Nr. 79

Hans W. Gottinger

Complexity, Bounded Rationality and Problem-Solving

March 1979

COMPLEXITY, BOUNDED RATIONALITY AND PROBLEM-SOLVING

Hans W. Gottinger
University of Bielefeld

1. Introduction

We claim in this paper that the three notions 'complexity', 'bounded rationality' and 'problem-solving' are intrinsically interrelated.

(i) Complexity appears to be a structural property of any observable system (social, biological, mechanical), decision-making mechanism, organization, bureaucracy that imposes constraints upon the computability, selectivity, control, decision-making power, hence limits its proper functioning, limits rationality.

(ii) Structural constraints such as complexity modify the handling, manipulation, controllability of the system and its solution requires heuristics, search, step-by-step procedures leading to problem-solving in a task environment.

Let us see how we can establish the links in a meaningful way.

2. Complexity

We present here some formal definitions toward developing a more general theory of complexity for problem solving situations that may prove useful to understand the concepts to be used throughout the following section. In this particular context, such a general theory of complexity has been introducted earlier by C. Futia {1975}, more generally see Gottinger {1976}.

(1) If A is a non-empty set of symbols, then let A^* represent the set of all strings whose members are elements of A, i.e. $A^* = \{(a_1,\dots,a_n) : n \geq 1 \text{ and } a_j \in A\}$. Then we define a sequential machine as a function f: $A^* \to B$ where A is the basic intput set, B is the output set and $f(a_1,\dots,a_n) = b_n$ is the output at time n if a_j is the input at time j $(1 \leq j \leq n)$. This is the external description of a sequential machine by specifying a function $f : A^* \to B$.

The internal description involves a circuit (A,B,Z,λ,δ), where A and B are defined as above, Z is the (nonempty) set of internal states, $\lambda: Z \times A \to Z$, $\delta: Z \times A \to B$ are the next state and the output functions respectively. The step from the external to the internal description of a system is referred to as identification. It is a problem to show that given f we may find a C and a $z \in Z$ such that C 'realizes' f with $f = C_z$.

Suppose f, the external description, is a representation of a human decision-maker who in the process of thinking and acting performs like f. Then C_z represents a computer program that simulates f and has the same performance as f.

For example, let $C_z: A^* \to B$ be the system given by starting $C = (A,B,Z,\lambda,\delta)$ in state $z \in Z$, then C_z is defined inductively in a straight-forward way:

$$C_z(a_1) = \delta(z,a_1)$$

$$C_z(a_1,\dots,a_n) = C_{\lambda(z,a_1)}(a_1,\dots,a_n) \text{ for } n>2 .$$

(2) Let $f: A^* \to B$ a machine. Then f^S, <u>the semigroup of</u> f, is given by the congruence $\equiv_f$ on A^* where for $t, r, \varepsilon A^*$, $t \equiv_f r$ if and only if $f(\alpha\ t\ \beta) = f(\alpha\ r\ \beta)$ for all $\alpha, \beta\ \varepsilon A^* \cup \{1\}$. Then, if $[t]_f$ denotes the equivalence class of the equivalence relation $\equiv_f$ containing t, we have $f^S = \{[t]_f : t\ \varepsilon A^*\}$ and $[t]_f \cdot [r]_f = [tr]_{f_S}$ (where tr denotes the product in A^* and $\cdot$denotes the product in f^S). $\{1\}$ is the empty string.

(3) A semigroup S is <u>combinatorial</u> if and only if each subgroup of S is of order 1.

(4) A <u>right mapping semigroup</u> or <u>right transformation semigroup</u> is a pair (X,S), where X is a nonempty set, and S is a subsemigroup of $F_R(X)$ the semigroup of all mappings of X into X under the multiplication $(f.g)\ (x) = g(f(x))$. For each $x \varepsilon X$, $s \varepsilon S$, let $xs = (x)s$. Then the following conditions are satisfied:

(1) $x(s_1, s_2) = (xs_1)s_2$.

(2) $s_1, s_2\ \varepsilon S$ and $s_1 \neq s_2$ imply $xs_1 \neq xs_2$ for some $x\ \varepsilon X$.

(5) (Wreath Product) Let (X_j, S_j) be right mapping semigroups for $j = 1, \ldots, n$. Let $X = X_n \times \ldots \times X_1$. Let S be the semigroup of $F_R(X)$ consisting of all functions $\psi: X \to X$ satisfying the following two conditions:

(i) (triangular action) If $p_k: X \to X_k$ denotes the kth projection map, then for each $k = 1, \ldots, n$ there exists $f_k: X_k \times \ldots \times X_1 \to X_k$ such that

$$p_k \psi(t_n, \ldots, t_{k+1}, t_k, \ldots, t_1) = f_k(t_k, \ldots, t_1)$$

for all

$t_i \varepsilon X_i$, $i = 1, \ldots, n$.

(ii) (kth component action lies in S_k) We require $f_1 \varepsilon S_1$, and, for all $k = 2, \ldots, n$ and all $\alpha = (t_{k-1}, \ldots, t_1) \varepsilon X_{k-1} \times \ldots \times X_1$, the function $g\alpha\ F_R(X_k)$ given by $g\alpha\ (Y_k) = f_k(Y_k, t_{k-1}, \ldots, t_1)$ is an element of S_k.

Then $(X_n,S_n)\wr\ldots\wr(X_1,S_1) = (X,S)$ is the <u>wreath product</u> of $(X_n,S_n),\ldots,(X_1,S_1)$, and $(X_n,S_n)w\ldots w(X_1,S_1)$ ist the abstract semigroup determined by (X,S).

(6) Let (X,S) and (Y,T) be right mapping semigroups. Then we write $(X,S)|(Y,T)$, read (X,S) <u>divides</u> (Y,T), if and only if (1) there exists a subset Y' of Y and a subsemigroup T' of T such that Y' is invariant under the action of T' (i.e., $Y'T'\subseteq Y'$); and (2) there exists a map $\theta : Y' \twoheadrightarrow X$ ($\twoheadrightarrow$ means onto) and an epimorphism $\phi : T' \twoheadrightarrow SS$ such that $\theta(yt) = \theta(y)\phi(t)$ for all $y \in Y'$, $t \in T'$.

(7) (Krohn-Rhodes Decomposition) Let (X,S) be a right mapping semigroup. Then the (group complexity $\#_G(X,S) = \#_G(S)$ is defined to be the smallest non-negative integer n such that

$$S|(Y_n,C_n)w(X_n,G_n)w\ldots w(Y_1,C_1)w(X_1,G_1)w(Y_o,C_o)$$

holds with $G_1,\ldots,G_n$ being finite groups and $C_o,\ldots,C_n$ finite combinatorial semigroups (flip-flops), i.e. the minimal number of alternations of blocks of simple groups and blocks of combinatorial semigroups necessary to obtain (X,S). Hence by making full use of decomposition results on sequential machines one could redefine complexity in terms of the phase space decomposition.

Therefore, complexity finds its group-theoretic roots in the fact that the transformation semigroup can be simulated (realized) by the wreath product of all pairs of component machines whose semigroups are simple groups and those machines whose semigroups are finite combinatorial semigroups (= flip-flop machines). Intuitively speaking, <u>a combinatorial semigroup corresponds to a machine that virtually does no computation but rather switches inputs and outputs among various input-output configurations. This</u> property reminds us of information theory when selecting events which have information measure zero. These types of machines generate regular patterns to be expected, they do not yield any surprise. Therefore, their behavior does not produce information. Since everybody understands it, it cannot be complex. This result has some immediate impact on possible applications. It suggests that if we are able to detect subsystems that behave like flip-flops we could erase these subsystems without changing the structural complexity associated to other subsystems but, nevertheless, decreasing the computational complexity in terms of length of computations.

On the other hand, simple groups conform to machines that perform simple arithmetic operations (such as addition, multiplication,...). Many examples of that sort have been given by John Rhodes{1974}. A simple group constitutes the basic (irreducible) complexity element which increases the complexity of the machine by just one unit. Hence punching out groups of that kind in the decomposition lowers complexity at most by one. Now what is the significance of the Krohn-Rhodes theory? It shows us to which extent we can decompose a machine into components that are primitive, irreducible and that the solution depends on the structure of components and on the length of computation. Hence complexity does not depend only on how long a chain of components there are, but also on how complicated each component is. Therefore, complexity takes account of the total number of computations in a chain (the computational aspect) but also of the inherent complexity of the subsemigroups (submachines) hooked together via the wreath product (the structural aspect). The structural aspect can heuristically be represented by the amount of 'looping' in a computer program that computes S on X. This has been proposed by C. Futia {1975} for computing sequential decision or search rules. These are the key features of an algebraic theory of complexity.

3. Bounded Rationality

In the traditional theory of decision-making it is generally acknowledged that at least two definitions of rationality are conceivable, depending on whether the approach is abstract (normative), based on non-contradictory reasoning, or pragmatic (descriptive), based on experience. We hold that these two concepts are not necessarily mutually exclusive, if we add one important aspect to the description of rationality, e.g. computability. Rationality in the normative sense is too restrictive by granting the decision-maker unlimited computational resources which obviously fail to hold in view of complex (ill-structured) situations. On the other hand, rationality in the descriptive sense is too elusive and diffuse to be of any analytical or even predictive value since it violates unique links to consistency and coherence standards of normative postulates.

Reasons for using strategies of 'bounded rationality' could be itemized as follows:

(1) limited computational resources of the decision-maker,

(2) thresholds of complexity beyond which individuals are unable to discriminate, choose and reveal cognitive limits,

(3) many choice processes represent essentially ill-structured problems.

Let us take a moment to discuss the last point.

An ill-structured problem (ISP) fails to satisfy at least one or likely several of the listed conditions:

DSC (Definite Single Criterion): there is a definite single criterion for testing any proposed solution,

RPS (Representation in Problem Space): there is at least one problem space in which can be represented the initial problem state, the goal state, and all other states that may be reached,

TPS (Transformation in Problem Space): attainable state changes (legal moves) can be represented via transitions in a problem space, APPS (Accurate Prediction in Problem Space): if the actual problems involve acting upon the external world (environment), then changes of the state by applying operators can be predicted, controlled and directed toward the goal state with any desirable degree of accuracy, conditional on the knowledge of the environmental states,

PAC (Practical Amount of Computation): all basic processes underlying the step-by-step procedure of problem-solving search involve only a 'practicable amount of computation' so that only a practicable amount of search is needed for terminating the problem solving process.

Starting with characteristic DSC we note that ISPs usually involve a representation of multiple criteria, requiring complex trade-off statements which in fact would enhance the number of computational steps.

In the set-up of a decision problem the trade-offs may pertain to either of the following different situations:

(a) Two or more values are affected by the decision, but they are known to the decision-maker.

(b) At least one of the outcomes is subject to uncertainty, e.g. involves a lottery that has to be traded against a sure prospect. What is the certainty equivalent?

(c) The power to make a decision is dispersed over a number of individual actors or organizational units representing different values, goals.

These trade-off problems have been treated, one way or the other, in recent contributions to decision theory, see J. Marschak and R. Radner {1972}, J. Steinbrunner {1974} , R.L. Keeney and H. Raiffa {1976}. They are exclusively confined to static problems which are not sufficient to exhaust ISPs.

Conditions RPS and TPS refer to the dynamic nature of the problem space and require that the problem to be solved is well-defined and well-structured per se so that the goal structure is clearly determined a priori.

APPS is the condition that alludes to the possible stochastic nature of the problem, in which the nature of uncertainty plays a definite role, and which reflects itself in the random character of environmental states.

Finally, PAC is the crucial condition here in which the computational aspect is of major importance. We have to look for effective heuristic procedures that at least partly compensate for excessive computational routines going far beyond the information processing capabilities of human decision-makers. Hence, what is needed is to make an ISP well-structured by using procedures that apply PAC to an ISP.

As H. Simon {1973} argues, 'practicable amounts of computation' are only defined relatively to the computational power and there is a continuum of degrees of definiteness between the well-structured and ill-structured ends of the problem spectrum.

Example: Limited Rationality in Chess-Playing Programs.

A good paradigm of limited rationality is provided by designing chess-playing programs. There are various reasons for studying outcomes and strategies in games in connection with the problem of complexity and problem-solving programs.

(1) First, people are involved in complex games and attempt to find good strategies. Does there exist a computer program that matches the best human play? Furthermore, if it exists, is there anything in the structure of the program that would be beneficial to be learnt by the human problem-solver? According to Newell, Shaw and Simon {1967} : 'We do assert that complexity of behavior is essential to an intelligent performance - that the

complexity of a successful chess program will approach the complexity of the thought processes of a successful human chess player'.

(2) So far computer programs as applied to a general class of problems did rather poorly, as compared to humans, although recently there have been some fascinating improvements as evidenced by chess-playing programs such as 'chess 5.o' (David Slate, Northwestern University). In a state-of-the-art survey Newell, Shaw and Simon (1963) have pointed out that there are just too many alternatives for a computer to examine each move, so an adequate chess-playing program must contain heuristics which restrict it to the examination of reasonable moves. also to win a game you need not select the best moves, just the satisfactory ones.

(3) Studying game playing sheds a crucial light on the concept of learning in games which is not well unterstood. To teach an intelligent person the rules of chess, by itself, does not make him an expert player. One must have experience. If we could define effective game playing programs which profit from experience we have at least some clue how to practice problem-solving in real life situations that require strategic planning.

(4) How a computer program should acquire chess knowledge is an interesting and difficult point. One way, of course, is for certain records to be built into the original program. To an extent this is done. Most recent chess-playing programs contain the sequence of moves and counter-moves for standard defenses. The situation at mid-game is more difficult, since so many positions might arise .

How is it possible that good chess-players still outperform computer programs of chess which are much more powerful in computing strategies? The answer is that they evidently activate useful heuristics that more than offset their lack of computational power.

We claim that activating successful heuristics is intrinsically connected to the notion of structural complexity in dynamic algebraic systems. Chess belongs to the class of two person games with complete information and no chance moves. It is known that there exists for each board position (or more generally for each state of the game) one (or several) optimal moves. A tabulation of the optimal moves is a tremendous task. Chess has on the average over 10^{120} board positions, hence the table would have to have the same number of entries. Such a complete search for the optimal move is so enormous that it transcends the capabilities of any physical computer, in other words 'brute force computing' is not likely to be the solution.

By designing the first chess playing program Shannon {1950} proposed two principles on which an algorithm for playing chess could formulated:

(1) Scan all the possibilities (moves) and construct a search tree with branches of equal length. Hence, all the variants of the moves to be searched for are computed to the same depth. At the end of each variation (at the end of the branch) the position is evaluated by means of a numerical evaluation function. By comparing the numerical values, one can choose the best move in any given starting position, simply by the minimaxing procedure, i.e. averaging strategies by the evaluation function.

(2) Not all possibilities are scanned, some are excluded from consideration by a special rule, e.g. by a special search. Some special search methods have been proposed

and proved to be successful, see B. Raphael {1976} , ch. 3. In this method, with the same computational resources, the depth of computation can be greater.

In the first case, information of high value will be treated equally with information of low value, or collecting information is uniformly assigned equal cost to each node. A substantial part of the work will be useless,i.e. not leading to a desirable goal (checkmate). This is a modified breath first search with a numerical evaluation function and minimaxing procedure.

In fact, in option (2) it appears that highly selective search, the drastic pruning of the tree or <u>in depth search</u> is likely to be more successful to treat highly complex decision problems. For this purpose one needs a heuristic, as a rule of thumb, strategy or trick which drastically limits search for solutions, they even do not guarantee any solution at all, but a useful heuristic offers solutions which are good enough most of the time.

The pay-off in using heuristics is greatly reduced search and, therefore, involves a 'practicable amount of computation'.

In fact, summarizing the experience of various chess-playing programs, we observe that some programs have put more emphasis on computing power along tree search in the direction of option (1) whereas others have traded off computing speed against sophistication or selectivity as sources of improvement in complex programs. Selectivity is a very powerful device and speed a very weak device for improving the performance of complex programs. By comparing two major chess-playing programs, the Los Alamos and the Bernstein program, we see that they achieve roughly the same quality of performance by pursuing different routes, the computational vs. the heuristic approach: the first by using no selectivity and being very fast, the second by using a large amount of selectivity but not relying on computational speed. So, in a way, Bernstein's program introduces more sophistication to the chess program. Most of the major game-playing programs are based upon (local) look ahead and minimax techniques. As might be expected such programs have been most successful in games that have challenged the memory ability of human players, but not in games that require experience, thinking, creativity, sophistication, such as chess.

Quoting J. McCarthy {1974} : 'I think there is much to be learned from chess, because master level play will require more than just improving the present methods of searching trees. Namely, it will require the ability to identify, represent, and recognize the patterns of position and play that correspond to chess ideas...'.

4. Problem-Solving

Let us start with a definition of a problem according to Newell,Shaw and Simon {1963} : 'A problem exists whenever a problem-solver desires some outcome or state of affairs that he does not immediately know how to attain'. To generate all kinds of task-related information that pertains to 'problem-solving' is to involve heuristics that reflect practical knowledge, experience, but also logical consistency, smartness, sophistication.

A theory of problem-solving is concerned with discovering and understanding systems of heuristics. A particular, interesting method is provided by GPS, consisting of means-end analysis and planning, a subject matter we will briefly describe in Sec. 5. and Sec. 6.

Problem-solving has developed into a challenging subdiscipline of artificial intelligence, but the methods and techniques used are of sufficient general interest for dealing with decision-making situations of politicians, bureaucrats or managers. It is likely that these decision-makers could improve their decisions if they make use of a formal theory of problem-solving. The state-space approach is a very appropriate problem-solving representation, since it has a natural association to dynamic algebraic systems and complexity.

Assume the existence of a finite or countable set Z of states, and a set $\mathcal{J}$ of operators consisting of semigroups S acting upon Z. The problem-solver is seen as moving through space defined by the states, in an attempt to reach one of a desired set of goal states. A problem is solved when a sequence of semigroup operators $S = S_1, S_2, \ldots, S_n$ could be found for some decomposition of the state-space such that a nested relationship holds for some initial state z_o to generate the goal state

$$z_n = S_n(S_{n-1}, (\ldots S_2(S_1(z_o))\ldots)).$$

One could establish a one-to-one correspondence between the problem of finding S and the problem of finding a path through a graph. Let Z be defining the nodes of a graph, with arcs between nodes i and j if and only if there is an operation $S \varepsilon \mathcal{J}$ connecting z_i with z_j. The graphic representation of state-space problem solving has three advantages. It is intuitively easy to grasp, it leads to a natural extension in which we associate a cost with the application of each operation S_i. Finally, in many cases the next step to be explored can be made a function of a comparison between a goal state and a final state.

How does a theory of problem-solving relate to decision theory?

The ingredients of the conventional decision problem under uncertainty consist of

(i) a set of actions available to the decision-maker and subject to control by himself,

(ii) a set of mutually exclusive states of nature, one and only one of them can occur,

(iii) a set of consequences that obtain if the decision-maker chooses particular actions and a certain state of nature turns out to be true.

If the decision-maker is rational and satisfies certain consistency criteria on the choice of actions, he will attempt at maximizing expected utility or expected pay-off. In this problem it appears that uncertainty about which event obtains is his most severe restriction in following an optimal course of actions. On the other hand, apparently, the decision-maker need not cope with computational constraints, either there are no physical or psychological limits on his ability to handle an immense amount of data, facilitating his choice problem, or else costs of computation are virtually known, so that the decision-maker need only determine his net pay-off making allowance of the computational costs.

A problem-solving situation, requiring decision-making in contrast reveals special features that could be circumscribed by degree of difficulty, limited decision-making capabilities or resources, intrinsic complexity in finding acceptable or satisfying strategies (solutions). These characteristics require adequate methods such as complexity-bounded search, heuristics etc..

Consider the description of a genuine problem in this framework. In the 'missionaries and cannibals' problem, three missionaries and three cannibals wish to cross a river from the left bank to the right. They have available a boat which can take only two people on a trip. All can row. The problem is to get all six safely to the right bank subject to the constraint that at no time may the number of missionaries on either side of the river be exceeded by the number of cannibals on that side. To translate the puzzle into a formal problem, let a state be defined by the number of missionaries and cannibals on the left bank and the position of the boat. The starting position is (3,3,L) and the goal (terminal) state (3,3,R). The permissible moves of the boat define the operators. The problem is solved in a number of steps, whereby the minimal number, if it exists, constitutes the optimal solution. Problem-solving is certainly linked to 'survivability', given a chess position, change it into a position in which the oponent's king is checkmated. En route to this position, avoid any position in which your own king is checkmated or in which a stalemate occurs. The board positions define the states, and the piece moves the operator.

In this example the terminal state need not be fixed, but in the process problem-solving may be subsequently redefined and modified subject only to the restricition that at no point 'survivability' is endangered (endogeneous value generation).

Methods of decision analysis, as proposed by H Raiffa {1968}, for instance,are restricted in several ways:

(1) they are basically <u>off-line</u> procedure, i.e. limit choices to the 'givens' once stated,

(2) they limit complexity to the <u>determination of uncertainty</u> via probability,

(3) they address only to '<u>well-structured</u>' decision problems, where the whole set of alternatives is laid out before the decision-maker and where he knows how to achieve a particular course of action,

(4) they apply only to situations where the <u>goal structure has been fixed in advance</u> or no change of goals is anticipated in the process of taking a course of actions,

(5) they pertain to the computational part of decision-making using expected utitility as the unique performance index, but making no use whatsoever of the strength of heuristics, sophistication, creativity, innovation etc., that is the unique feature of complex decision processes.

There have been recent criticisms on the major defects of contemporous decision analysis. They can be loosely summarized as follows:

(a) Complexity is an outcome of physical constraints on information processing and therefore a matter of design.

(b) Complexity is a matter of economic constraints imposed by costs of making decisions.

(a) and (b) could be considered of being of independent significance. The first point has been emphasized here from the view of systems complexity, the second point, not less important, has been more related to costs of economic decision-making. As Th.S. Ferguson {1974} remarks, 'one of the drawbacks of decision theory in general and of the Bayesian approach in particular, is the difficulty of putting the cost of the computation into the model. There are no doubt examples in which quick and easy rules are preferable to optimal rules for a Bayesian simply because it costs less to perform the computations. An example of a physical constraint of a problem-solving mechanism, as in chess-playing programs, is given by the well-known travelling salesman problem.

A salesman wants to visit all cities $C_1, C_2, \ldots, C_n$, pass through each city exactly once (starting from and returning to his home base city C) while minimizing his total mileage. The set of objects in the travelling salesman problem is the set of all acyclic permutations of the cities, i.e. the set of feasible tours. The number of these turn our to be bounded by $(n-1)!/2$ which is an extremely large number for moderate n. By Stirling's formula $n! \cong (\frac{n}{e})^n$, hence n! increases very rapidly. For instance, for n = 10 the number is about 180,000 and for n = 11 it is nearly 2 million. Several exact mathematical solutions of this problem have been proposed, but they amount to sensible complete enumeration of the alternatives, that is, enumeration of the more likely cities. Such methods seem to work up to about n = 20 and then break down because of excessive demand upon computer time. For some promising heuristic solutions see G.L. Thompson {1967}.

5. A Case in Heuristics: General Problem Solving (GPS)

In the framework of subjective probability assessment Tversky and Kahnemann (1974) found 'that people rely on a limited number of heuristic principles which reduce the complex task of assessing probabilities and predicting values to simpler judgmental operations. In general, these heuristics are quite useful, but sometimes they lead to severe and systematic errors'.

Now GPS is one of the major problem-solving programs that may also be useful as a normative program for human problem-solving.

From a purely computer science view, GPS is a logical generalization of computer programs that have been written to solve problems in specific areas such as propositional calculus, integral calculus and plane geometry. In the view of psychology, GPS is considered as a model of information-processing characteristics of the mind, based on the idea that human heuristics could be made explicit in computer programs.

Another root of GPS lies in the work on the logic theorist (LT) program that was designed for solving sentential calculus problems in Whitehead and Russel's Principia Mathematica. It discovered proofs that are beyond the grasp of most college students, and the technique reveals sophistication rather than brute force search.

Two basic principles are intertwined in GPS: means-end analysis and planning (or recursive problem solving).

Means-end analysis (MEA) is a general purpose heuristic for making sure that an operator is only applied to the problem if there is some purpose to the application. In more economic type problem situations means-end analysis can be extended to an appropriate cost-benefit analysis where instead of 'difference operators' other suitable types of operators can be used. However, the general structure of MEA remains unchanged. Let us first address to this set-up. It involves

(a) A set of 'objects' that are relevant for the *problem definition*, $O_1, \ldots, O_n$.

(b) A set of attributes $X_1, \ldots, X_k$ attached to the objects, $k \geq n$, describing their problem-relevant location, characteristics, specifics, sometimes represented by proxies, defining the problem situation at each state of the problem-solving process.

(c) A set of operators acting on the set of attributes in terms of

$$\left.\begin{array}{ll} \text{additivity operators:} & A_1, \ldots, A_k \\ \text{difference operators:} & D_1, \ldots, D_k \\ \text{multiplicative operators:} & M_1, \ldots, M_k \end{array}\right\} \begin{array}{l} \text{operators} \\ Q_1, \ldots, Q_k \end{array}$$

such that they satisfy *algebraic operations* of finite simple groups.

(d) A terminal goal structure containing the desirable attributes $X_1^*, \ldots, X_k^*$ such that the operations applied to $X_1, \ldots, X_k$ generate the desirable set $\{X_1^*, \ldots, X_k^*\}$ after a finite number of steps.

The goal structure may be either imposed *externally* as part of the task environment or *successively generated endogeneously* in the problem-solving process.

(e) A problem solution exists after a finite number of steps.

For purposes of illustration, let us identify the various elements involved with chess-playing. In chess, a set of objects embraces all pieces on the board, pawns, bishops, rooks, king and queen etc., interacting with each other (a). Attributes of these objects apply to their location on the board, of course, in relation to the location of all other objects, whether they are attacked, defended, or attacking, in retreat and relatively safe. Hence, attributes of the same piece change frequently with each move (b).

Operators correspond generally to the rules of the game, move, specifically they are identified with the allowable moves of each piece, a bishop, a rook etc. These moves satisfy a certain algebraic structure, which is obvious since chess is a discrete game (c).

A definite goal structure is clearly imposed on chess prescribing a configuration of attributes such that the opponent's king is checkmated. However, a goal structure may be endogeneously formed, by missing subgoals such as king safety, material balance, center control etc. and therefore adapts to the terminal goal to 'avoid being checkmated' (d).

(e) Finally, a problem solution exists by reaching 'checkmate' od 'remis'! Consider, for illustration, a much more simplified problem, the so-called Monkey Problem:

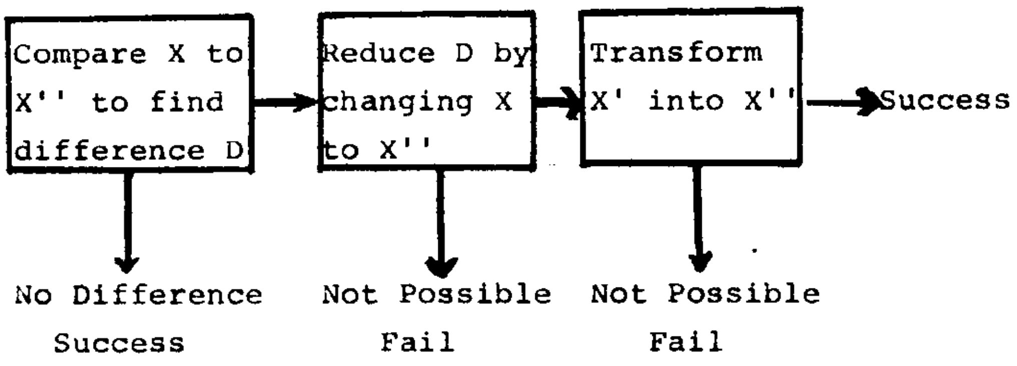
A monkey is in a cage. Suspended over the center of his cage, out of reach, is a bunch of bananas. There is a box in the corner. What should the monkey do to get the bananas?

Here the objects are three: the monkey, the box and the bunch of bananas. The situation can be described by stating: the altitude of the monkey, the location of the monkey in the cage, the location of the box, and the location and altitude of the bananas.

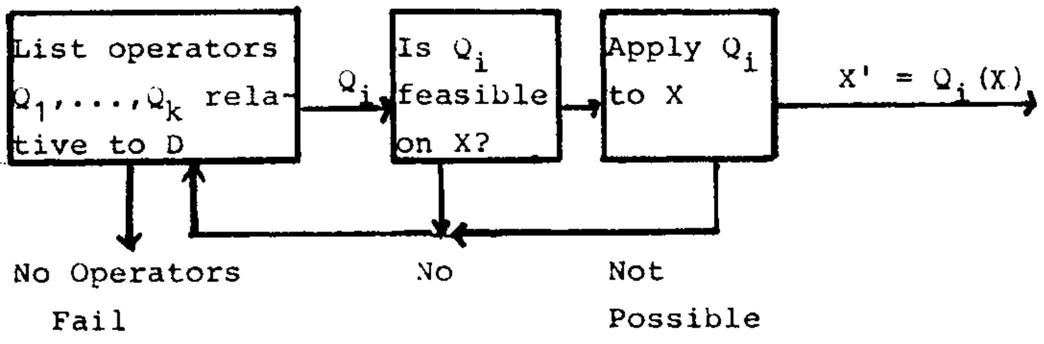
The operators are the things that the monkey can do: walk, climb, reach for bananas, and push the box. The goal structure is simply related to 'reaching the bananas'. In this example, MEA uses the difference between attributes of objects to guide the problem-solving process. The steps for an analysis of the abstract problem transfer attribute X into attribute X' such that X' is closer to X^{*}' could proceed as follows:

(1) The first step is to find that a difference D exists between the attributes X and X'' (if no differences are found you consider the subproblem solved and move ahead).

This is represented in the flow chart:

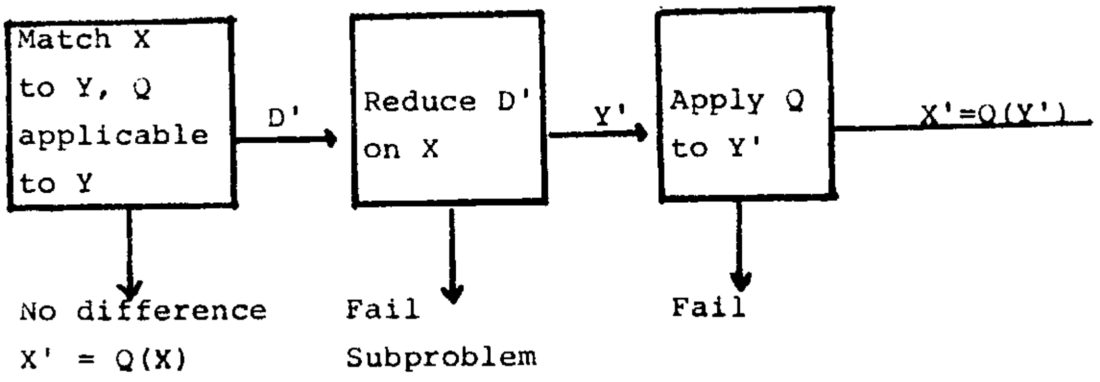


(2) Once differences have been evaluated, difference reduction is achieved by subsequent application of an operator sequence. Starting with given D and X a list of operators is to be determined which, if applicable to X, will alter the characteristic **D** on **X**. Let Q_i be the first such operator. A check is made whether the form of X is compatible with the application of Q_i. If feasibility is guaranteed the subgoal of applying the operator Q_i to X is established. If this succeeds the transformation $X' = Q_i(X)$ is made, and the result constitutes an intermediate success in the problem-solving program.



In many cases steps (1) and (2) will do, but in some cases the situation is more complicated.

(3) Then, one more step must be explained, how do we solve the subproblem of applying Q to X? This involves a test whether the assignment of attributes to a particular object is unique. Think of a car. Are there attributes or characteristics uniquely assigned to a car or do they also apply to other transportation modes? Suppose the set Y gives an exhaustive and unique representation of the object 'car', then X is matched to Y to determine if there are any differences in form. If there are not, i.e. if X and Y are identical, then Q_i is applied directly to form $X' = Q_i(X)$. Suppose, however, that difference D' between Y and X is found. Then proceed according to (1), i.e. establish a subgoal of transforming X into a special case Y' of Y. The solution to this subproblem may require further difference reduction and operator application. If Y' is finally established, Q_i is applied to construct X'.



These steps require that GPS be a recursive program, i.e. that it be capable of calling itself as a subroutine. This leads to a discussion of recursive problem-solving and change of context in solving subproblems.

6. Planning

Thus far MEA, as being a part of GPS, is only an evaluation device confined to making balances (trade-offs) in local situations, such as relating benefits to costs at each step of the problem-solving process, or at each move in a chess game. Now it is clear, that a successful problem-solver, as evidenced in chess-playing, evidently pays more attention to global guide-lines of progress and then fits local improvement into this global framework.

For example, in planning an itinary one first decides what cities one wishes to visit. Then in a 'boxes-within-boxes' procedure, air-line, taxi and limousine schedules are then fitted into the global plan. Human problem-solvers rely very much on intuition, organization of mind and fixing a planning horizon to achieve problem-solving on a global scale. It appears that global planning and local look-ahead rules are intertwined by subtle, complicated trade-off evaluations in the sense 'does the local step contribute to the global plan?', or by feed-backs from locally feasible steps toward global goals that might modify the global framework.

For example, in chess-playing and likewise strategic systems, from a global perspective it may pay-off to lose some valuable pieces, e.g., a rook or a bishop to achieve global success (check-mate), but from a local perspective it appears to be the wrong thing to do. All this could cast some doubt on the question: Can some provision for global planning be introduced into computer-aided problem-solving? In principle, yes, as Newell and Simon (1972) have proposed. The general approach is that of successive approximation. If this proves successful, we could go back to the human problem-solver and provide him with analytical guide-lines or skills that he might apply instead of purely intuitive, ad-hoc methods to achieve global success. These skills may be taught or communicated to other persons so that problem-solving becomes a professional activity. We will see how this approach fits neatly into MEA, in fact, completes the previous analysis so that a measure of strucutral complexity can be applied to enlarged problem spaces. Starting with the observation that complex problems require more time for its solution, the fixing of the time horizon is of essential importance for problem-solving. The larger the time horizon, however, the more difficult it becomes to set up a tight problem-solving framework. In other words, with the length of the time horizon one must become increasingly flexible. The global conceptual framework containing the global objectives must take care of encompassing as many options as there are available. This amounts to first solving the global program in a simplified planning space and then to adjust the solution to the more detailed, more definite problem space as one approaches closer to the situation 'what to do next'. This is the essential of recursive problem-solving. Newell's and Simon's method consists of simplifying a problem by considering only 'important' differences between states, then solving the problem in the simplified problem as a way of setting subgoals. A difficult GPS problem is (hopefully) made simpler by solving a simpler but related GPS problem.

7. Conclusions

Structural complexity is a measure of an algebraic structure ('module') that pertains to a class of heuristics and cuts drastically on the computational dimension of the problem-solving process.

We have argued that in large-scale decision problems there is necessarily a complexity-trade-off between structural and computational complexity.

The complexity theory of the algebraic theory of machines points to the fact that any non-purely-routine type operating system carries 'modules' of a simple problem-solving power as well as computational steps that can be identified with routine-type operations. This seems to explain the major strengths and weaknesses of human and computer problem-solving capabilities. The human decision-maker is comparatively strong in activating heuristic principles pertaining to structural complexity, but being restricted to depth-tree search, whereas the computer is comparatively strong in searching for many different types of solution in a breadth-type search, emphasizing computational routines by computational power and speed. The construction of useful heuristics built into computer programs, aimed at solving major tasks of a problem-solving variety, becomes a tremendous challenge to artificial intelligence, amounting to substituting computational complexity by structural complexity.

Useful heuristics with high structural complexity must include:

(i) <u>long-run 'look ahead' rules</u>, fixing the planning horizon,

(ii) <u>reasoning by analogy</u>, e.g. evaluating subtle patterns of change,

(iii) <u>depth-tree search</u>, e.g. exploiting more relevant information affecting the goal or payoff -structure in the search process,

(iv) <u>experience</u> entering problem recognition,

(v) <u>endogeneous value generation</u>, striking a delicate balance between local and strategic behavior.

A successful heuristic, revealing high structural complexity, should adapt these components repeatedly to the changing problem structure.

The tradeoff balance between structural and computational complexity can hardly be determined in advance, but in the history of chess-playing programs there are indicates that such balance exists. By comparing two differently designed chess-playing programs, the Los Alamos Program (1956) and the Bernstein Program (1958), Newll, Shaw and Simon (1963) definitely make a statement on the complexity tradeoffs in terms of overall global performance of the two programs:

'To a rough approximation, then, we have two programs that achieve the same <u>quality of performance</u> with the same total effort by two different routes: the Los Alamos program by using no selectivity and being very fast, and the Bernstein program by using a large amount of selectivity and taking much more effort per position examined in order to make the selection. ... For instance, suppose both the Los Alamos and the Bernstein programs were to explore three moves deep instead of two as they now do. Then the Los Alamos program would take about 1ooo times ($3o^2$) as long as now to make a move, whereas Bernstein's program would take about 5o times as long (7^2), the latter gaining a factor of 2o in the total computing effort required per move'.

From this we may conclude that as the depth of the moves increases it becomes correspondingly more difficult, at some point even practically impossible, to trade off computing speed and power, as represented by computational complexity, for sophisticated heuristic search procedures given by structural complexity.

REFERENCES

C. Futia (1975), 'The Complexity of Economic Decision Rules', Murray:Hill: Bell Laboratories.

H.W. Gottinger (1976), 'Complexity and Dynamics', IEEE Transactions SMC - 6, 1976, 867-873.

J. Rhodes (1974), Application of Automata Theory and Algebra, Lecture Notes: Depth. of Math., Univ. of Calif., Berkeley.

J. Marschak and R. Radner (1972), Economic Theory of Teams, Yale Univ. Press: New Haven.

R.L. Keeney and H. Raiffa (1976), Decision Analysis with Multiple Conflicting Objectives,Wiley: New York.

J. Steinbrunner (1974), A Cybernetic Theory of Decision, Princeton Univ. Press: Princeton.

H. Simon (1973), 'The Structure of Ill-Structured Problems', Artificial Intelligence 4, 1973, 181-2o1.

J. Newell, P. Shaw and H. Simon (1967), 'Chess-Playing and the Problem of Complexity', in Feigenbaum (ed.) Computers and Thought, McGraw Hill: New York.

J. Newell, P. Shaw and H. Simon (1963), 'General Report on GPS', in R.D. Luce et al. (eds.) Readings in Mathematical Psychology II, Wiley: New York.

C.E. Shannon (195o), 'A Chess Program', Scientific American.

B. Raphael (1976), Thinking Computers, W.H. Freeman: San Francisco.

J. McCarthy (1974), 'Book Review of Lighthill: Artificial Intelligence', Artificial Intelligence 5, 1974, 317-322.

H. Raiffa (1968), Decision Analysis, Addison-Wesley:New York.

Th.S. Ferguson (1974), 'Prior Distributions on Spaces of Probability Measures', Ann. Statist. 2, 1974, 615-629.

G.L. Thompson (1967), 'Some Approaches to the Solution of Large-Scale Combinatorial Problems', in M. Shubik (ed.), Essays in Honor of O. Morgenstern, Princeton Univ. Press, Princeton.

A. Tversky and D. Kahnemann (1974), 'Judgment under Uncertainty', Science 185, 1974, 1124-1131.