WIS-CS-74-213

COMPUTER SCIENCES DEPARTMENT
University of Wisconsin
1210 West Dayton Street
Madison, Wisconsin 53706

Received April 1, 1974

A CONCEPT OF EQUIVALENCE BETWEEN FORMALLY DEFINED COMPLEXES OF INTERACTING DIGITAL SYSTEMS

by

Pamela Z. Smith

and

D. R. Fitzwater

Technical Report # 213
April 1974

A CONCEPT OF EQUIVALENCE BETWEEN FORMALLY DEFINED COMPLEXES OF INTERACTING DIGITAL SYSTEMS

by

Pamela Z. Smith

and

D. R. Fitzwater

ABSTRACT

The environment of a formally defined complex of interacting digital systems is specified as a set of ideal, independent observers who can provide input as well as monitor any observable computation. Precise definitions are given for the behavior of an observer toward a system complex, the behavior of a system complex toward an observer, and the relative rates at which the systems of a complex can finish steps and transmit messages. These ideas are used in a very general definition of functional equivalence between system complexes; a sample equivalence proof and suggestions for more specialized forms of equivalence are added. It is argued that this definition compares favorably with other notions of equivalence because it is based on effective observations and allows equivalence classes to be as rich as possible.

CONTENTS

A CONCEPT OF EQUIVALENCE BETWEEN FORMALLY DEFINED COMPLEXES OF INTERACTING DIGITAL SYSTEMS

I. INTRODUCTION

This report is a continuation of the research presented in [2] and [3] on the properties of formally defined systems.

Our purpose at this time is to learn how to analyze a complex of interacting digital systems--algorithmically, and without recourse to simulation or human interpretation of its computations. The immediate benefits will be diagnostic feedback to the designer, and knowledge of design constraints that will make precise analysis practical. We hope to learn, eventually, how to optimize system complexes, but the problem is especially challenging within the context of the formal definition universe. One reason is that whatever changes we make must be valid improvements, no matter how the system complexes are ultimately implemented. The other is that we must be able to prove that the optimized complex is equivalent to the original.

Equivalence is recognized as the key to understanding in many aspects of the study of computation. If a set of objects is partitioned into equivalence classes, all the members of a class are known to be alike in a significant way, and unlike in insignificant ways. When the objects so classified represent computations, correctness is easily defined as membership in the same equivalence class as a standard of correctness. Each class may have other canonical members which are optimal with respect to various criteria.

Many notions of equivalence have been proposed for partial recursive functions and the sequential processes that compute them. In [7], Wegner gives an interesting classification of models of computable functions upon which a classification of equivalence relations can be based. The first type is that of the input-output or representation-independent models, in which the function is defined by a rule of correspondence between elements of the domain and elements of the range. Two functions f and g from the same domain to the same range are functionally equivalent if and only if they are defined on exactly the same arguments, and when they are defined, $f(x) = g(x)$. Functional equivalence is the basis for Manna's formal approach to the verification of programs [5] (any two programs which satisfy a specific correctness criterion are equivalent). Manna's definition of correctness of the program P with respect to an input predicate $\phi(x)$ and an output predicate $\psi(x,z)$ is that for any x such that $\phi(x)$ is true (the input vector x is within the proper domain of the program), $P(x)$ is defined and $\psi(x,P(x))$ is true.

The next type of model in Wegner's scheme is the representation-dependent one, in which functions are characterized by their representations in a specific formalism, such as a programming language. The assumption is that semantics are defined by associating semantics with individual symbols of the representation. This may be the best category in which to mention equivalence of program schemas [4]. The idea is that a program schema is a partial representation-dependent specification of a function; a schema plus an interpretation make a complete specification. Luckham, Park, and Paterson's definition of strong equivalence between two schemas S and S' is that for all interpretations I and all input vectors x, either both Value (S,I,x) and Value (S',I,x) are undefined, or both are defined and Value (S,I,x) = Value (S',I,x). Note, however, that for the complete function specifications (S,I) and (S',I), this is the same as functional equivalence except for the means of specifying the function.

If the semantics of a representation specifying a function are defined in terms of execution-time transformations, then we have an implementation-dependent model. Wegner's formulation of this class of models, also given in [7], is the information structure model: each function is specified as a triple (I, I^{o}, F) where I is the set of all information structures or states that can arise during computation, $I^{o} \subset I$ is the set of all initial states, and F is a binary state transition relation, $F : I \rightarrow \mathcal{P}(I)$ (the set of all subsets of I). The many forms of equivalence possible here are all instances of mapping equivalence, a relation between information structure models which holds when certain mappings can be established between states in the computational sequences generated by those models. An example of this is output equivalence as defined by Berry [1]: if the two information structure models to be proven equivalent are M and M', for every computation C in M there is a corresponding computation C' in M', such that C' halts if and only if C halts, and if they both halt, they both yield the same output (output is defined by a projection function on the state). The same must also be true of a mapping from computations in M' to computations in M. This is also the same as functional equivalence except for the way the function is specified.

Wegner's final type is that of device-dependent models, in which functions are characterized by their hardware implementations. These do not stimulate much theoretical interest, because of their lack of general applicability.

We have seen that all these equivalence relations are basically the same as functional equivalence: they treat representations of two partial recursive functions as black boxes, calling them equivalent if they transform the same inputs to the same outputs. Almost nothing has been said about equivalence between assemblages of processes interacting both synchronously and asynchronously (which are, of course, realistic models of computations performed by digital computers today). The problem is mentioned by Wegner [7] and Berry [1], who both point out that functional equivalence is useless for operating systems, because they do not halt and do not have a one-to-one correspondence of inputs to outputs. Their solution is to deal with operating system equivalence exclusively on the implementation-dependent level, and to use some form of mapping equivalence between successive states of the systems which does not depend on arrival at a halting state.

Our problem of defining equivalence between system complexes is essentially the same as defining operating system equivalence, and we are also bound to an implementation-dependent (in Wegner's terms) model of computation. But we have chosen our definition differently than Wegner and Berry because we wanted it to be more general than mapping equivalence, which places severe constraints on the computational structures that can be proven equivalent.

What we wanted was a definition analogous to functional equivalence in that the actual computation can be viewed as a black box--only results matter. This was achieved by generalizing the notion of input and output to that of a conversation between the environment and the system complex. Formalizing such an idea requires a clear understanding of the relationship between a system complex and its environment. We hope to come to such an understanding in this report.

II. OBSERVING SYSTEMS

1. The Ideal Observer

A system complex has no usefulness or meaning unless it can engage in communication with its environment. From the viewpoint of the system complex, this communication takes place via the standard mechanisms for asynchronous communication between systems: an incoming message is associated with a channel name and is received in the σ''s of all the systems just as if it had been sent by one of the systems in the complex, and a message sent by a system in the complex is assumed to be broadcast outside the complex as well.

With this mechanism a user of the system complex can send any message to it (provided he knows the right channel name!) and can observe any part of the computation which is observable. Here "observable" refers to the well-defined sequence of process state sets of a system produced during interpretation by the primitive automaton--computations within an RPR, for instance, cannot be observed or influenced, but they are defined on another level that is meant to be unobservable. The simple production

$\{\ \$\ \underline{\chi}\text{: < any characters every appearing in } \sigma\text{> } \underline{\pi}\text{: }\} \rightarrow \$_1 \rightarrow \text{observer}$

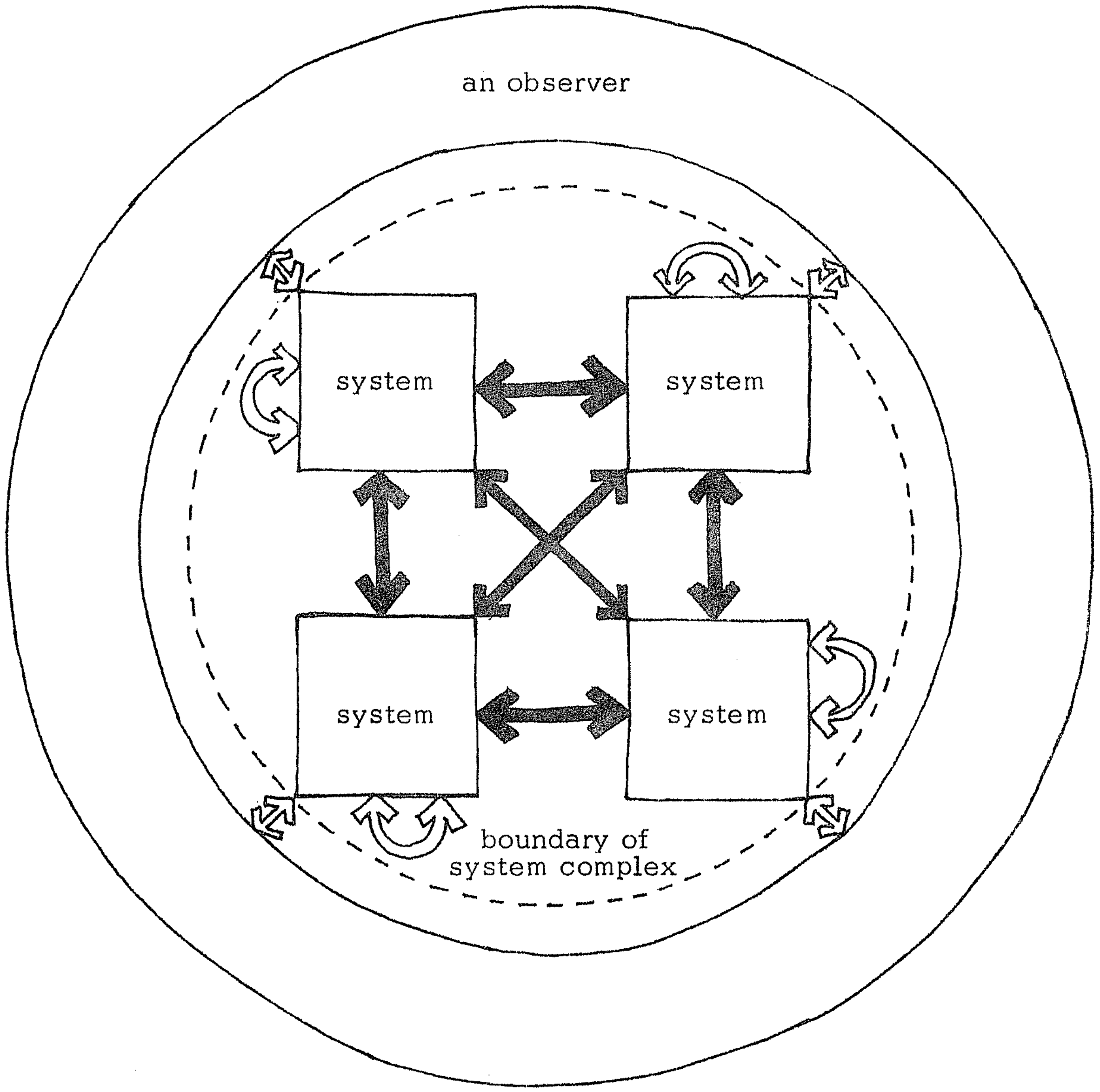
would make the complete contents of every process state set of a system visible to anyone accepting on the channel "observer".

The designer of a system complex should be able to specify what is significant about its computations. This is the only way that we will be able to find non-trivial equivalent forms of it: system complexes which preserve the essential aspects of its computations, but differ from it in interesting ways.

The designer gives us this information about the system complex to be analyzed in the form of a set of observing systems. These systems are not formally defined except by constraints on the interactions they can have with the system complex. These constraints are, effectively, a specification of the inputs the complex can expect to receive and the messages it generates which will be accepted by its environment.

Each observer is ideal in the sense that it is assumed to have communication links with each system in the complex so good that message transmission between them is effectively instantaneous, and to cycle so fast that messages to it are always accepted before they are overwritten. Although it is obvious that no physical device could implement such an observer (how could one device be close enough to all members of a set of asynchronous devices to have "instantaneous" electronic connections to all of them?), it is necessary to define observers this way, so as to satisfy any requirements that a user might have. In other words, some local properties of a real observer could approximate adequately those of the ideal observer. Since we have no way of knowing where or when this will occur, we must assume it is always the case. Figure 1 may help clarify the relationships.

Take as an example the problem of message loss by overwriting of a buffer. It is surely possible for a user of an implemented system complex to need every single message that comes over a certain channel (the buffer name), no matter how closely each follows the other, and to implement some device that guarantees acceptance of each one. If our analysis is to be valid for this user, the formally specified observer corresponding to his



⬌ two-way communication link with finite transmission time

⇔ two-way communication link with infinitely small transmission time

Figure 1.

implemented observer must also be guaranteed to accept all the messages sent to it. The only difference between the formal representation and the implementation is this: since the speed of the formal complex is unbounded (but finite!), the speed of an ideal observer is also unbounded; the speed of an implemented complex is always bounded by its technology, and so the construction of an ideal observer is simply a matter of using slightly better technology.

It is instructive to consider the alternative to defining our observers as ideal. If the designer knows that the user of the complex cannot tolerate message loss and will implement his interface to avoid it, for purposes of analysis, he _must_ specify a formal observer who loses no messages. Either we help him, or he has to tamper with the system complex to introduce, at the source of the messages, cooperation with the observer that will prevent message loss.

2. Specification of Observers

The formal specification of an observer states which channel names he can use to sent messages into the complex, what kind of messages can be expected on each channel, and which channels he accepts messages on. The description for each of these categories is in terms of regular languages. Regular languages pervade our analysis, for reasons that are fully explained in [6], but for now let us just say that, because of the nature of the formal definition universe, regular languages are excellent finite characterizations of the structures of infinite sets of process states.

Each observer is specified as:

(1) a set of disjoint regular languages, called output channel languages--the observer can send messages to the complex only on channels contained in these languages;

(2) for each output channel language, a corresponding regular message language, containing all the messages which can be sent on channels in the output channel language; and

(3) a regular language called the input channel language--the observer accepts a message if and only if it is sent on a channel contained in this language.

Except for this interface, an observing system is not defined, and can be arbitrarily complex in its behavior.

3. Independence of Observers

The reason for allowing a designer to specify a set of observers instead of just one is that each observer is understood to be autonomous, making it possible to model the effects of simultaneous, but separate, demands on the system complex. The separateness of observers is most useful for factoring the analysis of the system complex. If the observers are really independent of each other, we can carry out our procedures with respect to one observer at a time, ignoring the rest of the environment.

As a matter of fact, because of the nature of observations (which are always made from a particular frame of reference), we have no choice but to analyze with respect to one observer at a time. An assertion about a system complex can only be said to be true or false with respect to some observer, although one could also make an assertion about the complex with respect to some well-defined class of observers.

A problem arises because the observers specified by the designer may not be independent--if one observer can send a message which later affects what another observer sees, the second observer is dependent on the first. To analyze with respect to the second observer, we will have to merge it with the first, so that all the influences on the behavior of the complex will be involved in the analysis.

The difficult part is finding an effective test of independence for observers. It is possible to describe all the computat'ons, finite or infinite, that can ever be generated by a system complex, in a finite graph whose nodes are regular languages--and which can be derived algorithmically from the formal definition of the complex alone. These graphs are called finite process structures, and are discussed at length in [6]. A finite process structure for a system complex and its observers can be used to test the observers for independence.

Figure 2 is a finite process structure for a complex with two systems and three observers. Each node in a system represents a regular language that process states in the system can belong to. The arcs represent successor relationships; arcs with the same letter label are components of multiple-component arcs representing interactions between process states. Here is how a graph like this can be used to test for independence:

We define an equivalence relation on the nodes of a finite process structure called influence. Node A influences node B if there is an arc component having one of them as its origin node and one of them as its destination node. This makes it symmetric, and since we define it to be reflexive and transitive, it is an equivalence--and so it specifies a unique partition of the nodes of any finite process structure into influence classes or domains which cannot affect each other in any direct way. In Figure 2 there are two domains: {A,B,C,D} and {E,F,G,H,I,J,K,L,M,N}.

Each observer is connected by arcs to one or more domains, but if an observer only accepts messages from a domain, he cannot affect it or any other observer through it. Thus two observers are independent unless one of them accepts messages from the same domain that the other one sends messages to. The observer sending the messages may still be considered independent in analysis with respect to it, but the observer accepting messages from the domain is dependent on the observer sending messages to it, and must be merged with that observer before analysis. In the figure, Observers 1 and 3 are independent, but Observer 2 depends on Observer 3.

There are two ways in which this test might be less than satisfactory. One is that influence domains are rather vague characterizations, taking no account of path directions, etc. It might be impossible for two observers that the test says are dependent to ever affect each other, because crucial connections are one-way in the wrong direction.

The other is that it depends on finite process structures, which can give different pictures depending on how their nodes are chosen. For instance, if the left-hand system had only one node in place of nodes C and E, representing the union of their languages, the test would have said that Observer 1 was also

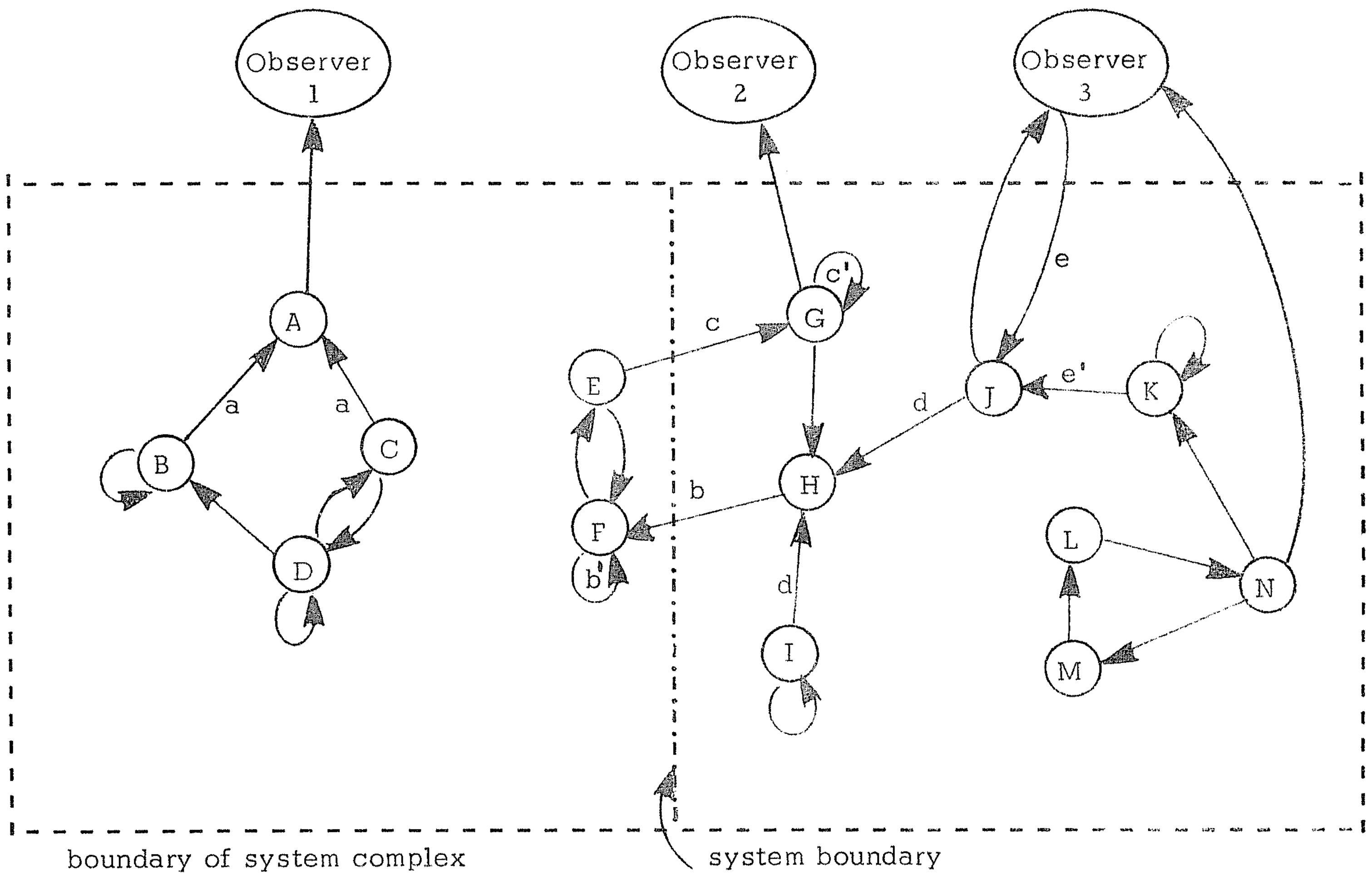

Observer 1
Observer 2
Observer 3
A
B
C
D
E
F
G
H
I
J
K
L
M
N
a
a
b
b'
c
c'
d
d
e
e'
boundary of system complex
system boundary

Figure 2.

dependent on Observer 3. This can always be solved, however, by looking at a sufficiently detailed process structure. The partition into domains is actually quite invariant to the choice of nodes for the finite process structure. To see why this is so, imagine an ultimate, most detailed finite process structure in which the domains are completely accurate characterizations of which process states can affect others. When nodes of this process structure are combined to make simpler graphs (the languages of the combined nodes are unions of the languages of the nodes from which they came), the domains do not change unless two nodes from separate domains are combined, in which case the two domains are combined. Thus a partition into domains based on any finite process structure for a system complex is at worst a poor approximation to the "real" domains.

The most important point is that either of these flaws in the independence test always causes errors on the safe side: the test may indicate that two observers are dependent when they are really not, but it will never say they are independent when they are really dependent. Consequently, the worst that could ever happen would be the loss of a degree of freedom to change the system complex--never would we be caused to make a change under an invalid proof of equivalence.

All we have to do now is define a merge of two observers. The process can be iterated if other observers fail to be independent of the new observer.

(1) The new set of output channel languages is the union of the sets of output channel languages from the two observers, except if two languages intersect, their

intersection is made into a separate language:

$\{0_1, 0_2, 0_3 \cup 0_4\} + \{0_4 \cup 0_5, 0_6\} = \{0_1, 0_2, 0_3, 0_4, 0_5, 0_6\}$.

(2) The message language for an output channel language is the union of all the message languages that were formerly associated with the output channel language or any superset of it:

$\{M_1, M_2, M_{34}\} + \{M_{45}, M_6\} =$

$\{M_1, M_2, M_{34}, M_{34} \cup M_{45}, M_{45}, M_6\}$.

(3) The new input channel language is the union of the two input channel languages from the former observers:

$C_1 + C_2 = C_1 \cup C_2$.

III. BEHAVIORS

1. Relative Rates

Our formal definition of a system complex contains no information, explicit or implicit, about the speeds or message transmission times of any system. The systems may run at any rates relative to each other and to their observers, and so a formally defined design can be considered satisfactory if and only if its behavior at all relative rates is acceptable to all its observers. It may be true that all implemented designs are subject to physical constraints on their speeds, but until we understand the significance of relative rates and have a general notion of equivalence when they are not constrained, we cannot expect to make use of such information.

From now on we will be talking about the system complex with respect to a single observer. We must have a way to measure relative rates, and since we obviously mean "relative to the observer", we will measure them in Observer's Standard Time. OST is zero when the observer begins a session of use of the system complex, traditionally called an experiment. OST is the integer n at the instant the observer sends its nth collection of message sets to the system complex. If the last collection the observer sends, during the experiment, is the qth, then when the experiment ends and the observer ceases to receive messages, OST is q + 1.

In other words, the observer has an internal clock on which all message exchanges are timed. By convention, it begins experiments, ends experiments, and sends message sets only when the time is an integer. This is not restrictive because the relationship between OST and any other kind of time, for instance, real time, can be any monotonically increasing function. Figure 3 is a possible graph of OST as a function of real time.

Any experiment is completely determined by the set of SR's representing the system complex, the sequence of message sets sent by the observer, and the set of rates for the complex relative to that observer.

A set of relative rates for a complex of n systems is a set of n finite vectors $\{R_1, R_2, \ldots, R_n\}$, one for each system. A vector R_i is composed of pairs (t_{ij}, V_{ij}) where t_{ij} is a time (in OST, of course) and V_{ij} is a vector of n times. System i finishes its jth step at t_{ij}. If u_{ijk} is the kth entry in V_{ij}, any message set sent by system i to system k at the end of its jth step would arrive at time u_{ijk}.

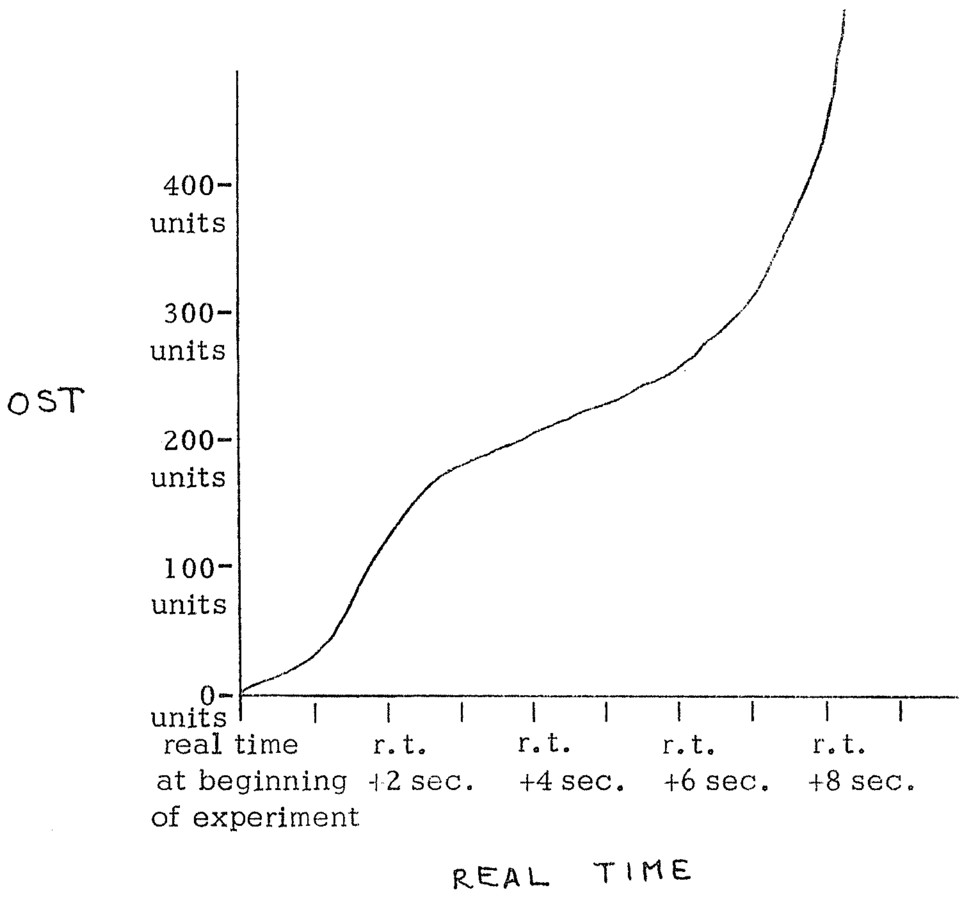
The sets of vectors which can be allowed as relative rates are subject to the following five constraints:

(1) $\forall i,j,k$: $t_{ij} > 0$, $u_{ijk} > 0$;

(2) $\forall i$: if $j < k$, then $t_{ij} < t_{ik}$;

(3) $\forall i,j$: $t_{ij} < u_{ijk}$ unless $i = k$, in which case $t_{ij} = u_{ijk}$ (all message transmissions to systems in the complex require finite time except a system's transmission to itself, which is instantaneous);

(4) $\forall i,k$: if $j < p$, then $u_{ijk} < u_{ipk}$ (message sets sent from one system to another arrive in the order they were transmitted--this is a basic property of most real communication links, and the assumption is necessary for the



400- units
300- units
OST
200- units
100- units
0- units
real time at beginning of experiment
r.t. +2 sec.
r.t. +4 sec.
r.t. +6 sec.
r.t. +8 sec.
REAL TIME

Figure 3.

modeling of real-time interactions, and simplicity, among other things);

(5) $\forall i,q$ such that $i \neq q$, $\forall k,j,p$: $u_{ijk} \neq u_{qpk}$.

This last constraint deserves some explanation. It (in conjunction with constraint (4)) ensures that never do two different message sets associated with the same channel name arrive at the σ'' of a system simultaneously. The formal definition universe demands a well-defined result: that the later message set overwrites the earlier; there is a conflict-resolver in every system to guarantee that this happens. However, if we allow the relative rates to indicate simultaneity, then the result of the conflict resolution will be random (or, at best, probabilistic) in the eyes of the observer--and the model is no longer deterministic.

We are actually standing on solid philosophical ground, claiming simply that no two events are simultaneous if you measure them accurately enough, and that we can measure them accurately enough. It is still possible for an observer to view the behavior of the complex as a statistical phenomenon, and certainly some kinds of averaging will be necessary to tame complexities that arise in analysis. The point is that we are talking about artificial, rather than natural systems: since we design them (formally specifying _all_ their properties), it should be possible for us to know exactly what they are going to do.

Any set of vectors which satisfies this definition is a set of relative rates. In the course of an experiment over t units of time, an observer is affected only by those system steps whose finishing times are specified and are less than or equal to t.

2. Behaviors of Observers and System Complexes

An experiment is really a conversation between the observer and the system complex. The outcome is the sequence of message sets observed, and is called a behavior of the system complex. We have not yet defined, however, the third factor determining the outcome--the sequence of message sets sent by the observer, called the behavior of the observer. These two kinds of behaviors are very similar, as befits two halves of the same conversation.

A behavior is a finite sequence of message sets:

$$<t_1: (C_1, \{m_{11}, m_{12}, \ldots m_{1n_1}\}), t_2: (C_2, \{m_{21}, m_{22}, \ldots m_{2n_2}\}), \ldots, t_p: (C_p, \{m_{p1}, m_{p2}, \ldots m_{pn_p}\})>$$

where $\{m_{i1}, m_{i2}, \ldots m_{in_i}\}$ is a non-empty set of messages generated in the same step of the same system (or sent at one time by an observer) and associated with the same channel name, C_i. It is transmitted at t_i. The behavior sequence is arranged in ascending order of transmittal times, although adjacent message sets may have the same transmittal times as long as they don't have the same channel names (this would always be the case for different message sets generated in the same step of the same system, because all such message sets are transmitted at the instant the system step ends). A behavior is either a sequence of message sets from the observer to systems in the complex, or from systems in the complex to the observer; in either case transmission is defined to be instantaneous, and so a message set sent at t_i is received at t_i.

If the sequence is an observer's behavior, each t_i is an integer, and t_{i+1} equals t_i or t_i+1.

Each C_i must be in one of the observer's output channel languages, and all the m_{ij} must be in the corresponding message language. In the behavior of the complex (which is actually _observed_ behavior), each C_i must belong to the input channel language of the observer, and each t_i is a real number.

Although the number of message sets that a complex can send to an observer in a finite period is unbounded, it is always finite, and so the behavior of a complex is always a finite sequence.

3. Behavior Sets

An experiment with a given system complex and a given observer's behavior can result in an infinite set of different complex behaviors, because of the infinite number of relative rate sets the system complex can obey. This set is called a behavior set. Since each of its members represents an acceptable behavior of the formally defined complex, the entire set must be taken into consideration when deciding what the system complex does, or whether another system complex is equivalent to it.

The reason that times are recorded along with message sets is that they allow conditional predicates to be decided. For instance, the observer's question, "If the system complex sends me message set A and I then send it message set B, is message set C the next one I will receive?" can be answered by looking at all the behaviors in the behavior set which include message set A transmitted at a time before the transmission time of B in the observer's behavior. If all of these behaviors show C as the first message set sent with time greater than the transmission time of B, then the answer is yes.

We do not intend, by this glibness, to ignore serious problems. The test explained above is not effective because it is impossible to generate or examine all the members of a behavior set. The point is that as long as the infiniteness exists, it is best to acknowledge the fact, and then go on to look for ways of dealing with it.

The alternative to recording transmission times, as a way of deciding questions about conditional behavior, is to define behaviors with conditional clauses. One could then specify that the observer was to send a certain message if and/or when he received a certain message sequence from the complex. This scheme has the advantage of removing that vast and meaningless variety from the behavior set stemming from non-critical differences in transmission times--who cares if a message set arrives at OST 9.561 or 9.562, except in the rare cases when something critical happens at 9.5614? The disadvantage, however, is that waiting for a message results in an infinite experiment if it never comes, and we have unnecessarily involved ourselves in the halting problem. This is a sufficient reason for rejecting the idea of conditional behaviors.

IV. EQUIVALENCE

1. A Definition of Equivalence

Now that we have a precise characterization of the relationship between a system complex and its environment that corresponds to the input-output relationship induced by a partial recursive function, we are ready to define an equivalence relation for system complexes analogous to functional equivalence.

Two system complexes are equivalent with respect to a set of independent observers if, for every observer: for every possible behavior of that observer, the behavior sets produced in response by the two system complexes are identical.

This definition is exactly like a mapping equivalence, such as the one in [1], in its worst aspect: the sheer infiniteness of cases to be verified. In the particular definition Berry uses in his proof (which he calls operational equivalence), two information structure models are equivalent if and only if there is an essentially isomorphic mapping between the sets of computations generated by each one. Since the set of states in these computational sequences is infinite, and the state succession is nondeterministic, proving such an isomorphism is a tall order. In our definition, of course, the response of the system complex to one behavior of an observer corresponds to an information structure model; we have to prove that the two infinite behavior sets produced are identical, i.e. isomorphic. Berry deals with the complexity by confining his proof technique to two information structure models which are very similar, so that he can work only with their differences, and assume anything he doesn't mention is isomorphic in the two models. We could use this proof technique, (and will, in the sample proof in section IV.3), but another approach to the problem is suggested in IV.2.

If our claim that our equivalence relation is more general than operational equivalence is true, then it should be possible, by relaxing the definition of operational equivalence, to get a definition resembling ours somewhat. This might be the case if the mapping between information structure sequences could be defined so that only certain states in the sequences had to correspond-- and the choice of those states was determined by the content of the states themselves, not something more global like position in the sequence. The mapping, then, would actually be defined on a projection of the state which selected a portion intended to be observed by the environment. This definition would differ from ours mainly because of fundamental differences between the models of computation being used, not because of the notions of equivalence.

There is no doubt that this is a very strict definition, but some equivalence relations under it do come to mind. One case will be illustrated by the equivalence proof in section IV. 3. Another is that the finite process structure model of a system complex enables us to find output-free processes, i.e. processes that are not observed and cannot affect other computations. These processes can be removed without altering any behavior of a system complex, and so this definition of equivalence could be used in proofs about such transformations, but even here there are design considerations to be examined. Suppose the situation were described by the fragment of a finite process structure shown in Figure 4 (dangling arcs come from or go to parts of the graph that are not shown). The output-free process is marked; both the 1-arcs labeled "x" are associated with the same production. Then the only way to eliminate the useless process from the computations of the system would be to change the general production whose antecedent matches in L_1, L_2, and perhaps some other languages, into a much more specific production whose antecedent does not match any string in L_2. In the implementation of the design, that might require having several specific operators instead of one general one, at an obvious cost.

One very important equivalence relation is that between the different system complexes describing the same design at different levels of abstraction. As long as the mapping between two levels is an isomorphism at the interface of the complex with an observer, that observer should always see the same behavior sets from both complexes. This would be the case when the mapping from the lower (more detailed) level to the higher level involved hiding processes in RPR's, because none of the information in the lower level would really have been lost at the higher level--making an isomorphism possible. The advantage of the higher level description, of course, is that the hidden computational specifications inside RPR's are gone from the structure and complexity of the system complex at the higher level.

2. Specialized Definitions of Equivalence

As mentioned before, it will be very difficult to deal with most useful types of equivalence relation using this definition. What we must do, in practice, is find abstractions of behavior sets which are more manageable than the behavior sets themselves.

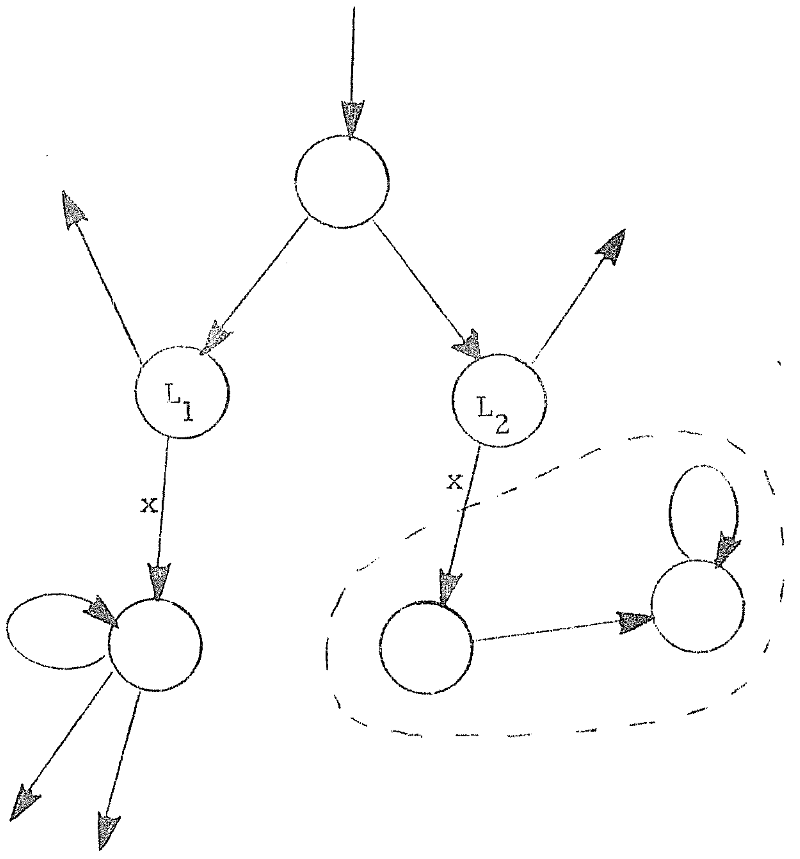


Figure 4.

A certain abstraction of a behavior set can be called an intrinsic behavior, and we can prove a specialized equivalence between two system complexes (where the specialization is defined by the abstraction) by showing that their intrinsic behaviors are identical under any behavior of the observer. It is quite apparent that different users will have different ideas of what an appropriate abstraction is, and so it will not be possible to define one universal form of intrinsic behavior.

The closer our intrinsic behavior comes to being decidable in a finite number of experiments, the better chance we have of using it to prove non-trivial equivalences. Here is an example with some obvious applications. Each behavior in a behavior set is associated with at least one set of relative rates under which the system complex produces it. Suppose we consider only those behaviors produced by relative rate sets in which all system step times and all transmission times fall within certain bounds, on the assumption that any implementation we are interested in will satisfy those bounds. Then suppose we define intrinsic behavior as the set of message sets which is the intersection of all the sets of message sets found by regarding each individual behavior as a set instead of a sequence. This intrinsic behavior tells us what message sets the observer will <u>always</u> receive from the system complex, as long as it obeys the bounds on relative rates.

This intrinsic behavior is especially interesting because it can be approximated by one experiment. One behavior observed from an acceptable set of relative rates will include all the message sets that can possibly be in the intrinsic behavior. Successive experiments will refine it, but the experiments can be stopped at any time that the approximation seems good enough.

3. A Proof of Equivalence

As an example of these ideas, we will give a proof of equivalence based on the general definition.

In section II.3 we showed a finite process structure for a system complex in which connectedness relationships showed that Observer 1 and the domain connected to it are completely independent of the rest of the system complex. The boundary between this domain and the rest of the system complex is not a system boundary, but it could be. The proof will show that the complex resulting from splitting the original left-hand system into two systems is equivalent to the original system complex.

When we speak of splitting systems, we are referring to a way of factoring a computation based on a regular language partitioning of its process state set. For a system represented by the left-hand part of the finite process structure in Figure 2, we can see that any process state that ever appears in its σ is either in the regular language L_1 = (language (A) $\cup$ language (B) $\cup$ language (C) $\cup$ language (D)) or in the regular language L_2 = (language (E) $\cup$ language (F)). The system can be split by making two SR's to replace it, each with the same set of productions. The only difference is that all the initial states in L_1 become the initial state set of the other.
state set of the other.

A finite process structure for the new system complex could look exactly like Figure 2 except that there would be an additional system boundary. The argument that the process state sets of the two new systems are always contained in L_1 or L_2, respectively, is based on considering the three ways that a process state can enter a σ: it can be in the initial σ, it can be generated by the system, or it can be accepted as a message. The initial σ's of the two new systems are certainly partitioned correctly. It is obvious from the finite process structure that states in L_1 or L_2 can only generate states in L_1 or L_2, respectively, under the productions of the original system. Finally, the split has no effect whatsoever on message reception and acceptance, since all messages generated are sent to all systems of the complex.

Note that any system can be split like this as long as (1) it has a finite process structure whose nodes can be partitioned into two subsets, with L_1 being the union of all the node languages in one subset, and L_2 the union of the other, and (2) no arc component has an origin node in one of the two languages and a destination node in the other. Figure 5 is an example somewhat different from Figure 2. The complex has only one system, and its nodes can be partitioned into three subsets, L_1, L_2, and L_3, which are not connected directly by any arcs. This system could be split into two systems along the lines $(L_1 \cup L_2)$ vs. L_3, $(L_1 \cup L_3)$ vs. L_2, or $(L_2 \cup L_3)$ vs. L_1, according to the above procedure. This is not to say that the new system complexes would be equivalent to the original, but only that the process state set of each new system would always be contained within its assigned language. The following theorem will give sufficient conditions for the new system complex to be equivalent to the original.

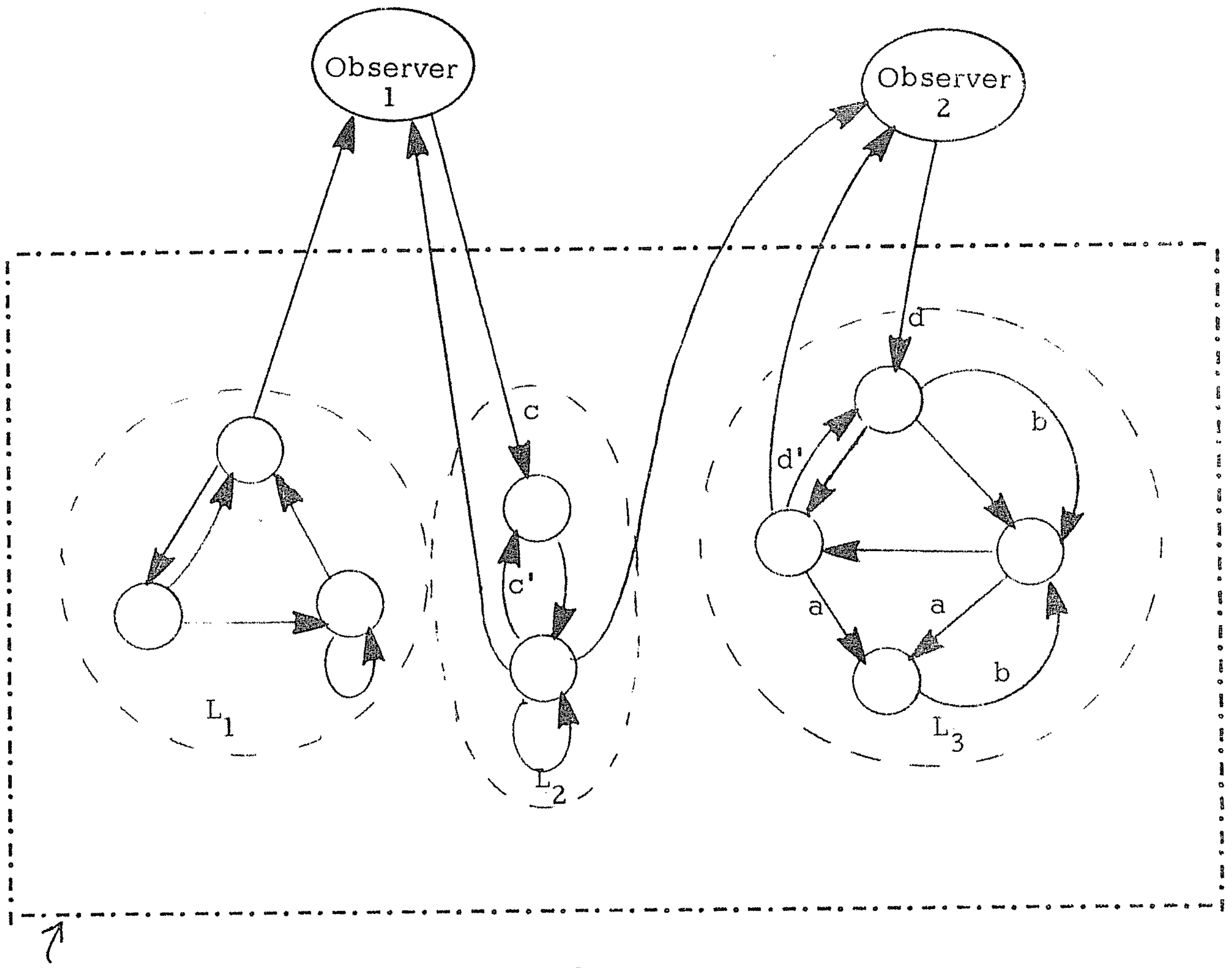

Observer 1
Observer 2
c
c'
d
d'
a
a
b
b
L_1
L_2
L_3
boundary of system and system complex

Figure 5.

Theorem Given a set of SR's and a set of independent observers such that there exists a finite process structure describing them in which one system consists of part or all of two separate domains of influence, and there is no observer to which both those domains of influence are connected, then the system complex derived from the original by splitting that system along the boundary of the domains of influence is equivalent to the original with respect to the given observers.

Proof For any observer O_i, and any behavior of that observer B_{ij}, let P_{ij} be the behavior set produced by the original system complex, and Q_{ij} be the set produced by the transformed one. We must show that $P_{ij} = Q_{ij}$. We will do this by showing first, that for any set of relative rates of the original complex, there is a set of relative rates at which the new one exactly simulates the original, showing exactly the same behavior to the observer, so $P_{ij} \subseteq Q_{ij}$. Then we will show that for every set of relative rates for the new complex, there is a set of relative rates at which the original complex exactly simulates it to the observer, so $Q_{ij} \subseteq P_{ij}$.

Part I: $P_{ij} \subseteq Q_{ij}$

Suppose the original system complex has n systems (and the system to be split has index n), so a set of relative rates governing the production of a behavior in P_{ij} is a set of vectors $\{R_1, R_2, \ldots, R_n\}$. The new complex has n+1 systems (the two new ones are given indices n and n+1) so that a set of relative rates for it is $\{S_1, S_2, \ldots, S_n, S_{n+1}\}$. Derive $\{S_1, S_2, \ldots, S_n, S_{n+1}\}$ from $\{R_1, R_2, \ldots, R_n\}$ as follows:

(1) $S_i = R_i$, $i = 1, 2, \ldots, n-1$, except that each V_{ij} in S_i has n+1 components instead of n. The observer may be connected to a domain of influence in the new complex containing system n, or containing system n+1, but not both. Assume without loss of generality that it is connected to a domain containing system n. Then $u_{i,j,n+1}$ in S_i ($i = 1, 2, \ldots, n-1$; $j = 1, 2, \ldots$) is irrelevant, and can be any number which does not violate the constraints on relative rate sets.

(2) $S_n = R_n$ except that each V_{nj} has n+1 components instead of n, and $u_{n,j,n+1}$, $j = 1, 2, \ldots$ is any number which does not violate the constraints on relative rate sets (it is irrelevant because we know from the finite process structure that the split systems never accept messages from each other).

(3) S_{n+1} is any vector that does not violate the constraints on relative rate sets.

We assert that the behavior of the new complex observed by O_i under $\{S_1, S_2, \ldots S_n, S_{n+1}\}$ in response to B_{ij} is identical to that produced by the original complex under $\{R_1, R_2, \ldots, R_n\}$ and seen by O_i in response to B_{ij}.

The truth of this assertion is obvious when it is pointed out that an observer cannot be affected in any way by a domain of influence it has no connection with, in this case the domain including system n+1. As for all the other domains, their computations are exactly the same in the new and old complexes, at the specified relative rates. System n in the new complex generates the same sequence of process state sets as did the corresponding half of system n in the original, and all the other systems of the complex are untouched. Furthermore, all system steps and message transmission in the domains of the new system complex that do not include system n+1 occur at exactly the same times, so that all message exchanges are undisturbed, including those between the observer and the system complex.

Part II: $Q_{ij} \subseteq P_{ij}$

Assume that the behavior of the new complex we are trying to simulate with the original complex is produced under a set of relative rates $\{S_1, S_2, \ldots, S_n, S_{n+1}\}$, and that O_i is connected to a domain containing system n (as above). Construct $\{R_1, R_2, \ldots, R_n\}$, a set of relative rates for the original complex, as follows: $R_i = S_i$, $i = 1, 2, \ldots, n$, except that the (n+1)st component of each V_{ij}, $u_{i,j,n+1}$, is removed.

As before, nothing seen by O_i is affected by this change from the new system complex to the original, so the observed behaviors are identical. The computations of system n+1 in the new complex are now constrained to run in lockstep with those of system n, but no domain affecting O_i contains system n+1, so the constraint is meaningless.

Q.E.D.

4. Conclusions

To summarize, we feel that there are two important reasons why this definition of equivalence deserves serious attention. The first is that it depends only on behavior which is always well-defined and always effectively observable. It is still true that equivalence cannot be proven by exhaustive testing of cases, but the situation is much better, from a systems analysis point of view, than for functional equivalence or mapping (operational) equivalence. The functional equivalence of two functions is not effectively verifiable even for a single argument, because one or both of the functions may be undefined on that argument. The verification of mapping equivalences requires an examination of internal states of a computation--it is easy to see that, in practice, two systems could not be tested for operational equivalence without tampering with their implementations (and incurring all the risk that something will be changed thereby) to make internal states observable.

The other advantage of the definition of equivalence we have presented is that it allows the designer maximum freedom to specify exactly which aspects of computations are significant and must be preserved. Since all other aspects are considered variable, the classes of system complexes which can be proved equivalent to each other are as large and interesting as possible. The specification of observers determines which message transmissions are observable outside the complex--all other computational sequences are considered irrelevant to equivalence. Furthermore, the use of intrinsic behaviors can single out certain aspects of the observed message sequences for attention, so that even these need not be rigidly preserved. We should not overlook the usefulness of very relaxed versions of equivalence as approximations to rigorous equivalence, because of the considerable difficulties involved in proving any non-trivial equivalences between system complexes.

REFERENCES

[1] Berry, Daniel M. "The Equivalence of Models of Tasking." SIGPLAN Notices 7, no. 1, January 1972.

[2] Fitzwater, D. R., and Smith, Pamela Z. "A Formal Definition Universe for Complexes of Interacting Digital Systems." Computer Sciences Technical Report #184, University of Wisconsin, Madison, Wisconsin, 1973.

[3] Johnson, Robert T. Proving Assertions About the State Structure of Formally-Defined, Interacting Digital Systems. Ph.D. Thesis, University of Wisconsin, Madison, Wisconsin, 1973.

[4] Luckham, D. C., Park, D. M. R., and Paterson, M. S. "On Formalized Computer Programs." Journal of Computer and System Sciences 4, no. 3, June 1970.

[5] Manna, Zohar. "The Correctness of Programs." Journal of Computer and System Sciences 3, no. 2, May 1969.

[6] Smith, Pamela Z, and Fitzwater, D. R. "Finite Process Structures." To be published as a Computer Sciences Technical Report, University of Wisconsin, Madison, Wisconsin, 1974.

[7] Wegner, Peter. "Operational Semantics of Programming Languages." SIGPLAN Notices 7, no. 1, January 1972.

BIBLIOGRAPHIC DATA SHEET	1. Report No. WIS-CS-74-213	2.	3. Recipient's Accession No.

4. Title and Subtitle
A CONCEPT OF EQUIVALENCE BETWEEN FORMALLY DEFINED COMPLEXES OF INTERACTING DIGITAL SYSTEMS

5. Report Date
April 1974

6.

7. Author(s)
Pamela Z. Smith and D. R. Fitzwater

8. Performing Organization Rept. No. 213

9. Performing Organization Name and Address
Computer Sciences Department
University of Wisconsin
1210 West Dayton Street
Madison, Wisconsin 53706

10. Project/Task/Work Unit No.

11. Contract/Grant No.

12. Sponsoring Organization Name and Address

13. Type of Report & Period Covered

14.

15. Supplementary Notes

16. Abstracts

The environment of a formally defined complex of interacting digital systems is specified as a set of ideal, independent observers who can provide input as well as monitor any observable computation. Precise definitions are given for the behavior of an observer toward a system complex, the behavior of a system complex toward an observer, and the relative rates at which the systems of a complex can finish steps and transmit messages. These ideas are used in a very general definition of functional equivalence between system complexes; a sample equivalence proof and suggestions for more specialized forms of equivalence are added. It is argued that this definition compares favorably with other notions of equivalence because it is based on effective observations and allows equivalence classes to be as rich as possible.

17. Key Words and Document Analysis. 17a. Descriptors

asynchronous communication
system equivalence
proving assertions about systems
systems design tools

17b. Identifiers/Open-Ended Terms

17c. COSATI Field/Group

18. Availability Statement Available to public	19. Security Class (This Report) UNCLASSIFIED	21. No. of Pages 33
	20. Security Class (This Page) UNCLASSIFIED	22. Price