Gestalt Programming: A New Concept in Automatic Programming

DOUGLAS T. ROSS

Synopsis: In any human endeavor there are three major phases: conception, expression, and execution. Gestalt programming is an attempt to make these three phases as nearly identical to each other as possible with respect to computer programming. In this paper the word Gestalt is used to mean a concept of a task to be performed by a computer. In a Gestalt system of programming, the Gestalt, or idea, is expressed simply and unambiguously in a special language, rather than through the laborious assembling of machine codes, pseudocodes, subroutines, etc. Using a Gestalt system, the expression itself in effect ties together integrated units of computer behavior, which function singly or in interrelation, to achieve the desired effect. The purpose of a Gestalt system is to facilitate the transmission of general ideas as in a conversation, between a human and a computer, so that the maximum use of their respective capabilities can be made.

After presenting the abstract theory of Gestalt programming this paper discusses several Gestalt systems in use today at the Massachusetts Institute of Technology (MIT) and describes briefly the types of computer hardware which are best suited to this application.

AS computer techniques have developed over the last few years, there has been a growing trend toward more sophisticated methods for connecting the human, who states the problem, to the computer, which is to solve the problem. Great strides in automatic coding schemes and algebraic coding schemes have been made, and the feasibility and value of these techniques are now well established.

Out of this trend has come, as a natural consequence of the maturing technology, a desire to use computers for solving problems which cannot be completely specified in terms which the computer can handle. This type of problem is united with automatic problem stating, referred to in the foregoing, in the general problem of using humans and computers together to solve problems. In the one case, the goal is to state the problem so that the computer can execute the solution, and in the other case, the goal is not only to state the problem to the computer, but also to assist the computer in obtaining the solution. In both cases, the human and the computer do only those parts for which they are best suited.

If the human and computer are to work together to solve a problem, there must be some means provided for the transmission of ideas or results between the two, since the contributions of each will depend upon the actions of the other. There is no known way in which ideas can be transmitted directly, so that an intermediate stage of expressing the idea in some language is always required. A language consists of two parts; a vocabulary and a set of syntactical rules. An idea is then transmitted by transmitting the expression of the idea; i.e., a sequence of words from the vocabulary. The final stage in the transmission is recognition by the receiver.

A major problem, then, in using humans and computers together is to choose an appropriate language for the interchange of ideas. This language must bridge the gap between the fundamentally incompatible characteristics of the two parties. The human is quick-witted but slow, while the computer is slow-witted but extremely fast.

Most people connected with the computer business seem to be superbly equipped for voluble discussion on any topic. It would therefore appear at first that the language should be chosen for the convenience of the slow-witted computer. Such is definitely not the case, however, because once the computer has been given a language, it becomes a very formidable associate, firing questions and answers at a rate which very quickly becomes alarming to the human. For this reason the first rule in establishing a language is that it must be as natural and convenient as possible for the human to use, not only in the interest of reliability, but for psychotherapeutic reasons as well. Programmers with persecution complexes are already far too numerous.

Since the language is to be used for the transmission of ideas, the most natural way to obtain convenience for the human is to have the language operate entirely at the idea or concept level. In other words, the language should be designed so that general statements can be made easily by the human, with the computer itself filling in the necessary details. This concept should work in the other direction too, i.e., the statements made by the computer to the human should be pertinent digests at the idea level, and not detailed reports.

In order to use the human and computer together efficiently, a statement in the language must lead to direct and immediate recognition and reaction. This may be accomplished by designing the language so that when a statement expressing an idea is made, the receiving party, human or computer, is able to recognize immediately the elemental concepts which are to be united to give the desired idea.

A word already exists which carries all of the connotations of simultaneity and sudden bringing-together of basic units into a single entity or pattern, and that word is "Gestalt" as it is used in the Gestalt theory of psychology. Since there is no single word in the existing computer terminology which works both ways between human and computer, and includes the connotations of being at a high level of communication and implicitly including active execution, the word Gestalt will be borrowed from psychology, and will be used in this paper with very nearly the same meaning in connection with computer programming.

The decision to introduce this new word is not capricious in any way, but is made to facilitate the presentation, and to assist in the establishment of a new emphasis and point of view with respect to the general problem of the interconnections between humans and computers. The actual material discussed in this paper is, for the most part, not new, but the way in which it is discussed is new. This new approach has been found to be very fruitful and clarifying, and is the primary motivation for this paper.

Although the idea of using humans and computers together to solve problems is relatively new, enough examples have been developed by various groups throughout the United States to demonstrate that these techniques show considerable promise. After mentioning a number of applications, (some of which have not yet been tried), to motivate the discussion, this paper considers in some detail the various stages involved in designing computer systems of this type by solving a hypothetical example. The abstract structure of such systems is then outlined, using the example for illustration. Finally several systems in

DOUGLAS T. ROSS is with the Massachusetts Institute of Technology, Cambridge, Mass.

daily use at MIT are described, and some concluding remarks about the probable impact of these techniques upon computer technology are made.

Conversation Versus Communication

A suitable definition of the word Gestalt as it applies to computer programming is that it is a concept of a task. This definition is meant to imply that the Gestalt is not the task itself nor even how the task is to be performed, but merely the idea or concept of that task. For example a Gestalt might be "Integrate $f(x)$," and this idea certainly is not equivalent to the task of integration nor does it tell how the integration is to be performed. The more specific Gestalt "Integrate $f(x)$ using Simpson's rule" still does not prescribe detailed steps of applying Simpson's rule to the particular function in question.

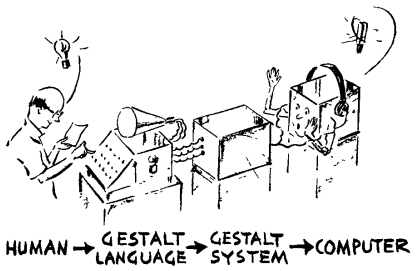

Fig. 1. Communication from human to computer

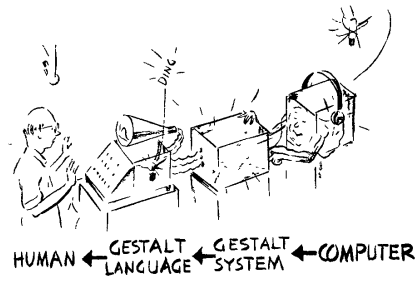

Fig. 2. Communication from computer to human

Fig. 1 shows schematically how a Gestalt is transmitted from the human to the computer. The human simply expresses his idea by pushing buttons which correspond to words or phrases in a special language, the Gestalt language. The Gestalt system then translates the expression into terms which the computer can understand, and the computer can then proceed with the execution of the task.

Fig. 2 shows the analogous situation from the computer to the human. By means of the Gestalt system the computer's idea is expressed in a special language which the human can easily understand. The human is then prepared to perform the task required by the computer.

These two illustrations show the process of communication from the human to the computer and communication from the computer to the human. If the human and computer are to work together to solve the problem, the intermediate languages and translating systems must be designed not merely for the purpose of communication, but for the convenience of fluent conversation. In other words, as Fig. 3 shows, the solution to the problem will, in general, be found only by a more or less extended conversation between the two working as a team, work being divided up so that optimum efficiency and reliability are achieved.

The remarkable flexibility of modern computers makes it possible for them to assume many guises. When more than one role is assumed by a computer in the solution of a problem, it sometimes becomes difficult to talk about the general aspects of that solution because the same mechanism has such different characteristics. This is quite definitely the case when Gestalt programming is discussed, because the computer serves in two capacities; one with respect to stating the problem and one with respect to solving the problem. In this paper the word "computer" usually means the aspect of the computer which is concerned directly with the problem to be solved. The term "Gestalt system" usually means the set of computer programs which aid in the stating of the problem by performing the necessary translation between the Gestalt language and the computer, as shown in the aforementioned illustrations. Often, however, the meaning of Gestalt system is expanded to include the Gestalt language and the physical representation of that language as well, as in the statement, "This problem can be solved by the design of an appropriate Gestalt system." The context makes clear which is intended.

Applications

Before developing the theory of Gestalt programming, several examples of problems will be presented which require or could greatly benefit from the use of human participation. It should be borne in mind, however, that although it is

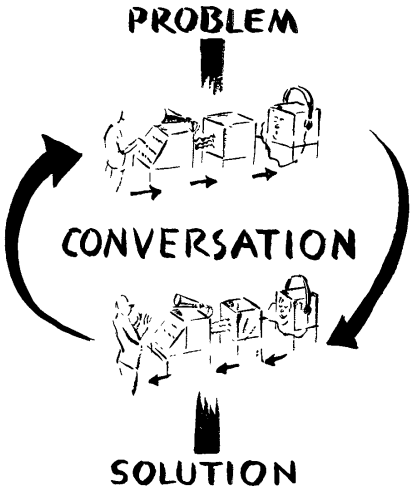

Fig. 3. Solution by conversation

problems such as these which have led to the concept of Gestalt programming, many problems which at present are not considered to require human participation could more effectively be attacked through the use of these techniques.

In almost any control application of computers, whether air-traffic control or automatic factory control, it is necessary to have at least human monitoring with the ability to make sudden changes in the computing scheme. This type of application usually places high priority on reliability and speed.

Large-scale data-reduction problems, basically automatic, often require a human choice between several alternate procedures, a choice dependent in a complex and sometimes whimsical way upon a number of intermediate results. Often partial results can be salvaged from an otherwise worthless set of data, providing appropriate techniques are chosen. By using an appropriate Gestalt system and human participation, these results can be obtained at almost normal processing speed.

Even in strictly computational work a human could greatly expedite the obtaining of solutions if the proper techniques are used. For example, in the solution of complicated partial differential equations or in linear programming problems and game theory, it seems probable that methods could be devised whereby the human could "steer" the computer directly to the solution, rather than obtaining an enormous mesh of solutions, most of which are not of real interest. This type of operation could very well be instrumental in the application of computers to aid in management decisions. A properly designed language would allow executives to converse with the

computer directly and without costly delays.

Perhaps the most intriguing application of human participation in this sense is the use of a Gestalt system in experimental programming, since such a system can be used to generate other Gestalt systems. By experimental programming is meant the programming of a large, complicated program for which the basic steps in the solution are not known. As the programming develops, the programmer must be able to do his design work at the concept level, leaving to the Gestalt system all of the details of translating his growing concepts into actual computer behavior.

Additional applications for human–computer team work can easily be found, but this brief listing should serve to show that the possible uses cover a wide range of problems. This paper does not treat programming details, for these will vary widely for each application, but does try to establish the general problems which are common to all of these applications. In addition to recognizing these problems, a general methodology or plan of attack for solving them is formulated.

Example of Gestalt System Design

The general principles of the design of a Gestalt language are best illustrated by carrying a single example through all of the various stages. Consider the case of an automatic factory whose main features are shown in Fig. 4. Three main processes are involved, followed by an assembly process. These processes are flanked by a raw materials input section and a shipping output section. Besides the primary product, it may be desired to ship directly the outputs from processes two and three. The main duty of operating this factory is to be the responsibility of a computer, but a human operator is to be in charge of setting the requirements for the various stages and overseeing the entire operation.

Present-day computers are not equipped for oral input so that some means other than a spoken language must be used to enable the operator to converse with the computer. Written languages using an intermediate medium such as punched tape or cards have long been used for communicating with computers, but a closer approach to the ease, speed, and flexibility of a spoken language can be achieved by letting each word or phrase which is to be used be represented by a single unique switch or push button. A statement is then "spoken" by pushing appropriate buttons.

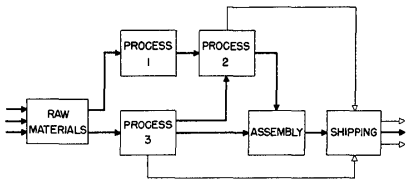

Fig. 4. Diagram of automatic factory

In the automatic factory it will at times be necessary for the operator to refer to each part of the diagram of Fig. 4. The easiest way to do this is to construct a panel with a toggle switch associated with each part as shown in Fig. 5. Now assume that the statements about the factory which the operator must make are of the form: "Increase, decrease, hold, or set the rate, amount, or storage of the input, output, or mixture of the products at the points indicated by switches which are on." The facilities for making such statements are shown in Fig. 5, where the circles connected by lines indicate push buttons with mechanical linkages so that only one button in the column can be pushed at any one time. Provisions should also be made for specifying numerical quantities so that a particular rate or amount can be specified. This facility might be in the form of keyboards or perhaps dials which can be set.

To continue the example, a statement of the foregoing form may be a demand, meaning that the computer is to jeopardize the efficient operation of other sections of the factory, if necessary, in order to comply with the statement. On the other hand, the operator may wish the computer to adjust the factory gradually to comply with the statement, but at all times maintain previous requirements; i.e., the statement is a goal toward which the computer should strive. Finally, the operator may have an estimate from a market survey, that a certain product may be in greater demand soon, so he wishes the computer to adjust the factory toward this tentative condition if it can do so with no loss of efficiency at any point. These three qualifications may be placed on the general statement by pressing one of the buttons labelled demand, goal, or estimate. In other words, the meaning of the statement expressed in the other buttons is modified by these buttons as in a language: e.g., "Run to the store, slowly."

Another whole level of meaning is made possible by considering the computer to be able to simulate the factory as well as control it. The general statement, modified as shown, may be further modified by requesting the computer either to evaluate the effect of the statement, by simulation, or to execute the statement by controlling the factory. This is assumed to be the final modification of the statement so that, the words evaluate and execute are associated with special buttons called activate buttons.

The panel should be wired so that the computer does not look at any of the buttons until one of the activate buttons has been pushed. At this time all of the items necessary to express the idea have been pushed so that when the computer looks at the buttons it is immediately confronted with a complete Gestalt. For example the Gestalt might be, "Evaluate the effect of a demand for an increase in the amount of output from process 2."

The completed panel, shown in Fig. 5, is the physical representation of the Gestalt language for this example. That it does in fact constitute a language may be seen by noting that it does have both a vocabulary and grammatical rules. The vocabulary consists of the various buttons and keyboards, and the rules are contained in the mechanical linkages of the columns of buttons and the fact that buttons modify the meanings of other buttons.

The corresponding language from the computer to the human will not be given in detail. It probably would consist of graphical displays, numerical displays, flashings of indicator lights, and audible alarms. The indicator lights probably would be located in the control panel beside the toggle switches to give them easily understood meanings. For example, the computer might reply to the foregoing Gestalt by saying that if the amount of output from process 2 is increased by demand, the rate of mixture at process 3 must be increased, which will require an increase in one of the raw materials. This Gestalt might be shown by lights at process 3 and the raw material arrow, and a graph showing the dependence of these quantities on the amount of increase at process 2. The computer would not only be able to answer questions posed by the operator but could ask policy-type decisions on operating the factory when two

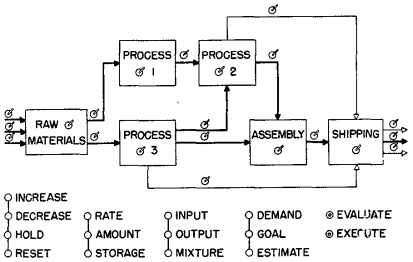

Fig. 5. Gestalt language for automatic factory

equally efficient modes of operation were discovered by the computer. Naturally the computer would keep track of all inventories and would request new supplies of raw materials whenever necessary, with sufficient lead time to maintain operation.

Note that the Gestalt language in both directions has been designed for the convenience of the human operator. In this way the human can always keep his thinking at the problem level and never has to be concerned with how a given task is to be accomplished. Because the language is more a human than a computer language, there is a routine but complicated translation to be done before the computer can actually use the language. This translation is the job of the Gestalt system proper, which is mechanized as a computer program on the same computer which controls the factory. Besides the major job of translating between the Gestalt language and the basic computer characteristics, the Gestalt system also must check statements made by the operator for consistency and completeness. In other words, the Gestalt system supplements the mechanical linkages and layout of the push buttons in ensuring that the rules of the language are obeyed. In this way any ambiguous statements made by the operator are caught before they are acted upon and, in the other direction, the computer cannot speak gibberish.

Principles of Gestalt System Design

With the experience of the example given in the foregoing, the basic principles of designing Gestalt systems can briefly be summarized. The general field to which the system is to be applied may be considered as a topic for conversation between the human and the computer. Usually this topic will be a broad generalization of the problem which initiates the interest in a Gestalt system. In the example, the topic would be control of an automatic factory. At the present state of the art, it is essential that the scope of discussion about a topic be limited to only those aspects which are of immediate interest; in the example, the scope is restricted to the particular factory.

The topic for conversation is broken down into the finest logical divisions necessary to cover the entire scope unambiguously and with a minimum of rules for combination. In the example these divisions are the words, numbers, and locations which were assigned buttons and the various basic units of the computer to human language. All of these various types of basic units will be called items, and the complete set of items forms the vocabulary of the Gestalt language. Thus a Gestalt is expressed in this language by combining items according to syntactical rules. In particular, an item may modify other items. A well-designed language will have a proper balance between items with very specific meanings, to give entry to broad areas of discussion, and items with very general meanings and thus high modifying potentialities, so that a very large and comprehensive body of Gestalts can be expressed with very little equipment.

When a Gestalt language is being designed, the items are always chosen for the convenience of the human, the goal being to have a language which is as natural to use as is possible. This statement may lead to the question why the language should not be English since that is certainly the most natural for the human. This question is clearly answered by the automatic factory example, since obviously it is more natural to select a switch associated with a box or line in a diagram than to try to describe that box by an English phrase. A similar remark applies to the computer-to-human language because a graph or diagram often conveys a complicated idea more readily than a description.

One basic principle on the choice of items cannot be overemphasized, and that is that their meanings must be unique. In other words, a button labelled "increase" must never result in a decrease as a result of modification by another item. Note that this requirement of uniqueness of meaning of individual items does not conflict with previous statements that the meaning of an item is modified by another item, since the modification is an elaboration of meaning, not a change of meaning. For a complicated problem it is often very difficult to find the minimum set of items which completely cover the scope with absolutely invariant meanings, but it is foolhardy to stop short of this goal since the only way to have a workable system is to have the human remember the pathological cases, which defeats the fundamental principle of having the human always think only at the problem level.

Implementing a Gestalt System

The considerations of the previous section have shown that the special Gestalt language is designed entirely on the basis of the problem and the convenience of the human. Once this language has been designed the human is allowed to discuss the problem only in that language so that, in effect, a part of the programming of the problem has been accomplished by programming the human. Note that this is not purely a characteristic of Gestalt languages, since every time any coding scheme or particular computer is applied to a problem, a large number of possible solutions are automatically eliminated by the characteristics of the computer or coding scheme. The aspect which is characteristic of Gestalt languages is that ideally, at least, the programming of the human is entirely beneficial.

The next step, and it is by no means a trivial one, is to program the computer so that it can converse in the Gestalt language as well, i.e., to construct the Gestalt system proper. Because the language was designed for the convenience of the human, it is usually a difficult programming task, but since everything is well defined, it can always be done.

Almost every recognized programming technique can be used to advantage in the realization of Gestalt systems. On the other hand, as might be expected, the peculiar problems which arise often lead to new techniques, or to strong desires for modification of computer logic itself. The cross-fertilization between advanced programming techniques and computer design will be more and more fruitful as these applications expand.

The final important part of the implementation of a Gestalt is the choice of a suitable medium to represent the language. The automatic factory example has already shown the advantages of diagrams and push buttons, but each problem will have its own most appropriate media. The governing criteria on the choice of representations are the rate at which the conversation is to take place and the complexity of the Gestalts when expressed as statements in the Gestalt language.

For low rates of conversation and very complex expressions, the standard input-output media, such as punched tapes or cards and high-speed printers, are probably most appropriate. The spectrum of possibilities also includes intervention switch devices for high rates of conversation. These devices, of which toggle switches and activate buttons are examples, are all characterized by the fact that a unique binary digit accessible to the computer is set to a 0 or a 1 by the setting of the device. Finally, at the ultrahigh conversation rate, there are such mechanisms as steering wheels and joysticks whose positions can be sensed by the computer. For the computer-to-human vocabulary there is a large number of audible and visual indicators, and, of course, the high-speed, very flexible oscilloscope-type output tubes.

In many applications it is desirable to give the human a variety of media for expressing the same Gestalt so that he may choose the most convenient at the time. In all cases, whatever medium is used, the principle of uniqueness should always be observed and the rules of syntax should be positively included by either mechanical or programmed interlocks.

The major steps in the design of Gestalt systems are summarized in the following. In any particular application the considerations of the various sections would undoubtedly be intermingled, but this listing can be used as a check-list summary of the basic points.

Steps in Design of Gestalt System

From the problem:
- Pick a topic for conversation.
- Restrict the scope of discussion.

Design the Gestalt language:
- Choose items which cover the scope.
- Define rules of syntax for combining items.

Design the Gestalt system:
- Determine rate of conversation.
- Choose unique representations for items.
- Establish interlocks by programs or linkages.

Present-Day Examples of Gestalt Systems

There is, of course, a growing number of computer systems which have many of the attributes which have been discussed. In general, however, most of these systems operate at medium to low rates of conversation. Three systems which operate at high rates and are in daily use on the MIT Whirlwind I computer will be briefly described to illustrate more concretely than the applications cited previously, that these techniques are not futuristic in any way, but are sound, practical investments for today.

The Comprehensive System

The MIT comprehensive system (CS), with the topic of "operating a computing facility," has elaborate utility programs, as well as automatic programming aids, under intervention-switch control. In this system Gestalts from the human to the computer may be expressed either in typewritten form, using appropriate mnemonic codes, or by pushing sequences of buttons. These Gestalts automatically call in any one of many programming systems including the CS system for the Whirlwind computer, several simulated computers used in academic courses in programming, a system for programming the Univac Scientific *1103* computer, and a system for programming numerically controlled machine tools. In addition, a large number of post-mortem and error-diagnosis programs are instantly available, as well as a growing number of data-handling and computer-operating routines. Through the use of one of these routines, the director tape program, it is possible to replace the human operator by written Gestalts to operate any number of programs in any sequence with the pushing of only one button.

The Automatic Trouble Locator

The automatic trouble locator program, with the topic of "maintaining a computing facility," is a good example of humans and computers working together. This program is primarily under intervention-switch control and automatically operates the marginal checking equipment of the Whirlwind I computer. The human sets up the general sequence of tests which are to be made by expressing his desires to the computer. The computer then proceeds with the tests, and, since the computer does not have facilities for visual input, it may ask the human to look at the wave forms at critical points, which are displayed on a monitoring scope. The operator need only tell what general type of wave form is being displayed and then the computer proceeds with the analysis. If the computer encounters a marginal piece of equipment, it types out English phrases telling which individual tubes or components require replacement, and then tells how long it took to do the job by a phrase such as "That only took 2 minutes and 33 seconds, are you sure you did it right?", which must be acknowledged by the Gestalt "Yes" before the checking can continue.

The Gestalt system approach will probably find its widest application, at least initially, in the development of similar elaborate systems for greatly improved routine operation and maintainance of other computing facilities. Experience has shown that the results are well worth the effort of devising such systems.

Data Reduction and Experimental Programming

The third Gestalt system in use at MIT is one whose topic is "automatic reduction of armament test data and experimental programming for armament control." This system is the one which has led to the analysis of this paper and is being developed for the Air Force Weapons Guidance Laboratory by the Servomechanisms Laboratory, MIT, using the Whirlwind I computer as a research tool.

This system is so designed that it includes all of the facilities of the MIT comprehensive system. Besides the comprehensive system vocabulary, this system has a large and growing vocabulary of items represented by uniquely assigned push buttons and switches. The rules of syntax which must be remembered by the human are almost entirely conjunctive in nature, the other syntactical rules being inherent in mechanical and programmed interlocks. Any syntactical error, i.e., a meaningless or contradictory combination of switches set by the human, is immediately followed by a unique and explanatory alarm. Conversely, any meaningful statement is properly understood by the computer. The computer-to-human vocabulary primarily uses output oscilloscope displays to express Gestalts, but indicator lights and audible alarms are used where appropriate. The rules of syntax are almost entirely programmed into the computer, i.e., the computer cannot speak gibberish or give misleading information.

Every item in each vocabulary requires a section of programming in the Gestalt system, some absurdly simple and some extremely elaborate. The combined sections are much too large to fit into the magnetic core memory of the computer, so that an essential part of the system is a control program which establishes the proper connections between the various program sections. This facility is so designed that individual sections can be changed easily at any time using the comprehensive system, and still mesh properly with all other sections.

This Gestalt system also has a logging program which provides a written record of the complete conversation between human and computer. This log can also be played back by the Gestalt system, the computer simulating the human actions for rerun purposes. In this way, an interrupted conversation can automatically be resumed.

In operation this system is designed so that the human can interject comments or questions into the computer's operation almost instantaneously and at any time. Some alarm conditions are automatically corrected, with suitable indication, and various types of trouble-shooting can automatically be carried out by the computer on request.

This Gestalt system is in a continual state of flux and improvement. As soon as a new feature is completed, it is usually obsolete in terms of future plans. There are an amazing number of challenges which appear with each new phase, but the results are rewarding. One of the biggest deterrents to progress is the

large amount of work involved in changing the Gestalt system program to correspond to the change of vocabulary required to include some new feature. It is hoped that a solution to this difficulty will be found by writing a program to generate translation programs which will translate from statements in arbitrary Gestalt languages into selections of computer behavior.

The goal of the experimental programming phase of this work is to allow the programmer to alter drastically his planned attack on a very large and complex problem, and try out the new solution within a matter of days, while the new approach is fresh in his mind. All too often a volatile thought pattern disappears in the months of arduous toil required to program a complex problem using ordinary techniques. It is unlikely that present and future problems being considered at the Servomechanisms Laboratory could be solved with limited manpower without the use of these techniques.

Concluding Remarks

It seems appropriate to close this paper by again acknowledging the very real debt which is owed to all of the various schools of computer programming for substantial contributions upon which this paper is based. The emergence and development of these various techniques in the past several years have established firmly the intellectual climate necessary for continued expansion in these directions. There are several groups in the United States which for some time have been developing systems for using computers which have many, if not all, of the attributes of Gestalt programming systems as defined here. The purpose of this paper has been to try to establish the outlines of the abstract structure of this type of system. It is hoped that this analysis will prove useful to all who are interested in connecting humans and computers by clarifying the problems and relationships involved.

In its full generality Gestalt programming is not just a computer technique, but is a problem-solving technique, i.e., a point is reached where it is difficult to tell which is more important, the human, the problem, or the computer. The extension of these techniques and concepts is sure to have a profound influence on the design and operation of future computers, so much so that it seems probable that the term "computer" for describing these mechanisms will become less and less appropriate. The day is fast approaching, if it is not already here, when the arithmetic capabilities of a machine will be its least valuable attributes. If the logical trend toward more and more elaborate systems of this type continues, the primary attribute of a computing machine will be its flexibility in the most general sense. Even if significant advances in the speed of computer elements can be achieved, these gains will be swiftly swallowed up if the logical design of these machines is not advanced to fit the peculiar requirements of these techniques, to obtain the same results with much fewer operations.

At the present state of the art, these future developments can only be sensed in a most intuitive way, although, for example, the growing concept of a microprogrammed computer appears to be a well-founded first step. Continued and rapid advance in these directions both in programming techniques and in computer design, can only be achieved by building on experience gained in studies using present-day facilities. It is hoped that the presentation of these ideas will encourage the participation of other groups in this fascinating line of endeavor.

A Truly Automatic Computing System

MANDALAY GREMS R. E. PORTER

LIKE many comparable groups, members of the computing facility at the Boeing Airplane Company feel that it takes too long to prepare a problem for a digital computing machine. The daily repetition of effort expended in outlining a problem for coding, the tedious task of coding the instructions, and the time consumed in checking-out or "debugging" the instructions all emphasize this fact. In this jet age, it is vital to shorten the time from the definition of a problem to its solution.

A new plan of attack for problem setup is necessary to shorten the elapsed time by shifting more of the monotonous burden of coding to the machine. It is a generally accepted belief that whenever rules for computing can be definitely established, they can be defined as a set of machine instructions. Therefore, the starting point for an automatic computing system is clarifying these rules to fit the requirements of a general problem.

A natural way to communicate a mathematical problem to a computer is by the written equation. This can be accomplished by a system allowing a digital computing machine to accept a problem directly in equation form together with a list of input data. The elapsed time for a problem is therefore shortened because this system eliminates the tedious task of coding the machine instructions. The setup time for each problem is then more dependent on the complete understanding of the mathematics and the logic rather than on the physical characteristics of one special computer.

MANDALAY GREMS and R. E. PORTER are with the Boeing Airplane Company, Seattle, Wash.

The BACAIC System

The Boeing Airplane Company Algebraic Interpretive Computing System, commonly called BACAIC, is a means of communicating directly with a machine. It is a self-contained system for solving a mathematical problem on a digital computer. This problem must be of a type which can be completely described by a set of algebraic and logical expressions. A working record of the entire system, including a file of library subprograms, is kept on magnetic tape. The library is made up of pieces originally constructed in a consistent fashion. This is important in order to establish a general pattern of rules for a system to follow.

The integrated system performs two distinct functions for each problem: