INTERNATIONAL ATOMIC ENERGY AGENCY
UNITED NATIONS EDUCATIONAL, SCIENTIFIC AND CULTURAL ORGANIZATION

INTERNATIONAL CENTRE FOR THEORETICAL PHYSICS
34100 TRIESTE (ITALY) - P.O.B. 586 - MIRAMARE - STRADA COSTIERA 11 - TELEPHONES: 224281/2/3/4/5/6
CABLE: CENTRATOM - TELEX 46392 ICTP

SMR/51 - 40

AUTUMN COURSE ON SYSTEMS ANALYSIS, THEORY, METHODS AND APPLICATIONS

(25 October - 23 November 1978)

COMPLEXITY AND SYSTEMS ANALYSIS IN PUBLIC PROGRAMMES

H.W. GOTTINGER
Institute of Mathematical Economics
University of Bielefeld
Bielefeld
Fed. Rep. of Germany

Complexity and Dynamics: Applications of Dynamic System Theory

HANS W. GOTTINGER

***Abstract*—An attempt is made to define a general measure of complexity in the context of algebraic system theory. The relationship between complexity and breakdown is investigated, and further ramifications and hints on applications are given.**

I. Introduction

The purpose of this paper is to exhibit the relationship between complexity and catastrophe in the framework of algebraic system theory. In general, a dynamic system $(T, Z, X, Y, \mathcal{X}, \mathcal{Y})$ is characterized by the following properties.

1) $X = \{x: T \to \mathcal{X}\}$ and $Y = \{y: T \to \mathcal{Y}\}$, where $\mathcal{X}, \mathcal{Y}$ are sets of input and output values and X, Y sets of input and output functions, respectively.
2) T is an ordered subset of the reals or the integers.
3) Definition of input segment: for every $x \in X$ let $x_{(t_1,t_2]}$ be the input segments restricted to time interval $(t_1, t_2] \in T$. For

$$x, x' \in X,\ t_1 < t_3,\ \exists x'':\ x''_{(t_1,t_3]} = x'_{(t_2,t_3]} \cdot x_{(t_1,t_2]}$$

(concatenation).
4) $\lambda: T \times T \times Z \times X \to Z$ is the transition function given by $z(t) = \lambda(t,\tau,z,x) \in Z$ with $\tau \in T$ as initial time.
5) An output function given by $\delta: T \times Z \to Y$, $y(t) = \delta(t,z(t))$.

Manuscript received October 8, 1974; revised July 21, 1976. This work was supported by Thyssen-Stiftung, Cologne (Köln).
The author is with the University of Bielefeld, Germany.

II. The Role of Algebraic System Theory

Unfortunately, there has been a split in system theory in terms of the use of different mathematical tools. Discrete-time dynamic systems use algebraic techniques (those originating in abstract group theory, homological algebra, category theory), and much of automata theory is of that sort. Continuous-time dynamic systems are firmly embedded in analysis involving properties of continuity, differentiability, convergence, etc. Much of control theory is closely related to the latter properties. On the other hand, many of the computational techniques such as dynamic programming are more closely related to the algebraization of systems. A more unified view seems to be necessary.

In fact, there are some striking advantages for the algebraic approach. *First*, algebra is a natural tool because it emphasizes the design of a system as a collection of objects—very similar to the formation of algebraic structures—by constructing new objects from given objects via algebraic operations. These operations again have an immediate intuitive appeal and significance in the design of systems such as *composition*, *partition*, *division*, and *order* (replacement). A good example is the equivalence of an empirical object, such as a machine, and an algebraic object, such as a semigroup of transformation. In fact, the equivalence is so natural that any biological, ecological, or economic system evolving in time can be viewed as a transformation semigroup (TSG) in which time is an irreversible resource. (Some authors even went so far as to consider the TSG and the associated notion of a machine as the key mathematical notions of this century.) *Second*, algebra is natural for computational work, and this is an important factor in applications. Modern computers accept digital instructions, and those in turn require an algebraization of systems. A good example of that sort is provided by equilibrium theory in mathematical economics (see [10]) which makes extensive use of measure theory and topology to derive equilibrium properties of an economic system. Equilibrium properties, however, could be shown to be equivalent to the existence of fixed points in economic systems. Unfortunately, with the tools used, such fixed points cannot be computed. To compute such fixed points one has to retreat to digitalization and discretization, e.g., devising efficient combinatorial-algebraic algorithms. For approximation of such fixed points, the innovating tools of H. Scarf [13] can be used relating to mathematical programming.

It is still open to question whether the derivation of fixed points for very large systems is more than of conceptual value since the complexity of such systems may be beyond any reasonable computation. This is, in this author's view, one of the main weaknesses of the mathematical tools currently in use in economics.

Third, algebra is intuitively appealing; it starts with very simple structures for which common sense justifications can be given. Also, it avoids requiring too much; for example, differentiability and continuity contain some idealization for which empirical justification may be hard to find.

Fourth, algebraic system theory emphasizes qualitative aspects of systems to the extent that *we are interested in properties such as survival or break-down*. This is achieved by determining complexity bounds of the system's design (design complexity). By the fact that algebraic system theory is related to computational structures, we are in the position to construct a complexity theory for dynamic systems which is also amenable to applications. As a matter of fact, system theory and algebra are so closely related that the former can be made a part of applied algebra (see [3]).

III. General Remarks on Complexity

In different connotations, complexity assumes different facets: largeness, size, multidimensionality, or hierarchy. In particular, the concepts of hierarchy and complexity are intrinsically related. In fact, hierarchy is a heuristic device to approach complexity. For example, if you ask a person to draw a complex object, such as a human face, he will almost always proceed in hierarchic fashion. First he will outline the face. Then he will add or insert features: eyes, nose, mouth, ears, hair. He will proceed this way until he reaches the limits of his anatomical knowledge. Therefore, pattern-recognizing processes proceed in a "boxes-within-boxes" form.

Structured programming and decomposition methods of linear and nonlinear programming may also serve as examples of that sort. One way of understanding complexity in natural systems (i.e., in case of humanistic or biological systems) is by construction of analogies with artificial systems. By imitating a comparable kind of complexity in artificial systems we may have some hope that by hierarchical structuring we will understand this system sufficiently well. Maybe then we have at least some key to understand the natural system. It is therefore not surprising that great significance has been attached to the concept of complexity in the sciences of the artificial (computer, system science, and artificial intelligence). One question which immediately arises is, what kind of system we are going to choose for modeling complexity of natural systems? There exist different theories of complexity which are valid for well-defined systems, and these theories are not always natural or even compatible with each other. One theory of complexity is proposed in connection with the construction of Turing and universal machines and related algorithms generating computable functions. A Turing machine has a simple mind, there are no resource and no time constraints involved, and the machine can do virtually everything, even the most complex tasks. A Turing machine has an infinite memory and an infinite tape, although this is not necessary for the working of the machine. First, we have to observe that the notion of complexity does not carry much intuitive appeal here. For example, what sense does it make to talk about a Turing machine having infinite complexity? One result which really matters in this context is Blum's theorem (the speed-up theorem). In essence it states: given two Turing machines one being faster than the other, there always exists an algorithm which computes functions faster on the slower machine than any given algorithm which computes these functions on the more powerful machine. Hence, for any complex algorithm there exists a more complex algorithm which computes functions faster. According to the speed-up theorem we could always devise an algorithm with infinite complexity. As could be expected, a Turing machine is not a good model for an organism or system which is striving to survive. In contrast we need a different model which takes into account time and re constraints. This model is a variant of the more general alg system introduced above, e.g., a sequential machine.

IV. Algebraic Machine Theory and Complexity

First we introduce some concepts to make the notion sequential machine precise. Let A be a nonempty set, le represent the set of all strings whose members are element (the alphabet), i.e.,

$$\sum A = \{(a_1,\cdots,a_n): n \geq 1 \text{ and } a_j \in A\}.$$

Then we define a machine (or sequential machine) as a fu $f: \sum A \to B$, where A is the basic input set, B the basic o set, and $f(a_1,\cdots,a_n) = b_n$ is the output at time n if a_j input at time j $(1 \leq j \leq n)$. Next we look inside the ma (or "blackbox") by defining a circuit C as quintuple $(A,B,Z$ where A and B are as defined above, Z is the nonempty internal states, $\lambda: Z \times A \to Z$ is the next state function $\delta: Z \times A \to B$ is the output function. (This essentially Mealy machine.)

We could also define a machine f in terms of a circuit. $C_z: \sum A \to B$ is a machine given by starting circuit C in $z \in Z$. This circuit then is defined inductively as follows:

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

$$C_z(a_1,\cdots,a_n) = C_{\lambda(z,a_1)}(a_2,\cdots,a_n), \qquad \text{for } n \geq 2.$$

We say C *realizes* the machine $f: \sum A \to B$ *if and only i there exists a z in Z such that* $C_z = f$. In fact, the noti realization is crucial. We have the following situation. Gi specific dynamic system, a black-box, from which we only (observe) the inputs, the outputs and the states over time. can we compose parts of that system which hocked tog "simulate" the original system, i.e., which altogether hav same performance as the original system? There may be possible realizations of a given system, but not all of them lead to the same performance. Indeed, there may be jus unique optimal realization in the sense that any other realiz performs worse. Decomposition theory in mathematical gramming is nothing else than devising methods to find optimal realization. This, of course, requires that the stru of the system is observable, hence, that we are able to deter the complexity of the system. In sequential machine theor problem of realizability of one machine by copies of machines can be approached by decomposition into compc machines, i.e., serial-parallel or cascade decomposition.[1] fact, by the fundamental results of decomposition we wi able to show that decomposition is just a counterpart o problem of complexity in sequential machines.

Examples of sequential machines are organisms which respond *in real time* to their environments just to stay alive fighting a bear). Ecological systems (bird colonies) or biolo systems (metabolism of cells) constitute systems striving survival. Also, all types of competitive economic sys challenged by adversary environments belong to this cate Furthermore, modular or neural nets being applied in ph logical or brain research constitute finite automata (seque machines), which are very similar to the design of modern di computers (see [2]). In general, extreme notions such as sur or nonsurvival (death) which are characteristic for pure forn

[1] A cascade decomposition is just a generalized form of hierarc structuring.

competitive systems are involved. In many organizational forms which do respond to a not entirely adversary environment, survival or nonsurvival are only of relative degree. Sometimes other criteria have to be substituted, and one of those is efficiency. In general, if an organization survives then it is operating efficiently. However, in the real world, an organization may survive even if it is not running efficiently, that is, if it operates in an environment that sustains artificially its operations for an indefinite period of time. Government bodies, bureaucratic entities, and administrations in general may provide such examples. Therefore, the design of systems is crucial for approaching efficiency (see [6]).

It is possible to characterize sequential machines in at least three different ways which, in view of results, prove to be equivalent. The first characterization is used by engineers and system scientists, whereas the other two classifications have a strong mathematical flavor. It is sometimes useful to interpret results in either of the given characterizations. Sometimes, when it seems appropriate, we give results in machine-theoretic or mathematical language. One can characterize finite state sequential machines as a) machines (circuits) (A,B,Z,λ,δ) with inputs, outputs and internal states, b) as abstract finite semigroups S, or c) as transformation semigroups (X,S) where the set of transformations S on X (the phase space) comprises the motions in X induced by input sequences. Transformation semigroups act as the necessary links between machines and abstract finite semigroups, and therefore we will exhibit most of the properties of the machines in terms of transformation semigroups. In order to exhaust the rich algebraic structure of sequential machines it is necessary to show the correspondence to algebraic entities and objects.

1) Let $f: \sum A \to B$ be a machine. Then f^s, the *semigroup of* f, is given by the equivalence (congruence) $\equiv_f$ on $\sum A$ where for $t, r \in \sum A$, $t \equiv_f r$, if and only if $f(\alpha t\beta) = f(\alpha r\beta)$ for all $\alpha, \beta \in \sum A \cup \{1\}$, where $\{1\}$ is the empty string. Then, if $(t)_f$ denotes the equivalence class of the equivalence relation $\equiv_f$ containing t, we have $f^s = \{(t)_f : t \in \sum A\}$ and $(t)_f \cdot (r)_f = (tr)_f$ (where tr denotes the product in $\sum A$ and $\cdot$ denotes the product in f^s).

2) A semigroup S is combinatorial if each subgroup of S is of order 1, i.e., a semigroup is combinatorial iff it contains no subsemigroups which are groups of cardinality greater than one.

Since there exists a correspondence between machine structure and semigroup structure, we observe that partitioning of a machine involves a decomposition of the respective semigroup. Partitioning of machines via covers has been emphasized in texts using machine-theoretic analogies. Instead we are considering, in what follows, decompositions of corresponding transformation semigroups.

3) A right mapping semigroup or right transformation semigroup 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 $(f \cdot g)(x) = g(f(x))$]. For each $x \in X$, $s \in S$ let $xs = (x)s$, then the following conditions are satisfied:

1) $x(s_1s_2) = (xs_1)s_2$.
2) $s_1, s_2 \in S$ and $s_1 \neq s_2$ imply $xs_1 \neq xs_2$ for some $x \in X$.

Such TSG's are sometimes called *faithful*.

4) (Wreath product). Let (X_j,S_j) be a right mapping semigroup for $j = 1,\cdots,n$. Let $X = X_n \times \cdots \times X_1$. Let S be the subsemigroup of $F_R(X)$ consisting of all functions $\psi: x \to X$ satisfying the following conditions:

1) triangular action. If $p_k: X \to X_k$ denotes the kth projection map, then for each $k = 1,\cdots,n$ there exists $f_k: X_k \times \cdots \times X_1 \to X_k$ such that $p_k\psi(t_n,\cdots,t_{k+1},t_k,\cdots,t_1) = f_k(t_k,\cdots,t_1)$ for all $t_i \in X_i$, $i = 1,\cdots,n$.
2) kth component action lies in S_k. We require $f_1 \in S_1$, and, for all $k = 2,\cdots,n$ and all $a = (t_{k-1},\cdots,t_1) \in X_{k-1} \times \cdots \times X_1$, the function $g_a \in F_R(X_k)$ given by $g_a(x_k) = f_k(x_k,t_{k-1},\cdots,t_1)$ is an element of S_k. Then $(X_n,S_n) \sim \cdots \sim (X_1,S_1) = (X,S)$ is the *wreath product* of $(X_n,S_n),\cdots,(X_1,S_1)$, and $(X_n,S_n)w\cdots w(X_1:S_1)$ is the abstract semigroup determined by (X,S).

By definition

$$(X_2,S_2) \sim (X_1,S_1) = (X_2 \times X_1, (X_2,S_2)w(X_1,S_1) \equiv S.$$

5) Let S and T be semigroups. We write $S \mid T$, i.e., S divides T, iff S is a homomorphic image "onto" (= epimorphism) of a subsemigroup T' of T, or $S \leftarrow T'$, and $\mid$ is transitive. Similarly, let (X,S) and (Y,T) be right mapping semigroups. Then we write $(X,S) \mid (Y,T)$ read (X,S) *divides* (Y,T), iff

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 $\phi: T' \twoheadrightarrow S$ such that $\theta(yt) = \theta(y)\phi(t)$ for all $y \in Y'$, $t \in T'$.

Realization of a machine by component machines hooked together corresponds to semigroup division. This definition is perfectly equivalent to a definition given in the machine theoretic literature [9]. Let us state the following.

General Problem: Let X be the entire phase space on which a right semigroup S is acting, corresponding to a machine f: $\sum A \to B$ which is realized by serial-parrallel or cascade decomposition into component machines. Then it is possible to find decompositions of (X,S) into transformation subsemigroups (X_k,S_k), $k = 1,\cdots,n$ such that one obtains the minimal solution of $(X,S) \mid (X_n,S_n) \sim \cdots \sim (X_1,S_1)$?

Example: Let an experiment E be applied to a phase space X, (the space of all trajectories or behavior lines of the system), i.e., $E(X) \equiv X$, with input set $E(A) \equiv A$, action $E(\lambda) \equiv \lambda: X \times A \to X$. $\lambda(x,a)$ is the experimentally observed "effect" of a perturbation or stimuli $a \in A$ upon the system being initially in state x. Let E be an experiment giving rise to (X,A,λ). Let $E(S) \equiv S$ be the *semigroup of the experiment* E that, by definition, equals $\{\cdot a_1 \cdots a_n : n \geq 1$ and $a_j \in A\}$ where $\cdot a_1 \cdots a_n$ denotes the mapping of X into X given by $x \to x \cdot a_1 \cdots a_n$ with $x \cdot a_1 \cdots a_n$ defined by recursive relations

$$x_0 \cdot a_1 \cdots a_n = \lambda(x_0,a_1 \cdots a_n) = b_n$$

with

$$x_0 = y_0 \qquad y_i = \lambda(b_{i-1},a_i).$$

Thus $x_0 \cdot a_1 \cdots a_n = b_n$ is the resulting state after inputs $a_1,a_2,\cdots,a_n$ have been applied to the system in that order which was initially in state x_0. Then a physical theory for E is a solution of $(X,S) \mid (X_n,S_n) \sim \cdots \sim (X_1,S_1)$. *In order to invoke some optimality property one wants to have a minimal solution.*

In classical physics most of the fundamental processes are time-reversible, and we are dealing with *groups of transformations*. On the other hand, societal and economic processes are time-irreversible[2] and therefore we are dealing with semigroups instead of groups.

The deeper algebraic idea of realizing a machine by component machines is provided by the Krohn–Rhodes prime decomposition theory (see J. Rhodes [11] or M. A. Arbib [1] for a comprehensive account with many ramifications), and even more fundamentally by the Jordan–Hölder theory in which it is proved that each finite group can be built from a fixed set of simple groups (called Jordan–Hölder factors). In machine-theoretic language the corresponding problem is to factor finite state machines into the smallest (possible) component machines to obtain a prime decomposition theorem for finite state machines. A solution of this problem brings into play complexity as a natural tool.

6) Let (X,S) be a right mapping semigroup. Then the (group) complexity $\#_G(X,S) = \#_G(S)$ is defined to be the smallest nonnegative integer n such that

$$S \mid (Y_n,C_n)w(X_n,G_n)w\cdots w(Y_1,C_1)w(X_1,G_1)w(Y_0,C_0)$$

holds with $G_1,\cdots,G_n$ being finite groups and $C_0,\cdots,C_n$ 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, i.e., optimal decomposition implies $\#_G(S) = \min \#_G\{X : X$ is a serial-parallel or cascade decomposition of $S\}$.

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, a *combinatorial semigroup corresponds to a machine that virtually does no computation but rather switches inputs and outputs among various input–output configurations.*

Examples: Take a three-element semigroup, let us say $U_3 = (\{a,b\},\{S_a,S_b,Id\})$, $a \neq b$,and every input $x \in \{a,b\}$ containing three transformations S_a,S_b,Id such that $xS_a = a$, $xS_b = b$, $xId = x$. Then U_3, in machine-theoretical language, would be called a flip-flop or identity reset machine where S_a,S_b,Id, respectively, will correspond to the instructions "drop x and replace it by a," "drop x and replace it by b," and "do nothing." The flip-flop plays a prominent role in the Krohn–Rhodes decomposition theory in that it constitutes the combinatorial part of the irreducible subsemigroups that, composed together with simple groups, realize the given semigroup machine. The link between the Jordan–Hölder theory of decomposition of finite groups and the Krohn–Rhodes theory of decomposition is given by the statement that "finite semigroup theory is finite group theory plus the 'flip-flop'." Another machine belonging to this class, generating a combinatorial semigroup is the *delay one* machine, D_1. A delay one machine over $\{a,b\}$ by definition is $D_1: \sum \{a,b\} \to \{a,b^*\}$ with $D_1(a_1) = *$ and $D_1(a_1,\cdots,a_n) = a_{n-1}$. Likewise, for all delay machines with higher order $1,2,\cdots,U_3{}^J$ and D_1 have complexity zero. This 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, etc.). Many examples of that sort have been given by Rhodes [11]. 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 and 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 not only 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 computational aspect can heuristically be represented by the amount of "looping" in a computer program that computes S on X. This has been proposed by Futia [4] for computing sequential decision or search rules. These are the key features of an algebraic theory of complexity. It opens some interesting questions about which applications can be given to an algebraic complexity theory. Such questions will be taken up in a later discussion.

V. Axioms of Complexity

Let $F_R(X)$ be the transformation semigroup of all functions on X. A function $\#_G: F_R(X) \to N$ (nonnegative integers) is called a complexity function. One aspect of such a complexity function is to exhibit various properties of groups or semigroups which play a role in the decomposition theory. Instead we are interested in more general conditions to show that group complexity is virtually equivalent to machine complexity. This gives us another clue to translate or to transform algebraic complexity into complexity of real systems. In general, there are two ways where transformation semigroups are useful: i) representing different machines or systems by semigroups for computing their complexity, and ii) representing one particular machine by different designs and each design having its own transformation semigroup.

Let M_F denote the collection of all machines (realizable by finite state circuits). Let $\theta: M_F \to N$ and $f,g \in M_F$. Let $f: \sum A \to B$ and $g: \sum A' \to B'$ be two machines. Then f is less than or equal in capability to g, e.g., $f \mid g$, iff there exists a homomorphism (trivial code) $H: \sum A \to A'$ and a function $j: B' \to B$ such that $f = gjH$.

The *parallel combination* of f and g, denoted by $f \oplus g$ is by definition $f \oplus g: \sum (A \times B) \to A' \times B'$ with $f \oplus g((a_1,b_1),\cdots,(a_n,b_n)) = (f(a_1,\cdots,a_n),g(b_1,\cdots,b_n))$. This definition can be easily generalized to an arbitrary number of machines in parallel combination.

Let $H: B \to B'$ be the trivial code depending on the number of inputs, denoted by H^Γ. Let f^σ be the extension of f if successively inputs $a_1 \cdot (a_1,a_2),\cdots,(a_1,\cdots,a_n)$ will be applied to f. Then serial combination of f and g, denoted by $f \otimes g$, is by definition $gH^\Gamma f^\sigma = k = f \otimes g$. Thus $k(a_1,\cdots,a_n) = g(h(f(a_1),h(f(a_1,a_2)),\cdots,h(f(a_1,\cdots,a_n))$.

[2] This fact has also been emphasized by N. Georgescu-Roegen [16].

Looking at the corresponding circuit construction we consider serial parallel decomposition in terms of state and output representations. Therefore, let $M_1 = (A,B,Z_1,\lambda_1,\delta_1)$ and $M_2 = (C,D,Z_2,\lambda_2,\delta_2)$ be two state dependent output circuits. Let $H: \sum B \to \sum C$ be a homomorphism. Then a new machine $(A, D, Z_1 \times Z_2, \lambda, \delta)$ is the series composition of M_1 by M_2 with connecting homomorphism H, given in terms of $M_2 \otimes H \otimes M_1$ where

$$\lambda[(z_2,z_1),a] = (\lambda_2[H(\delta_1(\lambda_1(z_1,a)),z_2),\lambda_1(z_1,a)],\delta(z_2,z_1)) = \delta_2(z_2)$$

Likewise, define the parallel composition of M_1 and M_2 by

$$M_1 \oplus M_2 = (A \times C, B \times D, Z_1 \times Z_2, \lambda_1 \oplus \lambda_2, \delta_1 \oplus \delta_1),$$

where

$$\lambda_1 \oplus \lambda_2((z_1,z_2)(a,c)) = (\lambda_1(z_1,a),\lambda_2(z_2,c)),$$

$$\delta_1 \oplus \delta_2((z_1,z_2)) = (\delta_1(z_1),\delta_2(z_2)).$$

The following result (which is easy to prove just by computation) demonstrates the interconnectedness of circuit and machine construction. Let

$$(M_1)_{z_1} = f \qquad (M_2)_{z_2} = g.$$

Then

$$(M_2 \otimes H \otimes M_1)_{(z_1,z_2)} = gHf^a$$

$$(M_1 \oplus M_2)_{(z_1,z_2)} = f \oplus g.$$

Let us rewrite the state transitions in the serial composition in a different form, making use of $z_1 \cdot a = \lambda_1(z_1,a)$, $z_1 \cdot t = \lambda_1(t,z_1)$, $t = (a_1,\cdots,a_n) \in \sum A$ when $t = 1$ and $t \in \sum C$ when $t = 2$ so that $(z_1 \cdot t_1) \cdot t_2 = z_1 \cdot t_1 t_2$. Then we have

$$(z_2,z_1) \cdot a = (z_2 \cdot H(\delta_1(z_1,a)),z_1 \cdot a) = (z_2',z_1').$$

$$\delta(z_2,z_1) = \delta_2(z_2).$$

This reformulation provides us with more intuitive properties of triangularization and kth component action lying in S_k used in the definition of the wreath product (Section IV-4). We observe that z_1' depends only on z_1 and a (and not on z_2). This is the meaning of *a acting in triangular form*. Also since $z_1 \to z_1 \cdot a$ is an element of M_1^S, the semigroup of M_1 associated to S_1, for each $a \in A$, so $z_2 \to z_2 \cdot H(\delta(z_1,a))$ is an element of M_2^S (for each fixed $z_1 \in Z_1$ and fixed $a \in A$). We then say that a has kth *component action in* M_k^S.

Let us now state the *Axioms of Complexity*.

Axiom 1: a) $f \mid g$ implies $\theta(f) \le \theta(g)$ and b) $\theta(f_1 \oplus \cdots \oplus f_n) = \max\{\theta(f_i): i = 1,\cdots,n\}$.

Axiom 2: For all machines $f_1, f_2 \in M_F$

$$\theta(f_2 \otimes f_1) \le \theta(f_2) + \theta(f_1).$$

If there is a feedback operation $\bigcirc\!\!\!\!\!\!\ominus$ from f_2 to f_1, then

$$\theta(f_2 \otimes f_1) \le \theta(f_2) + \theta(f_1) + \theta(f_2 \bigcirc\!\!\!\!\!\!\ominus f_1).$$

Axiom 3: $\theta(U_3^{f_1}) = 0$, $\theta(D_1) = 0$.

On the basis of Axioms 1-3 (plus continuity)[3] we have $\theta(f) = \#_G(f^S)$.

VI. Evolution Complexity

Up to now we were only dealing with design complexity of a particular machine or system. Under *design complexity* we understand that complexity (number) associated to the transformation process in which full use of the system potential is made. However, under design complexity, this transformation process need not result in stable configurations; for example, unstable configurations may result from a gap between computability requirements of the entire system and the computational capacities of the connected subsystems aiming at realizing the entire system. This situation occurs if there is no solution of $(X,S) \mid (X_n,S_n) \sim \cdots \sim (X_1,S_1)$, or, equivalently, if there is only a solution of $(X',S') \mid (X_n,S_n) \sim \cdots \sim (X_1,S_1)$ where (X',S') may represent some system with a lower performance than (X,S).

Under *control complexity* we understand that specific complexity (number) that results from computations which keep the entire system or at least part of it under complete control. Mathematically, represent the global input–output behavior by a semigroup $S \in F_R(X)$ associated to the system $f: \sum A \to B$. Then S is under *complete control* iff there exists an input sequence $t = (a_1,\cdots,a_n)$ that applied to S_j, $j = 1,\cdots,n$ leaves S_j in stable state $z_{f(t)}$, i.e., that is invariant with respect to changes of t,

$$S_j \cdot t = S_{f(t)},$$

for all j in the decomposition. Thus the control of S is complete iff all local states in the decomposition are invariant with respect to input sequences. In practice, we could adopt the above definition for all values of t in a sufficiently large interval. Otherwise the control of S is said to be not complete, and no global stable configurations will result.

Unstable configuration may occur if some subsystems are unable to compute (adjust) fast enough in order to adopt changes of input sequences. This is behind the intuitive meaning of control complexity.

The relation between design and control complexity is called *evolution complexity*. A system is said to be in *perfect harmony* or *perfect balance* whenever the utilization of its potentialities is complete, i.e., when design complexity and control complexity coincide.

Example: One problem is to define perfect harmony assuming the finite state machine modeling and then render in a precise form the evolution complexity relation. The cell is divided into two parts: *metabolism* and *genetic control*; both are finite state machines interacting with each other. It is essentially the Jacob-Monod model. This simple model has been applied in a non-biological context (see [8]). One way to consider the interaction within a cell is as follows. G is attempting to control M where G samples the output of M and then puts a correction input back into M (which is the usual setup in control theory). If G does it according to the design complexity and does not compute more or less than is required, then stable configurations will result, and design and control complexity will coincide. Otherwise, both will deviate, and possibly a breakdown cannot be avoided.

Fortunately, in many cases of biological or ecological systems the principle of evolution works as follows. An evolving organism transforms itself in such a manner so as to maximize the contact with the complete environment subject to reasonable understanding and control of the contacted environment. This is why we consider most natural systems to be adaptive.

An intrinsic relation between complexity and catastrophe does exist. First, a system which develops beyond its complexity bound, e.g., that is required to perform more computations than is inhibited in its design, is not going to survive but break down. This is intuitively quite acceptable but can be proved by determining upper complexity bounds along the lines as suggested in the previous section.

[3] Continuity applies to continuous decompositions on the set of machines M_F.

Next, the evolution complexity relation is crucial. The smaller this relationship the more balanced (stable) the system tends to be. We consider this situation as *qualitatively stable*. Such a system will survive if qualitative stability holds. On the other hand, the larger this relation, the more unstable the system will become up to a point where a breakdown cannot be avoided. Such a breakdown will constitute a *catastrophe*. (All the examples in this context relate to work presented by Gottinger [7], [8].) Recall that we want to model competitive systems, e.g., systems that strive for survival, in a biological context they relate to the Darwin–Wallace theory of evolution.

Example 1: In economic theory one is interested in what properties the competitive process invokes, and the notions of competitive equilibrium and Pareto optimality apply in this context. It can be proved that every competitive process acting in a so-called classical environment is a Pareto satisfactory process. In this case where the environment is "nice," i.e., where preferences and technology are convex and no indivisibilities and no externalities do exist, one can prove that the competitive process will show a stable and optimal pattern. This situation corresponds to our case where design and control complexity coincide. On the other hand, if externalities are present we are able to show that the design complexity of the system increases, whereas the control complexity remains unchanged. Therefore, in this case the competitive system acting under externalities does not produce an optimal pattern in terms of Pareto optimality. This can be demonstrated for very simple models of the tragedy-of-the-commons-type (see [8]).

In the tragedy-of-the-commons-type situation it is obvious that the existence of externalities does lead to a nonoptimal pattern of the competitive system. The same holds true for various environmental models of the competitive type as discussed by the author [8], where in general, a breakdown, catastrophe, or chaos cannot be avoided unless there are significant changes in the level of contact or control by eliminating external effects.

Example 2: Structural models of spatial distribution of cells of the Conway–Schelling type could be explored in terms of evolution complexity. These models aim at explaining optimal and nonoptimal spatial configurations of cell development or more complex humanistic structures such as urban structures. Conway recommends playing the game by starting out with some initial pattern—searching for cells which are going to survive, or die, or are birth cells—then determine all kinds of configurations possible in this game, hence show the complete life history of cells according to the rules. Conway demonstrated that, given some initial configurations, most patterns show a stable configuration in a finite number of subsequent generations. In our approach each cell could be modeled as a finite state machine interacting with other component machines governed by the rules of Conway's game. The interaction of all finite state (component) machines in simulating the entire system produces an overall pattern that according to its survival chances is judged on the basis of the evolution complexity relation. It would be quite natural to apply these ideas to more complex social systems as those pertaining to urban and city formation. In fact, this has been done by Schelling [14]. Schelling is primarily interested in some practical problems, e.g., in segregation patterns observed in cities and in the major factors that cause the patterns. Like Conway's, the game devised can be played on a checker board with checkers of different colors representing different races, groups of different linguistic or cultural ba grounds, etc. It involves definite rules to generate certain spa patterns, for example, that every black checker wants at l half of his neighbors to be "black checkers," and every wl checker wants at least a third of his neighbors to be "wl checkers." Every checker who tends to be discontented with neighborhood is allowed to move. Every move is going to cha the initial configuration, and the question is, under a given fi number of moves (stages of the game) do the configurations ad a stable pattern? A stable pattern is reached when everyb settles in a satisfactory environment, that is, when everyb does not see any reason to move elsewhere.

Summarizing, therefore, the following applications of c plexity to dynamic systems can be made:

1) competitive economic models of resource allocat models of the "tragedy-of-the-commons-type,"
2) specific models of resource depletion and environme pollution (see [8]),
3) structural models of spatiotemporal development (Conw game, Schelling's dynamic models of segregation, see [14], and [8]).

VII. Complexity in Thom's Program

Some of the remarks at the beginning stating the advanti of algebraic system theory may also be directed toward eva tion of Thom's program [15], which is in the framework differential topology and global analysis, and therefore is ro in quite different mathematical considerations. One of the n technical problems here is to relate discrete descriptions of "form" (via transformation semigroups) to differentiable mo of the form. Thom works with differentiable functions. No the topology of the form is carried or generated by a machine digitalized abstract simplicial complex, the changes in the f can be taken to be simplicial transformations. Take a mach and generate a configuration with this machine bearing tc logical properties or at least approximating such proper then a study of the change of the form via transformation s groups could replace a study of vector fields on manifold suggested by Thom. And this would lead to an algebraiza of Thom's theory and would make it amenable to large-s computation. However, it is not clear *how to generate topology of the form by algebraic tools*.

A second point is the connection between topological c plexity of a form and information as discussed by Thom. topological complexity assumes the form of a metric on the s states (endowed with a suitable topology) that comprises transformation process from one form to another. Thus t logical complexity is related to the topology of the form unless generated by a discrete model cannot be covered algebraic complexity. This gives rise to questions as to whe one should have a total complexity measure comprising algeb and topological complexity (see [4]). One restriction of algeb complexity is that it is *on-line*, e.g., computed when the syste running. We probably need a detailed theory of dynam systems. Many dynamical phenomena (e.g., evolution, gro of form) should be modeled first descriptively, e.g., wh system is doing within finite time and not what it should doing guided by a potential to optimize. The first aspect ca treated adequately by finite state machine theory, whereas latter aspect is more in the spirit of global analysis. These aspects are complementary rather than substitutable. T cover both the descriptive and the normative aspect.

7

One interesting thing appears both on the finite side and on the differentiable side, namely Lie Groups. Of course, Lie Groups occur continuously, but they also appear on the finite side since finite state machines reduce to finite semigroups which reduce to finite groups which reduce to SNAG's (simple non-Abelian groups). However, most SNAG's come from Lie algebra via the Chevalley method.

VIII. Conclusions

We explore two important aspects of complexity in connection with algebraic system theory: design and control complexity. The relationship between both, or evolution complexity, is crucial for answering questions on social dynamics, in particular on qualitative stability, catastrophe, or adaptation. The complexity itself is composed of two major features: the length of computations and the (algebraic) structure of decomposed components.

A theory of complexity is associated to "real time computation," hence, a measure of complexity only seems meaningful if we are able to compare complexity between machines. This is not possible in case of Turing machines with infinite complexity.

In the case of stochastic automata, the structure of these components may be affected by "unreliability." By that we mean that the elements may function unreliably. Since behavior of the basic elements becomes unpredictable, they may increase the complexity of the overall system. Unreliability is a basic concern in von Neumann-type automata; its impact on complexity is of eminent interest.

Another point of interest is the relationship between "error" and "complexity" in systems. R. Rosen [12] regards "error" as a basic property of complex systems, resulting from "uncontrollable deviations between the behavior of an abstract functional model, and the behavior of a real system."

In our approach "error" results from the gap between the computational requirements (imposed by the design complexity) and the computational capability (reflected in the control complexity). As could be expected, cumulation of errors (in terms of increasing the evolution complexity relation) will definitely result in a breakdown of the system.

Acknowledgment

The author would like to thank Ms. Goergel, University of Bielefeld, for typing the manuscript.

References

[1] M. A. Arbib (Ed.), *Algebraic Theory of Machines, Languages, and Semigroups*. New York: Academic Press, 1968.
[2] ——, *Brains, Machines and Mathematics*. New York: McGraw-Hill, 1974.
[3] G. Birkhoff and H. Bartee, *Modern Applied Algebra*. New York: Macmillan, 1972.
[4] C. Futia, "The complexity of economic decision rules I and II," Murray Hill: Bell Laboratories, Jan. 1975.
[5] M. Gardner, "The fantastic combinations of John Conway's new solitaire game 'Life'," *Scientific American*, vol. 222/223, pp. 120–123, 1970.
[6] H. W. Gottinger, "Computable organizations: Representation by sequential machine theory," *Annals of Systems Research*, vol. 3, pp. 81–108, 1973.
[7] ——, "Complexity and information technology in dynamic systems," *Research Memorandum, Volkswagen Foundation, Kybernetes J.*, vol. 4, pp. 129–141.
[8] ——, "Notes on dynamic systems and social processes," in *General Systems*, vol. 20, pp. 121–134, 1975.
[9] J. Hartmanis and R. E. Stearns, *Algebraic Structure Theory of Sequential Machines*. Englewood Cliffs, NJ: Prentice Hall, 1966.
[10] W. Hildenbrand, *Core and Equilibria of a Large Economy*. Princeton: Princeton University Press, 1974.
[11] J. Rhodes, *Applications of Automata Theory and Algebra*. University of California, Berkeley, 1973, Lecture Notes.
[12] R. Rosen, "Complexity and error in social dynamics," Center for Theoretical Biology, S.U.N.Y. Buffalo, unpublished manuscript, 1975.
[13] H. Scarf, *The Computation of Economic Equilibria*. New Haven and London: Yale University Press, 1973.
[14] Th. C. Schelling, "Dynamic models of segregation," *Journal of Math. Sociology*, vol. 1, pp. 143–186, 1971.
[15] R. Thom, *Stabilité Structurelle et Morphogénèse*. Reading, Mass.: W. A. Benjamin, 1972.
[16] N. Georgescu-Roegen, *The Entropy Law and the Economic Process*. Cambridge: Harvard University Press, 1971.

8

Journal of Cybernetics, 7:000-000, 1976

TOWARD AN ALGEBRAIC THEORY OF COMPLEXITY IN DYNAMIC SYSTEMS

HANS W. GOTTINGER

University of Bielefeld, F.R. Germany

Consider a large economy with many agents, each agent may receive messages from other agents upon which he immediately acts producing new messages for other agents, etc. A legitimate question to ask about such a process is what is its complexity (i.e., the complexity of the overall system)?

INTRODUCTION

A simple model suitable for determining the complexity of the overall system [which] would involve each agent as a two-state automaton and the economic system as a network of on-off automata. That is, each agent either receives the message as it is delivered to him or he does not receive or does not accept this message. The model, in fact, is taken from Arbib (1964), as a basic model of the brain, where a nerve cell is considered to be a two-state automaton and the brain is a network of "on-off" neurons. This framework, of course, can be extended to a stochastic automaton, although the argument becomes harder to follow.

Elsewhere (Gottinger, 1975a,b) I have argued that a finite state automaton can be used for modeling a competitive economic system, that is, a system that is bound to survive or break down. Notice that in this system only "global" qualitative properties of survival or breakdown are considered and they may substitute for much more restricted notions of classical economics such as equilibrium or disequilibrium. [Incidentally, in physics we observe a similar shift, namely from equilibrium thermodynamics to nonequilibrium thermodynamics, see Nicolis and Prigogine (1973).

Some models of organizations based on sequential machine theory have already been suggested in unpublished papers by Marschak and McGuire (1973), and Gottinger (1973). This paper is in the spirit of these investigations, but here we emphasize the determination of complexity in such systems.

We proceed first by giving a general nontechnical description of the system we have in mind, then we turn to the description of an economic system and show how it works within this framework. We in fact analyze a dynamic finite state system which interacts with its environment under constant evolutionary stress imposed by real time, space, and resource constraints.

Its general characteristics are described by

1. *A set of inputs,* e.g., those changing parameters of the environment which will affect the system behavior in a predictable way.
2. *A set of outputs,* i.e., those parameters that act upon the environment leaving observable changes in the relationship between the system and the environment.
3. *A set of states,* i.e., those internal parameters that determine the relationship between inputs and outputs and which may reveal all necessary information embodied in the past.
4. *The state transition function,* which determines the dynamics of how the state will change when the system is fed by various inputs.
5. *The output function,* which determines what output the system will yield with a given input when in a given state.

How can we put an economic system in this framework?

SEQUENTIAL MACHINE MODELING OF AN ECONOMIC SYSTEM

In this regard we reformulate a well-known description of an economic system as originally formulated by Hurwicz (1959, chap. 3), and translate it into a proper system-theoretic language. What are the basic ingredients? First, we encounter the environment, which consists of the set of the economy's resources, a set of technological possibilities, and the set of consumers' and investors' preferences. In system-theoretic terms the environment could be represented by an appropriately dimensioned vector of parameters where each subvector contains parameters of resource

availability, technological conditions, and preference patterns, respectively.

Second, in a market-type economy, changes in these parameters would produce messages via prices. In this framework the messages may result as outputs of the decision-making units. They in turn form additional inputs for other units.

Third, we do not distinguish between *physical* and corresponding *informational* activities, since there would be no natural distinction in the transformation process. For example, changes in the physical parameters as inputs will in general not only produce a change in physical output but also a change in the message generated by it (e.g., a new price quotation).

The initial state of the system is given by a distribution of the environmental parameters. In the almost classical Hurwicz paper an allocation process consists of a message set and a response function mapping messages into action for a given environment. Here we assume the more general case that the environment is essentially "dynamic," thus is subject to changes in the process of transforming inputs to outputs. Also, the message set, as defined by Hurwicz, is rather limited. We wish to include changes of environmental parameters (acting externally), as well as those of informational activities (output changes) of other units (acting internally). Except for these generalizations an allocation process would be just a transition function or rather a set of transition functions.

To be more specific about what is involved here let us introduce some formal definitions.

1. If A is a nonempty set of symbols, then let A^* represent the set of all strings whose members are elements of A, i.e.,

$$A^* = \{(a_1, \ldots, a_n) : n \geq 1 \text{ and } a_j \in A\}$$

Then we define a sequential machine as a function $f : A^* \to B$ where A is the basic input set, B is the output set and $f(a_1, \ldots, a_n) = b_n$ is the output at time n if a_j is the input at time j ($1 \leq j \leq n$). In fact, this is the external description of a sequential machine by specifying a function $f : A^* \to B$ that maps the set A^* of sequences of inputs to single outputs in B. The external description is useful to trace the *global* behavior of an activated sequential system, but it fails to tell us what might be going on inside the system.

2. Therefore, let's look inside the machine (or "black box") by defining a circuit as a quintuple (A,B,Z,λ,δ), where A and B are defined as

above, Z is the (nonempty) set of internal states, $\lambda : Z \times A \to Z$ is the state transition function, and $\delta : Z \times A \to B$ is the output function. The step that we take to go from the external to the internal description of a system is commonly referred to as the *identification problem*. It is a problem to show that given f we may find a C and a $z \epsilon Z$ such that C realizes ("simulates") f with $f = C_z$. The identification problem comes up later in the context of decomposing a transformation semigroup, representing the machine, into parts and hooking them together via the wreath product to realize the given machine. There is no way to identify the system unless we are able to determine its complexity, so complexity is crucial for identification.[1]

3. Let $C_z : A^* \to B$ be the machine given by starting the circuit $C = (A,B,Z,\lambda,\delta)$ in state $z \epsilon Z$, then C_z is defined inductively in a straightforward way:

$$\begin{cases} C_z(a_1) = \delta(z,a_1) \\ C_z(a_1, \ldots, a_n) = C_{\lambda(z,a_1)}(a_2, \ldots, a_n) \text{ for } n \geq 2 \end{cases}$$

A good organizational model of an economy, put in terms of a sequential machine, would be a serial-parallel decomposition of a large finite state machine (circuit) (A,B,Z,λ,δ) where each decision-making unit can be identified as a component machine connected in some serial-parallel fashion with other component machines (see Figure 1). I call such a decomposition a cross connection (web). It resembles the construction of neural networks where every neuron as part of the network is considered to be a finite-state automaton or a McCulloch-Pitts cell (Minsky, 1967, chap. 3). Here the construction is sufficiently general even to cover hierarchically structured economies such as multilevel planned economies (Moiseev and Schmidt, 1973).

The output of any given subsystem affects the other subsystems with a unit time delay, so let us assume for simplicity that all subsystems start working simultaneously. The stimulation by inputs (or by change of inputs)

[1]This identification problem has an interesting analogy to the ideas of realizing the function of a brain by a machine. The machine then serves as artifact which is better known than the functioning of the brain (acting like a "black box"). The following question arises in this context: (D. M. MacKay, p. 263). "How far is it possible to envisage an artificial mechanism that would not only imitate human behavior, but work internally on the same principle as the brain?"

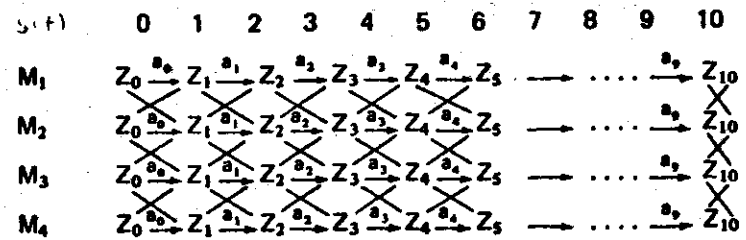
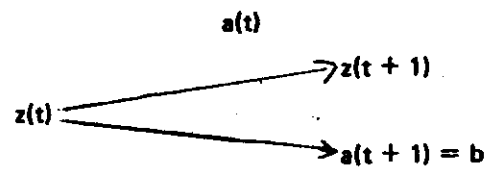


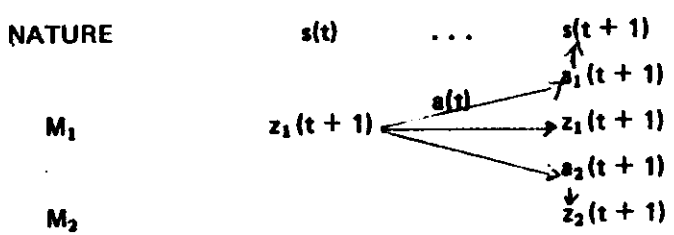
FIGURE 1. Environmental states s(t), A

from the environment and its resulting generation of action can be simply described in this network (cross connections).

An input (stimulus) a(t) to Machine 1 (M_1) at time t changes its state by $\lambda_1[z(t), a(t)] = z(t+1)$ and provides as output $\delta_1[z(t), a(t)] = a(t+1) = b$. This in turn serves to stimulate M_2 at time $t+1$ and so on down the line. It also works back the same way. By adopting the convention that the diagram



represents the action of input a(t) upon state z(t) at time t, it is possible to build a matrix representation of the entire system. This matrix representation is illustrated in the network of cross-connection. In addition to the above scheme we may wish to include nature's states explicitly, and therefore split the output in essentially two parts: $a_1(t+1) = b_1$ acting on nature, and $a_2(t+1) = b_2$ acting on some other subsystem (as illustrated below).



SOME FUNDAMENTAL ISSUES OF COMPLEXITY

Before approaching complexity in the context of specific models we should pause for a moment and first indicate why complexity is so crucial for the analysis of systems we encounter in real life. Since if we know the complexity we could possibly devise better (mathematical) methods of modeling and controlling real systems. So far our tools seem to be rather limited. Mathematics so far has been quite successful in dealing with either small or infinitely large systems. At one extreme there are a small number of parts to a system of fairly simple connections. Here mathematics is very helpful in exhausting all the possibilities by getting, say, a complete solution of the set of equations. At the other extreme one works with some fictitious assumptions when the number of particles gets to be of the order of magnitude of some very high power of ten. If they are sufficiently homogeneous it is assumed that there are infinitely many of them. Here certain properties of continuity, some sort of smoothness may be invoked so that the whole power of classical analysis comes into the play. (Theorems on the core of an economy in mathematical economics are based on these assumptions but the fiction of an infinitely large number of economic agents having an infinitely small impact on economic interactions seems to be a rather unrealistic viewpoint.) Unfortunately, most of the large systems relating to societal problems fall into a range that is intermediate to these. The systems are much too "complex" to get explicit solutions for them, and yet the number of parts is not large enough nor are the parts homogeneous enough to be able to pass to the limit.

What could be a general characterization of such complex systems? The description of Simon (1969, chap. 4) that we understand a complex system as "one made up of a large number of parts that interact in non-simple way" can at most be considered as a plausible description in a colloquial sense but falls short of contributing to the question of what complexity is.

In this regard it seems useful to ask a couple of questions about the behavior of the system. One major question, for example, is about identification or as Bellman (1965) has put it: "Given some information concerning the structure of a system, and some observations of inputs, outputs and internal behavior over time, deduce all the missing information concerning structure, inputs and outputs."

The second question one may ask is, does it produce errors in the sense that its behavior is hard to predict?[2]

As has been pointed out by Rosen (1975a), the concept of error is crucial and arises naturally in social and biological systems. In fact, error and complexity seem to be positively correlated, it is frequently the cause of sudden changes, mutations, perturbations, and break-downs occurring within the system.

A third question to ask, related to the second one, is what is the level of control we can exert upon the system? Controllability seems to be a desirable property of systems independent of complexity. But this, in fact, is not the case. It is useful to mention here that controllability is not always inversely related to complexity as one would intuitively expect. In the engineering field we observe systems operating that are highly complex (according to some standard) but they are also highly controllable. However, in the special economic and biological field the more interesting aspect is that these systems are only partially controllable if at all. This observation has given rise to the distinction between design and control complexity (Gottinger 1975a,b, 1976).

The level of controllability is really what matters in dealing with complex systems, and its relationship to complexity is not sufficiently well understood. A case in point is provided by our experience with the behavior of economic systems.

> The fact that our economic system has responded to attempts to control it in surprising and unexpected ways has often been offered as an illustration that complex systems are counterintuitive. What does such a statement mean? It means simply that those intuitions regarding system behavior which we presently possess were formed on systems which are in some way simple, and that properties of these simple systems do not generalize in any obvious way. (Rosen, 1975a)

[2]Under "error" we subsume all those functional activities of a system that are erroneous or mistaken if they will not carry the system from its initial state to its desired final state or goal. Here, in other words, an "error" is intrinsic to the system (design related). Another kind of error may be seen in unsatisfactory computational performances of subsystems hooked together to realize the original system. This kind of error will be discussed (Gottinger, 1976) in connection with the derivation of computational requirements and computational capacities of subsystems. It forms the crux in distinguishing design and control complexity.

One of the problems correlated with the level of controllability in complex systems is that of adaptability. In economics it has been widely presupposed as a kind of fact. All classical models of the competitive economy have assumed this built-in-process of adaptability. Of course, one may argue that these models have been derived from idealistic or simplistic assumptions, somewhat abstracted from the real world, however, they have often been chosen for policy recommendations. These recommendations are based naïvely on the assumption that the model is only an image of a real system and therefore that the basic properties of the system are simply preserved by the model which has more than once been contradicted in particular cases. Again this suggests that a real system basically differs in a qualitative way from any simplifying model attempting to describe it.

Rosen (1975b) has looked at it in the following way:

> One of the main reasons for asking questions about the adaptions of social systems and their institutions is a dissatisfaction with their current performances, and a sense that such poor performance (as measured by some standard) arises from a state of nonadaption or maladaptation! (p. 31)

He goes on to say that biology deals with just such questions, namely with the evolutionary, physiological and behavioral transitions from unadapted or maladapted states and adapted ones.

Of course, it would not be at all satisfactory to replace "complexity" by some other more restrictive notions such as largeness, size, and multidimensionality. Part of the problem is that complexity is not merely a quantitative property but foremost a qualitative one. Rather frequently complexity has been approached by putting models in simplistic terms, that is, to engage in the attempt to approximate the behavior of real systems by structuring the model in interconnected parts that seemed natural in a particular context. For example, hierarchical structuring has been used as a natural device to simulate the behavior of such systems (Simon, 1969). However, there may be many forms of hierarchical structuring and what is necessary here is to find a decomposition of the system that takes care of its intrinsic complexity. There are certain observations that may help to detect when systems act in a way that is not sufficiently well understood, even being counterintuitive, and where a lack of predictability regarding the system's behavior arises. Of course, a system is never universally complex, a system may be complex in some respect (in some critical situation) but not

in others, or it is complex if it is used for certain goals but not for others (Rosen, 1975a). Different concepts of complexity from various viewpoints have been discussed by Bremermann (1974a,b).

The problem of complexity is a scientific and intellectual challenge by its own, but one which has been neglected in the past. Rather, there has been the customary attitude that complex processes could be understood in terms of simple underlying universal laws derived from comparatively simple model constructions of real systems. True, complexity has been approached in model building in various ways, e.g., by increasing the number of equations and/or variables and/or constraints as in mathematical programming or large-scale system theory, or by randomization in probabilistic modeling of large systems. However, the point is that complexity has always been handled in terms of increasing the dimension of the model, in a strict quantitative fashion, but hardly anything has been contributed to its qualitative-structural aspect. We cannot expect that the behavior of a small scale system This is what von Neumann (1960) presumably has in mind when he argued that there is a kind of "threshold" of complexity below which the world behaves with its familiar regularities, but above which entirely new models of behavior appear (such as self-reproduction, evaluation and free will) which are *sui generis* with no counterparts in systems of lesser complexity. This idea has been further pursued with respect to socioeconomic systems by Albin (1975) on the basis of cellular automata theory which is related to the present work.

Ultimately, it is also of no use to discount complexity by simply arguing that there are simpler and more tractable problems at hand that seem profitable to deal with. There is a potential danger in the attitude of Bellman and Kalaba (1965), namely, "that in order to understand we must throw away information. We cannot, at least at this level of our intellectual development, grapple with a high order of complexity. Consequently, we must simplify." If so, and in view of the qualitative aspect of complexity, we are extremely limited in our tools to handle, explain, and control complex systems. On the other hand, we are increasingly confronted with problems of far reaching social, economic, and political significance requiring a pressing resolution and those problems impose on us the necessity to deal with the very nature of complexity.

Rosen (~~(in press)~~) calls the qualitative feature of complexity a system's property and claims that it can be measured. In terms of construction he suggests to lock measurement devices between interacting subsystems registering the amount of computation being performed in the process of

interaction. Complexity, as a system's property has been treated under the term "organizational complexity" by Weaver (1948), and more recent by Brewer (1973). Weaver, in his paper (which, unfortunately, has been largely overlooked), draws a distinction between "disorganized" and "organized" complexity. He claims that problems of disorganized complexity relate to those situations where events occur in an accidental, random fashion. (An accident may hit many individuals in a random fashion, completely unpredictable at an individual level, but an insurance company feels confident to calculate the average risk per accident.)

A problem of disorganized complexity is one, according to Weaver (1948)

> in which the number of variables is very large, and one, in which each of the many variables has a behavior which is individually erratic, or perhaps totally unknown. However, in spite of this helter-skelter, or unknown behavior of all the individual variables, the system as a whole possesses certain orderly and analyzable average properties. (p. 538)

The conditions under which such systems act are similar to those we mentioned at the beginning of this section where infinite analysis applies, i.e., these cases can be handled by probability theory and statistical techniques. "Organized" complexity, on the other hand, pertains to a situation that defies statistical analysis in which the number of variables is not too small but not too large either. The variables are related in a complicated, yet organized way and they require a different treatment of complexity as a system property. It is interesting that Weaver himself (1948) refers for support of his concept of "organized" complexity to economic examples:

> How can currency be wisely and effectively stabilized? To what extent is it safe to depend on the free interplay of such economic forces as supply and demand? To what extent must systems of economic control be employed to prevent the wide swings from prosperity to depression? These are also obviously complex problems, and they too involve analyzing systems which are organic wholes, with their parts in close interrelation. ... These new problems ... cannot be handled with the statistical techniques so effective in describing average behavior in problems of disorganized complexity. (p. 539)

It is precisely this kind of complexity which bears the most challenging task for future scientific investigations.

Brewer (1973) distinguishes a system's *analytic* size from its *empirical* or *actual* size. By analytic size he means the number of variables, the specification of their relationships (of the model describing the real system).

> The greater the divergence between the analytic size of a descriptive model and the actual size of the social system the more information is lost and the less reliable the model is as a basis for policy recommendations.
>
> ... Thus we assume that analysts would attempt to increase the complexity or analytic size of the descriptive models they use to understand such a social system.

Somewhat related, but developed in a more rigorous framework, I distinguish (Gottinger, 1976) between *structural* and *computational* complexity (see Figure 2). The first roughly indicates the complexity of the subsystems that hooked together (in a particular way to be explained) realize the entire system. The latter means the computational length that is required by (inter-)connecting subsystems to realize the entire system.

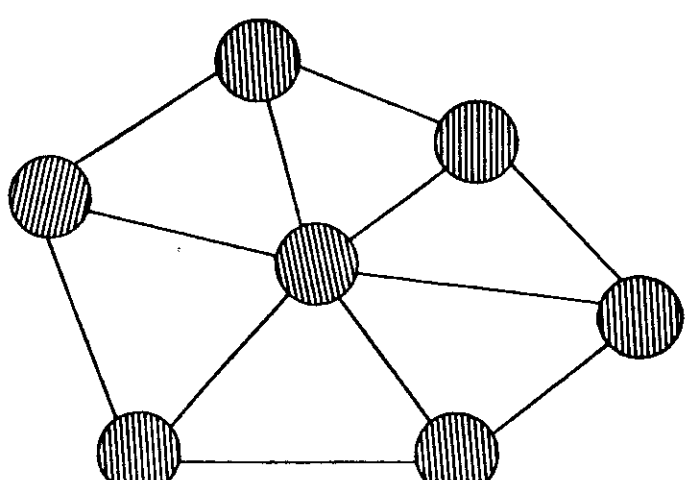


FIGURE 2. Representation of computational (links) and structural (circles) complexity.

For conceptual purpose one would like to keep these two notions apart, in practice, however, one would like to measure both characterizations of complexity simultaneously.

Our ideas to realize a given real system by any appropriately structured artificial system corresponds to what Brewer (1973) must have in mind when he speaks of increasing the analytic complexity of a model to cope with the complexity of the real system. His analytic complexity, in fact, plays the same role as our control complexity, whereas his empirical complexity is much akin to our design complexity.

With respect to analytic complexity Brewer (1973) says:

> One loses control. Confidence in the symbol system's structure decreases, as the number of elements, interconnections, relationships and measurement errors increases ... At some level of size for a given model we decidedly lose the ability to make structural revisions, i.e., to improve the model theoretically. (p. 32)

The time seems ripe for a mathematical theory of complexity being an intrinsic property of a dynamic system preserving the intuitive meaning of complexity in approaching real systems. Such a mathematical theory of complexity should obtain a similar status as probability theory. Whereas probability can be conceived as a measure of uncertainity in particular situations, complexity can be considered as a measure of the level of understanding toward a system's behavior. The usefulness of this view can be domonstrated in discussing properties of dynamic systems (Gottinger, 1975b, 1976).

COMPLEXITY OF THE MODEL

The concept of mathematical complexity was introduced by Krohn and Rhodes (Rhodes, 1971) as a natural outgrowth of the Krohn-Rhodes Prime Decomposition Theorem for Finite Semigroups and Machines. A survey of results in this field has been provided by Tilson (1971). Some definitions are in order and the notation is borrowed from the first chapter of Arbib (1968).

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

Definition 2: A semigroup S is combinatorial if and only if each subgroup of S is of order 1.

Definition 3: A *right mapping semigroup* or *right transformation semigroup* 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 $(f \cdot g)(x) = g[f(x)]$. For each $x \in X$, $s \in S$, let $xs = (x)s$. Then the following conditions are satisfied:

1. $x(s_1 s_2) = (xs_1)s_2$
2. $s_1, s_2 \in S$ and $s_1 \neq s_2$ imply $xs \neq xs_2$ for some $x \in X$

Definition 4 (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 subsemigroup of $F_R(X)$ consisting of all functions $\Psi : X \to X$ satisfying the two following conditions:

1. (triangular action). If $p_k : X \to X_k$ denotes the k^{th} 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 $t_i \in X_i$, $i = 1, \ldots, n$.
2. (k^{th} component action lies in S_k). We require $f_1 \in S_1$, and for all $k = 2, \ldots, n$ and all $\alpha = (t_{k-1}, \ldots, t_1) \in X_{k-1} \times \ldots \times X_1$, the function $g_\alpha \in F_R(X_k)$ given by $g_\alpha(y_k) = f_k(y_k, t_{k-1}, \ldots, t_1)$ is an element of S_k.

Then $(X_n,S_n) \ldots (X_1,S_1) = (X,S)$ is the *wreath product* of (X_n,S_n), $\ldots$, (X_1,S_1), and $(X_n,S_n) w \ldots w (X_1,S_1)$ is the abstract semigroup determined by (X,S).

Definition 5: Let (X,S) and (Y,T) be right mapping semigroups. Then we write (X,S) | (T,Y), read (X,S) divides (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., $Y'T' \not\subseteq Y'$).
2. There exists a map $\Theta : Y' \to\to X$ ($\to\to$ means onto) and an epimorphism $\phi : T' \to\to S$ such that $\Theta(yt) = \Theta(y)\phi(t)$ for all $y \epsilon Y'$, $t \epsilon T'$.

Definition 6: Finally, let S be a finite semigroup. Then by definition $\#_G(S)$, the (group) *complexity* of S is the smallest nonnegative integer n such that

$$S \mid (Y_n,C_n) \text{ w } (X_n,G_n) \text{ w } \ldots \text{ w } (Y_1,C_1) \text{ w } (X_1,G_1) \text{ w } (Y_0,C_0)$$

holds with $G_1, \ldots, G_n$ finite groups and $C_0, \ldots, C_n$ finite combinatorial semigroups.

It is obvious from the above definition that it would be futile to attempt to find the shortest wreath product representation for $(C'_{z_0})^S$, the semigroup of the machine C'_{z_0}. However, there are upper and lower bounds available which are computable; this is the route which will be taken in this section. But first it is necessary to look at the structure of $(C'_{z_0})^S$. As an example, I want to begin with C_{z_1} (the three-state model) and look at its semigroup in order to pave the way for an analysis of C'_{z_0}.

The elements of $C_{z_1}{}^S$ are equivalence classes of input strings, but they may be visualized more easily as transformations on the set of states $Z = \{z_1, z_2, z_3\}$. For example, let (131) represent $[(a_1)]C_{z_1}$, the class of inputs which sends z_1 to z_2, z_2 to z_3, and z_3 to z_1. By direct computation it is possible to establish that there are 24 such elements:

(111)	(211)	(311)
(112)	(212)	(312)
(113)	(221)	(313)
(121)	(222)	(322)
(122)	(223)	(323)
(123)	(231)	(331)
(131)	(232)	(332)
(133)	(233)	(333)

Since $C_{z_1}{}^S$ is a subsemigroup of $F_R(X_3)$, the full transformation semigroup on three letters, no two of these elements belong to the same equivalence class.

Furthermore, it is easy to see why $C_{z_1}{}^S \neq F_R(X_3)$, for the only

elements of $C_{z_1}{}^S$ with rank three (i.e., taking three states to three different states) are those corresponding to combinations of inputs with rank three. This is because the product (composition) of a transformation of rank n and any other transformation (on either the right or the left) will always be a transformation with rank $\leq n$. So, since a_2 is the only rank-three input and since $C_{z_i}(a_2,a_2,a_2,a_2) = C_{z_i}(a_2)$ for $i = 1,2,3$, there are only three rank-three elements of

$$C_{z_1}{}^S : [(a_2)]C_{z_1} = (231),\ [(a_2,a_2)]C_{z_1} = (321)$$

and

$$[(a_2,a_2,a_2)]C_{z_1} = 123$$

Now there is an immediate upper bound for $\#_G(C_{z_1}{}^S)$, since Krohn, Mateosian, and Rhodes (1967)

$$\#_G(C_{z_1}{}^S) \leq |\text{ spec } (C_{z_1}{}^S)| = |\ \{r > 1 : r = \text{rank } (t) \text{ for some } t \epsilon C_{z_1}{}^S\}$$
$$= |\ \{2,3\}\ | = 2$$

This also follows from the machine inequality

$$\#_G(C_{z_1}{}^S) = \Theta(C_{z_1}) \leq |\ Z\ | - 1 = 3 - 1 = 2$$

where

$$Z = \{z_1, z_2, z_3\} \quad \text{(Rhodes, 1971)}$$

In order to apply any of the existing lower bounds to $C_{z_1}{}^S$, something must be known about the local structure of this semigroup. For this purpose it is convenient to define the Green relations:

Definition 7: Let S be a semigroup. An ideal is a nonempty subset $I \subseteq S$ such that $IS \subseteq I$ and $SI \subseteq I$ (i.e., for all $i \epsilon I, s \epsilon S$, $is \epsilon I$ and $si \epsilon I$). I is a right or left ideal if the first or second, respectively, of these conditions holds. For $s \epsilon S$, $L(s) = S^1 s, R(s) = sS^1$, and $J(s) = S^1 sS^1$ are, respectively, the *principal left ideal, principal right ideal,* and *principal ideal* generated by s. Define binary relations J,L,R,H, and D on S as follows:

1. $s_1 J s_2$ if and only if $J(s_1) = J(s_2)$.
2. $s_1 L s_2$ if and only if $L(s_1) = L(s_2)$.
3. $s_1 R s_2$ if and only if $R(s_1) = R(s_2)$.
4. $s_1 H s_2$ if and only if $s_1 L s_2$ and $s_1 R s_2$.
5. $s_1 D s_2$ if and only if there exists $s \in S$ such that $s_1 L s$ and $s R s_2$ or, equivalently, if and only if there exists $t \in S$ such that $s_1 R t$ and $t L s_2$.

L,*R*,*J*, and *H* are equivalence relations on S and L_s,R_s,J_s, and H_s denote the *L*,*R*,*J*, and *H* equivalence classes, respectively, containing s.

Furthermore, the following orderings are defined on the *J*,*R*, and *L* classes of S:

1. $J_a \leq J_b$ if and only if $J(a) \subseteq J(b)$.
2. $R_a \leq R_b$ if and only if $R(a) \subseteq R(b)$.
3. $L_a \leq L_b$ if and only if $L(a) \subseteq L(b)$.

Thus principal left ideals, principal right ideals, and principal ideals of $C_{z_1}{}^S$ are the following:

1. Principal Left Ideals:

$$L[(111)] = \{(111)\}$$

$$L[(222)] = \{(222)\}$$

$$L[(333)] = \{(333)\}$$

$$L[(112)] = L[(121)] = L[(122)] = L[(211)] = L[(212)] = L[(221)] = \{(111), (112), (121), (122), (211), (212), (221), (222)\}$$

$$L[(113)] = L[(131)] = L[(133)] = L[(311)] = L[(313)] = L[(331)] = \{(111), (113), (131), (133), (311), (313), (331), (333)\}$$

$$L[(223)] = L[(232)] = L[(233)] = L[(322)] = L[(323)] = L[(332)] = \{(222), (223), (232), (233), (322), (323), (332), (333)\}$$

$$L[(123)] = L[(231)] = L[(312)] = C_{z_1}{}^S$$

2. Principal Right Ideals:

$$R[(111)] = R[(222)] = R[(333)] = \{(111), (222), (333)\}$$

$$R[(112)] = R[(113)] = R[(221)] = R[(223)] = R[(331)] = R[(332)] = \{(111), (112), (113), (221), (222), (223), (331), (332), (333)\}$$

$$R[(121)] = R[(131)] = R[(212)] = R[(232)] = R[(313)] = R[(323)] = \{(111), (121), (131), (212), (222), (232), (313), (323), (333)\}$$

$$R[(122)] = R[(133)] = R[(211)] = R[(233)] = R[(311)] = R[(322)] = \{(111), (122), (133), (211), (222), (233), (311), (322), (333)\}$$

$$R[(123)] = R[(231)] = R[(312)] = C_{z_1}{}^S$$

3. Principal Ideals:

$$J[(111)] = J[(222)] = J[(333)] = \{(111), (222), (333)\}$$

$$J[(112)] = J[(113)] = J[(121)] = J[(122)] = J[(131)] = J[(133)] = J[(211)] = J[(212)] = J[(221)] = J[(223)] = J[(232)] = J[(233)] = J[(311)] = J[(332)] = C_{z_1}{}^S - \{(123), (231), (312)\}$$

$$J[(123)] = J[(231)] = J[(312)] = C_{z_1}{}^S$$

From this information it is easy to see that the *L*, *R*, and *J* classes of $C_{z_1}{}^S$ are as follows:

1. *L* Classes:

$$L_{(111)} = \{(111)\}$$

$$L_{(222)} = \{(222)\}$$

$$L_{(333)} = \{(333)\}$$

$L_{(112)} = \{(112), (121), (122), (211), (212), (221)\}$

$L_{(113)} = \{(113), (131), (133), (311), (313), (333)\}$

$L_{(223)} = \{(223), (232), (233), (322), (323), (332)\}$

$L_{(123)} = \{(123), (231), (312)\}$

2. *R* Classes:

$R_{(111)} = \{(111), (222), (333)\}$

$R_{(112)} = \{(112), (113), (221), (223), (331), (332)\}$

$R_{(121)} = \{(121), (131), (212), (232), (313), (323)\}$

$R_{(122)} = \{(122), (133), (211), (233), (311), (322)\}$

$R_{(123)} = \{(123), (231), (312)\}$

3. *J* Classes:

$J_{(111)} = \{(111), (222), (333)\}$

$J_{(112)} = \{(112), (113), (121), (122), (131), (133), (211), (212), (221), (223), (232), (233), (311), (313), (322), (323), (331), (332)\}$

$J_{(123)} = \{(123), (231), (312)\}$

An examination of the above equivalence classes yields the following generalization.

Proposition 1: Let $X = \{1,2,3\}$. With each element t of $C_{z_1}^{S}$ we associate three things: (1) the range t(X) of t, (2) the partition $\pi_t = t \cdot t^{-1}$ of X corresponding to t, i.e., the equivalence relation on X defined by $x\pi_t y, x,y \in X$, if t(x) = t(y), and (3) the rank |t(X)| of t. Then we may describe the *L*, *R*, and *J* classes of $C_{z_1}^{S}$ as follows:

1. Two elements of $C_{z_1}^{S}$ are *L* related if and only if they have the same range.
2. Two elements of $C_{z_1}^{S}$ are *R* related if and only if they have the same partition.
3. Two elements of $C_{z_1}^{S}$ are *J* related if and only if they have the same rank.

Proof: Clifford and Preston (1961, pp. 51-52) prove the above proposition for $F_R(X_n)$. It generalizes to $C_{z_1}^{S}$, but due to the fact that $L_{(123)} = R_{(123)} = J_{(123)}$ (the highest *L*, *R*, and *J* classes of $C_{z_1}^{S}$, respectively, in the same sense that $L_t \leq L_{(123)}$, $R_t \leq R_{(123)}$, and $J_t \leq J_{(123)}$ for all $t \in C_{z_1}^{S}$) is a cyclic subgroup of $C_{z_1}^{S}$ of order three. It is not true in general for any subsemigroup of $F_R(X_n)$. Clifford and Preston prove Proposition 1(3) for *D* equivalence rather than *J* equivalence, but for finite semigroups $J = D$ (Arbib, 1968, p. 153).

This proposition may be further generalized as follows.

Proposition 2: Call $t \in F_R(X_n)$ an *order-preserving transformation* if $t = (x_i \ldots x_n x_1 \ldots x_{i-1})$ for $x_1, \ldots, x_n \in X_n = \{1, \ldots, n\}$ and $x_1 \leq \ldots \leq x_n$. An *order-preserving partition* is defined analogously, i.e., π is order-preserving if and only if there exists an order-preserving transformation with partition π. Let S be the subsemigroup of $F_R(X_n)$ consisting of all of the order-preserving elements.

1. There is a one-to-one correspondence between the set of all *J* classes of S and the set of all cardinal numbers $r \leq n$ such that the *J* class J_r corresponding to r consists of all elements of S of rank r.
2. Let r be a cardinal number $\leq n$. There is a one-to-one correspondence between the set of all *L* classes in J_r and the set of all subsets Y of X of cardinality r such that the *L* class corresponding to Y consists of all elements of S having range Y.
3. Let r be a cardinal number $\leq n$. There is a one-to-one correspondence between the set of all *R* classes contained in J_r and the set of all order-preserving partitions π of X_n for which $|X_n/\pi| = r$ such that the *R* class corresponding to π consists of all elements of S having partition π.
4. Let r be a cardinal number $\leq n$. There is a one-to-one correspondence between the set of all *H* classes in J_r and the set of all pairs (π, Y), where π is an order-preserving partition of X_n and Y is a subset of X_n such that $|X_n/\pi| = |Y| = r$, such that the *H* class corresponding to (π, Y) consists of all elements of S having partition π and range Y.

Proof: Cf. Clifford and Preston (1961, p. 52-53), and Proposition 1. These facts will be used later in the analysis of $(C'_{z0})^S$.

One last fact which is obtainable by generalization from the discussion of $F_R(X_n)$ by Clifford and Preston is the following proposition.

Proposition 3: Let S be the set of all order-preserving elements of $F_R(X_n)$, let Y be a subset of the set $X_n = \{1, \ldots, n\}$, and let π be an order-preserving partition of X_n such that $|Y| = |X_n/\pi|$. Let H be the *H* class of S determined by the pair (π, Y) as in Proposition 2(4). If H contains an idempotent, then H induces and is isomorphic with the cyclic group $Z_{|Y|}$ of order |Y|.

Proof: See Clifford and Preston (1961, p. 53-54).

It is now possible to illustrate each of the *J* classes of $C_{z_1}{}^S$ by an "eggbox picture," where each row is an *R* class, each column is an *L* class, and the intersection of each row and column is an *H* class (Figure 3).

Associated with every *J* class of a semigroup S is a semigroup $J^* = (J \cup \{0\}, \cdot)$ where

$$x \cdot y = \begin{cases} xy \text{ if } xy \in J \\ 0 \text{ otherwise} \end{cases}$$

In other words, if the product of two elements x,y of J is again in J, $x \cdot y$ is defined to be xy, but if the product xy "drops" to a lower *J* class (i.e., has rank less than the rank of x and y, in the case of $C_{z_1}{}^S$), then $x \cdot y$ is defined to be 0. Furthermore, J^* is either null, $(J^*)^2 = 0$, or 0-simple (not null and having no ideals other than 0 and itself) (Arbib, 1968, p. 152). J is called regular if J^* is 0-simple and null if J^* is null. In the case of the semigroup consisting of all of the order-preserving transformations of $F_R(X_n)$, it is obvious that all *J* classes will be regular, for each *J* class J_m $(m \leq n)$ will contain the idempotent $(1 \ldots m \ldots m)$, Thus, since $C_{z_1}{}^S$ is a special case of this semigroup for $n = 3$, it must have all regular *J* classes.

Proposition 4: Let S be the semigroup consisting of all of the order-preserving elements of $F_R(X_n)$. If J is a *J* class of S, then J is isomorphic to a regular Rees matrix semigroup (Arbib, 1968, p. 7).

$J_{(123)}$	{123}
{1} {2} {3}	(123)*
	(231)
	(312)

$J_{(112)}$	{23}	{13}	{12}
{2} {13}	(232)	(231)	(121)*
	(323)*	(313)	(212)
{3} {12}	(223)*	(113)*	(112)
	(332)	(331)	(221)
{1} {23}	(233)	(133)*	(112)*
	(322)	(311)	(211)

$J_{(111)}$	{3}	{2}	{1}
{123}	(333)*	(222)*	(111)*

FIGURE 3. "Eggbox" representations of the *J* classes for $C_{z_1}{}^S$. Starred elements are idempotents (i.e., elements e such that $e^2 = e$). They show which *H* classes are subgroups of $C_{z_1}{}^S$. Column headings represent the range of each *L* class, while row headings represent the partition corresponding to each *R* class (i.e., {1} {23} means that 1 goes to a unique element of {1,2,3}, but 2 and 3 are mapped to the same element).

Proof: All of the J classes of S are regular. Thus J^* is 0-simple for each J class J and is isomorphic to a regular Rees matrix semigroup (Arbib, 1968, p. 157).

Now with each J class J of $C_{z_1}{}^S$ it is possible to associate the structure matrix of the Rees matrix semigroup isomorphic to J. This is done as follows.

Select and H class of J containing an idempotent e, denoting R_e and L_e by R_1 and L_1, respectively, and consequently H_e by H_{11}. This amounts to moving any H class with an idempotent into the upper left hand corner of the "eggbox picture" of J, simultaneously adjusting the rows and columns according. Let $\{R_a : a\epsilon A\}$ and $\{L_b : b\epsilon B\}$ be the sets of R classes and L classes, respectively, of $C_{z_1}{}^S$ contained in J. For each $a\epsilon A$ and $b\epsilon B$, select and fix an element r_a of H_{a1} and an element z_b of H_{1b}. Define the $|B| \times |A|$ matrix $C = (c_{ba})$ over $H_{11}{}^0$ as follows

$$c_{ba} = \begin{cases} z_b r_a \text{ if } z_b r_a \in H_{11} \\ 0 \text{ otherwise} \end{cases}$$

It is clear that a different choice of representatives z_b and r_a would yield a different structure matrix; however, the Rees matrix semigroups so obtained would be isomorphic.

The structure matrices for $J_{(111)}$, $J_{(112)}$, and $J_{(123)}$ of $C_{z_1}{}^S$ are as follows

$$J_{(123)} : [(123)^*]$$

$$J_{(112)} : \begin{bmatrix} (323)^* & (312)^* & 0 \\ 0 & (323)^* & (323)^* \\ (232) & 0 & (323)^* \end{bmatrix}$$

$$J_{(111)} : \begin{bmatrix} (333)^* \\ (333)^* \\ (333)^* \end{bmatrix}$$

Now let 1 represent the idempotent of each structure group H_{11} and let g = (232); thus the above matrices become

$$J_{(123)} : [1]$$

$$J_{(112)} : \begin{bmatrix} 1 & 1 & 0 \\ 0 & 1 & 1 \\ g & 0 & 1 \end{bmatrix}$$

$$J_{(111)} : \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}$$

Proposition 5: Let S be a finite semigroup and let $J_1 > J_2 > \ldots > J_n$ be a chain of regular J classes of S. For each $i\epsilon \{1, \ldots, n\}$ let G_i be the structure group of the Rees matrix representation of J_i (i.e., the group in the H_{11} position of each J class). Furthermore, define the structure matrix C_i of J_i to be normalizable to zeros and ones (the identity of G_i) if and only if it is possible to obtain from it a matrix consisting only of zeros and ones by a process of multiplying rows and columns by elements of G_i. Let G_i' be the subgroup of G_i generated by the idempotents of J_i. Finally, define the orbit of G_i' on J_{i+1} to be the set of L classes of J_{i+1} obtained by a right group action of G_i' on J_{i+1} (i.e., the L classes whose elements are of the form $j_{i+1}g_i'$ where $j_{i+1}\epsilon J_{i+1}$ and $g_i'\epsilon G_i'$ and where j_{i+1} is fixed by the identity of G_i'). If C_{i+1}, the structure matrix of J_{i+1}, restricted to the orbit of G_i', is not normalizable to zeros and ones, call C_{i+1}' (of the restricted J_{i+1}) *essential*, and let k be the largest non-negative integer such that there exists a chain of essential groups

$$G_1 > G_2' > \ldots > G_k'$$

where $G_i' > G_{i+1}'$ if and only if $J_i > J_{i+1}$. Then $k \leq \#_G(S)$.

Proof: $k \leq \#_I'(S)$, defined by Rhodes and Tilson (1972) as follows.

Let L_i be the L class of J_i containing G_i, and let S_i be the semigroup generated by the chain $L_i > L_{i+1} > \ldots > L_n$ of L classes. Now let L_i',

$\ldots, L_n'$ be the L classes obtained by deleting from $L_1, \ldots, L_n$ those elements belonging to R classes which yields only zeros in their respective structure matrices restricted to the orbit of the preceding essential group, and let S_i' be the subsemigroup of S_i generated by $L_i', \ldots, L_n'$. Define $EG(S_i')$ to be the subsemigroup of S_i' generated by its idempotents. Then

$$S \geq S_1' \geq EG(S_1') \geq S_2' \geq EG(S_2') \geq \ldots \geq S_k' \geq EG(S_k')$$

is a chain of subsemigroups of S such that S_i' is a noncombinatorial "T_1" semigroup (i.e., generated by a chain of its L classes) contained in $EG(S_{i-1}')$, for $i = 1, \ldots, k$. This is guaranteed by the fact that each structure matrix C_i, $i = 2, \ldots, k$, is not normalizable to zeros and ones. $\#_I(S)$ = maximum of lengths of all such chains of subsemigroups of S. Thus $k \leq \#_I(S) \leq \#_G(S)$.

In the case of $C_{z_1}{}^S$ notice that, if $J_{(123)}$ is denoted by J_1, $J_{(112)}$ by J_2, and $J_{(111)}$ by J_3, then $G_1 > G_2' = G_2$ is such a chain. $G_2 = G_2'$ here due to the fact that $g = (232) = (121)\cdot(133)\cdot(233)$ and $G_2 = \{1,g\}$. Thus $k = 2 \leq \#_G(C_{z_1}{}^S)$, but it was already shown (see page [illegible]) that $\#_G(C_{z_1}{}^S) \leq 2$, so $\#_G(C_{z_1}{}^S) = 2$.

Let us now discuss C_{z_0}', the ten-state model. $(C_{z_0}')^S$ contains as elements equivalence classes of input strings, but as in the discussion of $C_{z_1}{}^S$, they will be considered here as transformations on $Z' = \{z_0, \ldots, z_9\}$.

Proposition 6: $(C_{z_0}')^S$ is the subsemigroup of $F_R(X_{10})$ consisting of all its order-preserving elements.

Proof: 1. Notice that the transformations corresponding to each of the unit length input strings are order-preserving. Thus $(C_{z_0}')^S$ is a subsemigroup of $F_R(X_{10})$ and consists only of order-preserving transformations, ~~since the product (composition) of two order-preserving only of order-preserving transformations,~~ since the product (composition) of two order-preserving transformations is another order-preserving transformation and since every element of $(C_{z_0}')^S$ corresponds to a product (concatenation) of unit length input strings.

2. Now notice that $(C_{z_0}')^S$ contains (1234567890), the transformation representing $[(a_2)]C_{z_0}'$. This element generates all other order-preserving

transformations of rank ten (i.e., taking ten states to ten different states), the collection of which is a cyclic group isomorphic to Z_{10}. Now multiplying any rank-nine element (say (2234567801) = $[(a_3)]C_{z_0}'$) on the right by each of the elements of rank ten gives a rank-nine transformation of any desired range. Similarly, by multiplying a rank-nine element on the left by all of the rank-ten elements, all possible order-preserving partitions of X_{10} of order nine may be obtained. In order to get a rank-nine transformation of range Y and partition π, choose one of the already existing rank-nine elements with partition π and multiply it on the right by an appropriate rank-ten transformation. Thus all order-preserving elements of $F_R(X_{10})$ of rank nine and ten are included in $(C_{z_0}')^S$.

Now all order-preserving elements of $F_R(X_{10})$ of rank n ($n = 1, \ldots, 8$) are obtained in like manner from the elements of higher rank.

Proposition 7: $\#_G[(C_{z_0}')^S] \leq 9$.

Proof: As for $C_{z_1}{}^S$, there are two ways to show this, one by thinking of $(C_{z_0}')^S$ as a semigroup and the other by considering the machine C_{z_0}':

1. $\#_G[(C_{z_0}')^S] \leq |\operatorname{spec}[C_{z_0}')^S]|$
$= |\{r > 1 : r = \operatorname{rank}(t) \text{ for some } t \in (C_{z_0}')^S\}|$
$= |\{2,3, \ldots, 10\}| = 9$

2. $\#_G[(C_{z_0}')^S] = \theta(C_{z_0}') \leq |Z'| - 1 = 10 - 1 = 9$

By Proposition 2 it is now possible to go directly to the "eggbox picture" of each of the J classes of $(C_{z_0}')^S$. However, since in this case the J classes are quite large (in comparison to those for $C_{z_1}{}^S$), it is advantageous to look only at the "relevant" portion of each J class. This is accomplished in two ways:

1. Included are only those L classes of J_i which are in the orbit of the essential group G_{i+1}' belonging to J_{i-1} (as defined in Proposition 5).
2. All R classes of J_i which yield only zeros in the structure matrix for J_i [when restricted as described in (1)] are excluded.

Since all of these J classes are regular, each J_i' is isomorphic to a regular Rees matrix semigroup (Proposition 4) and so, in particular, it is possible to compute their structure matrices C_i (i.e., the structure matrices of the "reduced" J classes).

Proposition 8: $\#_G[(C'_{z_0})^S] = 9$.

Proof: Notice that $C_2, \ldots, C_9$ are not normalizable to zeros and ones. Furthermore, $G_i' = G_i$ in each of these cases, since a generator of G_i is obtainable as the product of idempotents as follows

$$e_{(1,\,12\text{-}i)} \cdot e_{(12\text{-}i,\,11\text{-}i)} \cdot e_{(11\text{-}i,\,10\text{-}i)} \cdots e_{(3,2)} \cdot e_{(2,1)}$$

$$= e_{(1,\,12\text{-}i)} \cdot \left| \prod_{n=-12}^{-i-2} \right| e_{(-i\text{-}n,\,-i\text{-}n\text{-}1)}$$

$$= g_{11\text{-}i}^{-1}$$

where $g_{i\,1\text{-}i}$ is the non-normalizable element of C_i. For example, if $i = 2$, then

$$e_{(1,\,12\text{-}i)} \cdot \left| \prod_{n=-12}^{-i-2} \right| e_{(-i\text{-}n,\,-i\text{-}1)}$$

$$= e_{(1,\,10)} \cdot e_{(10,9)} \cdot e_{(9,8)} \cdots e_{(2,1)}$$

$$= (0123456780) \cdot (0123456799) \cdot (0123456889)$$

$$\cdot (0123457789) \cdot (0123466789) \cdot (0123556789)$$

$$\cdot (0124456789) \cdot (0133456789) \cdot (0223456789)$$

$$\cdot (1123456789)$$

$$= (1234567891)$$

$$= g_9^{-1}$$

Hence $\langle g_{11\text{-}i}^{-1} \rangle = G_i' = G_i$, since G_i is cyclic.

Thus G_i' is essential for $i = 2, \ldots, 9$, and

$$Z_{10} \equiv G_1 > G_2' = G_2' > G_3' = G_3 > \ldots > G_9' = G_9$$

is a chain of essential groups with length $k = 9$. Therefore $9 \leqslant \#_G (C'_{z_0})^S$ and by Proposition 7, $\#_G[(C'_{z_0})^S] \leqslant 9$. Thus

$$9 \leqslant \#_G(C'_{z_0})^S \leqslant 9 \text{ and } \#_G[(C'_{z_0})^S] = 9$$

THE GENERALIZED MODEL

In fact, the set of internal states Z' could be enlarged in order to simulate the action potential with more quantitative accuracy. It will be shown now that as $|Z'| = n$ is increased, the complexity of the model remains n-1. Let $C_{z_0}{}^n$ represent the n-state machine simulation. The elements of $(C_{z_0}{}^n)^S$, the semigroup of $C_{z_0}{}^n$, will again be treated as order-preserving transformations on $Z^n = \{z_0, \ldots, z_{n-1}\}$.

Proposition 9: $(C_{z_0}{}^n)^S$ is the semigroup of $F_R(X_n)$ consisting of all of its order-preserving transformations.

Proof: 1. As with $(C'_{z_0})^S$, all of the transformations corresponding to unit length input strings are order-preserving, due to the nature of the model, and so $(C_{z_0}{}^n)^S$ is a subsemigroup of $F_R(X_n)$ and consists only of order-preserving transformations. This requires the same argument as in part (1) of the proof of Proposition 6.

2. $(C_{z_0}{}^n)^S$ contains $[123\ldots(n\text{-}1)0]$, since a stimulus sends each state z_i to the next state z_{i+1}, with the exception of z_{n-1}, which goes to z_0. $[123\ldots(n\text{-}1)0]$ generates all other order-preserving transformations of rank n, the collection of which is a cyclic group isomorphic to Z_n. Now at least one rank-n-1 element exists since any stimulus of unit length is presentable by the transformation $[01\ldots(i\text{-}1)\,(i\text{+}1)\ldots(n\text{-}1)0]$. Multiplying any rank-n-1 element on the right by each of the elements of rank n gives a rank-n-1 transformation of any desired range. Similarly, by multiplying a rank-n-1 element on the left by all of the rank-n elements, all possible order-preserving partitions of X_n of order n-1 may be obtained. In order to get a rank-n-1 transformation of range Y and partition π, choose one of the already existing rank-n-1 elements with partition π and multiply it one the right by an appropriate rank-n transformation. Thus all order-preserving elements of $F_R(X_n)$ of rank n and n-1 are included in $(C_{z_0}{}^n)^S$.

Now all order-preserving elements of $F_R(X_n)$ of rank $1, \ldots, n\text{-}2$ are obtained in like manner from the elements of higher rank.

Proposition 10: $\#_G[(C_{z_e}{}^n)^S] \leq n\text{-}1$.

Proof: There are again two methods available:

1. $\#_G[(C_{z_e}{}^n)^S] \leq |\text{spec } [(C_{z_e}{}^n)^S]|$

$$= |\{r > 1 : r = \text{rank } (t) \text{ for some } te(C_{z_e}{}^n)^S\}|$$

$$|\{2, 3, \ldots, n\}| \wedge$$

2. $\#_G[(C_{z_e}{}^n)^S] = \theta(C_{z_e}{}^n)^S \leq |Z^n| - 1 = n\text{-}1$.

By Proposition 10 it is possible to go directly to the "eggbox picture" of each of the "reduced" J classes of $(C_{z_e}{}^n)^S$.

Since all of these J classes are regular, each J_i^* is isomorphic to a regular Rees matrix semigroup (Proposition 4), and so it is possible to compute their structure matrices C_i.

Proposition 11: $\#_G[(C_{z_e}{}^n)^S] = n\text{-}1$.

Proof: Notice that $C_2, \ldots, C_{n-1}$ are not normalizable to zeros and ones. Furthermore, $G_i' = G_i$ in each of these cases, since a generator of G_i is obtainable as a product of idempotents as described in the proof of Proposition 8, with the following modification:

$$e_{(1,\, n+2-i)} \cdot e_{(n+2-i,\, n+1-i)} \cdot e_{(n+1-i,\, n-1)} \cdots e_{(3,2)} \cdot e_{(2,1)}$$

$$= e_{(1,\, n+2-i)} \cdot \left| \prod_{j=n-2}^{-i-2} \right| e_{(-i-j,\, -i-j-1)}$$

$$= g_{n+1-i}^{-1}$$

where g_{n+1-i} is the non-normalizable element of C_i.

Hence $\langle g_{n+1-i}^{-1} \rangle = G_i' = G_i$, since G_i is cyclic. Thus G_i' is essential for $i = 2, \ldots, n\text{-}1$, and

$$Z_n \cong G_1 > G_2' = G_2 > G_3' = G_3 > \ldots > G_{n-1}' = G_{n-1}$$

is a chain of essential groups with length $k = n\text{-}1$. Therefore $n\text{-}1 \leq \#_G[(C_{z_e}{}^n)^S]$ and by Proposition 10, $\#_G[(C_{z_e}{}^n)^S] \leq n\text{-}1$. Thus $n\text{-}1 \leq \#_G[(C_{z_e}{}^n)^S] \leq n\text{-}1$ and $\#_G[(C_{z_e}{}^n)^S] = n\text{-}1$.

By Proposition 11, the complexity of $C_{z_e}{}^n$ is maximal regardless of the number of states n. This is significant in that it implies that the action potential is inherently complex, independent of the finite-state machine by which it is simulated. This result supports a theory of evolved and evolving organisms outlined by Rhodes (1971). He states that "an evolving organism transforms itself in such a manner so as to maximize the contact with the complete environment subject to reasonable control and understanding of the contacted environment." Interpreting this statement in terms of automata theory, an "evolving" finite-state machine transforms itself in such a manner so as to maximize its number of states while maintaining its complexity at a near-maximal level. Evolved organisms, on the other hand, must be evolutionarily stable (i.e., complexity must be within approximately 5% of the contact). In this case, the fact that the action potential has absolutely maximal complexity regardless of the cardinality of the set of internal states $|Z'|$ indicates that it is indeed stable under the forces of evolution.

REFERENCES

Albin, P. S. 1975. *The analysis of complex socioeconomic system.* Lexington, Mass.: Heath.

Arbib, M. A. 1968. *Algebraic theory of machines, languages, and semigroups.* New York: Academic.

Arbib, M. A. 1964. *Brains, machines and mathematics.* New York: McGraw-Hill.

Bellman, R. 1965. Mathematical aspects of the theory of systems. *Rand P-3032, Rand Corporation,* Santa Monica, California, May 1965.

Bellman, R., and Kalaba, R. 1965. *Quasilinearization and boundary value problems.* New York: American Elsevier.

Bremermann, H. 1974a. Algorithms, complexity, transcomputability and the analysis of systems. in *Cybernetic congress Nürnberg* eds. Händler, Keidel, Spreng. pp. 250-263. Munich: Oldenbourg.

Bremermann, H. 1974b. Complexity of automata, brains and behavior. In *Physics and mathematics of the nervous system,* eds. M. Conrad et al. pp. 304-331. Berlin and New York: Springer Verlag.

Brewer, G. D. 1973. Analysis of complex systems: An experiment and its implications for policy making. *Rand -P-4951,* Rand Corporation, Santa Monica, January 1973.

Clifford, A. H., and Preston, G. B. 1961. *The algebraic theory of semigroups,* vol. 1. Providence, R. I.: Amer. Math. Soc.

Gottinger, H. W. 1973. Computable Organizations: Representation by Sequential Machine Theory, *Ann. Syst. Res.* 3:81-108.

Gottinger, H. W. 1975a. Complexity and information technology in dynamic systems. *Kybernetes* 4:129-141.

Gottinger,' 1975b. Notes on dynamic systems and social processes. *Gen. Syst.* 20:121-134.

Gottinger, H. W. 1976. Complexity and Catastrophe: Application of Dynamic System Theory, *Working Paper* No. 248, *Western Management Science Institute,* Univ. of California, Los Angeles, March 1976.

Hurwicz, L. 1959. Optimality and informational efficiency in resource allocation processes. In *Mathematical methods in social sciences.* Stanford: Stanford University Press.

Krohn, K., Mateosian, R., and Rhodes, J. 1967. Complexity of Ideals in Finite Semigroups and Finite-State Machines. *Math. Syst. Theor.* 1:59-66, 373.

Mackay, D. M. 1954. On comparing the brain with machines. *Am. Sci.* 42:261-268.

Marschak, Th., and McGuire, C. B. 2973. Design of Organization and Information Technology. Unpublished manuscript, University of California, Berkeley.

Minsky, M. L. 1967. *Computation: Finite and infinite machines.* Englewood-Cliffs, N.J.: Prentice-Hall.

Moiseev, N. N., and Schmidt, A. G. 1973. Some Problems of Centralized Economy. Cowles Foundation Disc. Paper No. 358 (Yale University), April 5, 1973.

Nicolis, G., and Prigogine, I. 1973. Fluctuations and the mechanism of instabilities. In *Proc. 3rd int. conf. from theoretical physics to biology,* pp. 89-109, Versailles 1971. Basel: Karger 1973.

Rhodes, J. L. 1971. Application of automata theory and algebra. Lecture Notes, Dept. of Math., Univ. of California, Berkely.

Rhodes, J. L., and Tilson, B. R. 1972. Improved lower bounds for complexity of finite semigroups. *J. Pure Appl. Alg.* vol. 3.

Rosen, R. 1975a. Complexity and error in social dynamics. *Int. J. Gen. Syst.* vol. 1.

Rosen, R. 1975b. *Biological systems as paradigmas for adaptation.* Center for Theoretical Biology, *State Univ. of New York at Buffalo, unpublished manuscript.*

Rosen, R. 1976. Complexity as a system property. Int. J. Gen. Syst. (In press).

Simon, H. A. 1969. The architecture of complexity. In *The sciences of the artifical.* Cambridge, Mass.: M.I.T. Press.

Tilson, B. 1971. Decomposition and complexity of finite semigroups. *Semigroup Forum* 2:189-250.

von Neumann, J. 1960. Probabilistic logics and the synthesis of reliable organisms from unreliable components. In *Automata Studies* ed. C. E. Shannon, Princeton: Princeton Univ. Press.

Weaver, W. 1948. Science and Complexity *Am. Sci.* 36:538.

COMPUTABLE ORGANIZATIONS-REPRESENTATION BY SEQUENTIAL MACHINE THEORY*

HANS W. GOTTINGER**

Abstract

In this paper we investigate certain types of organizational forms which are considered to be sequentially computable rather than Turing computable, i.e. we are considering those organizations which are subject to definite resource and time constraints and which can be split into elementary computational operations.

It is argued that organizations could be effectively modelled in the sequential machine framework and that topics dealt within conventional organization theory (on Hurwicz' lines) could be treated more generally. Furthermore, problems concerning the structure of information technology, incentive compatibility and computational complexity fit naturally into this approach.

Finally we expose an algebraic theory of adjustment processes based on semigroups of transformations which could be solved by certain types of functional equations.

1. Introduction and motivation

In recent years, particular research efforts have been directed toward explaining structure, behavior and performance of economic organizations. It has been increasingly recognized in most approaches that we should look upon organizations in a normative fashion – from a designer's point of view – e.g. how to construct organizations which will perform certain tasks we want them to do. To some extent we are interested in their existence and then ask the question how they would perform 'best', i.e. most efficiently or at least satisfactorily given their existence. A particular organizational form, the competitive economy, has received most attention. The question is essentially the following: Let an economy consist of agents, involved in a competitive process, and so, that they act in response to their changing 'environments' and to actions by other agents – subsume in 'messages' (prices). Now an adjustment process in

Research sponsored in part by the Army Research Office – Durham, Grant DA-ARO-D-31-124-73-G174

* Paper presented at the Cambridge Conference on Public Systems, Cambridge University, November 23-25

** Visiting Professor, Universität Bielefeld, Institut für Entwicklungsforschung, Bayerische Akademie der Wissenschaften, München

this organization, more informally, is a kind of scheme or process which this organization reveals at each iteration and which would satisfy certain properties to the best of all members of this organization. In this context, an adjustment process can be viewed as a sequence of aggregate actions (behavior patterns) taken by each agent. A class of (economic) environments is the triple $X = (\Omega, \mathscr{R}, \mathscr{T})$ where Ω describes the set of resources, $\mathscr{R}$ a set of preference relations on Ω and $\mathscr{T}$ a set of feasible technologies. Any given environment can be represented as a parameter x of the class X.

For different classes of environments, L. Hurwicz [3] has studied adjustment processes in terms of difference equations in which agents respond to messages from other agents including themselves (memorizing). (Of course, the agent may be completely or only partially ignorant about the environment, in this case stochastic responses have to be considered.) Hence, in technical language, an adjustment process is a triple $(\lambda, \delta, \mathscr{M})$, consisting of a response function λ (possibly a vector for a finite number of agents), an outcome function δ, independent of changes in the environment but depending on the amount of resource endowment, trade, production, etc. given the environment and a message space ('language') $\mathscr{M}$ whose elements ('messages') generate new messages (via the response function λ) for any given environment x. There is associated a message acting as a stimulus on every agent. If sufficient information has been collected by the agents (and the response resulting in different types of actions such as trading, producing, storing, etc. is uniquely determined such that additional information will not result in a different response), the process is called to be *in equilibrium* and the message received at that stage is *stationary*. To every informational equilibrium value of the process $\bar{m} = \lambda(\bar{m}, x)$ there may correspond a (Pareto-) satisfactory outcome level $\delta(\bar{m})$ which is preferred to any other outcome level for any given environment. The behavior pattern of such an economic system can be studied in terms of a particular social welfare function satisfying an optimality criterion (Pareto optimality) given an environment of a particular kind (classical or non-classical environments). A class of environments is called 'classical' if externalities and indivisibilities are absent and if both technology and preferences are convex; otherwise it is called non-classical. On the basis of the adjustment process new states will be generated up to a point where the final state is compatible with the welfare criterion. Some important results in this area have been obtained, notably by Hurwicz [3], for a class of processes which may or may not be Pareto satisfactory for all conceivable environments. In particular, it has been shown that the competitive process acting in a classical environment is Pareto satisfactory. In principle, at least, a similar adjustment process could be established by a central agent having only partial information about the environment, constituting an algorithmic approach to the solution of the problem.

On the other hand, in nonclassical environments with externalities and indivisibilities present and technology not necessarily convex other types of processes different from the competitive process have been studied with respect to optimality properties. It is well known that the evaluation of the process has to be based primarily on the informational requirements necessary to establish a Pareto satisfactory process, and secondarily on the incentive compatibility with the actions of the various agents. The first point has to do with the *computability* of the adjustment process, i.e. with the capacity of various agents to process and disseminate information. There are actually two aspects of the first point: one aspect concerns the purely 'technological' problem of selecting the appropriate or even the minimal 'information-handling equipment' capable to do the job. Since information-handling usually involves costs the other aspect relates to the problem of selecting those information-handling equipments which cause minimal costs. Both aspects deal with the question of informational efficiency in various organizations. (Both aspects will come up later in a different framework). As it is known, the question of informational efficiency, in a more imprecise formulation, gave rise to controversies about the choice of economic systems many years ago.

The second point involves the question of goal-compatible behavior patterns of economic agents (incentive compatibility) which in a competitive system are satisfied, given the classical environment, by assuming profit – and utility maximization. We will not deal with the second point in this paper, although this point will come up at various instances.

Recent work on adjustment processes along Hurwicz' lines (see Reiter [7]) contains mainly some mathematical refinements of previous results which center around the question of informational efficiency. It is assumed that the space of environment X, the message space $\mathscr{M}$ and the space of actions A are all topological spaces whereas the adjustment process starts from some subset of the message space defined by a correspondence $\mu: X \to \mathscr{M}$ and a response function $\lambda: \mathscr{M} \to A$. Hence the adjustment process (μ, λ) is induced by an initial message set $\mu_0(x)$. The outcome function $\delta: \mathscr{M} \to A$ may be introduced in the appropriate context. Reiter shows – technicalities omitted – that the response function satisfied some 'nice' system properties which could be derived from the topological structure of the underlying spaces. Contrary to this approach we consider it more natural that such a response function reveals its structure and behavior in the context of a device which is known as a *sequential machine*.

The perspective is to consider sequential machines as basic analogues for modelling complex 'humanistic' systems (organizations), and to treat adjustment processes in terms of transformations on the set of states of a machine.

Not only would we be interested in translating the language of the economic theory of organizations into proper machine language but also would we like to answer some specific questions within the framework adopted. We list these questions now, somewhat informally, since we provide suitable definitions later.

1) Given a machine M what 'information technology' is necessary and sufficient to realize this machine by serial, parallel, serial-parallel or cascade decomposition into component machines. In other words, what kind of information technology is needed to accomplish the task of the original machine by an appropriate sequence of submachines?

2) If several information technologies are compatible with the performance of the original machine, then does there exist a unique optimal one? If so, are the costs of information processing, induced by the information technology feasible in view of an initial resource endowment given to the machine?

3) What corresponding type of adjustment process could be derived for an optimal information technology?

The ultimate goal, of course, is the attempt to construct a computational theory of organization where we are able to show – as the engineer does by constructing a machine from pieces of hardware – how an organization should be structured in order to achieve its goals. It is well known that practising engineers, although they construct all sorts of finite machines, have so far relied predominantly on empirical techniques, e.g. how to put various pieces of hardware together and have neglected design methods provided by the algebraic theory of sequential machines. Only in recent time this theory receives increasing attention in practising circles. Now, it seems to me, that the economist should also adopt a designer's point of view when he is talking about structure, behavior and performance of an economic system or organization. As outlined above, various other approaches have been suggested to arrive at a normative theory of organization, but not much has been done to approach it on methodological grounds of automata theory which seems to be a natural one in designing an organization.

2. Structure of sequential machines

In order to keep the presentation self-contained we present some notions of machine theory. Most of this material is taken from Hartmanis and Stearns [2]. In general, automata as represented by sequential machines form discrete systems, and the notions applied fall in the realm of algebra. We will try to give some intuitive justification for modelling organizations as sequential machines.

Definition: A sequential machine[1]) is a quintuple $\langle X, Y, Z, \lambda, \delta \rangle$ where X is a nonempty set of inputs, Y a set of outputs, Z a set of states, $\lambda: X \times Z \to Z$ a

We restrict all sets to be finite. In the context of looking at economic systems formulated as sequential machines all sets and functions involved have a definite interpretation. X denotes the set of environments (to which there is associated a message set $\mathcal{M}$ so that to every message $m \in \mathcal{M}$ there corresponds a state of the environment $x \in X$). We consider the response to be represented by a function $\lambda: X \times Z \to Z$ and the outcome function to be $\delta: X \times Z \to Y$ where the state set Z represents the physical and informational activity of the system. Now there is one problem by transforming the set of environment into an input set of a sequential machine.[1a]) An intuitively appealing way is to let the machine only accept those pairs of commodity bundles and production vectors as inputs which have been chosen by the agents.

Definition: An organization is the machine $\langle X, Y, Z, \lambda, \delta \rangle$ with symbols in brackets as appropriately defined above.

Let me provide an example why it is reasonable to view an organization in machine-like terms.

Example: We consider some kind of control device where you (the designer) want to control someone's action according to the message received. Take such organization as an AIRPORT PARKING LOT or TOLL BRIDGE and look at it strictly from a designer's point of view: how should a parking lot be operated? The first thing to do is to announce an exhaustive list of instructions and to make it available to everyone entering the parking lot. There may be a set of instructions such as: 'Stop until 75 cents (in coins) are deposited (red light). Then go if light turns green'. Now everything is fine if this set of instructions is complied with. However, there are other possibilities to be taken care of by the organization constituting a penalty-reward system. Consider the following cases:

1) the message is not received for whatever reasons (nothing happening).
2) the instructions are only complied with incompletely (only one quarter is deposited but not two, three, etc.).
3) the instructions are flagrantly violated (no money is deposited).

In all these cases appropriate actions have to be taken describing the response to the message given the state and they are reflected in the following table. We could look at this organization as a human automaton, but we could also look at it as an electrical device which simulates the human machine, in fact, it could be a device which transforms the state-message pair into an action-next state pair. Of course, this requires quite a bit of hardware construction, but what it mainly amounts to is to put stimulus, response or state as voltages on a bundle of lines (wires) and to encode them in proper form (for example in binary form). The organization we would like to describe as an electrical device would then be represented by the following scheme (here → denotes

		no message received	message incomplete	message violated
states	0	stop	stop, go to row 1	alarm, go to row 0
	1	stop	stop, go to row 2	alarm, go to row 0
	2	stop	go to row 3	alarm, go to row 0
	3	go	go	stop

next states

Figure 1.

stimuli

states		00	01	10
	00	00	00→01	10→00
	01	00	00→10	10→00
	10	00	01→11	10→00
	11	01	01	00

next states

Figure 2.

In case of the states we have the following correspondences: 0–00, 1–01, 2–10, 3–11. There are similar assignments to stimuli and responses, as exhibited in Fig. 2. Both devices, the human and the electrical one, obviously perform the same tasks, in terms of performance one machine is as good as the other. It is hence natural to describe the second machine as a homomorphic image (or homomorphism) of the first, since it is supposed to transform all operations performed by the first machine into the same operations performed by the second machine.

Now, for this simple kind of example, which obviously is a crude one, all that we want to conclude is that, in principle, there is no difference between an engineering design and the design of a human organization. Other examples of control systems and organizational designs are discussed by T. Marschak and C. B. McGuire [5]. They describe different control systems in terms of car-driving. Consider a car driving along a windy road. The conditions of the road may constitute the stimuli to the car-driver, e.g. left curb, right curb, going

straight. The question is how to control a car in order to stay on the road, hence it concerns various steering actions given the stimuli. Incidentally, to the best of my knowledge, Marschak and McGuire were the first to view organizational behavior in the sequential machine framework.

Although this might be obvious for execution-type operations as described above, we will face difficulties where managerial-type decisions will come to play or where problems of incentives, competence, cooperation, competition etc. enter the picture of the organization's performance. In fact, it is this type of situation for which one might question the applicability of sequential machine theory to the design of organizations. In this context, John Rhodes, in a private conversation, argued that situations requiring extensive logical operations and computations might better be covered by a theory of TURING MACHINES rather than of SEQUENTIAL MACHINES.[2]) On the other hand, I do not find it unreasonable to argue that managerial ability, for example, could find its proper treatment on the basis of computational complexity of a sequential machine, pertaining to such notions as speed of recollection, recognition (of observations), execution, decoding of messages, minimal number of erroneous actions etc. In fact, it would seem to be appropriate to view computational complexity as a copy or multiple of elementary computations. If a machine is too complex, i.e. generates too many states to compute its own solution we would like to decompose it into simpler parts so that they altogether solve the computational problem. The question of decomposition of a machine naturally comes up in the decision-theoretic description of an economic organization In view of suggestions due to Marschak and McGuire we consider first two kinds or organizations, a decision and a pay-off machine, hooked together, to make a new machine. More precisely, we could define:

Definition: Given an organization $M = \langle X, Y, Z, \lambda, \delta \rangle$. Then it is possible to represent M by a serial decomposition into decision machine $M_1 = \langle X, A, Z_1, \lambda_1, \delta_1 \rangle$ and payoff machine $M_2 = \langle A, Y, Z_2, \lambda_2, \delta_2 \rangle$ to generate the machine $M_1 \oplus M_2 = \langle X, Y, Z_1 \times Z_2, \lambda, \delta \rangle$ with $\lambda[(z_1, z_2), x] = [\lambda_1(x, z_1), \lambda_2(a, z_2)] = [\lambda_1(x, z_1), \lambda_2(\delta_1(x, z_1), z_2)]$ and $\delta[(z_1, z_2), x] = \delta_2[z_2, \delta_1(x, z_1)]$.

For reasons of nontriviality, M_1 and M_2 have fewer states than M.

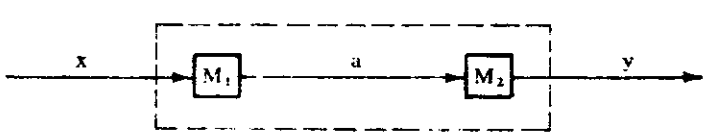


Figure 3.

Serial connection of decision machine M_1 and payoff machine M_2.

A slightly more general case is provided by a serial decomposition of M into three types of machines

$M_0 = \langle X, M, Z_0, \lambda_0, \delta_0 \rangle$ (message machine)
$M_1 = \langle M, A, Z_1, \lambda_1, \delta_1 \rangle$ (decision machine)
$M_2 = \langle A, Y, Z_2, \lambda_2, \delta_2 \rangle$ (payoff machine)

to generate a new machine

$$M_0 \oplus M_1 \oplus M_2 = M = \langle X, Y, Z_1 \times Z_2 \times Z_3, \lambda, \delta \rangle$$

with somewhat more complicated state and output functions than those given in the foregoing example.

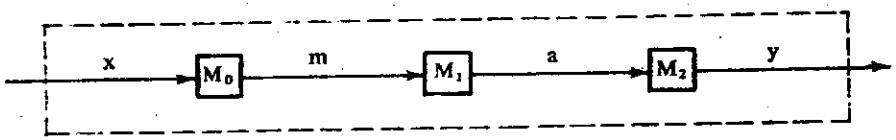


Figure 4.

In an analogous way we could talk about parallel decomposition.

Definition: A (decision) machine M can be realized by parallel decomposition into component machines to generate

$M = M_1 \otimes M_2 = \langle X_1 \times X_2, A_1 \times A_2, Z_1 \times Z_2, \lambda, \delta \rangle$ with state representation

$\lambda[(x_1, x_2), (z_1, z_2)] = (\lambda_1(x_1, z_1), \lambda_2(x_2, z_2))$ and output representation $\delta[(x_1, x_2), (z_1, z_2)] = (\delta_1(x_1, z_1), \delta_2(x_2, z_2))$.

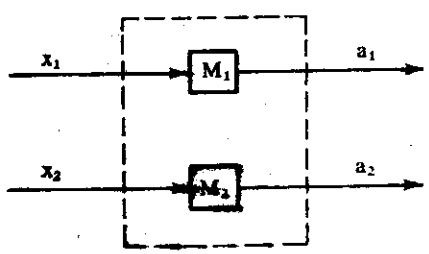


Figure 5.

Parallel connection

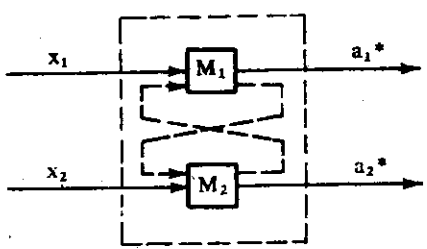


Figure 6.

Cross connection and parallel connection

Given these definitions we actually could consider a combination of both, e.g. serial-parallel decompositions, and in terms of applications these prove to be the most interesting ones.[2a]) We neglect here some more complicated versions of decompositions, which are not loop-free,[3]) for example those known as

cross decompositions as shown in Fig. 6. These cross decompositions are usually handled in connection with abstract network systems. However, under some restrictive circumstances we could achieve the same effect by an appropriate serial-parallel decomposition without loops. We only need to consider an appropriate restructuring of the machine exhibited in Fig. 6. This is illustrated in Fig. 7.

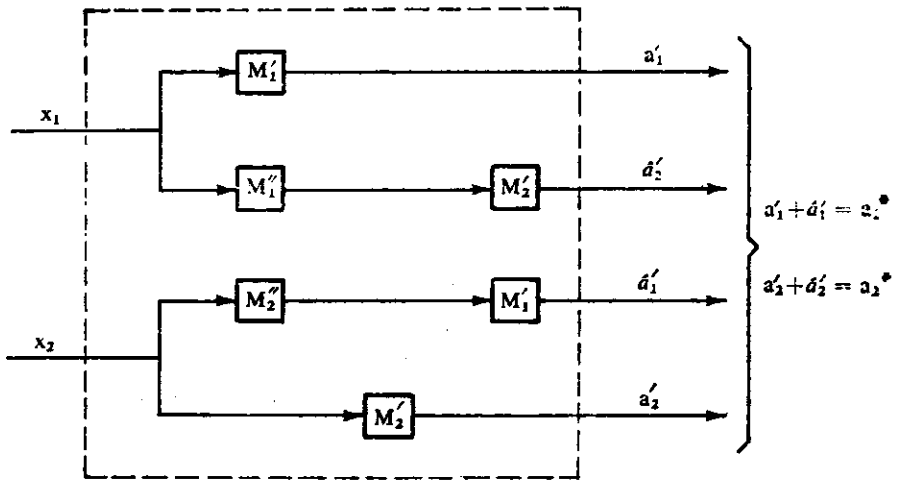


Figure 7. Serial-parallel restructuring of two cross-connected parallel machines.

Care must be taken of the operations $\oplus$ and $\otimes$, for example, distributivity does not hold for both, in particular, even commutativity does not hold for $\oplus$. One can easily check the validity of permissible operations by drawing machine diagrams and finding the corresponding state and output representations.[4]) A somewhat stronger form of decomposition which essentially could be treated within the same mathematical frame work has become known as *cascade decomposition*.

A simple illustration of a *cascade machine* is this:

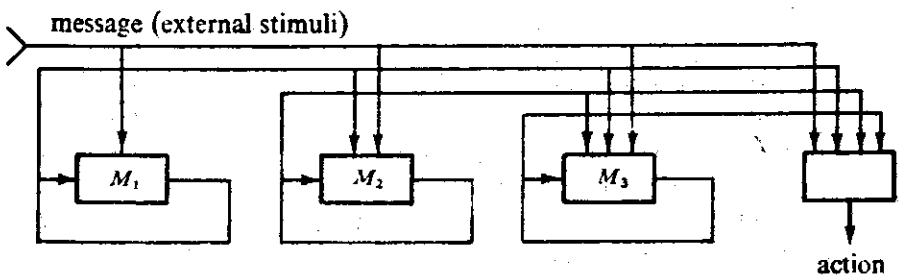


Figure 8. Cascade machine.

There are messages (external stimuli) affecting all component machines, however, every machine produces its own messages affecting all other machines on line. As one realizes, the information process in such cascade form tends to be increasingly complex, the highest degree of complexity is obtained by M_3. This will lead to a peculiar resolution principle with which we are dealing later.

Which kind of decomposition one would like to choose for an organization depends on various factors, certainly on the economic environment it faces, on nature and extent of its performance, and last but not least there is some [illegible] important consideration. One could argue that decomposing an organi[illegible] into information, decision and payoff machine is a rather artificial pro[illegible] since we all know that often parts of an organizational unit do all this simultaneously. However, besides emphasizing the point that we are not primarily interested in what actual organizations do, the crucial point in the attempt to construct an organizational unit is how much information the system as a whole needs in order to select the 'right' actions and to produce a 'desirable result'. The second question, equally important, is how to disseminate information among organizational units in order to achieve this result. A possible third question is that of cooperation or even competition between these units so that an optimal use is made in the allocation of 'informational resources (resource compatibility)[5]).

An organization as a machine - informs, computes, remembers, acts and reveals its state behavior and output structure. For doing all this the informa[illegible] may well be substantial. In fact, it might turn out that [illegible] machine only works on the basis of highly aggregated messages (data) [illegible] will constitutes a considerable loss of information. It is therefore essential to know whether a particular machine preserves the original informational content supplied by the messages. The related question derived from the informational requirements to operate a machine finds its counterpart in the economics of centralization and decentralization. Often it has been recognized that an economic system shows a poor performance because the computational capabilities do not match the informational requirements given the amount of input data and number of states in the system. In an intuitive sense one could argue that computational complexity[6]) of a machine (organization) is related to its 'information technology'. This notion has been introduced by C. B. McGuire [6] by emphasizing the cost structure associated to the technology. We here use the notion in a genuinely technological way, bound to the machine structure, here called an 'information-handling equipment', which is analogous to the number of storage components, versatility of operations etc. in a com[illegible] We introduce the notion to describe precisely situations which are linked [illegible] machines. Cost considerations will enter the picture later, recall that we first have to solve problem (1) in the previous section, before proceeding to problem (2).

Before developing the structure of information technology we need to state some preliminary definitions.

Definition: Let M be a finite machine. A machine $M' = \langle X', Y', Z', \lambda', \delta' \rangle$ is said to be a submachine of $M = \langle X, Y, Z, \lambda, \delta \rangle$ if X', Y', Z' are subsets of X, Y, Z respectively and if

$\lambda' : X' \times Z' \to Z'$, λ' and λ identical on $X' \times Z'$;
$\delta' : X' \times Z' \to Y'$, δ' and δ identical on $X' \times Z'$.

Definition: Let $\mathscr{X}$ denote the set of all finite non-null sequences (the alphabet) of X in a machine M, denote by $\bar{x} = x_1, x_2, \ldots, x_n$ an element of $\mathscr{X}$. Define $\bar{\delta} : \mathscr{X} \times Z \to Z$. Then two machines M_1 and M_2 with identical input and output alphabets are said to be equivalent (state equivalent) iff $\bar{\delta}_1(z, \bar{x}) = \bar{\delta}_2(z, \bar{x})$ for all $z \in Z$, and $\bar{x} \in \mathscr{X}$. In case $M_1 = M_2$ this is trivially true. Clearly, two machines M_1 and M_2 with the same input and output alphabet and state set are equivalent iff $M_{1z} = M_{2z}$ for all $z \in Z$, i.e. iff at every state one machine produces the same as the other machine.

Definition: A machine M is reduced to a machine M'; iff $z \in Z$ is equivalent to $z' \in Z'$ implies $z = z'$.

Usually a reduced machine has fewer states, in fact, it can be made unique in the sense that it has the smallest number of states since every other reduced machine with the same number of states must be isomorphic to it.

Definition: A machine M' is a homomorphic image of machine M if there exists a homomorphism $h = (h_1, h_2, h_3)$ such that $h_1 : Z \to Z'$, $h_2 : X \to X'$, $h_3 : Y \to Y'$ are 'onto' mappings and

$$h_1[\lambda(z, x)] = \lambda'[h_1(z), h_2(x)],$$
$$h_3[\delta(z, x)] = \delta'[h_1(z), h_2(x)].$$

That is to say M' is homomorphic to M if every state and output configuration in M has a corresponding configuration in M'. Likewise, we call $h = (h_1, h_2, h_3)$ an isomorphism if every mapping h_1, h_2, h_3 is one-to-one.

Definition: (Realization) If M' is a homomorphic image of M, then by using the notion of homomorphism M can be used to realize (imitate) M'. In fact, this homomorphism is an assignment of M into M', consisting of mappings

$h_1 : Z \to$ nonempty subsets of Z',
$h_2 : X \to X'$, $h_3 : Y \to Y'$ satisfying the relations
$\lambda'[h_1(z), h_2(x)] \subseteq h_1[\lambda(z, x)]$ and
$h_3[\delta(z', h_2(x))] = \delta(z, x)$.

(Likewise if M is a homomorphic image of M'.)

The homomorphism concept between machines proves to be very helpful, for given the presumption of an operation preserving mapping between M and M', we could realize a given machine M (or a reduced version of it) if it is a homomorphic image of M', for example by placing a combinational circuit in front and back of M' (see Fig. 9).

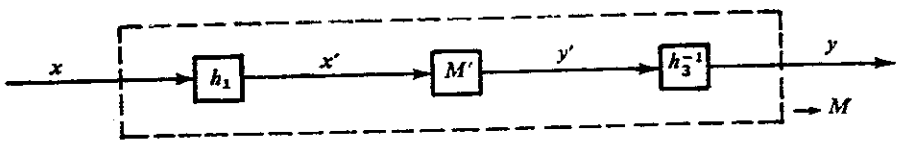


Figure 9. Realization of M by M' via combinational circuit.

In this case M is state homomorphic to M'.

Hence M' would perform subcomputations for M. Now realization becomes important in case of realizing a machine by various types of decompositions. In fact, we could look at state partitions of a particular machine, each partition consisting of several blocks of states. Now given a machine M we may consider a state partition of M, say π, which induces π-images of M, say M_π. M_π could be thought of as performing subcomputations for M depending on which block of π contains the state of M. Looking at the set of all partitions of a machine M' it can be verified that this set forms a lattice under the natural partition ordering. The partial order in this lattice is a comparative relation on the fineness or coarseness of the underlying partition. It also permits interpretation as a relation of comparative information as suggested in another context by myself [1]. The very essence of *information technology* lies in the state decomposition (partition) of machines and in the information structure revealed by the partitioning. The lattice reflects the information structure of all M_π machines, possibly in serial-parallel connection, which realize the original machine M. I call this the *information technology of* all M_π machines realizing M.

Example: Let us give a simple example where π-images of a machine M perform subcomputations which in parallel connection realize completely M. Let $Z = \{1, 2, \ldots, 6\}$, let $\pi_1 = \{\langle 1, 2, 3\rangle, \langle 4, 5, 6\rangle\}$ and $\pi_2 = \{\langle 1, 6\rangle, \langle 2,5\rangle, \langle 3, 4\rangle\}$ be two partitions of Z, hence define the image machines by M_{π_1} and

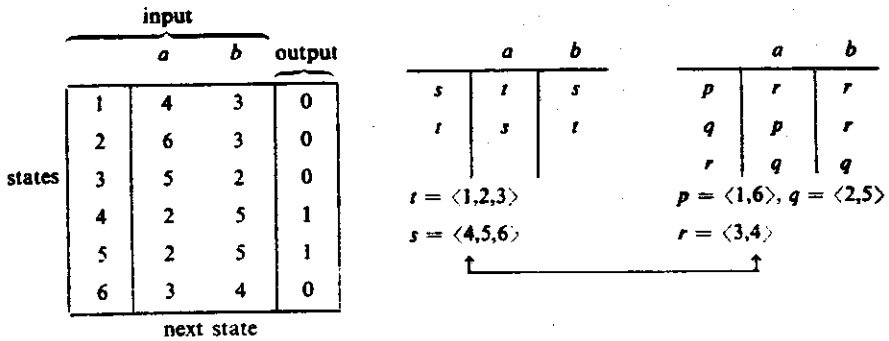


Figure 10.

Representation of M, M_{π_1} and M_{π_2}. Circled number is a unique state given input b, realized uniquely by $M_{\pi_1} \otimes M_{\pi_2}$ in block s given b and block r given b, respectively.

As can easily be seen every block of π_1 has exactly one state of M in common with every block of π_2. Hence the states of M_{π_1} and M_{π_2}, when being operated jointly, uniquely determine a state of M.

We observe that if π_1 and π_2 are partitions of (the states of) M, then also $\pi_1 \cdot \pi_2$ and $\pi_1 + \pi_2$ form partitions of M and the binary operations '$\cdot$' and '$+$' determine the 'inf' (g.l.b.) and 'sup' (l.u.b.) respectively, hence satisfy the definition of a lattice. Let P be the set of all possible partitions of M, with $\{\langle Z\rangle\}$ being the *unit* partition and $\{\langle 1, 2, \ldots, n\rangle\}$ the *null* partition. Then P forms a partition lattice and the g.l.b. $\prod_1^n(\pi_i) = \pi_1 \cdot \pi_2 \cdot \ldots \pi_n$ forms the coarsest among all finest partitions in P, likewise the l.u.b. $\sum_1^n(\pi_i) = \pi_1 + \pi_2 + \ldots + \pi_n$ forms the finest among all coarsest partitions in P. Since P is a lattice it is perfectly legitimate to conceive the operations '$\cdot$' and '$+$' or the generalized operations '$\prod$' and '$\sum$' as partial algebraic operations (see[1]). And the partial order is regained iff a l.u.b. (π_1, π_2) and a g.l.b. (π_1, π_2) exists in P for any $\pi_1, \pi_2 \in P$. Viewing the lattice of partitions as 'information technology' we would like to give '$\leqq$' the meaning of 'not more informative than'. Then in the lattice of partitions, partitions are only partially ordered according to their information content (induced by the state behavior of machines). A machine independent approach suggesting this form of interpretation has been proposed by the author [1]. A general requirement, not always met by structuring a machine, is that of an *output consistent* partition.

Definition: A partition π on the state Z of a machine M is output consistent iff $z \equiv z'$ given π (i.e. z and z' belong to the same block of π) implies $\delta(z, x) = \delta(z', x)$ for all $x \in X$.

Since sometimes the lattice of partitions of Z does not fulfill this requirement

to the associated machine one might find – as an interesting counterpart – that a possible lack of output consistent partitions reflects redundancy of state information, hence by performing computations for machine M it would be sufficient to confine the computational process to the realization of a reduced machine M_R. M_R can be constructed (or induced) by a homomorphism between the original machine M and M_R. Let then M_π and M_{π_R} be those machines that compute M and M_R respectively. Then M_π and M_{π_R} are trivially equivalent. This property certainly has a meaningful interpretation in organization theory. Output – consistency, in fact, is a desirable subproperty of the substitution property (S.P.) of machines.

Definition: A partition π on Z has the *S.P.* iff $z \equiv z'$ given $\pi \Rightarrow \lambda(z, x) = \lambda(z', x)$ given π.

This property actually ensures that if some M_π could perform subcomputations for M then for any given block B_π in π we could find a smaller block B'_π contained in B_π where for every input the state transition function acting on the smaller block generates only states in the larger block, i.e. there is a unique block to block transformation on π.

One technical problem might arise in the case of realizing a machine by a sequence of M_{π_i} machines serially connected. For example, if M_{π_1} is the first machine in line doing subcomputations for M, then we would have to know about those remaining states which still have to be computed in order to realize M. This is necessary to know what kind of M_{π_2} machine, say, is required to do supplementary subcomputations. Now if we could think of an organization to achieve a certain performance standard within some time limit (in terms of computational, not historic time), one has a fairly accurate vision which states have to be computed at various instances of time. Hence this gives some hint on answering the question which information technology could be used for the realization of M by serially connected M_{π_i} machines. This problem is rather deep and we will deal with it next in a more general way.

Adopting the idea that we can effectively compute a machine by various kinds of compositions of its π_i – images M_{π_i}, we would be basically interested in the following

Problem: Given any π-partition of a machine, could we find another π'-partition which fits π in an appropriate way?

We call such a pair (π, π') *complete* if it exists and constitutes the entire information technology needed to realize M. This problem can be given different kinds of interpretation but to what it really amounts to is to determine clearly what kind of complementary information π' is needed for machine $M_{\pi'}$, in order

to compute jointly with M_π the states and possibly outputs of the original machine M. More generally, we could consider the minimal partition

$$\pi_* = \prod(\pi_i : (\pi, \pi_i) \text{ is complete w.r.t. } M)$$

and a maximal partition

$$\pi^* = \sum(\pi_i : (\pi, \pi_i) \text{ is complete w.r.t. } M).$$

In the first case π_* describes the largest amount of information (given the partition π) necessary to compute the next state(s) of M for all π_i finer than π_*. In the latter case π^* represents the least amount of information (given π) to compute the next state(s) of M for all π_i coarser than π^*.

Example: Given a partition $\pi_1 = \{\langle 1, 2, \rangle, \langle 3, 4\rangle, \langle 5\rangle\}$, then compute all possible states onto which all blocks of π_1 are mapped. Assume they are given by the sets $\{4, 5\}$, $\{1, 4\}$, $\{2, 3\}$, then $\pi_* = \{\langle 1, 4, 5\rangle, \langle 2, 3\rangle\}$.

We already know that the set of partitions forms a lattice L under the natural partition ordering, the set of partition pairs will be a subset $\mathscr{P} \subseteq L_1 \times L_2$. We call $\mathscr{P}$ the *pair algebra*[7]) satisfying a closure, completeness and boundedness property e.g.

a) (π_i, π_j) and (π'_i, π'_j) in $\mathscr{P}$ imply that

$$\prod_i \{(\pi_i, \pi_j), (\pi'_i, \pi'_j)\} \text{ and } \sum_i \{(\pi_i, \pi_j), (\pi'_i, \pi'_j)\} \text{ are in } \mathscr{P}.$$

b) For any π in L_1 and π' in L_2, the trivial partitions $(0, \pi')$ and (π, I) are in $\mathscr{P}$.

c) For some $\pi \in L$ there exists $\pi_* = (\pi, \pi')$ and $\pi^* = (\pi, \pi'')$ constituting *g.l.b.*'s and *l.u.b.*'s in $\mathscr{P}$, respectively.

Obviously, $\mathscr{P}$ is again a lattice under the natural partition ordering $\leqq$ since $(\pi_1, \pi_2) \leqq (\pi'_1, \pi'_2)$ in $L_1 \times L_2$ is equivalent to $\pi_1 \leqq \pi'_1$ in L_1 and $\pi_2 \leqq \pi'_2$ in L_2, and $\mathscr{P}$ has the zero element $(0, 0)$ and the unit element (I, I).

In some sense the lattice L_1 describes the ordering of information about the machine (we have got) whereas L_2 describes the ordering of information to which the previous information can be transformed by M. Hence M is considered to be a transformation machine which already suggests that any adjustment process, to be defined later, acts as a 'transformation walk' on the lattice of partition pairs.

In many cases it would be sufficient to start out with a subset (not necessarily sublattice) $\mathscr{P}_0$ of $\mathscr{P}$ containing all initial partition pairs. If additional information is needed to compute the next state(s) of M then this information can be obtained by modifying $\mathscr{P}_0$ in an algorithmic fashion, i.e. by refining the first component and or coarsening the second component of the pair. In an organizational context this procedure is very much like the process of interchange of messages between various subunits.

Since the lattice of partition pairs is uniquely associated to the machine structure it is possible to reveal the informational skeleton of the machine in this way. In particular, given a machine M it is possible by an appropriate decomposition to compute the next states and outputs by π-images of the machine obtained by partition analysis. One question then naturally arises: which information obtained by partitioning the states of the original machine is sufficient to compute the future states of this machine? The following list is not claimed to be exhaustive but it provides the main steps to be checked:

Algorithm:

a) Start with a certain partition based on present information and past history.
b) Look for future states which have to be computed.
c) Look for that π' that requires the minimal amount of information in terms of the partition ordering.
d) If π' does not fit π, look for some π'' which is finer or coarser than π', or take concatenations $\pi_1 \cdot \pi_2 \cdot \ldots \cdot \pi_n$ (in case of serial decomposition) or $\pi_1 + \pi_2 + \ldots + \pi_n$ (in case of parallel decomposition).
e) Compute the partition pair and determine its locus in the lattice of partition pairs (pair algebra).
f) Determine (technological) informational efficiency by the minimal dimension of the sublattice in $\mathscr{P}$ given by the computed partition pair (π, π') as illustrated in Fig. 11.

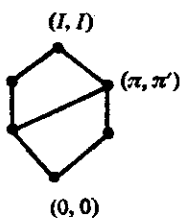


Figure 11. Dimension of sublattice reflects highest informational efficiency or minimal information needed to realize M.

We could define a dimension function as a function
D: $\mathscr{P} \to [0, 1]$ with the following properties:

(i) $0 \leqq D[(\pi, \pi')] \leqq 1$ for every $(\pi, \pi') \in \mathscr{P}$, in particular $D[(0, 0)] = 0$, $D[(I, I)] = 1$.
(ii) If $(\pi, \pi') > (0, 0)$, then $D[(\pi, \pi')] > 0$.
(iii) Let $\perp$ denote an algebraic independence relation, if $\perp\{(\pi_1, \pi_1'), \ldots, (\pi_n, \pi_n')\}$, then $D(\bigcup_1^n(\pi_i, \pi_i')) = \sum_1^n(D(\pi_i, \pi_i'))$

(iv) $D[(\pi_1, \pi_1') \cup (\pi_2, \pi_2')] + D[(\pi_1, \pi_1') \cap (\pi_2, \pi_2')] = D[(\pi_1, \pi_1')] + D[(\pi_2, \pi_2')]$.
(v) D is order-preserving on $\mathscr{P}$, D can be shown to be unique.[8])

The algorithm then contains the following instruction. Choose that (π, π') in $\mathscr{P}$ which has minimal dimension in terms of D. Of course, in case $\mathscr{P}$ represents a metric lattice D would be identical to a metric on $\mathscr{P}$. Again the economic analogue of this procedure can be easily presented, it relates to the problem of how much and what kind of informational decentralization is necessary (and not whether it is necessary at all) to resolve the computational burden brought upon by a highly complex organization. On the other hand, given a set, say of parallel connected component machines $M_{\pi_1}, M_{\pi_2}, \ldots, M_{\pi_n}$ realizing M, could we find a simpler set of component machines which will do the job as well. This relates to the question of *information redundancy* and amounts to finding the smallest sublattice within the lattice of partition pairs, given the performance standard of the original machine, where informational efficiency could be measured by the dimension of the sublattice. The task to avoid information redundancy can be approached by an algorithmic search procedure substituting M_π by $M_{\pi'}$, in case both are equivalent machines (in the precise meaning defined above) but where π' is finer than π so that $M_{\pi'}$ requires less information than M_π. Such an algorithmic procedure finds its counterpart in a policy aiming at the change of the organizational design (organizational change).

We have to mention at least one technical difficulty arising in the case of redundant information. Suppose that partitions π_1 and π_2 are sufficient to realize M. Then the sum $\pi_1 + \pi_2$ represents a redundant computation which should be factored out, but in some instances it might occur that factoring out will cost additional memory. Thus, in general, when dealing with the problem of factoring out information redundancy one should only select partitions which do not enlarge the memory requirements. Here we have dealt only with with the construction of the information technology involving the partitioning of the state set of a machine. We could, however, think about partitioning in a broader sense affecting the input, output and state set simultaneously. Given a machine M, we then say a $X-Z$ partition determines an 'input-state' set, accordingly, a $Z-Y$ partition determines a 'state-output' set, both sets form pair algebras. In general, $M = \langle X, Y, Z, \lambda, \delta \rangle$ could be replaced by the partition machine $M' = \langle X_\tau, Y_\omega, Z_\pi, \lambda_{\tau\pi}, \delta_{\tau\omega} \rangle$ τ, ω, π denote partitions, induced by a π-partition on Z of M. In fact, M' is a homomorphic image of M where $h_1 : X \to X_\tau$, $h_2 : Y \to Y_\omega$ and $h_3 : Z \to Z_\pi$, and M may be realized by a serial-parallel decomposition of M'. In all discussions concerning performance of economic systems (Reiter [7]) the question of performance and size of message space arises. It is generally acknowledged that there exists some kind of trade-

off between both, characterizing an efficiency frontier of allocating information. In particular, a legitimate question is what is the minimal size of the message space still able to sustain a certain performance standard. Nothing is known about the absolute size of a message space but something could be said about the ordering of message spaces given different economic environments where the competitive process is the most natural to start with because of its Pareto optimal property. The efficiency question can be translated appropriately in our framework. Now translated in the language of machine theory we are interested in finding the minimal information technology sustaining the realization of a machine. Whereas the traditional approach actually studies the size of a message space (or information-carrying capacity) in terms of topological properties we believe that this is rather unnatural from a machine-theory viewpoint where information technology (here message-transferring technology) really has no algebraic counterpart.

Although principally, we could solve the technological aspect of informational efficiency we still have to take care of the economic problem of finding an information technology with minimal costs. Here machine theory doesn't provide tools for the direct solution of this problem. The reason is that engineers and computer scientists are not so much worried about monetary costs of operating components or pieces of hardware, all that they are worried about is the feasibility of the design with the performance standards set out in advance. However, they are much concerned about problems like computational complexity (measured in terms of number of diodes used in the realization), real-time computation, and algorithmic efficiency of a machine. These are important parameters of 'computational costs' and they have some relevance for economic considerations, too. Nevertheless, we wish to treat costs associated to the information technology in a more unified analytical way. If we could find some link between computational complexity and costs of information we will be able to speak intelligently in economic terms about the optimal size of a machine. Now it seems intuitively reasonable to argue that the cost of operating a machine is associated to the information technology necessary to realize the machine, or more explicitly, is associated to a certain partition pair satisfying this requirement. Hence, we would like to associate the cost function to the lattice of partition pairs mapping the state set generated by the partitions into an appropriately defined vector space, the cost space. Unfortunately, we do not know much about the properties of this function, except, perhaps, that it is monotone-increasing. Informally, this means that handling more information is more expensive, or that handling more complex messages causes higher information costs. However, this implicitly assumes that information handling equipment is completely divisible and equally

of computations. On the other hand we know that more complex computations could be handled more efficiently by more advanced technology which introduced might even decrease total unit costs of information-processing. Hence, there is no uniform pattern regarding cost function specifications of information technology and this basically requires a broad range of empirical investigations on that matter. The problems of specification of cost functions for a certain information technology often appear in discussions on advantages or disadvantages of decentralized or centralized economic organizations. In general, however, if we consider large organizations it is safe to argue that costs of information processing are roughly proportional to 'computational complexity' of the machines which is increasing with the dimension of the lattice of partition pairs measured from its zero element. This brings us closer to the concern of computer scientists representing a measure of computational complexity by costs of computation. The problem of computational complexity will arise later in another context.

One possibility to deal analytically with the problem of costs of information processing should be pursued here explicitly in some general form.

Definition: A partially ordered vector space is a cost space C if

C is endowed with a tolerance relation R, such that for each $c \in C$ there exist a, $b \in C$ such that $a < c < b$ and $a < c' < b$ implies $(c, c') \in R$.

In fact, we usually think of a cost space as the reals, under a tolerance of the form $(c, c') \in R \leftrightarrow |c > c'| < \varepsilon$ for some fixed $\varepsilon > 0$.

Now, given a machine $M = \langle X, Y, Z, \lambda, \delta \rangle$ and its homomorphic image $M' = \langle X_\tau, Y_\omega, Z_\pi, \lambda_{\tau\pi}, \delta_{\tau\omega} \rangle$, and a cost space C associated to Z_π, a cost function for M' is a function $\phi : X_\tau \times Z_\pi \to C$ with representation $\phi(x_\tau, z_\pi) = \sum_{\pi \in P} \phi(x_\tau, z_\pi)$. We could then formulate an optimal control problem in a tentative way.

Problem: Let z_0 and z_1 be two states of Z_π and $Z_{\pi'}$ respectively, called the initial and the terminal state. We say that $x_\tau = (x_1, \ldots, x_n) \in X$ transfers M' from z_0 to z_1 if $\lambda(z_0, x_\tau) = z_1$ for all x_τ, whereby $\lambda(z_0, x_1, \ldots, x_k) \neq z_1$ if $k < n$. Among all such x_τ in X_τ find that sequence $x_{\pi_1}, \ldots, x_{\pi_n}$ for which $\phi(x_\tau, z_\pi)$ is a minimum.

Let then $M' = \langle X_\tau, Y_\omega, Z_\pi \times C, \lambda_{\tau\pi}, \delta_{\tau\omega} \rangle$ be the machine with cost function ϕ and cost space C. We define the machine $(M', \phi) = \langle X_\tau, Y_\omega, Z_\pi \times C, \lambda^c_{\tau\pi}, \delta^c_{\tau\omega} \rangle$ by $\lambda^c_{\tau\pi}(z_\pi, c, x_\tau) = (\lambda_{\tau\pi}(z_\pi, x_\tau), c + \phi(z_\pi, x_\tau))$, and $\delta^c_{\tau\omega}(z_\pi, c, x) = \delta_{\tau\omega}(z_\pi, x)$.

3. Cascade decomposition of organizations

In this section we basically show that the main ideas and concepts, valid for serial-parallel decompositions, apply as well to cascade decompositions – with some modifications. Looking at the cascade machine of Fig. 8 we realize that the information technology of M_3 'covers' that of M_2 and that of M_2 'covers' that of M_1. In general, this cover property of information technologies revealed by component machines is used to prove a decomposition theorem for cascade machines.

1) First we have to make sure that a realization of a machine M by cascade decomposition really exists, that is we have to verify that a nested sequence of partitions (constituting the information technology of cascade machines) can be constructed.
2) Suppose we could find another information technology for M and M could be realized by a different cascade decomposition which is finer than the former (obviously it cannot be coarser). Then we require that the latter partition is a nested sequence of *preserved* partitions which keeps most of the information of the former partition. This can be made clear by the following construction

$$\begin{array}{ccc} Z & \xrightarrow{h} & Z \\ \downarrow & & \downarrow \\ Z_J & \longrightarrow & Z_K \end{array} \qquad K, J \text{ index sets } K \subseteq J,$$

and this diagram commutes, i.e. $Z \to Z_J \to Z_K = Z \xrightarrow{h} Z \to Z_K$. Now this property gives us a possible criterion to find that minimal partition which still yields a nested sequence of preserved partitions. It tells us something about availabilities of different information technologies preserving the capabilities of a given machine.

3) Suppose a machine M is given permitting realization by cascade decompositions. We want to construct a machine M' such that M' is a homomorphic image of M.
Analogously we consider the following diagram to commute.

$$\begin{array}{ccc} Z' & \longrightarrow & Z' \\ {\scriptstyle g}\uparrow & & \uparrow{\scriptstyle g} \\ Z & \xrightarrow{h} & Z \\ \downarrow & & \downarrow \end{array}$$

With the interpretation we proceed according to step 2), except that the homomorphic image of a machine could be considered as a redesigned machine.

When we consider (at least partial) dependence between various states obtained by a cascade machine it is appropriate to consider a nested sequence of covers instead of partitions. In distinguishing covers from partitions we note that partitions contain mutually exclusive blocks whereas covers don't. We then call a cover C_1 coarser than a cover C_2 (C_2 finer than C_1) iff each block of C_2 is a subset of some block of C_1.

The definition of a cover lends itself to the consideration of set systems in the sense of Hartmanis & Stearns [2]. This can easily be seen from the definition of a set system:

Definition: A class of distinct (not mutually exclusive) sets $\mathscr{S} = \{S_i\}$ of the set S is a set system if

(i) $\cup S_i = S$,

(ii) $S_i \subset S_j \Rightarrow S_i = S_j, i \neq j$.

Hence every element of S belongs to at least one subset (block) of $\mathscr{S}$, and no block properly contains another block although blocks may be overlapping. If blocks in $\mathscr{S}$ are increasing in size then we could identify blocks as covers. Covers can be treated in a lattice-theoretic context and lend themselves to considerations of information technologies via pair algebras. The same techniques go through in this respect. The choice of the information technology could be constrained by – what is known as – the computing power of the machine. In case of cascade decomposition of the machine the informational process might get too complex to be compatible with the computational abilities of some cascade machines. This would put a 'technological' limit on the performance of the machine with which the organizational designer as well as the computer engineer has to cope with. One way out of this difficulty would lead to a restructuring of the information technology, e.g. by constructing a different lattice of partitions or covers. In other words, we would try to refine the information technology in the sense that we could break a given partition into finer 'pieces' (i.e. more blocks) which would reduce the computational burden of the individual units.

4. Adjustment processes

The current status of sequential machine theory suggests three different, though related conceptualizations of finite state sequential machines, e.g.

1) a structural description in terms of machine language as that given at the

2) an abstract finite semigroup S, a complete algebraization of 1)
3) a transformation semigroup (Z, S) where S is a set of transformations acting on Z describing state transitions on Z induced by input sequences.

We find it most useful to adopt the last approach for it leads naturally to investigations of dynamic properties of machine behavior. This yields a new interpretation of adjustment processes in organizations. Given the information technology and the lattice of partition pairs one could describe a sequential machine as a set of mappings of the set of states into itself where each mapping corresponds to an input. If we have an input sequence then a composition of mappings corresponds to this input sequence. In general, these mappings form a finite semigroup of transformations on the set of states of the machine.[9])

Example: The concept of the semigroup of transformations is very natural for various branches of sciences, and certainly pertains to dynamic processes of economic or social systems. Let $\mathcal{E}$ be an economy which by generating messages occurs in various states $s, s', s'', \ldots$ according to certain actions of its agents interacting with each other. Suppose the system is in some state s then as a result of the aggregate actions of the agents it will be 'transformed' to a new state s', say (which may of course coincide with the original s if the given state is not affected by the actions). Thus every action in $\mathcal{E}$ is simply a transformation in the set of states of the system, and a sequential machine forms such an appropriate system. Consider now that actions are sequentially produced by certain activities of the agents, then sequential actions could be concatenated to produce new actions. Then obviously the transformation produced by the last action (in a sequence) is, so to speak, conditioned on its past history, and forms the product of subsequent transformations corresponding to successive actions.

In this way the totality of the actions in the economic system, being closed with respect to successive applications is naturally a semigroup of transformations of the set of all states of the system under consideration.

Hence, it is simply a matter of taste whether we regard the process of transformation in a system as a machine (and so describe it explicitly in machine-theoretic language) or whether we consider it, more abstractly, as a semigroup of transformations of a set of states. Since to every machine structure there is a corresponding semigroup structure, partitioning of a machine involves a decomposition of semigroups. Both descriptions are formally equivalent, although the first seems to be more appropriate for modelling an organizational form, whereas the second gives more insight into the algebraic and computational structure of a machine, in particular, in connection with finding solutions via functional equations. The semigroup of transformations can be understood as the 'computational capability' of a machine to transform a past history into

future states and may be viewed as an adjustment process acting in an organizational design given the information technology with which this design is associated, and given the performance standards. Here again the structural behavior of a machine is reflected by an algebraic concept of a sufficiently general nature. For the simple case of a state machine $\langle Z, X, \lambda \rangle$ the semigroup induced by inputs is the set of input functions

$$x : Z \to Z \text{ for all } x \in X,$$

represented by $(z)x = x(z) = (z, x)$. To put the input function $x(z)$ in the form $(z)x$ is convenient for considering the more general case of an input sequence $x_1, x_2, \ldots$ where the semigroup consists of a concatenation of input functions $x_1 \cdot x_2 \ldots$ associated to Z, hence $(z)x_1 \cdot x_2 \ldots$. This concatenation satisfies the closure and associativity postulate of a semigroup. Depending on the length of the input sequence one could enlarge the state flow table up to the number of possible concatenations of the input functions. Hence, we consider an adjustment process as the behavior of a machine M associated to its semigroup of state transformations for a given information structure (lattice of partition pairs), and we denote it by (M,ϕ) where $\phi = (z)x_1 \cdot x_2 \ldots x_n$. One key notion which comes up in connection with this type of adjustment process is that of *computational capability* containing a slight generalization of the notion of realization by machines. Take again the simplest case of a state machine.

Definition: A state machine $M = \langle Z, X, \lambda \rangle$ has the same computational capability as state machine $M' = \langle Z', X', \lambda' \rangle$ iff there exists an assignment (α, β) such that

a) $\alpha : Z' \to \mathcal{S}$ ($\mathcal{S}$ is the class of partitions into non-empty disjoint subsets of Z)
b) $\beta : X' \to \mathcal{X}$ ($\mathcal{X}$ is the set of sequences over X)
c) $\lambda(z, \beta(x')) \in \alpha(\lambda'(z', x))$ for all $z \in Z$, $z' \in \alpha(z)$, and $x' \in X'$.

The difference to the realization concept is that here β maps inputs into input sequences, and α maps states into subsets of states.

There are structural constraints which limit the possibility of serial decomposition of a machine M. M is called a reset machine iff each input is an identity or constant mapping. For example, if M is realized by a serial decomposition $M_1 \oplus M_2$ and if M has the capability of a two-state reset machine then either M_1 or M_2 have the same capability. These machines, M_1 and M_2, represent *prime* capabilities which cannot be further decomposed. In other words, they are *simple machines* whose semigroups are simple groups and machines which are two-state reset machines. A reset machine actually implies that the organization adopts a stationary (cyclical) pattern, i.e. is not moving along newly generated states.

Let P be some decomposition of M, characterizing its information technology. If S denotes the semigroup of transformations on M, then (P, S) is a transformation group. The group complexity of S is defined as the cardinal number $\#_G(S)$, and the group complexity of M is given by

$$\#_G(M) = \min \{\#_G(\pi) : \pi \in P\}.$$

This complexity measure can be used for measuring the computational complexity of adjustment processes in machines, and in fact, it corresponds to the minimal dimensionality of the lattice of partition pairs. Hence, we get different algebraic measures for complexity of computations in organizations.

5. Conclusions

We have presented a fragment containing several ideas how to design an organization which performs certain tasks. As a starting point we chose the well-developed economic theory of optimal organizations and we attempted to translate some of the key notions into the language of sequential machine theory. The former theory shows several shortcomings which we wish to avoid: First, it does not provide a theory on the design of an organization, hence it does not show the 'architecture of complexity' and the 'economy of construction.' Second, it does not come to grasp with the problem of information decentralization generated by an appropriate information technology. Third, it does not provide means to perform computations in organizations since the analytical framework used does not lend itself to computational experience. The practical aspects of sequential machine theory in the design of organizations would be two-fold. First, given certain performance standards is the design of a particular organization compatible with meeting these standards? If so, does there exist a 'better' design in terms of being more efficient and/or less costly?

Second, given certain performance standards how would you design an organization which meets these standards in a most efficient and/or in a least costly way?

Although both aspects seem to be related, they represent different approaches to the problem. In the former case the 'organizer' is engaged in a check-up of the existing organizational structure and proposes changes if the feasibility requirement is not satisfied. In the latter case, the organizer is actively involved in the design of the organization and is left with considerable leeway to construct the organization subject only to meeting some performance standards. It is this case to which most of the research interest will be directed, hopefully.

In the former case, where existing organizations reveal inefficiencies of various sort, due to rigid structural conditions (bureaucratization), bottlenecks in informational allocation (over-centralization) or informational redundancy (over-democratization), much emphasis should be put on minimizing losses of efficiency caused by bottlenecks and waste. This might not be possible if the 'organizer' simultaneously acts under customary constraints of meeting the performance standards and maintaining the basic organizational structure. Then either he has to drop some standards or to 'revolutionize' the organization – in either case he might get fired. There are quite a few organizations which 'organize' to achieve certain goals and sequentially commit errors in their computations and where further computations at least partially consist of trying to erase such mistakes, to the effect that these computations again may be subject to mistakes, etc. In this case, by maintaining the basic organizational structure as a constraint, the 'organizer' is likely to minimize possible losses of efficiency.

What we tried to show here is that the machine theory approach provides interesting models for organizational design, beyond that one might speculate and test on a sound basis that it will form the core of 'organizational science' comprising many fields and some parts of social science.

In future studies, we have in mind to expose this theory to some experimental work, i.e. to select a reasonable class of 'red-tape organizations' and to check whether, given their alleged performance standards, they are able to meet these standards under present design conditions.

Acknowledgment. This paper has been written during my visit at the department of electrical engineering and computer sciences, University of California, Berkeley, in the academic year 1973/1974. I had stimulating discussions with T. Marschak, J. Marschak, C. B. McGuire, J. Rhodes, R. Radner, L. A. Zadek. None of these persons, of course, may share my views expressed in this paper. I am also indebted to B. van Rootselaar for pointing out several mistakes in the final draft; for all further mistakes I will accept responsibility. Finally I am grateful to the Volkswagen Foundation for comprehensive financial support.

References

1. H. W. Gottinger. Qualitative Information and Comparative Informativeness, *Kybernetik* 13 (1973), 81–94.
2. Hartmanis, J. and R. E. Stearns, *Algebraic Structure Theory of Sequential Machines*, Prentice-Hall: Englewood Cliffs, 1966.
3. L. Hurwicz, Optimality and Informational Efficiency in Resource Allocation Processes, Ch. 3 in: *Math. Methods in the Social Sciences*, Stanford Univ. Press: Stanford, Calif. 1959.
4. Hurwicz, L., On Informationally Decentralized Systems, Ch. 14 in: *Decision and Organization* (R. Radner and C. B. McGuire, eds.) North-Holland: Amsterdam 1972.

5. McGuire, C. B. and T. Marschak, *Design for Organizations* (unpublished notes: University of California, Berkeley 1971).
6. McGuire, C. B., *Information Technology* (unpublished notes: University of California, Berkeley 1972).
7. Reiter, S. and K. Mount, The Informational Size of Message Spaces, *Center for Math. Studies in Economics and Management Science*. Northwestern University, Evanston, Discussion Paper No. 3, 1972.
8. Rhodes, J., *Applications of Automata Theory and Algebra* (unpublished notes: University of California, Berkeley, 1973).

Notes

1) In many instances it is more appropriate to consider a more general definition of a sequential machine and to replace X by a nonempty set of finite sequences over X, denoted by ΣX or $\bar{X}$. A sequential machine is then a mapping $F: \Sigma X \to Y$ and $f(x_1, \ldots, x_n) = y_n$ is the output at time n if x_i is the input at time i for $1 \leqq i \leqq n$. In this case the state set is not explicitly considered, although it is generated by 'realtime computation'. For reasons of considering the 'information technology' given the set of states we will stick to the previous definition, for which J. Rhodes [8] uses the term (sequential) circuit reserving the term 'machine' for the more general definition.

1a) We only consider the environment as an 'input', hence as a fixed part. Beyond this rather narrow viewpoint presented here, it is perfectly legitimate, not only mathematically interesting, to view the environment itself as a machine (variable part). In fact, this problem of machine interaction is pursued by J. Rhodes [8] in most of his applications of automata theory to biology, psychology and psychiatry. The distinction resembles that of decision-theory (games against nature) and game theory proper.

2) This idea has been further elaborated in recent notes by John Rhodes [8]. In these notes certain situations are analyzed, involving quite distinct areas, but all situations involve 'real time computation' where 'machines' respond in real time with its environment just to stay alive. The situation is quite different for Turing machines, where there is no time and no space constraint with limited computability. (Computer scientists speak in the first case of *on-line computing*, in the second of *off-line computing*). These qualifications have to be adjusted to concrete situations. In universities, for example, the organization of research by competent scientists is hard to evaluate in the sequential machine framework. On the other hand, the usual type of work performed by secretaries and administrative assistants, less so on a more professional level can be subject to 'organizing' via sequential machine theory.

2a) One way to increase computational power in the realization of machines is by emphasizing parallel decompositions given some level of serial decomposition. Thus, the computational power (speed) of an organization realized by serial-parallel decomposition can be substantially increased by increasing the number of parallel connections, if possible. As an interesting analogy we mention that the design of high speed computers relies heavily on parallel computations.

3) We do not explicitly consider feedback maps which could be considered as 'two-sided internal stimuli' acting on component machines in the process of realization. Feedback would substantially increase complexity of computations. This is in perfect agreement with on-line computation, and simplifies certain aspects which are at the present stage of greater importance.

4) See Hartmanis and Stearns [2].

5) As indicated before, the question of incentive-compatibility remains open here, at the present stage it suffices to say that incentive-compatibility will appear in simpler form than it does in the conventional economic theory of organizations (see Hurwicz [4]). Here what it really amounts to is that organizational units perform computations in accordance to

what they are expected to achieve under 'real-time computation.' Delay in computations, misspecification of messages, misallocation of funds, resources etc. – these possibilities might jeopardize the process of realizing the original machine and, provided all other structural conditions are met, may basically reflect 'incentive incompatibility.'

We use the concept of computational complexity in a different, but related sense to Rhodes' treatment of complexity who exposes an algebraic theory of complexity for sequential machines: First, complexity is related to the computational capability of a machine, e.g. the more capable a machine is (in terms of input received and output generated) the more it is considered to be complex. Second, complexity of a machine to be realized is (at most) the maximum of complexities of its component machines. Here we are more interested in a narrower concept of complexity related to the structure of information technology.

7) Recall from above that every partition pair constitutes by itself a feasible information technology. Thus every point of the lattice of partition pairs (or pair algebra according to [2]) is itself a lattice, hence we can speak of a lattice of lattices. Instead of considering a pair (π, π'), for simplicity, we could take an n-tuple $(\pi_1, \ldots, \pi_n)$ (corresponding to a sequential process in n-stages) which itself forms a lattice.

8) Furthermore, if D and D' are both dimension functions and exist, then D and D' are uniquely related up to positive linear transformations, hence dimension in a lattice is measurable on an interval scale. We consider the minimal dimension of the lattice as a measure of *computational* or informational efficiency.

9) A mapping of the set Z into itself is called a transformation. We could denote S_z as the set of transformations on the state set Z, and this is a (finite) semigroup with respect to the operation of forming the product transformations under successive application of concatenating inputs (in terms of mappings), hence S_z is a multiplicative set of transformations of the set Z. One could consider a semigroup of transformations as a natural tool for the study of general processes with a wide range of applications. The semigroup property for multistage decision processes such as dynamic programming has already been recognized by R. Bellman (1957). This property is strongly connected with the representation of such processes by functional equations. In fact, if we think of actual computations of a semi-group of transformations they would involve appropriate solutions of functional equations.

HANS W. GOTTINGER

COMPLEXITY AND SOCIAL DECISION RULES

1. Introduction

The notion of 'global rationality' underlying the construction of 'economic man' that is generally accepted at least in normative economics has come increasingly under attack by those who care for more fruitful behavioral assumptions in economic reasoning. This notion is intrinsically related to various optimization programs that have been implemented in economics but that have been found only of limited use in realistic, complex situations. H. A. Simon [8] deserves credit having observed the limitation of global rationality and suggesting a modification of this program by introducing his concept of 'limited rationality'. To a great extent these ideas were carried forward in studying human thought processes where it was found that decision-makers, for purposes of problem-solving, go through several stages of goal formation, a hierarchical representation of goals, super- and subgoals, where at every stage goal attainment rather than optimization is called for. Such programs are motivated by the complexity of problem-solving tasks that are treated successfully by decomposing problem-solving in a sequential way and by associating to every stage of the process the attainment of a subgoal. Goal-oriented behavior, therefore, is non-optimizing behavior and only improvement-related with respect to the attainment of the next goal in a sequence. (G. W. Ernst and A. Newell [2].)

Simon [11] relates a need for revision of the 'economic man' to the limitation of access of information and computational capacities being available to human decision-makers. The computational dimension is probably the most important aspect of characterizing 'limited rationality'. In fact, this point has been brought up in a similar connection by H. Leibenstein [6] where he interprets 'rationality' in terms of 'calculatedness' (computability) and tightness or looseness of calculatedness is supposed to cover the whole spectrum between rationality and limited rationality.

The computational dimension of limited rationality as applied to the social choice process is analyzed here in a more rigorous fashion than

Hans W. Gottinger and Werner Leinfellner (eds.), Decision Theory and Social Ethics, Issues in Social Choice, 251–269.

HANS W. GOTTINGER

has been done before. It turns out that complexity is an essential tool for analyzing constraints on the decision process. Moreover, any axiomatic system of 'limited rationality', yet to be defined, must contain complexity as a primitive notion.

The paper attempts to show how a social decision function can be constructed, by unconventional tools, such that it is compatible with individual decision functions. Complexity enters the construction as the basic limiting factor.

2. Choice processes and complexity

On the level of individual or social choice problems complexity relates to the ability or inability of human beings to make effective choices in a consistent or rational way. In this regard complexity exhibits some kind of uncertainty that cannot be treated properly in terms of probabilities.

One clear indication when complexity enters individual decision-making is given by the inability to prove that a utility function representing preferences or choices does exist. If this proves to be a legitimate question on the level of individual decision-making, it is even more so on a social choice level.[1] F. S. Roberts [9, 127] proposes two ways out of this dilemma:

> ... one approach to the decision-making is to describe a procedure whereby we can modify or redefine or make explicit our preferences in the course of decision-making in order to become more 'rational' (i.e., that such a utility function will exist).

A second approach, somewhat less demanding, is to settle for a utility assignment which best approximates the utility function.

It is doubtful that the first approach leads to a satisfactory solution. Since even if it is possible to teach individuals how to act more rationally than they used to behave, they will never be 'perfect computers' and there is a threshold of complexity beyond which they cannot effectively handle situations, for instance, making choices among many alternatives. Put in a different way, you can try to teach subjects how to make optimal decisions in a simple course of actions, as J. Marschak [7] suggests on the basis of psychological studies on that matter. But still teaching optimality does not cope with the problem

that people simply make mistakes because of complexity or *embarras de richesse* in selecting among many alternatives—in the same way as people may understand simple arithmetical rules but cannot solve complicated arithmetical problems in the large because of time, resource and computational constraints.

The alternative then is that people adopt reasonable behavior strategies (in the sense of being within their 'computational budget') which cope with the intrinsic complexity of (social) choices, e.g., those rules exhibiting non-optimizing behavior.

Regarding the second approach, much of the contribution by measurement theory has been in the direction of weakening preference requirements (for example, Luce's semi-order theory, avoiding indifference, but admitting thresholds).

The weaker assumptions aim at reducing the computational burden of decision-makers, yet they fail to make explicit the complexity bounds in forming decision rules.

Many choice processes in the real world, in contrast to theoretical constructs used by choice theorists, represent essentially *ill-structured* problems to the extent that solutions of these problems are not readily available and they involve an excessive amount of computational power. In general, a problem is considered to be *well-structured* if it satisfies a number of criteria, the most important of which relate to the existence of at least *one* problem space that provides for solvability with the help of a practicable (reasonable) amount of computation or search. Apparently well-structured problems such as theorem-proving and chess playing in artificial intelligence turn out in many instances to be *ill*-structured, given the problem-solving power of problem-solving methods. There seems to be an intrinsic relationship between well- or ill-structuredness of a problem and the threshold of complexity (in von Neumann's sense) below which a system shows a regular, stable and predictable behavior but beyond which often quite different, sometimes counterintuitive modes of behavior can occur. A problem can be well-structured in the small, but ill-structured in the large. According to H. Simon [13] "the difficulty stems from the immense gap between computability *in principle* and *practical* computability in problem spaces as large as those of games like chess". This generally applies to complicated choice processes.

Therefore, the problem of complexity is similar to the problem a chess player faces when searching for a 'satisfactory' strategy in chess.

The social choice problem resembles the choice of strategies in chess-playing to the extent that the decision-maker is involved in a choice problem of combinatorial dimension. To search for all game-theoretically possible alternatives goes far beyond the computational ability of the human being.

One conclusion, therefore appears to be obvious: we have to depart from behavioral hypotheses involving optimizing behavior, as convenient as it might be in mathematical terms, since it does not come to grips with non-trivial choice problems in complex situations. We do not have to leave the grounds of rationality. A rule-of-thumb method may be rational in a restrictive sense, thus we have to view it in terms of 'limited rationality'. Rule-of-thumb methods may be applied for various reasons: either because the individual faces expected costs of computation to be far beyond expected utility of further searches in choice-theoretic behavior or he (she) is faced with an immense mass of alternatives to the effect that he (she) is psychologically outstripped by the ensuing 'complexity of computation'. Chess players tend to choose simpler decision rules. They do not consider all possible strategies and pick up the best, but generate and examine a rather small number, making a choice as soon as they discover one that they regard as satisfactory. According to H. Simon [12], "limits of rationality in chess involve (a) uncertainty about the consequences that would follow from each alternative, (b) incomplete information about the set of alternatives, and (c) complexity preventing the necessary computations from being carried out".

All three properties may be subsumed under a more general concept of complexity in choice-theoretic situations. For example, uncertainty and lack of information may here assume different aspects to what is widely known in statistical decision theory and the economics of uncertainty, e.g., uncertainty resulting from computational incapability when faced with a large number of choice alternatives. These are essentially non-probabilistic situations. Thus, complexity is an important tool for evaluating decision rules, in fact, it may prove instrumental for an axiomatic analysis of 'bounded rationality' which is still lacking.

3. Some formal prerequisites

We present here some formal definitions toward developing a more general theory of complexity for social choice 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 introduced earlier by C. Futia [3], more generally see Gottinger [5].

(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., $A^* = \{(a_1, \ldots, a_n): n \geq 1 \text{ and } a_j \in A\}$. Then we define a sequential machine as a function $f: A^* \to B$ where A is the basic input set, B is the output set and $f(a_1, \ldots, a_n) = b_n$ 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 $(A, B, Z, \lambda, \delta)$, where A and B are defined as above, Z is the (nonempty) set of internal states, $\delta: Z \times A \to B$ is the output function, $\lambda: Z \times A \to Z$ is the next-state function. 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 $f = C_z$.

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:

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

$$C_z(a_1, \ldots, a_n) = C_{\lambda(z,a_1)}(a_2, \ldots, a_n) \quad \text{for} \quad n > 2.$$

(2) Let $f: A^* \to B$ be a machine. Then f^S, *the semigroup of* f, is given by the congruence $\equiv_f$ on A^* where for $t, r \in A^*$, $t \equiv_f r$ if and only if $f(\alpha t \beta) = f(\alpha r \beta)$ for all $\alpha, \beta \in A^* \cup \{1\}$. Then, if $[t]_f$ denotes the equivalence class of the equivalence relation $\equiv_f$ containing t, we have $f^S = \{[t]_f: t \in A^*\}$ and $[t]_f \cdot [r]_f = [tr]_f$ (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 *combinatorial* if and only if each subgroup of S is of order 1.

(4) A *right mapping semigroup* or *right transformation semigroup* 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 $(f \cdot g)(x) = g(f(x))$. For each $x \in X$, $s \in S$, let $xs = (x)s$. Then the following conditions are satisfied:

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

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

(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 two following 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 $t_i \in X_i$, $i = 1, \ldots, n$.

(ii) (kth component action lies in S_k) We require $f_1 \in S_1$, and, for all $k = 2, \ldots, n$ and all $\alpha = (t_{k-1}, \ldots, t_1) \in X_{k-1} \times \ldots \times X_1$, the function $g_\alpha \in F_R(X_k)$ 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 *wreath product* of $(X_n, S_n), \ldots, (X_1, S_1)$, and $(X_n, S_n) w \ldots w (X_1, S_1)$ is the abstract semigroup determined by (X, S).

(6) Let (X, S) and (Y, T) be right mapping semigroups. Then we write $(X, S) \mid (Y, T)$, read (X, S) *divides* (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 $\theta: Y' \to\!\!\to X$ ($\to\!\!\to$ means onto) and an epimorphism $\phi: T' \to\!\!\to S$ such that $\theta(yt) = \theta(y)\phi(t)$ for all $y \in Y'$, $t \in T'$.

(7) (Krohn–Rhodes Decomposition [8]) Let (X, S) be a right mapping semigroup. Then the (group complexity $\#_G(X, S) = \#_G(S)$ is defined to be the smallest nonnegative integer n such that

$$S \mid (Y_n, C_n) w (X_n, G_n) w \ldots w (Y_1, C_1) w (X_1, G_1) w (Y_0, C_0)$$

holds with $G_1, \ldots, G_n$ being finite groups and $C_0, \ldots, C_n$ 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,

a combinatorial semigroup corresponds to a machine that virtually does no computation but rather switches inputs and outputs among various input–output configurations. This property reminds us of information theory when selecting events which have information measure zero. Since these types of machines generate regular patterns which are expected, they do not yield any surprises. Therefore, their behavior does not produce information. Since everybody understands it, it cannot be complex. This result has some immmediate 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 [8]. 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 will lower the complexity at most by one. Now what is the significance of the Kroh–Rhodes theory? It shows us to what extent we can decompose a machine into components that are primitive, irreducible and that the solution depends on the structure of components and the length of computation. Hence, complexity does not depend only on how long a chain of components is, but also on how complicated each component is. Therefore, complexity not only takes into account the total number of computations in a chain (the computational aspect), but it also does so for the inherent complexity of the subsemigroups (submachines) hooked together via the wreath product (the structural aspect). The structural aspect can be heuristically represented by the amount of 'looping' in a computer program that computes S on X. This has been proposed by C. Futia [3] for computing sequential decision or search rules.[2] These are the key features of an algebraic theory of complexity.

4. An example of a decision or search rule

The subsequent example has been adapted as an illustration from a similar search problem presented by Futia [3]. An individual, as a

member of society (or voter) is subsequently confronted in a 'large' market of public goods to choose among different kinds of commodities or services (nuclear energy, missiles, health care, etc.) offered to him for sale by different government agencies at different prices (i.e., tax rates). In order to receive a tax rate quotation (or possibly some other relevant information) from any given agency, the voter must incur some (not necessarily monetary) cost constituting his marginal search cost. The voter's goal is: given a certain bundle of public goods that satisfies his aspiration he wants to search for low tax rates such that his final taxes (plus total search costs) will be kept as low as possible. This problem can be formalized as follows:

Let t_i denote the tax rate quotation of agency i. Let $t=(t_1, \ldots, t_n)$ be the tax structure and suppose $t_i \in [0,1]=I$. Denote by I^n the n-dimensional Cartesian product of I, and define a probability density F on I^n representing the voter's initial belief about which tax rates the agencies are likely to quote. The order of quotations presented to the voter is considered to be irrelevant, thus, for simplicity, it is assumed that F is symmetric, i.e., if p is a permutation of $\{1, 2, \ldots, n\}$ and if $t^p=(t_{p(1)}, \ldots, t_{p(n)})$ then $F(t)=F(t^p)$.

The set-up of this problem enables us to construct a decision rule which prescribes to the voter, for each i, whether to stop searching after receiving i quotations or whether to continue searching on the basis of the i quotations he has received. A decision rule is assumed to be a mapping from a set of observations into a set of actions. In this problem, for each i, let the set of actions be $A=$ {'accept', 'reject'}, and the set of observations be $O_i=I^i$. Then a decision rule is a sequence of functions $D=(D_1, \ldots, D_{n-1})$, where $D_i: O_i \to A$ if $(t_1, \ldots, t_i) \in O_i$, then $D_i(t_1, \ldots, t_i)$ records the voter's decision to either accept the tax rates that have been quoted to him and choose (by vote) the given bundle of public goods presented to him, or to continue searching and reject tax rates $t_1, \ldots, t_i$.

Now it is perfectly legitimate to ask, for this kind of problem, what is the voter's optimal decision rule? This question could be answered by the machinery provided in statistical decision theory to find optimal solutions for search problems (see Gottinger [5a]).

Instead, we are interested here in the basic ill-structuredness of the problem given by the complexity of the decision rule. To this end, on the basis of the previous section, we proceed to associate with every decision rule D a (computer) program f_D which computes D. This

permits us to define the complexity of the program by the amount of 'looping' between subprograms (computational complexity) and the intrinsic complexity of the subprograms (structural complexity). Hence, a sequential machine is used as a metaphor for determining complexity of sequential decision rules. This can be further illustrated by elaborating on the problem above by using the sequential machine framework.

Let A = set of observable tax rates = finite subset of $[0, 1]$. Let $B = \{$'stop', continue to $i+1$, $i = 1, 2, \ldots, n\}$. Then the machine f_D is defined inductively on the length of the input sequence by

$$D_1(t_1) \text{ if } m = 1, \text{ or if}$$
$$f_D(t_1, \ldots, t_{m-1}) = \text{'stop'} \quad \text{or} = D_i(t_{m-i}, \ldots, t_m) \text{ if } f_D(t_1, \ldots, t_{m-1}) = \text{'continue to } i+1\text{'}$$

The computational length and the structural complexity of subsystems that are needed to compute f_D reflects a measure of complexity for f_D (equivalently for the decision rule D). Obviously, optimal is a rule that is generally more complex and more expensive but which may very well be beyond the computational power and sophistication of the voter. Hence the voter, facing an ill-structured problem wants to make it well-structured by seeking a decision rule which matches his computational ability and sophistication.

5. Complexity of Decision Rules

We suppose that the decision-maker identifies alternatives in his choice space and does express preferences between at least two alternatives by simply computing, else he finds alternatives 'undecidable' or 'incomparable' that cannot be computed. Preference statements are therefore translated into computing devices, indifference statements are kept out because of possible vagueness. The decision-maker, represented as a simple finite-state machine, can be decomposed by performing these tasks.[3] In the first case the job to be done, e.g., computing preferences, is achieved by a simple group machine (that is a decision machine acting as a simple group in the mathematical sense). In the second case the activity consists of a combinatorial machine, acting as a 'flip-flop' which does not compute anything.[4]

Realizing a decision-rule therefore means a decomposition of the decision process according to the decomposition of machines into component machines that, when 'hooked' together, (via the wreath product) realize the overall machine. Of course, the complexity of decision rules may vary; a 'sophisticated' decision-maker may activate more simple groups, less flip-flops, or groups that compute faster, more accurately and more reliably. This type of decision-maker will carry more structural complexity in the sense given in the previous section.

A (social) decision rule is a sequential decision rule and as such is considered to be a finite state machine (associated to a finite semigroup), and according to complexity theory it has a finite decomposition. In this regard the results of Krohn–Rhodes complexity theory apply. The idea involved here is to factor a social choice process into parts (components) where the global process is modelled as a transformation semigroup associated to a social decision rule, and the local parts are represented by transformation subsemigroups. The new tools originate from decomposition results in automata theory.

Consider a choice set of finitely many alternatives $X = \{a, b, \ldots, x, y, z\}$ and let $D_i = 1$ iff i prefers x to y, $D_i = 0$ iff i is 'undecided' about x and y, $D_i = -1$ iff i prefers y to x. Let $\mathfrak{D}$ be a nonempty set of decision rules D_i, $\mathfrak{X}$ a nonempty collection of subsets of X, a social decision function (SDF) then is a function $F: \mathfrak{X} \times \mathfrak{D} \to P(X)$, $P(X)$ being the power set. A SDF for individual i is given by $F(\{x, y\}, D_i)$, $x, y \in X$.

Social decision functions are in fact decision machines in the sense that they decide on propositions about accepting or rejecting social states, computing them by discrimination, (preference, non-preference). By doing this, they generate, as outputs, decision rules and induce next states representing changes in preference profiles or configurations. There is good reason to argue that we should leave out indifference statements since they cannot clearly be distinguished from the phenomenon of 'undecidability'. Intransitive indifference arises in situations where a chain of indifferences, each of which seems reasonable, adds up to a sufficiently large difference to yield a definite preference between the first and the last items in the chain. We would like to avoid intransitive indifference, therefore we require the decision machine only to accept preference rather than indifference statements.

In order to construct such a decision machine let us state the following

PROBLEM: Let $X^n = X_1 \times \ldots \times X_n$ be the social choice set when the DM is confronted with a sequence of finitely many social alternatives. Let $A_0 \subseteq A_1 \subseteq \ldots \subseteq A_n$, more precisely given in the time sequence $t_0, t_1, \ldots, t_n$, be those sets of alternatives in which the DM can actually find comparisons (in the sense that he prefers alternatives in these sets and finds himself in a position to compute preferences). Let $\mathscr{A}$ be a nonempty collection of all $A_0, A_1, \ldots, A_n$. Then he constructs selection functions $\rho_0, \rho_1, \ldots, \rho_n$, $\rho_i: X^n \to \mathscr{A}$ such that for all $x_i \in X_i$, $\rho(x_i) \in A_i$. In a way, ρ_i constitutes a reduction mechanism by reducing all possible alternatives with which the DM is confronted to those which are computed as actual choices.[5] It is said that the DM accepts the decision rule $D_i(x_0, \ldots, x_i)$ if $\rho(x_0, \ldots, x_i) \in A_i$, more explicitly, accept $D_0(x_0)$ if $\rho(x_0) \in A_0$, accept $D_1(x_0, x_1)$ if $\rho(x_0, x_1) \in A_1$, etc.

There is an upper bound, representing the complexity bound of the DM, beyond which he is unable to compute his preferences. The upper bound somewhat restricts him in selecting decision rules which are 'beyond his complexity'. Therefore, let $\mathscr{k}(D)$ be the largest integer satisfying the bound such that $A_{\mathscr{k}(D)-1} \subsetneq A_{\mathscr{k}(D)}$. How is the bound to be determined?

In a different context, regarding the complexity of (dynamic) finite-state systems, I distinguish between *design* and *control* complexity.

To quote (cf. Gottinger [5]), under *design* complexity I understand that complexity (number) associated to the transformation semigroup in which full use of the system potential is made. Under *control* complexity I understand that specific complexity (number) that results from computations which keep the entire system or at least part of it under complete control. A *qualitatively* stable decision rule would be a rule for which design and control complexity coincide. However, in most practical cases design complexity will exceed control complexity. Since one cannot assume that the control complexity of an average (unsophisticated) DM can be increased by teaching him how to behave in a rational manner one should pick up designs of decisions rules for which there is a reasonable understanding and control.[6]

Example. In a game of chess the number of all possible strategies to achieve a check-mate corresponds to the design complexity of a chess-playing program. The number of all actual strategies chosen by a particular chess player to achieve success corresponds to his control complexity. Given two chess players both initially endowed with the same knowledge of how to play chess, and if in a sufficiently long

sequence of how to play chess, and if in a sufficiently long sequence of repetitive plays one does better than the other, he exhibits a better understanding of the game, e.g., a higher control complexity.

In a certain way both concepts are naturally associated to 'programs of optimization' and 'programs of satisficing' or bounded rationality), respectively. That is to say, design complexity pertains to that decision rule (which is best in some appropriate sense). In general an optimization principle is involved, which, however, cannot be realized given the limited computational resources of the DM (control complexity).

To which extent this bound can effectively be determined by experiments appears to be a problem in experimental psychology. However, it is possible, at least in principle, to give a set of criteria under which it can be determined whether a DM chooses decision rules violating his bound of complexity.[7] Whenever individuals violate in experiments a set of consistency postulates (such as transitivity), namely those which they have accepted at the very beginning, they will realize that they have committed computational errors. Thus commitment of errors or violating consistency postulates seem to be suitable criteria for determining complexity bounds of computation. In experimental situations, subjects then have to be confronted with various decision rules of a different complicated character and the class of decision rules in which no errors or almost no errors occur constitute those which satisfy the control complexity of the DM.

Those decision rules are called qualitatively stable. Only qualitatively stable decision rules guarantee that social, economic and political processes can be controlled in any effective way by social choice, otherwise the amount of error, misrepresentation of preferences, etc., could easily lead to a destabilization of the social system, and a degree of rationality can no longer be maintained.

6. A CONSTRUCTION OF COMPATIBLE SOCIAL DECISION RULES

Let $P_1, P_2, \ldots$ be sets of computable preference profiles for $i = 1, 2, \ldots$ individuals of the social group achieving a common social decision rule D (matching the preference profile of the social group). Let there be $D_1, D_2, \ldots$ decision rules acting as sequential machines such that D_k computes the preference profile P_k. Then we define the complexity of

the social decision rule D, $\theta(D)$, to be equal to $\min\{\theta(D_j): j = 1, 2, \ldots\}$.

In short, the complexity of a social decision rule is bounded by the minimum complexity of any individual decision rule D_j which is able to generate any individual preference profile matching the preference profile of the social decision rule. We proceed to associate a social decision rule (SDR) D for the social choice problem with a finite semigroup $S(D)$. We could envisage the social choice process as a transformation semigroup $(X, S(D)$ where X is the set of social choice alternatives each individual (in the social group) is searching for, while elements of $S(D)$ will be finite sequences of preference quotations generating the preference profile.

We could define $X = \{*\} \cup A_0 \cup A_1 \cup \ldots \cup A_{n-1}$ with $\cup$ disjoint. this is the set of the DM's choice histories. Then $\rho(x_1, \ldots, x_i) \in A_i$ represents the history of the DM's preference statements who has completed i searches and has made choices over $x_1, \ldots, x_i$.

A DM will stop searching if further searching will violate reasonable consistency criteria. The stop rule of searching is imposed by the complexity bound of the social decision rule. By construction, the complexity of D, $\theta(D)$, is equal to the complexity of $S(D)$, $\theta(S(D))$. Again the complexity of the SDR D is bounded by the minimum complexity of the individual decision rules D_j (finite state machines) which by interacting realize a compatible social decision rule.

The procedure how to generate a computable SDR when all members of the society set up their own individual decision rules can be described as a *sequential game* among the members. If the game has a von Neumann value we agree to say that a compatible SDR has been realized.

For simplicity, let us assume that there are only two members of the society which after having computed their individual decision functions want to find a compatible SDR (which satisfies both).

Assume that the game starts in C_{z_0} with strategy ρ constituting the selection rule of the first member of society, then the circuit $C = (A, B, Z, \lambda, \delta)$ is the preference profile with $\lambda: A \times Z \to Z$ and $\delta: Z \times A \to B$. Let A be the set of social choices that have been made by Player I (and the configuration is revealed to Player II). Then B is the set of resulting social choices of Player II that adjust to the preference profile of Player I. Z is the set of adjusted social choice configurations of the game as they appear to Player I. $\lambda(z, a)$ and $\delta(z, a)$ are

interpreted as follows: if z is the adjusted social choice configuration, as it appears to Player I, and a is the choice which enters as input to Player II, let $z \cdot a$ be the social choice configuration after the choice a is made on the configuration z. Let $D(z \cdot a) = b$ be the decision rule generated by C when the position presented to C is $z \cdot a$. Define $\delta(z, a) = b = D(z \cdot a)$ and furthermore define $\lambda(z, a) = (z \cdot a) \cdot b \equiv (z \cdot a) \cdot D(z \cdot a)$, where z_0 is the initial position. Suppose our SDR can be put in binary form, whenever the 'compute preferences' key is followed we assign 1, otherwise 0.

The latter case will be interpreted as meaning that no consistent preference statement can be made since the number of choices involved is too large. Therefore we have to eliminate redundant choice alternatives. Under these circumstances, we could consider, for at least two players, the construction of a compatible SDR to be equivalent to a game tree with binary outcomes.

Example. In this game each player plays zero or one successively—corresponding to the construction of the decision rule. Let us assume the circuit C is a player who responds to the action of the first player, and the circuit C'. W denotes a win for the player, L denotes a loss for the player. The payoff is $+1$ for W, and -1 for L. Clearly, the von Neumann value for this game is $+1$ for the player who goes second. Assuming C goes second the strategies achieving the von Neumann value $+1$ can be listed (and read out of the game tree), as in Figure 1.

Let $C = (A, B, Z, \lambda, \delta)$ be defined as follows:

$$A = \{0, 1(0, L), (0, W), (1, L), (1, W)\}, \quad B = A,$$

$$Z = \{\phi, a, b, c, d, e, f, \ldots, r, s, t\}.$$

Then $C_{z_0}: A^* \to B$ induces a sequential social decision rule to which there is associated a complexity, the complexity of the transformation semigroup (X, S). The problem is to find a minimal complexity of the transformation semigroup that permits a construction of a social decision rule compatible with the choice behavior of individual members of the society. In view of (a)–(d) we succeed in doing this by finding the string of minimal length, i.e., the decision rule with the minimal complexity. The upper bound for the complexity follows from the

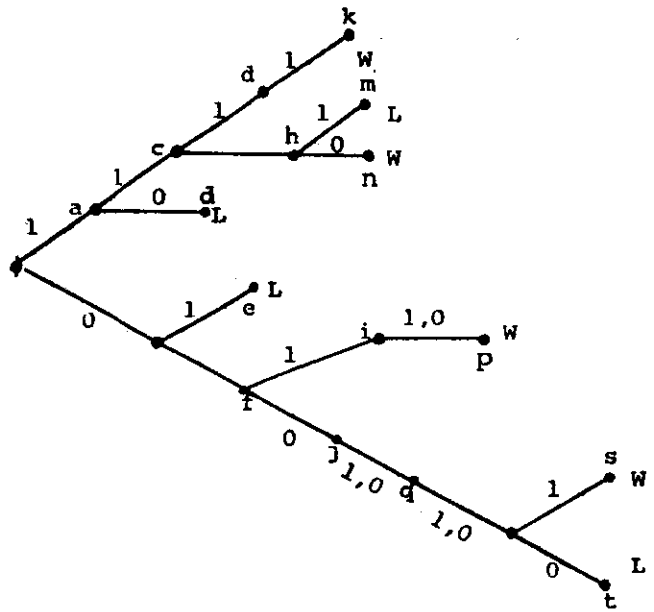


Fig. 1. Game tree with binary outcomes (winning strategies for C)
(a) $(\phi, 1)$, $(a, 1)$, $(c, 1)$, $(g, 1)$, (k, W),
(b) $(\phi, 1)$, $(a, 1)$, $(c, 0)$, $(h, 0)$, (n, W),
(c) $(\phi, 0)$, $(b, 0)$, $(f, 1)$, $(i, (1, 0))$, (p, W),
(d) $(\phi, 0)$, $(b, 0)$, $(f, 0)$, $(j, (1, 0))$, $(q, (1, 0))$, $(v, 1)$, (s, W)

following result:

PROPOSITION (J. Rhodes): Let S be a semigroup of mappings on the finite set X (sequential choice space). Let r be the maximum range (or fixed points) of any idempotent $e = e^2 \in S$. Then $\#_G(S) \leq r - 1$.

Proof. Let I be the ideal generated by the idempotents of S. Then S/I is combinatorial and $I \leq \{f: X \to X: |f(x)| \leq r\} = I_r$. Further I_k, $k = 1, \ldots, n$ are the ideals of $F_R(X)$, the semigroup of all mappings of X into X. Then by the results of Rhodes *et al.* [8] it can be shown that $\#_G(S) = \#_G(I)$ and $\#_G(I_r) \leq r - 1$. q.e.d.

7. Summary and Extension

We have noticed how choice processes could be factored into component subprocesses and how these are associated to properties of

transformation semigroups. A social choice process could be understood as a sequential game, as an interaction between individual choice processes in such a way that the interaction generates a SDR that is compatible with all individual choice processes. To achieve this, we use new tools of 'limited rationality', derived from automata theory, embodied in the system of social decision-making. Complexity as a crucial factor in the choice of decision rules is related to limitations of human decision-making in terms of their capacity to recall, memorize and compute only relatively few items among which consistent choices can be made. In contrast to conventional social choice theory we only consider preference profiles that are in a certain sense 'computable', thus restricting the social choice process to reasonable behavior rules. It is not yet clear to what extent the ideas expressed herein will have an impact on traditional social choice theory, namely relating to Arrow-type impossibility or possibility theorems. In actual human decision-making, alternatives are often examined sequentially, consequently we consider this approach to be basically of sequential type, whereas traditional theory is static, e.g., all alternatives are evaluated before a choice is made. Furthermore, in view of Arrow's assumptions on constructing a social welfare function (SWF) it appears that the assumption of 'unrestricted domain' of the choice set will no longer hold because of imposing strict computational requirements.

An obvious extension would consist of using complexity of decision rules as a primitive notion for an axiomatization of economic behavior that introduces explicitly behavioral assumptions related to limited computability. The DM is not only limited in his choice behavior by computational requirements, but equally important, he is also restricted by acting as a member of a group or social class where, in order to achieve some consensus (for example, a common group decision function), he has to adjust his behavior to past choices of other group members. This is illustrated by looking at the adjustment mechanism as a sequential game. The determinants of the game (environmental conditions, previous choice configurations) are themselves determined as outcomes of complicated cognitive processes, bounded by complexity. Complexity of this kind virtually covers two aspects: one is *structural*, the other *computational*. Structural complexity here relates to the 'sophistication' of the DM, how he can reason when confronted with difficult tasks, depending on his problem-solving capability (as discussed in the example of the missionaries and the cannibals, see Ernst and Newell [2]).

Computational complexity relates to experience, to the ability to learn doing things, organizing computations. Both factors are likely to be highly correlated, but to a certain degree there will be tradeoffs between both, thus they are comprised in one complexity measure. For a particular decision-making design both factors add up to yielding the control complexity which together with the given design complexity provides the fundamental *evolution* complexity relation. This again has a clear interpretation in defining *qualitatively stable* decision rules.

ACKNOWLEDGEMENT

Research supported in part by Thyssen-Stiftung, Cologne (Köln), W. Germany.

Most work has been done during my visit to Western Management Science Institute, UCLA, in the summer of 1976. The help and hospitality of this institution is much appreciated. In particular, I am indebted to Miss Naomi Yano, Los Angeles, and Ms. Goergel, Bielefeld, for typing the manuscript.

University of Bielefeld

NOTES

[1] Likewise, a similar problem arises if you want to capture (probabilistic) uncertainty by the representation of finite subjective probability measures. Here it is by no means clear that the representation is unique. P. Suppes [14] reports, in referring to Scott's axioms of finite probability, derived from a qualitative probability structure:

> The more profound difficulty . . . is the *combinatorial explosion* (my italics) that occurs in verifying the axioms when the number of events is large. To check connectedness, for example, we need only consider pairs of events, and to check transitivity, only tuples of events. But, it is fundamental for the kind of axiom schema required to express necessary and sufficient conditions in the finite case that n-tuples of events of arbitrary n must be studied as the number of events increases. As a possible empirical theory of belief, or as a rational one, this seems impractical, and even for fairly small experiments, the effort to determine whether there is a representing unique probability measure requires the use of a moderate size computer facility.

P. Suppes then sets out to search for simpler axioms, which he terms 'inexact measurement', that attempts to reduce the implicit complexity of finding unique measures of belief.

[2] In a different context, such a problem-solving machine transforming 'tasks' into 'satisfactory actions' (controls) as a model for an adaptive mechanism has been described by B. R. Gaines [4].

[3] This decision-making process, organized in this way, is somewhat related to the heuristic conceptualization of the decision-making process as proposed by R. Selten in his 'Chain Store Paradox' [10]. The simple group machine pertains to his *level of reasoning* which is characterized by a conscious effort to analyse the situation in a rational way on the basis of explicit assumptions whose validity is examined in the light of past experience and logical thinking. On the other hand, the combinatorial group machine applies to his *routine level* where "decisions are made without any conscious effort". Now it seems evident that the higher level of reasoning brings 'sophistication' in the decision process, increases complexity (structurally) whereas routine decisions do not establish structural complexity by itself. This is not to say, in agreement with Selten, that the higher level always yields the better decision, but this is to say that decision problems of the problem-solving variety require the activation of computational devices with more rather than less structural complexity. But in general, again in agreement with Selten, it depends on the nature of the decision problem.

[4] According to C. Futia [3], since combinatorial semigroups ('flip-flops') generate no feedbacks, he argues that feedbacks are only provided by the basic complexity elements, the simple groups, in the Krohn–Rhodes decomposition. Since complexity of his sequential decision rule D, equivalent to the complexity of the associated semigroup $S(D)$, is considered to be proportional to the amount of 'feedback' or 'looping' in a computer program that executes D, it is obvious that he measures only a restrictive notion of complexity, what I call structural complexity. However, he neglects the number of wires or interconnections between *all* components within the Krohn–Rhodes decomposition, i.e., the length of computations, what I call computational complexity. But only structural plus computational complexity provides a comprehensive measure of complexity for sequential processes. The distinction between both is important, particularly in view of possible tradeoffs between both in the design of decision rules and by comparing decision rules with different designs.

[5] Now this reduction mechanism induces the choice space to be partitioned into at least two parts, one part which is 'computable', generated by computable preference statements, the other part is 'non-computable', imposed by indecisiveness in choosing among alternatives. Therefore the actual choice space generated by the selection functions is derived from the following equivalence: computable choice space equals given choice space modulo non-computable choice subspace.

[6] Another way of looking at it utilizes H. Simon's [13] distinction between a well-structured and an ill-structured problem. A *stable decision rule* is equivalent to a well-structured problem. An unstable decision rule results from the possible 'computational gap' which may occur in the problem-solving process. As Simon [13, p. 186] puts it: "... definiteness of problem structure is largely an illusion when we systematically confound the idealized problem that is presented to an idealized (and unlimitedly powerful) problem-solver with the actual problem that is to be attacked by a problem-solver with limited (even if large) computational capacities". So, in a way, if the problem-solver's control complexity is below the design complexity of the decision rule, he himself encounters an ill-structured problem, or equivalently, his decision rule is unstable. Then, it is desirable to redesign the decision rule in such a way that his

ill-structured problem becomes well-structured to the extent that the new design coincides with the computational power of the problem-solver.

[7] H. Simon suggests a 'common sense' test based on the introspective knowledge of our own judgmental process.

BIBLIOGRAPHY

[1] Arbib, M. A., (ed.), *Algebraic Theory of Machines, Languages and Semigroups*, Academic Press, New York, 1968.

[2] Ernst, G. W. and Newell, A., *GPS: A Case Study in Generality and Problem Solving*, Academic Press, New York, 1969.

[3] Futia, C., 'The Complexity of Economic Decision Rules' I, Bell Laboratories, Murray Hill, Jan. 1975.

[4] Gaines, B. R., 'Axioms for Adaptive Behavior', *Int. Jour. Man-Machine Studies* **4** (1972), 169–199.

[5] Gottinger, H. W., 'Complexity and Dynamics: Application of Dynamic System Theory', *IEEE Transactions* SMC (1976), 867–873.

[5a] Gottinger, H. W., 'On a Problem of Optimal Search', Working Paper No. 30, Oct. 1975, Inst. of Math. Economics, Univ. of Bielefeld, W. Germany, to appear in *Zeits. Oper. Research*, Ser. A, 1977.

[6] Leibenstein, H., *Beyond Economic Man: A New Foundation for Micro-economics*, Harvard Univ. Press, Cambridge (Mass.), 1976.

[7] Marschak, J., 'Guided Soul-Searching for Multiple-Criterion Decisions', in M. Zeleny (ed.), *Multiple-Criteria Decision-Making*, Springer Verlag, New York 1976, pp. 1–16.

[8] Rhodes, J., *Application of Automata Theory and Algebra*, Lecture Notes, Dept. of Mathematics, Univ. of California, Berkeley, CA.

[9] Roberts, F. S., 'What if a Utility Function Does Not Exist,' *Theory and Decision* **2** (1972).

[10] Selten, R., 'The Chain Store Paradox', Working Paper No. 18, Institute of Mathematical Economics, University of Bielefeld, W. Germany, July 1974.

[11] Simon, H. A., 'A Behavioral Model of Rational Choice', in H. A. Simon, *Models of Man*, J. Wiley, New York, 1957, Chapter 14.

[12] Simon, H. A., 'Bounded Rationality', in C. B. McGuire and R. Radner (eds.), *Decision and Organization*, North Holland, Amsterdam, 1971.

[13] Simon, H. A., 'The Structure of Ill-structured Problems', *Artificial Intelligence* **4** (1973), 181–201.

[14] Suppes, P., 'The Measurement of Belief', *Jour. Royal Statist. Soc.* **36** (1974), 160–175.

COMPLEXITY, BOUNDED RATIONALITY AND PROBLEM-SOLVING *

Hans W. Gottinger
University of Bielefeld
D - 4800 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, organisation, 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.

---

* Paper presented to the Wittgenstein Symposium, Kirchberg, Austria, Aug. 13-19, 1978

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 introduced earlier by C. Putia [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. $A^* = \{(a_1,\ldots,a_n) : n \geq 1 \text{ and } a_j \in A\}$. Then we define a sequential machine as a function $f: A^* \to B$ where A is the basic input set, B is the output set and $f(a_1,\ldots,a_n) = b_n$ 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, $\lambda: Z \times A \to B$ is the output function. 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:

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

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

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

(4) A right mapping semigroup or right transformation semigroup 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 $x \in X$, $s \in S$, let $xs = (x)s$. Then the following conditions are satisfied:

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

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

(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 $\phi : X \to X$ satisfying the two following 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

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

for all $t_i \in X_i$, $i = 1,\ldots,n$.

(ii) (kth component action lies in S_k) We require $f_1 \in S_1$, and, for all $k = 2,\ldots,n$ and all $\alpha = (t_{k-1},\ldots,t_1) \in X_{k-1} \times \ldots \times X_1$, the function $g_\alpha \in F_R(X_k)$ 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 wreath product of $(X_n,S_n),\ldots,(X_1,S_1)$, and $(X_n,S_n)w\ldots w(X_1,S_1)$ is the abstract semigroup determined by (X,S).

(6) Let (X,S) and (Y,T) be right mapping semigroups. Then we write $(X,S) \mid (Y,T)$, read (X,S) divides (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 $\phi : T' \twoheadrightarrow S$ 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

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

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, a combinatorial semigroup corresponds to a machine that virtually does no computation but rather switches inputs and outputs among various input-output configurations. This 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 rationalty, e.g. computability. Rationality in the normative sense is too restrictive by granting the decision-maker unlimited computational resources which obviously fail to be utilized in 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: there is a definite single criterion for testing any proposed solution,

RPS: 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: attainable state changes (legal moves) can be represented via transitions in a problem space,

APPS: 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, contitional on the knowledge of the environmental states,

PAC: 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 involv a representation of multiple criteria, requiring complex trade off statements which in fact would enhance the number of computational ste

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 sufficie 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 3.0' (David Slate, Northwestern University). In a state-of-the-art survey Newell, Shaw and Simon [1967] 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 understood. 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 practise 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 startegies? 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 be 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 optim(2) it appears that highly selective search, the drastic pruning of the tree or in depth search 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 in large problem spaces. Heuristics do not guarantee optimal 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, summarising 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 [97]: '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 [96]: '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-ends analysis and planning, a subject matter we will briefly describe in Sec. 5.

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 $\mathcal{S}$ 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_0 to generate the goal state

$$z_n = S_n(S_{n-1}, (\ldots S_2(S_1(z_0))\ldots)).$$

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 $S \in \mathcal{S}$ 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 negligible.

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 opponent's king is chessmated. 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 of problem-solving may be subsequently redefined and modified subject only to the restriction that at no point 'survivability' is endangered (endogenous 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> procedures, 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</u>

fixed in advance 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 utility 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 form 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 traveling salesman problem is the set of all acyclic permutations of the cities, i.e. the set of feasible tours. The number of these turn out

to be bounded by $(n-1)!/2$ which is an extremely large number for moderate n. By Stirling's formula $n! \simeq (\frac{n}{e})^n$, 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 Heuristic: 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 generalizati 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 defintion, $O_1,\ldots,O_n$.

(b) A set of attributes $X_1,\ldots,X_k$ attached to the objects, 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

additivity operators: $A_1,\ldots,A_k$
difference operators: $D_1,\ldots,D_k$
multiplicative operators: $M_1,\ldots,M_k$
} operators $Q_1,\ldots,Q_k$

such that they satisfy algebraic operations of finite simple groups.

(d) A terminal goal structure containing the desirable attributes $X_1^*,\ldots,X_n^*$ such that the operations applied to $X_1,\ldots,X_n$ generate the desirable set $\{X_1^*,\ldots,X_n^*\}$ 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. There moves satisfy a certain algebracih 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' or 'remis

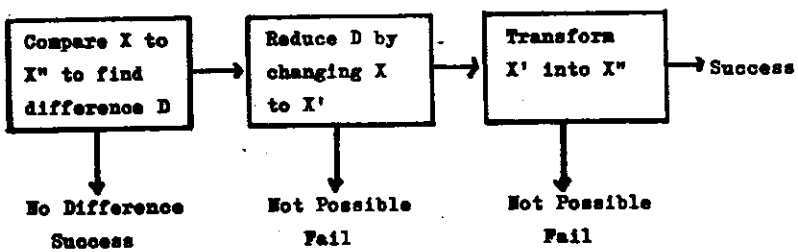
Consider, for illustration, a much more simplified problem, the so-called Monkey Problem:

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?

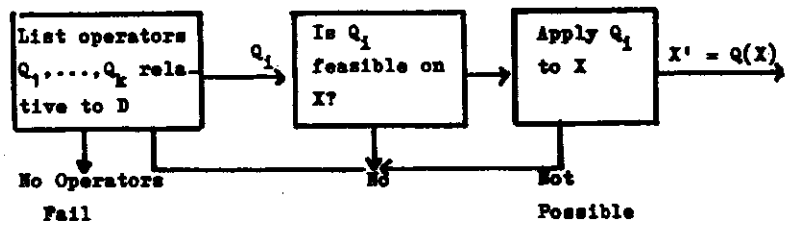
Here the objects are three: the monkey, the box and the bunch of bananas. The situation can be described by stating: the altitude of the monkey, the location of the monkey in the cage, the location of the box, and the location and altitude of the bananas.

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:

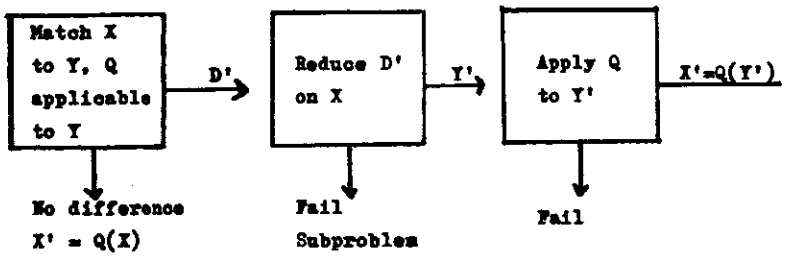


(2) Once differences have been evaluated, difference reducation is achieved by subsequent application of an operator sequence. Starting with given D and X a list of operators is to be determined which, if applicable to X, will alter the characteristic D on X. Let Q_i be the first such operator. A check 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 inter,ediate 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 of 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 (2), 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 sub-problems.

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: Dept. 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, 181-201.

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 (1950), 'A Chess Program', Scientific American.

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

J. McCarthy (1974), 'Book Review of Lighthill, J.: 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.

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 cost 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, airline, 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 planned 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 feed-backs from locally feasible steps toward global goals that might modify the global framework.

For example, in chess-playing and elsewhere 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 structural 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 space, and finally using the solution to the simplified problem as a way of setting sub-goals. A difficult GPS problem is (hopefully) made simpler by solving a simpler but related GPS problem.

Abstract

Structural Characteristics of Economic Models:
A Study in Complexity

Hans W. Gottinger
University of Bielefeld
P.O.B. 8640
D - 4800 Bielefeld 1

It is claimed that economic model-building has only limited policy implications unless the issue of complexity is taken explicitly into account. It is shown how structural connectedness works in economic models and what impacts decomposition has on the model's behavior. Results of the Ando-Fisher theorem of decomposition are applicable here. A simple measure of complexity is proposed which has its roots in decomposition results of dynamic algebraic systems. The complexity measure is applied to a model of an economic system and reveals the computational dimension of the model's parameters. Reduction of computational complexity can be achieved by utilizing the potential problem-solving power of the interacting subunits ('modules'). The relation to problem-solving methods in artificial intelligence appears to be relevant.

Structural Characteristics in Economic Models:
A Study in Complexity

Hans W. Gottinger
I.M.W., University of Bielefeld,
D-4800 Bielefeld 1, F.R.G.

1. Introduction

In the past twenty years we have experienced a tremendous advancement of tools and techniques in analyzing, diagnosting, predicting and controlling economic processes. Sophisticated models have been built that claim to predict future economic trends of micro and macro-economic variables, largely facilitated by the availability of large-scale computational resources. These models also form the basis of policy recommendations such as the Brookings models or similar large-scale econometric models. There have been some major improvements in the design, estimation, statistical structure, testability of economic model-building as have been in decentralized decision-making, distributed computation and hierarchical control. Yet, there is still one major link missing which we consider as one of the most fundamental properties of any model-building in the large that one has not come to grips with the structural constraint of complexity, e.g. the infor-

mation processing limits arising in the control of dynamic systems. Also in a recent survey on large-scale systems (N.R. Sandell et al [1]) it is stated that an adequate measure of complexity is lacking for control systems and that a major task of constructing a unified theory of decentralized control is to include a formal measure of system complexity. We focus on this structural constraint of complexity by designing models that explicitly cope with this issue.

2. Preliminary Considerations

From a methodological view there are two ways to understand and predict the behavior of (discrete-time) dynamic systems. The first is to evaluate changes in the future by past performance, to project realizations in the past into the future, what may be summed up by 'prediction by trends'. Intuitively, this is based on the idea that any system's behavior 'cannot escape the past to shape the future', that any evolving system develops its own memory which conditions it to past events. The more data are compiled from the past the more the memory activates its own self-organized dynamics. This appears to exclude purely random behavior in the past, exhibited by a random sequence, hence the occurrences of events do not seem to be statistically independent. The second is by constructing a model which is considered to be a representative mapping of the system under investigation. A model attempts to capture the basic technical, structural and behavior characteristics of the underlying system and on this ground estimates its potentialities for future development. This is referred to as 'prediction by models'. Now 'prediction by trends' assumes regularity of the process as if social and economic processes satisfy laws of statistical regularity. It comes very close to conceiving the world of being in a state of 'disorganized complexity' according to W. Weaver [2], meaning that we can view courses of events essentially as 'random sequences' and that the statistical law of large numbers is the basic concept to predict future behavior. Already F.A. Hayek [3] remarked that economic processes cannot be observed in a statistical fashion. The problem with a statistical assessment of complexity, or disorganized complexity, lies in the attempt to average over a large number of random variables, as represented by the mean or expected value, which, indeed, treats all phenomena of the system alike or as uniform whereas in reality there exist different structural relationships between elements that bear different characteristics and that may have a non-negligible impact on the behavior of the entire system. Hence, it is very doubtful that economic systems can be explained on the basis of pure random phenomena. I have shown elsewhere (Gottinger [4]) that this implies that economic systems would perform as if they were infinitely large systems. Instead, I have pointed out that economic systems are much more akin to finite complex systems' (FCS), i.e. systems that are much too complex to get explicit solutions for them, as is the case in 'simple' systems, nor is the number of parts large enough or are the parts homogeneous enough to be able to pass to the limit as in infinitely large systems. An FCS has some peculiar properties, not shared by its better worked out counterparts that makes it difficult to analyze, understand and predict its behavior. It appears to be (1) highly sensitive

in responding to changes or disturbances of its environment, (2) strongly interdependent with regard to actions of its components, (3) following a threshold discretionary type of behavior with qualitative jumps, (4) partially or locally controllable inducing global effects. In fact, these properties can be observed in modern economic systems that render them a higher degree of instability as similar behavior patterns are traced in ecosystems (R.M.May [5]). Hence behavior of an FCS is not of a purely random character. The reason for this is that it reveals certain imperfections like lagged responses, maladjustment, non-stochastic dependence between parts, discontinuity that appears not to be compatible with randomness.

3. A General Approach to Dynamic Systems and Complexity

The approach starts with the basic identification problem. The technicalities have been treated elsewhere (Gottinger [6],[7]). Given some natural complex system, a black box, where we only observe outputs and inputs over time but we are ignorant about what is happening 'inside', e.g. about local properties or parameters of transition. Is it possible to find an 'artificial' system that simulates the original natural system? Systems we have in mind in this context are those which respond in real-time to their environments just to stay alive (bull fighting a bear). Ecological systems (bird colonies) or biological systems (metabolism of cells) constitute systems striving for survival, also all types of competitive economic systems challenged by adversary environments belong to this category. In general, extreme notions, such as survival or non-survival (death), which are characteristic for pure forms of competitive systems

are involved. Here interest is focused on global properties of dynamic systems. The design orientation follows from the identification process, e.g. by taking interconnecting part of the artificial system 'hooking' them together in order to simulate the 'black box'. The approach is algebraic, since it starts from finite-state dynamic systems, e.g. sequential machines, the general characteristics of which are described by:

1. A set of inputs, e.g. those changing parameters of the environment which will affect the system behavior in a predictable way.
2. A set of outputs, i.e. those parameters which act upon the environment leaving observable changes in the relationship between the system and the environment.
3. A set of states, i.e. those internal parameters which determine the relationship between inputs and outputs and which may reveal all necessary information embodied in the past.
4. The state transition function which determines the dynamics of how the state will change when the system is fed by various inputs.
5. The output function which determines what output the system will yield with a given input when in a given state.

There are some obvious advantages, theoretical and practical ones, to use an algebraic approach. First, algebra is a natural tool because it emphasizes the design of a system as a collection of objects. Second, algebra is natural for computational work, and this is an important factor in applications. Modern computers accept digital instructions and those in turn require an algebraization of systems. Third, algebraic system theory emphasizes qualitative aspects of systems to the extent that we are interested in properties such as survival or breakdown. This is achieved by determining complexity bounds of the

system's design (design complexity). By the fact that algebraic system theory is related to computational structures we are in the position to construct a complexity theory for dynamic systems which is also amenable to applications.
Systems of that sort reveal a natural bound of complexity. Complexity here has two facets, computational complexity and structural complexity.
Structural complexity refers to the inherent capability of parts of the system, of what they are able to perform.
Computational complexity refers to the length of computation given by the extent of interconnection between parts. The most important distinction is that between design and control complexity.
Under design complexity it is understood that complexity (number) associated to the transformation process in which full use of the system potential is made. Design complexity can only be achieved if a system is working without faults, if the components live up to computability requirements, if the parts function reliably. Under control complexity we understand that specific complexity number which keeps the entire system or at least part of it under complete control. Only if design and control complexity coincide stable configurations in the state and output space will result, or the system runs smoothly. The relation between design and control complexity is an indication for stability or harmony of a dynamic system.

4. Decomposable Systems

We wish to explore the nature of the interdependence in economic systems, to show how these systems might be decomposed into their component parts for analytic purposes, and to relate the results to the choice of models in policy analysis and projection.
By the previous arguments and those provided by the algebraic theory of machines we are advised that there must be general principles of decomposition which arise in the design process of algebraic (computational) systems. From these general principles of decomposition we emerge with a natural theory and measure of complexity which is structural (intrinsically related to the parts, the basic building blocks) and computational, e.g. addressed to the computational links between these parts. We will see how this framework is useful for approaching more specific type questions in economic model building.
The core for the investigation in the present section is provided by the Ando-Fisher theorem [8] on the decomposibility and independence in dynamic economic systems.
The Ando-Fisher theorem and related results deal with general ways how to decompose a system into its component subsystems for purposes of analysis and prediction. They distinguish between (completely) decomposable and nearly (completely) decomposable systems. In the latter case, a nearly decomposable system consists of subsystems where each is causally dependent on the rest of the system but the rest of the system is only weakly dependent (or weakly coupled) on (with) each subsystem. In contrast, a decomposable system only allows for within syutems dependencies but ignores intersystems dependencies altogether. The Ando-Fisher theorem asserts, for linear dynamic systems, that

(1) provided inter-systems dependencies are sufficiently weak (up to a negligible degree) relative to intra-systems dependencies, then, in the short run, the relative behavior of a nearly decomposable system becomes almost indistinguishable to that of a decomposable system.

(2) provided inter-systems dependencies are sufficiently weak relative to intra-system dependencies, then also in the long run relative behavior of the nearly decomposable system and the decomposable system is approximately the same even though their behavior in terms of absolute levels and rates of change may be very different. In other words, if we give a nearly decomposable system enough running time, and that even when influences having been neglected have had time to make themselves fully felt, the relative behavior of the variables within any subsystem will be approximately the same as would have been the case had those influences never existed.

Now there are obviously two critical factors in the validity of the last result:

(a) the degree of approximation depends on the number of computational cycles (e.g. running time) and is certainly a result in the limit,

(b) the degree of approximation depends on a prepostulated, sufficiently weak linkage among the subsystems, below a certain threshold value where the system moves continuously in an orderly manner. This however presupposes a very large number of subsystems, acting uniformly on an equal power base, as in the classical model of economic

equilibrium.

Both assumptions (a) and (b) are hard to defend in the context of analyzing real-life social and economic systems. For these systems, as real-time systems, we simply haven't got enough running time to force this approximation upon the system's behavior. Time is a scarce and valuable resource.

Second, there are only a few, relatively large subsystems, the relative behavior of which have a great potential impact on the entire system, and where the intra-system behavior is not limited to the subsystem itself.

In fact, we would very much argue in the opposite direction. The characteristics of large-scale complex systems are such that they are very sensitive to discretionary behavior of their subsystems and that actions, outcomes and consequences of these subsystems very often induce snowball-effects that pervade other subsystems and enforce actions, perturbations, maladjustments, all sorts of reactions that deeply affect smoothness, regularity, stability, controllability, and is highly unpredictable because of the non-linearities involved.

5. Systems, Modelling and Complexity

Let us start by describing a dynamic system very much like an automaton. The ingredients are given by

an internal state vector $z_t = [z_{1,t}, z_{2,t}, \ldots z_{n,t}]$

an external state vector $x_t = [x_{1,t}, x_{2,t}, \ldots x_{n,t}]$

representing exogeneous factors, driving the system from 'outside', and not incorporated in the decomposition.

An output function δ that maps strings of inputs, say $(a_1,\ldots a_n)$ into single outputs $b_1,b_2,\ldots b_n$ which enter as inputs into other parts (components) of the system, the state transition function λ with

$z_t = \lambda(z_t, z_{t-1},\ldots,x_t)$, given in

difference equation form. The function λ, without any specification yet, represents the hypothesis about the process, inferred from the observations of real world systems. The external state vector x_t can be conceived as a primary input factor (stimulus) which sets the system into motion, but which itself may be suitably partitioned as to which component is primarily affecting the process.

The behavior of the system is generated by the set of time series of the z_{it} which are produced as the model generates successive state descriptions plus the external states given exogeneously.

Now in the chartist approach to studying the future of systems it is assumed that the connections between different components either do not exist or are sufficiently weak that they can be safely ignored. The value of any component z_{it} depends only on its previous values and possibly random disturbances u, thus predicting a future course of actions or events means extrapolating past performance.

In a comprehensive modelling approach, as suggested here, at least some of the possible interdependencies among components exist in general. The behavior of each component may depend on its past behavior as well as on other variables in the system, the corresponding exogeneous factors x_{it},

$i = 1,\ldots,n$, respectively, and possibly the random disturbances u, i.e.

$$z_{it} = \lambda(z_{i,t-1}, z_{i,t-2},\ldots,z_{k,t}, z_{k,t-1}, \ldots, x_{it},u) \quad k \neq i.$$

Two immediate problems in the model procedure arise from the Decomposition Scheme exhibited in Fig. 1

Insert Fig. 1 here

(1) As the number of components and sufficiently strong connections among components increase the behavior of the system becomes increasingly obscure by complex interactions which resemble very much non-linearities in total system's behavior in correspondence with size.

(2) The structure and size of the components themselves present a potential source of complexity depending on whether and to which extent the component system is sensitive to disturbances, errors, threshold phenomena, etc.

(3) As the number of components and interdependencies in the system enhances, increasingly longer sequences of calculations are required to deduce the behavior of the system which results in computational complexity.

The solutions of these three problems would enable us to determine the complexity of the system on-line, as it is running

from some initial time to some target time in the future. But knowing the complexity would permit us to design control strategies which are effective in guiding the system toward relative stability or harmony.

Therefore to understand the complexity of such systems we should be able to understand the strongly connected, coupled nature of its subsystems. For this purpose we need a measure of complexity that reflects the structural performance of each of the connected subsystems in terms of state space configurations plus the number of computational links that are established among the various subsystems and that reflect the richness of state representations in the global trajectory space of the entire system.

Illustration

Take the population subsystem [POP] consisting of initial eight states, e.g. $z = (z_1, z_2, \ldots, z_8)$, then after inputs are fed into the [POP] system a new state configuration obtains

Insert Fig. 2 here

A simple measure of structural complexity, in accordance to Gottinger [6], [7], could be given by the number

$$\#_s(z) = \sum_{i=1}^{8} \overline{tt}_s(z_i) = \sum_{i=1}^{8} \text{(length of time needed to reach a satisfactory state) x (number of feasible states to be attainable)}.$$

On the other hand, the computational complexity indicates the number of links between various subsystems times the number of interactions that ensue until the computational cycle (in real-time) is completed, as roughly indicated only for one specific subsystem, say [POP], (for all other subsystems likewise) in the following illustration

Insert Fig.3 here

In the satellite system, connecting all neighboring subsystems, the computational complexity related to the [POP] subsystem amounts to

$$\#_c([POP]) = (\#\text{ links to [EN]} + \#\text{ links to [ECO]} + \ldots + \#\text{ links to [P]}) = \sum_{i=1}^{n} \#_i(\text{links to } i).$$

Then the total computational complexity comprising all links among all subsystems within the satellite system is given by

$$\#_c([POP],[EN],[ECO],[EC],[G],[P]) = \#_c(Z_F,F) = \sum_{k=1}^{6} \#_k(Z_{Fk},F_k)$$

Using the notation, let Z_F be the state space over which the entire finite state system is running, let F be all feasible semigroups of transformations acting on this space, then in accordance with decomposition results of algebraic finite state systems we establish a comprehensive complexity measure comprising of structural and computational complexity, e.g.

$$C(Z_F,F) = \prod_{k=1}^{m} \#_s(Z)\#_c(Z_F,F),$$

and this is the measure we use in analyzing the structure of a particular economic model.

6. Structure of the Model

We wish to explore the nature of interdependences in societal syster to show how systems might be decomposed into their component parts for analytical purposes and to relate the results to the choice of models in policy analysis and projection. According to what has been illustrated in Fig. 1 we choose a kind of partition of the overall system into parts that comprise the main activities of complex societal systems - this is very much in spirit of earlier attempts at modelling complex political systems for purposes of simulating their behavior, see R.D. Brunner and G.D. Brewer [9]. The components of this model consist of the set of variables and parameters listed under the detailed description later.

[POP] - the population subsystem is relatively specialized to the growth and distribution (density) of the population N_t, being composed of the urban and rural population.

[EN] - the energy subsystem is simpled out as the major resource sector which could be of mixed private-public activities, and where the energy basket consists of the output E_t of primary and secondary energy resources generated.

[ECO] - the environmental system puts environmental restrictions (water and air pollution) on growth of production in urban and industrial areas.

[EC] - the economic subsystem is relatively specialized to the production and distribution of economic goods and services within the private sector.

[G] - the government subsystem comprises all activities of the public sector which are directed at providing goods and services demanded by the market place or generated by political considerations.

[P] - the political subsystem is specialized to the production of changes in the size and distribution of mass support for the government MV_t to permit majority rule and conservation of power, furthermore, determines the consumptive and distributive characteristics of government expenditures G_t. The relationships in the model are grouped according to subsystems, these relationships are hypotheses about the way in which each variable changes as a function of the others, with the magnitudes of the changes being determined in part by the parameters.

The degree of connectedness or interdependence implied in the relationships can be explored by noting the presence or absence of causal links among the variables. Call this the structural connectedness of the model. In fact, there are direct causal and indirect causal connections (links) between the output and input variables, depending on whether they occur within the components blocks or are cross-connected among different blocks. The static description of the structural connectedness underestimates the degree of connectedness among variables as the model operates through time. For example, since government revenue R_t is merely a function of gross output Y_{t-1}, variation in G_t within one real-time cycle is limited to the variation in this variable. However, if the model is operated through time, the number of variables causally connected to R_t increases. Thus while R_t is a direct

function of Y_{t-1} (or some aggregates thereof), it is also an indirect function through this variable of C_{t-1}, I_{t-1}, G_{t-1}, and through these variables R_t is an indirect function of several other variables, etc. Obviously, the power of the connectedness matrix increases in correspondence with an increse in the number of real-time cycles of the model. The analysis of structural connectedness has obvious limitations for the chartist approach to prediction. If the analysis suggests that even in a system that is loosely connected in its static description every variable in the system may ultimately depend upon every other variable as the system runs on-line through time, then a simple extrapolation of trends which ignores these dependencies may indeed be highly misleading. In a situatich where the number of outside factors tend to be increasing to infinity, we would well expect some statistical regularity in offsetting influences among the factors, and,indeed, a chartist approach may be a good thing to follow by neglecting these dependencies. This would correspond to view the real situation as a state of 'disorganized complexity'. However, in the course of previous arguments, the case of an FCS would not support this point. Then a serious modelling and prediction effort would have to take into account indirect effects that may have a highly nonlinear character. However, there is the possibility that in a particular system or in a particular application of the model, certain of these dependencies may be sufficiently weak that they can be safely ignored. If this were the case, the chartist's approach

which ignores these dependencies and the comprehensive modelling approach taking them more fully into account may lead approximately to the same conclusions.

7. The Model's basic sets of relationships

(a) [POP]-Subsystem

Rate of population change

$P(\dot{N}) = \alpha_1$ (average number of children per family) + β_1 (number of women in child-bearing age) + λ_1 (availability of birth control devices and liberal abortion policies) — exogeneous factors [EF]

+ δ_1 (expected average family income, given some current level of family income) — [EC]

+ ϵ_1 (population supporting energy consumption level) — [EN]

+ η_1 (population density index level) — [ECO]

+ ζ_1 (provision of public goods and services relevant to child rearing activities) — [G]

+ θ_1 (level of wide-spread political satisfaction and trust in political institutions) — [P]

(b) [EN] - Subsystem

Supply of Energy

$S(E_t) = \alpha_2$ (availability of domestic energy resources) + β_2 (energy provision through imports) + λ_2 (potential of mobilizing new energy sources via technology) — [EF]

+ δ_2 (price expectations and market prices) [EC]

+ ε_2 (level of manpower directed toward energy production) [POP]

+ η_2 (upper limits of energy provision by environmental factors, i.e. strip-mining, health and environmental hazards, pollution) [ECO]

+ ξ_2 (degree of government intervention in terms of price or quantity regulation) [G]

+ θ_2 (political interference in the industrial organization of the energy industry) [P]

$D(E_t) = \mu N_t + \lambda Y_t$, energy consumption is proportional to total population and to the level of G.N.P.

$\Delta(D(E_t) - D(E_{t-1})) = \mu(N_t - N_{t-1})$, the rate of energy consumption is proportional to the rate of population change.

(c) [ECO] - Subsystem

Abatement Performance (measured in terms of level of exhaust emission)

$AP = \alpha_3$ (index of industrial distribution)

+ β_3 (environmental immersion factors)

+ λ_3 (level of environmental technology) [EF]

+ δ_3 (G.N.P. as related to regional and sectoral industrial activity) [EC]

+ ε_3 (rate of population change and structural population shifts from rural to urban sectors) [POP]

+ ξ_3 (government regulation and industrial incentive structure in terms of emission limits, waste disposal, land use zoning regulations, noise restrictions) [G]

+ η_3 (energy supply and mixture of energy sources) [EN]

+ θ_3 (effectiveness of government control, efficiency/inefficieny ratio of supervision) [P]

(d) [EC] - Subsystem

Links of G.N.P. aggregates by definition:

$Y_t = C_t + I_t + G_t + F_t$ (F_t = foreign trade, given exogeneously).

$C_t = m_t(1-\tau)Y_{t-1}\dot{N}$,

$I_t = I_{t-1}\dot{N} + r(\dot{C}\dot{N})^{+f(G_t)}$, $\dot{C} = dC/dt$

where f represents the extent of government expenditure restricting private investment, to take care of welfare - state limits on private investment activity.

Growth of G.N.P.

$\dot{Y} = \alpha_4$ (resource endowment)

+ β_4 (R & D level)

+ λ_4 (level of technology used) [EF]

+ δ_4 (minimal level of energy supply needed to support growth) [EN]

+ ε_4 (minimal abatement performance AP = maximal pollution, noise, waste disposal limit) [ECO]

+ η_4 (size of government investment activities) [G]

+ ξ_4 (rate of population change related to productive employment) [POP]

+ θ_4 (government guidance and organizational support) [P]

(e) [G] - Subsystem

Government revenue is proportional to G.N.P. Y_t via the

proportionality factor τ (tax rate).

$R_t = \tau Y_t$.

Government expenditure

$G_t = \alpha_5$ (international structural changes in terms of balance-of-payments surplus/deficit) [EF]

$+ \beta_5$ (rate of population change and age structure of society) [POP]

$+ \lambda_5$ (energy supply $S(E_t)$ and R & D expenditures on new energy sources) [EN]

$+ \delta_5$ (abatement performance: safety and environmental regulations) [ECO]

$+ \varepsilon_5$ (level of economic and industrial activity)

$+ \eta_5$ (employment and capacity level)

$+ \xi_5$ (index of inflation) [EC]

$+ \theta_5$ (political support by major segments of the population (social groups)) [P]

(f) [P] Subsystem

Majority of votes from all or some segments of the population to achieve majority rule in Parliament.

$MV_t = \alpha_6$ (psychological conditions of trust, honesty and stability in society) [EF]

$+ \beta_6$ (rate of population change)

$+ \lambda_6$ (change of migration patterns from rural to urban areas)

$+ \delta_6$ (occupational shifts) [POP]

$+ \varepsilon_6$ (energy supply and distribution of energy resources) [EN]

$+ \eta_6$ (level of pollution affecting major social groups and key industrial areas) [ECO]

$+ \xi_6$ (composition of rate of G.N.P. in relation to political goals agreeing with major interests of the population) [EC]

$+ \theta_6$ (provision of government services and collective goods) [P]

8. Evaluation of Complexity

Now the static description of the structural connectedness underestimates the degree of interaction among subsystems as the interaction operates through time, this is true since the various feedback- and feedforward mechanisms reinforce the interaction in a nonlinear way. We consider the connected system as a real-time on-line dynamic system, see Martin (10), that starts operating at a specific initial state for which sufficient information is available, and which operates on an equal length time interval of one year. The selection of one year (in a real-time computational cycle) is arbitrary and used only for illustration purposes although parameter changes in this model appear to have only longer term impacts. Then the the number of links to be estimated (pertaining to computational complexity) amounts to the following set-up. The entire system consists of altogether six modules, [POP], [SN], [ECO], [EC], [G], [P], plus the external environment, exogeneous factors [EF]. Each module consists of at least five linking factors to other modules, neglecting the exogeneous factors [EF] which are considered to be predetermined, the information of which is only directly accessible by the module concerned. Each of these five linking factors or parameters depend upon at most eight factors

in some other module, therefore indirectly involving [EF].
In the first case we refer to direct links, in the second to indirect links of each of the modules involved.
For illustration look at the determination of G_t in Module [G]:

Insert Fig. 4 here

With six active decision-making modules operating the total number of structural parameters to be computed amounts to at least 240 structural parameters, e.g. five direct links given by the parameters of each module time$_8$ eight indirect links for each such parameter in some other module times six, the number of interactive modules.
Now granting that at least 240 structural parameters be estimated, and suppose a response cycle of yearly data for a period of a ten year planning model is used, then in view of the non-linearities of the system interactions as the system operated through time and extends the length of indirect links, we can show, by complete enumeration, that we arrive at a total number of 240^{10} structural links which is a dramatic increase. In other words, the same number of structural parameters would have to be estimated in order to reach a control complexity that fits the design complexity of the entire system - a formidable task. (Still, it would be even more dramatic if we relate the response cycle to quarterly or even monthly data.) Now this holds for a problem-solving mechanism acting as a brute-force search, as built into large-scale computer programs, in which

(i) the modules do not provide any structural complexity, i.e. no intrinsic problem-solving power as given by a special heuristic ,
(ii) there is complete ignorance on the controller's side concerning the parameter variations in the dynamic process.

In other words, this corresponds to a centralized controller who perceives the modules simply as 'black boxes' with no self-steering capability and who at each real-time response cycle is ignorant about (or does not learn about) parameter variations that ensue.
In the alternative case the controller may activate the problem-solving power, heuristic search capabilities of the modules where each module, as a decentralized unit, by itself intrinsically computes all indirect links, thereby decreasing the computational burden of the controller. Hence the computational process reduces to direct links only, at least five for each module, so that for each response cycle 5 x 6 = 30 structural parameters have to be estimated as compared to 240. This amounts to 30^{10} structural links in a ten years period. In fact, the reduction of the number 240^{10} to 30^{10} could be assigned to the 'smartness' or 'structural complexity' of the modules, precisely the structural complexity amounts to $240^{10} - 30^{10}$, the residual in the reduction of computational complexity. Therefore, we see how a tradeoff between structural complexity and computational complexity evolves in such an interactive model.

Insert Fig. 5 here

9. Conclusions

The most interesting aspect of this analysis for problem-solving methodologies pertains to the distinction between design and control complexity. If the design complexity of the system amounts to 240^{10} structural links, the only way to cope successfully with the control of such systems is to use a decomposition that makes most out of the intrinsic computational capabilities of the modules, their smartness or sophistication, much in the same way as chess playing programs become smarter to match master chess players by building into the programs some sophisticated heuristics.
This sheds new light on the question of centralization vs decentralization of decision-making in organizations, in particular in view of these results one could hardly hope to achieve rational centralized economic planning against market-type decentralized mechanisms (see F.A. Hayek [11]). At least the need for decentralization is now widely acknowledged in Soviet work on program-planning. As N.N. Moiseev and A.G. Schmidt [12] put is: 'it is not difficult to see that no matter how advanced the level of the techniques for data processing and data transmission may be, a certain level of decentralization in management will always be necessary.'

Furthermore, in general mixed-type economies, one can conclude that government regulation will easily find its limit of workability because of the computational burden that ensues.
Another point worth of consideration is that the analysis of structural connectedness had obvious implications for the chartist approach to projection. Since even in a system that is loosely connected in its static description, every parameter in the system may ultimately depend upon every other parameter as the system operates through time. A simple extrapolation of trends which ignores these dependencies may indeed be highly misleading. If in a particular system it turns out that most of these dependencies (links) are sufficiently weak so that they can be safely ignored (according to the Ando-Fisher Theorem), the projections of the chartist who neglects these dependencies and the fundamentalist who takes them fully into account may be approximately the same. But even 'simple' systems which are only loosely connected in their static description are highly interconnected in their dynamic behavior.
Let's close with a methodological note.

Throughout this paper we attempted to answer the question how complexity appearing as a structural constraint on large-scale dynamic economic systems can be successfully handled from a controller's point of view. For answering this question it appears necessary to know what units of decomposition are most appropriate for designing and understanding complex systems. The sheer amount of computation, reflected by a measure of computational complexity, prohibits the decision-making or information-processing power