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

Technical Report

A UNIFIED MINIMAL REALIZATION THEORY, WITH DUALITY, FOR MACHINES IN A HYPERDOCTRINE
(Announcement of Results)

E. S. Bainbridge

Mathematics
Faculty of Science and Engineering
University of Ottawa
Ottawa 2, Canada

with assistance from:

National Science Foundation
Grant No. GJ-29989X
Washington, D.C.

administered through:

OFFICE OF RESEARCH ADMINISTRATION ANN ARBOR

June 1972

Summary

This report gives a preliminary account of a new formulation of automata and system theory. Using Lawvere's notion of a hyperdoctrine, a single unified theory encompassing the minimal realization theories of sequential machines; algebra automata; a new generalization of algebra automata to non-equationally definable theories, giving arbitrary recursive computations as behaviours; linear systems; machines with algebraic structure; and machines with restricted input; is presented. In addition, the concept of a coordinatized machine is introduced, and a duality for such machines which is closely connected to the minimal realization theory is described. Coordinatized machines with each of the above types of structure may be defined; moreover, linear systems have a natural coordinatization relative to which the general duality specializes to the Kalman duality.

CONTENTS

0. Introduction

This report gives a preliminary account of results from the author's doctoral dissertation, currently in preparation.

In recent years there have been attempts to unify the minimal realization theories of various branches of automata and system theory, notably those of sequential machines and linear systems, as in [12]. Also, attempts have been made to unify the duality of linear systems theory with the apparent duality or partial duality between reachability and observability for automata, as in [9]. Last year, Goguen [2] achieved a unification of the minimal realization theories of sequential machines and affine systems, among others; and also initiated the study of adjunctions between behaviour and realization functors.

In this report I present another, quite different, formulation of machine theory which unifies automata theory and linear systems theory, and also contains a new type of duality for machines which specializes in the case of linear systems to give the familiar duality. The interesting features of this formulation are these:

- The mathematical apparatus used is that of a hyperdoctrine, which was introduced by Lawvere as a model of higher

order logic, proof theory, and sheaf theory. Machines and their behaviours, as well as other features of machine theory are expressed easily and naturally using the operations of a certain hyperdoctrine.

- Special cases of the type of machine considered include sequential machines; algebra automata; a new generalization of algebra automata to non-equationally definable theories, giving arbitrary recursive computations as behaviours; linear systems; machines with algebraic structure; and machines with restricted input.

- A general minimal realization theory is developed which specializes in each of the above cases to the known theory (if any). Also, a more comprehensive set of behaviour-realization adjunctions than has previously been discussed is presented.

- The notion of a coordinatized machine is introduced, and a general theory of duality for such machines is developed. These machines also have a canonical realization theory which is closely related to the realization theory of ordinary machines. Behaviour-realization adjunctions also carry over to coordinatized machines. A linear system has a natural coordinatization in this sense, and the Kalman duality is seen to be a special case of the duality for coordinatized machines.

- There is considerably more mathematical structure in the hyperdoctrine considered than is actually used in the

preliminary applications. I give some examples to show the system theoretic significance of some of this extra structure; and I think these examples indicate that we may expect significant new features to emerge on further investigation.

1. A Fundamental Example

Many of the observations of this section have been made elsewhere; but the crucial one, the use of the left and right adjoints to the forgetful functor from right actions to sets, is new. This formulation of sequential machine theory motivates the development of section 3.

Let X be a monoid. We denote a right action $(a, x) \mapsto a \cdot x : A \times X \to A$ of X on a set A by $(A, \cdot)$.

For any set A, we may define a right action of X on the set $A \times X$ by

$$(a, x) \cdot x' = (a, xx').$$

The assignment $A \mapsto (A \times X, \cdot)$ may be extended to a functor from the category of sets to the category of right X-actions. This functor is left adjoint to the forgetful functor from X-actions to sets.

For any set A, we may define a right action of X on the set A^X of functions from X to A by

$$(f \cdot x')(x) = f(x'x).$$

The assignment $A \mapsto (A^X, \cdot)$ may be extended to a functor from the category of sets to the category of right X-actions. This functor is right adjoint to the forgetful functor from right X-actions to sets.

These observations lead to an elegant formulation of machines and their behaviours, as follows.

Let $(Q, \cdot)$ be a right action of a monoid X of inputs on a set Q of states. Let $\alpha: I \to Q$ be a read-in function with domain I, a set of initial conditions. Let $\lambda: Q \to J$ be a read-out function with co-domain J, a set of final conditions.

Denote by sub u_X (this notation will be explained later) the forgetful functor from right X-actions to sets.

A machine may thus be formulated as a diagram

$$I \xrightarrow{\alpha} \text{sub } u_X\,(Q, \cdot) \xrightarrow{\lambda} J.$$

Using both adjunctions, we obtain

$$(I \times X, \cdot) \xrightarrow{\overline{\alpha}} (Q, \cdot) \xrightarrow{\overline{\lambda}} (J^X, \cdot)$$

Using the right adjoint to sub u_X we obtain

$$I \times X \xrightarrow{\overline{\lambda} \circ \overline{\alpha}} J.$$

This is precisely the behaviour of the machine, that is, the function assigning to each initial condition $i \in I$, and each input $x \in X$ the final condition $\lambda((\alpha i) \cdot x) \in J$ which results.

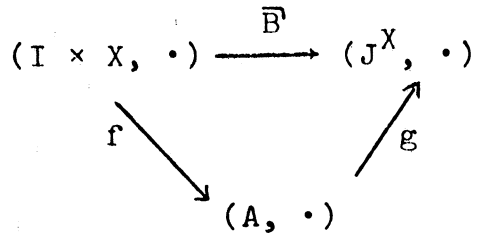
Conversely, given a behaviour

$$I \times X \xrightarrow{B} J$$

we have, via the right adjoint to sub u_X

$$(I \times X, \cdot) \xrightarrow{\overline{B}} (J^X, \cdot)$$

Any factorization of $\overline{B}$, say



may be interpreted, using both adjunctions, as the machine

$$I \xrightarrow{f} \text{sub } u_X\ (A, \cdot) \xrightarrow{g} J$$

Furthermore, the behaviour of this machine is precisely B.

Moreover, the condition that a machine

$$I \xrightarrow{\alpha} \text{sub } u_X\ (Q, \cdot) \xrightarrow{\lambda} J$$

be reachable, in the usual sense, is simply that

$$I \times X \xrightarrow{\text{sub } u_X\ \overline{\alpha}} Q$$

be onto; and the condition that the machine be observable, in the usual sense, is simply that

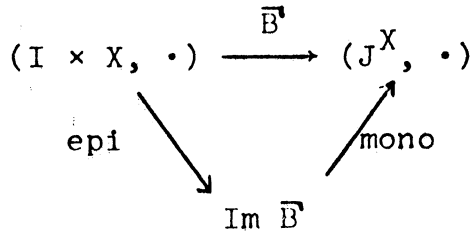
$$Q \xrightarrow{\text{sub } u_X\ \overline{\lambda}} J^X$$

be 1 - 1.

In the category of right X-actions, any morphism has a unique epi-mono factorization through its image (also coimage). In particular, given a behaviour

$$I \times X \xrightarrow{B} J$$

we factor $\overline{B}$ through its image



Since the forgetful functor has both right and left adjoints, it preserves monomorphisms and eipmorphisms, so that the resulting machine is both reachable and observable. Indeed, it is the familiar minimal realization of B.

2. Hyperdoctrines

Lawvere [1] introduced the notion of a hyperdoctrine to model, among other things, higher order logic. Much of the terminology derives from this interpretation.

Hyperdoctrines provide the appropriate setting in which to pursue the development of section 1. The use of the hyperdoctrine (cat, Sets) is not an empty generalization; it is essential to the applications we shall consider.

A cartesian closed category C has finite (including empty) products; and has, for each $A \in \mathrm{Ob}\ C$ a right adjoint

$$(\cdot)^A : C \to C$$

for the functor $(\ ) \times A : C \to C$. The functor $(\ )^A$ is called exponentiation by A. The empty product; that is, the terminal object, is denoted by 1.

A hyperdoctrine consists of:

- a cartesian closed category T, the category of types.
- for each object X of T, a cartesian closed category P(X) called the category of attributes of type X.
- for each morphism $f: X \to Y$ in T a functor $\mathrm{sub}\ f: P(Y) \to P(X)$ called substitution along f.

In the examples we consider, sub( ) is functorial, although in general one requires this only up to coherent natural isomorphism.

- for each morphism $f: X \to Y$ of types, left and right adjoints for sub f, $\Sigma f: P(X) \to P(Y)$ and $\Pi f: P(X) \to P(Y)$, called respectively existential and universal quantification along f.

The hyperdoctrine which shall be of most interest to us is the hyperdoctrine (cat, Sets) described below. The structure which appears in the hyperdoctrine (cat, Sets) is well known in the literature (under other names, in most cases). We omit details of the cartesian closed structure of the type and attribute categories, and of the existential and universal quantification functors.

The type category is the category cat of small categories.

The category P(X) of attributes of type X is the category of set-valued functors from X, with natural transformations as morphisms.

Substitution is composition: for $f: X \to Y$, $\Psi: Y \to$ Sets,

$$\text{sub } f\ (\Psi) = \Psi \circ f$$

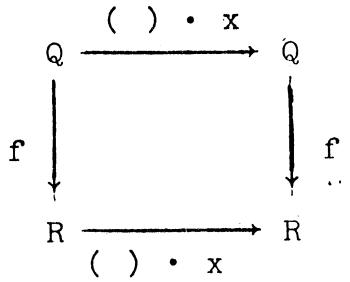
For each $f: X \to Y$, sub $f: P(Y) \to P(X)$ has a left adjoint $\Sigma f: P(X) \to P(Y)$ and a right adjoint $\Pi f: P(X) \to P(Y)$.

3. A Reformulation of Machine Theory

The contents of this section generalizes and extends that of section 1, and is new. First we recapture section 1, and then imitate that development in the hyperdoctrine (cat, Sets).

A monoid X may be viewed as a category with a single object.

Set valued functors from X^{op}, where X is a monoid viewed as a category, are precisely right actions of X (on the set to which the single object is mapped). Natural transformations are just morphisms of right actions, as one can see from the diagram



Furthermore, if $\underline{1}$ is the one object, one morphism category, $P(\underline{1})$ may be identified with Sets in an obvious way.

The crucial observation which establishes link between machine theory and the hyperdoctrine (cat, Sets) is the following. Define u_X: $\underline{1} \to X^{op}$ (the only functor). Then sub u_X: $P(X^{op}) \to P(\underline{1})$ is the forgetful functor from right X-actions to Sets. Σu_X: $P(\underline{1}) \to P(X^{op})$ assigns to each set A, the right action $(A \times X, \cdot)$ described above. Πu_X: $P(\underline{1}) \to P(X^{op})$ assigns to each set A the right action

$(A^X, \cdot)$ described above.

Thus all of the constructions of section 1 may be done using the operations of the hyperdoctrine (cat, Sets).

Indeed, we may define a machine $M = (E, u, X, \Phi, I, \alpha, J, \lambda)$ in (cat, Sets) to consist of a functor $u: E \to X$ which is onto objects, called the behaviour scheme of M (compare u_X); $\Phi \in \mathrm{Ob}\ P(X)$; $I, J \in \mathrm{Ob}\ P(E)$; and morphisms α, λ in $P(E)$ of the form

$$I \xrightarrow{\alpha} \mathrm{sub}\ u\ \Phi \xrightarrow{\lambda} J$$

We define a morphism $(h, f, k): M \to M'$ of machines over the behaviour scheme u to consist of $h: I \to I'$, $f: \Phi \to \Phi'$, $k: J \to J'$ such that the diagram

$$\begin{array}{ccccc} I & \longrightarrow & \mathrm{sub}\ u\ \Phi & \longrightarrow & J \\ \downarrow{\scriptstyle h} & & \downarrow{\scriptstyle \mathrm{sub}\ u\ f} & & \downarrow{\scriptstyle k} \\ I' & \longrightarrow & \mathrm{sub}\ u\ \Phi' & \longrightarrow & J' \end{array}$$

commutes.

We define a behaviour over $u: E \to X$ to be a morphism in $P(E)$ of the form

$$B: \mathrm{sub}\ u\ \Sigma\ u\ I \to J$$

for some $I, J \in \mathrm{Ob}\ P(E)$.

We define a morphism $(h, k): B \to B'$ of behaviours over the behaviour scheme u to consist of morphisms $h: I \to I'$, $k: J \to J'$ such that the diagram

$$\begin{array}{ccc} & & B \\ \text{sub } u\ \Sigma\ u\ I & \longrightarrow & J \\ \downarrow{\scriptstyle \text{sub } u\ \Sigma\ u\ h} & & \downarrow{\scriptstyle k} \\ \text{sub } u\ \Sigma\ u\ I' & \longrightarrow & J' \\ & & B' \end{array}$$

commutes.

We define the behaviour of a machine $M = (E, u, X, \Phi, I, \alpha, J, \lambda)$ to be the morphism

$$\beta(M) = \underline{\overline{\lambda}\ o\ \overline{\alpha}}$$

where $\overline{\ }$ etc indicate the transforms relative to the appropriate adjunctions. We may extend β to a functor.

A machine is <u>reachable</u> if sub u $\overline{\alpha}$ is an epimorphism and <u>observable</u> if sub u $\overline{\lambda}$ is a monomorphism.

We can then show the

<u>Theorem</u>. Every behaviour B: sub u Σ u I → **J is realized** by a unique (up to isomorphism) reachable and observalbe machine.

Denote by Mach(u), Rch(u), Obs(u), Beh(u) the categories of machines, reachable machines, observable machines, behaviours over a fixed behaviour scheme u.

Then we have the following adjunctions. Some of these adjunctions have not been discussed, even in the case of ordinary machines, as far as I know.

The functor βu: Mach(u) → Beh(u) has a left adjoint $\overline{\mu} u$: Beh(u) → Mach(u) and a right adjoint $\underline{\mu} u$: Beh(u) → Mach(u). As we saw in section 1, machines which realize a given behaviour B arise from factorizations of $\overline{B}$. $\overline{\mu} uB$ arises from

$$\Sigma\ u\ I \xrightarrow{1} \Sigma\ u\ I \xrightarrow{\overline{B}} \Pi\ u\ J$$

and in ordinary machine theory corresponds to the free monoid machine.

$\underline{\mu}uB$ arises from

$$\Sigma \cup T \xrightarrow{\overline{B}} \Pi \cup J \xrightarrow{1} \Pi \cup J.$$

This machine is the "all possible output functions" machine, with left translation as the action.

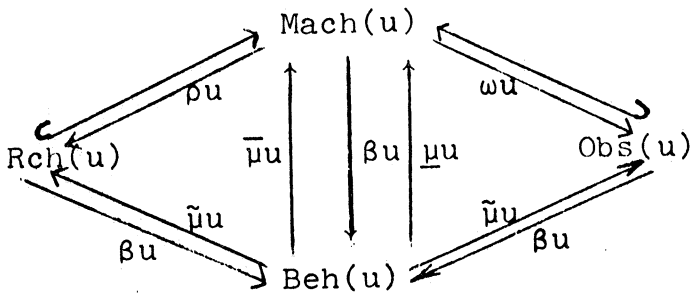
The category Rch(u) is coreflective, with the coreflector (right adjoint to inclusion) ρu: Mach(u) → Rch(u) which assigns to each machine its reachable part.

The category Obs(u) is reflective, with the reflector (left adjoint to inclusion) ωu: Mach(u) → Obs(u) which assigns to each machine its observable component.

The restriction of βu to Rch(u) has a right adjoint namely $\tilde{\mu} u$: Beh(u) → Rch(u) which assigns to each behaviour its reduced (reachable, observable) realization. This corresponds to the behaviour-realization adjunction discussed by Goguen [2].

The restriction of βu to Obs(u) has a left adjoint, namely $\tilde{\mu} u$: Beh(u) → Obs(u) which assigns to each behaviour its reduced realization.

These adjunctions may be summarized in the diagram



Except for the lower left adjunction, whose counterpart has been discussed by Goguen [2] for sequential machines with quite general algebraic structure on their state and input spaces, I do not know of any discussion of these adjunctions, even for ordinary machines.

4. Applications

Suitable specializations of machines in (cat, Sets) lead to various familiar types of machine. It should be kept in mind that the preceding minimal realization theory applies in each case, as well as the development which we shall see in later sections.

4.1 Sequential machines

Taking $u = u_X$, where X is a monoid, we obtain sequential machines with input monoid X, as we have seen.

4.2 Algebra Automata

An Ω-algebra, where $\Omega = \bigcup_{n=0}^{\infty} \Omega_n$ is a union of disjoint sets Ω_n of n-any operation symbols, is a set A together with functions $\psi_A: A^n \to A$ for each $\psi \in \Omega_n$, each n.

The free Ω-algebra on generators I, $\Omega(I)$, where I is any set consists of expressions formed as follows:

(i) $I \cup \Omega_0 \subset \Omega(I)$

(ii) if $\xi_1, \ldots, \xi_n \in \Omega(I)$ and $\psi \in \Omega_n$, then

$$\psi\xi_1 \ldots \xi_n \in \Omega(I).$$

In particular, for any natural number k, we define $\Omega(k) = \Omega(\{x_1, \ldots, x_k\})$.

An equational theory in Ω consists of a set $T = \bigcup_{n=0}^{\infty} T_k$ where T_k is a set of formal equations of the form $\xi = \xi'$

where if $k = 0$, $\xi, \xi' \in \Omega(0)$; if $k > 0$, $\xi, \xi' \in \Omega(k) - \Omega(k-1)$.

A T-algebra is an Ω-algebra in which all equations of T are satisfied.

The free T-algebra on generators I, T(I), where I is any set, is the quotient of $\Omega(I)$ relative to the smallest congruence containing all identifications which are substitution instances of equations in T.

In particular, for any natural number k, we define $T(k) = T(\{x_1, \ldots, x_k\})$.

In [3] Lawvere showed how, given an equational theory T, one may define a category $\underline{T}$ in such a way that the T-algebras are precisely the product-preserving functors from $\underline{T}$ to Sets. The category $\underline{T}$ has a single object k for each natural number, and the morphisms from k to ℓ are given by

$$T[k, \ell] = T(k)^{\ell}.$$

Composition in $\underline{T}$ is given by substitution of representative expressions.

A right action of a monoid may be viewed as a monadic ($\Omega_i = \phi$ for $i \neq 1$) algebra. Thatcher and Wright [4] initiated the generalization of automata theory to non-monadic algebras.

For a given theory T, a T-algebra automaton is a T-algebra A together with functions $\alpha: I \to A$, $\lambda: A \to J$. The behaviour of the T-algebra automaton (A, α, λ) is the function

$$B\colon T(I) \to J$$

defined by $B(\xi) = \lambda(\xi_A(\alpha))$ where $\xi_A\colon A^I \to A$ is the function induced by evaluation of the expression ξ in A (strictly, since ξ is an equivalence class of expressions, we evaluate a representative; and this is of course well-defined since A is a T-algebra)

With $I = \phi$, $J = 2$, we are using A to recognize a certain subset of the initial T-algebra $T(\phi)$, as in [5]. With I, $J = A$; α, $\lambda = 1_A$, we are evaluating expressions on data from A.

To formulate algebra automata as machines in (cat, Sets), let $\underline{T}$ be as above. We denote by S the skeleton (i.e., isomorphic objects are equal) of the category of finite sets. We have an embedding $u\colon S^{op} \to \underline{T}$ as follows: For any function

$$f\colon \{x_1, \ldots, x_k\} \to \{x_1, \ldots, x_\ell\}$$

we have

$$(f(x_1), \ldots, f(x_k)) \in T(\ell)^k,$$

that is, a morphism $\ell \to k$ in $\underline{T}$. This gives the functor u.

Sets may be viewed as T-algebras for which Ω, and hence T are empty. In this case, the associated category may be seen to be S^{op}. That is, product preserving functors from S^{op} to Sets may be interpreted as sets.

It is easy to see that if $\Phi\colon \underline{T} \to$ Sets is a product

preserving functor corresponding to a T-algebra, then sub u Φ is the underlying set. The left adjoint Σu is an extension of the free algebra functor, which is left adjoint to the forgetful functor, so $\Sigma u(I) = T(I)$.

An algebra automaton gives us a machine in (cat, Sets) over the behaviour scheme u

$$I \xrightarrow{\alpha} \text{sub u } \Phi \xrightarrow{\lambda} J$$

where Φ is product preserving. Its behaviour is a functor sub u Σ u I $\to$ J; that is $T(I) \to J$ and may be seen to be the behaviour in the usual sense.

On the other hand, given a function $T(I) \to J$ we may calculate the reduced realization. The resulting functor $\underline{T} \to$ Sets may be shown to be product preserving; that is, a T-algebra, in fact, the minimal realization of [6].

4.3 Recursive Computation

Recent work by Lawvere and Tierney has made it possible to treat non-equational theories in the same way as equational theories are treated by Lawvere. Given a theory (set of sentences) T in higher order logic, one can construct a topos $\underline{T}$ (a topos is a cartesian closed, finitely bicomplete category in which the functor assigning to every object its set of subobjects is corepresentable) with the property that models of T are precisely the topos structure preserving functors from $\underline{T}$ to Sets.

We can formulate in such a language definitions of

arbitrary recursive functions. (These can not be formulated in an equational theory, since they involve essentially partial operations, such as minimization). The behaviour of the analogue of an algebra automaton would be the evaluation of terms (descriptions of recursive functions); that is, the function assigning to a recursive function description and data in a model, the value of that recursive function.

This generalization of algebra automata has not been discussed elsewhere.

4.4 Linear Systems

It seems to me that this is a particularly convincing example, since the formulation in (cat, Sets) meshes precisely with Kalman's elegant formulation of linear systems [7]. In fact our formulation explains certain features of Kalman's which, as it turns out, arise as left and right adjoints.

A discrete-time, constant linear system over a field K is specified by linear maps

$$\delta: X \to X$$

$$\alpha: K^m \to X$$

$$\lambda: X \to K^p$$

where X is a finite dimensional vector space over K. The dynamical interpretation of the system is

$$x(t + 1) = \delta x(t) + \alpha u(t)$$

$$y(t) = \lambda x(t)$$

where $t \in Z$, $x \in X$, $u \in K^m$, $y \in K^p$.

We paraphrase the formulation of Kalman [7]. X may be interpreted as a left module over the polynomial ring K[z] with the action

$$f \cdot x = f(\delta)(x)$$

where $f \in K[z]$ and $f(\delta)\colon X \to X$ is f evaluated at $\delta \in \mathrm{Hom}[X, X]$. Therefore we may represent the linear system as a diagram

$$K^m \xrightarrow{\alpha} X \xrightarrow{\lambda} K^p$$

where X underlies a K[z] action. The diagram is in the category of vector spaces over K.

We now formulate linear systems as machines in (cat, Sets).

For any ring R, the modules over that ring may be defined by an equational theory (with $\Omega_2 = \{+\}$, $\Omega_1 = R$, $\Omega_0 = \{0\}$ and suitable equational axioms). Let T_K be the category associated with the theory of K-modules for a fixed field K; that is, the theory of vector spaces over K. Let $T_{K[z]}$ be the category associated with the theory of modules over the polynomial ring K[z], There is a natural embedding $u\colon T_K \to T_{K[z]}$ induced by $K \subset K[z]$. We have thus represented a linear system as a diagram

$$K^m \xrightarrow{\alpha} \text{sub } u\ X \xrightarrow{\lambda} K^p$$

where X is here interpreted as a K[z]-module (since sub u

is just the forgetful functor from K[z] modules to K vector spaces).

We can now see why Kalman must formulate his input-output maps as K[z]-linear maps from $K^m[z]$ to $K^p[[z^{-1}]]$ (formal power series in z^{-1}). We are able to show that $\Sigma \; u \; K^m = K^m[z]$ and $\Pi \; u \; K^p = K^p[[z^{-1}]]$. If the behaviour of a linear system in our sense is $B: \text{sub} \; u \; \Sigma \; u \; K^m \to K^p$, then the behaviour in Kalman's sense is just $\bar{B}: \Sigma \; u \; K^m \to \Pi \; u \; K^p$.

It follows that our reduced realization is the classical one.

4.5 Machines with Algebraic Structure

One can define a tensor product of theories with the property that product preserving functors from $\underline{T}$ into the category of product preserving functors from $\underline{T}'$ to Sets are in a 1 - 1 correspondence with product preserving functors from $\underline{T} \otimes \underline{T}'$ to Sets. (see [10]). For example, $T_{K[z]} = T_K \otimes T_N$ where T_N is the theory of N-actions, where N is the monoid of natural numbers under addition.

Thus if we want to consider T-algebra automata in the category of T' algebras, we will have functors $\underline{T} \to \text{Sets}^{\underline{T}'}$ and hence $\underline{T} \otimes \underline{T}' \to \text{Sets}$. Using the behaviour scheme $\underline{T}' \to \underline{T} \otimes \underline{T}'$ (the tensor product has natural injections) we can discuss these machines in our framework.

Goguen [2] has a formulation which unifies much of realization theory. (also the study of behaviour-realiza-

tion adjunctions was initiated in [2]). He constructs a triple (monad, standard construction) in a closed monoidal category which is analogous to $A \mapsto A \times S^*$ where S^* is the free monoid on a set S. A machine is a diagram

$$1 \to A \to J$$

where A is an algebra over this triple (he does not describe his construction this way). I expect that many, if not all, of his examples can be formulated in the present framework using the tensor product of theories $\underline{T} \otimes \underline{T}'$, where T' is a theory whose algebras form the closed monoidal category, and T is a theory whose algebras in $\mathrm{Sets}^{\underline{T}'}$ are the algebras over the triple.

4.6 Machines with Restricted Input

Let G be the state graph of a (non-deterministic) generator for some set of strings over the alphabet S. We define a category G having as objects the states of G and as morphisms all triples (g, σ, g') where σ is a string resulting from a transition from g to g'. Composition is given by

$$(g', \sigma', g'') \circ (g, \sigma, g') = (g, \sigma\sigma', g'').$$

For any category X, Ob X is the category with objects as in X, but only identity morphisms.

Suppose G has an initial state 0 and a final state 1. Let L be the language generated by G; that is, the set of

all $\sigma \in S^*$ such that $(0, \sigma, 1)$: $0 \to 1$ in $\underline{G}$. A behaviour over L is a function

$$B: I \times L \to J$$

for some sets I, J. For given sets I, J, B is essentially a behaviour over the scheme u: Ob $\underline{G} \to \underline{G}$ (the obvious functor), namely the morphism $\underline{B}$: sub u Σ u $\underline{I} \to \underline{J}$ given below

	$\underline{I}$	sub u Σ u $I \xrightarrow{\underline{B}} J$
0	I	$I \times \underline{G}[0, 0] \longrightarrow 1$
⋮		
g	ϕ	$I \times \underline{G}[0, g] \longrightarrow 1$
⋮		
1	ϕ	$I \times \underline{G}[0, 1] \xrightarrow{B} J$

A machine $M = (\text{Ob } \underline{G}, u, \underline{G}, \Phi, \underline{I}, \underline{\alpha}, \underline{J}, \underline{\lambda})$ will be a diagram

	$\underline{I} \xrightarrow{\underline{\alpha}}$	sub u $\Phi \xrightarrow{\underline{\lambda}} \underline{J}$
0	$I \xrightarrow{\alpha}$	$\Phi(0) \longrightarrow 1$
⋮		
g	$\phi \longrightarrow$	$\Phi(g) \longrightarrow 1$
⋮		
1	$\phi \longrightarrow$	$\Phi(1) \xrightarrow{\lambda} J$

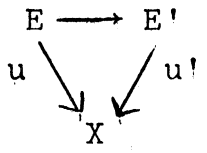
Such a machine can be interpreted as follows. Assign to each $g \in$ Ob G a machine with input monoid $\underline{G}[g, g]$. When a transition from state g to state g' occurs, the machine at g is switched off and the machine at g' is switched on with an appropriate initial state. In our example, only the machine at state 1 has outputs, but we could change this if we wanted to realize a different type of behaviour.

5. Other Features of the Formulation

We have used some powerful apparatus to formulate machine theory. So far we have not made use of much of the mathematical structure available. Here we present some examples which show that we can expect to find machine theoretic significance in this extra structure. These applications were not designed into the formulation and hence provide evidence that this is indeed an appropriate setting for much of automata theory.

5.1 State Graphs

The category (cat, X) where X is a small category has as objects functors u: E → X and as morphisms commutative triangles



The functor

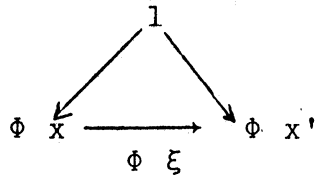
$$(\text{cat}, X) \to P(X)$$

$$u \mapsto \Sigma\, u\,(1_E)$$

where 1_E is terminal in P(E) has a right adjoint

$$\Phi \mapsto ((1, \Phi) \to X)$$

where 1 is terminal in Sets, and $(1,\Phi)$ has as objects morphisms $1 \to \Phi\, x$, $x \in$ Ob X and as morphisms commutative triangles



where ξ: $x \to x'$ in X. The functor to X is projection.

In the terminology of Lawvere [8], (cat, Sets) satisfies the comprehension schema. Lawvere introduced this notion to model the relation between predicates P(x) and their extensions $\{a \mid P(a)\}$. In our case, $(1,\Phi)$ is just the state graph of the action Φ. That is, $(1,\Phi)$ has as objects (in the case where Φ is a monoid action) the elements of the set on which the action is defined, and has as morphisms arrows

$$q \xrightarrow{x} q'$$

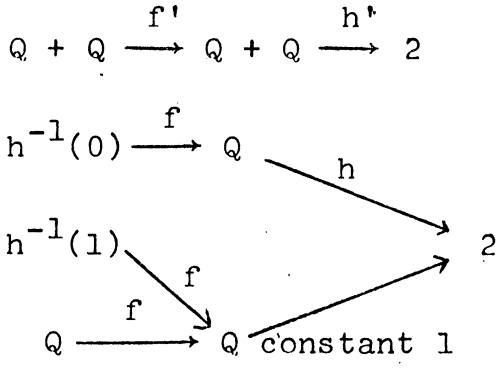
where $q' = q \cdot x$. The projection $(1, \Phi) \to X$ is just the labelling of the arrows of the state graph by monoid elements; that is, $(q \xrightarrow{x} q') \to x$.

For the other types of machines discussed in section 4, $(1, \Phi)$ may also be interpreted as a state graph.

Goguen [11], and the present author (independently, in unpublished work *) have given formulations of an interconnection of systems as a diagram in a suitable category, with the resultant system being expressable as the limit of the diagram. The above adjunction is the appropriate way to relate this formulation of sustems to the better developed theory of systems with inputs as actions.

5.2 Computation Time

Suppose we have a function $f: Q \to Q$. To any $q \in Q$ and function $h: Q \to 2$ we may assign the natural number: least n $(h(f^n(q)) = 1)$. We can call this the computation time of the program: ' do f until h' when started on the data q. We may suppose without loss of generality that $\{q \mid h(q) = 1\}$ is closed under the action of f, otherwise we replace the original f, h with the equivalent f', h' given by



* Private communication to Y. Give'on August 1967; also Dissertation prospectus, Department of Computer and Communication Sciences, University of Michigan, October 1968.

which do satisfy the condition.

We may interpret Q, f as an N-action, where N is the monoid of natural numbers under addition, with the action given by

$$q \cdot n = f^n(q).$$

The category Sets^X, where X is a small category, is a topos (see 4.3). Thus we have a <u>subobject classifier</u> $\Omega \in \mathrm{Ob}\ \mathrm{Sets}^X$ with the property that the subobjects of $\Phi\colon X \to \mathrm{Sets}$ are in a natural 1 - 1 correspondence with morphisms $\Phi \to \Omega$ (compare characteristic functions; in Sets, $\Omega = 2$).

Given an element $q \in Q$ we may consider the corresponding function

$$1 \xrightarrow{q} \mathrm{sub}\ u_N(Q, \cdot)$$

and hence

$$\Sigma\ u_N\ 1 \xrightarrow{\ulcorner q \urcorner} (Q, \cdot)$$

The action $\Sigma\ u_N\ 1$ may be taken to be on N, namely $m \cdot n = m + n$. We have a function

$$[(Q, \cdot), \Omega] \to [(N, \cdot), \Omega]$$

induced by composition with $\ulcorner q \urcorner$.

This assigns to every subaction of $(Q, \cdot)$ (such as determined by h above) a subaction of $(N, \cdot)$. A subaction

of $(N, \cdot)$ is easily seen to be a set of the form $\{m \mid m \geq k\}$ for some k, or the empty subaction. We may see that to $\bar{h}$: $(Q, \cdot) \to \Omega$, corresponding to h, is assigned $\{m \mid m \geq \text{least } k(h(f^k(q)) = 1)\}$. Thus the function

$$[1, \text{sub } u_N(Q,\cdot)] \times [(Q,\cdot), \Omega] \to [\Sigma\, u_N\, 1, \Omega]$$

may be interpreted as assigning to each $q \in Q$ and subaction h the computation time on q of : do f until h. This may be generalized.

Using the duality of the next section, the above construction (in a generalized form) may be given a dual interpretation. Also, using the exponentiation in Sets^X the above construction and its generalizations can be represented internally in the category, and given a suitable interpretation.

6. Duality

Arbib and Zeiger [9] have investigated the duality between reachability and observability for sequential machines. There is a well known duality of this type for linear machines. We present here a duality of this type for general machines in (cat, Sets) which includes in a certain sense the linear machine duality as a special case. The construction is similar in part to that of [9] but involves the essential extra notion of a **coordinatized machine**.

6.1 Coordinatized Machines

There is a contravariant adjunction on the right between $P(X)$ and $P(X^{op})$. That is, there are contravariant functors (both of which we denote by #)

$$P(X) \underset{\#}{\overset{\#}{\rightleftarrows}} P(X^{op})$$

such that

$$[\Phi,\ \Psi^{\#}] \cong [\Psi,\ \Phi^{\#}]$$

The functor #: $P(X) \to P(X^{op})$ is the extension to a functor of the assignment

$$(X \xrightarrow{\Phi} \text{Sets}) \mapsto (X^{op} \xrightarrow{\Phi^{op}} \text{Sets}^{op} \xrightarrow{2^{(\,)}} \text{Sets})$$

That is, $\Phi^{\#}(x) = 2^{\Phi x}$.

We note the following relations between # and the other operations of the hyperdoctrine (cat, Sets)

$$(\text{sub } u\ \Phi)^{\#} = \text{sub } u^{op}\ \Phi^{\#}$$

$$(\Sigma\ u\ \Phi)^{\#} = \Pi\ u^{op}\ \Phi^{\#}$$

Let $(*, R)$ be a left action of a monoid X. The right action $(*, R)^{\#} = (2^R, \cdot)$ is given by

$$(\rho \cdot x)(r) = \rho(x * r).$$

Suppose that elements $r \in R$ are interpreted as nodes of a network at which binary values appear. On an input signal $x \in X$, each node receives its next value from the node $x * r$. The resulting right action on 2^R is as above. In other words, a right action of the form $(*, R)^{\#}$ may be interpreted as that realized by a certain type of network.

If $(Q, \cdot)$ is a subaction of $(*, R)^{\#}$ we may say that the network $(*, R)$ realizes the action $(Q, \cdot)$. We may assume that the function $R \to 2^Q$ obtained by transforming $Q \subset 2^R$ is 1 - 1, otherwise we replace $(*, R)$ by the smaller network $\mathrm{Im}((*, R) \to (*, 2^Q))$ which realizes $(Q, \cdot)$ and has the property.

Thus in general, we define a coordinatized action of X, X a small category, to be a subobject $\Phi \hookrightarrow \Psi^{\#}$, $\Phi \in \mathrm{Ob}\ P(X^{op})$, $\Psi \in \mathrm{Ob}\ P(X)$, whose transform $\Psi \to \Phi^{\#}$ is a monomorphism.

A coordinatized machine $K = (E, u, X, \Phi, \Psi, I, \alpha, J, \lambda)$ consists of a scheme $u: E \to X$, a coordinatized action $\Phi \hookrightarrow \Psi^{\#}$ over X, $\alpha: I \to \mathrm{sub}\ u\ \Phi$ in $P(E)$, and $\lambda: J \to \mathrm{sub}\ u^{op}\ \Psi$ in $P(E^{op})$. We represent a coordinatized machine by a diagram

$$I \xrightarrow{\alpha} \mathrm{sub}\ u\ \Phi \to \mathrm{sub}\ u\ \Psi^{\#} \xrightarrow{\lambda^{\#}} J^{\#}.$$

Note that is $\Phi \hookrightarrow \Psi^{\#}$ is a coordinatized action, then so is $\mathrm{sub}\ u\ \Phi \hookrightarrow \mathrm{sub}\ u\ \Psi^{\#}$. The dual of the above coordinatized machine

is

$$J \xrightarrow{\lambda} \text{sub } u^{op}\ \Psi \to \text{sub } u^{op}\ \Phi^{\#} \xrightarrow{\alpha^{\#}} I^{\#}.$$

The behaviour of a coordinatized machine is the behaviour of its abstract part

$$I \to \text{sub } u\ \Phi \to J^{\#}$$

A coordinatized machine also has a concrete part, a network

$$J \to \text{sub } u^{op}\ \Psi \to I^{\#}$$

The fundamental result for coordinatized machines is that every behaviour of the form

$$B\colon \text{sub } u\ \Sigma\ u\ I \to J^{\#}$$

has a unique coordinatized realization B such that the abstract part and the concrete part of B are both reachable. This is the coordinatized machine obtained as follows. Consider

$$\Sigma\ u\ I \to Im\ \overline{B} \to \Pi\ u\ J^{\#}$$

and the resulting

$$\Sigma\ u\ J \to Im \to (Im\ \overline{B})^{\#}.$$

The coordinatized action is

$$Im\ \overline{B} \to Im^{\#}.$$

We define a morphism of coordinatized actions

$$(\Phi \hookrightarrow \Psi^{\#}) \to (\Phi' \hookrightarrow \Psi'^{\#})$$

to consist of $\Phi \xrightarrow{f} \Phi'$ and $\Psi' \xrightarrow{g} \Psi$ such that

$$\begin{array}{ccc} \Phi & \hookrightarrow & \Psi^{\#} \\ f\downarrow & & \downarrow g^{\#} \\ \Phi' & \hookrightarrow & \Psi'^{\#} \end{array}$$

commutes. We can then reconstruct the counterparts of the adjunctions of section 3. However we now replace observable by addressable (has a reachable concrete part). Furthermore, everything is dualizable, including behaviours, and these functors commute with the dualities.

6.2 Duality for Linear Machines

For linear machines, we have the following situation. Morphisms from m to n in the category $T_{K[z]}$ corresponding to the theory of left K[z]-modules may be taken to be $n \times m$ matrices over K[z]. Transposition gives us a contravariant functor $t: T_{K[z]} \to T_{K[z]}$ with $t^2 = 1$. Thus $T_{K[z]}^{op}$ is the theory of right K[z]-modules. Any K[z] module V is a vector space over K, and so we may speak of the dual vector space V*. It is easy to see that V* may be given a right K[z]-module structure.

Thus given a linear machine

$$K^m \xrightarrow{\alpha} \text{sub } V \xrightarrow{\lambda} K^p$$

we construct a coordinatized linear machine as follows. Define $\tilde{V} = (V^* \circ t)^{\#}: T_K \to \text{Sets}$, where $t: T_K^{op} \to T_K$ is transposition. (This amounts to $\tilde{V}(n) = 2^{V^{*n}}$).

We have the 1 - 1 functions

$$V^n \hookrightarrow 2^{V^{*n}}$$

corresponding to

$$V^n \times V^{*n} \to 2$$

where $((v_1, \ldots, v_n), (v_1^*, \ldots, v_n^*))$ is mapped to 1 if $\sum_i v_i^*(v_i) = 1$ and 0 otherwise. These give a natural transformation $V \to \tilde{V} = (V^* \circ t)^\#$ which is a coordinatized action in the above sense. We have the coordinatized linear machine

$$K^m \xrightarrow{\alpha} \text{sub } u\ V \to \text{sub } u\ \tilde{V} \xrightarrow{\tilde{\lambda}} \widetilde{K^p}.$$

The dual of this coordinatized machine is

$$K^p \cong (K^p)^* \xrightarrow{\lambda^*} \text{sub } u\ V^* \to \text{sub } u\ \widetilde{V^*} \xrightarrow{\widetilde{\alpha^*}} \widetilde{(K^m)^*} \cong \widetilde{K^m}$$

which is precisely the coordinatized machine corresponding to the usual dual of a linear machine.

References

[1] LAWVERE, F. W. Adjointness in Foundations, Dialectica, Vol. 23, No. 3/4 (1969).

[2] GOGUEN, J. A. Discrete-time Machines in Closed Monoidal Categories I, Quarterly Report #30, Section IIB Institute for Computer Research, University of Chicago 1971.

[3] LAWVERE, F. W. Functorial Semantics of Algebraic Theories, Proc. Nat. Acad. Sci. U. S. A., 50, 869-872 (1963).

[4] THATCHER, J. W. and WRIGHT, J. B. Generalized Automata Theory with an Application to a Decision Problem of Second Order Logic. IBM Research RC 1713 Nov. 16, 1966.

[5] EILENBERT, S. and WRIGHT, J. B. Automata in General Algebras, IBM Research Note NC 725 July 1967.

[6] ARBIB, M. A. and GIVE'ON, Y. Algebra Automata 1: Parallel Programming as a Prolegomena to the Categorical Approach, Information and Control.

[7] KALMAN, R. E. Algebraic Theory of Linear Systems, in Topics in Mathematical System Theory; Kalman, Arbib, Falb, McGraw-Hill 1969.

[8] LAWVERE, F. W. Equality in Hyperdoctrines and Comprehension Schema as an Adjoint Functor. Proc. Symp. Pure Math. Vol. XVII Applications of Categorical Algebra AMS 1970.

[9] ARBIB, M. A. and ZEIGER, H. P. An Automata-Theoretic Approach to Linear Systems, IFAC Symposium, Sydney Australia, August 1968.

[10] PAREIGIS, B. Categories and Functors, Academic Press 1970.

[11] GOGUEN, J. A. Categorical Foundations for General Systems Theory. Presented at Midwest Category Theory Seminar, New Orleans January 1969.

[12] ARBIB, M. A. Automata Theory and Control Theory - A Rapprochement. Automatica, Vol. 3 pp. 161-189 (1966).