COMPUTABLE ORGANIZATIONS-REPRESENTATION BY SEQUENTIAL MACHINE THEORY

by

Hans W. Gottinger

Memorandum No. ERL-M426

March 7, 1974

ELECTRONICS RESEARCH LABORATORY

College of Engineering
University of California, Berkeley
94720

COMPUTABLE ORGANIZATIONS-REPRESENTATION BY SEQUENTIAL MACHINE THEORY*

Hans W. Gottinger**

Department of Electrical Engineering and Computer Sciences
and the Electronics Research Laboratory
University of California, Berkeley, California 94720

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.

Research sponsored in part by the Army Research Office - Durham, Grant DA-ARO-D-31-124-71-G174.

*Paper presented at the Cambridge Conference on Public Systems, Cambridge University, November 24-26, 1973.

**Visiting Professor from Universität Bielefeld, Kommission für Entwicklungsforschung, Bayerische Akademie der Wissenschaften, München.

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 $\mathcal{E}$ 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 resulting in 'messages' (prices). Now an adjustment process in 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 $X = (\Omega, \mathcal{R}, \mathcal{T})$ where Ω describes the set of resources, $\mathcal{R}$ a set of preference relations on Ω and $\mathcal{T}$ 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 $(\lambda,\delta,\mathcal{M})$, consisting of a response function λ (possibly a vector for a finite number of agents), an outcome function δ, independent of the environment but depending on the amount of resource endowment, trade, production, etc. given the environment and a message space ('language') $\mathcal{M}$ whose elements ('messages') generate new messages (via the response function λ) for any given environment e. 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 <u>in equilibrium</u> and the message received at that stage is <u>stationary</u>. To every informational equilibrium value of the process $\bar{m} = \lambda(\bar{m},e)$ there may correspond a (Pareto-) satisfactory outcome level $\delta(\bar{m})$ which is not 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 non-classical environments with externalities and indivisibilities present and technology not necessarily convex other types of processes different from the competitive process have been studied w.r.t. 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 $\mathcal{M}$ 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 $\mu: X \to \mathcal{M}$ and a response function $\lambda: \mathcal{M} \to A$. Hence the adjustment process (μ,λ) is induced by an initial message set $\mu_0(x)$. The outcome function $\delta: \mathcal{M} \to A$ may be introduced in the appropriate context. It is clear - technicalities omitted - that the response function satisfies some 'nice' 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 <u>sequential machine</u>.

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. Later we will give some examples demonstrating the usefulness of this analogy. 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]

(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 resuorce endowment given to the machine.[2]

(3) What a 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 modern 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 transitional state function, $\delta: X \times Y \to Y$ an output function.

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 $\delta: X \times Y \to Y$ where the state set Z represents the physical and informational activity of the system. Now there is one problem by transforming the set of environments 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 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.

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

next states

Figure 1.

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 from (for example in bianry form). The organization we would like to describe as an electrical device would then be represented by the following scheme (here → denotes an instruction).

stimuli

states		00	01	10
	00	00	00 → 00	10 → 00
	01	00	00 → 10	10 → 00
	10	00	01 → 11	10 → 00
	11	01	01 → 11	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 of 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.

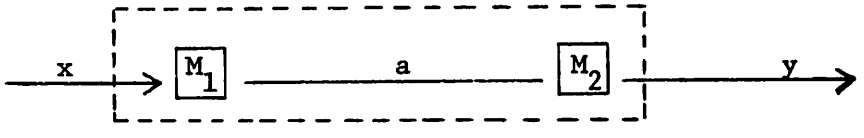


Fig. 3

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 as those given in the foregoing example.

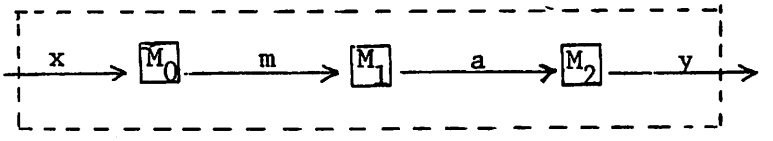


Fig. 4

In an analogous way we could talk about <u>parallel decomposition</u>.

<u>Definition</u>: 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 $\delta[(x_1,x_2),(z_1,z_2)] = (\delta_1(x_1,z_1), \delta_2(x_2,z_2))$.

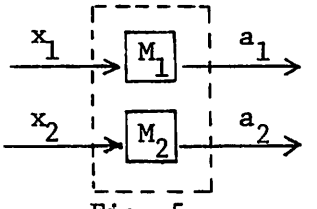


Fig. 5

Parallel connection

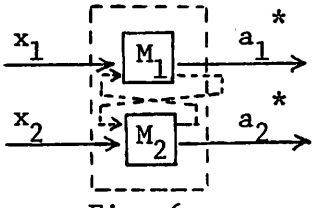


Fig. 6

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.

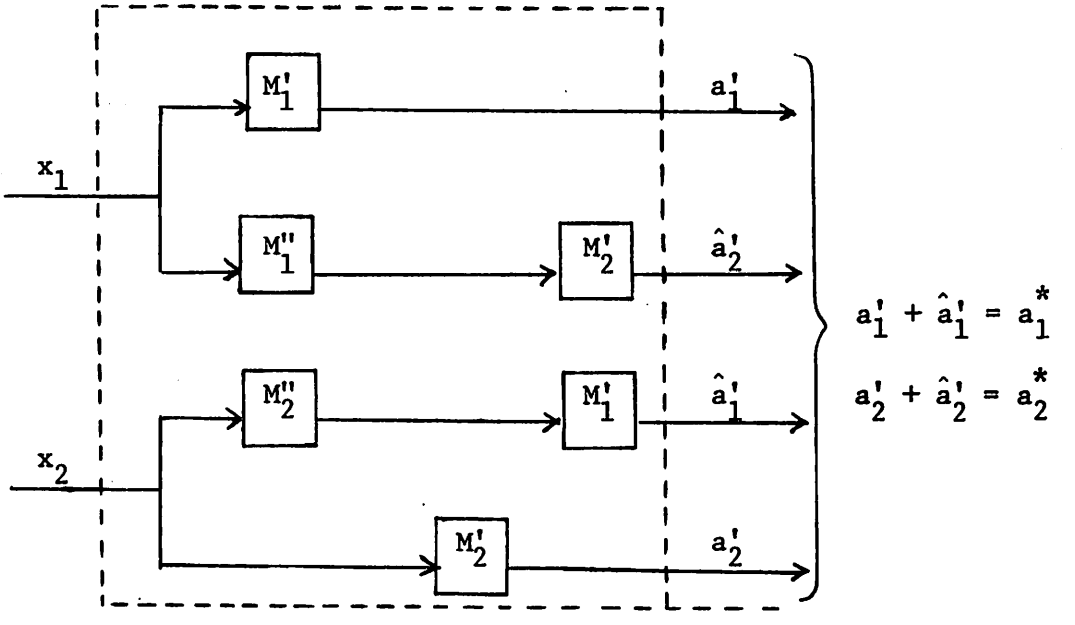


Fig. 7: Serial-parallel restructuring of two cross-connected parallel machines.

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:

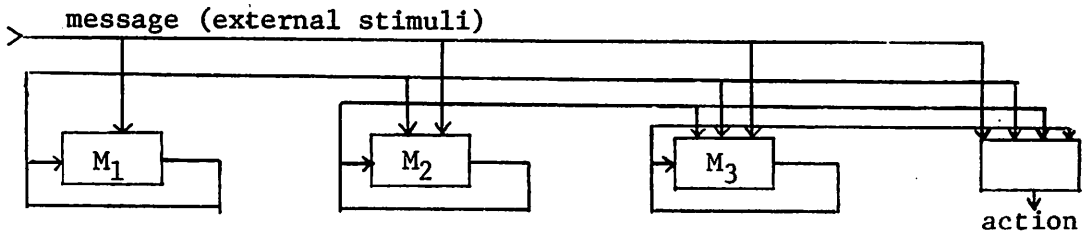


Fig. 8: Cascade machine.

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 other important consideration. One could argue that decomposing an organization into information, decision and payoff machine is a rather artificial procedure 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 ma-e in the allocation of 'informational resulrces' (incentive compatibility)[5].

An organization - as a machine - informs, computes, remembers, acts and reveals its state behavior and output structure. For doing all this the informational reuqirements may well be substantial. In fact, it might turn out that such a machine only works on the basis of highly aggregated messages (data) which in itself constitutes a considerable loss of information. It is therefore essential to know whether a particular machine preserves the original information 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 computer. We adopt this notion to describe precisely situations which are linked up to realizations of 'big' machines by partition 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

λ': $X' \times Z' \to Z'$, λ' and λ identical on $X' \times Z'$;

δ': $X' \times Z' \to Y'$, δ' and δ identical on $X' \times Z'$.

Definition: Let $\mathcal{X}$ 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 $\mathcal{X}$. Define $\bar{\lambda}: \mathcal{X} \times Z \to Z$. Then two machines M_1 and M_2 with identical input and output alphabets are said to be equivalent (state equivalent) iff $\bar{\lambda}_1(z_1,\bar{x}) = \bar{\lambda}_2(z_2,\bar{x})$ for all $z_1 \in Z$, $z_2 \in Z_2$ and $\bar{x} \in \mathcal{X}$. 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 sets Z_1, Z_2 respectively are equivalent iff $\{M_{z_1} : z_1 \in Z_1\} = \{M_{z_2} : z_2 \in Z_2\}$, i.e. 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 statesince 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

$$h_1[\lambda(z,x)] = \lambda'[h_1(z),h_2(x)] \quad ,$$

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

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

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

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) by its homomorphic image M', for example by placing a combinational circuit in front and back of M' (see Fig. 9).

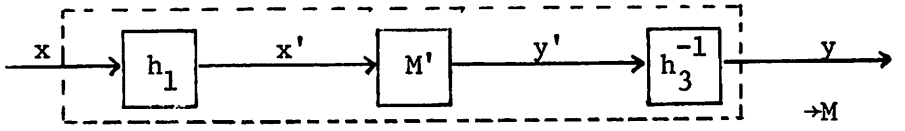


Fig. 9: Realization of M by M' via combinational circuit.

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 homomorphic π-images of M, say M_π. M_π could be thought 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 M_{π_2}. It is helpful to represent M, M_{π_1} and M_{π_2} by the flow table:

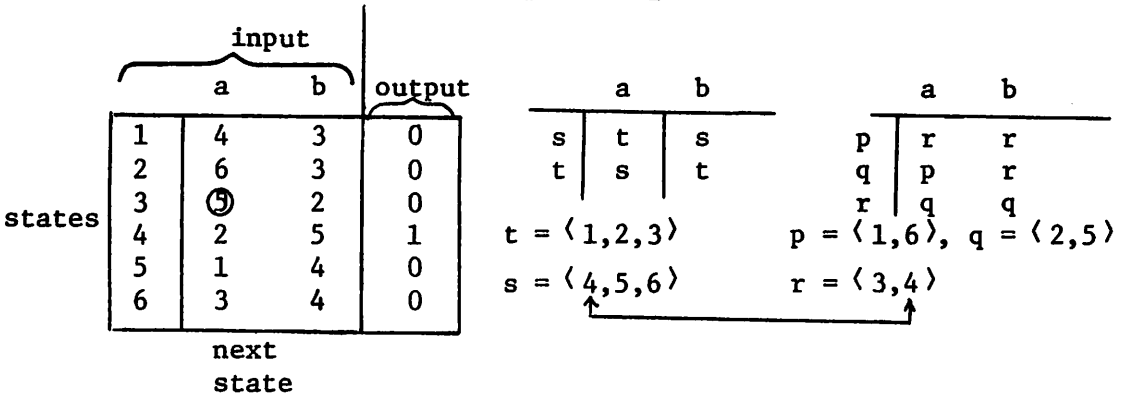


Fig. 10

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 '.' 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. $\Pi_1^n(\pi_i) = \pi_1 \cdot \pi_2 \cdot \ldots \cdot \pi_n$ forms the coarsest among all finest partitions in P, likewise the l.u.b. $\Sigma_1^n(\pi_i) = \pi_1 + \pi_2 + \ldots + \pi_n$ forms the finest among all coarsest partitions in P. Since P is a lattice it is perfectly legitimate to conceive the operations '.' and '+' or the

generalized operations 'Π' and 'Σ' as partial algebraic operations (see [1]). In this case the equivalence holds: $\pi_1 \leq \pi_2$ iff a l.u.b. (π_1,π_2) and a g.l.b. (π_1,π_2) exists in P for any π_1, $\pi_2 \in P$. Viewing the lattice of partitions as 'information technology' we would like to give '$\leq$' 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 $z \overset{-}{=} z'$ given 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 one would have to consider a sublattice of partitions if it exists. With regard 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 an immediate consequence 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.[2]

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

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

and a maximal partition

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

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 $\mathcal{P} \subseteq L_1 \times L_2$. We call $\mathcal{P}$ the pair algebra[7] satisfying a closure, completeness and boundedness property e.g.

a) (π_i,π_j) and (π_i',π_j') in $\mathcal{P}$ imply that

$\Pi_i\{(\pi_i,\pi_j),\ (\pi_i',\pi_j')\}$ and $\Sigma_i\{(\pi_i,\pi_j),\ (\pi_i',\pi_j')\}$ are in $\mathcal{P}$.

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

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

Obviously, $\mathcal{P}$ is again a lattice under the natural partition ordering $\leq$ since $(\pi_1,\pi_2) \leq (\pi_1',\pi_2')$ in $L_1 \times L_2$ is equivalent to $\pi_1 \leq \pi_1'$ in L_1 and $\pi_2 \leq \pi_2'$, and $\mathcal{P}$ 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) $\mathcal{P}_0$ of $\mathcal{P}$ 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 $\mathcal{P}_0$ in an algorithmic fashion, i.e. by refing 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 $\mathcal{P}$ given by the computed partition pair (π,π') as illustrated in Fig. 11.

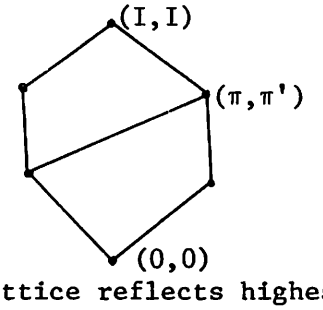


Fig. 11. Dimension of sublattice reflects highest informational efficiency or minimal information needed to realize M.

We could define a dimension function as a function

D: $\mathcal{P} \rightarrow [0,1]$ with the following properties:

(i) $0 \leq D[\pi,\pi')] \leq 1$ for every $(\pi,\pi') \in \mathcal{P}$, 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 $D(\overset{n}{\underset{1}{\cup}} (\pi_i,\pi_i')) = \overset{n}{\underset{1}{\Sigma}} (D(\pi_i,\pi_i'))$

(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 $\mathcal{P}$, D can be shown to be unique.[8)]

The algorithm then contains the following instruction. Choose that (π,π') in $\mathcal{P}$ which has minimal dimension in terms of D. Of course, in case $\mathcal{P}$ represents a metric lattice D would be identical to a metric on $\mathcal{P}$. Again, the economic analogue of this procedure can be easily presented, it relates to the problem of how much and what kind of informational decantralization 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 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 an 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 effective for all kinds of computations, i.e. independent of the size and complexity 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

1) C is endowed with a tolerance relation R, saying that for any pair of elements $(c,c') \in R$. (Any costs should be feasible with the given tolerance R)
2) For each $c \in C$ there exist $a, b \in C$ such that $a < c < b$ and $a < c' < b$ implies $(c,c') \in R$.

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_o and z_1 be two states of Z_π and Z_π, respectively, called the initial and the terminal state. We say that $x_\tau = (x_1,\ldots,x_n) \in X$ transfers M' from z_o to z_1 if $\lambda(z_o,x_\tau) = z_1$ for all x_τ, whereby $\lambda(z_o,x_1,\ldots,x_k) \neq z_1$ if $k < n$. Among all such x_τ in X_τ find that sequence

$z_{\pi_1},\ldots,z_{\pi_n}$ for which $\phi(x_\tau,z_\pi)$ is a minimum.

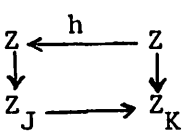
Let then $M' = \langle X_\tau, Y_\omega, Z_\pi \times C, \lambda_{\tau\pi}, \delta_{\tau\omega} \rangle$ be the machine with cost function ϕ and cost space C. We define the machine $(M',\phi) = \langle X_\tau, Y_\omega, Z_\pi \times C, \lambda^c_{\tau\pi}, \delta^c_{\tau\omega} \rangle$ by $\lambda^c_{\tau\pi}(z_\pi,c,x_\tau) = (\lambda_{\tau\pi}(z_\pi,x_\tau), c + \phi(z_\pi,x_\tau))$, and $\delta^c_{\tau\omega}(z_\pi,c,x) = \delta_{\tau\omega}(z_\pi,x)$. Some of these ideas will come up later if we turn to generalized adjustment processes.

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 <u>preserved</u> partitions which keeps most of the information of the former partition. This can be made clear by the following construction

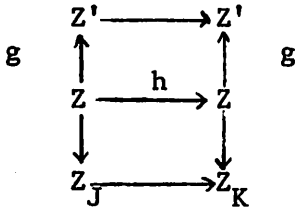


K, J index sets $K \subseteq J$,

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 diagram:



to commute.

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 $\mathcal{S} = \{S_i\}$ 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 $\mathcal{S}$, and no block properly contains another block although blocks may be overlapping. If blocks in $\mathcal{S}$ 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 beginning,

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'',... 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-thoeretic 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

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

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 <u>computational capability</u> containing a slight generalization of the notion of realization by machines. Take again the simplest case of a state machine.

<u>Definition</u>: 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 assignemtn (α,β) such that

a) α: $Z' \rightarrow$ ($\mathcal{S}$ is the class of non-empty disjoint subsets of Z)

b) β: $X' \rightarrow \mathcal{X}(\mathcal{X}$ 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.

Footnotes

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 $\mathcal{X}$. 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 $1 \leq i \leq n$. In this case the state set is not explicitly considered, although it is generated by 'real-time 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 unlimited 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.'

6) 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 multi-stage 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.

References

[1] H. W. Gottinger, Qualitative Information and Comparative Informativeness, Kybernetik 13, 1973, 81-94.

[2] J. Hartmanis 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] L. Hurwicz, On Informationally Decentralized Systems, Ch. 14 in: Decision and Organization (R. Radner and C. B. McGuire, eds.) North-Holland: Amsterdam 1972.

[5] C. B. McGuire and T. Marschak, Design for Organizations (unpublished notes: University of California, Berkeley 1971).

[6] C. B. McGuire, Information Technology (unpublished notes: University of California, Berkeley 1972).

[7] S. Reiter and K. Mount, The Informational Size of Message Spaces, Center for Math. Studies in Economics and Management Science, Northwestern University, Evanston, Ill., Discussion Paper No. 3, 1972.

[8] J. Rhodes, Applications of Automata Theory and Algebra (unpublished notes: University of California, Berkeley, 1973).