THE UNIVERSITY OF MICHIGAN
COLLEGE OF LITERATURE, SCIENCE, AND THE ARTS
Computer and Communication Sciences Department

Technical Report

TOWARDS A FORMAL THEORY OF MODELING AND SIMULATION. I.

Bernard P. Zeigler

with assistance from:

Department of Health, Education, and Welfare
National Institutes of Health
Grant No. GM-12236
Bethesda, Maryland

and

National Science Foundation
Grant No. GJ-519
Washington, D.C.

administered through:

OFFICE OF RESEARCH ADMINISTRATION ANN ARBOR

July 1970

TOWARDS A FORMAL THEORY OF MODELING AND SIMULATION.I*

Bernard P. Zeigler
Department of Computer and Communication Sciences
The University of Michigan
Ann Arbor, Michigan

SUMMARY

A system (automaton) theoretic formulation of the structure underlying a simulation is given. It consists of a ternary relation involving a system S to be simulated, an abstract model M and a computer C, and morphisms from S to M and C to M. A notion of automaton morphism, called structure preserving, is defined and shown to be stronger than the classical automaton behavior and automaton homomorphism concepts. Relevant properties of structure preserving morphisms are presented.

This formulation is used to discuss the complexity of systems and models vis-a-vis computer simulation. Two general classes of complexity measures are defined: the cumulative class M_Σ and the maximization class M_{max}. M_Σ attributes to a crossproduct function the sum of the complexities of its components while M_{max} attributes to the function the maximum complexity of its components. In either of the modes of simulation - sequential or parallel, at least one of a time or space measure is of the form M_Σ.

An important subclass of the M_Σ measures is shown to have the property that a structure preserving morphic image of an automaton is never more,

---

* This work was done while the author was an Institute of Science and Technology postdoctoral fellow. It was originally presented as a paper at the VI'th Annual International Congress on Cybernetics, Namur, Belgium, under the title," On The Formulation of Problems in Simulation and Modeling in the Framework of Mathematical System Theory".

and often less, complex than the automaton itself. This property is not shared by M_{max} measures nor by weaker morphisms. These results are employed to discuss the necessity of simulating a simplified model of a system, rather than the system itself, when both time and space resources are limited.

1. Introduction

The widespread use of electronic computers for simulating natural and artificial systems has given impetus to the development of a host of simulation languages and techniques. However, problems often arise in modeling and simulation which are more fundamental than problems peculiar to the particular computer, language or technique employed. Such problems would profit from the development of a more embracing conceptual framework for their understanding and solution. These more fundamental problems arise for example, when dealing with systems of great complexity where simulating the system in detail would be overly demanding of computer resources. Under such circumstances, it is necessary to construct a model of the system which is simpler with respect to computer implementation. At the same time, the model must preserve interesting properties of the system if it is to be at all useful.

My interest in the abstract structure of simulation grew out of my collaboration with Roger Weinberg in his computer simulation of a living cell (Weinberg (1970), Weinberg and Zeigler (1970)). We found the system theoretic notions of homomorphism and aggregation of state coordinates helpful in explicating the structure of his model, its ability to predict interesting properties of simulated cell behavior and its reduced complexity vis-a-vis computer implementation (Zeigler and Weinberg, 1970).

This work has been extended to a more inclusive formulation of the simulation concept. Broadly conceived, the elements of a simulation are: a system S to be simulated, an abstract model M of this system and a computer C on which the simulation is to be carried out. A triadic relation holds

such that the model stands simultaneously in a relation, call it M-S, which determines what properties of the system are to be preserved in the model and a relation M-C, which governs the manner in which the model is implemented on the computer.

Each of the elements of the simulation S,M,C will be formalized as an automaton quintuple $A = \langle S_A, Q_A, O_A, M_A, N_A \rangle$. Here S_A, Q_A, O_A are sets (input symbols, internal states, output symbols, respectively) and $M_A: Q_A \times S_A \rightarrow Q_A$ $N_A: Q_A \rightarrow O_A$ are the transition function and output function.

[The essential assumption made in this formulation of dynamic systems is that of discrete time operation; finiteness of S_A, Q_A, O_A is not implied.

By formalizing the system to be simulated as an automaton in the same manner as its models, I am making certain assumptions concerning the existence of a "best" model for the system S (which after all is known only behaviorally, not structurally) and the identification of the system with its best model. For further discussion of this point see Suppe (1967) and Zeigler (1970)].

Formal approaches to modeling have identified the M-S relation with automaton homomorphism i.e., the preservation of transition and output functions (Ashby (1956), Klir and Valach (1967)). However, an adequate account of the M-S relation must recognize that a "continuum" of system aspects may or may not be preserved in the model (c.f., Klir, 1970). In this continuum, the automaton homomorphism lies midway between the weakest, behavior preservation and the strongest, structure preservation.

In fact, one can set up chains $R_1 < R_2 < R_3 \ldots$ of binary preservation relations over automata where $R_i < R_j$ just in case R_j properly refines R_i. The position of R_n is this chain (i.e., the value of n) will reflect its strength as preservation relation. Thus,

if AR_1A' means that A can simulate the behavior of A' and AR_2A' means that A' is a homomorphic image of A, then $R_1 < R_2$, [it is known that if A' is a homomorphic image of A then A can simulate the behavior of A' but the converse proposition is not true unless A' is reduced (e.g., Arbib, 1969)]. The lower end of these chains, i.e., behavior preservation and automaton homomorphism, are well known. But the higher end, that of structure preservation, has heretofore received little attention. Since the latter is crucial to my formulation of simulation I have developed it to some depth.

Justification for this formulation, its relation to prior work, its applicability to real situations and some of the mathematical elaboration necessary to its articulation have been given considerable attention and will be presented elsewhere (Zeigler, 1970). In this paper, attention will be focused on the use of this formalism to examine some issues relating to the complexity of simulations. Accordingly, the necessary background development will be given only in outline form.

The plan of this paper is as follows: Function and structure morphisms are outlined and their relevant properties described in Section 2. Section 3 formulates the structure underlying a simulation and distinguishes sequential and parallel modes of simulation. In Section 4, complexity measures are defined and interpreted in terms of the utilization of time and space resources by structured functions. The behavior of these complexity measures under function and structure morphisms is then discussed. These results are then employed to discuss the necessity of simulating a simplified model of a system, rather than the system itself, when both time and space resources are limited.

2. System Function and Structure Preserving Morphisms

In this paper, a structured automaton will be a mathematical representation of a network of automata (e.g., a network of formal neurons). [It will not be the most detailed representation possible, however, since it does not represent the wire identifications by which the net is formed (c.f., Holland, 1969).]

An automaton $A = \langle S_A, Q_A, O_A, M_A, N_A \rangle$ is structured by coordinatizing each of the sets S_A, Q_A, O_A; i.e., expressing each via a cross-product of a family of sets; and (consistent with this structure) expressing each of the functions M_A, N_A via a cross-product of a family of coordinate functions. (c.f., Zeigler (1968), Hartmanis and Stearns (1966)).

The relation which expresses the preservation of structure from a structured automaton A to a structured automaton A' will be called a structure preserving morphism. Such a morphism will preserve not only the transition and output functions of the abstract automaton but also, in a sense to be defined, the manner in which these functions arise out of the local coordinate functions in its structured representation.

2.0 Structured Functions

These concepts are most easily developed in terms of arbitrary functions $f: A \to B$.

The domain A is structured by injecting it into a cross-product of an indexed family $\{A_\alpha | \alpha \epsilon D_A\}$ i.e., $A \subseteq \underset{\alpha \epsilon D_A}{\times} A_\alpha$; D_A is the set of coordinates for A. (Similarly, the range $B \subseteq \underset{\beta \epsilon D_B}{\times} B_\beta$.)

Associated with a structured set A is a family of coordinate projections $\{P_\alpha | \alpha \in D_A\}$ where $P_{\alpha_i}: A \to A_{\alpha_i}$ is defined by $P_{\alpha_i}(a_{\alpha_1}, a_{\alpha_2}, \ldots, a_{\alpha_i}, \ldots) = a_{\alpha_i}$.

Given an indexed family of functions $\{f_i: A \to B_i | i \in E\}$ the cross-product function $\underset{i\in E}{\times} f_i: A \to \underset{i\in E}{\times} B_i$ is defined by $\underset{i\in E}{\times} f_i(a) = (f_1(a), f_2(a), \ldots, f_{|E|}(a))$. [$|E|$ is the cardinality of E.]

Using the cross-product we can extend the projections of a structured set A to project on all nonempty subsets: we have the set $\{P_C | P_C = \underset{\alpha\in C}{\times} P_\alpha, \phi \neq C \subseteq D_A\}$. (Thus for example, $P_{\{1,3\}}(a_1, a_2, a_3)) = (a_1, a_3)$.)

$f: A \to B$ is structured by first giving structures to A and B and then presenting an indexed family of functions $\{f_\beta | f_\beta: (\underset{\alpha\in I_\beta}{\times} A_\alpha) \to B_\beta, \beta \in D_B, I_\beta \subseteq D_A\}$ such that $f = \underset{\beta\in D_B}{\times} f_\beta$.* The latter equation is in fact a set of $|D_\beta|$ equations of the form

$$P_\beta(f(a)) = f_\beta(P_{\alpha_1}(a), \ldots, P_{\alpha_{|I_\beta|}}(a))$$

or more compactly using functional composition ($\circ$)

$$P_\beta \circ f = f_\beta \circ P_{I_\beta}.$$

Given any function g with domain $X \subseteq \underset{\alpha\in D}{\times} X_\alpha$, we say that g depends on a set $C \subseteq D$ if for all $x, x' \in X$

$$P_C(x) = P_C(x') \Rightarrow g(x) = g(x').$$

C is said to be a location for g, written loc g, if g depends on C and for any $C' \subseteq C$, if g depends on C' then $C' = C$ (C is a minimal dependence set). More than one location for a function may exist (see Zeigler, 1968). A function g with domain $X \subseteq \underset{\alpha\in D}{\times} X_\alpha$ is in reduced form if D is a location for g. Every function may be put in reduced form (nonuniquely).

---

* The parentheses indicate that f_β need be defined only on $P_{I_\beta}(A)$. I will omit them hereafter.

The structured function $f:A \to B$ is in reduced form if every f_β is in reduced form.

The dependencies underlying the structure of f can be displayed as a directed graph called the dependency digraph representing f. It has as points $D_A \cup D_B$ and there is a line from $\alpha \varepsilon D_A$ to $\beta \varepsilon D_B$ just in case $\alpha \varepsilon I_\beta$ i.e., coordinate α is in the input set of function f_β.

Structuring of functions can be done recursively. For example, for $f:A_1 \times A_2 \to B$, one assigns structures separately to A_1, A_2 and B. Thus, if $\{A^i_\alpha | \alpha \varepsilon D_{A_i}\}$ coordinatizes A_i, i=1,2 then $\bigcup_{i=1,2} \{A^i_\alpha\}$ coordinatizes $A_1 \times A_2$ via $A_1 \times A_2 \subseteq \underset{\alpha \varepsilon D_{A_1}}{\times} A^1_\alpha \quad \underset{\alpha \varepsilon D_{A_2}}{\times} A^2_\alpha$.

In this way, one can assign a structure to an automaton <S,Q,O,M,N>.

2.1 Function and Structure Preserving Morphisms

Now follows some results concerning preservation relations holding between functions $f:A \to B$ and $f':A' \to B'$ in their abstract and structured forms.

A function preserving morphism (briefly, function morphism) from f to f' is a pair (h_A, h_B) where $h_A:A \to A'$ (onto), $h_B:B \to B'$ (onto) and $f' \circ h_A = h_B \circ f$.

Applied to automata, the function morphism becomes the usual homomorphism.

A structure preserving morphism (briefly, structure morphism) from f to f' (structured functions) is a 4-tuple $(d_A, d_B, \{h_{\alpha'} | \alpha' \varepsilon D_{A'}\}, \{h_{\beta'} | \beta' \varepsilon D_{B'}\})$ where

1. $d_A:D_A \to D_{A'}$ (onto), $d_B:D_B \to D_{B'}$ (onto);

2. for each $\alpha' \varepsilon D_{A'}$, $h_{\alpha'}: \underset{\alpha \varepsilon (\alpha')}{\times} A_\alpha \to A'_{\alpha'}$ (onto) and for each

$\beta' \varepsilon D_{B'}$, $h_{\beta'}: \underset{\beta \varepsilon (\beta')}{x} B_\beta \rightarrow B'_{\beta'}$ (onto)

(here $(\alpha') = \{\alpha | d_A(\alpha) = \alpha'\}$ and $(\beta') = \{\beta | d_B(\beta) = \beta'\}$);

3. for each $\beta' \varepsilon D_{B'}$, $f'_{\beta'} \circ h_{I'_{\beta'}} = h_{\beta'} \circ f_{(\beta')}$

(here $h_{I'_{\beta'}} = \underset{\alpha' \varepsilon I'_{\beta'}}{x} h_{\alpha'} \circ P_{(\alpha')}$ and

$f_{(\beta')} = \underset{\beta \varepsilon (\beta')}{x} f_\beta \circ P_{I_\beta}$);

4. for each $\beta' \varepsilon D_{B'}$, there exists a location for $f'_{\beta'} \circ h_{I'_{\beta'}}$ $(= h_{\beta'} \circ f_{(\beta')})$,

call it loc $f'_{\beta'} \circ h_{I'_{\beta'}}$, such that loc $f'_{\beta'} \circ h_{I'_{\beta'}} \subseteq (I_{\beta'}) \cap I_{(\beta')}$

(here $(I'_{\beta'}) = \underset{\alpha' \varepsilon I'_{\beta'}}{\cup} (\alpha')$ and $I_{(\beta')} = \underset{\beta \varepsilon (\beta')}{\cup} I_\beta$).

Visualization of structure morphism* may be aided by (Figure 1) and the following fact:

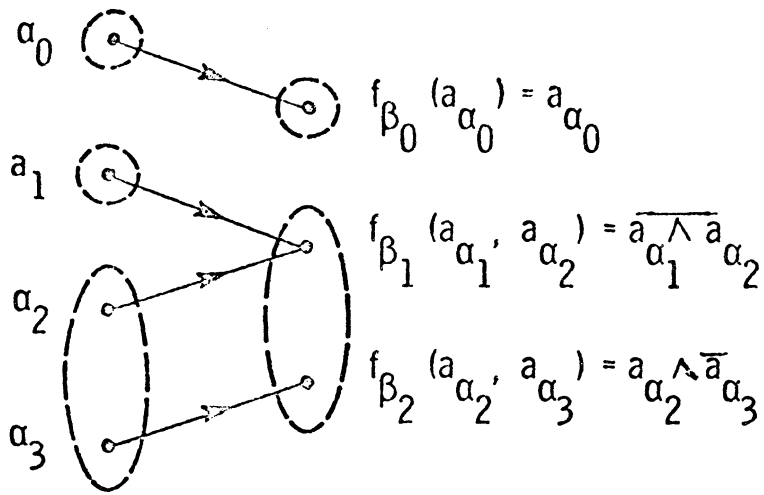
Every structure morphism can be expressed as a composition of an I-morphism followed by a D-morphism.

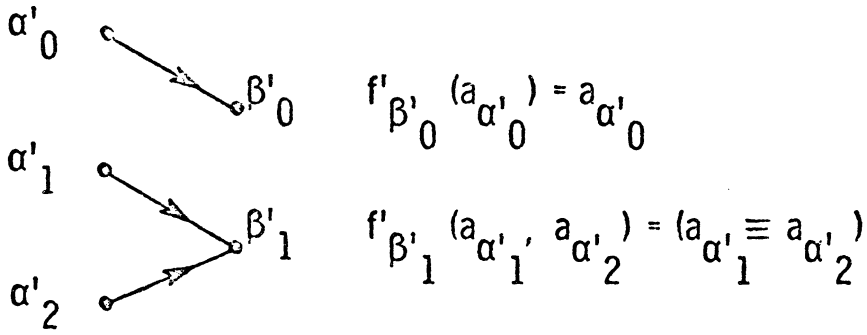
A morphism $(d_A, d_B, \{h_{\alpha'}\}, \{h_{\beta'}\})$ is an integration morphism (I-morphism) if $\{h_{\alpha'}\}$ and $\{h_{\beta'}\}$ are all one-one (as well as onto). In effect, functional characteristics are not altered in this process but coordinate functions are lumped into larger functionally equivalent blocks. This describes essentially the integration process employed in integrated circuit design, (e.g., Habayeb (1968)).

A morphism $(d_A, d_B, \{h_{\alpha'}\}, \{h_{\beta'}\})$ from f to f' is a digraph morphism (D-morphism) if d_A and d_B are one-one (as well as onto maps). The thing to notice here is that the coordinates $D_A \cup D_B$ can be regarded as common to f and f'. When f' is reduced, the digraph representing f' turns out to

---

* This is a precise formulation of the process of valid aggregation of coordinates discussed in Zeigler & Weinberg (1970); see also Simon (1961), Ijiri (1968).





	α_0	α_1	α_2	α_3		f_{β_0}	f_{β_1}	f_{β_2}	f
a_0	0	0	0	0		0	1	0	b_0
a_1	0	0	1	1		0	1	0	b_0
a_2	0	1	0	1		0	1	0	b_0
a_3	0	0	1	0		1	1	1	b_1
a_4	1	1	1	1		1	0	0	b_2

A

B

	$h_{\alpha'_0}$	$h_{\alpha'_1}$	$h_{\alpha'_2}$	h_A	$h_{\beta'_0}$	$h_{\beta'_1}$	h_β
a_0	0	0	1	a'_0	0	0	b'_0
a_1	0	0	1	a'_0	0	0	b'_0
a_2	0	1	0	a'_1	0	0	b'_0
a_3	0	0	0	a'_2	1	1	b'_1
a_4	1	1	1	a_3	1	1	b'_1

B'

	α'_0	α'_1	α'_2	$f'_{\beta'_0}$	$f'_{\beta'_1}$	$f'_{\beta'_2}$
a'_0	0	0	1	0	0	b'_0
a'_1	0	1	0	0	0	b'_0
a'_2	0	0	0	0	1	b'_1
a'_3	1	1	1	1	1	b'_1

A'

Figure 1: Illustrating two structured functions f and f' and a structure preserving morphism running from f to f'. It may be verified directly that all four conditions for structure morphismhood are satisfied e.g.,
$f'_{\beta'_1}(h_{\alpha'_1}(\alpha_1), h_{\alpha'_2}(\alpha_2,\alpha_3)) = h'_{\beta_1}(f_{\beta_1}(\alpha_1,\alpha_2), f_{\beta_2}(\alpha_2,\alpha_3))$.
In this case, $(I_{\beta'_1}) = \{\alpha_1,\alpha_2,\alpha_3,\} = I_{(\beta'_1)}$ but this need not always be the case.

be a subdigraph of the digraph representing f (see Theorem 1). It may happen, though it is not necessary, that every line of the former is also a line of the latter in which case the digraphs are isomorphic [thus in this case the structure morphisms reduce to the morphisms of heterogeneous algebra (Birkoff and Lipson, 1969), c.f., also the notion of similarity of functions (Klir and Valach, 1967)].

[An isomorphism (in the category theory definition e.g., Mitchell (1965) of structure morphisms turns out to be both an I-morphism and a D-morphism].

Both function morphisms and structure morphisms form categories i.e., identity morphisms exist, morphisms from f to f' and from f' to f'' compose to form a morphism from f to f'' and such composition is associative.

Any structure morphism from f to f' (structured functions) induces a function morphism from f to f' (regarded as abstract functions). However, the converse is not the case. Thus the structured morphism is a stronger preservation relation between automata than is homomorphism, as alluded to in Section 1.

This assymetry derives in part from the following property of structure morphisms not shared by function morphisms:

Theorem 1: *If there is a structure morphism running from f to f', where f, f' are in reduced form, then the digraph representing f' is a subdigraph of a condensation of that representing f, i.e., for each* $\alpha' \varepsilon I_{\beta'}$, *there exists* $\alpha \varepsilon (\alpha')$ *and* $\beta \varepsilon (\beta')$ *such that* $\alpha \varepsilon I_{\beta}$.

Thus even though a function morphism may run from f to f' (as abstract functions) there can be no corresponding structure morphism unless the digraphs match up as in Theorem 1. (For example, see Figure 2).

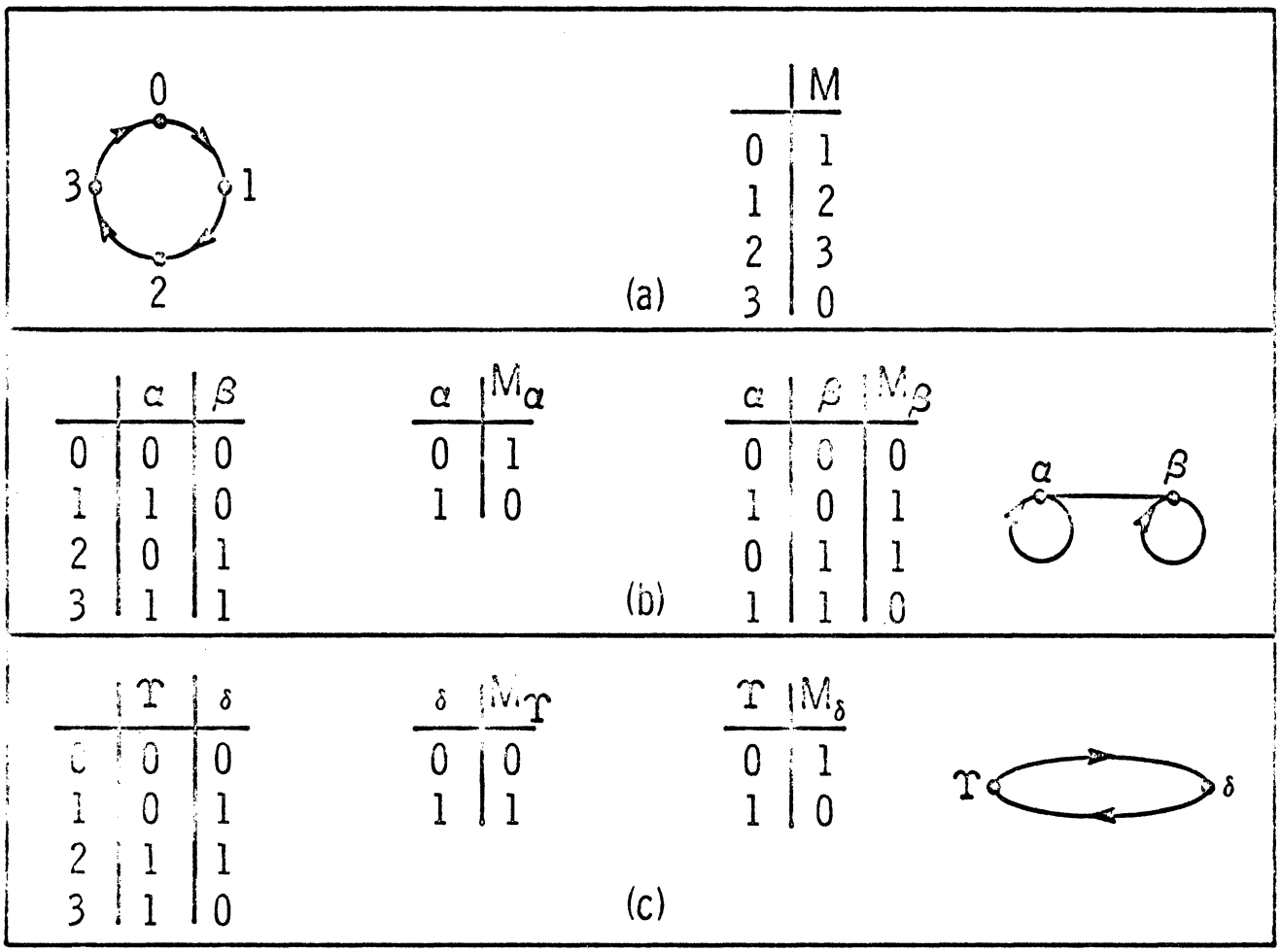


Figure 2: The cycle in a) is realized by the nonisomorphic structures in b) and c). There can be no structure preserving morphism between these structures even though the transition functions are isomorphic. For example, the identity map, i is such that loc $i_\Upsilon = \beta$ and loc $i_\delta = \{\alpha, \beta\}$.

An even stronger distinction exists:

There are pairs of functions f,f' such that a) f is structured, b) there is a *function* morphism from f(abstract) to f', but c) no nontrivial *structure* assignment exists for f' which will enable a *structure* morphism to run from f to f'. (For example, see Figure 3).

This completes the exposition of structure morphism. Proofs and details will be given by Zeigler (1970).

3.0 Basic Formulation of Simulation Structure

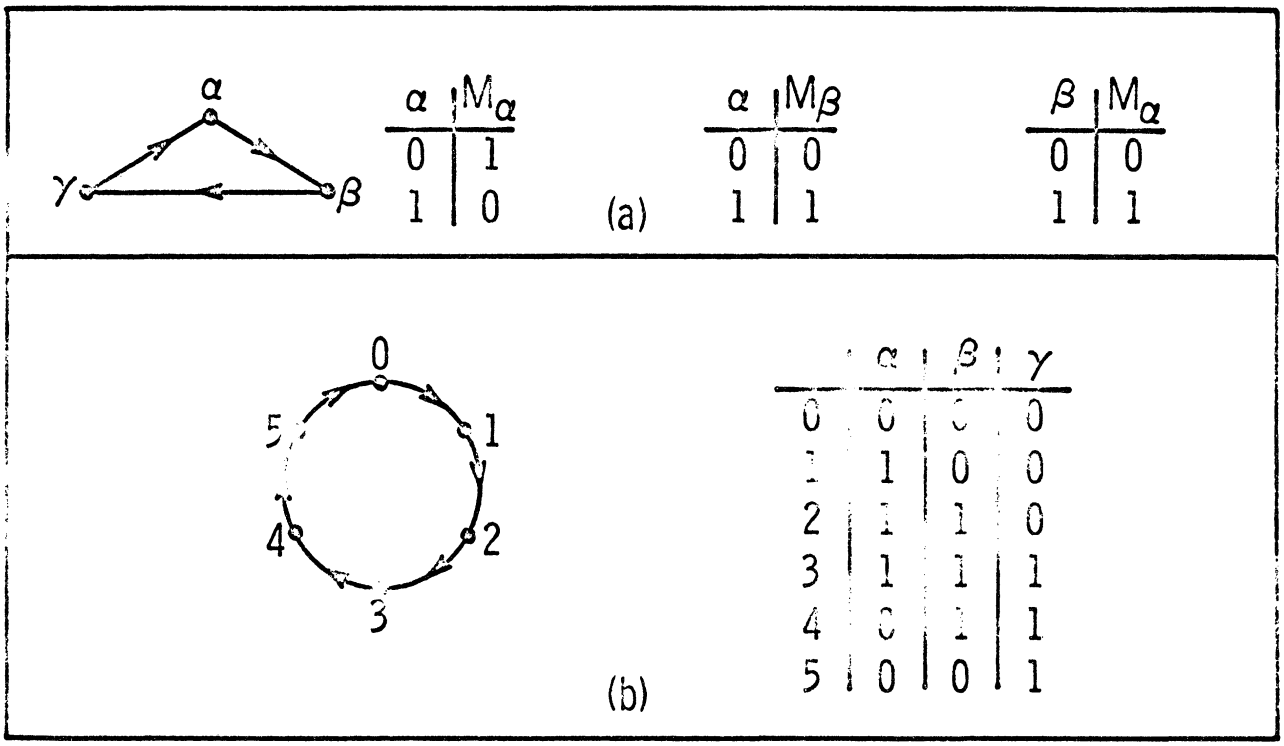
The following concept allows us to take into account the time scale and the state space region of system operation.

An automaton $A = \langle S_A, Q_A, O_A, M_A, N_A \rangle$ will be said to be a macro subautomaton of an automaton $B = \langle S_B, Q_B, O_B, M_B, N_B \rangle$ if the following conditions obtain:

1) S_A is in one-one correspondence with a subset of S_B^+ (the set of nonempty finite sequences over S_B); let i denote this correspondence.
2) $Q_A \subseteq Q_B$
3) $O_A \subseteq O_B$
4) $M_A(q,s) = \tilde{M}_B(q,i(s))$ for all $q \in Q_A, s \in S_A$ ($\tilde{M}_B$ is the extension to S_B^+ of M_B).
5) $N_A(q) = N_B(q)$ for all $q \in Q_A$.

The basic mathematical formulation of the simulation relation employed in this paper can now be presented. It consists of the following:

$\mathcal{S} = \{A_s\}$ - a set of macro subautomata of an automaton S (the system to be simulated);



α	M_α
0	1
1	0

α	M_β
0	0
1	1

β	M_α
0	0
1	1

	α	β	γ
0	0	0	0
1	1	0	0
2	1	1	0
3	1	1	1
4	0	1	1
5	0	0	1

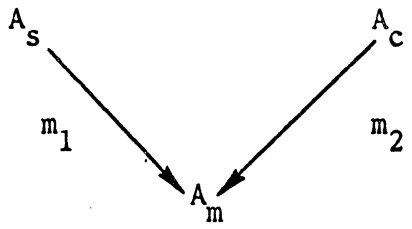
Figure 3: The structure shown in a) gives rise to the cycle of order 6 in the transition graph in b). The partition $\Pi_h = \{\overline{0,3};\ \overline{1,4};\ \overline{2,5}\}$ has substitution property and represents a homomorphism of the 6 cycle on a 3 cycle.

A nontrivial structure morphism must separate one of the coordinates from the other two, e.g., $u = \{\overline{\gamma;\alpha,\beta}\}$ and moreover $\Pi'_{[\gamma]} = \Pi_\gamma$ ($|\Pi_\gamma| = 2$). But it is not the case that $\Pi_h \leq \Pi_\gamma$ since 0 and 3 are in the same Π_h block but not in the same Π_γ block; this is also true of Π_α and Π_β. Thus, no nontrivial structure morphism can exist corresponding to the partition Π_h.

$M = \{A_m\}$ - a set of automata (the potential models of S);

$C = \{A_c\}$ - a set of macro-subautomata of an automaton C (the computer on which the simulation is carried out).

An instance of a simulation takes the form:



where $A \xrightarrow{m} B$ means that a morphism of kind m runs from automaton A to automaton B.

In explanation: an automaton $A_s \varepsilon S$ is the formal counterpart of a particular experiment (or series of experiments) on system S. In such an experiment only a subset of the state space may be operative and the intervals at which the system is observed may be of longer duration than its assumed basic time scale units (c.f., the definition of macro subautomaton).

Thus, the model A_m need not account for the complete system S but only one of its macro subsystems A_s appropriate to the experiments of interest.

The realization of the model by the computer corresponds to the usual notion in automaton theory of the "simulation" of one machine by another (e.g., Herman (1968), Arbib (1969)). In the macro subautomaton formulation, we permit the computer to take a number of time steps in simulating a single step of the model. Also, it is clear that only a portion of the computer state space may perhaps be operative in realizing the model and this portion may be larger than the model state space.

In sum, the formulation $A_c \xrightarrow{m} A_m$ employed here is a reformulation of a common notion of "simulation" in computer science in which time scale

change and memory expansion are formulatable concepts.

As alluded to earlier, the kind of morphism running from A_s to A_m is the formal counterpart of the preservation relation holding between the system, or better macro-subsystem, and the model. Thus, this morphism, designated m_1 in the diagram, may be behavior preserving, transition and output function preserving (the usual automaton homomorphism) or structure preserving in increasing degrees of strength.

The essence of computer simulation of a model is the iterative computation of model behavior in a chronological time sequence (c.f., Evans et al. (1967)). Formally, this is captured in the formulation of the model as an automaton. (This is opposed, for example, to the computation of closed form solutions of difference equations where the "model" would be a one argument (time) function.) Moreover, in the usual situation, C is a general purpose computer. In this case, the model structure (i.e., transition function and output function possibly in structured forms) is used to program or configure the computer. Formally, this is represented by requiring that m_2 (the morphism from A_c to A_m) be at least transition and output function preserving, and if the model is structured, structure preserving as well.

In working with this formulation, we can make one further abstraction which is justifiable in view of the recursive nature of the structure concept. The basic simulation formulation is reduced to the following elements:

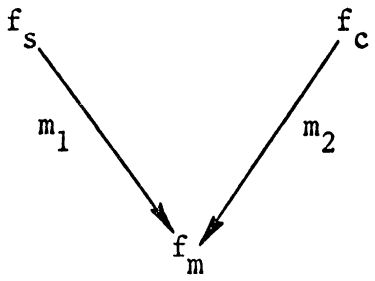
$F_S = \{f_s\}$ - a set of functions from which the transition and output functions of the macroautomata in A_s are taken.

$F_M = \{f_m\}$ - a set of functions bearing the same relation to the set A_M.

$F_C = \{f_c\}$ - a set of functions bearing the same relation to the set A_C.

An instance of a simulation involves two triples (for transition

and output functions) each of the form:



In this paper, attention will be focused on the case where m_1, m_2 are either function or structure preserving morphisms.

3.1 Sequential and Parallel Simulation

Two modes of simulation are commonly employed - sequential and parallel (e.g., Parnas, 1969). In this paper, we shall consider the following formulation of these concepts:

A simulation is parallel if the coordinate functions in the family $\{f_{c,\beta}\}$ structuring f_c are computed simultaneously and synchronously.

A simulation is sequential if $\{f_{c,\beta}\}$ is a (totally) ordered set and the computation of the $f_{c,\beta}$ proceeds according to this order.

Thus in a *parallel* simulation the output of f_c (e.g., the next state of the computer) will appear at its correct value at a time commensurate with the computation time of the slowest component function, $f_{c,\beta}$.

In contrast, in a *sequential* computation the output will appear at its correct value at time commensurate with the sum of the computation times of the component functions.

Those semiformal definitions will suffice for the further purposes of this paper. It is interesting to remark, however, that more complete definitions together with the above formalization of simulation structure allow one to deal formally with the sequential simulation of parallel

systems [e.g., the simulation by digital computer of cellular (tessellation) automata (Brender, 1970) and the parallel systems so modeled, e.g., Goodman et al. (1970), Mortimer (1970)].

4.0 Complexity Measures and their Interpretation

I will be concerned with the following notion of complexity for functions.

μ_G will denote a functional, parameterized by G, a set of functions, such that $\mu_G(f)$ is a measure of the complexity of a function f.

G is to satisfy the conditions:

1) For each $n \in N$ (the nonnegative integers) there is a (unique) $g_n \in G$ such that $g_n : N^{n+2} \to N$.

2) The $\{g_n\}$ are nondecreasing in the sense that

if $n \geq m$, and $x_i \geq y_i$ for $i=1,2,\ldots m$

then $g_n(x_1,x_2,\ldots x_m,\ldots x_n) \geq g_m(y_1,y_2,\ldots y_m)$.

To see how this works, let f be a function of n variables e.g., $f: \underset{\alpha \in I}{X} A_\alpha \to B$, where $|I| = n$. Then $\mu_G(f) = g_n(n, |B|, |A_{\alpha_1}|, \ldots, |A_{\alpha_n}|)$.

The complexity measures under consideration can be sensitive to the number of function variables n, the cardinalities of their ranges $|A_{\alpha_i}|$ and the cardinality of the image range $|B|$. Because they are functionals assumed to be everywhere defined these measures do not satisfy the Blum size or step-counting axioms (Blum, 1967); see also Strong (1970).

The dependence of complexity on the number of function variables n has received attention in the literature. To illustrate this consider the case where the $\{g_n\}$ depend only on n.

G will be said to be a g-class if there is a function $g:N \to N$ such that for each $n \in N$, $g_n(x_1,x_2,\ldots,x_{n+2}) = g(x_1)$, for all $x_i \in N$, $i=1,2,\ldots n+2$.

Consider the following cases:

a) constant growth, $g(n) = 1$ for all $n \in N$.

This g-class will prove useful in counting the number of component functions f_i in a cross-product Xf_i.

b) linear growth, $g(n) = kn$.

This g-class measures the number of input wires to a module which directly realizes a n-variable function and is thus fundamental to "fan-in" approaches to complexity (e.g., Minsky and Papert (1969), Zeigler (1968)). This growth function is also related to the minimum number of modules of bounded fan-in r required in a circuit realization of an n-variable function where $n>>r$.

For linear functions $f(x_1,\ldots,x_n) = \sum_{i=1}^{m} a_i x_i$ (arising in the simulation of linear systems) this g-class measures the number of additions required to compute the function.

c) logarithmic growth $g(n) = k[\log_r n]$, where $[x]$ is the least integer $\geq x$.

This g-class is related to the minimum depth of a circuit realization of a function of n variables using modules having fan-in r. Under the assumption of a unit delay for each module it is also related to the computation time (see Winograd (1967), Spira (1969), Arbib (1969)).

d) exponential growth $g(n) = a2^{bn}$.

This g-class is related to the depth measure of complexity for parallel program schemata introduced by Strong (1970). It also measures the storage space required for a function of n (finite) variables which is stored in tabular form.

I now wish to extend the measure μ to apply to *structured* functions f (the subscript G will hereafter be suppressed). The essential difference is that now we have a *family* of functions $\{f_\beta \mid \beta \in D_B\}$ (rather a single function) which are combined via the cross-product operation to realize the function f.

Two forms for such extensions will be considered. The first, denoted M_Σ is a cumulative measure defined by

$$M_{\Sigma}(f) = \sum_{\beta \epsilon D_B} \mu(f_{\beta});$$

the second, M_{max} is a maximization measure given by

$$M_{max}(f) = \max_{\beta \epsilon D_B} \{\mu(f_B)\}.$$

These forms reflect contrasting aspects of the computation of a structured function and the nature of such computation. As illustrated in Table 1, there is a kind of duality obtaining between sequential and parallel computation on the one hand, and time and space measures on the other.

In the table, complexity measures are considered to be divisible into two classes - space and time (nonexhaustive in view of the "hybrid" forms such as Strong's).

In the case of space measures, let us distinguish between state storage and function storage. State storage refers to the memory which holds the present and next value of the state of the computation. Function storage refers to the realization of the functions $\{f_{c,\beta}\}$ - by programs or subroutines in the case of sequential computation, or by modules in the case of parallel computation.

State storage is appropriately measured by M_{Σ} in both the parallel and sequential cases. One can determine, for example, the number of coordinates of the range of the structured function using the constant g-class ($g(n) = 1$) i.e., $M_{max}(f) = \sum_{\beta \epsilon D_B} \mu(f_{\beta}) = \sum_{\beta \epsilon D_B} 1 = D_B$.

(Note that in the case of the transition function, the range is the internal state space.)

Table 1 indicates that for *sequential* computations and *time* measures, the appropriate complexity measure for structured functions is M_{Σ}. This is so, since in a sequential computation the structured function is computed

Mode of Simulation \ Measure	Time	Space	
		Function Storage	State Storage
Sequential	M_Σ	$M_{max}; M_\Sigma$	M_Σ
Parallel	M_{max}	M_Σ	M_Σ

Table 1: The measure, cumulative (M_Σ) or maximizing (M_{max}) appropriate to the mode of simulation and complexity measure of a structured function.

by successively computing each of the f_β, the time taken being assumed to be the sum of the times required for each. In contrast, for *parallel* computations (all f_β computed simultaneously) the *time* required is assumed to be that of slowest computation (c.f., Winograd (1967), Spira (1969)) so the appropriate measure for structured functions is M_{max}.

However, M_Σ is appropriate for *space* measures of *parallel* realizations. This assumes that the set of computing elements required for simultaneous computation of a set of functions is the disjoint union of the elements required in the computation of the individual functions.

The symmetry of the table breaks down somewhat with respect to space measures for sequential computation. M_{max} is appropriate for function storage only under certain circumstances, which are none-the-less important. One such case is when all f_β are isomorphic, as in a cellular space simulation, so that only one function need be stored ($M_{max} = \mu(f_\beta)$ is this case).

A second application is to the amount of core (as distinct from disc storage) required if an "overlaying" procedure is used. Here only the program for the f_β currently being computed is stored in core in a block which need only be as big as the largest of the program sizes for the individual functions.

However, if all f_β are distinct, the total storage required would be commensurate with the sum of the individual storage requirements hence M_{max} would be appropriate.

Of course, one can think of situations in which the above interpretations fail (when the whole is not the sum of the parts). I offer them as good first approximations to actual situations and hence a strong motivation for studying abstractly the M_Σ, M_{max} measures introduced.

4.1 Behavior of Complexity under morphisms

The kind of questions I shall now ask are illustrated by the following: If it is known that a morphism runs from f to f' does it follow that the complexity of f' is less than that of f? Is it necessary to find a morphic image of f in order to reduce its complexity?

Often it is tacitly assumed that these questions are answerable in the affirmative. However, we shall see that this is generally not the case, and that only under certain circumstances can such assertions be made.

Before proceeding, it is noteworthy that the measures M_Σ and M_{max} assign the same complexity value to all the *reduced* functions in a structure morphism isomorphism class. That is, if reduced functions f and f' are isomorphic in the structure morphism category then $M_\Sigma(f) = M_\Sigma(f')$ and $M_{max}(f) = M_{max}(f')$.

In this sense M_Σ and M_{max} reflect properties of the category of *reduced* functions with structure morphisms.

This is not true when *nonreduced* functions are considered. However, the reduced functions play a central role in the wider class of nonreduced functions in the sense that if f and f' are isomorphic with respect to structure morphisms and f is reduced then $M_\Sigma(f) \leq M_\Sigma(f')$ and $M_{max}(f) \leq M_{max}(f')$.

With the above in mind, I shall restrict attention to reduced functions in what follows.

Let M be any complexity measure for functions - here I shall be concerned with $M \in \{M_{max}, M_\Sigma\}$, and let $f \xrightarrow{m} f'$ indicate that a morphism of type m (function preserving, structure preserving, etc.) runs from f to f'.

M will be said to be nonincreasing with type m morphisms if

$$f \xrightarrow{m} f' \Rightarrow M(f') \leq M(f).$$

M will be nondecreasing with type m morphisms if

$$f \overset{m}{\rightarrow} f' \Rightarrow M(f) \geq M(f'),$$

(for all functions f,f' and morphisms of type m).

M will be ambivalent with type m morphisms if it is neither nonincreasing nor nondecreasing (with type m morphisms). In other words, M is ambivalent just in case there is at least one function with a morphic image having higher complexity and one function with a morphic image having lower complexity.

Theorem 2: M_Σ, M_{max} are both nonincreasing with D-morphisms.

Proof: Let $(d_A, d_B, \{h_\alpha\}, \{h_{\beta'}\})$ be any D-morphism from $f:A \rightarrow B$ to $f':A' \rightarrow B'$.

$$M_\Sigma(f') = \sum_{\beta \epsilon D_B} \mu(f'_\beta) \qquad \ldots 1)$$

$$= \sum_{\beta \epsilon D_B} g_{|I'_\beta|}\left(|I'_\beta|, |B'_\beta|, |A'_{\alpha_1}|, \ldots, |A'_{\alpha_{|I'_\beta|}}|\right) \qquad \ldots 2)$$

$$\leq \sum_{\beta \epsilon D_B} g_{|I_\beta|}\left(|I_\beta|, |B_\beta|, |A_{\alpha_1}|, \ldots, |A_{\alpha_{|I_\beta|}}|\right) \qquad \ldots 3)$$

$$= \sum_{B \epsilon D_B} \mu(f_\beta) \qquad \ldots 4)$$

$$= M_\Sigma(f) \qquad \ldots 5)$$

That line 3) follows from line 2) is justified as follows:

Since f' is reduced, $I'_\beta \subseteq I_\beta$ so that $|I'_B| < |I_\beta|$, (Theorem 1 and discussion in Section 2.1).

Since the maps in $\{h_\alpha\}$ and $\{h_\beta\}$ are onto we have $|A'_{\alpha_i}| \leq |A_{\alpha_i}|$ for $i = 1, \ldots, |I'_\beta|$ and $|B'_\beta| \leq |B_\beta|$.

So line 3) follows from line 2) by the nonincreasing property (2) of the set G applied to each component β.

The proof for M_{max} is similar.

In contrast to the D-morphism, the I-morphism may entail increasing complexity. Recall that if an I-morphism runs from $f:A \to B$ to $f':A' \to B'$ then $|A| = |A'|$ and $|B| = |B'|$. Thus to the extent that lumping coordinates together decreases the number of coordinates (i.e., $|D_{A'}| < |D_A|$) the cardinality of the coordinate state sets must be correspondingly increased (i.e., $|A'_\alpha| = \underset{\alpha\varepsilon(\alpha')}{X} |A_\alpha|$). The coordinate state sets for the component functions f'_β, may thus be increased. Moreover, the indegrees of the points in the representing digraph may be increased by the condensation process i.e., some functions f'_β, may have more input variables than any of the functions f_β in (β').

Remembering that a general structure morphism is a composition of an I-morphism and a D-morphism we see that in order for the application of a structure morphism to result in a decrease in complexity, any increase in complexity introduced by the I-morphism must be compensated by a complexity decrease by the D-morphism. We will see that we cannot always be assured that this will happen.

To carry the discussion further, I will first consider only sets of complexity measures G which are sensitive only to the number of variables on which the function to be measured depends i.e., the g-classes of Section 4.0.

With respect to these measures it can be shown that if the underlying function g(n) does not grow too fast with n, then M_Σ is nonincreasing with structure morphisms.

The allowable rate of growth will be specified by requiring

$$g(n+m) \leq g(n) + g(m) \qquad \ldots *)$$

for all $n, m \in N$.

This is clearly satisfied by the constant function $g(n) = 1$ and the linear function $g(n) = kn$. It also holds for the logarithmic function $g(n) = k[\log_r n]$ (except possibly where n or m = 1; this can be accommodated by agreeing that $[\log_r 1] = 1$, which in actual fact is in accord with any physical interpretation of circuit complexity). It does not hold for the exponential function.

Theorem 3: Let g(n) be growth bounded according to *). Any M_Σ (based on g) will be nonincreasing with structure morphisms.

Proof: Let $(d_A, d_B, \{h_\alpha\}, \{h_\beta\})$ be a morphism from f to f'.

$$M_\Sigma(f') = \sum_{\beta' \in D'_{B'}} \mu(f'_{\beta'}) \qquad \ldots 1)$$

$$= \sum_{\beta' \in D'_{B'}} g(|I'_{\beta'}|) \qquad \ldots 2)$$

$$\leq \sum_{\beta' \in D'_{B'}} g(|I_{(\beta')}|) \qquad \ldots 3)$$

$$\leq \sum_{\beta' \in D'_{B'}} g\Big(\sum_{\beta \in (\beta')} |I_\beta|\Big) \qquad \ldots 4)$$

$$\leq \sum_{\beta' \in D'_{B'}} \sum_{\beta \in (\beta')} g(|I_\beta|) \qquad \ldots 5)$$

$$= \sum_{\beta \in D_B} g(|I_\beta|) \qquad \ldots 6)$$

$$= \sum_{\beta \in D_B} \mu(f_\beta)$$

$$= M_\Sigma(f)$$

In justification of this sequence: By Theorem 1, $|I'_{\beta'}| \leq |I_{(\beta')}|$ so that using the nonincreasing property of g, line 3) follows from line 2).

Since $I_{(\beta')} = \bigcup_{\beta\varepsilon(\beta')} I_\beta$ we have $|I_{(\beta')}| \leq \sum_{\beta\varepsilon(\beta')} |I_\beta|$ and by the same g property, line 4) follows from line 3).

Line 5) follows from line 4) by the growth bounding assumption *).

Line 6) follows from line 5) because the (β') are blocks of a partition, so that summing first within blocks and then over blocks picks up every point in D_B once and only once.

In one tries to push through the proof of Theorem 3 for M_{max} one is brought to a halt at line 5). The appropriate inequality $g(|\cup I_\beta|) \geq \max_{\beta\varepsilon(\beta')} g(|I_\beta|)$ runs in the direction opposite to that required to carry the proof through. This suggests the following

Assertion 1: M_{max} is ambivalent with structure morphisms.

Figure 4 shows the representing digraphs of function pairs f,f' such that f' is a structure morphic image of f. In Figure 4 a), $M_{max}(f') \leq M_{max}(f)$ while in Figure 4 b), $M_{max}(f') \geq M_{max}(f)$.

These results are readily extended to measures based on more general G sets provided the set of morphisms considered is restricted as follows:

Define the classes of structure functions C_p, p = 0,1,2,..., where $C_p = \{f:A \to B | A \subseteq \{0,1,...p-1\}^n, B \subseteq \{0,1,...p-1\}^m$, for some $m,n\varepsilon N\}$.

For example C_2 consists of functions whose domain and range are binary coded n-tuples.

Call a structure morphism from f to f' admissible provided that $f\varepsilon C_p$ and $f'\varepsilon C_q$ where $q \leq p$. (Note that I-morphisms which are not isomorphisms are not admissible. Thus, admissible morphisms which are not D-morphisms

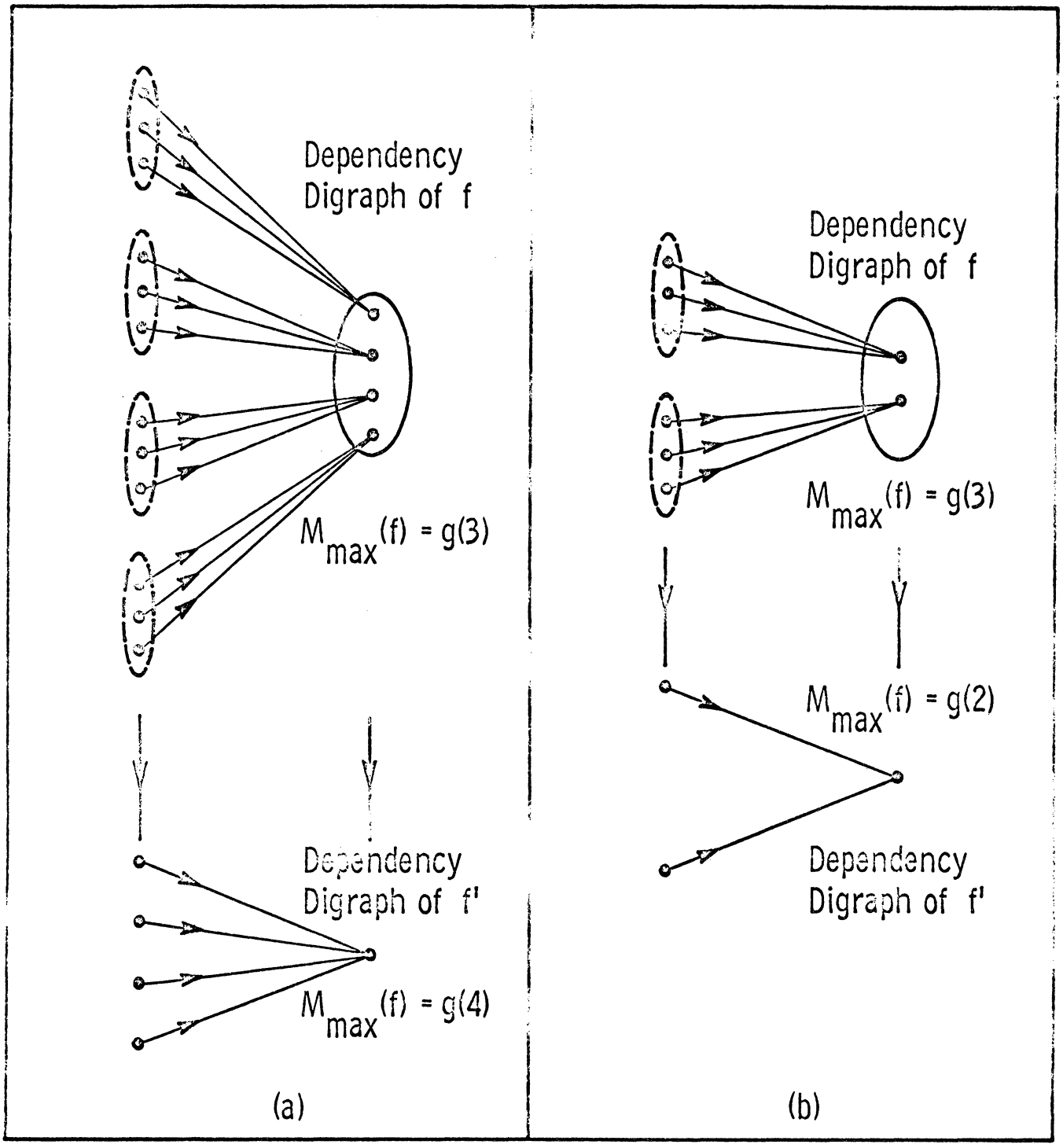


Figure 4: Illustrating the ambivalence of complexity measure M_{max} with structure morphisms. The representing digraphs of two pairs (f,f') are given such that a structure morphism runs from f to f'. In a) the complexity of f' is greater than that of f, while in b) the opposite is the case.

cannot be expressed as compositions of admissible I-morphisms and D-morphisms.)

By elaborating on the proof of Theorem 3 we can prove

Theorem 4: Let G be a measure set which is growth bounded as follows:

$$g_n(x_1+y_1,x_2,\ldots,x_{n+2}) \leq g_n(x_1,x_2,\ldots,x_{n+2}) + g_n(y_1,x_2,\ldots,x_{n+2})$$

for each $n\varepsilon N$ and $x_i \varepsilon N$, $i = 2,\ldots,n+2$.

Then any M_Σ (based on G) will be nonincreasing with admissible structure morphisms.

It remains true that M_{max} is ambivalent with admissible morphisms.

The foregoing results for structure morphisms are also suggestive in the case of function preserving morphisms. Recall that every structure morphism from f to f' induces a function morphism from f to f', but not conversely. From this it follows immediately that since M_{max} is ambivalent with structure morphisms it is also ambivalent with function morphisms.

On the other hand, the nonincreasing property of M_Σ for structure morphisms does not imply that the same is true for function morphisms. The possibility that function morphism not arising out of structure morphisms may permit increasing complexity is suggested by the strong dependence of lines 2) and 6) in the proof of Theorem 2 on properties of structure morphisms not shared by function morphisms. In fact, Figure 2 demonstrates that M_Σ is ambivalent for function morphisms (even for g measures which are growth bounded). (Note that while structure isomorphic reduced functions are assigned the same M_Σ measure, this is not the case for structure functions which are function isomorphic (but not structure isomorphic).)

Table 2 summarizes the above discussion.

It is still open as to whether, and to what degree, the growth bounded

Type of Morphism \ Measure of Complexity		M_{Σ}	M_{max}
Structure Preserving	D-morphism	Non-increasing	Non-increasing
	General, Admissible	Non-increasing	Ambivalent
Function Preserving		Ambivalent	Ambivalent

Table 2: The behavior of complexity measures under morphisms. Complexity is either nonincreasing (morphic image may be less complex) or ambivalent (morphic image sometimes more, sometimes less, complex).

condition *) on G sets can be relaxed in the proofs of Theorems 3 and 4.

4.2 Complexity of Systems and Models

Up to this point, we have been considering what might be called "formal complexity", in that the complexity of an object is determined by examining the object itself independently of its relation to other objects. We have seen that for these measures it is not always the case that complexity decreases in going from a function to one of its morphic images.

There is, however, a notion of complexity based on the "formal" concept for which this is true almost as a matter of definition. In this conception, the complexity of an object is taken to be the minimum complexity — measured formally — of a class of related objects (e.g., Arbib (1968)).

The "minimization" concept of complexity is of interest here since it characterizes the complexity of systems and models vis-a-vis computer simulation.

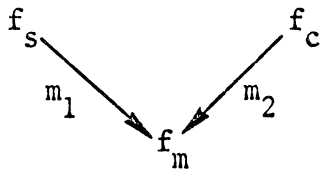
Recall that the formalization of simulation involved three classes of functions:

$F_S = \{f_s\}$ - a class of functions representing macro subsystems of the given system S.

$F_M = \{f_m\}$ - a class of functions representing models M of S.

$F_C = \{f_c\}$ - a class of functions representing macro subsystems of a computer or class of computers C.

Any instance of a simulation of system S using Computer C using model M is taken to be of the form:



where the arrows indicate the existence of function preserving or structure preserving morphisms.

For any function f, let

$$[f]_m = \{f_c \in F_C \mid \text{there is a morphism of type m from } f_c \text{ to } f\}.$$

Minimizing over such a set with respect to a "formal" complexity measure $M \in \{M_\Sigma, M_{max}\}$ results in a complexity measure

$$C[f]_m^M = \min_{f_c \in [f]_m} M(f_c).$$

Interpretively, $C[f]_m^M$ represents the least amount of resources (as determined by the measure M) that is required in order to simulate a function f, in a manner indicated by the morphism m, on a computer C.

Assertion 2: If morphisms of type m compose then

1. $f \xrightarrow{m} f' \Rightarrow [f']_m \subseteq [f]_m$;
2. $f \xrightarrow{m} f' \Rightarrow C[f']_m^M \leq C[f]_m^M$,

i.e., "minimizing" complexity measures are nonincreasing with type m morphisms (for any measure M).

Proof: Immediate.

In particular, in the context of simulation, the requirement that type m morphisms compose, implies that if a computer configuration f_c can simulate f_s (a system) in a manner m, it can also simulate f_m (a model of the system) in the same manner. Under these conditions, the complexity of the model vis-a-vis computer simulation is never greater than the complexity of the system*

More than this can be said when the dependence of the measure on morphism type is taken into account.

---

* Note, however, that Assertion 1 depends crucially on the composition of morphisms. In the simulation triple (f_s, f_m, f_c) where m_1, m_2 are morphism types which either do not compose or are such that $m_1 \circ m_2$ is of type m_1, one can no longer assert that the complexity of the model is not greater than that of the system.

Assertion 3:

1. If M is nonincreasing with type m morphisms then $C[f]_m^M \geq M(f)$

2. If M is nondecreasing with type m morphisms then $C[f]_m^M \leq M(f)$

Proof: Immediate.

4.3 The Necessity of Abstractive Models

These assertions help understand necessity for the use of models in simulation.

For assume that the amounts of computational resources available for any given simulation are limited. Formally, for each measure M, there is given an integer ℓ such that only the subset of F_c given by

$$F_c^{\ell} = \{f_c \in F_C | M(f_c) < \ell \}$$

can be considered.

For example, if M measures time, then ℓ may be the longest time one can wait for a computation to be completed (or the longest computer run time one can pay for). If M measures space, then ℓ may describe the size of the computer core and auxiliary stores.

Suppose also that the system to be simulated is more complex with respect to M than this limit i.e., $M(f_s) \geq \ell$.

If M is *nondecreasing* or *ambivalent* with type m morphism it may yet be possible to simulate the system in manner m (since by Assertion 2 there may exist $f_c \in F_C^{\ell}$ such that $f_c \overset{m}{\rightarrow} f_s$).

However, if M is *nonincreasing* with type m morphisms then it is *impossible* to simulate the system in manner m. (For suppose there is an $f_c \in F_C^{\ell}$ such that $f_c \overset{m}{\rightarrow} f_s$. Then by Assertion 3 , $M(f_c) \geq M(f_s) > \ell$, a contradiction.) Under these circumstances, the only recourse open is to

construct a model of the system which *is* simple enough to simulate. Formally, this corresponds to finding a model function f_m such that $f_s \overset{m}{\rightarrow} f_m$ and $M(f_m) < \ell$. (Since M is nonincreasing this condition is open to satisfy; its necessity follows from Assertion 2.)

Now, a class of measures of the form M_Σ which are nonincreasing with structure morphisms was described in Sections 4.0 and 4.1. These include function storage measures for parallel simulation, time measures for sequential simulation and state storage measures for both modes of simulation. The results of this section are directly applicable, and establish the necessity for employing simplifying models when the associated space and time resources are limited.

Recall that the M_{max} measures (including space measures for sequential simulation and time measures for parallel simulation) were ambivalent with structure morphisms. Our discussion allows the possibility that system simplification is not necessitated by limitations placed on these measures in isolation. E.g., when only time is limited in a parallel simulation it may be possible to find a computer configuration f_c such that $f_c \overset{m}{\rightarrow} f_s$ and $M_{max}(f_c) < \ell_{M_{max}}$, where $\ell_{M_{max}}$ is the time limitation. Note, however, that if the morphism in question is a structure morphism, then for any nonincreasing space measure M_Σ , we will have $M_\Sigma(f_c) \geq M_\Sigma(f_s)$.

In short, in reducing time complexity we may have to "trade-off", i.e., increase, space complexity.

When, as is realistically the case, *both* time and space resources are limited, at least one will involve an M_Σ type of complexity measure (Table 1) which, if it is nonincreasing with structure morphisms, will necessitate system simplification via modeling.

Conclusions

The structure morphisms formalized here are interesting because of their relation to aggregation and structure preserving techniques commonly employed in modeling. The tacit assumption that such techniques always result in models of reduced complexity has been shown to be incorrect. The reason for this is that such morphisms are compositions of complexity increasing and complexity decreasing processes.

Classes of complexity measures have been isolated for which application of a structure morphism may always be expected to reduce complexity. The rate of growth allowed by these measures, however, was limited such as to include linear and logarithmic growth, but to exclude polynomial and exponential growth. It appears that much more work is necessary to understand the behavior of these faster growing complexity measures. The development of a finer categorization of morphism types than given here would provide a better control for prediction of the complexity increasing or decreasing properties of morphisms. The base provided by the structure preserving morphism concept makes this a mathematically realistic possibility.

Acknowledgement

Ideas germane to this paper arose in stimulating contacts with Michael Arbib, Roger Weinberg and Guisepi Trautteur. Michael Arbib raised the question concerning complexity reduction under homomorphism in a personal correspondence.

REFERENCES

Arbib, M. A. (1968) Algebraic Theory of Machines, Languages, and Semigroups, Academic Press, New York.

___, (1969) Theories of Abstract Automata, Prentice-Hall, Inc. New York.

Ashby, W. R. (1956) An Introduction to Cybernetics, John Wiley, New York.

Birkoff, G. and Lipson, J. D. (1970) "Heterogeneous Algebras," Journal of Combinatorial Theory, Vol. 8, No. 1.

Blum, M. (1967) "A Machine-independent Theory of the Complexity of Recursive Functions," J. Assoc. Comp. Mach., Vol. 14.

Brender, R. F. (1970) "A Programming System for the Simulation of Cellular Spaces," University of Michigan Doctoral Dissertation.

Evans, G. W., II.; Wallace, G. F.; and Sutherland, G. L. (1967) Simulation Using Digital Computers, Prentice-Hall, Inc.

Goodman, E. D.; Weinberg, R.; and Laing, R. A. (1970) "A Cell Space Embedding of Simulated Living Cells," Proc. of Summer Computer Simulation Conference, Denver, Colorado.

Habayeb, A. (1968) "System Decomposition, Partitioning and Integration for Micro-electronics," IEEE Trans. on Systems Science and Cybernetics.

Hartmanis, J., and Stearns, R. E. (1966) Algebraic Structure Theory of Sequential Machines, Prentice-Hall, Inc., New Jersey.

Herman, G. T. (1968) "Simulation in the Theory of Computing Systems," University of London Doctoral Dissertation.

Holland, J. H. (1970) "Hierarchical Descriptions, Universal Spaces and Adaptive Systems," In Essays on Cellular Automata. A. W. Burks, Editor. University of Illinois Press.

Ijiri, Y. (1968) "The Linear Aggregation Coefficient as the Dual of the Linear Correlation Coefficient," Econometrica, Vol. 36.

Klir, G. J. and Valach, M. (1967) Cybernetic Modelling, Iliffe Books, Princeton, New Jersey.

___, (1969) An Approach to General Systems Theory, Von Nostrand, New York.

Minsky, M. and Papert, S. (1969) Perceptrons, M.I.T. Press, Cambridge, Mass.

Mitchel, B. (1965) Theory of Categories, Academic Press, New York.

Mortimer, J. A. (1970) "A Computer-Simulated Model of Mammalian Cerebellar Cortex," University of Michigan Technical Report 03296-3-T.

Parnas, D. L. (1969) "On Simulating Networks of Parallel Processes in Which Simultaneous Events May Occur," Journal of the Assoc. for Comp. Mach. Vol. 12, No. 9.

Simon, H. A. and Ando, H. (1961) "Aggregation of Variables in Dynamic Systems," Econometrica, Vol. 29.

Spira, P. M. (1969) "The Time Required for Group Multiplication," J. Assoc. Comp. Mach. Vol. 16.

Strong, H. R. (1970) "Depth-Bounded Computation," J. of Computer and System Sciences, Vol. 4.

Suppe, F. R. (1967) "The Meaning and Use of Models in Mathematics and the Exact Sciences," University of Michigan Doctoral Dissertation.

Winograd, S. (1967) "On the Time Requried to Perform Multiplication," J. Assoc. Comp. Mach. Vol. 14.

Weinberg, R. (1970) "Simulation of a Living Cell: Interdisciplinary Synergism," University of Michigan Doctoral Dissertation.

Weinber, R. and Zeigler, B. P. (1970) "Simulation of a Living Cell: Multilevel Control Systems," J. of the American Society for Cybernetics.

Zeigler, B. P. (1968) "On the Feedback Complexity of Automata," University of Michigan Doctoral Dissertation.

___, (1969) "On the Feedback Complexity of Automata," Proc. Third Annual Princeton Conference on Systems and Infor. Sciences, Princeton, New Jersey.

Zeigler, B. P. and Weinberg, R. (1970) "System Theoretic Model Analysis: Computer Simulation of a Living Cell," Journal of Theoretical Biology.

Zeigler, B. P. (1970) "Towards a Formal Theory of Modeling and Simulation. II," (Manuscript in preparation).