to the hypothesis, those at the other end to the conclusion, and the set of unit operations or other pieces of available technology to the axioms and previously proved theorems.

One of the better known theorem provers, the general problem solver (*16*), serves as a model for the phase of the synthesizer. Basically it consists of matching two objects to find any differences. If none exist, then the algorithm is through. Otherwise it is attempted to reduce these differences by finding and applying some suitable operator to the first object thus forming a new object which is again matched to the second object for the detection and elimination of any remaining or new differences. Sometimes the exact conditions required for the application of the operator are not met by the first object, in which case recursively differences between the first object and the required conditions for operator are detected and attempts are made to eliminate them exactly as before.

The data in the information channels have a hierarchial structure, that is, each channel is associated with a proposed stream with its temperature, pressure, flow rate, and phases, each of which contains information on the type of phase and the composition of species, each of which contains information on the identity of the species, whether it is a contaminant, concentration, etc. Versions of the synthesizer have been written arranging these data in linear lists, using the string manipulation language SNOBOL3, and also, more appropriately, arranging the data as branching trees, using the list processing language LISP 1.5. In any case, the structure of the data allows the matching and detection of information differences to be done with straightforward non-numeric programming techniques.

If an information difference is detected, the list of available technology is searched for some appropriate unit operation to reduce the difference in light of the other existing chemical and physical conditions and within the constraints of processing logic. If one is found, it is applied to the channel producing a new one with a different set of conditions and differences. If several acceptable operations are found, the consequences of each are considered separately, leading to multiple initial designs. If no acceptable operation is found, then either another difference is attacked or else the synthesizer chooses some operation and applies it to the channel, producing a new set of conditions for which hopefully appropriate difference-reducing operations can be found.

Heuristics also guide the decisions made during this unit matching phase of the synthesis. The decisions include which of several simultaneous differences to attack first, when to create differences to lessen the severity of others, what course of action to take when no directly applicable operation exists, and how far to continue such action when again no appropriate operations can be found. Much of this early research has been devoted to the identification of meaningful selection criteria, the development of weighting schemes for these criteria, and the procedures for evaluating the initial designs generated by the synthesizer. At present, appropriate changes are made by the programmer after each synthesis. In the future, the synthesizer will repeat and evaluate its own creations, and, on the basis of the demonstrative and heuristic syllogisms, learn and adjust its own selection criteria weights and possibly develop its own meaningful correlations.

Finally, the order in which the two synthesis phases are implemented greatly influences the flowsheet produced. Either the unit matching phase can be applied to each information channel as it is accepted, or it can be applied only once after all component matches have been made. Programs using each type of ordering have been written and compared.

The first algorithm which alternates between unit matching and component matching phases has a rather simple unit matching phase as it is applied at each time to only one information channel. However, the various streams created by the specified unit operations also become available sources of chemical species along with the previous

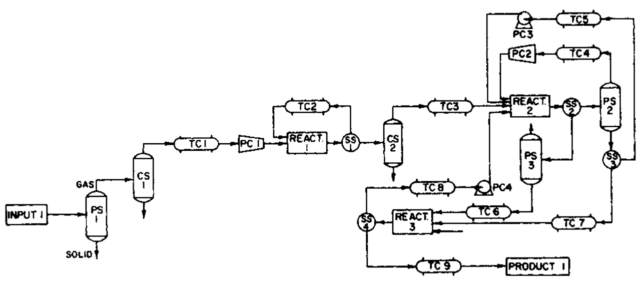


Fig. 6. Computer-generated diagram for sulfuric acid process. *PS*, phase separator; *CS*, component separator; *SS*, stream splitter; *PC*, pressure changer; *TC*, temperature changer.

reactor effluents and raw materials for subsequent component matching phases. It was found with this arrangement that as the algorithm worked its way through the reaction scheme, the synthesis became more dependent on the previously specified unit operations than on the reaction scheme.

The other alternative considered eliminates the dependence of the component matching phase on the previously synthesized structure (except for the possible effect of allowed contaminants) because all of the information channels are established before any unit operations are specified. However, this results in a much more complicated unit matching algorithm as now the information network is highly interconnected. Each species source is, in general, a source for several destinations, and each destination is the destination for several sources. It is analogous to a multihypothesis/multiconclusion theorem prover. In addition to the decisions required to eliminate information differences in a single channel, heuristic decisions must here be made to determine the order in which the channels are attacked and the degree of common processing of several channels from the same source or to the same destination. Despite these additional complexities, this seems to be a more satisfactory approach which is being further developed and investigated.

To illustrate the end result of current programs we show in Figure 6 a computer developed concept for the manufacture of sulfuric acid from the off gases of a mineral processing plant. Information was given to the computer on the chemical reactions which must be performed in three separate reactors under given conditions of temperature, pressure, and feed purity. Information was also given on the quality of the off gases, input 1, and on the required quality of the sulfuric acid to be manufactured, product 1. The computer then synthesized the supporting system of phase separators, stream splitters, component separators, pressure changers, and temperature changers. This network fills the gaps which exist among the process feed, the reactor feeds and products, and the process product. The engineer may transform this process concept into a process flowsheet by replacing the unit processes with process equipment. For example, a pressure changer symbol might be replaced by the symbol for a centrifugal pump.

CONCLUDING REMARKS

It has been demonstrated that a general system synthesis program is feasible, and can be created along the lines discussed here. The engineer need only supply a rough word statement of the design problem, and the computer can select the process equipment and the system configuration leading to a technically feasible process. The present system synthesis programs are primitive but have found use in preliminary design studies in industry for fluorination processes, synthesizing a variety of feasible processes, several of which were not known to experienced engineers.

However, an enormous amount of research is required before an ultimate synthesizer is available, for much of the ground to be covered has received no previous investigation. What has been presented is only a progress report to guide other researchers. It is clear that the schemes discussed here will be part of useful synthesis programs for they ensure feasibility.

Finally, we must remark that the development of a general computer-aided process synthesizer is not the ultimate goal of the research on process synthesis. The synthesizer is just a tool to be used to accomplish a more important goal. Extensive Monte Carlo studies are contemplated in which design goals are altered, process economic varied, and physical and chemical properties changed. The heuristics developed by the computer to generate the optimal designs under these different conditions, correspond to the rules of thumb that an experienced engineer would develop over a long and productive career of process design. We propose to use the computer-aided process synthesizers as a vehicle for developing design experience (*18*), and the ultimate goal is rules of process invention useful to the practicing engineer.

ACKNOWLEDGMENT

This work was sponsored by the National Science Foundation.

NOTATION

D	= local degrees of freedom
E_k	= matching equivalence class
f_i	= source of design information
F	= flow rate and presence of solvent
M	= number of sources of design information
N	= number of variables
T_r	= information channel equivalence class
W	= flow rate and presence of solvent
$\{X\}$	= set of design variables
x	= solute concentration
y	= solute concentration

LITERATURE CITED

1. Rudd, D. F., *AIChE J.*, **14** (1968).
2. Masso, A. H., and D. F. Rudd, *ibid.*, **15** (1969).
3. Kesler, M. G., and R. O. Parker, "Optimal Networks of Heat Exchange," Esso Mathematics and Systems, Inc. (1968).
4. King, C. J., and D. Gantz, private communication (1968).
5. Lee, K. F., A. H. Masso, and D. F. Rudd, *Ind. Eng. Chem. Fundamentals* (to be published).
6. Sargent, R. W. H., and A. W. Westerberg, *Trans. Inst. Chem. Eng.*, **42**, T190–97 (1964).
7. Lee, W., J. H. Christensen, and D. F. Rudd, *AIChE J.* (Nov. 1966).
8. Christensen, J. H., and D. F. Rudd, *ibid.*, **15** (1969).
9. Rudd, D. F., and C. C. Watson, "Strategy of Process Engineering," Wiley, New York (1968).
10. Hall, P., *J. London Math. Soc.*, **10**, 26 (1934).
11. Balas, E., *Operations Res.*, **13** (1965).
12. Geoffrion, A. M., *SIAM Rev.*, **9** (1967).
13. Balinski, M. L., *Management Sci.*, **12** (1965).
14. Newell, A., J. C. Shaw, and H. A. Simon, *Proc. Western Joint Computer Conf.*, 230 (1957).
15. Gelernter, H., in "Proceedings of the International Conference on Information Theory," UNESCO, Paris (1959).
16. Newell, A., J. C. Shaw, and H. A. Simon, in "Proceedings of a Conference on Learning Automata," Munich (1961).
17. Yagi, S., and H. Nishimura, "Chemical Processing Engineering," Chap. 9, Maruzen, Tokyo (1969).
18. Siirola, J. J., and D. F. Rudd, "Computer-Aided Synthesis of Chemical Process Designs," forthcoming publication.

Manuscript received December 2, 1968; revision received September 29, 1969; paper accepted October 2, 1969.