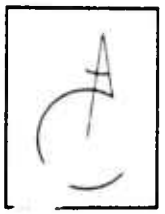


MASSACHUSETTS
COMPUTER ASSOCIATES, INC
26 PRINCESS STREET, WAKEFIELD, MASS. 01880 · 617/245-9540

FINAL TECHNICAL REPORT

(1 December 1973 - 31 March 1975)

For the Project

"Development of Theoretical Foundations
for Description and Analysis of Discrete Information Systems"

Report Date: 7 May 1975

Principal Investigator:

Anatol W. H[illegible] (617) 245-9540

Project Manager:

Robert E. Millstein (617) 245-9540

Program Code Number - 4D30

Contract Number MDA903-74-C-0162

Contractor:	Massachusetts Computer Associates Subsidiary of Applied Data Research, Inc.
Effective Date:	1 December 1973
Expiration Date:	31 March 1975
Amount:	$203,818.00

Sponsored by
Advanced Research Projects Agency
ARPA Order Number - 2581

Approved for public release;
distribution unlimited

MASSACHUSETTS
COMPUTER ASSOCIATES, INC.
26 PRINCESS STREET, WAKEFIELD, MASS. 01880 • 617/245-9540

May 7, 1975

Director
Advanced Research Projects Agency
Attn: Program Management
1400 Wilson Boulevard
Arlington, Virginia 22209

Dear Sir:

Enclosed please find the Final Technical Report for the contract entitled "Development of Theoretical Foundations for Description and Analysis of Discrete Information Systems".

It is submitted in accordance with the terms of Contract Number MDA903-74-C-0162.

Very truly yours,

James V. Botto
Manager, Administration

JVB/jr

Enclosures

UNCLASSIFIED
Security Classification

ADA955303

DOCUMENT CONTROL DATA - R & D
(Security classification of title, body of abstract and indexing annotation must be entered when the overall report is classified)

1. ORIGINATING ACTIVITY *(Corporate author)*
Mass. Computer Associates, Inc., 26 Princess Street, Wakefield, Massachusetts 01880
Subsidiary of Applied Data Research, Inc.

2a. REPORT SECURITY CLASSIFICATION
UNCLASSIFIED

2b. GROUP
N/A

3. REPORT TITLE
FINAL TECHNICAL REPORT (1 December 1973 - 31 March 1975)
For the project: "Development of Theoretical Foundations for Description and Analysis of Discrete Information Systems"

4. DESCRIPTIVE NOTES *(Type of report and inclusive dates)*
Final Technical Report (1 December 1973 - 31 March 1975)

5. AUTHOR(S) *(First name, middle initial, last name)*
Anatol W. Holt

6. REPORT DATE
May 7, 1975

7a. TOTAL NO. OF PAGES
287

7b. NO. OF REFS

8a. CONTRACT OR GRANT NO.
MDA903-74-C-0162

b. PROJECT NO.
ARPA Order No. 2581

c. Program Code 4D30

d.

9a. ORIGINATOR'S REPORT NUMBER(S)
CADD-7505-0811

9b. OTHER REPORT NO(S) *(Any other numbers that may be assigned this report)*
N/A

10. DISTRIBUTION STATEMENT
Approved for public release; distribution unlimited.

11. SUPPLEMENTARY NOTES

12. SPONSORING MILITARY ACTIVITY
Defense Supply Service - Washington
Room 1D245, The Pentagon
Washington, D.C. 20310

13. ABSTRACT

Volume I

First section
Development of ideas pertaining to the interpretation of Petri nets.

Section section
Initial specification of the mathematical foundation of our theory.

Third section
Marked graph theorems for the development of Communication Mechanics.

Volume II

This volume presents contributions to both theory and practice of various classes of net models as well as the connections between them. Chapters I, II, IV, V, and IX are devoted to the theory and chapters III, VI, VII, and VIII are devoted to the practice.

DD FORM 1 NOV 65 1473

UNCLASSIFIED
Security Classification

14. KEY WORDS	LINK A		LINK B		LINK C	
	ROLE	WT	ROLE	WT	ROLE	WT
Petri Net						
Information Theory						
Marked Graphs						
Nets						
Communication Mechanics						

VOLUME I

TABLE OF CONTENTS

I. Introduction

This report consists of three sections. The first is a further development of ideas pertaining to the interpretation of Petri nets - more exactly, of the special class of Petri nets which we have been laboring to define. Although the first section can be read independently of previous writings, it is best understood in the context of our last report.

The second section is our first attempt to specify the mathematical foundation for our theory. The purpose of this section is purely to set forth the important structural ideas, not to show how these ideas relate to the semantic content they are designed to support. It is our hope and expectation that we have created the platform that will permit the rapid development of computational techniques for the solution of significant system problems incorporating and extending the earlier work on marked graphs to nets exhibiting the phenomenon of concurrency as well as the phenomenon of conflict.

The third section reports on a new theorem due to Fred Commoner, about marked graphs, which has an important bearing on the development of Communication Mechanics.

II. Presence Structures

The structure of a system specifies relations between possibilities - possibilities of the happening of events or the holding of states. In our formalism we will not begin with states and events as the source of atomic possibilities but rather, new elements to be called presences. As is naturally true with states and events, the semantic content of any presence may be expressed by a sentence - a sentence which someone who reports on the fulfillment of that possibility asserts.

Also, as with the two-way classification of possibilities into events and states, presences are classified into two types.

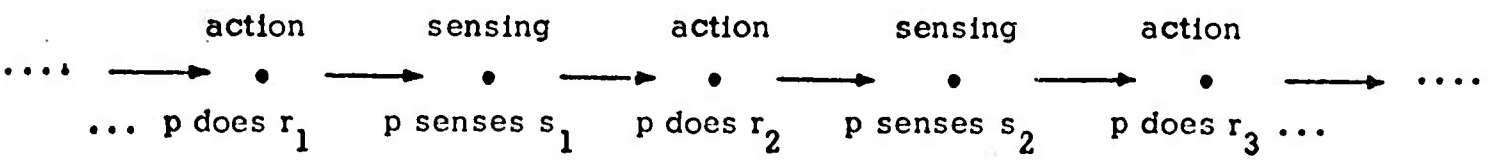
E1.

.1 actions

.2 sensings

Every action and every sensing is an action or sensing by a particular part. A behavioral sequence of a part is represented as a sequence of presences in which actions and sensings alternate.

E2.



The arrows which connect the presences in such a sequence are part of the 'becomes' relation - a relation already discussed in the last section and the subject of much that will follow below.

The content of E2 may best be understood by imagining the use of it by a man who tracks the behavior of p as described in E2:

E3. He registers p's advance, successively moving his finger from vertex to next vertex in the sequence E2. This establishes a 1-1 correspondence between 'becomes' arrows and tracking actions.

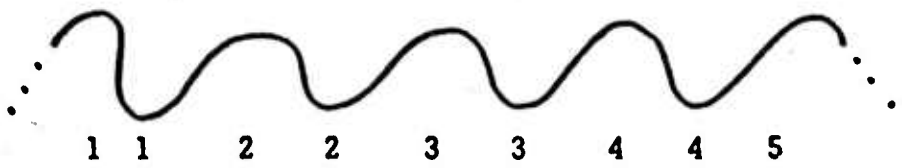
Exactly equivalent to E3 would be the following:

E4. He successively asserts the sentences associated with the vertices in E2 by uttering them.

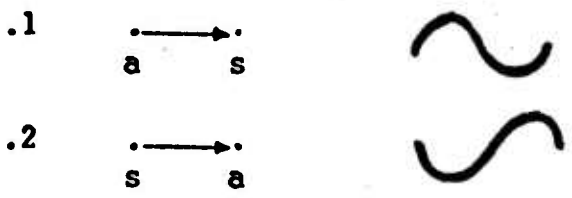
Now it is vital to notice that in E3 each tracking action affects two vertices: the one from which the finger is removed and the one on which the finger is deposited. In terms of E4, the equivalent fact is this: every act of sentence assertion is also the unassertion of a particular other sentence - namely, the last one uttered in the tracking sequence. Unasserting is not the same thing as denying. Thus, in E2, to assert 'p senses s_2' is not to deny 'p does r_2'. In other words, if one were to list those sentences in E2 which represent what, now, is not the case, 'p does r_2' would not be among them.[1] We will now give an example of tracking with two sentence types in which the significance of asserting and unasserting can be concretely seen.

[1] In a subsequent section of this chapter, we will describe denial (and undenial) as part of the tracking activity, just as we are now discussing assertion and unassertion.

E5.



Now imagine a hiker who progresses from left to right and someone who tracks his advance with sentences of the forms 'he is on hill i' and 'he is in valley i'. The tracker having asserted 'he is in valley i', one cannot deduce that, now, the hiker is not on hill i. And also: the tracker having asserted 'he is in valley i' but not yet unasserted it, one cannot deduce that, now, the hiker is not on hill i + 1; about all other hills and valleys in the sequence, one can deduce that, now, he is not on them or in them. Based on E5, we can use hills to symbolize presences of type <u>action</u> and valleys to symbolize presences of type <u>sensing</u> and represent a pair in sequence by the pictures:



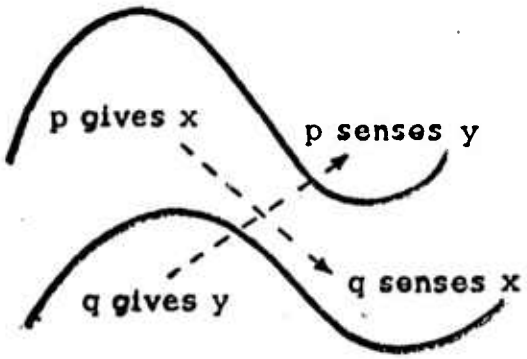
Here is a general statement of the meaning of the 'becomes' relation in terms of its effect on the behavior of a tracker:

E6. For two presences α and β, $\alpha \rightarrow \beta$ then <u>any act that asserts β is an act that unasserts α, and conversely.</u>

We are now prepared to give formal expression to the following semantic principle:

Every action is part of a transaction - more specifically, we will assume, a transaction between two parts. Here is our first general picture of a transaction, or event.

E7.



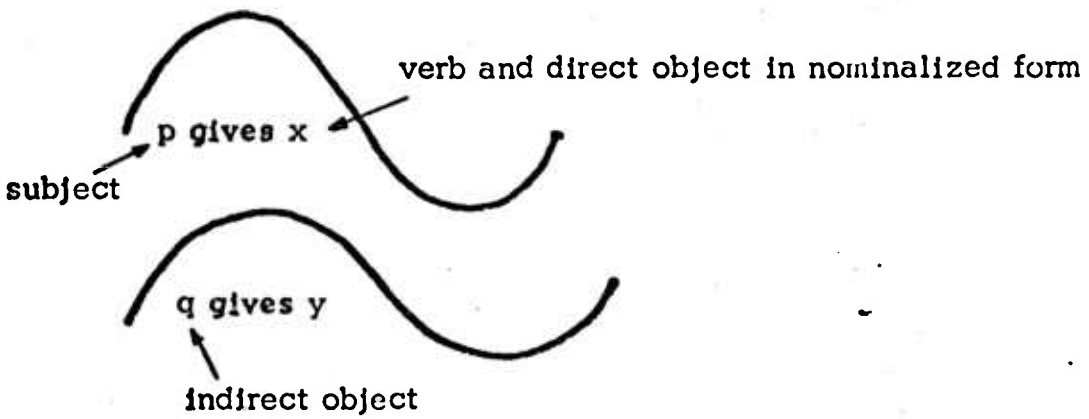
English sentences describing actions do so with the help of a number of syntactic positions of which the most important are:

subject	who does it
verb	what he (or it) does
direct object	to what (or whom)
indirect object	with reference to whom

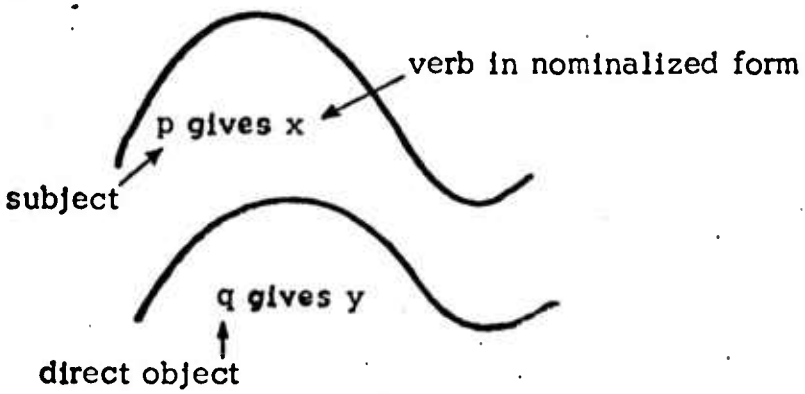
The way in which sentences describing actions map onto E7, broadly, in the following ways:

E8.

.1 With transitive verbs taking an indirect object



.2 With transitive verbs not taking an indirect object



The role of y will shortly be explained.

Examples:

E9.

	Sentence	p	x	q	y
.1	John hits the key	John	a hit	the key	a hit reception
.2	John kicks the ball to Bill	John	a ball kick	Bill	a ball kick reception
.3	The train approaches the station	The train	an approach	the station	an approach reception

Thus, in accordance with E7, E9.1 yields:

E10. John **gives** a hit ⟶ John **senses** a hit reception

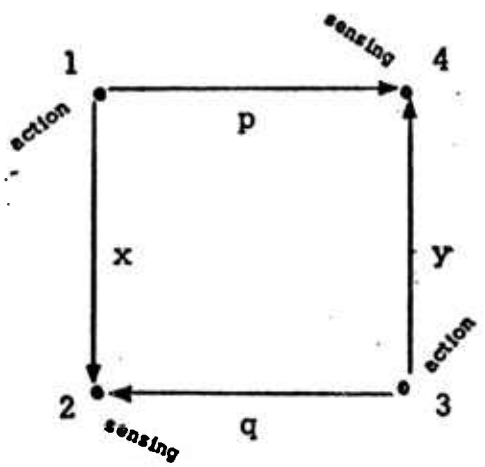
The key **gives** a hit reception ⟶ The key **senses** a hit

In the sentences in E10, the bold faced words are part of the sentence form while the noun phrases in simple type are particular values for the variables of the form. Other forms, yet to be shown, for sentences associated with presences will obey the same convention.

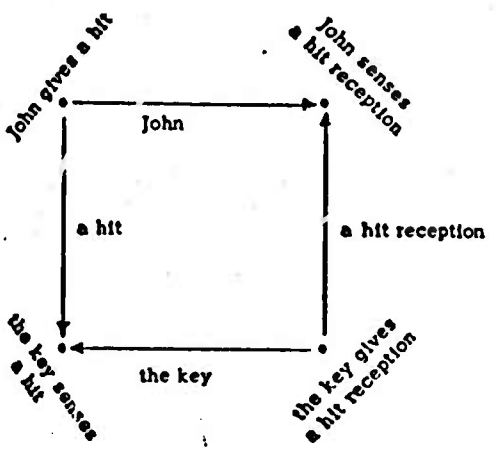
In this pattern, there is a step of propagation for each of the four identity elements - John, the key, a hit, and a hit reception; each is a step from an action to a sensing: John and the key are treated as parts while the hit and its reception are treated as values, exchanged by the parts in this transaction. The four sentences in E10 represent presences and the four arrows all are part of the 'becomes' relation. Corresponding to each 'becomes' arrow there is a noun phrase which is repeated at its two ends - a noun phrase representing the 'something' which is propagated from one presence to another.

E11.

.1

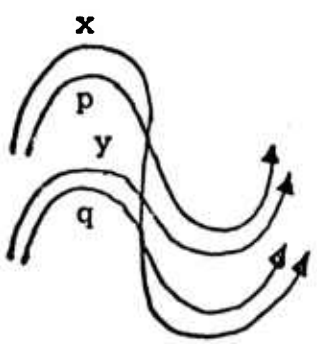


With reference to E10:



and, in the style of E7:

.2



How do we translate E11.1 into behavior of the tracker? By E6, the q arrow means that the unassertion of 3 and the assertion of 2 must be part of a single act; the x arrow means that the unassertion of 1 and the assertion of 2 must be part of a single act. Reading the p and y arrows analogously, we discover that figure E11.1 represents a single action on the part of the tracker in which 1 and 3 are unasserted and 2 and 4 asserted. To execute this action in the manner of E3 requires of him to utter the sentence 's_2 and s_4', where s_2 is the sentence associated with the vertex 2 and s_4 the sentence associated with the vertex 4, having previously asserted s_1 (associated with vertex 1) and s_3 (associated with vertex 3) and not yet unasserted them. The force of the assertion 's_2 and s_4' is to be understood as asserting s_2 and s_4, thus unasserting <u>both</u> s_1 and s_3. Also it is to be understood that s_2 and s_4 may subsequently be unasserted in separate tracking steps.

If the tracker carries on by sentence uttering then the sentences which a new assertion unasserts may have been the subject of assertion some unknown number of steps earlier in the sequence of tracking actions. The question thus arises: if the tracker performs his actions by sentence uttering, how is the connection formed between the sentence now asserted and the sentences which it unasserts? We can give the answer with reference to E11. <u>In uttering s_2, he mentions x and q. This unasserts the last sentence he uttered in which x was mentioned and the last sentence he uttered in which q was mentioned.</u> In this sense, both s_2 and s_4 will point backwards to the two relevant sentences which they unassert. This is exactly how verbal tracking reports always function. If we were last told: 'the airplane is about to take off' and later told 'the airplane is taking off' with no mention of the airplane in between, we know that the latter sentence unasserts the former because of the repeated mention of the airplane.

In all of the examples E9, and also generally in our formalism, there is a special semantic relationship between the values x and y which are exchanged in an event - signaled by the word 'reception': if x is called v, then y is called a v-reception.

This special relationship expresses itself both in the action pair and in the sensing pair participating in an event.

E12.

.1 The part that gives v is active while the part that gives a v-reception is passive. It presents itself for the administration of a v-shock while its partner delivers it. We will henceforth call the delivery a positive action and the presentation a negative action - or, an inaction.

.2 What the two parts sense, is the same fact seen from two different points of view - namely, that a v-shock has passed from the one to the other.

In the case of 'John hits the key':

E13.

.1 The key presents itself in its up-position so that it may be depressed by the strike which John delivers.

.2 John senses the increase in pressure against the key-bed even as the key senses the increase in pressure against John's finger.

The following, then, are formal properties of presence structures.

E14.

.1 Every action becomes exactly two sensings - (value and part propagation)

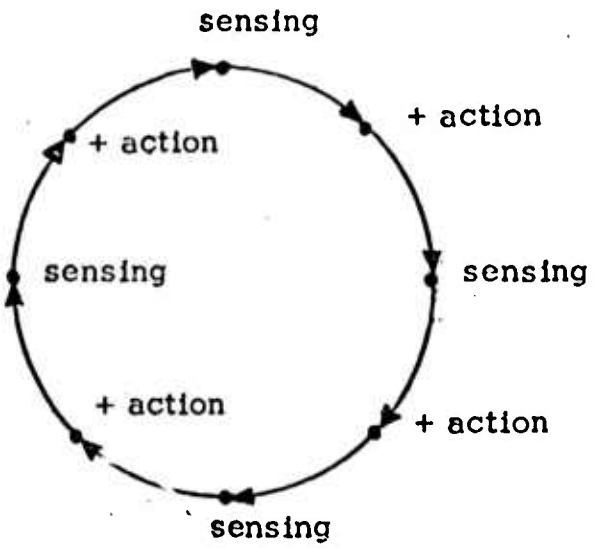
.2 Every sensing has exactly two actions as 'becomes' predecessors (value and part propagation, backwards)

.3 If action a_1 becomes sensing s_1 and s_2, then there exists another action, a_2 of a different part which also becomes s_1 and s_2 (as in E11.1).

This ends, for now, the discussion of events.

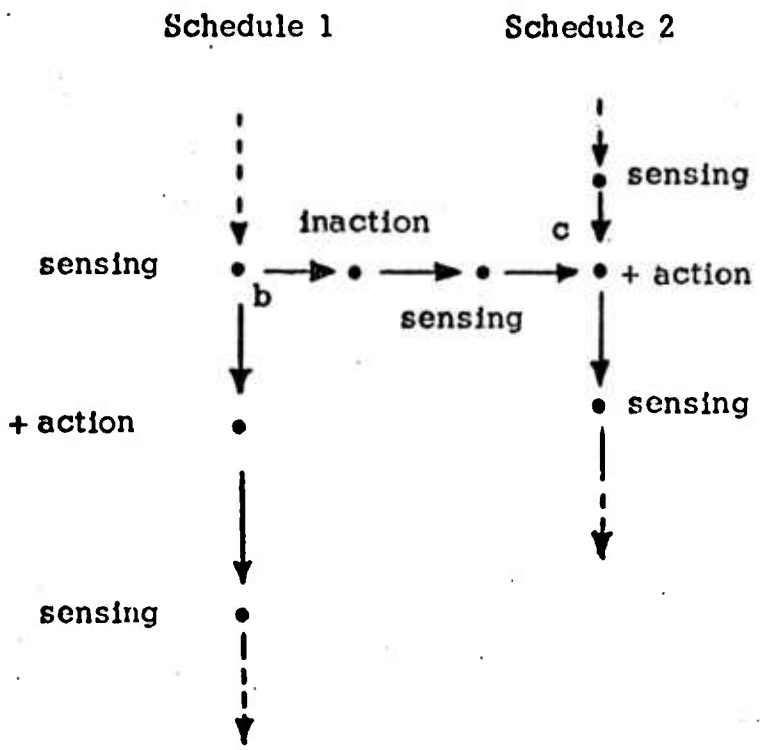
We will assume that the total behavioral possibilities of a part can be represented in terms of a set of possible cyclic schedules of action with connections between them. Here is a picture of such a schedule:

E15.



The connections between schedules are established thus: each sensing which signals the completion of an action on a schedule can be followed by an inaction which brings the part to a different schedule - in other words, that after each transaction, the part can be deflected from its present schedule to another:

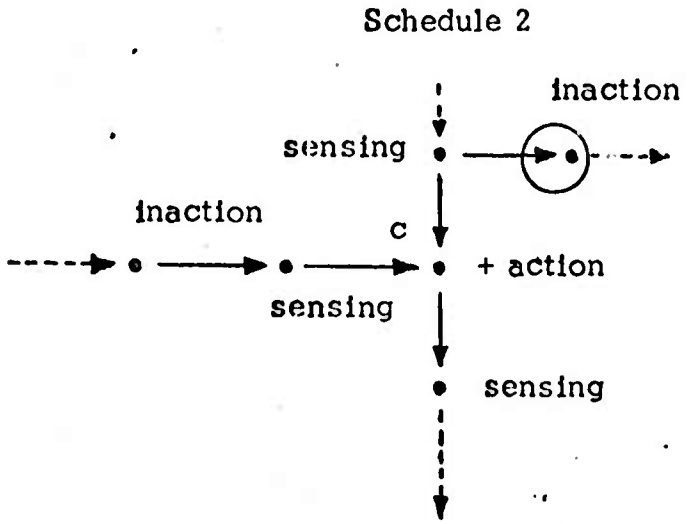
E16.



(Note: the inaction could not become a sensing on schedule 2 because that sensing would then have two actions belonging to a single part as 'becomes' predecessors, contrary to E14.3)

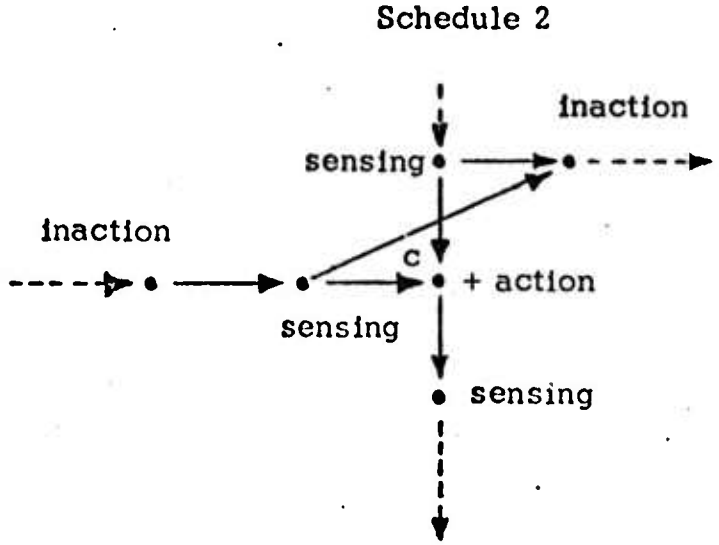
In E16 we see a sensing which points to two presences of type action (the fork at b) and an action which is pointed at by two sensings (the join at c). Just as the sensing at b allows a branch off of schedule 1 so, of course, does the sensing on schedule 2, just prior to c permit a branch away from that schedule.

E17.



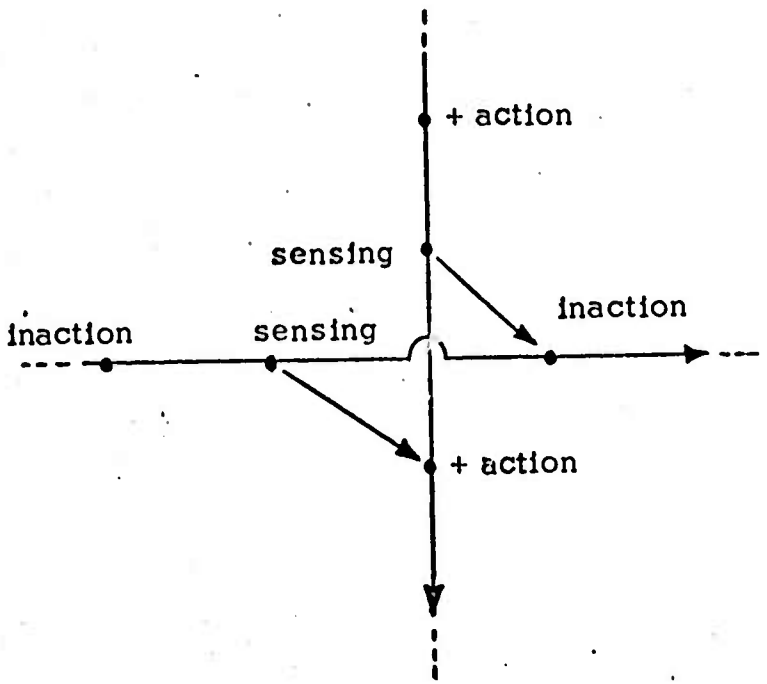
We now assume that <u>either sensing prior to the action at c can lead to the encircled inaction in E17:</u>

E18.



or, redrawn in a geometrically more suggestive manner:

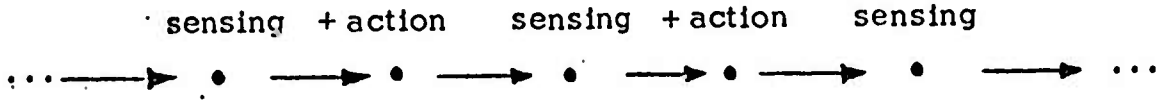
.2



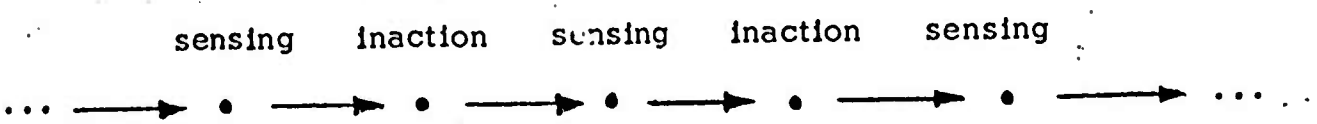
Thus the total behavioral possibilities for a part are covered by:

E19.

.1 A set of cyclic schedules to be called <u>active modes</u> (of the part)



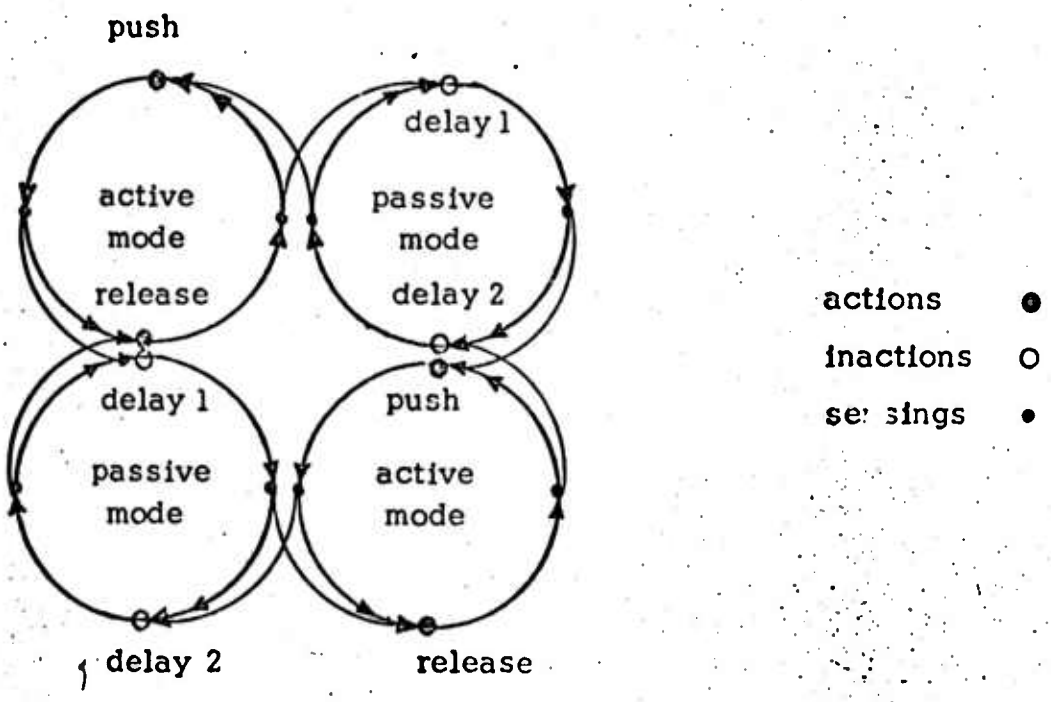
and .2 Cycles of schedule switching which we will call <u>passive modes</u>



with .3 The active and passive modes of a part connected to one another, as in E18.1.

We will now show an example: the behavior of a key pusher with enough freedom of action to transmit information, e.g., Morse Code.

E20.



E21. The key pusher p has a two-step action cycle: push, release. Delay can cause him to shift phase - to enable a release at a time (counted in steps) when another push would have been enabled and vice versa. Thus, p's action possibilities may be represented in two circuits of push/release which are in opposite phase:

.1	phase 1	phase 2
active mode 1	push	release
active mode 2	release	push

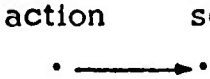
Every delay resulting in a phase shift can be followed by another delay resulting in a shift back. Thus, there comes into being two delay circuits which may also be aligned with respect to phase 1 and phase 2:

.2	phase 1	phase 2
passive circuit 1	delay 2	delay 1
passive circuit 2	delay 1	delay 2

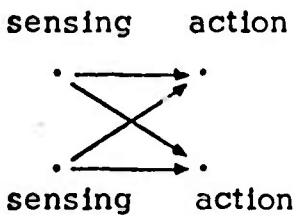
As long as the p continues in passive circuit 1, he keeps the key depressed; so long as p continues in passive circuit 2, he keeps the key undepressed.

We will now examine E20 from the structural point of view. One can think of the graph E20 as covered by the following two substructures:

E22.

.1 

8 copies in E20

.2 

4 copies in E20

(These substructures cover the arcs of E20 singly and the vertices doubly.)

The eight copies of E22.1 are "half events" - i.e., consist of two out of the four presences necessary for the construction of E11.1. They are "half" because E20 shows only the propagation pattern of the key pusher and not those of the partners with whom the key pusher transacts. (At least one of those partners is the key; who the rest might be, we will have later occasion to consider.)

E23. Now, note that <u>just as E11.1 consists of two actions which become two sensings, so E22.2 consists of two sensings which become two actions</u>. The one pattern can be obtained from the other by reversing all arrows or, equivalently, interchanging action and sensing labels. Also, in E22.2, just as in E11.1, one of the two actions is positive and the other negative.

There will be further structural comparisons to make in the sequel.

The construction E22.2 is what we will call a <u>state</u> (of a part). In contrast to E11.1, which has two presences belonging to one part and two to a <u>different</u> part, all four presences of the state belong to one part only. Given the formal properties of presence structures as expressed in E14 and E19, we can now see that:

E24. In tracking the behavior of a part successive steps pass alternately:

.1 through an event - where the part transacts with another part

.2 through a state - where the part may switch from one mode to another (as illustrated in E18.2)

We will now discuss presence structures in terms of their modes.

Every presence is a presence of some part and, hence, of some mode. Thus the presences of a presence structure S are partitioned by the modes. We will call two modes μ_1 and μ_2 state connected if there exists a state in which they meet (as in E18.2). For this relation we write $\mu_1 \circ \mu_2$. As is clear from its definition, this relation is symmetric. As we have already said, if $\mu_1 \circ \mu_2$ then μ_1 and μ_2 are modes of the same part.

We call two modes event connected if there exists an event in which they meet (as in E11.1). For this relation we write $\mu_1 \square \mu_2$. It too is symmetric. If $\mu_1 \square \mu_2$ then μ_1 and μ_2 must belong to different parts. If μ_1 and μ_2 are either event connected or state connected, then one of them must be active and the other passive (E12.1 and E18.2).

The following is a formal property which all presence structures will exhibit:

E25. .1 Modes μ and μ' belong to the same part if and only if there exists a sequence of modes $\langle \mu_1 , \mu_2 , \ldots \mu_n \rangle$ (possibly empty) such that

$$\mu \circ \mu_1 \circ \mu_2 \circ \mu_3 \circ \ldots . \mu_n \circ \mu'$$

The same fact can also be expressed thus:

.2 Modes μ and μ' belong to the same part if and only if $\mu \; o^* \; \mu'$, where o^* is the transitive closure of the relation o .

Finally, we can translate E25 into a criterion for when two presences belong to the same part:

E26. Two presences α_0 , α_n , belong to the same part, if and only if there exists a propagation path[1]

$$\alpha_0 \rightarrow \alpha_1 \rightarrow \alpha_2 \rightarrow \alpha_3 \ldots . \rightarrow \alpha_n$$ such that

if, for $0 \leq i < n$, the mode of α_i and α_{i+1} differ, then α_i is a sensing and α_{i+1} is an action - in other words, that along the path all changes of modes (if any) occur in passing through a state.

In regard to states, there are two matters still requiring discussion: The relationships between the sentences associated with the four presences of the state; the interpretation of the state in terms of tracking steps. We now turn our attention to the first of these two.

---

[1] In C10, propagation paths are defined as any sequence of presences where one is connected to the next by the relation <u>becomes</u>.

From our earlier discussion (E8 - E10) it appeared that the sentence forms associated with presences contain two variables - one for the designation of a part and another for the designation of a value. We are now in a position to explain that the variable designating a part does so by designating a part-in-some-mode - in other words, by designating a mode.

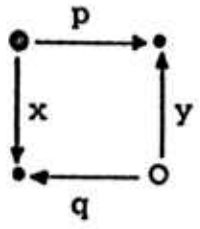
Thus, by looking at the sentences associated with two arbitrarily chosen presences, one cannot tell whether they belong to the same part or not. But, by E26, we see that the question is structurally determined.

We view the modes of a part as fundamental components of its behavioral possibilities. One can characterize every behavioral sequence of a part by stepping from sensing to sensing and stating for each step whether it switched mode or not.

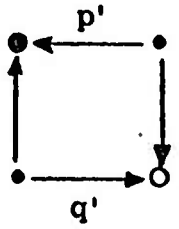
We will, hereafter, use the lower case letters p, q, etc. to designate modes and the upper case letters P, Q, etc. to designate parts. We can now go one step further than E23 in comparing events and states:

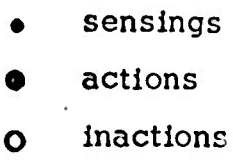
E27.

.1



.2





Event	State
p and q modes of different parts;	p' and q' modes of the same part;
p active; q passive	p' active; q' passive

It remains to examine how the values referred to in the four sentences associated with the presences of a state relate to one another. The following two principles determine this issue.

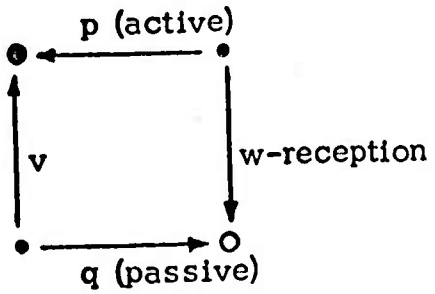
E28.

.1 If, after sensing a v-shock, a part acts positively, we will interpret its action as the transmission of the shock it received.

.2 If, after sensing a v-reception, a part acts negatively, we will interpret the reception it gives as the transmission of the reception it received.

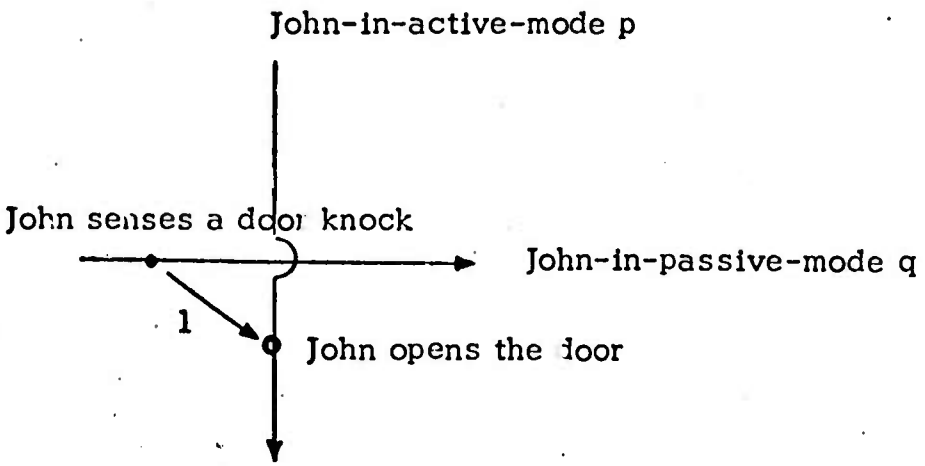
An example below will help to clarify the meaning of E28. But, before we give the example, we can apply E28 to the completion of E27.2:

E29.



E30. An example

Suppose that it is the duty of a part played by John to open the door when he hears a door knock. We model this duty as follows:



.1

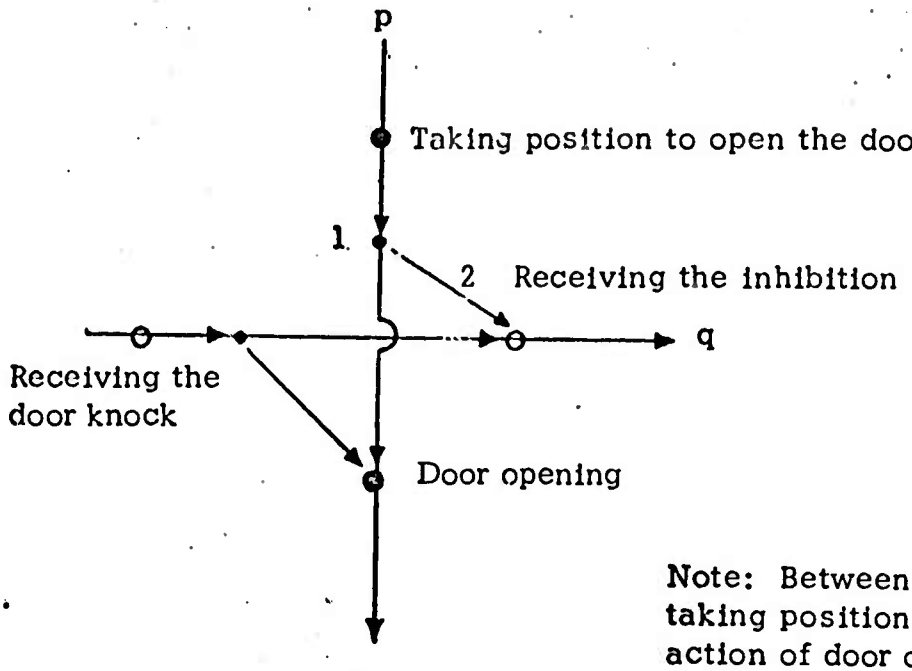
The expression 'John senses a door knock' refers the meaning of this sensing to the just prior action which delivered a knock (not an action on John's part); we can also express the meaning of the sensing with reference to the next following action (the door opening) thus:

.2 John senses the impulse to open the door

With this form we see the expression 'open the door' repeated in the two sentences at the head and the tail of the 'becomes' arrow 1 in the figure .1.

Now let us suppose that the action on the John's p mode just prior to the action of door opening is the action of taking position so that he can open the door. If, after this action he does not open the door, we will assume that it is the result of an inhibition he receives. In pictures:

.3



Note: Between the action of taking position and the next action of door opening there must, of course, be a sensing.

As in .2, on the preceding page, we can express the meaning of sensing 1 in .3 with reference to the next following inaction, thus:

.4 John senses receptivity to the inhibition.

With this form we see the expression 'the inhibition' repeated at the head and tail of the 'becomes' arrow 2 in .3.

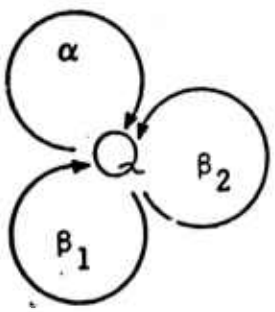
In regard to the intended interpretation, there are two behavioral sequences for John which .3 allows, but which would represent failures of system function.

.5 He may open the door without warrant (i.e., without a knock) due to a failure of inhibition after taking position.

6. He may fail to open the door although warranted (having received a knock) by virtue of an inhibition.

III. Zebras

A1.



.1 $\langle Q, \beta_1, \beta_2, \alpha \rangle$ - a zebra

.2 Q - the set of presences (finite, non-empty)

.3 β_1, β_2, α - three bijections

A2. Notations:

.1 If r is a relation, we will designate the transitive closure of r by r^* .

.2 If s is a set, we will designate the number of elements in s by $|s|$.

A3. Because Q is finite, β_1^* , β_2^* and α^* are all equivalence relations on the set Q .

A4. Definitions

.1 $M \triangleq Q/\beta_1^*$ - the set of modes

.2 $V \triangleq Q/\beta_2^*$ - the set of values

.3 $\Sigma \triangleq Q/\alpha^*$ - the set of slots

.4 $C \triangleq M \cup V \cup \Sigma$ - the set of characters

A5. Definition

$\beta \triangleq \beta_1 \cup \beta_2$ - the becomes relation

A6. Axiom

$|Q/(\beta)^*| = 1$ (Q is strongly connected by becomes)

A7. Axiom

$$\forall c_1, c_2 \in C: \; c_1 \neq c_2 \Rightarrow |c_1 \cap c_2| \leq 1$$

We can also express this as follows: For any two of the three bijections $\gamma_1, \gamma_2, \gamma_1 \neq \gamma_2$

$$|Q/(\gamma_1^* \cap \gamma_2^*)| = |Q|$$

Thus, every presence may be characterized by any pair of its three characters.

A8. Axiom

.1 $\overleftarrow{\beta}_1 \beta_2 = \overleftarrow{\beta}_2 \beta_1$

from which follows:

.2 $\beta_2 \overleftarrow{\beta}_1 = \beta_1 \overleftarrow{\beta}_2$

A9. Axiom

$\alpha \cap \overleftarrow{\alpha} = \emptyset$

A10. Axiom

$\overleftarrow{\beta}_1 \beta_2 \cup \beta_1 \overleftarrow{\beta}_2 = \alpha \cup \overleftarrow{\alpha}$

We can now prove:

A11. $\forall\, x \in Q$: $x[\overleftarrow{\beta}_2 \alpha \beta_1]x$ or $x[\overleftarrow{\beta}_2 \overleftarrow{\alpha} \beta_1]x$ but not both

This statement is equivalent to:

$\forall\, x \in Q$: $x[\beta_1 \overleftarrow{\beta}_2 \alpha \beta_1 \overleftarrow{\beta}_1]x$ or $x[\beta_1 \overleftarrow{\beta}_2 \overleftarrow{\alpha} \beta_1 \overleftarrow{\beta}_1]x$ but not both

which is equivalent to:

$\forall\, x \in Q$: $x[\beta_1 \overleftarrow{\beta}_2 \alpha]x$ or $x[\beta_1 \overleftarrow{\beta}_2 \overleftarrow{\alpha}]x$ but not both

which is an immediate consequence of A9 and A10.

A12. Axiom

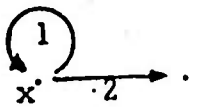
$$\forall x \in Q: \quad x[\overleftarrow{\beta}_1 \beta_2 \alpha]x \iff x[\overleftarrow{\beta}_2 \alpha \beta_1]x$$

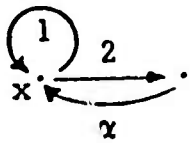
We will now look at some of the consequences of the structure specified so far. To do this, we represent zebras as directed graphs with labeled edges. The vertices correspond to presences. At each vertex, exactly three arcs enter and three arcs exit - corresponding to the three bijections. In labeling the arcs, we write '1' for arcs corresponding to β_1, '2' for β_2, and 'α' for α.

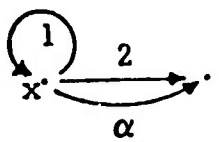
In the graph of a presence net the equivalence classes of β_1^*, β_2^* and α^* are represented as three families of circuits: circuits in which every arc is labeled 1, circuits in which every arc is labeled 2 and circuits in which every arc is labeled α. Because β_1, β_2 and α are bijections, no two circuits of the same family intersect at all, and a circuit from one family intersects a circuit of another family at, at most one vertex - because of A7. It follows, in particular, that:

A13. The graph cannot contain any cycles of length two where the two arcs are differently labeled.

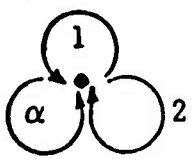
A14. $\forall\, x, y \in Q$: $x \beta_1 y$ or $x \beta_2 y$ or $x \alpha y \Longrightarrow x \neq y$

Suppose $x \beta_1 x$ and $\sim x \beta_2 x$: 

. By A10 we then must have either 

or 

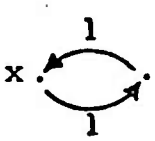
both of which violate A7. On the other hand, suppose $x \beta_2 x$: 

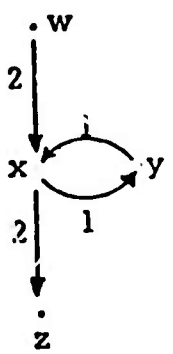
Then, by A10, we get 

, in violation of A9 .

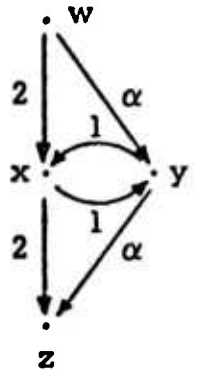
A15. $\forall\, x, y \in Q$: $x \beta_1^2 y$ or $x \beta_2^2 y$ or $x \alpha^2 y \Longrightarrow x \neq y$

Proof: That $x \alpha^2 y \Longrightarrow x \neq y$ follows at once from A9.

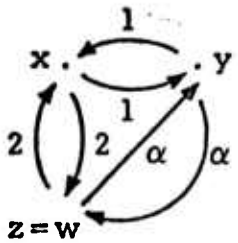
Next we will show that 

cannot exist. If it did, there would also exist a pair of vertices z and w such that 

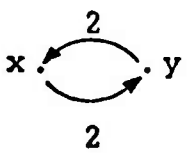
By A10 there must exist an α-arc connecting w to y and an α-arc connecting z to y. One of these arcs must end at y and the other must begin at y, since α is a bijection, e.g., thus:



In any case, w and z are now two presences which share two characters (a value and a slot) and thus, by A7, must be the same presence. Thus we get



which violates A9. Since the only axioms that were used are symmetric in β_1 and β_2, we have also proved the impossibility of



Q.E.D.

A15 and A13 together yield:

A16. In a zebra there are no cycles of length 2.

A17. By A10, and the fact that every α^* equivalence class is represented as an α-circuit in the graph, we see that

.1 $\forall x,y \in Q: \quad x \beta_1 y \Rightarrow (\exists n)(x \beta_2 \alpha^n y)$

.2 $x \beta_2 y \Rightarrow (\exists n)(x \beta_1 \alpha^n y)$

This result, A15, A6 and A7 allow us to assemble the following facts about the three bijections of a presence net.

.3 For each of the three bijections $\gamma: \gamma \cap \overleftarrow{\gamma} = \emptyset$

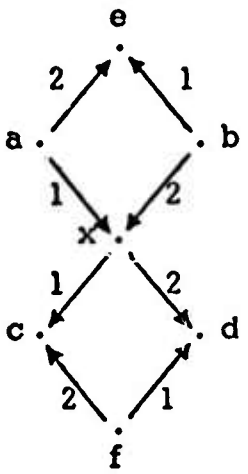
For any two the three bijections $\gamma, \gamma', \gamma \neq \gamma'$:

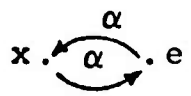
.4 $|Q/(\gamma \cup \gamma')^*| = 1$

.5 $|Q/(\gamma^* \cap \gamma'^*)| = |Q|$

A18. What we have shown so far is sufficient to demonstrate that, given any presence in a presence net, there must exist at least 6 other presences all distinct from one another, and distinct from x such that:

.1

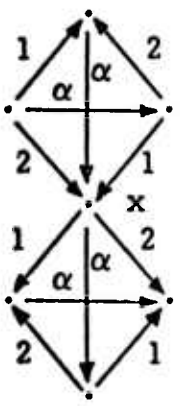


(If e = f then we would get  in violation of A9.)

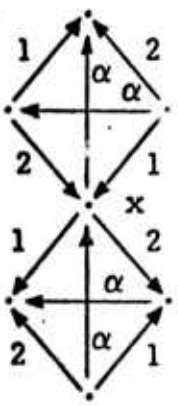
We also know that the pairs $\langle e,x\rangle$, $\langle x,f\rangle$, $\langle a,b\rangle$, $\langle c,d\rangle$ belong to $\alpha \cup \tilde{\alpha}$

We will now show all the possible arrangements of arcs labeled α in graph A18.1 consistent with the axioms:

.2

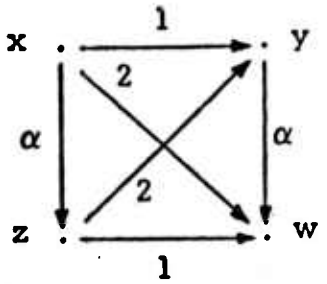


.3



A19. We call a set of 4 vertices $\{x, y, z, w\}$ a junction if they can be ordered $\langle x, y, z, w \rangle$ so that:

.1

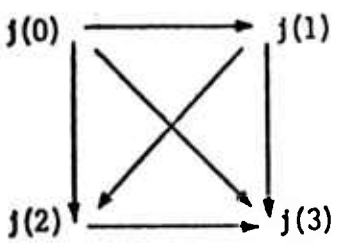


For any two junctions j_1 , j_2 , $j_1 \neq j_2$:

.2 $|j_1 \cap j_2| \leq 1$

.3 If $\{x, y, z, w\} = j$ is a junction, then there is one and only one ordered quadruple of these four vertices which satisfies .1 .

A20. A18.3 gives us a natural way to classify the vertices of a junction j , namely, by the number of arrows entering the vertex:



A21. Relative to a zebra Z, we define

.1 $J \triangleq \{j : j \text{ is a junction of } Z\}$

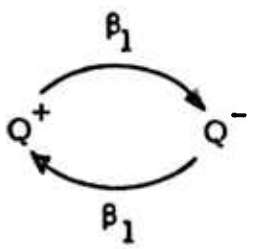
Every presence of a zebra belongs to exactly two junctions. From A18.2 and .3 we note that the presences Q are partitioned into two classes, as follows:

.2 $Q^+ \triangleq \{x : (\exists j_1, j_2)\ (x = j_1(3) = j_2(0))\}$

.3 $Q^- \triangleq \{x : (\exists j_1, j_2)\ (x = j_1(2) = j_2(1))\}$.

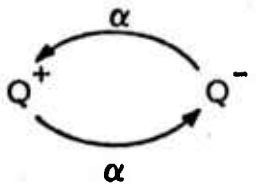
We will call Q^+ the set of <u>positive presences</u> and Q^- the set of <u>negative presences</u> of the zebra.

A22. A18.2 and A18.3 allow us to determine how β_1, β_2 and α map the blocks Q^+ and Q^-.

.1 

.2 

.3



Since we know that Q is strongly connected by any two of the three bijections (A17.4), we can assert that:

.4 $Q/(\beta_1 \cup \alpha)^{2*} = \{Q^+, Q^-\}$

A22.2 means additionally that:

.5 $Q/[(\beta_1 \cup \alpha)^2 \cup \beta_2]^* = \{Q^+, Q^-\}$

A23. A22.2 is the reason why zebras are called zebras. The β_2-circuits are the "stripes" of the zebra. Since the presences of a stripe constitute a value, we see that the values of a zebra are divided into two types: <u>positive values</u>, consisting of positive presences only, and <u>negative</u> values, consisting of negative presences only. Denoting the set of positive values by V^+ and negative values by V^-, we have $V = V^+ \cup V^-$.

.1 $\forall\, j \in J: \quad \exists\, v_1 \in V^+$ <u>and</u> $v_2 \in V^- : j \subset v_1 \cup v_2$

Thus we can say: <u>in every junction a positive and a negative value meet.</u>

A24. Definition

A double zebra $\mathbb{Z}$ is an ordered quintuple $\langle Q, \beta_1, \beta_2, \alpha_1, \alpha_2 \rangle$ such that both $\mathbb{Z}_1 = \langle Q, \beta_1, \beta_2, \alpha_1 \rangle$ and $\mathbb{Z}_2 = \langle Q, \beta_2, \beta_1, \alpha_2 \rangle$ are zebras.

A25. In regard to the character sets M_1, V_1 and Σ_1 of $\mathbb{Z}_1$ and M_2, V_2, Σ_2 of $\mathbb{Z}_2$, one sees at once that:

$$M_1 = V_2$$

$$V_1 = M_2$$

$$\Sigma_1 = \Sigma_2$$

and for the junctions J_1 of $\mathbb{Z}_1$ and J_2 of $\mathbb{Z}_2$:

$$J_1 = J_2$$

A26. Definitions

.1 We define the modes, values, slots, and junctions of a double zebra $\mathbb{Z}$ to coincide with those of $\mathbb{Z}_1$. We will also name the presences of a junction in $\mathbb{Z}$ $j(0)$, $j(1)$, $j(2)$ and $j(3)$ as in $\mathbb{Z}_1$.

.2 We define the positive presences of $\mathbb{Z}$ to coincide with Q_1^+, the positive presences of $\mathbb{Z}_1$, and likewise for negative presences.

.3 We define the active presences of $\mathbb{Z}$ to be Q_2^+, the positive presences of $\mathbb{Z}_2$, and the passive presences of $\mathbb{Z}$ to be Q_2^-, the negative presences of $\mathbb{Z}_2$.

.4 We designate the set of positive values by V^+

the set of negative values by V^-

the set of active modes by M^+

the set of passive modes by M^-

A27. Definition

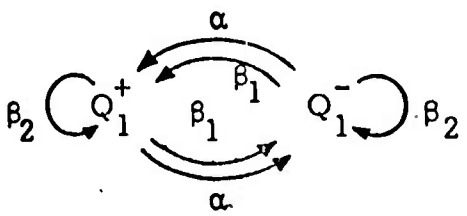
Given a double zebra $\mathbb{Z}$ we define:

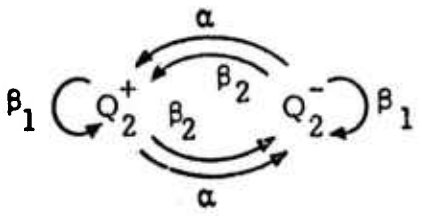
$$Q_3^+ \triangleq (Q_1^+ \cap Q_2^+) \cup (Q_1^- \cap Q_2^-)$$

$$Q_3^- \triangleq (Q_1^+ \cap Q_2^-) \cup (Q_1^- \cap Q_2^+)$$

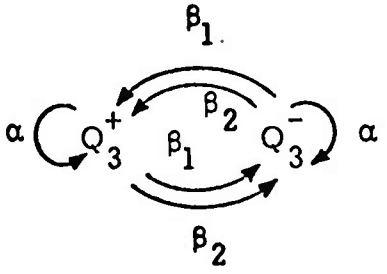
We will call the presences of Q_3^+ actions and the presences of Q_3^- sensings.

A28. Given a double zebra, and letting α stand for either α_1 or for α_2, we already know from A22 that:

.1 

.2 

We now can easily verify that

.3 

A29. Definition

Just as A28.1 and .2 give us grounds for partitioning modes and values into two kinds, so A28.3 gives us grounds for subdividing slots into two kinds:

.1 $\Sigma^+ \triangleq \{\sigma \in \Sigma: \sigma \subset Q_3^+\}$

.2 $\Sigma^- \triangleq \{\sigma \in \Sigma: \sigma \subset Q_3^-\}$

We will call the elements of Σ^+ phases, and the elements of Σ^- locations. Applying A27, we see that phases are sets of actions, while locations are sets of sensings.

Thus, we see that every double zebra is actually a triple zebra: striped by its values, modes, and slots.

A30. We call M, V, and Σ the three character types of a double zebra. For each of these three, there is a positive and negative subtype.

We can now strengthen A23.1, relative to double zebras.

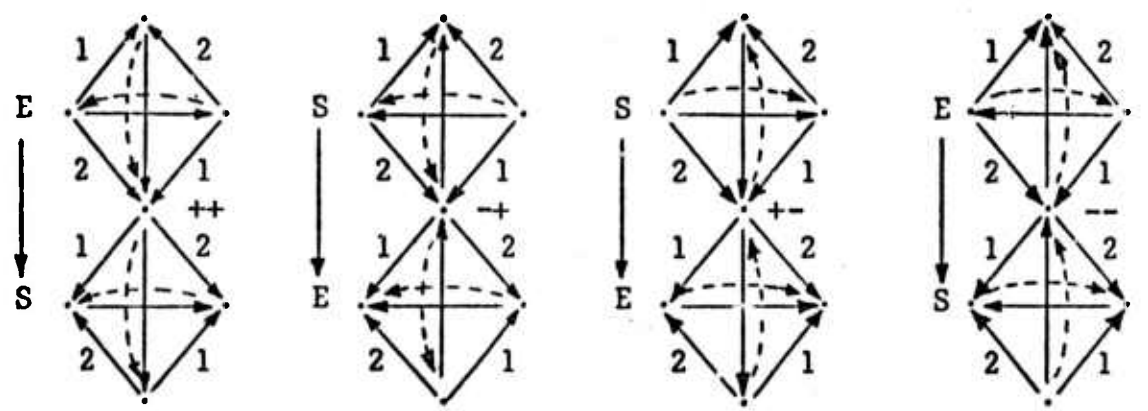
.1 For any two distinct character types, C_1 and C_2:

$$\forall (j \in J): \exists\ (a \in C_1^+) \text{ and } (b \in C_1^-) \text{ and } (c \in C_2^+) \text{ and } (d \in C_2^-):$$
$$j = (a \cup b) \cap (c \cup d)$$

A31. The presences of a double zebra fall into four classes - $Q_1^+ \cap Q_2^+$, $Q_1^+ \cap Q_2^-$, $Q_1^- \cap Q_2^+$, $Q_1^- \cap Q_2^-$. These four classes arise from the relations of bijections to one another. Since every vertex is the intersection of two junctions, we would expect, for a double zebra, to find four diagrams like the two diagrams A18.2 and .3 .

Here they are:

.1 .2 .3 .4



The unlabeled solid arcs are α_1-arcs; the dotted, unlabeled arcs are α_2-arcs. The pair of signs associated with each diagram tell the classification of the vertex shared by the two junctions - e.g., '−+' means the vertex belongs to Q_1^- and Q_2^+ .

A32. Definitions

As already implied in A28, all four presence types are represented in each junction. We are now prepared to notice that, in a

double zebra, there are two distinguishable types of junctions:

.1 $S \triangleq \{j : \{j(0), j(1)\} \subset Q_3^+ \text{ and } \{j(2), j(3)\} \subset Q_3^-\}$

.2 $E \triangleq \{j : \{j(0), j(1)\} \subset Q_3^- \text{ and } \{j(2), j(3)\} \subset Q_3^+\}$

Junctions belonging to S we will call states, and junctions belonging to E we will call events.

A33. One now observes that in each of the four diagrams A31.1 - .4, one of the two junctions is a state and the other is an event. From this follows for all double zebras:

.1 $\forall (x \in Q) : \exists (j_1 \in E) \text{ and } (j_2 \in S) : x = j_1 \cap j_2$

.2 $\forall (j_1, j_2 \in J) : j_1 \cap j_2 \neq \emptyset \implies (j_1 \in S \iff j_2 \in E)$

A34. Theorem

Given a zebra $Z = \langle Q, \beta_1, \beta_2, \alpha_1 \rangle$, there exists a double zebra $\mathbf{Z} = \langle Q, \beta_1, \beta_2, \alpha_1, \alpha_2 \rangle$ if and only if, in Z, the length of every β-circuit is divisible by 2.

Proof:

We will first show that divisibility by 2 implies the existence of an α_2 as required by a double zebra.

The assumption of divisibility by 2 together with A6 guarantees that $(\beta^2)^*$ is an equivalence relation and that $|^Q/(\beta^2)^*| = 2$. Whether two presences belong to the same or different equivalence classes depends only on whether there exists a path of even length between them, or not.

Now, by axiom A10, it is immediate that if $x \alpha y$ then there exists a β-path of even length between them, and thus we know that:

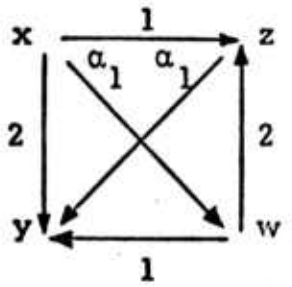
.1 $\alpha_1 \subset (\beta^2)^*$

Now let us call the two blocks of $^Q/(\beta^2)^*$ A and B. We define

.2 $\alpha_2 \triangleq \alpha_1 \restriction A \cup \overleftarrow{\alpha}_1 \restriction B$ (where '$\restriction$' means 'restricted to')

Having shown .1 we know that α_2 is a bijection. Now consider an arbitrary junction:

.3



Clearly $\{z, y\} \subset A \Longleftrightarrow \{x, w\} \subset B$

which means either we have $z\,\alpha_2\,y$ and $w\,\alpha_2\,x$ or $y\,\alpha_2\,z$ and $x\,\alpha_2\,w$ (but not both).

Thus, we have shown that α_2 has the required property.

That the existence of a bijection α_2 implies divisibility by 2 of the length of every β-circuit follows from A28.3. Q_3^+ and Q_3^- must be the equivalence classes of $(\beta^2)^*$. Q.E.D.

A35. What need be known about the characters, states and events of a double zebra in order to determine it uniquely? The following is a sufficient (though not necessary) condition.

If two double zebras have the same states, the same events, and for two distinct character types C_1 and C_2, the same C_1^+, C_1^-, C_2^+, and C_2^- then they are identical.

B. Double Zebra Transformations

B1. Definitions

Given a double zebra $\mathbb{Z} = \langle Q, \beta_1, \beta_2, \alpha_1, \alpha_2 \rangle$ we define:

.1 $\rho_1(\mathbb{Z}) = \langle Q, \beta_1, \beta_2, \overleftarrow{\alpha}_1, \alpha_2 \rangle$

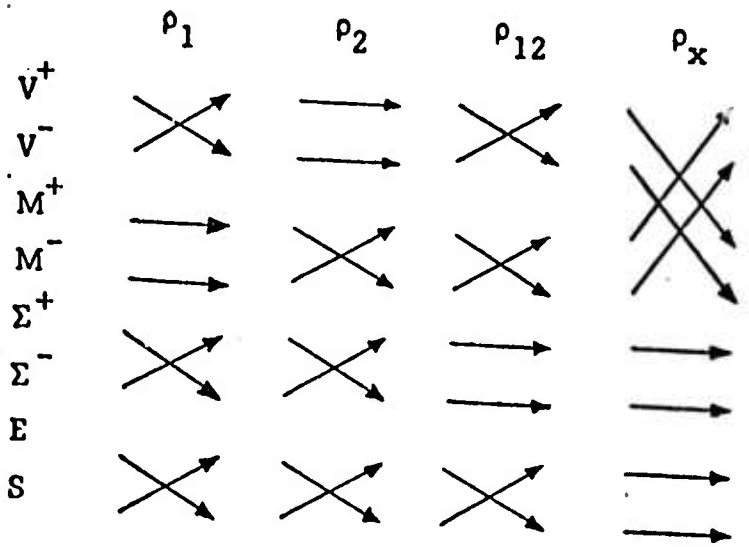
.2 $\rho_2(\mathbb{Z}) = \langle Q, \beta_1, \beta_2, \alpha_1, \overleftarrow{\alpha}_2 \rangle$

.3 $\rho_{12}(\mathbb{Z}) = \langle Q, \overleftarrow{\beta}_1, \overleftarrow{\beta}_2, \alpha_1, \alpha_2 \rangle$

.4 $\rho_x(\mathbb{Z}) = \langle Q, \beta_2, \beta_1, \alpha_2, \alpha_1 \rangle$

The reader can readily verify that all of these transformations yield double zebras and that their effect can be described as the interchange of character types or subtypes (and, therefore, of states and events) according to the following table:

B2.



The transformations exhibited in B2 have the property that they leave the set of characters C and the set of junctions J invariant. We will now consider the structure of the full group of double zebra transformations with this property.

A junction is naturally thought of as a tetrahedron - the vertices being the presences of the junction and the edge the characters which intersect it. Since we have characters of three types - modes, values, and slots - we can color the edges in three colors, edges that are opposite each other being colored alike. Since each character type is subdivided into negative and positive, there may also be assigned to each edge a sign. As we know from A27, only two of the colors can be signed independently, the disposition of signs on the third then being constrained to lie in one of the two possible ways. Here is an example:

B3.

In freely distributing signs over the edges of a tetrahedron so that opposite edges carry opposite signs, there are two possible distinguishable results: a tetrahedron which contains a face that is all negative (and a point

that is all positive) or a tetrahedron with a face that is all positive (and a point that is all negative). The definition forces all tetrahedra which represent junctions to have all negative faces, as in B3.

The tetrahedra, as far as described, have no features corresponding to the partition of junctions into states and events. This we may add by imagining the tetrahedra which represent junctions to be signed - say, positive for events and negative for states. Natural transformations of double zebras are those which can be expressed in terms of sign changes and color permutations in the model we just constructed. We will first examine the possible transformations with sign-change alone.

It is easy to see that these are spanned by ρ_1, ρ_2, and ρ_{12}, as defined in B1. When restricting one's attention to their effect on the signs on the edges of the tetrahedra, one sees that they generate all changes defined thus: negate all signs except those on one pair of opposing edges. No other transformation on the edge signs of the tetrahedron - with the exception of the identity transformation - are consistent with A27.

The generators ρ_1, ρ_2, and ρ_{12} generate a group of 8 elements. It is the abelian group determined by three generators, a, b, c and the relations $a^2 = b^2 = c^2 = 1$, the group of reflections on a 3-dimensional cube. A natural representation of the eight group elements in terms of

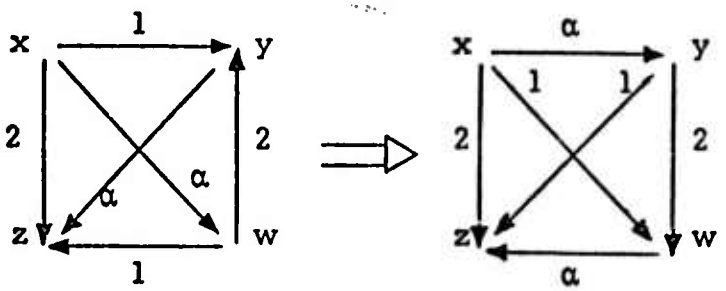
sign changes on the edge-pairs and the tetrahedron as a whole is this:

B4.

	ρ_1	ρ_2	ρ_{12}					
M	0	1	1	0	0	1	1	0
V	1	0	1	0	0	1	0	1
Σ	1	1	0	0	0	0	1	1
J	1	1	1	1	0	0	0	0

(where '1' sign interchange, and '0' means no sign change)

It remains to consider edge-color permutations, an example of which is ρ_x defined in B1. It is easy to see that if, in every junction of a zebra Z the β_2-circuits corresponding to negative values are reversed, then a new zebra Z' results, related to Z by color permutation.



namely (as B5 shows) the interchange of M and Σ. Furthermore, if Z was $\mathbb{Z}_1$ of a double zebra $\mathbb{Z}$, then there exists a double zebra $\mathbb{Z}'$ of which Z' is $\mathbb{Z}'_1$. The existence of an appropriate α'_2 depends only on whether all β'-circuits are of even length - i.e., that all circuits composed of α-arcs and β_2-arcs in $\mathbb{Z}$ are of even length, a fact which

is guaranteed by A34. In the light of B4, there must, therefore, exist a double zebra obtained from $\mathbb{Z}$ by the interchange of M and Σ.

Since we now see that double zebras can be transformed by either of two color interchanges of its three colors, M, V, Σ, all color permutations must be possible - i.e., the full group S_3 can be applied. Thus the entire group of double zebra transformations is the product group of S_3 and the group of cube reflections discussed above. The product group has 48 members and, thus, every double zebra belongs to a set of 48 distinct double zebras with the same set of junctions and the same set of characters.

C. An Approach to Zebra Redefinition

Let us reconsider the question posed in A35. Preparatory to this, we shall introduce the following definitions:

C1. .1 $Q_V \triangleq \{Q_1^+ , Q_1^-\}$

.2 $Q_M \triangleq \{Q_2^+ , Q_2^-\}$

.3 $Q_\Sigma \triangleq \{Q_3^+ , Q_3^-\}$

where the sets Q_i^+ , Q_i^- mean the same as in A28. Given these definitions, we will rename these latter sets Q_x^+ , Q_x^- where x stands for V, M, or Σ, depending upon which of the three partitions into two sets is meant.

Now, given a double zebra, the following list of partitions of Q is also given:

C2. .1 E - the set of events
.2 S - the set of states
} (the union of these two sets is the set of junctions)

.3 M - the set of modes

.4 V - the set of values

.5 Σ - the set of slots

.6 Q_M - the "striping" of the modes

.7 Q_V - the "striping" of the values

.8 Q_Σ - the "striping" of the slots

How many of these partitions are necessary to determine the double zebra uniquely?

C3. It is sufficient to know E, S and any two of the three stripings.

Proof:

Since all of these partitions result from the bijections β_1, β_2, α_1 and α_2 it is only necessary to check that these bijections are determined by four partitions as specified in C3.

Given A27, we know in any case that any two of the three stripings determine the third. Thus, we need only verify that given a particular pair of stripings and E and S, the bijections are uniquely determined.

Since all the forward images and backward images of a presence under the four bijections are contained in the two junctions to which the presence belongs, we need only verify that we can reconstruct, given a junction, all the ordered pairs within the junction which belong to the four bijections.

Suppose, then, that we are given an element e of E - i.e., a set of four presences $e = \{ x, y, z, w \}$, and we are also given the two stripings Q_V and Q_M. $Q_V \cap Q_M$ is a partition with four blocks and e contains one presence in each of the four blocks (A30). Without loss of generality, we may suppose:

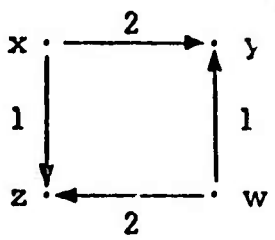
$$x \in Q_V^+ \cap Q_M^+$$

$$y \in Q_V^+ \cap Q_M^-$$

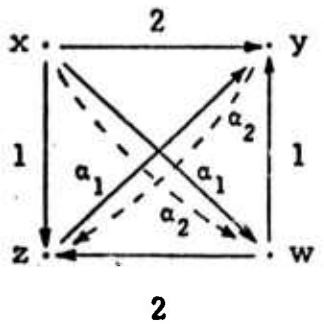
$$z \in Q_V^- \cap Q_M^+$$

$$w \in Q_V^- \cap Q_M^-$$

Since Q_V stripes the values, we know that either $\langle x, y \rangle \in \beta_2$ or $\langle y, x \rangle \in \beta_2$, and similarly $\langle z, w \rangle \in \beta_2$ or $\langle w, z \rangle \in \beta_2$. Since Q_M stripes the modes, we also know: either $\langle x, z \rangle \in \beta_1$ or $\langle z, x \rangle \in \beta_1$ and either $\langle y, w \rangle \in \beta_1$ or $\langle w, y \rangle \in \beta_1$. Applying A27 and A32.2, we discover that the following diagram is necessary:



Furthermore, applying A21.2 - .3, we can fill in:



Q.E.D.

Given this result, we can consider defining double zebras in terms of a set Q , and four partitions, E, S, and two stripings, A and B . The force of the axioms A6 - A12 must then be translated into axiomatically required relations among partitions. What follows here is not a working out of this program, but some observations preliminary to its fulfillment.

C4. E and S are partitions of Q with all blocks consisting of four elements. $A \cap B$ is a partition of Q into four blocks in such a way as to distinguish the four points of every junction from one another. Letting **0** represent the partition of Q consisting entirely of one-element blocks we can express these relations thus:

$$E \cap A \cap B = S \cap A \cap B = \mathbf{0}$$

C5. Letting $\mathbf{1}$ represent the partition of Q which has one block only we can also be confident of the following properties:

.1 $E \cap S = \mathbf{0}$

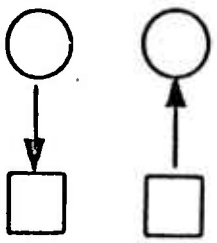
.2 $E \cup S = \mathbf{1}$

The first of these properties says that a state and an event have at most one presence in common. The second of these properties is necessary if axiom A6 is to hold for double zebras.

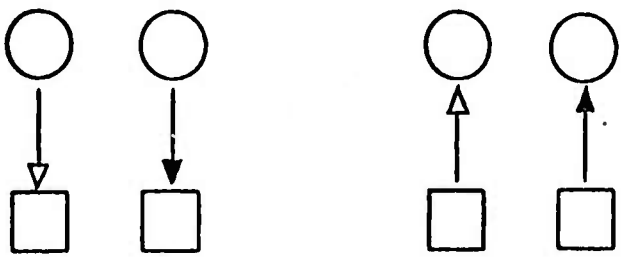
D. Double Zebras and Petri Nets

Double zebras may be represented as a class of Petri Nets with four kinds of connections between states and events in place of the customary two.

D1. .1 In ordinary Petri Nets:



.2 In Nets representing double zebras:



Nets with two-colored arrow-heads, as in D1.2 have been called Tolly-nets. We will now show by what conventions Tolly-nets may be understood as representations of double zebras.

From C3 we know that a double zebra is fully characterized by the four partitions E, S, Q_Σ, Q_V. We now proceed as follows.

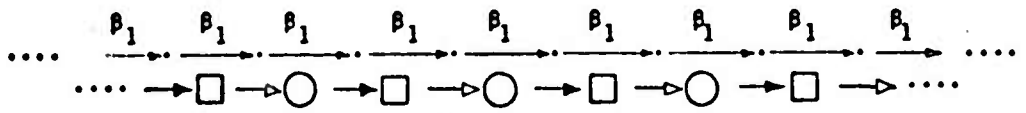
D2. .1 Construct an undirected graph whose vertices correspond 1-1 with the junctions of a double zebra. Two junctions j_1 and j_2 are connected by an edge iff $|j_1 \cap j_2| = 1$. If the junction is a state the corresponding vertex is symbolized by ◯ ; if it is an event, it is symbolized by □ . By A19.2 and A33 it is now clear that (1) there is a 1-1 correspondence between the edges of the graph and the set of presences Q and (2) if two vertices are connected by an edge, one of them is ◯ and the other of them is □ .

.2 We will now orient the graph. If an edge corresponds to a presence which belongs to Q_Σ^+ orient it from the state to the event; if it corresponds to a presence which belongs to Q_Σ^-, orient it from the event to the state.

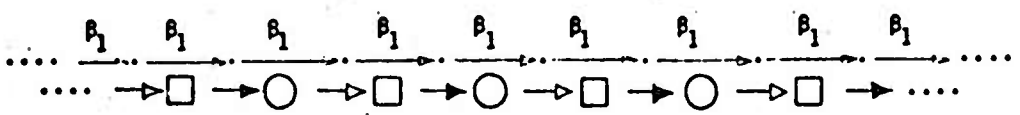
.3 We will now color the arrow heads. If an edge corresponds to a presence belonging to Q_V^+, make the arrow-head black; if it corresponds to a presence belonging to Q_V^-, make the arrow-head white.

We can show how the β_1, β_2, α_1 and α_2 circuits are represented with these conventions.

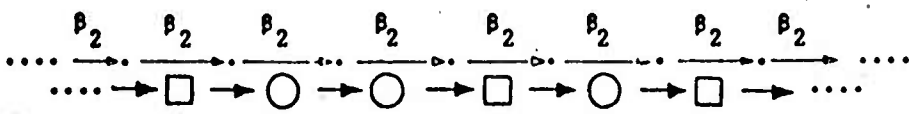
D3. .1 An active mode:



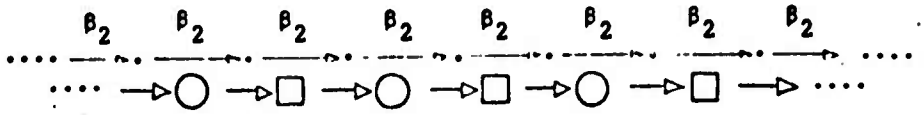
.2 A passive mode:



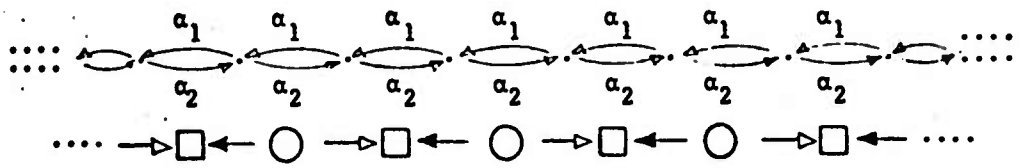
.3 A positive value:



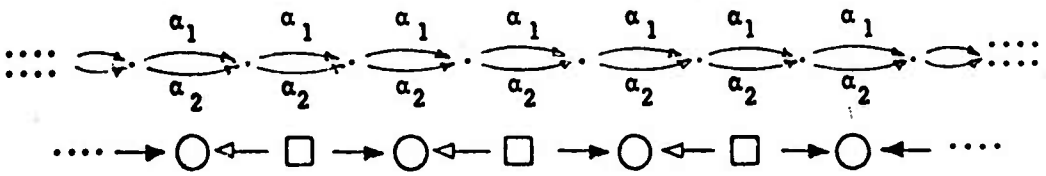
.4 A negative value:



.5 A phase (action slot)



.6 A location (sensing slot)



What are the characteristics of Petri Nets which represent double zebras? We can give a partial answer to this question.

D4. .1 Two-in, two-out: every state and every event has two arcs entering and two arcs exiting.

.2 Net purity: there is at most one arc connecting a given state to a given event.

.3 Strong connectivity

.4 Tolly-net coloring: for every state and every event, one entering arc is black and the other white; one exiting arc is black and the other white.[1]

.5 Two distinct cycles of the forms D3.1 - .6 may share at most one arc. (Axiom A7) Note that, in a Tolly-net one each of the six cycle types pass through each state and

---

[1] It is easy to see that every two-in, two-out Petri Net can be Tolly-colored.

each event - one for each of the six ways that there are of choosing two arcs (without regard to order) out of the four arcs incident on the state or event.

It is possible that any Petri-Net with Tolly-net arrow coloring satisfying D4.1 - .5 corresponds to a double zebra, but no effort has yet been made to check this.

E. Working Notes on State and Event Components

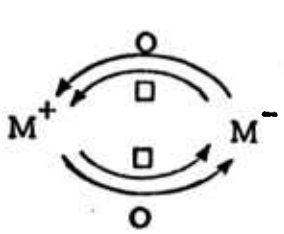
E1. We define the following relations of characters to characters in a double zebra:

.1 $\forall\, c_1, c_2 \in C: \quad c_1 \,\square\, c_2 \triangleq \exists\, e \in E: \quad c_1 \cup c_2 \supset e$

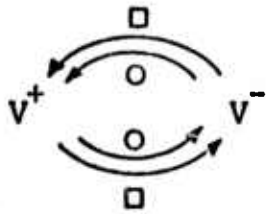
.2 $\forall\, c_1, c_2 \in C: \quad c_1 \,\bigcirc\, c_2 \triangleq \exists\, s \in S: \quad c_1 \cup c_2 \supset s$

E2. From the definition of a junction (A19), we see at once that for any character c and any junction j, either $|c \cap j| = 0$ or $|c \cap j| = 2$; from A30 we see that for any two characters c_1 and c_2 which are of distinct type, if $|c_1 \cap j| = |c_2 \cap j| = 2$ then $|c_1 \cap c_2 \cap j| = 1$. Therefore, if $c_1 \,\square\, c_2$ or $c_1 \,\bigcirc\, c_2$ then c_1 and c_2 must be characters of the same type. By A30 we also know that they must be of opposite subtype. The following three pictures summarize these remarks.

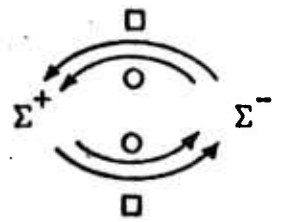
.1



.2



.3



E3. Since $\square$ and $\bigcirc$ are symmetric, it is clear that $\square^*$ and $\bigcirc^*$ are equivalence relations.

.1 We will call the blocks of $C/_{\bigcirc^*}$ the <u>state components</u> of $\mathbb{Z}$.

.2 We will call the blocks of $C/_{\square^*}$ the <u>event components</u> of $\mathbb{Z}$.

We will designate the set of state components $C_{\bigcirc}$ and the event components $C_{\square}$. From D2 we see that $C_{\bigcirc}$ and $C_{\square}$ partition into three blocks according to whether the component consists of modes, values,

or slots. These blocks are naturally named M_o , V_o , Σ_o , $M_\square$, $V_\square$ and $\Sigma_\square$. In some contexts, it will be useful to think of the components as sets of presences - i.e., the union of the characters of the component. Looked at that way, M_o , V_o , Σ_o , $M_\square$, $V_\square$ and $\Sigma_\square$ name six partitions of the presence Q . Given a character c we will write c_o and $c_\square$ to mean the state component and the event component to which c belongs. Thus, if c and c' belong to the same state component we can write $c_o = c'_o$.

Now a property almost certainly to be demanded of double zebras in connection with their principal interpretation is this:

E4. $\forall\, c \in C: \quad c_o \cap c_\square = c$

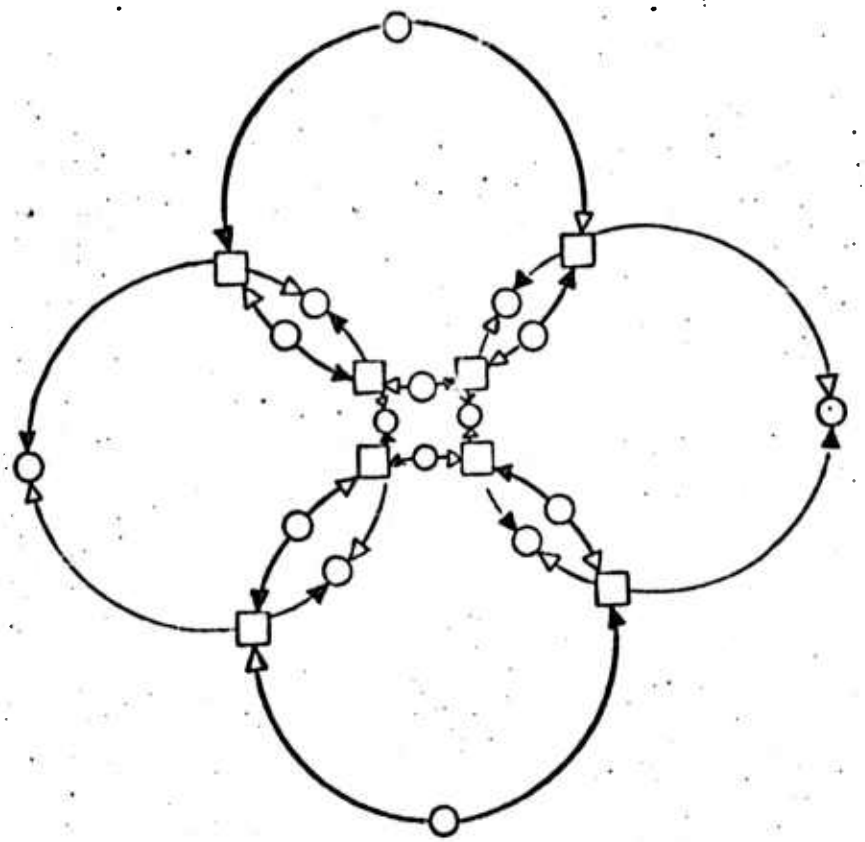
In our past work on Petri Nets, we have had an especial interest in two out of the six component types, namely

.1 M_o - the set of <u>parts</u>

.2 $V_\square$ - the set of <u>effects</u>

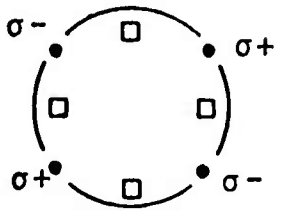
E5. It is interesting to note that the construction of C.A. Petri for the realization of the Quine function, namely:

.1



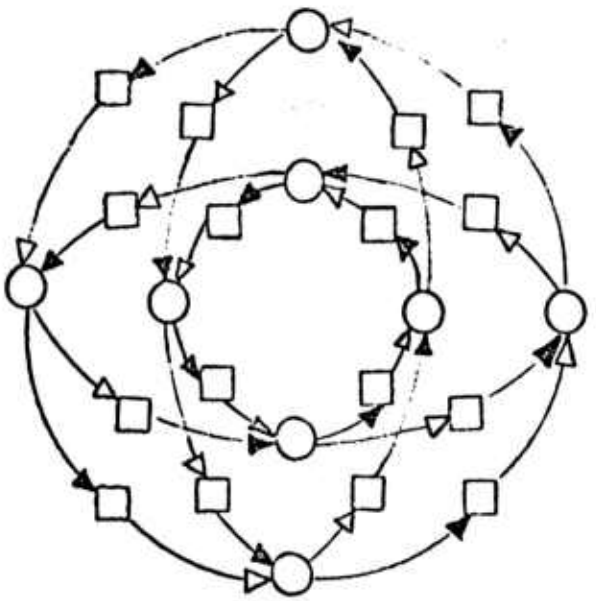
is a component belonging to $\Sigma_{\square}$. The upper and lower rings of figure .1 represent phases while the right and left rings represent locations. The component in .1 is seen to have this structure:

.2



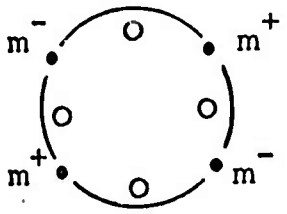
Here is another example of a component:

.3



This one belongs to M_o (i.e., is a part) and also involves four characters arranged as follows:

.4



E5.2 and .4 give rise to wondering about what structure can be found in (or through) the graph whose vertices represent the characters of a double zebra. Taking o and □ as the source of edges, we see the graph in three disconnected parts - M, V and Σ. We see each of these parts covered by an analogue to the junctions in zebras, namely the components. Just as with junctions, the components fall into two classes - state components and event components. These classes constitute a pair of partitions $\mathbb{S}$ and $\mathbb{E}$ of C. Defining $\mathbb{T} = \{M, V, \Sigma\}$ and $\mathbb{O}$ as the partition of C with singleton blocks, we have the properties:

E6. .1 $\mathbb{S} \cup \mathbb{E} = \mathbb{T}$

.2 $\mathbb{S} \cap \mathbb{E} = \mathbb{O}$

Property .1 comes from the fact that the graph of the underlying double-zebra is strongly connected in any two of its three bijections β_1, β_2 and α_1. Property .2 is a re-expression of E4.

The examples E5.1 and .3 - both corresponding to structures which have proved useful in interpreted net constructions, happened to be four-vertex components in analogy to the four-vertex junctions in zebras. Taking the signs on the vertices into account, there is a basis for introducing two more edges to the graph of these components (connecting vertices of like signs) which would make them formally even more similar to zebra junctions.

Let us now consider the relationship between E6.2 and the meanings expressed by Petri Nets which satisfy the requirement. An event represents an interaction between two parts, each in some particular mode. Thus, the event is a junction of two modes, one of these belonging to one part and one to the other. A state represents the possibility, for a part, of switching from one mode to another. Thus, a state is a junction of two modes both of which belong to the same part. All the modes of a single part are connected to one another via state junctions. We wish to guarantee formally that:

E7. Parts do not interact with themselves in events.

That is, if two modes m and m' are connected by O^* - and thus belong to the same part - they should connect via an event. This is a special case of what E6.2 guarantees. If we had both $m\ O^*\ m'$ and $m\ \square\ m'$ we would have both $m_O = m'_O$ and $m_\square = m'_\square$, in short, a mode state component and a mode event component which intersect on more than one mode.

Let us now consider some relationships between phases and locations implied by E6.2. We will use the letter φ in designating phases (action slots) and the letter λ in designating locations (sensing slots). First notice:

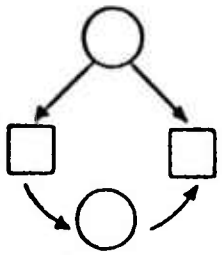
E8. .1 $\forall\ \varphi \in \Sigma^+$ and $\lambda \in \Sigma^-$: $\varphi\ \square\ \lambda \equiv \exists q \in \varphi$ and $q' \in \lambda$: $q\ \beta\ q'$

.2 and $\varphi\ O\ \lambda \equiv \exists q \in \varphi$ and $q' \in \lambda$: $q'\ \beta\ q$

E6.2 certainly excludes  . Here are examples of violations of this exclusion, shown in Petri Net terms.

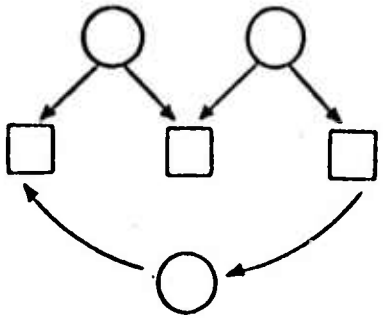
E9.

.1

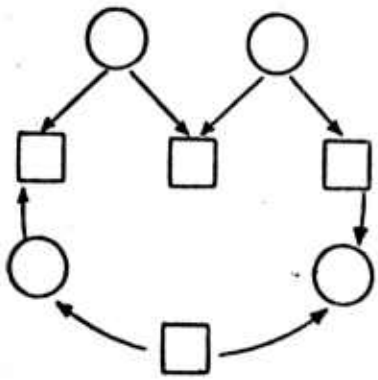


Axiom A7 alone already excludes this figure.

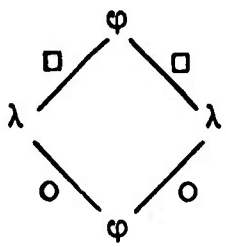
.2



.3



E6.2 also excludes:



- i.e., these are not two distinct locations which lie between the same pair of phases. Also: these are not two distinct phases which lie between the same pair of locations.

IV. The Representative Circuit Theorem For Marked Graphs

The purpose of these notes is to present a theorem of F. Commoner, which addresses the following question:

Consider a specified basic circuit C in a (strongly connected) marked graph G with live marking class $\mathcal{M}$. What is the relation between the marking on that circuit - i.e., the position of the token on C - and the marking on the rest of the graph? Can the marking on C be said to "represent" the marking on the whole graph, in the sense, say, that the marking on C restricts the possible markings on the rest of the graph, or, conversely, that the marking on the rest of the graph determines the marking on C ? When this is the case, we might say that the marking on the distinguished circuit alone determines some definable property of the marking on the rest of the graph - although it may not always be easy to formulate a statement of that property in words.

The satisfaction of this condition for a basic circuit C is interesting from the following interpreted point of view. Assume that the basic circuits of a marked graph represent the cycles of operation of the parts of a system. Now: to assume that a change of state for the part represented by C makes a describable difference to the relations the other parts have to one another - a difference in what events of interaction can occur as between those other parts - is to assume that, relative to C , the condition under discussion is satisfied.

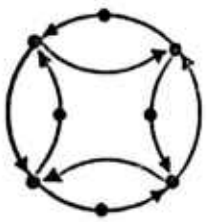
The theorem uses the following notations:

-- If G' is a subgraph of G , then G - G' means the graph G with the arcs of G' removed;

-- If M is a marking of G , then M $\restriction$ G' means the marking restricted to the arcs of G' .

Thus the statement of our question is: for some marking $M \in \mathcal{M}$, does $M \restriction (G - C)$ determine $M \restriction C$?

Take as an example the familiar four-billiard-ball system with the initial marking, as shown, and take the circuit indicated by 'C' as the distinguished basic circuit.



It will be seen by experiment that, in all the markings reachable from this initial marking, when the outer arc of C bears the token, only the markings ⌣ , ⌣ , ⌣ are possible, and when the inner arc of C bears the token, only the markings ⌣ , ⌣ , ⌣ are possible (where, for shorthand, we describe a marking by showing only the marked arcs in the graph). In this case, clearly, the markings of the marking class for the remainder of the marked graph are partitioned into two subclasses, one corresponding to each possible marking of the "representative" basic circuit. It is equivalent to say that, given the marking on the remainder of the graph, the marking on the distinguished basic circuit is determined (viz., to be the one corresponding to the equivalence class in which the given remainder-marking lies).

Commoner's theorem formulates this "representative" property as follows, for a strongly-connected graph with a live marking class and a given basic circuit C:

1. If any two markings in the marking class agree on all arcs of the graph not including C , then they agree on the whole graph.

That this formulation reflects the desired property can be clarified by considering the contrapositive form of the implication in (1): if two markings (of the live marking class) are different, then that difference cannot reside entirely on the distinguished basic circuit -- they must differ on the remainder of the graph, as well. Of course, the difference may lie entirely on the remainder sub-graph -- then the two markings belong to the same equivalence class corresponding to a marking on the "representative" basic circuit.

Commoner finds three other conditions on the graph which are equivalent to (1). An interesting point is that the other three conditions depend only on the connectivity of the underlying undirected graph, and not on the orientation, or the marking, of the arcs. The simplest of the three conditions is:

2. No pair of arcs on C constitute a cut set of the graph - i.e., the graph cannot be disconnected by removing any two arcs of the distinguished basic circuit.

Theorem: Given a strongly connected graph G with live marking class $\mathcal{M}$ and circuit C, basic with respect to $\mathcal{M}$, the following are equivalent:

1. For $M_1, M_2 \in \mathcal{M}$ $[M_1 \restriction (G-C) = M_2 \restriction (G-C)] \Rightarrow M_1 = M_2$

2. For any two arcs $\alpha \neq \beta$ on C, $G - \{\alpha, \beta\}$ is connected

3. For any two arcs $\alpha \neq \beta$ on C, there is a cycle D containing α but not β

4. For any two arcs $\alpha \neq \beta$ on C, there is a path P, touching C only at its end points, whose endpoints separate α and β on C

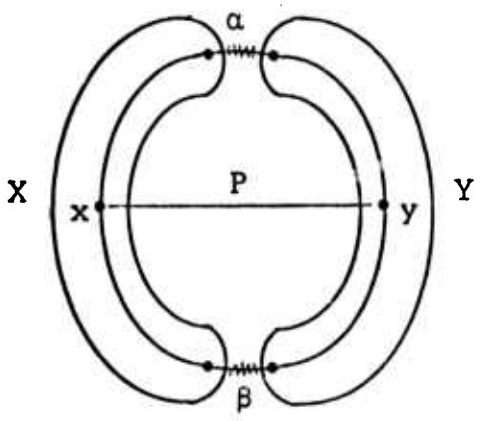
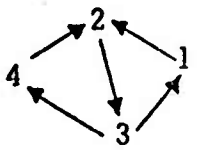


Figure 1

The equivalence of the four conditions is proved in the following pattern:



Proof:

(4) ⇒ (2). This is immediate: the path P, assumed in (4) serves to keep the graph connected even when α and β are excised.

(2) ⇒ (3). This is also immediate: remove α and β from the graph. Since it is still connected, by hypothesis, there is some path in the reduced graph from one (former) endpoint of α to the other endpoint of α; this path does not contain β, which is no longer in the reduced graph. Restoring α closes the path, producing a cycle containing α and not containing β.

(3) ⇒ (4). If (3) is true, let Q be a path from one endpoint of α to the other such that Q does not go over α or β. (Q can be the cycle D postulated in (3), with the arc α removed.) Then Q must contain an arc not on C, and Q must go from the region X to the region Y, shown in Figure 1 (the disjoint regions into which a circuit is divided by excising two of its arcs). If x is the last vertex on Q which lies in X, and y is the first vertex on Q which lies in Y and follows x on Q, then clearly the segment between x and y, $P = Q(x,y)$ is a path which touches C only at its endpoints. Thus (3) ⇒ (4).

(3) ⇒ (1). Let M_1 , $M_2 \in \mathcal{M}$, suppose $M_1 \lceil (G - C) = M_2 \lceil (G - C)$. C is a basic circuit, and a basic circuit contains exactly one token, let α be the arc of C which contains the token in M_1 , and β be the arc of C which contains the token in M_2 . We shall derive a contradiction by supposing $\alpha \neq \beta$, thus proving that if the markings agree on the remainder of G , they must agree on C . Suppose $\alpha \neq \beta$. Let D be the cycle postulated in (3). Now, from Theorem E13 of [Events and Conditions, p. 108], any two markings of a marking class place the same number of tokens on any cycle, so $N = M_1 - M_2$ must have a cycle loading of zero on every cycle, and specifically, $N(D) = 0$. Now for γ any arc in G ,

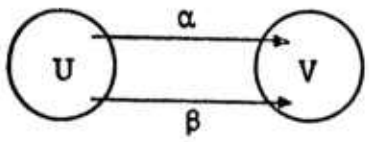
$N(\gamma) = 1$ if $\gamma = \alpha$ (Since α is marked by M_1 and not by M_2)

$N(\gamma) = -1$ if $\gamma = \beta$ (Since β is marked by M_2 and not by M_1)

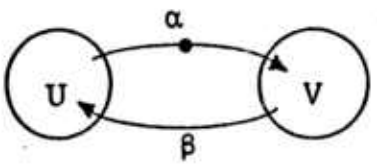
$N(\gamma) = 0$ otherwise (Since, by hypothesis, M_1 and M_2 agree on all arcs of G other than α and β .)

But since $\alpha \in D$ and $\beta \notin D$, $N(D) = \pm 1$, contradicting $N(D) = 0$.

(1) ⇒ (2). Let us suppose (2) is false; i.e., suppose there are two arcs $\alpha \neq \beta$ on C such that $G - \{\alpha, \beta\}$ is disconnected. Since $G - \alpha$ and $G - \beta$ are connected, $\{\alpha, \beta\}$ is a minimal disconnecting set, so $G - \{\alpha, \beta\}$ has exactly two components, U and V .

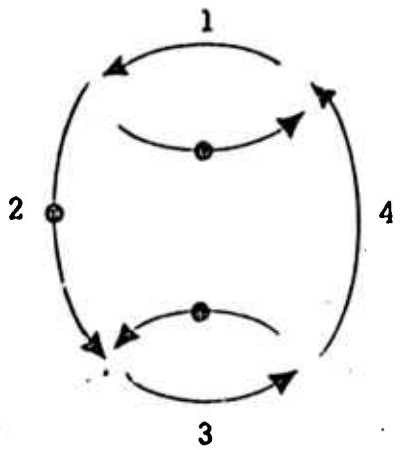


The arcs α , β cannot both go from U to V or from V to U since G is strongly connected. Thus they must go in opposite directions, and we shall say that α goes from U to V while β goes from V to U .



Since $\mathcal{M}$ is a live marking class, there is some marking $M_1 \in \mathcal{M}$ which places a token on α , i.e., such that $M_1(\alpha) = 1$. Let 1_γ be the characteristic function of $\{\gamma\}$, for any arc γ . Since a simple circuit C goes through α if and only if it goes through β, $1_\alpha(C) = 1_\beta(C)$ for any simple circuit C. Thus $M_1 - 1_\alpha + 1_\beta$ is circuit-equivalent to M_1 (has the same value as M_1 on every circuit) since $1_\alpha - 1_\beta$ is circuit-equivalent to zero. But $M_2 = M_1 - 1_\alpha + 1_\beta$ is just the marking M_1 with the token on α moved to β . Since M_2 is circuit-equivalent to M_1 , $M_2 \in \mathcal{M}$ by the circuit test. But $M_2 \restriction (G - C) = M_1 \restriction (G - C)$, contradicting (1).

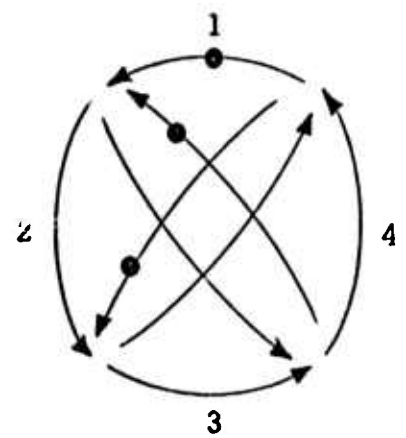
Examples:



C = (1234)

(1) is false, since the same marking with the token on 2 moved to 4 is reachable from this marking, and does not differ on the remainder of the graph.

Note that (2) fails, since {2,4} disconnect the graph.

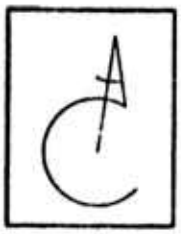


C = (1234)

(1) is true, since (2) holds.

Remark: Note that conditions (2), (3), and (4) depend only on the underlying undirected graph of G. In other words, given a connected (undirected) graph G and a distinguished set of arcs C which form a cycle in G, and such that one of the properties (2), (3), or (4) holds, then if arc-directions can be assigned so that G is strongly connected and C is a (directed) circuit, and if a marking is found which is live on G and under which C is a basic circuit, property (1) will also hold for the marked graph.

VOLUME II



MASSACHUSETTS
COMPUTER ASSOCIATES, INC.
26 PRINCESS STREET, WAKEFIELD, MASS. 01880 • 617/245-9540

Role/Activity Models and Petri Nets

by

Anatol W. Holt

March 14, 1975

A SUBSIDIARY OF APPLIED DATA RESEARCH, INC.

TABLE OF CONTENTS

INTRODUCTION

The Information Systems Theory Project has been developing a foundation - consisting of concepts and allied notations - for the specification and analysis of systems. It has been the hypothesis of this Project that Petri-nets can provide a skeleton strong enough to support a large body of system specification and analysis needs - large because of the very many different practical objectives which such specifications and analyses may serve. Designers, operators, managers, users, implementers, lawyers, accountants, etc., all have their separate needs to understand systems, yet must be able to communicate with one another wherever their practical concerns overlap.

It is generally the task of mathematical models to provide skeletons on which to hang a great diversity of applications and Petri nets may be viewed as encompassing a class of mathematical models. The distinguishing characteristic of this class is that its basic sentences are of the form "this follows from that" ("follows" in the sense of causality) rather than of the form "this equals that". Related to the last mentioned characteristic is the fact that Petri net models are all capable, in principle, of being used to simulate the behavior they represent. This is not to say, however, that all analytic procedures they enable depend upon simulation. Previously published theorems and algorithms as well as Chapters VI and VII of this report make this plain.

What considerations give rise to various species within the genus of Petri net models?

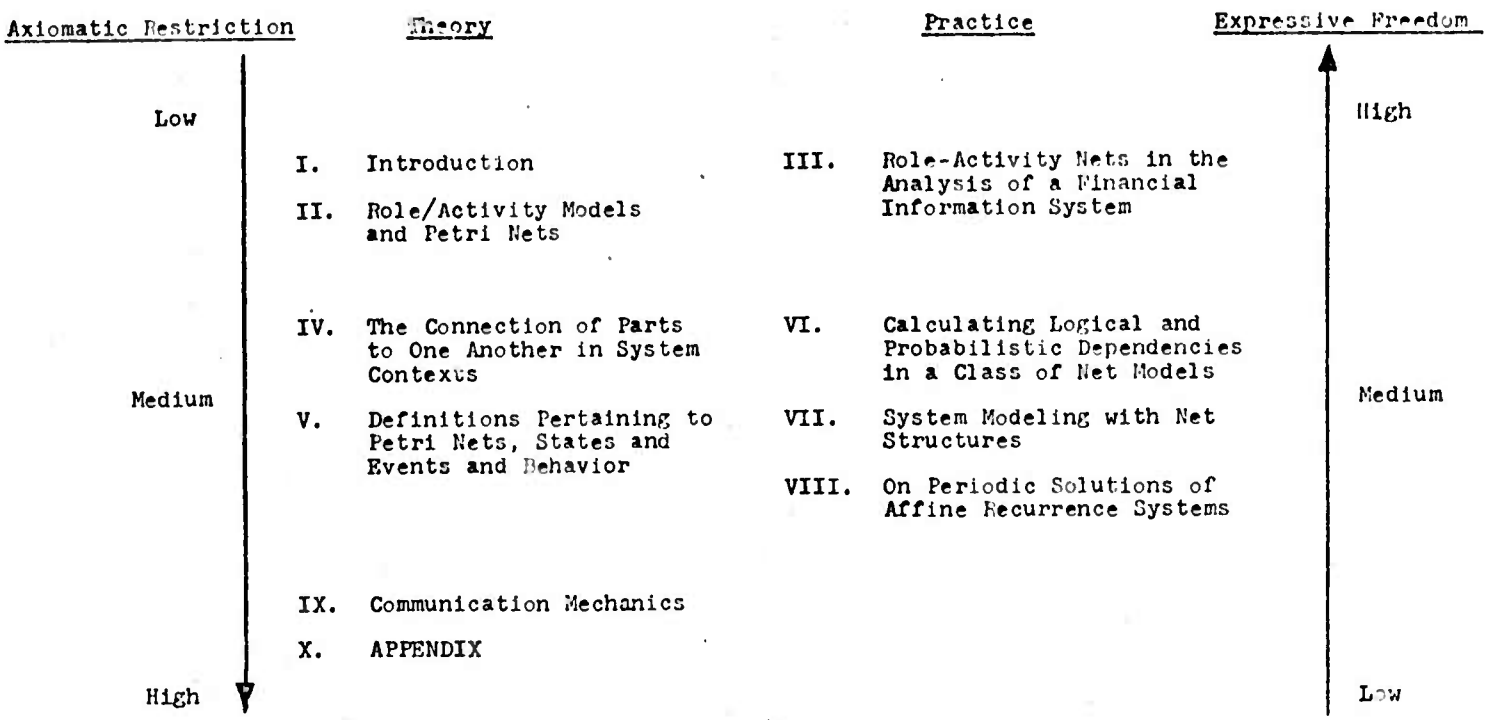
One of the fundamental needs in systems work is to relate system objectives to the facts of implementation. We believe that a significant part of the difficulty in doing this stems from the misdirected tendency to think that statements of objectives are different in kind from statements of how a set of objectives are to be met. Broad statements of system objectives must refer to the intended relations of behavior between human agencies and physical artifacts in the large, while specifications of implementation will refer to such relations of behavior in the small. Different classes of net models are applicable at various levels of the micro-macro scale. The smaller the parts, the more constrained their behavior by physical laws, the greater the number of axiomatic restrictions prevailing in the applicable class of net models. Broadly speaking, the force of physical law as it pertains to systems work - i.e., what is and is not physically implementable at some level of analysis - is to be expressed in the force of axiomatic restrictions definitive of some class of net models.

A second reason for differentiating classes of nets has to do with calculation needs. Different sets of axiomatic restrictions will lead to different natural possibilities for transforming nets, or for implementing algorithms upon them to count this or that. Increased axiomatic restriction will tend to be accompanied by increased calculational power. Thus implementation oriented nets will more promote the needs of calculation while nets for the expression of system purposes will more serve the needs of creating common understanding between different members of a community, all of whom must somehow deal with a system.

To cast a given set of system meanings into net forms with extensive axiomatic restrictions requires training and patience - just as it does to create correct mathematical models of any sort for systems bound by physical law. At the lower end of restrictiveness, the training and patience required are minimal. The members of a systems community who use restrictive models are those primarily concerned with how and whether it works; unrestrictive models are for those primarily concerned with what it is supposed to do.

A signal advantage of employing net forms at every level of expression is the establishment of orderly relations between them. It is possible to map - in the mathematical sense of that word - a net which purports to represent an implementation to a net which purports to represent a desired system and verify, by means of the mapping, that the implementation fulfills the intention. This ability has to some modest degree been demonstrated and is a major part of what is yet to come in the development of net theory. It is, in any case, an ability that has been most especially lacking in current systems practice. There is no formal step which connects the code of a computer program o the task it is supposed to accomplish, a circuit design to the logic requirements it is supposed to meet, the institutional and technical procedures that are supposed to implement the intent of a law and the law itself.

Our project, as well as other projects in our field, have worked on various classes of net models as well as on the connections between them. In this report we present contributions both to theory and practice at several levels of axiomatic restrictiveness, as discussed above. Here is our table-of-contents organized in that way:


Axiomatic Restriction
Theory
Practice
Expressive Freedom
Low
Medium
High
I. Introduction
II. Role/Activity Models and Petri Nets
III. Role-Activity Nets in the Analysis of a Financial Information System
IV. The Connection of Parts to One Another in System Contexts
V. Definitions Pertaining to Petri Nets, States and Events and Behavior
VI. Calculating Logical and Probabilistic Dependencies in a Class of Net Models
VII. System Modeling with Net Structures
VIII. On Periodic Solutions of Affine Recurrence Systems
IX. Communication Mechanics
X. APPENDIX
High
Medium
Low

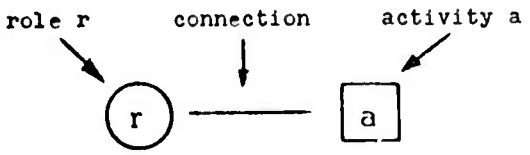
ABSTRACT

In this paper we develop a pair of concepts - roles and activities - as a point of departure for a formal theory of organization - or of systems. The resulting formal models are various species of bipartite graphs - directed or not - with rules for interpreting them as representations of behavioral relations. Some of the resulting structures are equivalent to Petri nets with their conventional "firing rule" (see Chapter V.). All of them are significantly interpretable as net topologies in the sense of C.A. Petri (see APPENDIX).

A. Role/Activity Interconnection

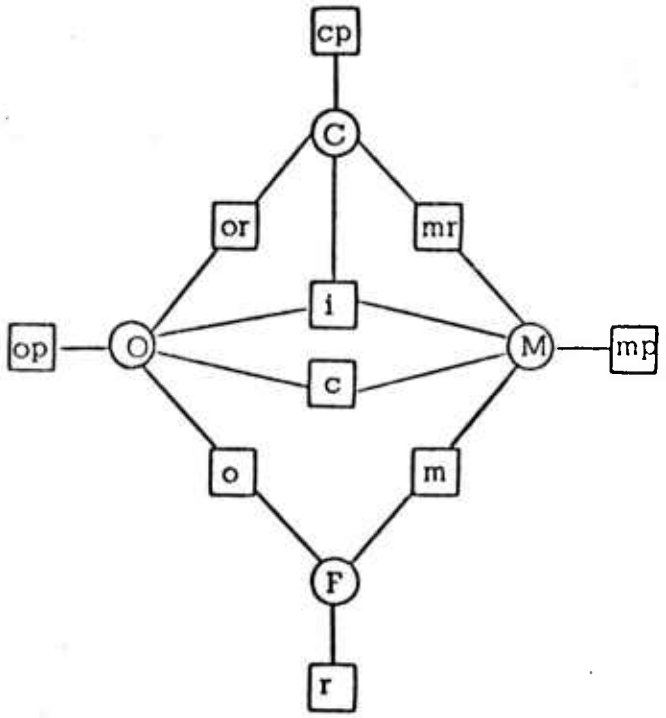
We will represent the interconnection between roles and activities as graphs with two types of vertices: vertices that represent roles, and vertices that represent activities. We will symbolize a role vertex by drawing a circle, and an activity vertex by drawing a square. In our first class of models, the connection between a role and an activity will be represented by an undirected edge.

A1.



If r is a role, we will designate the set of activities connected to it by the symbol r , and call these the activities of the role; if a is an activity, we will designate the set of roles connected to it a and call these the roles involved in the activity or, more briefly, the roles in the activity.

A2. Example[1]



C	controller	roles
O	operator	
M	maintenance man	
F	facility	
c	O and M communicate	activities
or	O reports to C	
mr	M reports to C	
i	C gives instruction to O and M	
o	O operates F	
m	M maintains F	
cp	C plans further action	
op	O plans further action	
mp	M plans further action	
r	F the facility runs	

---

[1] The picture in A2 is a graphic representation of an abstract structure called an undirected Petri net. Such nets may be formally characterized as follows:

An undirected Petri net is an ordered triple $\langle R, \Lambda, \alpha \rangle$ where

.1 R is a set of roles, symbolized by ○

.2 Λ is a set of activities, symbolized by □

.3 α is a relation $\alpha \subset R \times \Lambda$

.4 $R \cap \Lambda = \emptyset$

.5 $\text{dom}(\alpha) = R$

.6 $\text{ran}(\alpha) = \Lambda$

In the literature on Petri nets, the elements represented by circles are commonly called places and the elements represented by squares are called transitions.

B. System Dynamics

We will use undirected Petri nets (and subsequently other classes of nets) to specify, describe and analyze organizational behavior. For these purposes, the nets are to be used as game boards on which pieces - called tokens - are to be moved about according to rules. With respect to a given organization, we will be interested not just in a single net representation but in a class of related net representations for at least the following interpreted reasons.

B1 .1 Nets representing the behavior at various levels of detail

.2 Simple nets with complex rules for moving pieces and complex nets with simple rules for moving pieces - always representing the same behavior

.3 Nets which are more-or-less inclusive - i.e., shifting the boundaries between what is counted as belonging to the organization and what is counted as belonging to its environment.

Let us now take our first step in the direction of treating a Petri net as a game board. To make our description simple at the outset, we will suppose there to be one player of the "organization game", a player whom we may call the simulator. The game is to be played with a set of pieces, called tokens which, by their shapes or colors, are distinguished into as many types as there are roles on the board. The tokens represent persons or objects, singly or in aggregates, which play organizational roles. Anything which plays an organizational role will be termed an actor.

Each token, in addition to having its role - physically represented by its inalienable shape or color attribute - carries a "slate" on which characters may be written - characters which represent the attributes an actor may have that are variable over the time that he plays his role.

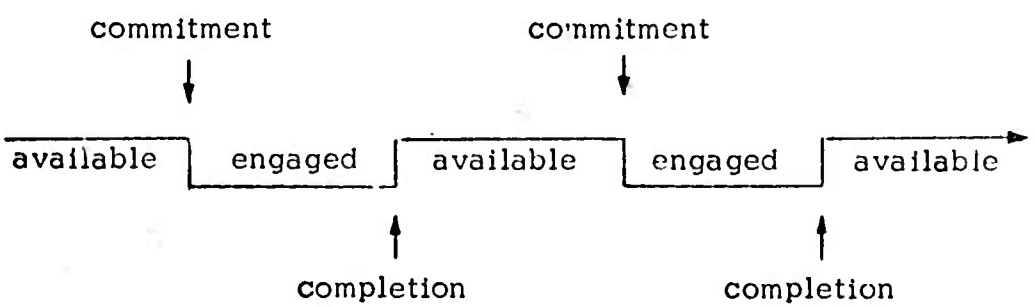
We will see an actor's total time as role player as divided into a succession of two kinds of periods:

B2. .1 Periods of engagement in an activity

.2 Periods of availability for engagement

The change from availability to engagement is termed a commitment; the change from engagement to availability is termed a completion.

B3. General form of an actor's behavior:



On the net, such a succession of periods would be represented in the following manner. A token, representing an actor, is placed on the circle corresponding to the actor's role. The residence of the token on the circle represents a period of availability for commitment to an activity. According to rules yet to be discussed, the token may next be moved to some square that is connected to the circle. This move represents a step of commitment, and the square to which the token is moved represents the activity to which the actor is committed. The residence of the token on the square represents a period of engagement; the token may then be moved from the square to the circle corresponding to its role. This move represents a step of completion and initiates a new period of availability.

An actor that plays a role can only be committed to the activities of that role. Correspondingly, a token on a circle can only be moved to squares that are connected to it.

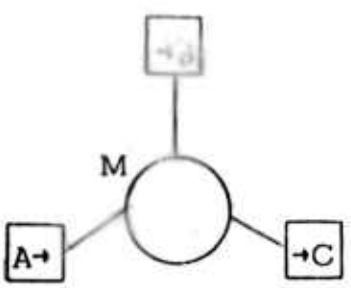
An actor may become committed to an activity only if he has attributes that make him a suitable participant. As a result of participation, he may change his attributes. Correspondingly, a token may enter a square with certain characters recorded on its slate, but these characters may be changed before it re-enters the circle.

The following are examples of rules for actor behavior, formulated in accordance with the generalities above.

B4. Example

.1 A messenger M picks up messages from A for delivery to either B or C, depending on the address on the message; after delivery, he is prepared for a new message pick-up.

We may model the context of this description as a role M with three activities: message pick-up from A; message delivery to B; message delivery to C.



To simulate the messenger's behavior we take a token with a slate with the following possible contents:

A →	ready to receive message from A
M → B	ready to deliver message to B
M → C	ready to deliver message to C

The token can only be committed to the activity [A→] if its slate carries the characters 'A→' ; if it participates in [A→] its slate will be changed to read either 'M→B' or 'M→C' ; it can only be committed to [→B] if its slate includes the character '→B' ; it can only be committed to [→C] if its slate includes '→C' ; in no matter which of these two activities it participates, its slate content will be replaced by 'A→' .

.2 Variation on .1: the same as .1, but whether the message is delivered to B or C depends not on the address on the message but whether B or C come to pick the message up. The possible slate contents of the messenger token are now reduced to two.

A→	ready to receive a message from A
M	ready to deliver the message M to B or C, but not to both

Delivery to either B or C results in replacement of the slate content 'M' by the slate content 'A→' .

.3 Another variation on .1: after message pick-up, the messenger must first try to deliver the message to B; if B does not pick it up, he must then deliver it to C . Possible slate contents are now the same as in .1, but the slate rewriting rules have changed as follows: As a result of participation in [A→] the slate content changes from 'A→' to 'M→B' ; as a result of participation in [→B] the slate content changes from 'M→B' to 'M→C' or to 'A→' , depending upon whether B does or does not pick up the message. The re-write for [→C] is the same as in .1 . Notice that the meaning of the activity [→B] has changed: it no longer means <u>delivery of a message to B</u>, but <u>attempted delivery of a message to B .</u>

.4 A variation on each of the examples .1 - .3: Let us suppose that no actor is permanently assigned to the playing of role M. Instead, let us suppose that when A wants to transmit a message he commissions an actor to play role M and that the actor's commission expires with the delivery of the message to B or C under any of the rules .1 - .3. The source of available actors for commissioning is not included in our picture nor is their destination after decommissioning. Now the simulator may proceed as follows with the net shown in .1. Starting with the net devoid of tokens, he places a token of type M with blank slate on the square [A→] signifying by that act the commitment of a suitable actor to being commissioned to play role M; he records on the slate a content representing a message to be delivered, and moves the token into circle M, signifying completion of commissioning; he moves the token into square [→B] or square [→C], as may be appropriate, signifying commitment of the actor to message delivery; he cleans the slate of the token and removes it from the net, signifying the decommissioning of the actor upon completion of his mission. Since a new actor might be commissioned to deliver a message before the commission of some other messenger has expired, several actors playing role M may appear on the net at the same time, and their number may vary in course of simulation.

In the class of organization models we are now considering, the effects of organizational functioning are reflected in changes of actor attributes and changes in the number of actors participating in the organization. These changes come about as a result of the taking place of activities in which the attributes of existing actors may change and actors may be commissioned or decommissioned in respect to the playing of particular roles.

For the purpose of simulation, there is associated with every square of a net a rule for how and when to enact that activity. In what way can the "when" part of the rule depend on the distribution of actors with their attributes on the net? It can only depend on the availability of an "appropriate" set of actors for participation in the activity. "Appropriateness" can only be expressed in terms of what roles they play and what other attributes they possess. In general, there may be many different cases of appropriate actor sets which enable the activity, one case differing from another, both in what roles and what attributes are specified. The when part of the rule may refer to other conditions that are to be assumed satisfied and are not expressed in terms of actor attributes found on the net. However, these "side conditions" may not refer to any actor attributes other than those belonging to the actors that are in fact to be committed to the activity.

We can re-express the idea of the when part of an activity rule in the following way. Thinking of each actor with his attributes as a potential resource for the taking place of activities, the conditions for the taking place of an activity, insofar as these are expressed by the net, can rest only in the availability of the appropriate resources.

Imagine, then, a current state-of-affairs in an organization represented by a distribution of tokens on a net. If the simulator finds a set of tokens on the circles connected to a given square which satisfy the when part of the rule for the activity, he can, in one act, move that set of tokens out of the circles and into the square, thus representing the commitment of that set of actors to that activity. The how part of the rule will tell him which, if any, of the actors committed to the activity are to be decommissioned, what new ones, if any, are to be commissioned, and what attributes are to be recorded on the newly constituted

actor set as a function of the attributes contributed by the actors that participate, and as a function of whatever side conditions are assumed to govern the activity. Upon completion of these changes, the simulator may represent the completion of engagement for the actors in the activity by moving the tokens, one-by-one, out of the square to their appropriate circles. If, in a given situation as represented by a distribution of tokens on the net, all requirements for the taking place of two activities are met, and a particular actor is required in both, only one of these activities can now take place - i.e., a single actor cannot be committed to two activities at the same time.

This description of simulation activity is obviously not yet formally complete, but it is sufficient to make clear the principal property of net representation relative to system dynamics - as follows.

Suppose that at some stage of simulation a particular actor A has a particular attribute X. How can this fact affect the subsequent attributes of other actors in the organization? To begin with, only actors who may become committed to an activity in which A also participates can possibly be affected by it. If and when such another actor B comes to possess an attribute Y which, at least in part, depends on the attribute X which A possessed, some other actor who co-participates with B in an activity may come to possess an attribute Z which depends, in part, on Y and thus, in part, on X. Recursive application of this form shows that the effect of attribute X of A can only propagate along connected paths of the net - from circle to square, to circle to square, etc. <u>This is, in respect to system behavior, the basic meaning of the structure of connection represented in a net.</u>

C. Directed Nets, and a First Use of Net Mappings

Let us now restrict our attention to nets with token games that have the following property:

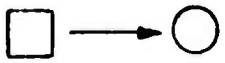
C1

.1 If the taking place of activity a ever results in commissioning an actor to play role r , then every taking place of activity a which involves role r at all, must result in such a commissioning.

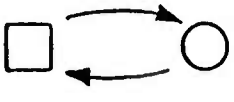
.2 If the taking place of activity a ever results in de-commissioning an actor who plays role r , then every taking place of activity a which involves role r at all, must result in such a de-commissioning.

Property C1 means that we can meaningfully adopt the following new notation in role-activity nets.

C2

.1  activity a is <u>initial</u> to role r - i.e., may result in commissioning an actor to play role r

.2  activity a is <u>terminal</u> to role r - i.e., may result in de-commissioning an actor who plays role r

.3  activity a is <u>medial</u> to role r - i.e., is an activity in which an actor playing role r may participate

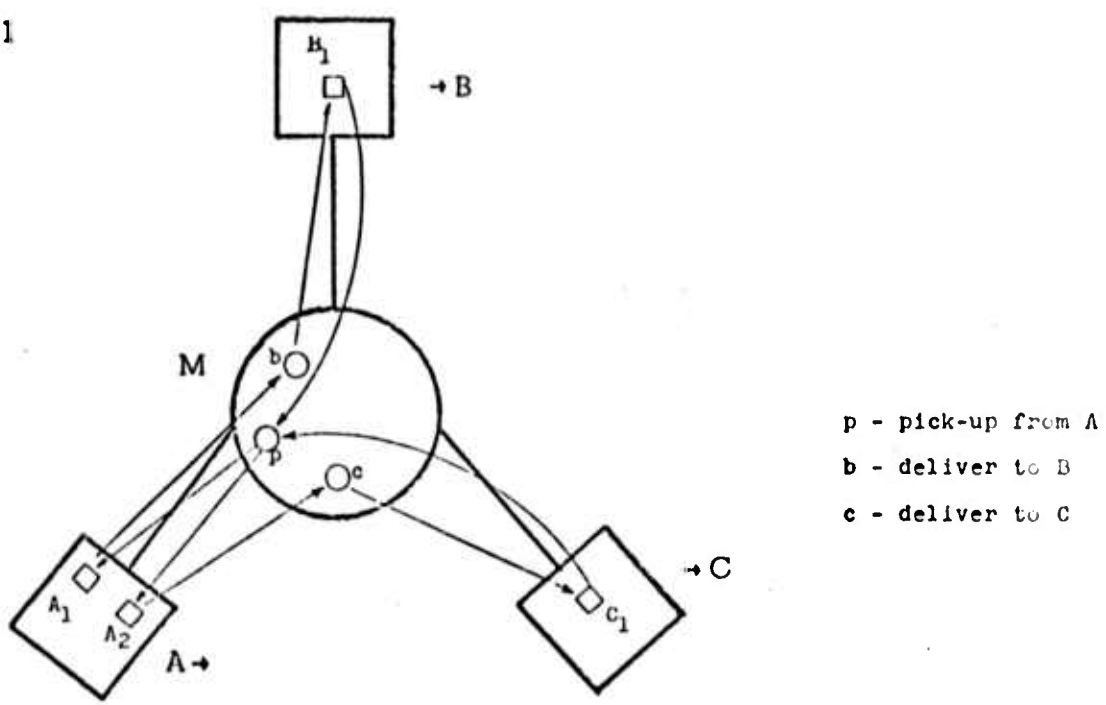
Property C1 guarantees that C2.1 - .3 is a partition of the activities of a role.

The differences in messenger behavior described in examples B4.1 - .4 were represented by differences in token games, all played on a single net (the net shown in B4.1). We will now represent these differences by constructing a series of different nets in all of which a single rule for the token game can be used. We will do this by breaking the messenger role up into smaller constituent roles, and breaking the activities up into smaller constituent activities.

C3. A messenger M picks up messages from A for delivery to B or C, depending on the address on the message; after delivery, he is prepared for a new message pick-up.

We may model the context of this description as a role M with three activities: message pick-up from A; message delivery to B; message delivery to C.

.1



C3.1 shows us two nets - one drawn light and, therefore, to be called $\mathcal{L}$, and one drawn dark and, therefore, to be called $\mathcal{D}$. The physical arrangement of the two nets expresses a mapping of $\mathcal{L}$ to $\mathcal{D}$, a mapping with the following properties:

C4

.1 Each role of $\mathcal{L}$ is mapped to a role of $\mathcal{D}$

.2 Each activity of $\mathcal{L}$ is mapped to an activity of $\mathcal{D}$

.3 If a role and an activity are connected in $\mathcal{L}$, then their images are connected in $\mathcal{D}$

This mapping is a net homomorphism and it is a special case of continuous maps with respect to net topology, as defined by C.A. Petri (see APPENDIX).

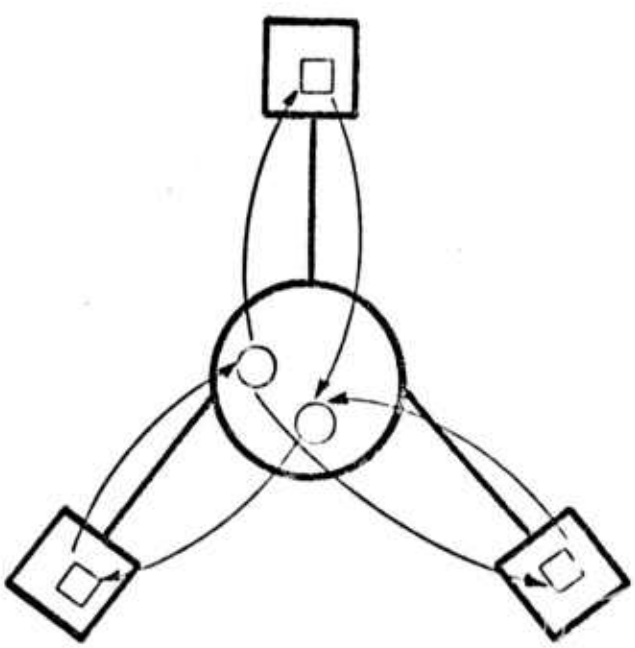
What token game on $\mathcal{L}$ corresponds to the token game defined in C4.1 for $\mathcal{D}$? In $\mathcal{L}$ we have three token types - representing actors in roles p, b, and c. The slates with characters are now everywhere superfluous. Any actor in role p can participate in activities A_1 or A_2 ; if he so participates, he is de-commissioned.

The taking place of activity A_1 always results in the commissioning of an actor to play role b while the taking place of activity A_2 always results in the commissioning of an actor to play role c ; any actor in role b can participate in B_1 which, if it takes place, results in the de-commissioning of that actor and the commissioning of an actor to play role p ; analogously, for role c and activity C_1 .

Although net $\mathcal{L}$ with the token game as just defined represents the accomplishment of the messenger function, it does this with the help of three roles and, correspondingly, three sets of actor specialists who each do their bit. That these bits are bits of a single total performance is expressed by the mapping of all three roles of $\mathcal{L}$ to the single role of $\mathcal{M}$. We can now express this meaning dynamically by a modification of the token game as just described - namely, to use a single token type for all three roles of $\mathcal{L}$, a type which represents the role M in $\mathcal{M}$. Now we can say of activity A_1 that its taking place results in an actor M losing his commission to play role p and an actor M gaining a commission to play role b. Since we can only recognize an actor by his attributes - in the present case, only the attribute M - we may as well say that in activity A_1 an actor M switches from role p to role b. Thus, in each of the activities of $\mathcal{L}$ an actor M - i.e., a messenger - switches from one role to another, roles which collectively may be termed sub-roles of the messenger role.

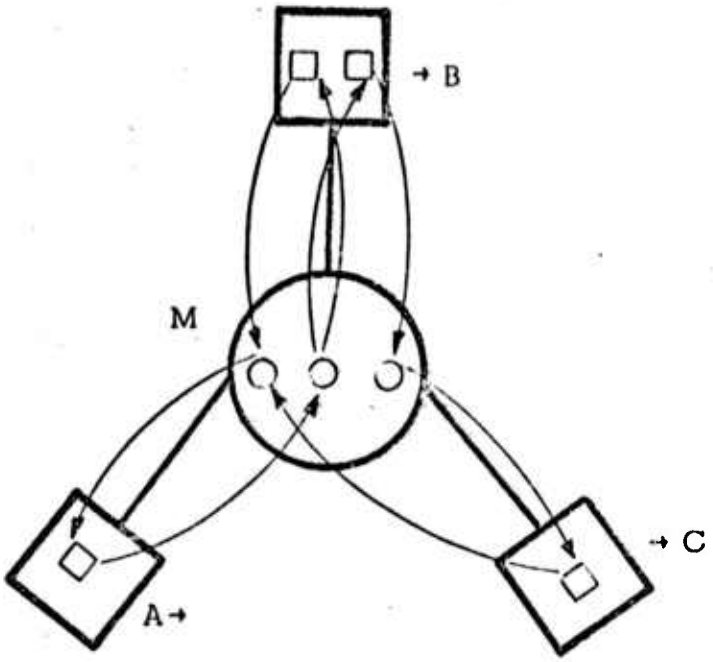
Analogously we may render B4.2 thus:

C5 .1



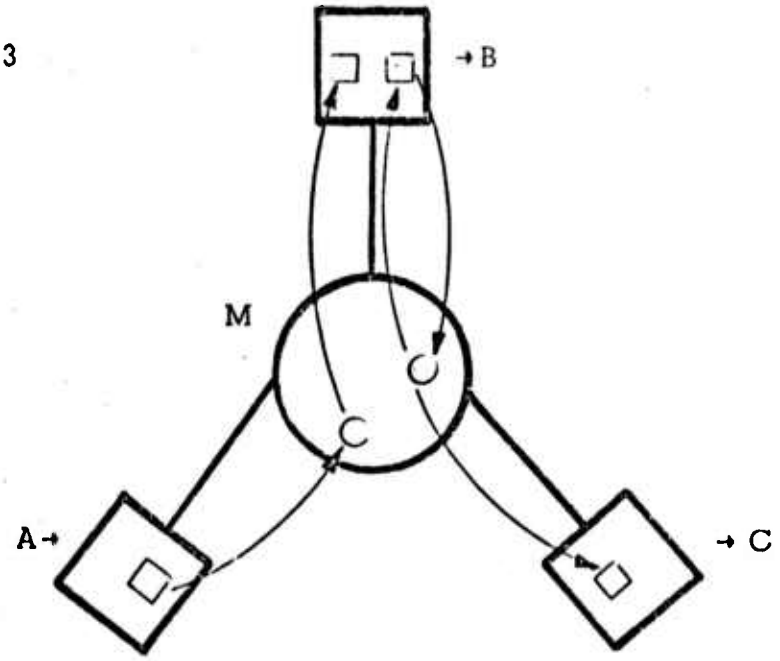
and B4.3 thus:

.2

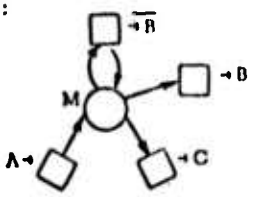


and B4.4, as applied to B4.3[1], thus:

.3



[1] It is interesting to note that, to represent this case as a net with a single role M, while taking conventions C, into account, one must split the activity [→B] into two activities - delivery and non-delivery - with the result:



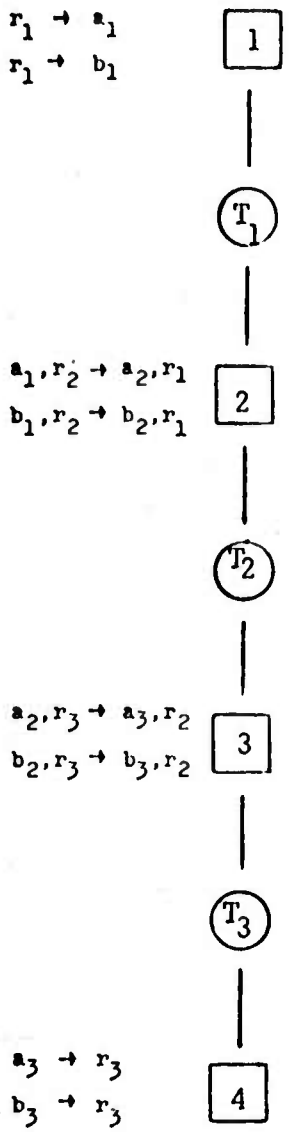
Notice that the dynamics associated with all of the light nets in C3 and C5 have the following characteristic in common.

C6. The taking place of an activity depends on the availability of an actor in each role to which the activity is terminal or medial and on nothing else. Upon completion, one actor is made available in each role to which the activity is medial or initial, and in no other role.

In nets with directed arcs drawn according to conventions C2, C6 yields the standard Petri net firing rule as described in A5 of chapter V.
Relative to every role-activity net $\mathcal{D}$ with token games as described above, one can construct a directed Petri net $\mathcal{L}$ and a net homomorphism which maps $\mathcal{L}$ to $\mathcal{D}$ (as in C3 and C5) and maps the standard Petri net token game in $\mathcal{L}$ to the token game in $\mathcal{D}$. In this chapter we will not express the content of this remark in a mathematically acceptable manner, but we will illustrate its content with our closing example.

We will model a chain of transmitters which pass messages - either a's or b's - from one to the next.

C7.



We will assign a simple actor to each transmitter role. The slate of an actor in role T_i can bear one of three characters:

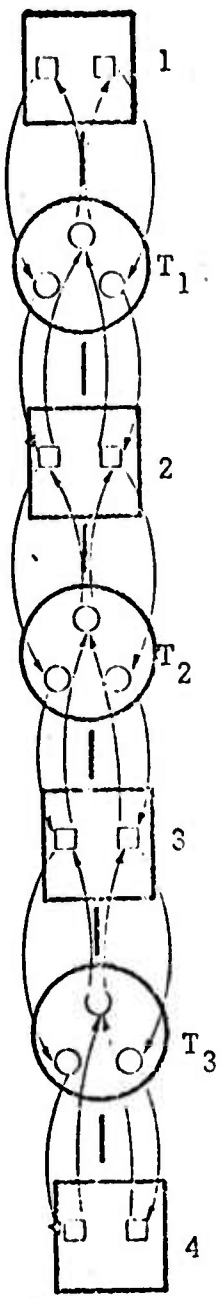
r_i - ready to receive

a_i - ready to transmit an a

b_i - ready to transmit a b

Next to each square in the net appear the slate rewriting rules when a set of actors engage in the activity. Each line next to the square represents a possible case for the activity - what actors with what attributes are required, and what new slate contents are produced. In activity 1 the simulator must decide on the basis of a side condition which part of the rule to apply.

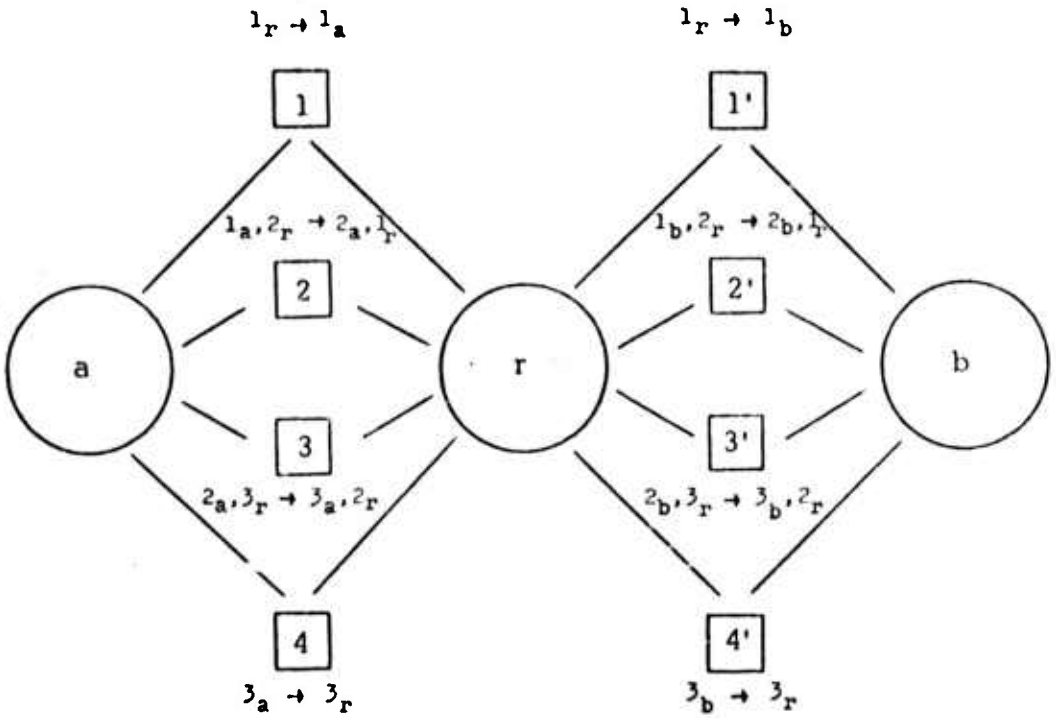
.2



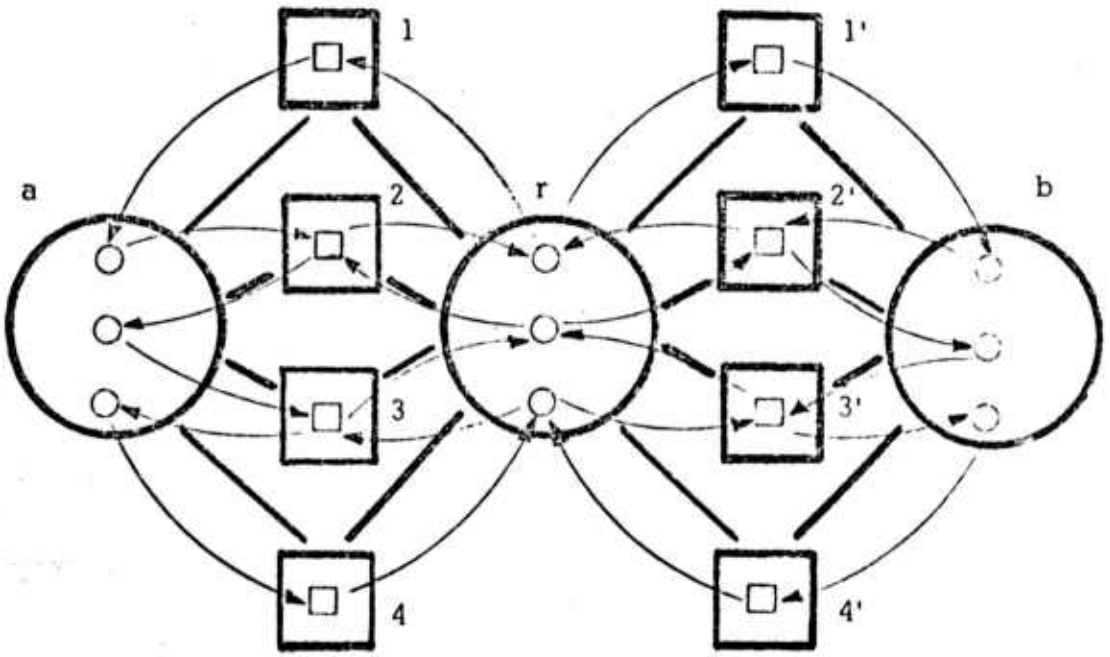
In this figure the dark net $\mathcal{L}_2$, and the light net $\mathcal{L}_2$ relate to one another exactly as the nets $\mathcal{L}$ and $\mathcal{L}$ in C3 and C5.

In .1 we represented the passing of messages, a or b , as meetings between pairs of transmitters. We can also think of these activities as meetings between messages and ready signals. For this purpose, we will define three roles - a , b , and r . Actors who play these roles - the role of an *a* message, the role of a *b* message and the role of *r* ready signal - will have attributes representing their positions, 1, 2, or 3 .

.3



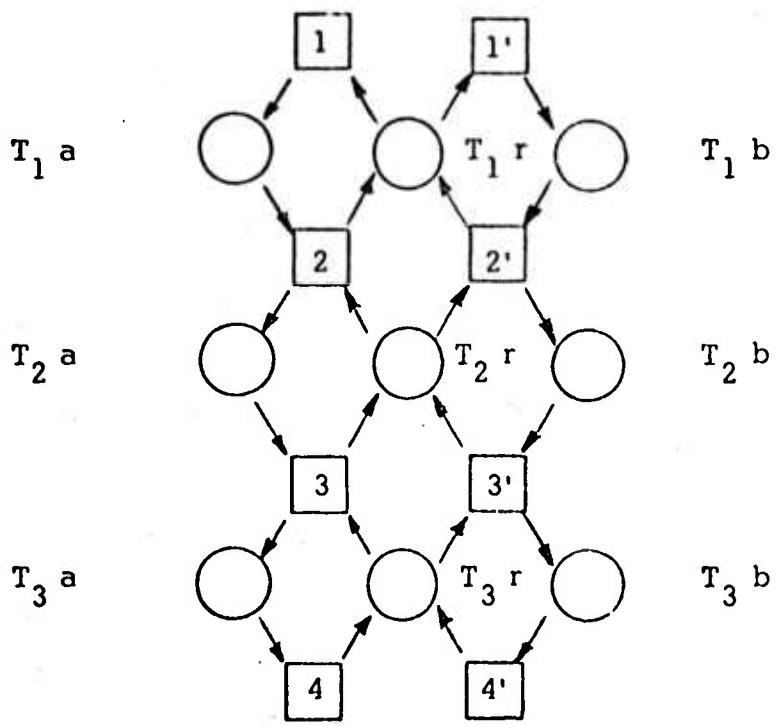
.4



We will refer to the dark net in this figure as $\mathcal{Q}_4$. The light net in this figure is identical to $\mathcal{L}_2$ in .2 .

We can regard the mapping of $\mathcal{L}_2$ to $\mathcal{A}_2$ and the mapping to $\mathcal{A}_4$, as associating with each role in $\mathcal{L}_2$ two labels: the name of the role to which it maps in $\mathcal{A}_2$ and the role to which it maps in $\mathcal{A}_4$. The following figure shows $\mathcal{L}_2$ with these labels associated with its roles.

C8.



Each role in C8 is uniquely labeled, and the labels express the meaning of these roles with reference to $\mathcal{A}_2$ and $\mathcal{A}_4$. If we now play the intended token game on C8 with nine token types - one per role - and regard the type as composed of two labels, we could interpret the simulated behavior in at least the following

two ways: actors of types T_1 , T_2 , and T_3 which, as a result of interaction, switch between roles a, b and r ; actors of types a, b and r which, as a result of interaction switch between roles T_1 , T_2 , and T_3 .

The roles T_1 , T_2 , and T_3 have no initial or terminal events, while the roles a , b and r do. (Referring to $\mathcal{S}_4$ we note that activity 1 is terminal to r and is initial to a, activity 1' is terminal to r and initial to b, activity 4 is terminal to a and initial to r , activity 4' is terminal to b and initial to r .) Thus, in the token game C8, the actors of types a , b and r not only switch between roles T_1 , T_2 , and T_3 , but are also occasionally de-commissioned, while occasionally new ones are commissioned. This corresponds to the interpreted reality that we have represented an open segment of a message transmission line into which new messages flow from an unrepresented source and disappear to an unrepresented destination - and the same for control signals.

There is an interesting difference to note as between the token game C7 and the token game C8 interpreted as actors of types T_1 , T_2 , and T_3 switching between roles a , b , and r . In C7, a_1 , a_2 , a_3 , b_1, b_2 , b_3 , and r_1 , r_2 , r_3 are simply nine different characters. In the context of C8, we have a formal interpretation for the intent that a , in the context of T_1 is the same as a in the context of T_2 , and a in the context of T_3 - and similarly for b and r .

MASSACHUSETTS
COMPUTER ASSOCIATES, INC.
26 PRINCESS STREET, WAKEFIELD, MASS. 01880 • 617/245-9540

Role-Activity Nets in the Analysis of a Financial Information System

by

J.M. Myers
18 Joy Street
Boston, Massachusetts 02114

March 14, 1975

PREFACE

In this paper we apply role-activity nets to the analysis of a computerized financial information system. For this purpose we can specialize the idea of role to that of agency as defined in the body of the report. The system described is the Public Finance Subsystem, developed primarily by Westinghouse for the City of Dayton, under Contract H-1215 from the Department of Housing and Urban Development. Our intent is to clarify how the various pieces of the Dayton System fit together to get an idea of what modifications might be possible or desirable with respect to its use by other cities.

The material presented here is a synthesis which draws mainly on the contributions of A.W. Holt and C.A. Petri, but also substantially on work of M. Hack, M.W. Marean, J.M. Myers, and R.M. Shapiro. What is presented is an introduction to a body of methods related to Pragmatic Analysis of Petri and Communications Mechanics of Holt. While no substantial mathematical apparatus appears here, the mathematically inclined reader might correctly surmise that such an apparatus is coming into being. Indeed, an axiomatic foundation and a substantial body of theorems already exists.

INTRODUCTION

To begin, let us draw a square to represent the Dayton System, more-or-less as represented by the Westinghouse documentation, and a circle to represent the City agencies and activities which must interact with that System, and without which the System would have no meaning.

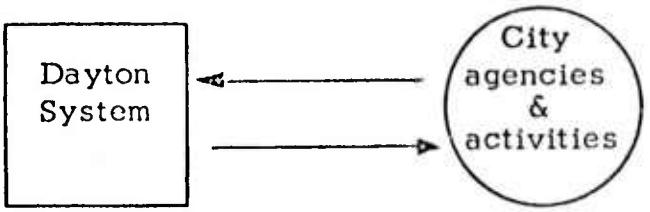


Figure A1: Grossest level of representation of Dayton System in contact with City agencies and activities.

The "Dayton System" as presented by Westinghouse is essentially a package of computer programs, forms, and manual activities which imply the participation of various agencies of the City of Dayton, that City now viewed as an organization of people acting as agents in activities with each other and with the "Dayton System". Because of this implication, we have drawn the link in Figure A1 to show information exchange. The highlighting of this link is the main result of Figure A1.

To better understand the system, we can pursue its anatomy. Westinghouse divides the Dayton System into nine components. Many of these components are in turn divided into smaller units called applications. Five of the components are documented; four will not be documented for several more months. Both components and applications are organized in terms of programs and files. A program belongs to only one application and hence to only one component. A file may be created, accessed, or updated by several applications in one or several components. There are around 120 files in the five documented components, and of these, 16 are used by more than one component. By analyzing the documentation, which includes tables of files used by programs and programs using files, we can expand Figure A1 to Figure A2.

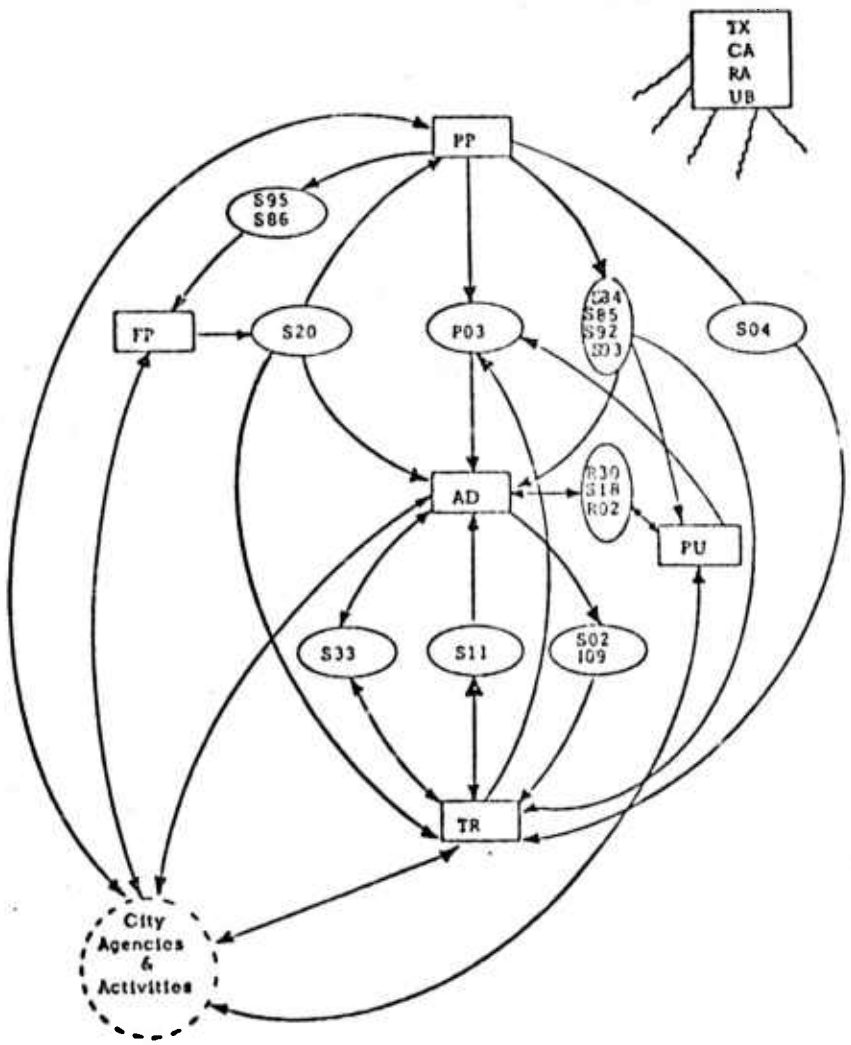


Figure A2: Expansion of Dayton System into documented components linked by files. (See Key on next page. Wavy lines denote unknown connections involving components not yet documented.)

Key to Figures

AD	Accounting Disbursing Component
AD.190	Program FADPS190 which, according to AD file index, creates the General Ledger Transaction Table (FADFDS31) but which is not mentioned in program descriptions
AD.935	Program FADPU935 which, according to the AD file index, creates the General Ledger Account Translation Table (FADFDS06) but which is not mentioned in program descriptions
FP	Planning, Programming and Budgeting Component
FP.910	Table Update Maintenance Program FFPPU910 which creates the Account Validation file (FFPFDS20) from input cards without contacting any other files in the system
FP-910	The PPB component not including FP.910
PP	Payroll/Personnel Component
PP.SM	Supplementary Maintenance Application of Payroll/Personnel Component
PP-SM	Payroll/Personnel Component not including Supplementary Maintenance
PU	Purchasing Component
TR	Treasury Management Component

TR.CW	Payroll Check Write Application of the Treasury Management Component
TR-CW	Treasury Management Component not including Payroll Check Write
TX, CA, RA, UB	Taxation, Cost Accounting, General Revenue Accounting, and Utility Billing Components not so far documented

FILES

S01	FTRFAS01 Start-End Check Numbers
R02	FPUFDR02 Encumbrance
S02	FADFDS02 Warrant Detail (specifies warrants to be written)
P03	FADFDP03 Accounting Transactions (feeds transactions into AD component from Payroll, Purchasing, and Treasury components)
S04	FPPFDS04 Payroll Check and Register (conveys all information except check number to the TR.CW application for the writing of paychecks)
S06	FADFDS06 General Ledger Account Translation Table
I09	FADFDI09 General Ledger
S11	FADFDS11 Outstanding Warrants

S18 FPUFDS18 Requisition Master

S20 FFPFDS20 Account Validation (used to check that charges are to combinations of Fund, Organization, and PPBS code stated as valid by, presumably, the Director of Budget

R30 FADFDR30 Liquidation (linked to Encumbrance File R02)

S31 FADFDS31 General Ledger Transaction Table

S33 FADFDS33 Accounts Payable Suspense (holds accounts-payable records in suspense until due for payment)

S84 FPPFDS84 Fund Translation Table (produced by PP.SM to hold titles of funds corresponding to fund codes; used by AD, PU, and TR components in producing reports)

S85 FPPFDS85 Organization Translation Table (produced by PP.SM to hold organizational titles corresponding to organizational codes; used by AD component)

S86 FPPFDS86 Classification Translation Table

S92 FPPFDS92 PPBS Translation Table

S93 FPPFDS93 Object of Expense Translation Table

S95 FPPFDS95 Position Master

Much less information is documented concerning the anatomy of the City agencies and activities. Westinghouse asserts that the reason for this lack is that the Dayton System is supposed to be widely applicable and not particular to the City of Dayton. However, in order to understand the Dayton System even in principle at least some of the City structure must be reconstructed. For example, it is only in reference to a pattern of authority and responsibility that the assignment of some functions to some components make sense. These assignments could in no way be determined functionally. An adequate reconstruction requires more information than is available from the documentation. However, a rough beginning can be made by examining the signatures required on various input forms, the titles of output reports, and the remarks of a general explanatory nature distributed throughout the documentation. In Figure A3 we represent the City agencies directly alluded to in the documentation by dotted circles. To keep the picture as simple as possible, we contract the files shown in Figure A2 into a single circle, shown solid. The documentation implies but does not fully define certain activities among the City agencies. These activities shown in Figure A3 as dotted squares. It is clear that administrative departments, such as Personnel, Payroll, and Purchasing need to be injected into the picture at least as buffers for the Departments and probably for other functions. However, we are making a picture only of what the documentation states or implies and the documentation does not define more than is shown in Figure A3.

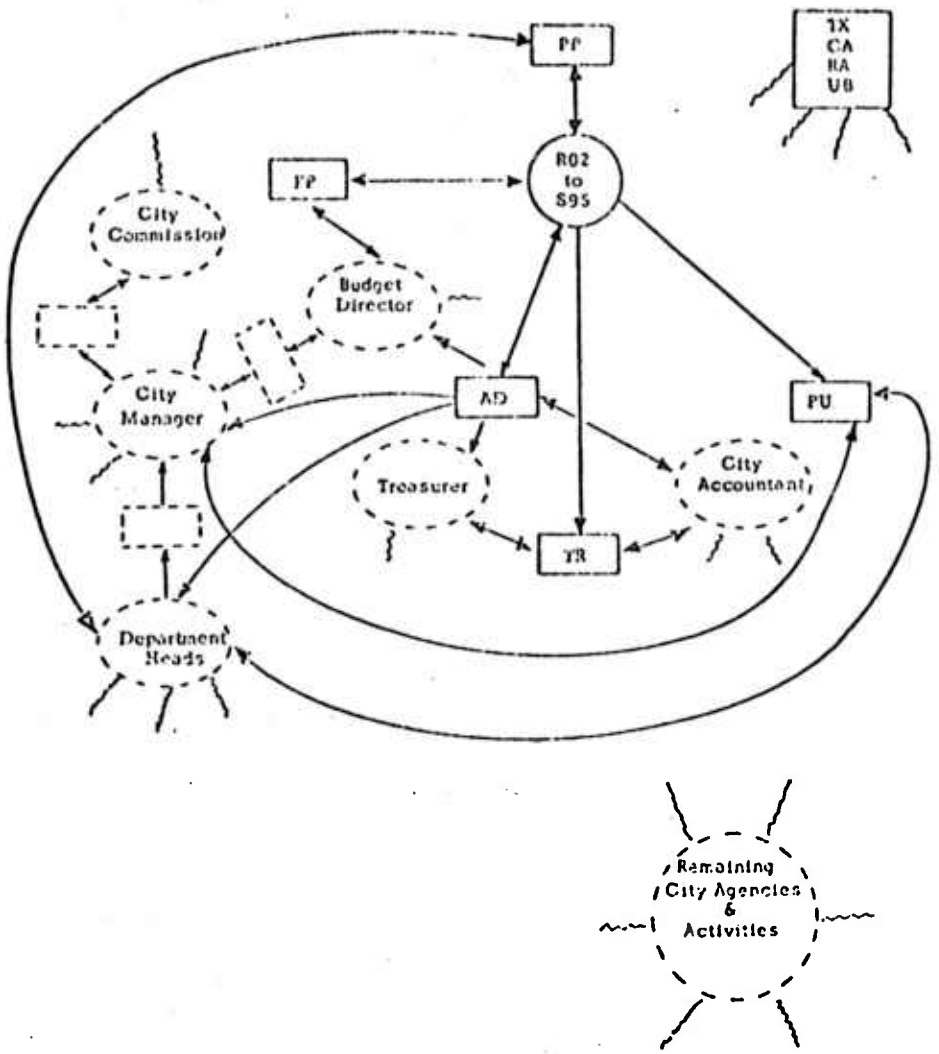


Figure A3: Partial expansion of City Agencies and Activities in connection with the Dayton System.

Flow from City agencies into components of the Dayton System typically takes the form of documents to be key punched: requisitions, time cards, etc. Flow from the components to the agencies consists of pay checks, edit reports, management reports, etc. The reports produced by each component are listed in the Westinghouse documentation, under "Output Reports" for each application of each component.

Mappings

Figures A1, A2, and A3 are all related to each other. We can and shall develop other related pictures. These relations can be understood in terms of mappings between nets. By a net, we mean a picture consisting of circles and squares with arrows from circles and squares and arrows from squares to circles. Whichever the arrow direction, we will say such a circle and square are connected. The smallest net has one circle and one square. No circle or square may stand alone without being connected to anything. Where we have drawn pictures with double-headed arrows, these are to be interpreted as shorthand for a pair of arrows, one going one way and the other going the other way. Where wiggly lines are used around a square or circle, these mean that the square or circle is connected but that we do not, at the moment, know how.

Each picture entails a set consisting of its squares and circles. By a mapping between pictures we mean a mapping, in the usual algebraic sense, from one such set to the set corresponding to a second picture.

It is easy to define a topology on nets. The closure of any subset is that subset together with any squares to which it is connected. Hence, closed subsets are connected only to circles, and open subsets are connected only to squares.

The mappings of interest are all continuous mappings in the sense that the inverse image of an open set must be an open set. Less formally, continuous implies that if a mapping puts a square into a circle, then it must put all the circles connected to the square into the same circle. Continuous also precludes losing any connections by breaking them. Mappings are classified by Petri in terms of properties preserved by the mapping. A mapping may be open, closed, quotient, surjective, injective, s-s, t-t, folding, or several of the above, or none of the above. These are defined as follows:

open if the image of any open set is open;

closed if the image of any closed set is closed;

quotient if both open and closed;

s-s if circles map only into circles;

t-t if squares map only into squares;

folding if s-s and t-t;

injective and surjective have their usual algebraic meaning.

To the above list I find it convenient to add two specializations:

square bundle for a surjective folding which is bijective with respect to circles;

circle bundle for a surjective folding which is bijective with respect to squares.

A square bundle pastes squares together without breaking any arrows, and leaves circles unchanged. A circle bundle pastes circles together without changing squares.

These properties of mappings are not independent; their relations to each other can be seen in the following diagram.

For this diagram and no other in this report, an arrow means implies.

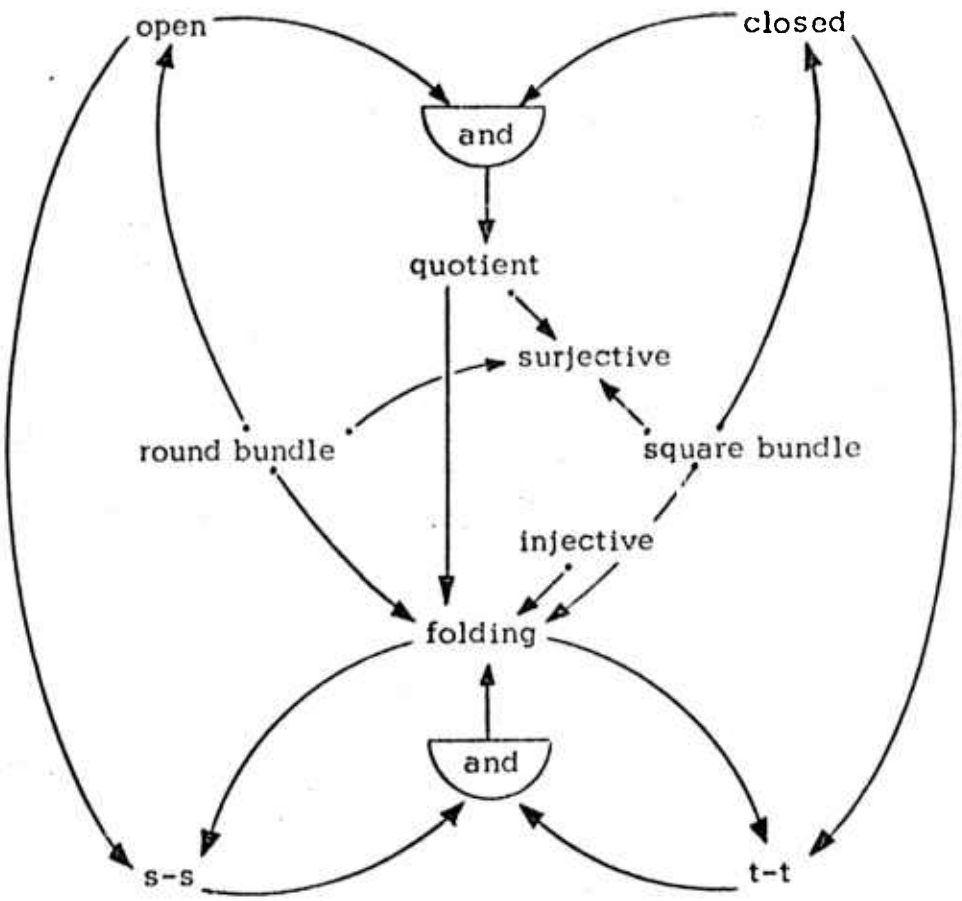


Figure A4: Relations between properties of continuous mappings.

The following sketches illustrate some of the types of mappings. Surroundings that are implied by single-ended links are assumed to be the same in both range and domain nets.

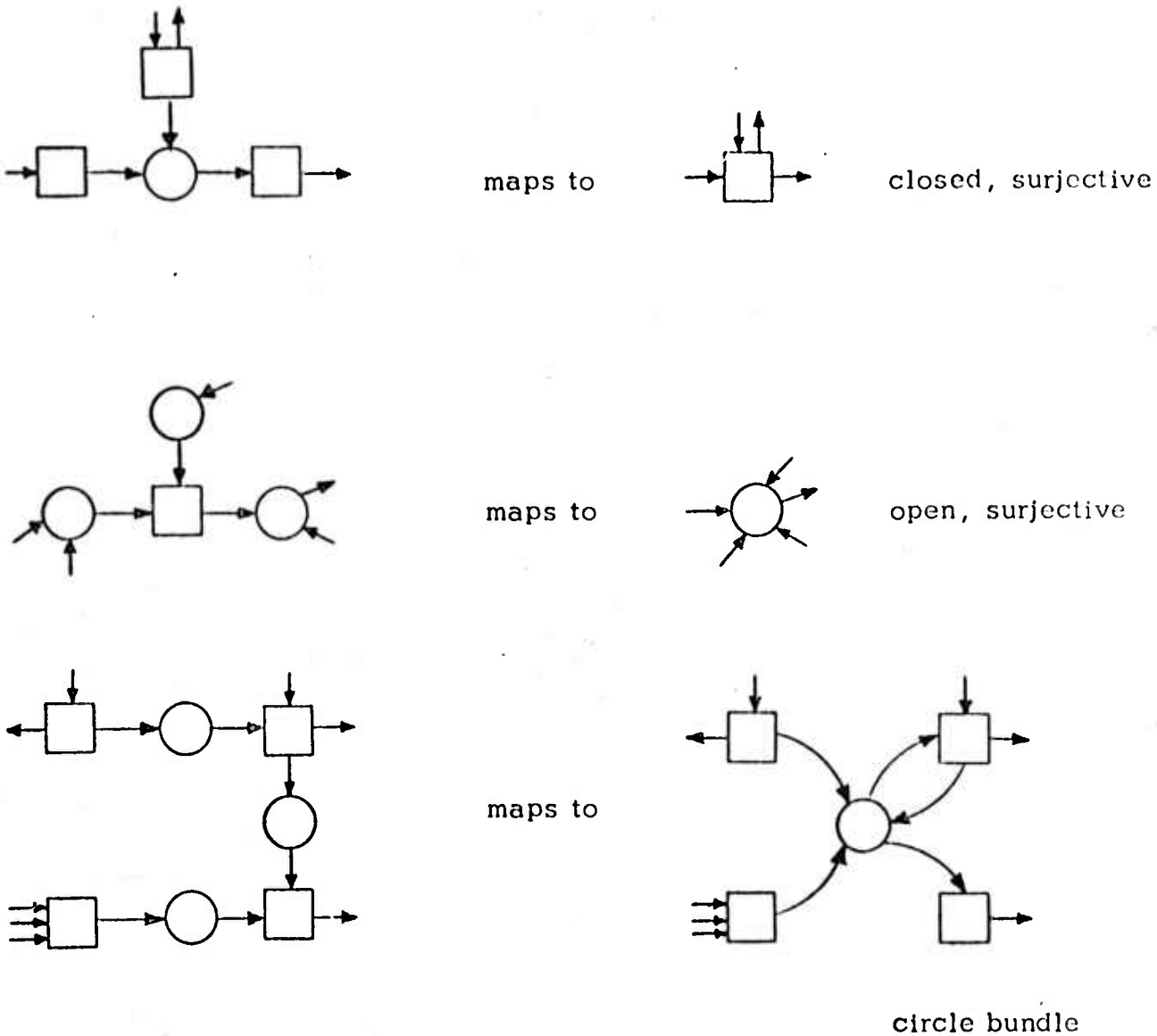
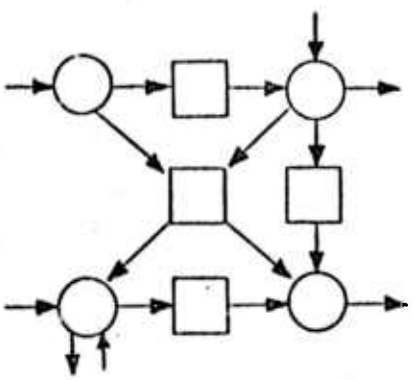
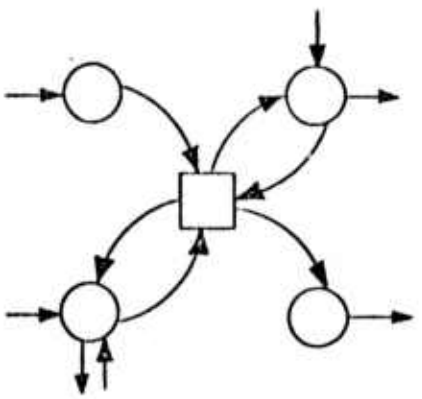


Figure A5: Examples of Continuous Mappings

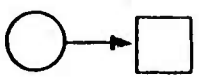
Figure A5 Continued



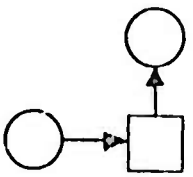
maps to



square bundle



maps to



injection

There is no continuous mapping from Figure A2 to Figure A3, but the two figures are obviously related. The relation is seen from the fact that both Figures A2 and A3 map to a common finest coarser net, and from a common coursest finer net. The finest courser net is obtained by circle bundling of the solid circles of Figure A2 into a single circle or by circle bundling of the dotted circles of Figure A3 into a single circle. The coarsest finer net is obtained by representing the files as in Figure A2 and the City Agencies and Activities as in Figure A3.

A picture of a system is not to be thought of in isolation, but rather in terms of a category of such pictures. One might say that each particular picture, for a given system, represents a point of view. For example, there is no end to the variations we could develop from Figure A2 and A3. We could represent resources such as the computer hardware, for instance by representing a processor as an agency (circle).

Synchronization

There is a shortcoming with the pictures as so far displayed; one cannot tell what activity is supposed to be complete before what other activity or activities begin. The synchronization can be portrayed by placing a token on each link of the net, and by providing appropriate rules and interpretations for the moving of such tokens. In this note, we consider only tokens of a single color. However, we note in passing that by using more than one color it is possible to portray not only synchronization, but also decisions and information flow.

In either case the rule is that when the last token moves to the circle end of its link, a token is moved away from the circle on the next link. We call a circle with this rule an ordered circle. In other words, the ordered circle exercises its links one at a time in a regular cyclic order. In all the cases I have seen where it is desirable to have a different rule for a circle, that rule can be expressed by viewing the circle having arisen from a mapping that took an open fragment of net composed of ordered circles and squares into the given circle.

For example, Figure A2 contains a circle representing a bundle of files S84, S85, S92, and S96. The circle has one incoming arrow and three outgoing arrows. The files are in this case table . PP has the opportunity to update these files once during each cycle of operation. After that opportunity, whether or not PP uses it, AD, PU, and TR have the opportunity to read the files. It does not matter in what order the readers read, but it does matter that opportunities for reading alternate with opportunity for PP to write. (Note that it is the opportunities which have to be synchronized; for synchronization it does not matter whether or not an opportunity is taken or not.) The appropriate rule is that if the last token not near the circle moves in along an output arrow, the token on the input arrow is to be extended away from the circle. If the last token not near the circle moves in on the input arrow, then move all the tokens on output arrows away from the circle. This rule is equivalent to viewing the circle for S84, etc., as mapped from a fragment as shown:

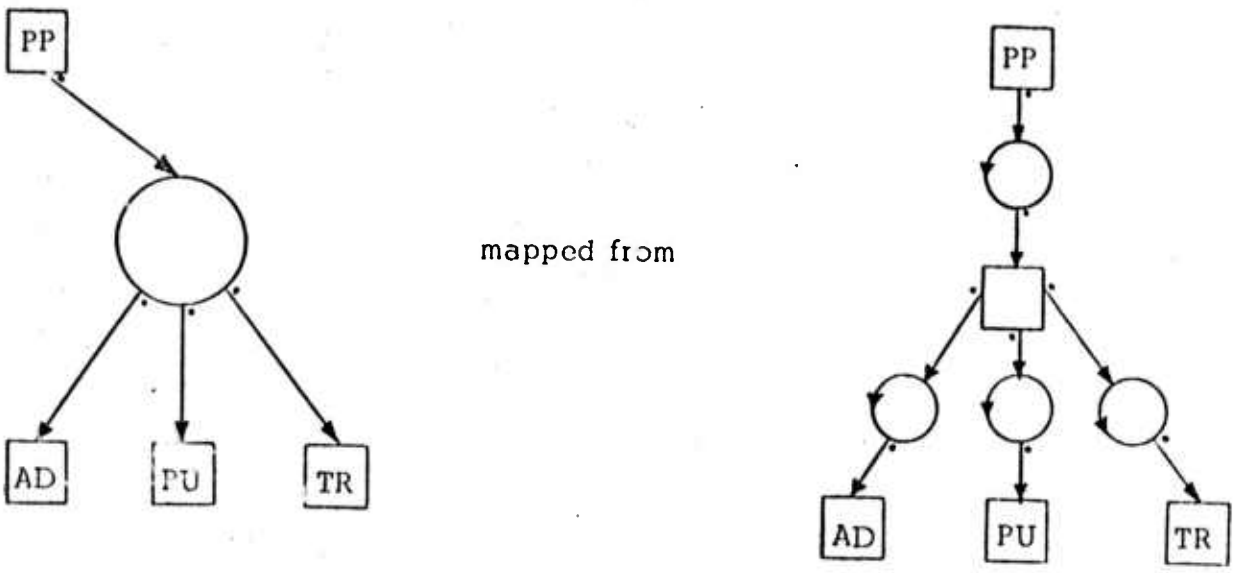


Figure A7: Expression of fan-out rule, in terms of ordered circles

The dots represent the positions of an initial marking. In general, the playing of a token game from some initial marking (i.e., placement of the tokens) will not produce all possible markings. Hence an initial marking determines a marking class.

M. Karr has called to my attention that the left and right parts of Figure A7 do not mean quite the same. The return of the token to PP for the left part means that, so far as the files are concerned PP can write again, but also that the files have been read by AD, PU, and TR. In the right part of Figure A7 the latter meaning is not strictly conveyed; the operation is buffered. If the stricter meaning is meant, we require something like a return path:

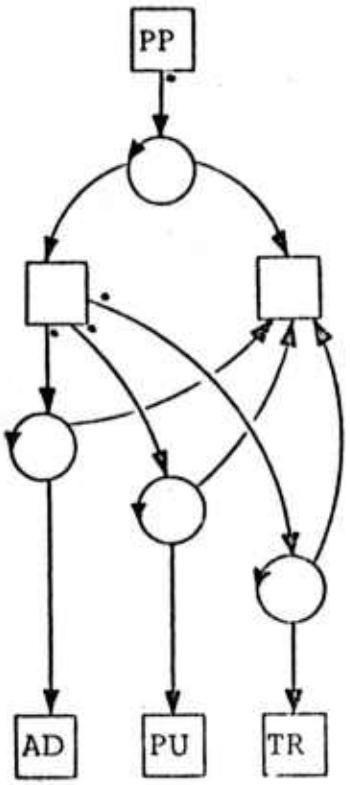


Figure A8: Expression of fan-out, including confirmation of reception

In marking Figure A8 I have used an observation of Marean that for an ordered circle one may consider the links as belonging to the circle and use only one mark per circle. This is because for an ordered circle, only one mark is ever away from the circle so it is the only one that needs to be tracked.

For purposes of analyzing the Dayton System, the meaning expressed in Figure A7 is adequate and describes the firing rule which we shall apply whenever, in connection with this analysis, we draw a circle with one arrow in and several out. By reversing all the arrows in Figure A7 we obtain the firing rule for the kind of fan-in that is meant for file P03 in Figure A2. As well as I can surmise from the documentation, the P03 file is partitioned into what is in effect three files so that PP, PU, and TR write in disjoint parts of it, so that the order of writing does not matter. If this surmise is correct, then the firing rule (i.e., the rule for moving tokens away) for P03 is obtained by viewing P03 as a contraction:

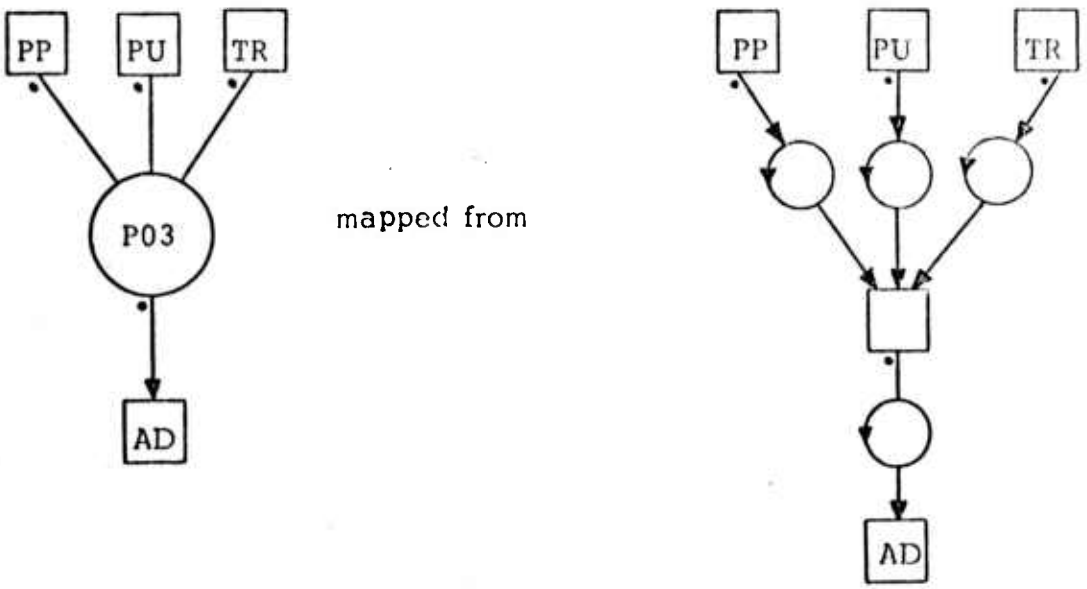


Figure A9: Fan-in showing alternation between three concurrent inputs and a single output

If the surmise is not correct, we can still represent the behavior. For example, if consultation with Westinghouse shows that the order of writing strictly matters, we have only to consider the circle for P03 to be an ordered circle.

Figure A2 contains double headed arrows as short hand for a pair of oppositely direct arrows. For the Dayton System, the reason for the double direction is to represent an update operation on a file where the file is first read and then written in. Such an update necessarily involves internal storage in the program; that is, the program contains both a file and internal sequencing. Both the storage and sequencing can readily be represented. For example, the firing rule for the S11 circle of Figure A2 is expressed by viewing S11 as an ordered circle and TR as contracted from TR' and TR" as shown in Figure A10.

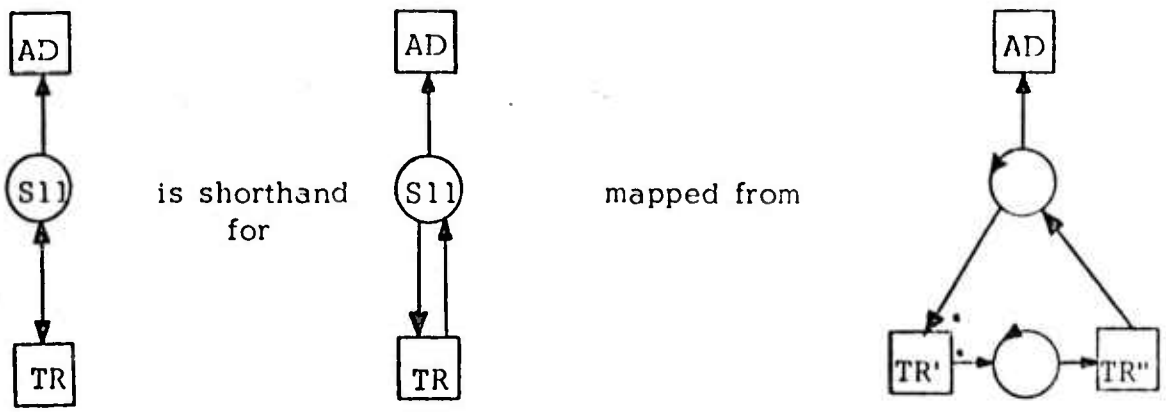


Figure A10: Expression of firing rule for update in terms of ordered circles

As a practical trick, one gets the same result by ignoring the heads on the arrows of S11, S33, and S02/I09, and just firing those circles as though they were ordered. The advantage of the mapping approach is two-fold. First it makes it more clear why such a short cut is acceptable, and second it provides an approach for expanding TR, etc., in more detail.

We can show an example of how a marking represents a marking class and hence a pattern of synchronization. Figure A2 contains a fragment as shown in Figure A11.

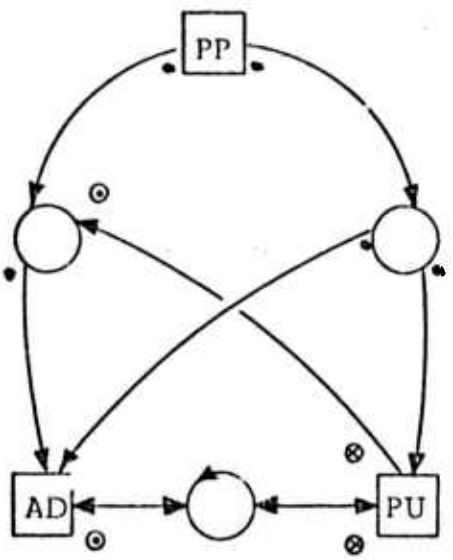


Figure A11: Fragment of Figure A2 to show effect of marking class

If tokens are placed on the positions marked • and ⊙ , then the firing rule produces the cyclic order of firing of programs: PP, AD, PU and back to PP. If instead the tokens are placed on the positions marked • and ⊗ , then the other cyclic order is produced: PP, PU, AD and back to PP. Hence, a system is described not by a net alone, but by a net in combination with a marking class. For that combination we use the expression marked net. (We surmise that the Dayton System is intended to operate according to the second cyclic order.)

Scrutiny of Figure A2 shows there is no way to mark it so that a reasonable pattern of operation ensues. Re-examination of the documentation shows that FP is actually two disjoint parts, FP.910 which needs to run just prior to PP, and FP-910 which needs to run after PP. In Figure A12 we show this, along with a little disection of AD, PP and TR which will be discussed shortly.

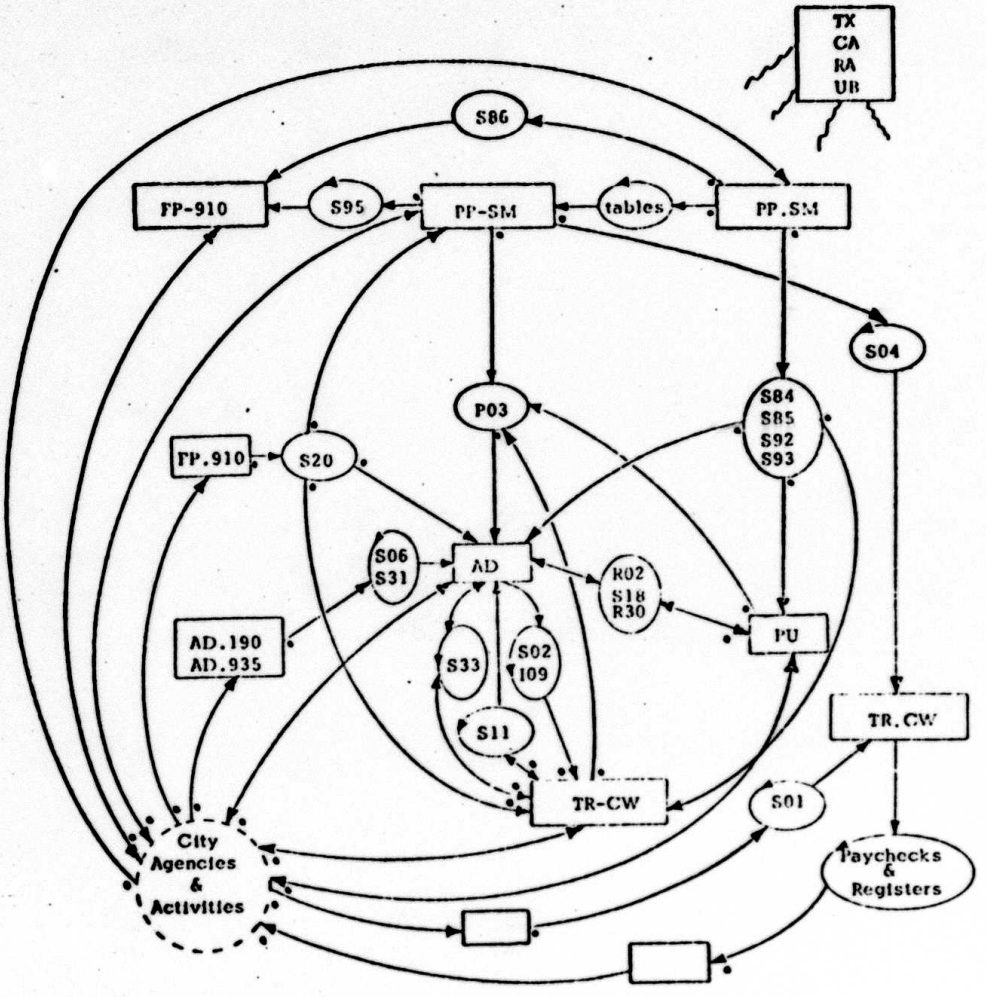


Figure A12: Expansion of Figure A2 with initial marking

For lack of detailed knowledge about how the Dayton System is tied into the City of Dayton, we choose the firing rule for the "City Agencies and Activities" of Figure A12 to be that normally associated with a program. The first move from the initial marking shown is thus to radiate out all the tokens shown around the dotted circle. In this special case, the implied ordering can also be shown by a PERT chart with a feedback loop. In general, the marked graph has the advantage of handling cycles of actions where actions from one cycle may overlap those of another.

Figure A12 does not display choice but, as we mentioned earlier, the idea of coloring the tokens and adjusting the firing rule to include changing colors brings choice into the picture. For example, not every day is payday. With colored tokens we can let black coming out of TR.CW represent paychecks and white represent the effect of an opportunity that was <u>not</u> taken with respect to paycheck generation.

It is instructive to map Figure A12 to a grosser picture. The interesting feature is that if we want to preserve a token game we can go to a much grosser picture than A2 but we cannot sensibly merge FP.910 with FP-910. Figure A13 shows what might be called the grossest level of mapping that is still instructive with respect to synchronization.

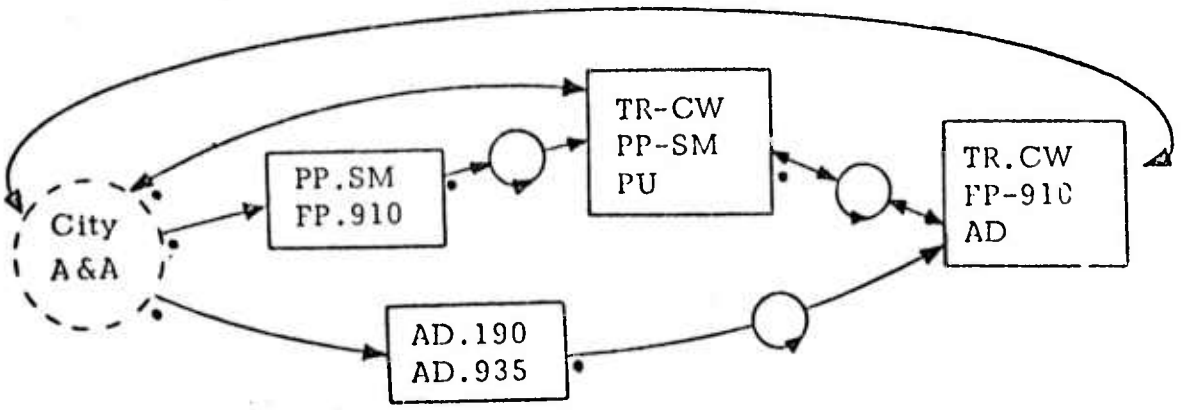
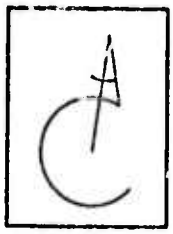


Figure A13: Grosser map of Figure A12

It should be noted that the operation implied by Figure A13 is more constrained than that implied by Figure A12. When one needs to see the maximum freedom available in synchronizing a system, one needs to go to the finest level of representation. It appears from Figure A13, and the separation of FP into FP.910 and FP-910 that one thing Westinghouse did not have in mind when deciding what properly made up a component, was synchronization.



MASSACHUSETTS
COMPUTER ASSOCIATES, INC.
26 PRINCESS STREET, WAKEFIELD, MASS. 01880 • 617/245-9540

The Connection of Parts to One Another in System Contexts

by

Anatol W. Holt

March 14, 1975

A SUBSIDIARY OF APPLIED DATA RESEARCH, INC.

The Connection of Parts to One Another in System Contexts

A. The Relation of Influence

We are dealing with systems of interacting parts. We assume that the parts and their relations of interaction are the system. Thus, so long as the system persists, so do its parts.

A1. We assume that at any (system) time it may be validly asserted of each of its parts that it is either in some one state, or that it is in transition from one state to another.

A2. We assume that any given behavior of a part in its system context is completely characterized by a sequence of state transitions, and that all of its behavioral possibilities may be characterized by a state transition graph.

A3. We assume that the effect of an interaction between two-or-more parts is a change-of-state of each of these parts. Thus, an interaction consists in a set of state transitions. Two transitions that belong to the same interaction are said to be coincident.

A4. We assume that every state transition belongs to one and only one interaction between two-or-more parts.[1]

---

[1] This assumption may be usefully weakened to: every state transition belongs to one or more interactions between two or more parts.

Thus it is clear that interactions can be modeled as Petri net events with an equal number of states into and out of the event - a ready-state and a done-state for each partner in the interaction. If e is such an event, then $^\bullet e$ is the set of ready-states, and $e^\bullet$ is the set of done-states of the interaction. Given a state s of a part P we call $s^\bullet$ the set of next-possible transitions and $^\bullet s$ the set of last-possible transitions for the part P in state s. Because P is represented by a state transition graph, assertions of the form 't is next-possible for P' or 't is last-possible for P' - specify the state of P. Since every transition is part of one and only one event, we can also talk of next-possible events instead of next-possible transitions without confusion.

We will now consider what can be deduced about the taking place of events from the holding of states.

Assumption A4 has the following consequence:

A5. The fact that an event is next-possible for part P can never guarantee that it takes place. Since, in the event, at least two partners interact, it must also be next-possible for at least one other part.

The following two examples help to show how this is meant.

A6. .1 That my state change as a result of getting a message out of a mailbox depends not only on my readiness to get it, but also on the presence of the message in the mailbox at the time that I attempt to get it. If I do get the message, both my state and the state of the mailbox will change.

.2 That a clock pointer moves from 12 to 1 depends not only on its being at 12 beforehand, but on the spring being sufficiently wound. The taking place of the event changes both the state of the pointer and the state of the spring.

A7. We may now ask: if the transition t, and hence event e, is next possible for part A, what can make it certain that e does not take place? It can only become certain when another partner, B, in the event e enters a state which precludes its entering the ready-state for e while ℓ is still next-possible for A. Under these circumstances we say that the state of B excludes the transition t of A. We also say: the transition t of A is excluded if t is next-possible for A and there exists a part whose state excludes the transition.

We will presently discuss under what structural circumstances it can be known that the state of one part excludes the taking place of a transition in another part - and we will especially examine the question in relation to Petri nets. What emerges from the discussion thus far is this: while the state of a part can never guarantee that an event take place, it can guarantee that an event not take place. Both halves of this statement derive from the fundamental fact that the taking place of every event requires the cooperation of at least two partners. (It takes two to tango; it takes only one to say no.)

A8. We may now ask: if the transition t is next possible for part A, what can make it certain that t takes place? Only when it is certain that:

.1 The event to which t belongs is next-possible for every one of its partners.

.2 All other next-possible transitions of all partners in the event are excluded.

We can now consider how to define the exertion of influence of one part on the behavior of another. If the behavior of a part in some state is to be subject to influence the state in question must lead to one of several transitions - i.e., $|s^\bullet| > 1$. Only in the choice between alternate transitions can behavioral differences

be expressed, given that behaviors are characterized by state transition sequences (as per A2). In particular, behavioral differences can not be expressed by the longer or shorter duration of the state of a part. (Much of the difficulty of our subject as well as much of its potential power in application derives from this restriction.)

A9. Definition: We say part B prevents the transition t of part A under exactly the following circumstances

.1 A and B are partners in the event e to which t belongs.

.2 The event e is next-possible for all of its partners except B.

.3 The state of B excludes the event e (as per A7).

A10. Definition: We say B influences the transition t of A under exactly the following circumstances

.1 It is certain that the event e to which t belongs takes place (as per A8).

.2 There is a next-possible transition t' of A which B prevents (as per A9).

Less formally, we can express the idea as follows: one part influences the behavior of another on a particular occasion if it is responsible for restricting the choice of next possible state transitions of the influenced part. Now given that a part engages in any next interaction it is clear that there must be some source of restriction as to its alternate choices. Since the part can only engage in one state transition at a time (see A1) the taking place of a particular transition means the exclusion of the other possible transitions. In that sense, the disabling of transitions is necessary to the enabling of transitions - namely, the enabling of the next-possible transitions that are not disabled. (The lack of disabling is illustrated by Buridan's Ass who while hungry stands at equal distance between two equally attractive bundles of hay and ... starves to death.)

B. Steps Towards Petri-net Models of Influencing

It is a prime objective of a theory of systems, to make possible the tracing of influence from part to part - in other words to calculate whether a change in behavior of one part results in a change of behavior of some other part - possibly distant from the first - and if so, to characterize the change of behavior of the affected part. Furthermore, it is common for the relations of influence to be mutual and for this reason difficult to characterize. We will now describe some steps towards the building of Petri net models in which relations of influence are subject to calculation.

B1. Complete and Incomplete System

We may model an interaction between system partners with the idea that not all the partners in the interaction lie within the system being modeled. We call such an interaction incomplete within the system. If modeled as a Petri-net event, such events must be formally designated as incomplete, for without such designation the possible behaviors of the system cannot be properly characterized. A particularly easy and useful method of designation is to associate with each event e an integer n which specifies how many partners interact in it. Then, if $|{}^\bullet e| = |e^\bullet| < n$, we know that e is incomplete within the system, and we also know how many partners are missing. Given a Petri-net for which it is assumed that the number of partners in each event is the same, say n, we can distinguish complete from incomplete events by comparing $|{}^\bullet e|$ or $|e^\bullet|$ with n; if it is less than n, the event is incomplete, and otherwise complete. For Petri-nets in which there is no such assumption of uniformity, we will adopt the following graphic convention:

.1  complete event

.2  incomplete event

We call a net complete if all of its events are complete. A complete net represents a system which cannot influence or be influenced by environment. Thus, the majority of systems that are practically interesting are incomplete. However, the concept of a complete - or isolated - system, plays a

fundamental role in the development of the theory, just as it does in physics. That is the proper conceptual context in which to consider the assumptions about the propagation of influence, and its relation to part behavior.

MASSACHUSETTS
COMPUTER ASSOCIATES, INC.
26 PRINCESS STREET, WAKEFIELD, MASS. 01880 • 617/245-0540

Definitions Pertaining to Petri Nets, States and Events and Behavior

by

Anatol W. Holt

March 14, 1975

A SUBSIDIARY OF APPLIED DATA RESEARCH, INC.

Definitions Pertaining to Petri Nets, States and Events and Behavior

These pages introduce some useful basic concepts related to Petri Nets, and especially with respect to those Petri Nets that can be regarded as models of interacting parts.

A. Petri Net Definitions and Notations

A1. A Petri Net is an ordered triple $\langle P, T, \alpha \rangle$ where

P is a set of places

T is a set of transitions

α is a relation "leads to", $\alpha \subseteq P \times T \cup T \times P$

It is also assumed that $P \cap T = \emptyset$.

A2. A place marking of a Petri Net is a function of the places of P of a net into the non-negative integers N.

A3. Given a Petri Net element $x \in P \cup T$, we define $x^{\cdot} \triangleq \{y: x\alpha y\}$ and ${}^{\cdot}x \triangleq \{y: y\alpha x\}$ - in other words, all the elements which x leads to and all the elements which lead to x.

A4. Given an arbitrary subset of places A of the places P of a net, we define $\hat{A}$ to be the marking which assigns 1 to every place of A and 0 to all other places (the characteristic function of A).

A5. Let m and m' and t be two markings and a transition of a Petri Net. We define $m[t\rangle m'$ to mean

.1 $m \geq {}^{\bullet}\hat{t}$

and .2 $m' = m - {}^{\bullet}\hat{t} + \hat{t}^{\bullet}$

and we say <u>m leads to m' by firing t forwards.</u> Similarly we define $m'\langle t]m$ to mean

.3 $m \geq \hat{t}^{\bullet}$

and .4 $m' = m - \hat{t}^{\bullet} + {}^{\bullet}\hat{t}$

and we say <u>m leads to m' by firing t backwards.</u> We notice that $m[t\rangle m' \iff m\langle t]m'$. This depends on the fact that for all m, $m \geq 0$.

A6. From this we get the following auxiliary definitions

.1 $m[t\rangle \triangleq (\exists m')\ (m[t\rangle m')$ t is firable at m, forwards

.2 $m[\cdot\rangle m' \triangleq (\exists t)\ (m[t\rangle m')$ m leads to m' in one step forwards

Each of these definitions and each of the definitions in A7 and A8 has an exactly corresponding backwards equivalent.

A7. Analogous to A5-6, we have for a sequence of transitions
$\sigma = \langle t_1, t_2, t_3 \ldots\ldots t_n \rangle$

.1 $m[\sigma\rangle m' \triangleq \exists$ a sequence of markings $\langle m_1, m_2, \ldots\ldots m_{n-1}\rangle$ such that $m[t_1\rangle m_1$ and $m_i[t_{i+1}\rangle m_{i+1}$ and $m_{n-1}[t_n\rangle m'$

.2 $m[\sigma\rangle \triangleq (\exists m')\ (m[\sigma\rangle m')$

.3 $m[-\rangle m' \triangleq (\exists \sigma)\ (m[\sigma\rangle m')$

A8. .1 $m[{}^{t}_{t'}\rangle m' \triangleq m[t,t'\rangle m'$ and $m[t',t\rangle m'$

.2 $m[{}^{t}_{t'}\rangle \triangleq (\exists m')\ (m[{}^{t}_{t'}\rangle m')$ — t and t' are forwards concurrent at m

.3 $m[{}^{t}_{t'} \triangleq m[t\rangle$ and $m[t'\rangle$ — t and t' are forwards co-possible at m

.4 $m[{}^{t-}_{t'-} \triangleq m[{}^{t}_{t'}$ and $\sim m[{}^{t}_{t'}\rangle$ — t and t' are forwards alternative at m

A9.

.1 The marking graph of a Petri Net $\langle P, T, \alpha\rangle$ is the graph of the relation $m[\cdot\rangle m'$.

.2 The maximal connected subgraphs of the marking graph are called the marking classes of the net.

.3 A marking class M is called <u>bounded</u> if there exists a marking b such that $(\forall m \in M)\ (m \leq b)$; it is called <u>safe</u> if $(\forall m \in M)\ (m \leq \hat{P})$.

.4 A marking class M is called <u>live</u> if: (a) the marking class is strongly connected.

.5 A marked Petri Net is a pair $\langle N, M \rangle$; where N is a Petri Net and M is a marking class of N . Since the marking classes partition the markings a marked Petri Net can be specified by specifying the net and a marking of the net. Thus $\langle N, m \rangle$ where m is a marking, specifies a marked net. The places and transitions of a marked net $\langle N, M \rangle$ are, of course, those of N .

A10. <u>Subnets</u>

.1 Given two nets, $N = \langle P, T, \alpha \rangle$ and $N' = \langle P', T', \alpha' \rangle$, we say that $N' \subseteq N$ if $P \subseteq P'$ and $T \subseteq T'$ and $\alpha \subseteq \alpha'$. Thus, the subnets of a net form a boolean algebra and unions, and intersections are defined.

A11. Given a net $N = \langle P, T, \alpha \rangle$ and a subnet $X \subseteq P \cup T$ of its elements, we define a subnet of N as a function of X , $\boxed{X} = \langle P', T', \alpha' \rangle$ thus:

.1 $P' = X \cap P$

.2 $T' = (X \cap T) \cup \{ t \in T: (\exists p \in P')\ (p(\alpha \cup \overleftarrow{\alpha})\ t) \}$

.3 $\alpha' = \alpha \restriction P' \times T' \cup T' \times P'$

Similarly we define $\textcircled{X} = \langle P'', T'', \alpha'' \rangle$ thus:

.4 $\quad T'' = X \cap T$

.5 $\quad P'' = X \cap P \cup \{p \in P: (\exists t \in T'')\ (p(\alpha \cup \overleftarrow{\alpha})t)\}$

.6 $\quad \alpha'' = \alpha \restriction P'' \times T'' \cup T'' \times P''$

A12. Place and Transition Components

Consider a Petri-net $N = \langle P, T, \alpha \rangle$ and a subset of its places P' such that:

.1 $\quad (\forall t \in T)\ (|t^{\bullet} \cap P'| = |{}^{\bullet}t \cap P'| \leq 1)$

We then call the subnet $\boxed{P'}$ a place component. Similarly, given a subset $T' \subseteq T$ such that:

.2 $\quad (\forall p \in P)\ (|p^{\bullet} \cap T'| = |{}^{\bullet}p \cap T'| \leq 1)$

we call $\textcircled{T'}$ a transition component of the net.

A place component decomposition of a Petri-net $N = \langle P, T, \alpha \rangle$ is a set of place components which partition the states of the net. An event component decomposition is a set of event components which partition the events of the net.

A13. Let f and g be two arbitrary functions which map the places of Petri Nets into the integers. We now define

$$f \circ g \triangleq \sum_{p \in P} f(p)\ g(p) \qquad \text{(inner product of the two functions)}$$

A14. Let $\langle N, M \rangle$ be a marked Petri Net, and P' the places of a place component. Then:

$$(\forall m, m' \in M)\ (\hat{P}' \circ (m-m') = 0)$$

A15. Therefore, we know, if N has a place component decomposition then all of its marking classes are bounded.

In particular, given a marking m such that $m \circ P_i \leq 1$ for each of the blocks of places P_i of its place components, we know that the marking class to which m belongs is safe.

A16. State/Event Nets

Let $\langle N, M \rangle$ be a marked Petri Net with a place component decomposition $\{N_i\}$, each with places P_i, and $(\forall m \in M)\ (\hat{P}_i \circ m = 1)$. Such a marked net is subject to the following interpretations:

.1 Each place component represents a part of a system. The Petri Net places may now be interpreted as the possible states of the parts, and the transitions of the place-component as the state transitions of the part. Thus the behavioral possibilities of the part are represented as a state transition diagram, exactly in the sense in which that term has been used in the theory of Markov processes, or in automata theory.

2 The parts are linked to one another through shared transitions. The transition, seen in the context of the total Petri Net (rather than in the context of a single part) may now be interpreted as an event in which some number of parts interact. The effect of the event on each of the participating parts is a state transition.

.3 A marking of the net may now be interpreted as a total system state - technically, to be called a case. The fact that for every marking m and every block P_i, $m \circ \hat{P}_i = 1$ is interpreted as meaning that in every case, every part is in one and only one state.

.4 The relation $m[t\rangle m'$ now means: case m leads to case m' by an occurrence of the event t. The only parts whose states can be different in m' than in m are those that interacted in the event.

Nets that are subject to these interpretations will be called state/event nets and symbolized by $\langle S, E, \alpha \rangle$ instead of $\langle P, T, \alpha \rangle$. In such nets, we will talk about state components and event components instead of place components and transition components.

B. System Behavior

We can represent a possible behavior of a system by a marking m and a sequence of events σ such that $m[\sigma\rangle$. We will call such pairs $\langle m, \sigma\rangle$ firing sequences. Under certain circumstances, we will wish to say that two distinct firing sequences represent the same system behavior. There are two initially independent sources for such a firing sequence equivalence.

B1. Part-by-Part Equivalence

A firing sequence $\langle m, \sigma\rangle$ of a state/event net induces, in a natural manner, firing sequences on each of its parts, π_i - namely, the firing sequence $\langle m_i, \sigma_i\rangle$ where m_i is just the state π_i which holds in m and σ_i is the subsequence of events of σ which are state transitions of π_i. We now define:

.1 The firing sequences $\langle m, \sigma\rangle$ and $\langle m', \sigma'\rangle$ are part-by-part equivalent iff for each part π_i, $\langle m_i, \sigma_i\rangle = \langle m'_i, \sigma'_i\rangle$.

From this follows at once: if $\langle m, \sigma\rangle$ and $\langle m', \sigma'\rangle$ are part-by-part equivalent then $m = m'$.

B2. Concurrency Equivalence

Two firing sequences that differ only by the relative ordering of concurrently possible firings should represent the same system behavior.

Let $\langle m, \sigma\rangle$ and $\langle m', \sigma'\rangle$ be two firing sequences with the following properties:

.1 $m = m'$

.2 There exists an integer i such that

$$\sigma = \langle e_1, e_2, e_3 \ldots\ldots e_i, e_{i+1}, \ldots\ldots e_n \rangle$$

and $$\sigma' = \langle e_1, e_2, e_3 \ldots\ldots e_{i+1}, e_i \ldots\ldots e_n \rangle$$

.3 $m[e_1, e_2, \ldots\ldots e_{i-1}\rangle m'$

.4 $m'\left[\begin{matrix} e_i \\ e_{i+1} \end{matrix}\right\rangle$

Then we call the two sequences concurrency neighbors. We call the transitive closure of this relation concurrency equivalence between firing sequences.

We now ask: under what circumstances are concurrency equivalence and part-by-part equivalence the same relation?

B3. If $\langle m, \sigma \rangle$ and $\langle m', \sigma' \rangle$ are part-by-part equivalent, then they are concurrency equivalent.

Proof: Let $\sigma = \langle e_1, e_2 \ldots\ldots e_n \rangle$ and $\sigma' = \langle e'_1, e'_2 \ldots\ldots e'_{n'} \rangle$. Part-by-part equivalence means that the number of occurrences of each event in σ must be the same as in σ'. Therefore, σ and σ' are permutations of one another, and $n = n'$.

We will now show that if σ and σ' agree in the first i-1 places, $1 \leq i \leq n$, then there exists a firing sequence $\langle m, \rho \rangle$ which is concurrency equivalent to $\langle m \ \sigma' \rangle$ and ρ agrees with σ in the first i places.

Let i be the least index such that $e_i \neq e'_i$. If π is a part for which e_i is the kth transition in sequence σ then e_i must also be the kth transition of π in σ'. Therefore, there must exist an index j such that

.1 $e'_j = e_i$

.2 $j > i$

.3 For all h, $i \leq h \leq j$, e_h' is not a transition for π

Thus, none of the parts for which e'_j is a transition have any of the events e'_h as transitions. Since any two distinct parts are state-disjoint, it follows that for every event e'_h, $\dot{e}_h' \cap \dot{e}_j' = \dot{e}_h' \cap e_j'^{\cdot} = \emptyset$ and $e_h'^{\cdot} \cap \dot{e}'_j = e_h'^{\cdot} \cap e_j'^{\cdot} = \emptyset$.

Therefore, if $m[e'_1, e'_2 \ldots\ldots e'_j\rangle m'$ and $m''[e_{j-1}, e_j\rangle m'$ then

$$m'' \left[\begin{matrix} e_{j-1} \\ e_j \end{matrix} \right\rangle .$$

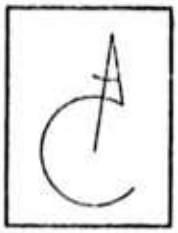
Thus we can construct a concurrency neighbor of $\langle m, \sigma' \rangle$ - call it $\langle m, \sigma'' \rangle$ where σ'' and σ' differ from one another by the interchange e'_j and e'_{j-1}. If, $j-1 > i$, we can repeat the procedure - i.e.,

construct a concurrency neighbor of $\langle m, \sigma'' \rangle$ - call it $\langle m, \sigma''' \rangle$ where σ'' and σ''' differ from one another by the interchange of e''_{j-2} and e''_{j-1}, and etc., until we have obtained a concurrency equivalent of $\langle m, \sigma' \rangle$ - call it $\langle m, \rho \rangle$ - where the ρ and σ agree in their first i positions.

B4. We will call a net $\langle P, T, \alpha \rangle$ <u>pure</u> if $(\forall p \in P)\ (^\bullet p \cap p^\bullet = \emptyset)$.

B5. Having in B3 established that part-by-part equivalence implies concurrency equivalence, we now ask if the implication also goes the other way. The answer is: it does, if the state/event net is pure.

To see this, we need only convince ourselves that, if two firing sequences are concurrency neighbors, they must be part-by-part equivalent. Suppose, therefore, that for the net $\langle S, E, \alpha \rangle$ with marking m, $(\exists e_1, e_2 \in E)\ (m[\begin{smallmatrix} e_1 \\ e_2 \end{smallmatrix}\rangle)$, and we ask if the firing sequences $\langle m, \langle e_1, e_2 \rangle \rangle$ and $\langle m, \langle e_2, e_1 \rangle \rangle$ are part-by-part equivalent. If $^\bullet e_1$ and $^\bullet e_2$ are disjoint then the two sequences are certainly part-by-part equivalent. On the other hand, suppose there exists an $(s \in S)\ (s \in {}^\bullet e_1 \cap {}^\bullet e_2)$. It is clear that s must hold in m and that it must hold after e_1 fires, and that it must hold after e_2 fires. This implies that $s \in {}^\bullet e_1 \cap e_2^\bullet$ and $s \in {}^\bullet e_2 \cap e_2^\bullet$, a two-fold violation of net purity.



MASSACHUSETTS
COMPUTER ASSOCIATES, INC.
20 PRINCESS STREET, WAKEFIELD, MASS. 01880 • 617/245-9540

Calculating Logical and Probabilistic Dependencies in a Class of Net Models

by

Robert M. Shapiro

Meta Information Applications, Inc.

March 14, 1975

SUMMARY

This paper describes a method for generating a set of affine recurrence relations from a class of role-activity nets. Since this class of role-activity nets is powerful enough to model interesting problems (involving both concurrency and choice), the method provides a link between nets as descriptive apparatus and existing mathematical techniques which can yield useful nontrivial results pertaining to the net models.

In the course of the paper we will define this class of role-activity nets; relate these nets to Petri nets and Marked Graphs by the use of net mappings; describe and illustrate how to produce a set of affine recurrence relations from these nets; and calculate dependency properties of the nets from these recurrence relations.

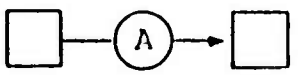
ACKNOWLEDGMENT

Michael Karr made a critical contribution to the applied mathematics in this paper. The work of Anatol Holt and Project ISTP provided the basis for the ideas described here.

THE CLASS OF ROLE-ACTIVITY NETS

We will now explain, in an informal manner, what special class of role-activity nets we will employ. These nets consist of roles (represented by circles) and activities (represented by boxes), interconnected in a way to be described, and with a set of rules, one per role, which define the effect of the activities by the use of boolean equations.

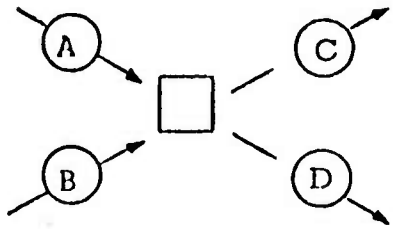
Each role participates in exactly two activities: this is represented by arcs between the role and each activity. Exactly one of these arcs is directed, pointing at the activity and away from the role. This suggests the direction of flow of influence in the net.

The simplest net is: 

Each role simply alternates between the two activities it participates in. When a role participates in the activity to which it points, it transmits the value it last received as a result of participating in its alternate.

An activity takes place when all participants are on hand; some of them transmit values, the others receive new ones. There is no restriction on the number of participants in an activity.

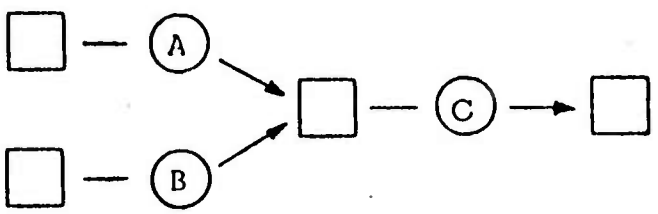
Example of an Activity:



All of the roles that are connected to the activity by undirected arcs receive new values when the activity takes place. These new values are determined by boolean expressions, one per role receiving a new value. Each expression is a boolean combination of the values transmitted to the activity by the roles that point to it.

Thus, in the above example of an activity: when roles A, B, C and D are all prepared to participate, the activity takes place. Using the role name also as a boolean variable which stands for the role's value, the effect of the activity is to set $C = e_1$ and $D = e_2$ where e_1 and e_2 are boolean expressions in A and B. After the activity has taken place, roles A and B must next participate in their respective alternate activities; when these occur they will receive new values. Roles C and D must next participate in their respective alternate activities; the values they have just received will then be transmitted to define new values for other roles.

An activity which has no roles pointing at it is an input activity. All roles which participate in it receive new values when the activity takes place. These new values are defined by boolean expressions on input variables. An output activity is such that all roles participating in it point to it.



In the above net, both A and B receive values from input activiti -. Thus the boolean expressions for A and B must be in terms of input variables. The value of C is defined by a boolean expression in A and B. This value is provided to an output activity.

NET MAPPINGS

These role-activity nets always have a live-marking. (By this we mean simply that there is at least one starting arrangement for roles such that all the activities can recur indefinitely many times.) This can be easily understood by reference to Figure 1 which illustrates the mapping from these structures to Marked Graphs. It is simple to demonstrate that the Marked Graph obtained by the mapping is always coverable by a set of basic circuits. A theorem from marked graph mathematics allows us to conclude that the marked graph has at least one live safe marking class. The mapping preserves activities and splits each role into two roles, thus forming a basic circuit for each of the roles in the original structure.

It is also possible to demonstrate via a mapping the existence of a live safe Petri net branched at both states and events for every structure in this class of role-activity nets. The mapping for this is illustrated in Figure 2. Here, each role is split into three roles which form a state component. Two of these roles represent the two possible values that a role can transmit to an activity. The third represents availability for participation in the activity that redefines the value of the role. Unlike the mapping to Marked Graphs, which maps activities one-one, in this mapping each activity is split into 2^n activities, where n equals the number of roles that transmit to it; i.e., the number of roles in the original structure that point at the activity.

In addition, each <u>input</u> activity is split into 2^n activities, where n equals the number of roles that participate in it.

The connections between roles and activities in the Petri net are dictated by the boolean expressions. In the resulting Petri net, all conflicts involve only input activities; the rest of the net is guaranteed conflict free.

THE BOOLEAN EQUATIONS FOR THE NET

We will now show how to generate a set of boolean equations for role-activity nets of the class we are considering. These equations depend not only on the net and the transformation rules associated with the activities, but also on the initial marking of the net, which must be chosen to yield a live safe marking class.

As in common practice, we will use subscripts on variables to represent their change with time. To set the equations up we will adopt the following convention:

> The initial value of a role variable will be subscripted '0' if, in the initial marking, the role is ready to transmit; otherwise, the subscript '1' will be used. All input variables will be subscripted '0'.

This procedure permits us to re-express the equation for any variable completely in terms of variables with a '0' subscript. These equations can be viewed as a set of boolean recurrence equations by replacing all '1' subscripts with 'i+1' and all '0' subscripts by 'i'.

The following example illustrates the procedure.

AN EXAMPLE

Persons X and Y wish to communicate with person Z. Z, however, can only receive one message at a time. Thus we introduce a communication discipline which guarantees that only one message will arrive at Z at one time. To do this, we grant X's message priority: if X is sending a message, Y's message, if any, will be blocked. Further, we provide a buffer for storing Y's message if it is blocked. If Y attempts to send a new message while an old one still resides in the buffer, a message will be lost.

Refer to System 1 (Figure 3) for the net and accompanying boolean equations that describe this situation. The role G transmits the blocking signal that causes Y's message (if any) to be diverted to the buffer H. B, C, E and F are message transmitters. As a starting condition, we choose H_0 to represent the state of the message buffer ($H_0 = 0$ means the buffer is empty). Input events determine whether X and Y wish to send messages, thus the equations for X and Y are expressed in terms of input variables.

AFFINE RECURRENCE RELATIONS

Suppose, now, that X and Y try to send messages with frequencies defined by the independent fixed probabilities α and β, respectively. In such a situation we might like to know what percentage of Y's messages are lost.

To do this we use the boolean equations for the net to generate a linear recurrence equation which expresses the probability (h) that a message is in buffer H. (Further in the text we will clarify the interpretation of this probability.) Once we have this, we can calculate the fraction of lost messages by multiplying the above probability by β.

Referring to Figure 3 once again, we express H_1 in terms of '0' subscripted variables. This permits us to express the probability $\Pr\{H_1\}$ in terms of input probabilities and the probability of the buffer being occupied initially. By replacing the subscript '1' with 'i+1' and '0' with 'i' and taking advantage of the fact that α and β are independent fixed probabilities, we obtain a recurrence equation for h. This can be solved by standard procedures for such equations. The solution then lets us calculate the fraction of lost messages, which depends only on the frequencies at which X and Y try to send messages (and not, for instance, on the probability h_0).

THE METHOD

In general, the method for generating the recurrence relations is as follows:

1. The variable of interest is expressed in terms of variables with '0' subscripts.

2. This boolean equation is transformed into an equation in probabilities.

3. If all of the probabilities are expressed in terms of input variables and the variable of interest, and the inputs are all independent fixed probabilities, the equation can then be solved by the standard methods appropriate for recurrence equations.

4. It may be that some of the probabilities involve logical expressions in terms of system variables other than the initial variable of interest. (System 3, Figure 5, to be discussed shortly, illustrates this possibility.) In this case, a boolean equation is generated from the original set of equations for each such logical expression. Each new equation is converted into an equation in probabilities. It can be proved that this process terminates, and that the resulting set of probabilistic equations is a set of affine recurrence equations. (We offer no proofs in this paper.)

   Note: To interpret the equations as recurrence equations, it is necessary to substitute subscript 'i+1' for '1', and 'i' for '0'.

MORE EXAMPLES

In System 2 (Figure 4) we make a minor alteration in System 1. Instead of granting X absolute priority for message sending, we insist that Y should not lose his turn two times in a row. To do this, we add a single role to the structure. I carries a signal which indicates whether Y's message was blocked last time around. If so, X's message, if any, is destroyed this time.

The resulting recurrence system illustrates a phenomenon that does not occur in System 1. For a certain set of input probabilities ($\alpha = \beta = 1$, i.e., X and Y both always try to send messages) the solution for the probability that the buffer has a message in it turns out to be periodic. In this case, the answer obtained by solving the recurrence relation for h under the assumption that h_i converges for sufficiently large i (i.e., that for any $e > 0$ there exists an n such that $h_{n+1} - h_n < e$) yields the average value for h. A complete discussion of periodic solutions is beyond the scope of this paper. Refer to the accompanying paper by Michael Karr.

In System 3 (Figure 5) we make a minor alteration to System 2. Instead of simply destroying X's message in case I indicates that Y was blocked last time around, we provide one more role to act as a buffer for X, analogous to how H acts for Y. Thus X will lose messages only if X tries to send a message when his buffer already has a message in it.

This net is interesting because it appears to be symmetric in respect to X and Y, yet an inspection of the boolean equations shows that they are not. This models our communication discipline, which started off favoring X and still does. Thus it turns out from the solution to the system of recurrence relations, that X has higher priority than Y except for the case where X attempts to send a message every time around. The solution involves invoking step four of the method, which was not required for System 1 or System 2.

RELATIONSHIP TO MARKOV CHAINS

It is also possible to generate a stochastic matrix from the role-activity nets we have discussed. This permits the application of Markov chain theory to the analysis of the behavior of these nets.

To generate the stochastic matrix, it is first necessary to calculate the number of states in the Markov process. This is simply 2^n where n equals the number of roles in the net. In other words, each state in the Markov process is one particular set of values for all the boolean variables in the system.

The stochastic matrix can be calculated by considering for each state, the transition probabilities to every state. The boolean equations for the role-activity net enable us to calculate these transition probabilities. This is so because a particular state defines the value of every boolean variable; thus we can calculate the next value for each boolean variable, to within the influence of inputs whose probabilities are assumed fixed and independent. We can then calculate the transition probabilities themselves.

This method tends to produce large stochastic matrices even for small problems. It is frequently possible to then simplify these matrices. However, the approach we have taken, that is the generation of recurrence relations, is a faster route to useful calculations for the nets we have examined. Recurrence relations also are better suited to the semantic intentions of the net models, some of which we discuss next. The same mathematical results may be obtained by either method.

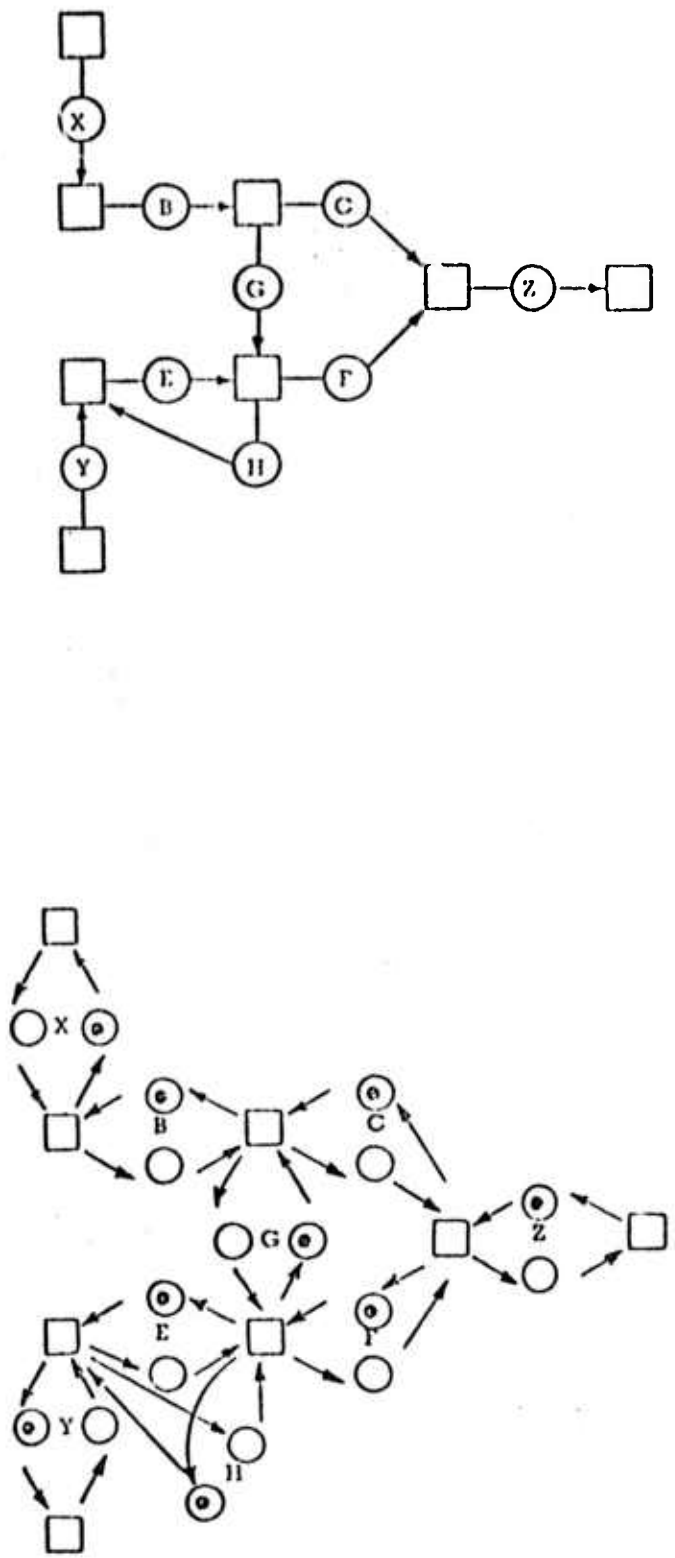
MULTICLOCK SYSTEMS

The boolean equations for a role-activity net, by virtue of the subscripting convention discussed previously, enable us to view the net as an n clock system, where n is equal to the number of activities in the net. In fact, we can think of a clock associated with each activity, which counts up each time that activity takes place. The subscript on a variable then reflects the number of times the activity which defines the value for that variable has taken place.

The calculations described earlier then have a simple interpretation. For any single variable the probability is equal to the percentage of clock ticks (on the clock associated with that variable's defining activity) for which the variable had value '1' . Equivalently, the inverse of the probability is the mean number of clock ticks between two occurrences of the value '1' (referred to as the mean recurrence time).

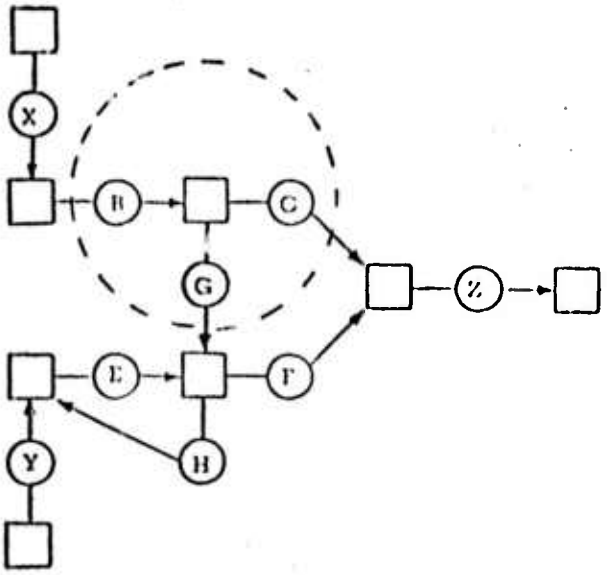
Step 4 of the method and System 3 (Figure 5) suggest that we can perform calculations on joint occurrences of several variables, or indeed on joint occurrences of several different subscript values for the same variable. Thus in System 3, we calculate the probability $\Pr\{H_i \cap J_i\}$, which turns out to be zero for all values of i . This means that H and J are never both equal to 1 for the same value of i on their clocks. We could also calculate $\Pr\{H_{i+1} \cap H_i\}$ which turns out to be $\equiv \emptyset$, or indeed $\Pr\{H_{i+2} \cap H_i\}$ which is in general not $\emptyset$. This does not mean that there is some 'system state' where H_{i+2} and H_i both occur. It does indicate the probability that H will be true at local time i and local time $i+2$.

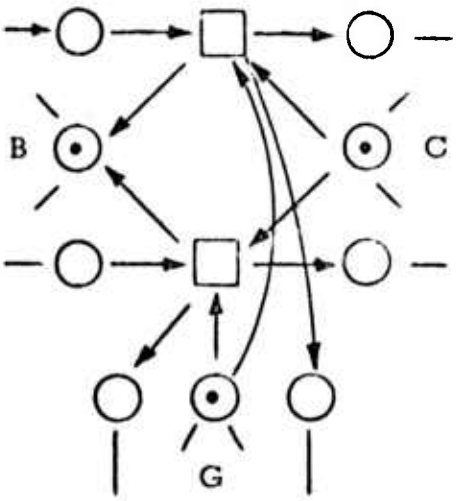
Figure 1



Role-activity net and Marked Graph, related by mapping with marking class for the Marked Graph corresponding to initial marking of Role-activity net as reflected in Boolean Equation system (see Figure 3).

Figure 2

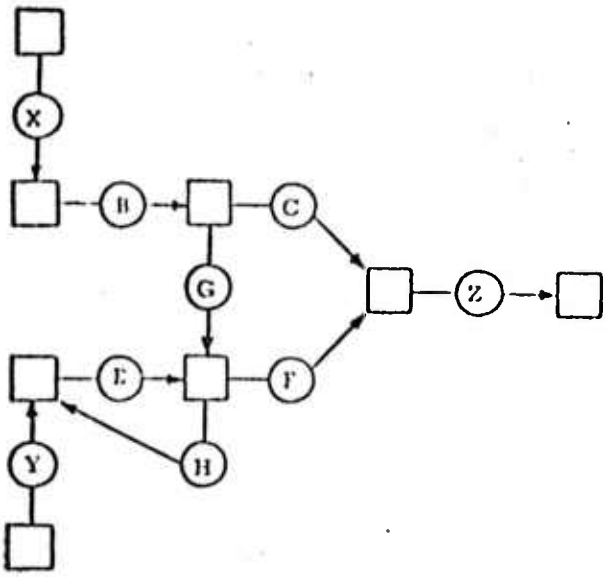




Role-activity net and Petri net Fragment related by mapping with marking for the Petri net corresponding to initial marking of Role-activity net as reflected in Boolean Equation system (see Figure 3).

System 1

Figure 3



$$X_1 = \alpha_0$$
$$B_1 = X_1$$
$$C_1 = B_1$$
$$G_1 = B_1$$
$$Y_1 = \beta_0$$
$$E_1 = H_0 \cup Y_1$$
$$F_1 = E_1 \cap \bar{G}_1$$
$$H_1 = E_1 \cap G_1$$
$$Z_1 = C_1 \cup F_1$$

$$H_1 = E_1 \cap G_1 = (H_0 \cup Y_1) \cap X_1 = (H_0 \cup \beta_0) \cap \alpha_0$$

$$\Pr\{H_1\} = \Pr\{(H_0 \cup \beta_0) \cap \alpha_0\} = \Pr\{(H_0 \cap \alpha_0) \cup (\alpha_0 \cap \beta_0)\}$$

$$= \Pr\{H_0 \cap \alpha_0\} + \Pr\{\alpha_0 \cap \beta_0\} - \Pr\{H_0 \cap \alpha_0 \cap \beta_0\}$$

$$= \Pr\{\alpha_0\} \cdot \Pr\{H_0\} + \Pr\{\alpha_0\} \cdot \Pr\{\beta_0\} - \Pr\{H_0\} \cdot \Pr\{\alpha_0\} \cdot \Pr\{\beta_0\}$$

Now let $h_i \equiv \Pr\{H_i\}$, $\alpha \equiv \Pr\{\alpha_i\}$, $\beta \equiv \Pr\{\beta_i\}$ for all $i \geq \emptyset$

We can then write the affine recurrence relation:

$$h_{i+1} = h_i \alpha + \alpha \beta - h_i \alpha \beta$$

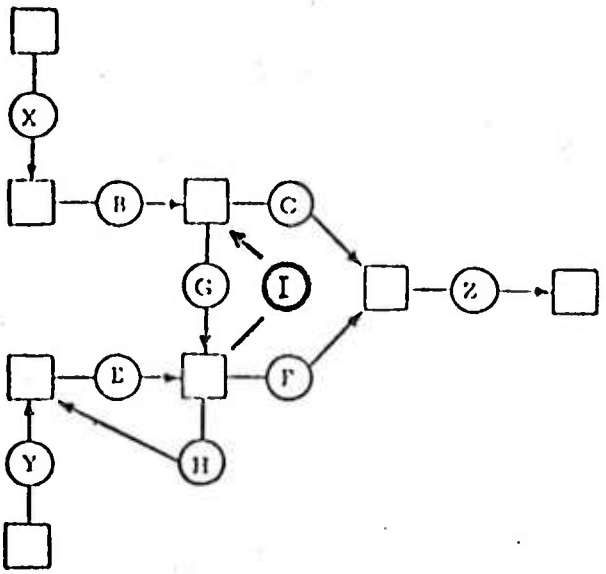
which has the cyclic solution of period 1 (steady state)

$$h = \frac{\alpha \beta}{1 - \alpha + \alpha \beta}$$

except for $\left\{ \begin{matrix} \alpha = 1 \\ \beta = \emptyset \end{matrix} \right\}$ when $h = h_0$

System 2

Figure 4



$$X_1 = \alpha_0$$
$$B_1 = X_1$$
$$C_1 = B_1 \cap \bar{I}_0$$
$$G_1 = B_1 \cap \bar{I}_0$$
$$Y_1 = \beta_0$$
$$E_1 = H_0 \cup Y_1$$
$$F_1 = E_1 \cap \bar{G}_1$$
$$H_1 = E_1 \cap G_1$$
$$I_1 = E_1 \cap G_1$$
$$Z_1 = C_1 \cup F_1$$

$H_1 = E_1 \cap G_1 = (H_0 \cup Y_1) \cap (B_1 \cap \bar{H}_0)$ (Note $I \equiv H$)

$= B_1 \cap \bar{H}_0 \cap \beta_0 = \alpha_0 \cap \beta_0 \cap \bar{H}_0$

$\Pr\{H_1\} = \Pr\{\alpha_0 \cap \beta_0 \cap \bar{H}_0\} = \Pr\{\alpha_0\} \cdot \Pr\{\beta_0\} \cdot \Pr\{\bar{H}_0\}$

Now let $h_i \equiv \Pr\{H_i\}$, $\alpha \equiv \Pr\{\alpha_i\}$, $\beta \equiv \Pr\{\beta_i\}$ for all $i \geq \emptyset$

Then:

$$h_{i+1} = \alpha\beta - \alpha\beta h_i$$

thus the cyclic solution of period 1 is

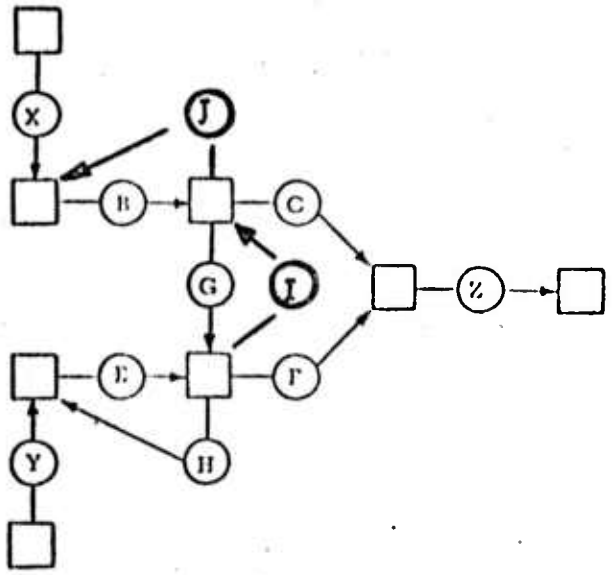
$$h = \frac{\alpha\beta}{1+\alpha\beta}$$

except for $\{\begin{matrix}\alpha = 1 \\ \beta = 1\end{matrix}\}$ when h is periodic with period 2.

i.e., $h_n = (-1)^n (h_0 - 1/2) + 1/2$

System 3

Figure 5



$$X_1 = \alpha_0$$
$$B_1 = J_0 \cup X_1$$
$$C_1 = B_1 \cap \bar{I}_0$$
$$G_1 = B_1 \cap \bar{I}_0$$
$$J_1 = B_1 \cap I_0$$
$$Y_1 = \beta_0$$
$$E_1 = H_0 \cap Y_1$$
$$F_1 = E_1 \cap \bar{G}_1$$
$$H_1 = E_1 \cap G_1$$
$$I_1 = E_1 \cap G_1$$

$H_1 = E_1 \cap G_1 = (H \cup Y_1) \cap (B_1 \cap \bar{H}_0)$ (Note $I \equiv H$)

$= (\bar{H}_0 \cap \beta_0 \cap J_0 \cup \bar{H}_0 \cap \alpha_0 \cap Y_1)$

$\Pr\{H_1\} = \Pr\{\bar{H}_0 \cap \beta_0 \cap J_0 \cup \bar{H}_0 \cap \alpha_0 \cap \beta_0\}$

$= \Pr\{\beta_0\} \cdot \Pr\{\bar{H}_0 \cap J_0\} + \Pr\{\alpha_0\} \cdot \Pr\{\beta_0\} \cdot \Pr\{\bar{H}_0\}$

$- \Pr\{\alpha_0\} \cdot \Pr\{\beta_0\} \Pr\{\bar{H}_0 \cap J_0\}$.

Let $h_i = \Pr\{H_i\}$ etc. Then

$h_1 = \beta \cdot \Pr\{\bar{H}_0 \cap J_0\} + \alpha\beta \cdot (1 - h_0) - \alpha\beta \cdot \Pr\{\bar{H}_0 \cap J_0\}$

$\bar{H}_1 \cap J_1 = (\overline{E_1 \cap G_1}) \cap I_0 = B_1 \cap H_0 = J_1$

$$J_1 = B_1 \cap I_0 = (X_1 \cup J_0) \cap H_0 = X_1 \cap H_0 \cup H_0 \cap J_0$$

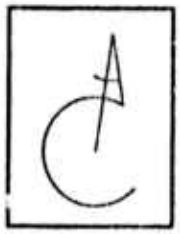
$$J_1 = \alpha h_i + \Pr \{H_0 \cap J_0\}$$

$$H_1 \cap J_1 = (E_1 \cap G_1) \cap (B_1 \cap I_0) = \emptyset$$

$$\therefore j_{i+1} = \alpha h_i \quad h_{i+1} = \beta j_i + \alpha\beta - \alpha\beta h_i - \alpha\beta j_i$$

Hence there is a cyclic solution of period 1

$$h = \frac{\alpha\beta}{\alpha^2\beta + 1} \qquad \text{except for } \begin{cases} x = 1 \\ y = 1 \end{cases} \text{ when } h \text{ has period } = 2 .$$



MASSACHUSETTS
COMPUTER ASSOCIATES, INC.
26 PRINCESS STREET, WAKEFIELD, MASS. 01880 • 617/245-9540

System Modeling with Net Structures

by

Robert M. Shapiro

Meta Information Applications, Inc.

March 14, 1975

A SUBSIDIARY OF APPLIED DATA RESEARCH, INC.

TABLE OF CONTENTS

Preface

This paper was presented at a meeting sponsored jointly by Gesellschaft für Informatik and Gesellschaft für Mathematik und Datenverarbeitung and held at Schloss Birlinghoven, West Germany, June 26 through June 28, 1974.

The paper is a companion piece to another paper presented by Dr. Anatol Holt at the same meeting. Reference is made to that paper (1) for motivational background and a description of the formalism underlying this work.

Abstract

This paper describes a new method for modeling systems, based on a formalism developed by the Information Systems Theory Project. The method is adequate for the modeling of systems which involve concurrency and choice in a cyclic context. It permits mechanized construction procedures, has a semantic basis and promises to lend itself to mathematical analysis.

ISTP is a project under the direction of A.W. Holt at Massachusetts Computer Associates, Inc.

Introduction

My interest in system modeling with net structures grew out of some practical problems I was engaged in 12 years ago. At that time I was developing some techniques for code optimization as part of a compiler-generator system. I needed a representational form adequate for modeling algorithms written in conventional programming languages. This representation would have to be able to model cyclic structures with branches and merges, corresponding to the loops and decision structure of conventional algorithms. Furthermore, in order to be truly useful, the representation would have to facilitate the application of analytical techniques that yielded interesting results vis-a-vis the system being modeled; in particular, analytical techniques for code optimization.

The introduction of computer systems with increasing capability for various modes of concurrent operation enhanced general interest in this area. My work along these lines resulted in the development of a technique for translating conventional algorithms into Petri Nets. The translation technique (2) produced a version of the algorithm in which most of the accidental sequencing constraints introduced by the programming language were removed. As a by-product some conventional optimization problems, such as the recognition of common subexpressions and loop invariants, became much easier to solve (3).

However, once the Petri Net representation was constructed, there was not much that could be done with it other than simulation. (See reference (4) for an example of optimization via simulation.) In other words, in general, Petri Nets do not guarantee that useful analytical techniques can be applied to them.

Subsequently, researchers in various parts of the world began to produce interesting mathematical methods for various special classes of Petri Nets. One such special class is Marked Graphs, for which a number of powerful analytical methods have been devised (5). Unfortunately, Marked Graphs are unsuited to the representation of systems in which choice plays a role.

This background sets the stage for the work we are now engaged in. My focus of interest in respect to the application of these modeling techniques has shifted somewhat. In particular, whereas in my earlier work I was concerned primarily with programming problems, the current effort has led to an approach which is oriented toward organizational problems in which people and machines perform roles within an organizational framework (1). Programming problems can also be viewed in this way.

The Formalism

Our work of the past few years has led to a new class of structures: T-nets (1). These structures have a number of interesting features, including:

(a) they are suitable for representing both concurrency and choice in a cyclic context;

(b) they are highly regular;

(c) they decompose into a number of semantically interesting components;

(d) in spite of the fact that they can represent choice, they have a Marked Graph aspect.

Thus, they are adequate for modeling interesting systems, permit mechanized construction procedures, have a semantic basis, and promise to lend themselves to calculational techniques.

I will briefly review some aspects of the formalism pertinent to what is to follow. For a general discussion of the formalism, refer to the paper by Anatol Holt (1). For a complete mathematical description of the formal apparatus, await further publications. For a brief review of Petri Nets, Marked Graphs and State Machines, refer to Appendix I (5).

Referring to Figure 1, the structures are composed of:

(1) states, represented by circles;

(2) events, represented by boxes (or bars);

(3) arrows, which connect states and events.

Both states and events have two input arrows and two output arrows (in an incomplete structure some of these arrows may be missing). The arrows are of two types, hollow and solid. (Petri Nets have only one arrow type.) In all cases, the two input arrows must be of alternate type and similarly with the output arrows. There are six pathways, definable in terms of arrow patterns across states and events. In a finite complete T-net, the formalism guarantees a number of properties for these pathways. Here we mention only a few:

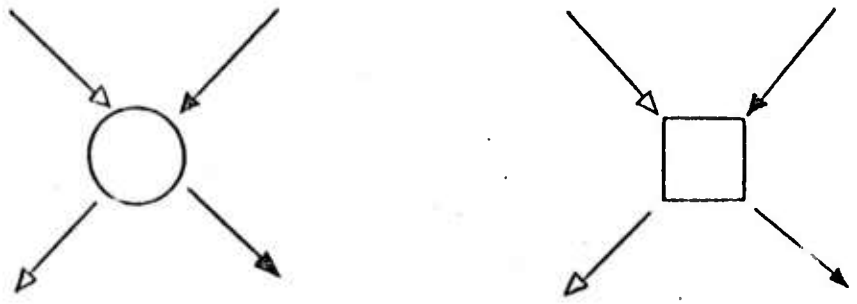
(1) Every path forms a simple cycle

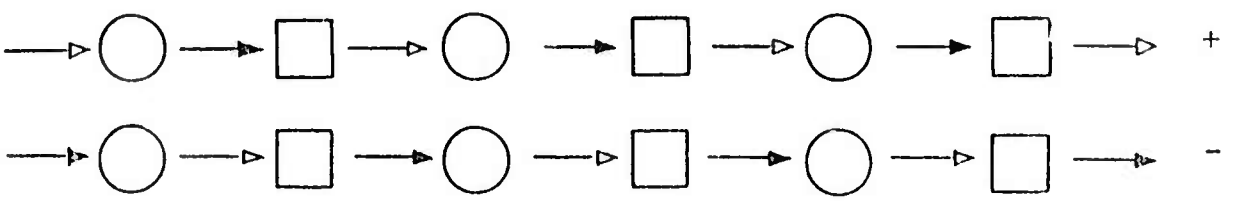
(2) The T-net is uniquely partitioned by each of six component types made up out of the six path types in the following way:

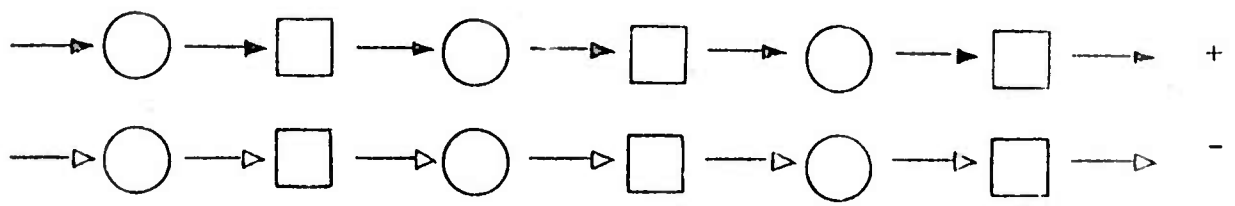
(a) M_o, state component, consisting of all mode paths that meet at states (See Figure 1a)

(b) $M_\square$, value component, consisting of all mode paths that meet at events

(c) R_o, item domain component, consisting of all ray paths that meet at states







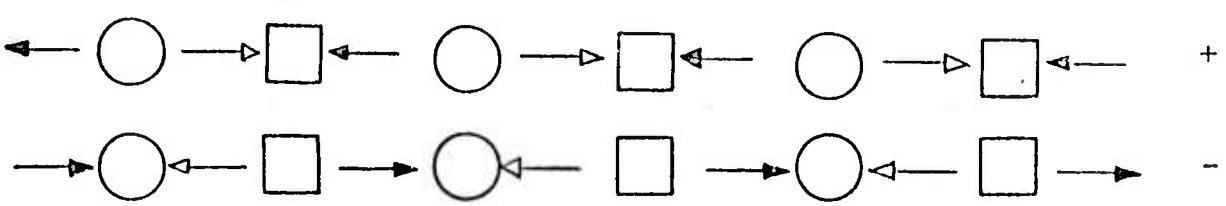
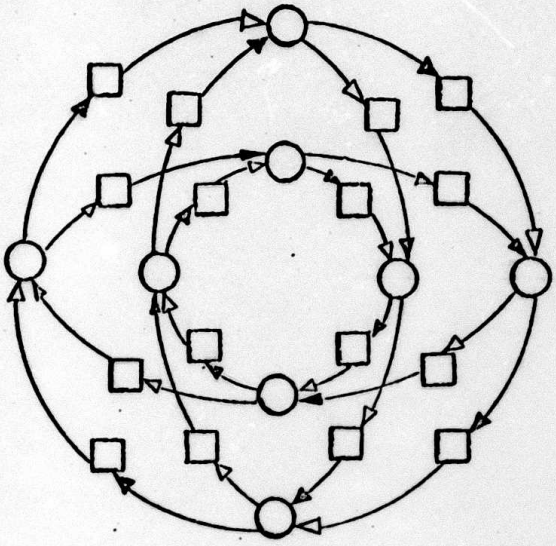


Figure 1



A state component M_o, consisting of four mode paths. M_o is complete because all four mode paths close and all states are saturated; i.e., have two inputs and two outputs. The events have only one input and one output arc, hence the manner in which this state component interacts with other state components is completely unspecified.

Figure 1a

(d) $R_{\square}$, effect component, consisting of all ray paths that meet at events

(e) Σ_{o}, transform component, consisting of all slot paths that meet at states

(f) $\Sigma_{\square}$, transfer component, consisting of all slot paths that meet at events.

In the examples presented in this paper, we focus on three of these component types: state components, item domain components and transfer components. In terms of the semantic basis, a state component corresponds to a role or part performed by some player, or alternatively, as a part in some machine. A state component can be thought of as a state machine which does one thing at a time, with alternate possibilities represented by branching and merging at states. In terms of markings, a state component M_o always has exactly one token on one arc in M_o at all times. (In Petri Nets, tokens are usually placed on states rather than arcs.) An item domain component can be thought of as a format or domain for a class of items. A token within a particular item domain represents a record or item of that class. An item domain can have tokens on various arcs concurrently. A transfer component corresponds to an interchange of items between parts.

Introduction to the Model

I am going to present a modeling example which utilizes T-nets. This model is a work-piece. In our work, it has been useful as a demonstration of how aspects of the formalism relate to problem semantics. In particular, this model conveys an idea of the semantic content and interrelationships between state components, item domain components and transfer components.

The model is necessarily simple to permit detailed inspection of the structures involved. Nevertheless, the elements necessary for modeling complex systems are all exhibited. I will describe the model in terms of the flow of items over a forking and merging track system. It is our view that this is, in fact, a general semantic framework for describing and analyzing systems.

It is not the intent of this paper to convey enough information to the reader to allow him or her to immediately use the representational techniques or to see in complete detail how the model is based on the formalism. Forthcoming publications will make this possible. Instead, I will only attempt to make plausible the semantic connections between the model and the underlying formalism.

The focal point of the model is the connection between information flow and synchronization. In the framework mentioned above, the flow of items over a forking and merging track structure, it will be shown that directional information which affects item flow is itself represented as item flow. Thus items will sometimes function as information which affects the flow of other items. Hence the structures will exhibit the effect of information on flow and provide a context for asking questions of the following type: What information is critical so that an item can flow from one point in the structure to another point; phrased in terms of system input, which inputs can conceivably effect the flow of particular items over part or all of the structure?

A Notation for the Model

In the last month, a higher level notation has been devised which permits more rapid construction of T-nets. It is this notation which I will use to illustrate system modeling. The notation should be regarded as a step in the process of developing higher level descriptive forms which can be mapped onto T-nets. Further steps must be taken in order to permit economical description of complex systems.

The model will be constructed out of units which we refer to as Quine transfers. These were originally employed in net structures by C.A. Petri. A Quine transfer is a particular kind of transfer component in the underlying formalism. Referring to Figure 2, such a transfer can be represented as a box with one side thicker than the others. There are four agents (state components) involved in this transfer. They are represented as four directed lines which touch the four corners of the box. The transfer, when it occurs, causes the four agents to exchange items in a certain way. The two agents labeled E and W, the ones touching the box on the side opposite the thick edge, determine which of two possible exchanges take place:

(1) N and E exchange, S and W exchange; i.e., exchange along edges

(2) N and W exchange, S and E exchange; i.e., exchange across

A part brings to a transfer one of two possible item types. In a Quine transfer, E and W as a pair are constrained to bring to the transfer the same item type, representing exactly two of the four possibilities. Thus, E and W can be

thought of as a dual representation of directional information, in much the same sense as a traffic signal for a switch says which way to go (green) and which way not to go (red).

N and W are unconstrained and thus bring to the transfer four possible patterns. If we assign to the two possible item types the designations 0 and 1, E/W with a 1 will designate possibility 1 (N and E exchange, S and W exchange). E/W with a 0 will designate the other possibility.

It is easy to see that N and S each end up with the item type brought to the transfer by the E/W pair. Hence, for N and S, the result is simply a function of E/W; namely, the identity function. On the other hand, we may write the result for E and W in the following way:

$$\left.\begin{aligned} E' &= E \wedge N \;\vee\; \overline{E} \wedge S \\ W' &= W \wedge S \;\vee\; \overline{W} \wedge N \end{aligned}\right\} \quad E \equiv W$$

Thus, E' and W' are functions of the original E/W item type and the N item type and the S item type. Please notice that if we constrain S to have the item type designated by 0, then $E' = E \wedge N$, and the transfer can be viewed as performing a logical function, with one argument represented by the E/W pair and the other by N, with S fixed at 0. This type of constraint thus permits the Quine transfer to be used to represent the performance of various logical functions, and also to be used to accomplish fanout. We will refer to such a constraint as enlogical, and use the expression enlogy to designate a value so constrained. (The word enlogy and its use are due to C.A. Petri.)

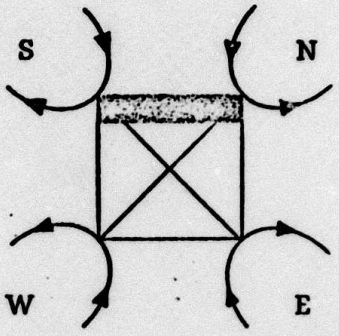


Figure 2

In the notation, a transfer can occur when all four parts involved in the transfer are prepared for the interchange. This will be represented by the presence of tokens on all four input arcs. (These tokens do not correspond to tokens in Petri Nets, since a token here represents two distinct possibilities. The Petri Net representation of the Quine transfer is given later in this paper.) In these structures, parts correspond to state components. Hence a part will always have exactly one token on it at all times. To trace a part, simply follow a directed line which touches the corner of a sequence of transfers. If the part is completely represented, this line will form a simple cycle. Various constraints in the notation arise from the underlying formalism but these need not concern us here.

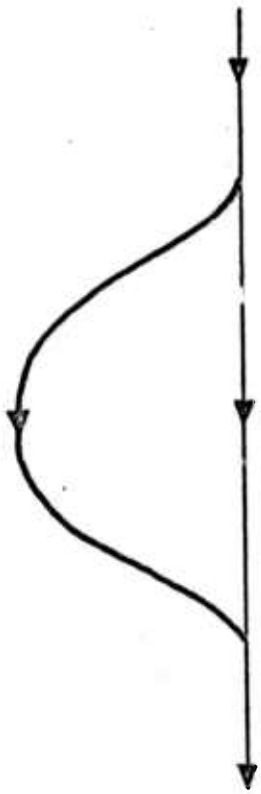
The representation of the concepts of concurrency and choice is a central concern in the development of a mathematical basis for system description. As I have already stated, these concepts can be thought of in a concrete physical context: the motion of items over a set of tracks which fork and merge.

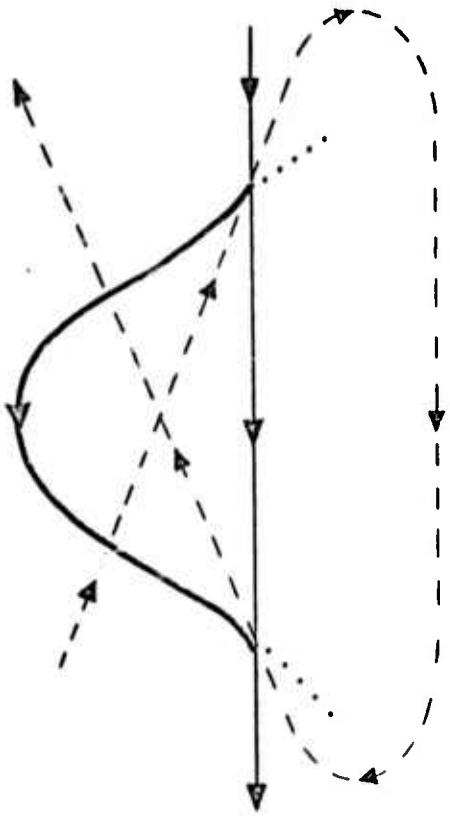
The exercise of choice corresponds to directing the flow down one branch of a fork. Concurrency is exhibited in the flow of items over different sections of the track at the same time. The ideas developed in the "Representation of Algorithms" (2) were based on these images. I am now going to discuss a model using the Quine transfer as a building block, represented in the manner I have just described.

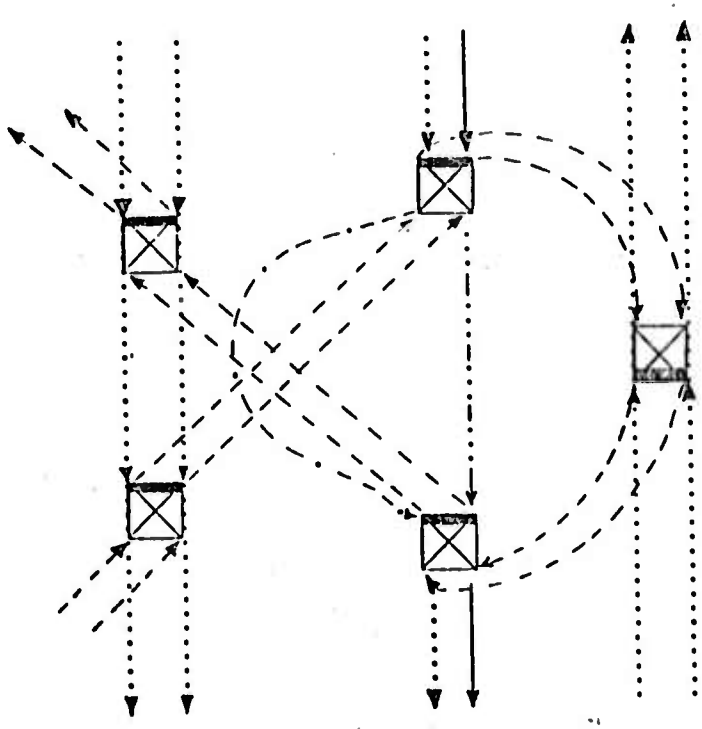
A Simple Model of Vehicular Flow

Referring to Figure 3, we see on the upper left a section of some larger structure which depicts the flow of items which represent waves of possible car arrivals in a forking and merging track network. In this section, vehicle flow forks and then merges. On the right, this section of the structure is fleshed out in more detail, showing the use of items as signals which control the direction of traffic flow. The flow of directional information is depicted by the line made up of dashes. Thus, the directional information is first used to determine which way traffic will flow at the fork; i.e., which branch will be taken. The same directional information is then used to determine from which branch into the merge a vehicle can come. The directional information flows in the "reverse" direction to traffic flow and functions both as a method of controlling the direction of traffic flow, as well as synchronization of traffic flow; i.e., the establishment of the necessary preconditions for vehicles to pass a fork or a merge. (In reference (1), the example about traders and agents refers to these backflow entities as permissions.)

The dotted continuation of the track structure permits us to view this structure in terms of the four part Quine transfer previously discussed. The top dotted path into the fork will be guaranteed to never have a car on it (enlogy). The bottom dotted path of the merge will also be guaranteed to never have a car on it. A semantic interpretation for a failure of this guarantee might be derailment; i.e., an inconsistency between directional information and where the approaching car is coming from, in the same sense as a train approaching an open switch.







directional information – – – – – –

possible car arrivals ——————

or ……

or – – – . – – –

enlogy .....

Figure 3

The lower half of Figure 3 depicts the track model represented in terms of Quine transfers. The five types of lines encode aspects of information flow that are deducible from the underlying formalism. They are employed here for pictoral clarity and are not part of the higher level notation.

The Quine transfer representation introduces some new synchronization features. Possible car arrivals and directional signaling are coordinated so that new directional information cannot be provided until the last directional information has been utilized at both the fork and the merge. This has the further consequence of guaranteeing there is no vehicle in the sections of traffic between the fork and the merge when another vehicle crosses the branch. (Recall that a part always has exactly one token on it.)

The box in the upper right hand corner of this structure is a Quine transfer that corresponds to the fork in the original track structure. The box in the lower right hand corner corresponds to the merge in the original track structure. The box in the lower left hand corner is the entry point for directional information and the box in the upper left hand corner is the exit point for this directional information. The box on the far right is a Quine transfer used to transmit directional information from part to part and is needed in order to permit the representation of track structures composed of a number of fork-merge substructures without imposing unintended restrictions on the relative locations of cars over the structure. If this transfer were omitted, the structure would not be a suitable building block for composing larger structures: such structures would be constrained to have only one vehicle, contrary to our semantic intentions.

The flow of directional information can be traced by following the dashed lines through the structure. Possible vehicle flow comes into the structure on the solid line entering the upper right transfer; it leaves the structure on the solid line out of the bottom right transfer. The two alternate tracks for vehicles within the structure are depicted by line (... _ ... _) or line (--- · --- ·). Lines which are guaranteed to have enlogy are depicted by (.....) .

In this structure some of the parts are complete and others are incomplete and determine the manner in which this substructure connects with similar substructures above and below. To trace a complete part, we can start on either one of the enlogy lines running from the upper left transfer to the lower left transfer. At each transfer go out on the line that touches the same corner you came in on. Thus, we can trace a closed simple circuit in which a single part participates in a fixed sequence of transfers; namely, (1) lower left, (2) upper right, (3) lower right and (4) upper left. In terms of part activities, the part (1) exchanges enlogy for directional information, (2) exchanges directional information for a possible car arrival, (3) exchanges a possible car arrival for directional information and (4) exchanges directional information for enlogy.

Two parts play this role in the structure. A third part (the solid line entering the structure) (1) exchanges a possible car arrival for directional information, (2) exchanges directional information for enlogy, (3) (in the structure above, not depicted but analogous to this structure at the bottom right) exchanges enlogy for directional information and (4) exchanges directional informati for a possible car arrival. A fourth part (the enlogy line entering the upper right transfer) (1) exchanges enlogy for directional information, (2) exchanges directional information for enlogy, (3) exchanges enlogy for directional information and (4) exchanges directional information for enlogy. To complete the description, there is a fifth and a sixth part which bring directional information into the structure and take used directional information out of the structure.

To permit the directional information to change, the underlying formal structure would be incomplete for these two parts. All the other parts can be complete and such a structure is exhibited in Figure 4. (Note that the lines (- - - - - - -), (. . . _ . . . _) and (- - · - - - ·) in the upper segment are not information equivalent to the analogous lines in the lower segment, whereas (. . . . . .) and (———) are. The directional information for each segment is independent! The line encodement is not part of the notation, and is only suggestive of the information flow analysis in the underlying structure!

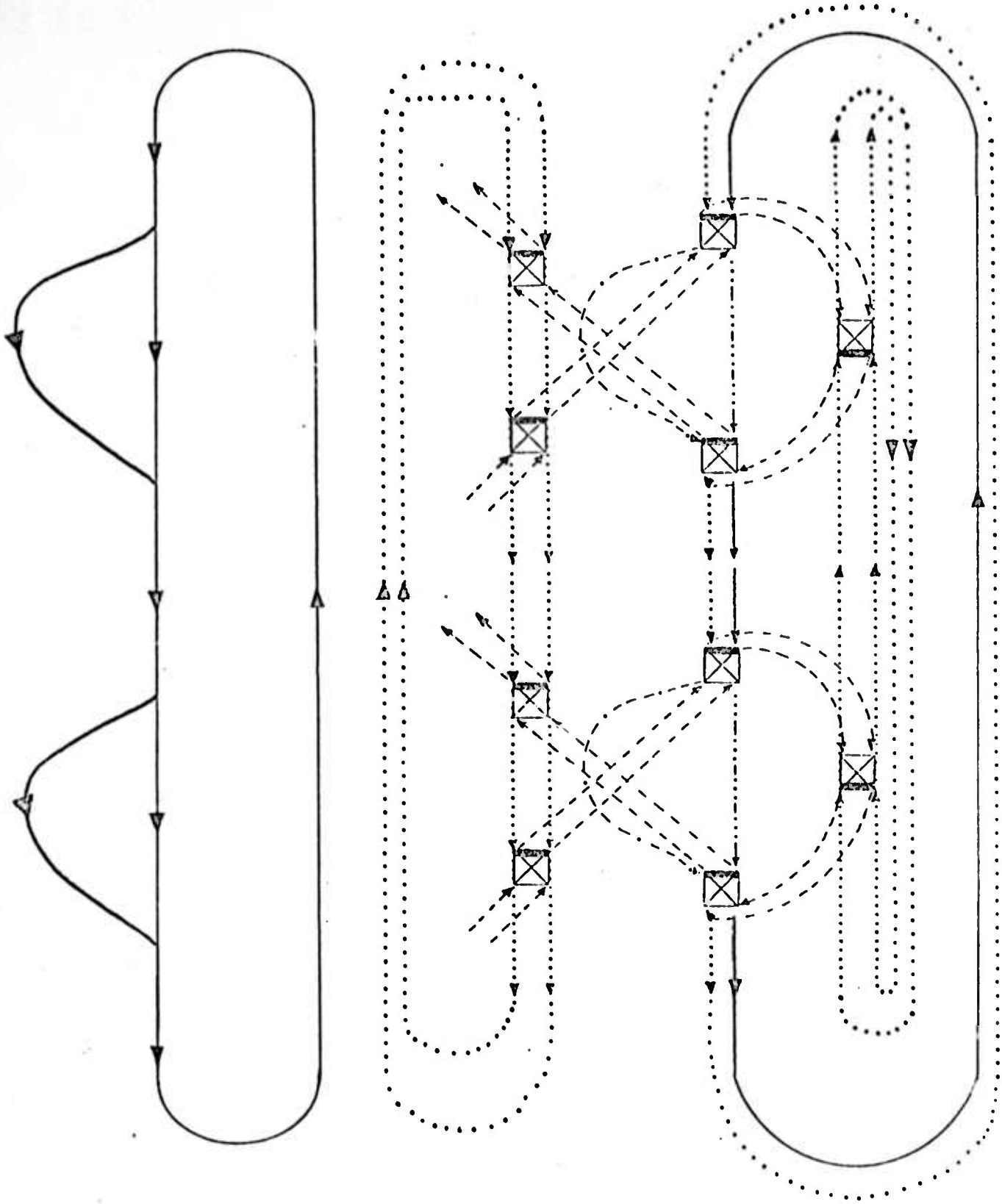


Figure 4

Figures 5, 6 and 7 depict a sequence of frames exhibiting traffic flow over the structure. Tokens are exhibited by means of small circles touching a particular line segment. Inside the circles, there is a key to what a token there stands for. Thus, g is used for green directional information (in the Quine transfer parts exchange along edges, described as exchange 1 in the discussion of the notation) and r is used for red directional information. e is used to represent enlogy, and c is a car.

The sequence first shows the effect of green directional information on possible vehicle flow. It then shows the effect of red directional information on possible vehicle flow. In the case of green directional information, a possible vehicle arrival is routed along the edge of the fork transfer and then the edge of the merge transfer. With red directional information, a possible vehicle arrival is routed diagonally across the fork transfer and then diagonally across the merge transfer. Thus, in either case, the use of directional items to control vehicle flow guarantees that the item which enters on the upper solid line will leave on the lower solid line.

We can now discuss, in more detail, some aspects of information flow in terms of the underlying components. Recall that in a Quine transfer what the N/S pair hold after the transfer takes place is completely determined by what the E/W pair had before the transfer. Thus, the flow of directional items in the structure, always entering a transfer as E/W and leaving as N/S, guarantee the preservation of the directional information and justify the use of the dashed lines as an exhibit of dependency of this information only upon what came in the lower left transfer.

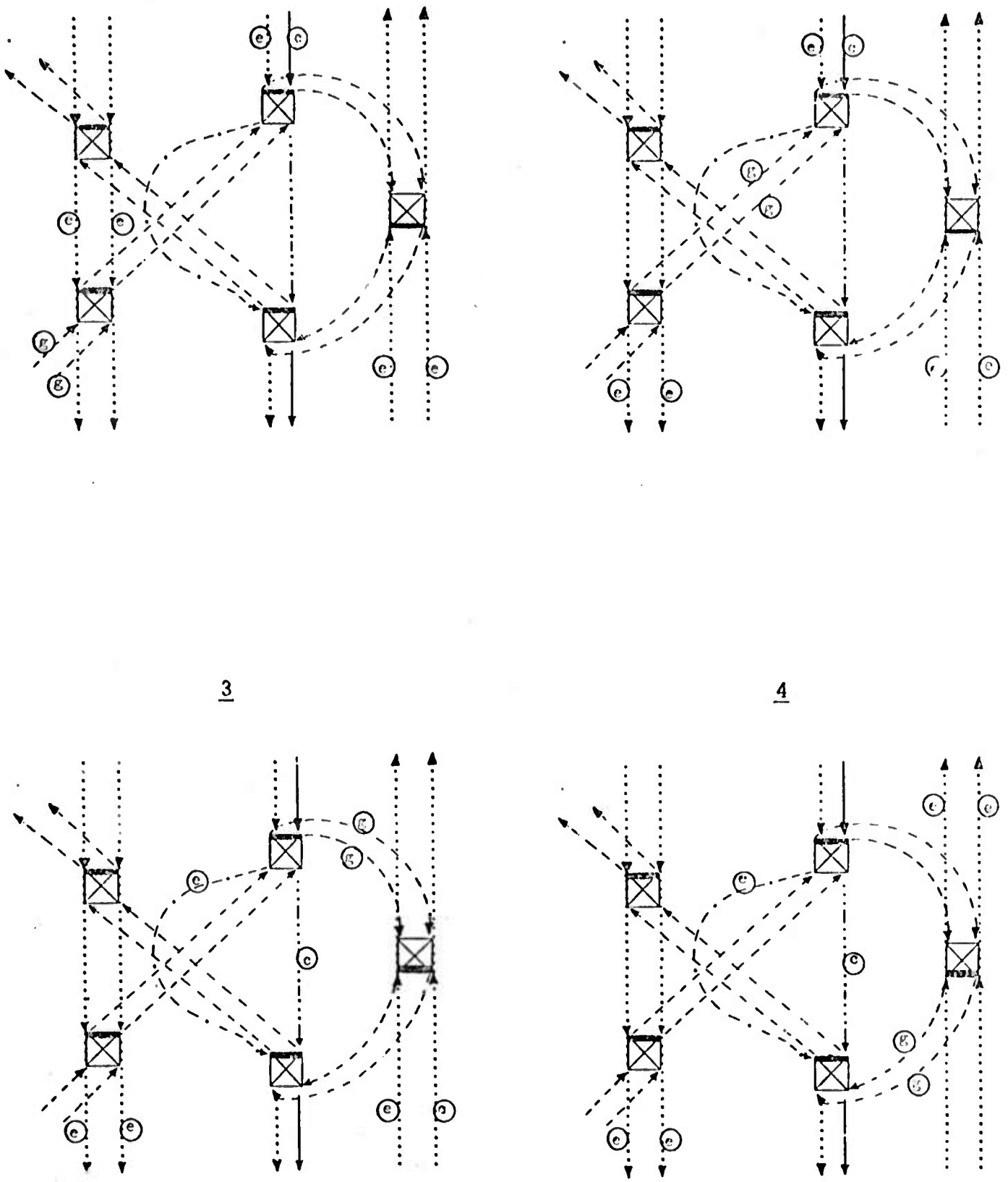
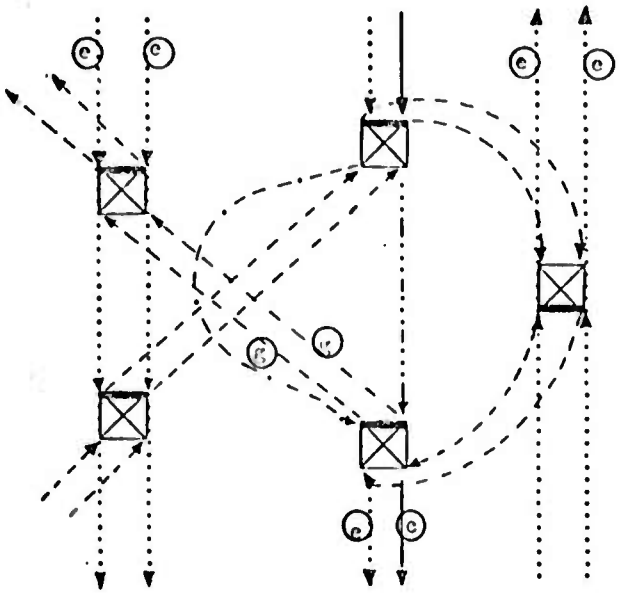
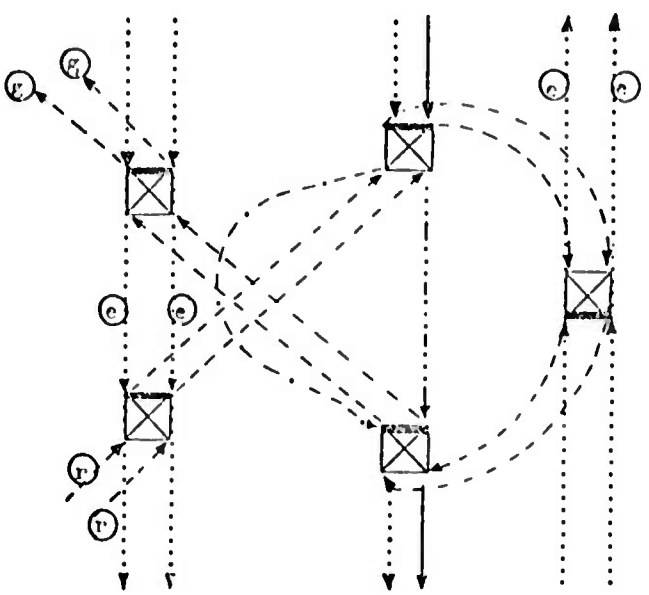


Figure 5

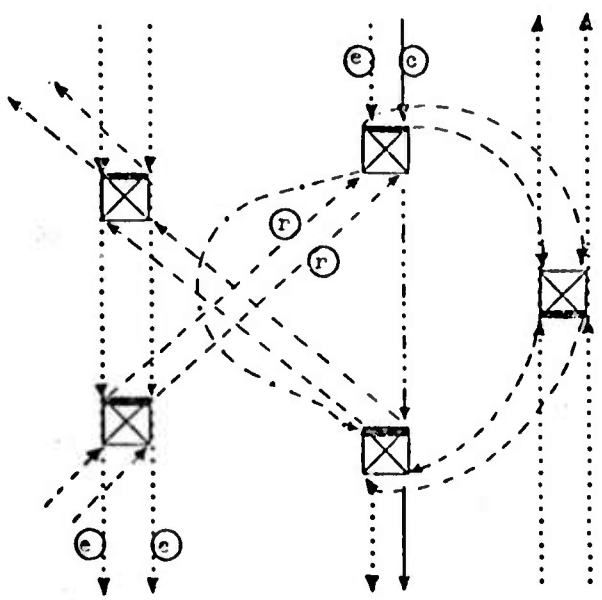
5



6



7



8

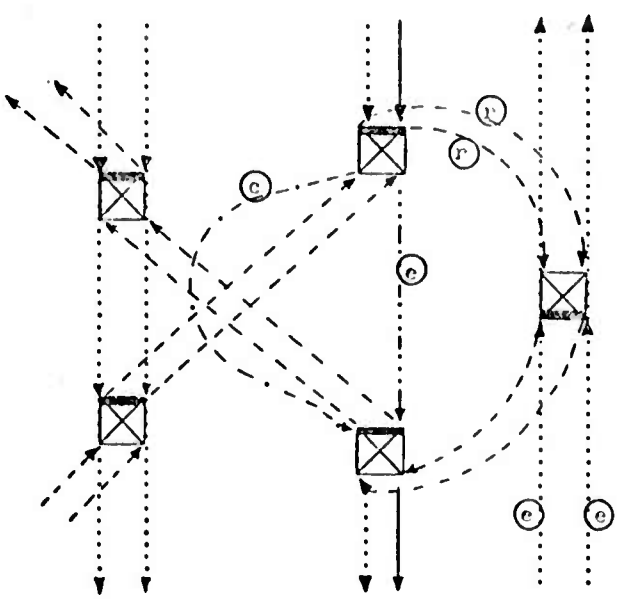
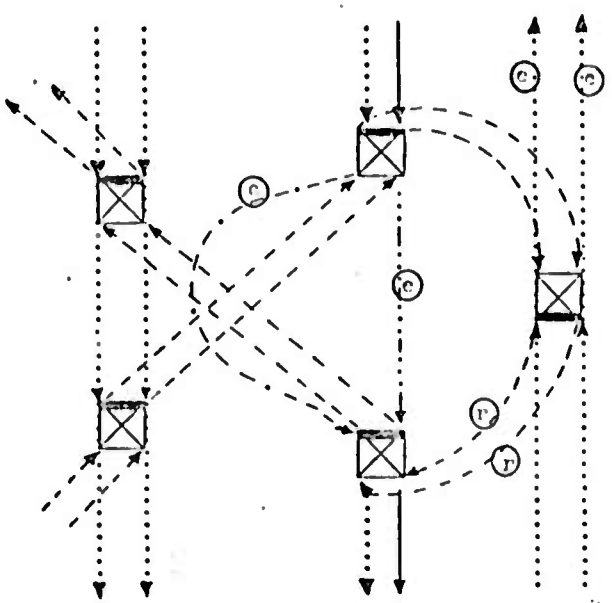
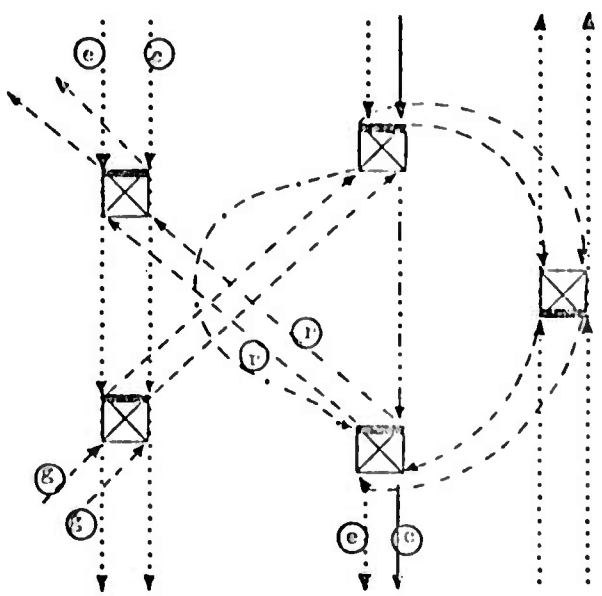


Figure 6

9



10



11

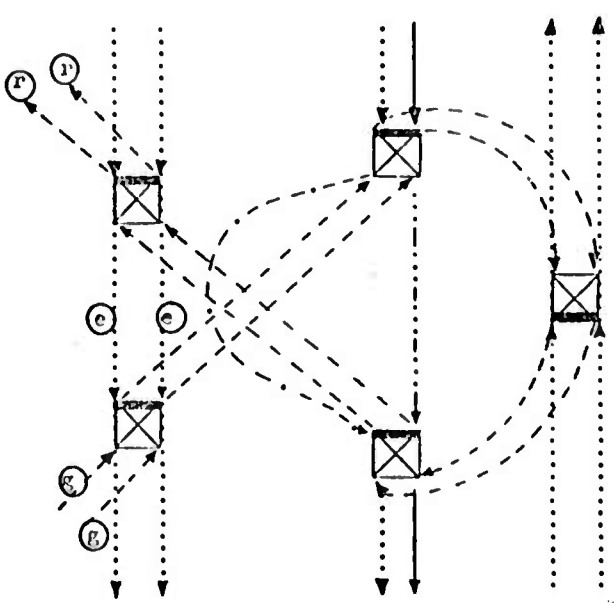


Figure 7

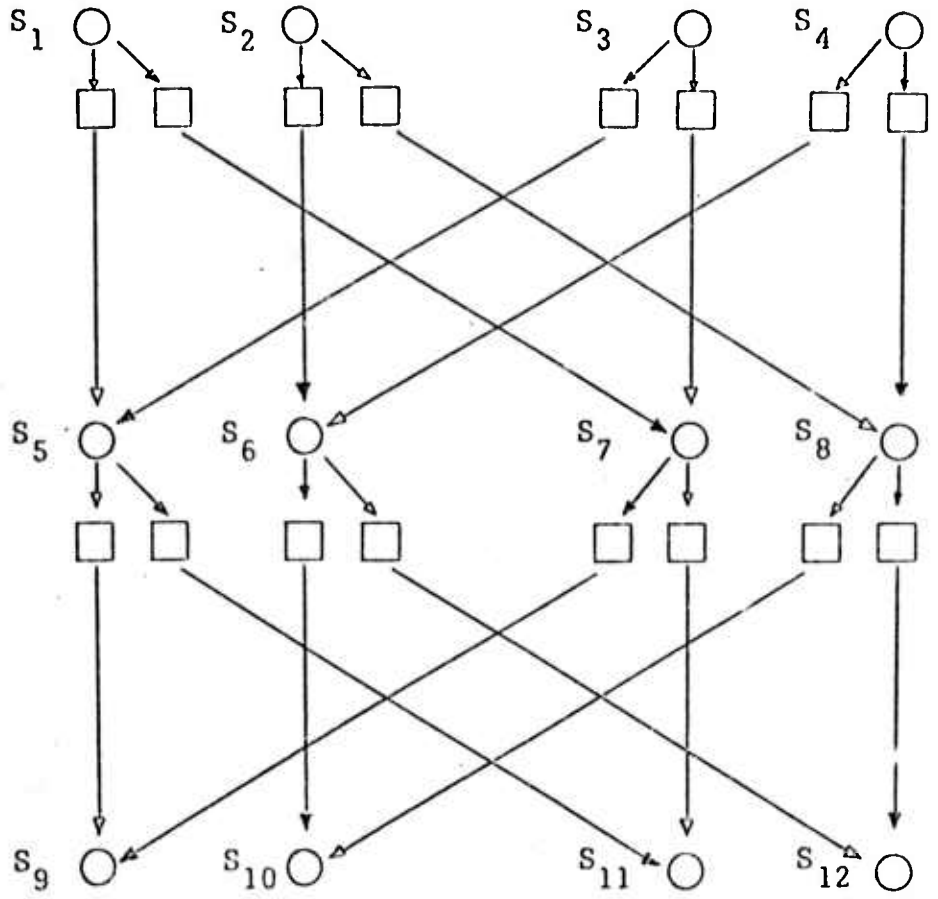
On the other hand, what E and W end up with is a function of both what the E/W pair brought to the transfer as well as what N and S bring to the transfer. The lines represented by (... - ... _) and (--- · --- ·) are functions of the directional information and whether a vehicle arrived or failed to arrive during this signaling sequence. After the bottom right transfer, because of the fact that the same control information is being used, we can depict the lines leaving the merge transfer in the same manner as the ones entering the fork transfer. Figure 8 depicts the underlying T-net structure of the fork transfer followed by the merge transfer, focusing on the rays that define the item domain for a car; i.e., track paths rather than the parts or the flow of directional items. (Refer to the discussion of components in the section on the formalism. The reader is reminded that he or she cannot see exactly how Figure 8 is arrived at from the track structure. Later in this paper the underlying structure of the Quine transfer is discovered. With that knowledge, it is possible to reconstruct Figure 8, to within certain symmetric arrangements which all work in the context of the simple semantics dealt with in the model under discussion.) Figure 9 shows a logical calculation, based on the Quine transfer logic, which proves that "what goes in, must come out" for this substructure. In other words, the directional information that determines the switching can have no bearing (in this case) on whether a car (item) arriving at the fork is guaranteed to be on the track after the merge. (If either or both of the track sections between the fork and merge also permitted branching and not all these branches merged before the main merge, the formalism would guarantee a difference between the information content at the "before branch point" and "after merge point".

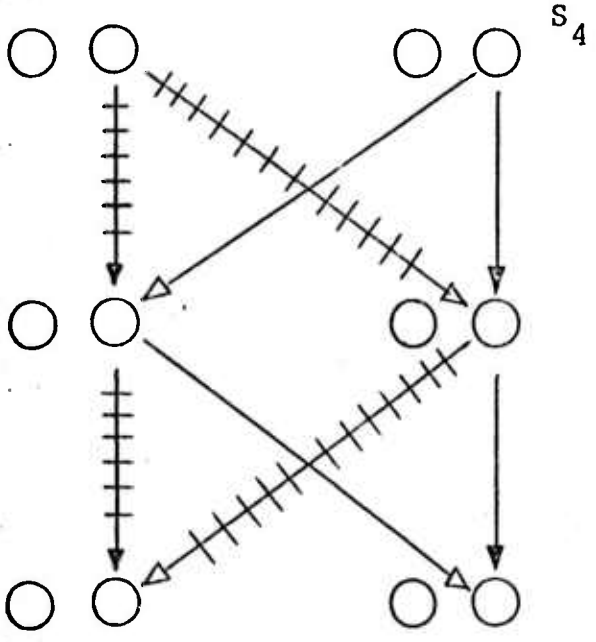
In the underlying formal structure there is a single item domain for cars over the whole structure. The presence of one or more cars is represented by tokens on one or more arcs of this structure. Enlogical constraints guarantee that parts of the car item domain never have tokens on them. In Figure 8, the bottom left illustration depicts the car item domain in a forked-merged section of track.

There is another item domain which permits the calculation of the spacing between cars. The number of car lengths between two cars is represented by an equal number of tokens in this structure. Because the semantics of the model are simple, we have not provided for cars changing relative position, so that this token count is guaranteed constant.

Directional information flow in the underlying structure is represented by two pairs of item domains. One pair corresponds to a green signal, and the other to a red signal.

One important modeling possibility is not exhibited in the example. This case involves the flow of directional information being controlled by other information. For instance, we have made no use of car position in determining the switch settings. The formalism and higher level notation handle this case, but the models are more complex, partly because directional information must always be dually represented (the Quine transfer requires the E/W pair to hold the same item type).





Car domain within structure
Hatched lines indicate part of item domain in which no car will ever be

Figure 8

$$S_{12} = (d \wedge S_8) \vee (\bar{d} \wedge S_6)$$

$$S_8 = (d \wedge S_4) \vee (\bar{d} \wedge S_2)$$

$$S_6 = (d \wedge S_2) \vee (\bar{d} \wedge S_4)$$

$$S_{12} = (d \wedge ((d \wedge S_4) \vee (\bar{d} \wedge S_2))) \vee (\bar{d} \wedge ((d \wedge S_2) \vee (\bar{d} \wedge S_4)))$$

$$= d \wedge S_4 \vee \bar{d} \wedge S_4$$

$$S_{12} = S_4$$

$$S_{r+8} = S_r \qquad r = 1, 2, 3, 4$$

d = directional information

Figure 9

The Quine Transfer

The higher level notation I have used to describe the vehicular flow model is based on the use of the Quine transfer. Figure 10 depicts the Quine transfer as a T-net. The upper figure focuses on the transfer component and on the state components (parts) involved in the interchange. The lower picture focuses on the rays of the item domains involved. In both figures, the same net structure is exhibited.

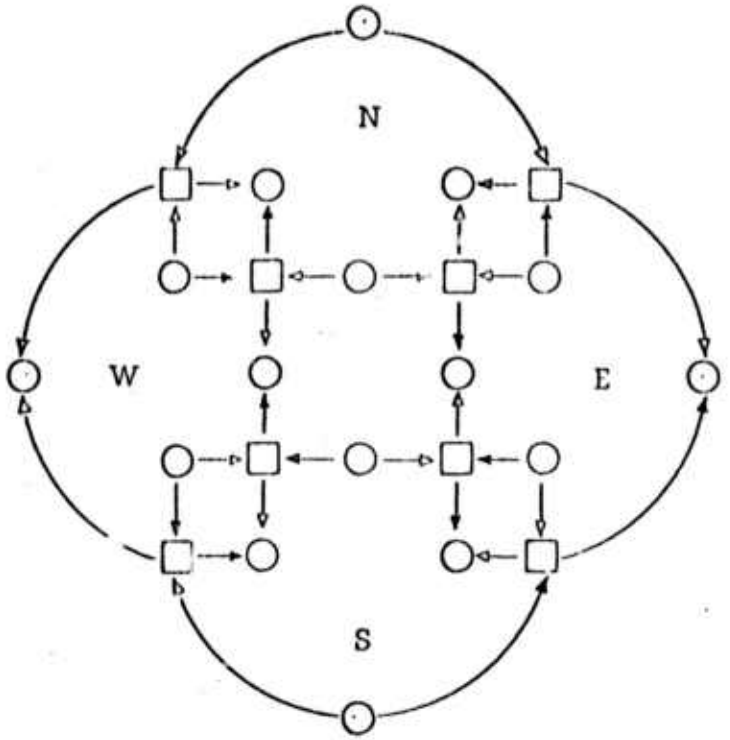
This structure is incomplete: there are eight states, two in each part, which have no input arcs. There are also eight states, two in each part, which have no output arcs. All of the events, however, are complete; they have two input arcs and two output arcs. A transfer component was defined earlier in the paper as consisting of all slot paths that meet at events. If you refer back to the definition of the two types of slot paths, you can verify that Figure 10 is indeed a single transfer component.

The definition of state component partitioned the states into four groups, as illustrated in the upper figure and labeled N, S, E and W. (There are no complete state components in the figure. Nevertheless, the definition given early in the paper, does partition the 16 states of Figure 10 into four state components. Consider the four states associated with N . Two of these states have no input arcs. These states each lie on two mode paths which lead to each of the other states. Each of the four mode paths meets two other mode paths at states, and the third mode path by one level of indirection. Thus, all four mode paths lie in the same M_o , and all four states are part of the same state component. Figure 10 represents a fragment of a net and hence does not guarantee that, for instance, N and E are separate parts since a mode path into or out of N might meet a mode path into or out of E elsewhere in the complete structure. However, the underlying formalism guarantees just that.)

The lower figure shows more clearly the rays that compose the item domains in the structure. For instance, there is a pair of rays that start at a single state in E and reach a single state in N. Another such pair goes from E to S . Two other such pairs go from W to N and S . One such ray pair is an item domain within the context of this substructure. The presence of a token (now in the sense of a Petri Net token) in a particular item domain within E, guarantees the presence of that token in a particular item domain of a particular part after the transfer. That is why N and S, after the interchange, have items which were totally determined by what E and W brought to the transfer.

As discussed previously, E and W are constrained to dually represent the same item type. We will now exhibit these markings in terms of the net structures.

Figure 10 can be thought of as a Petri Net simply by ignoring the distinction between the two types of arrow heads. In Figures 11 - 14, we use an 'X' next to a state to indicate the presence of a token. Figure 11 shows the E/W marking that guarantees that E and N exchange, W and S exchange. Figure 12 shows the E/W marking that guarantees the E and S exchange, W and N exchange. Figure 13 shows one complete marking possibility for the four parts before the transfer and Figure 14 shows what the resulting marking would be like after the transfer.



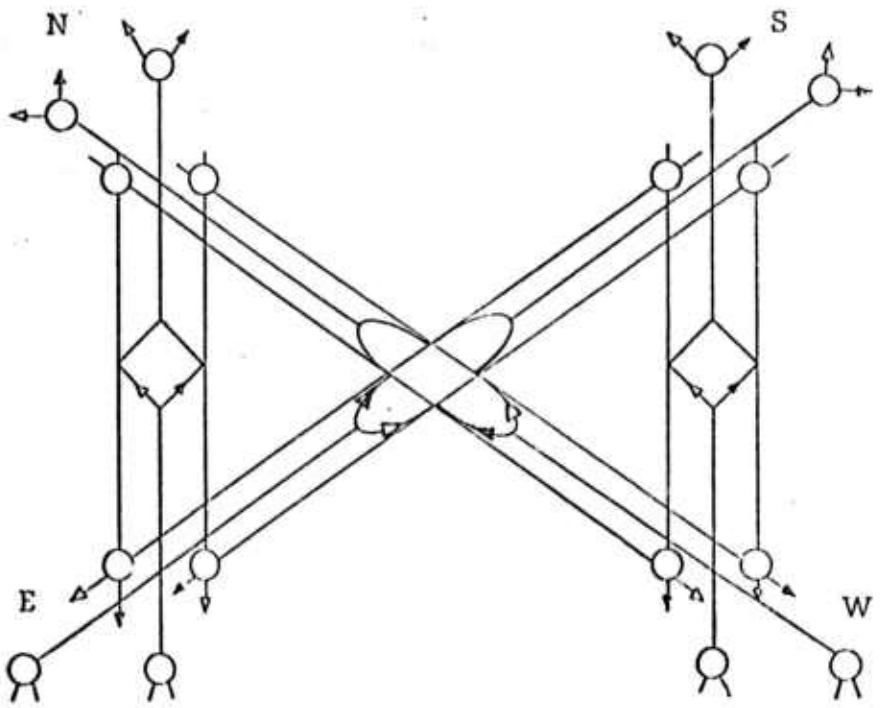
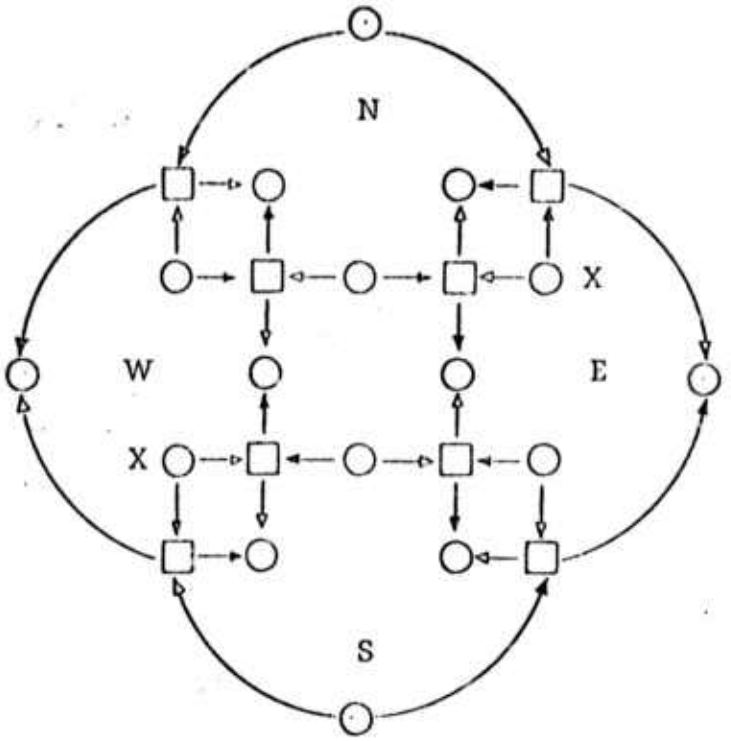


Figure 10



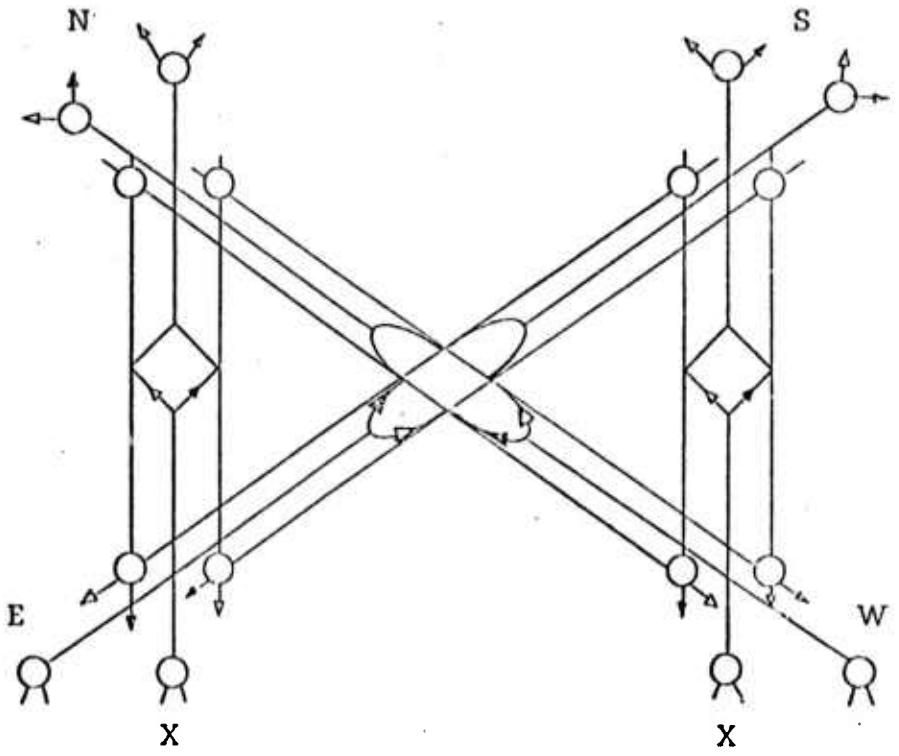
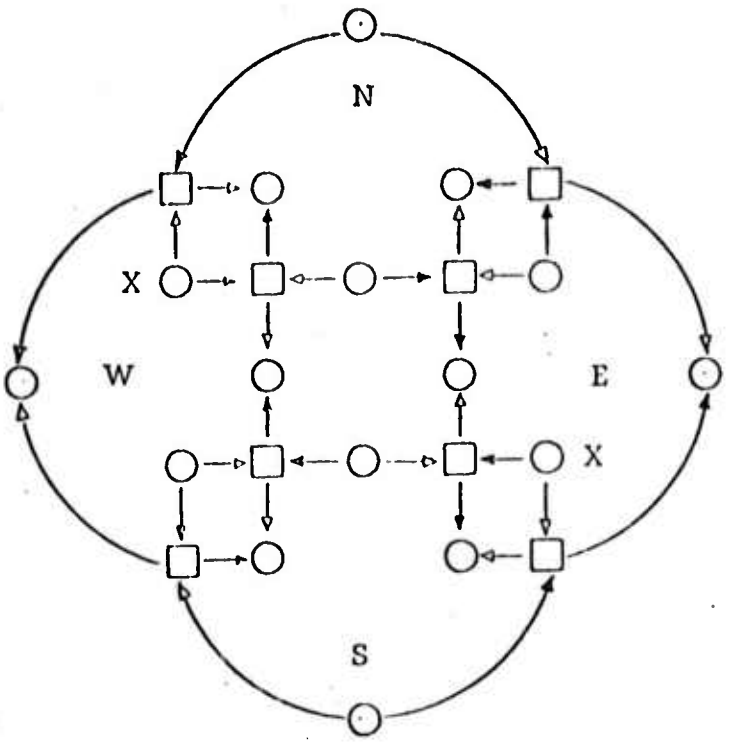


Figure 11



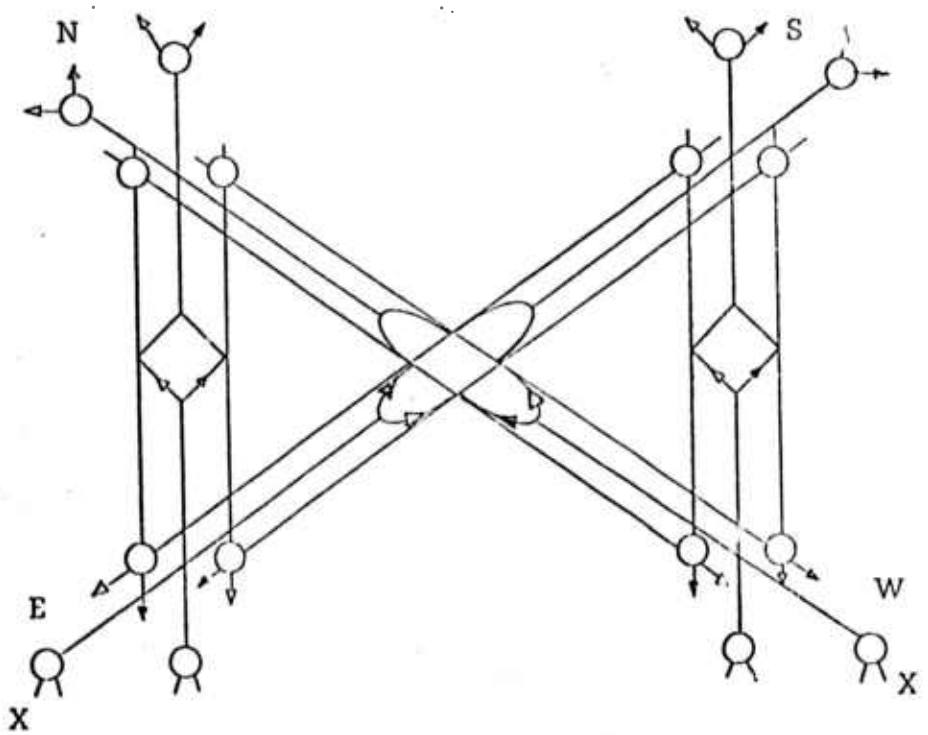
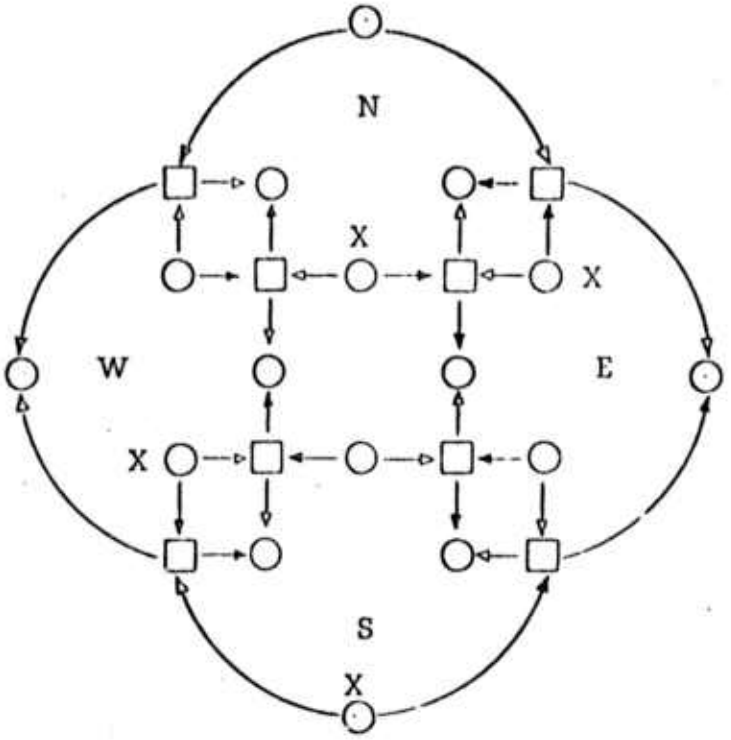


Figure 12



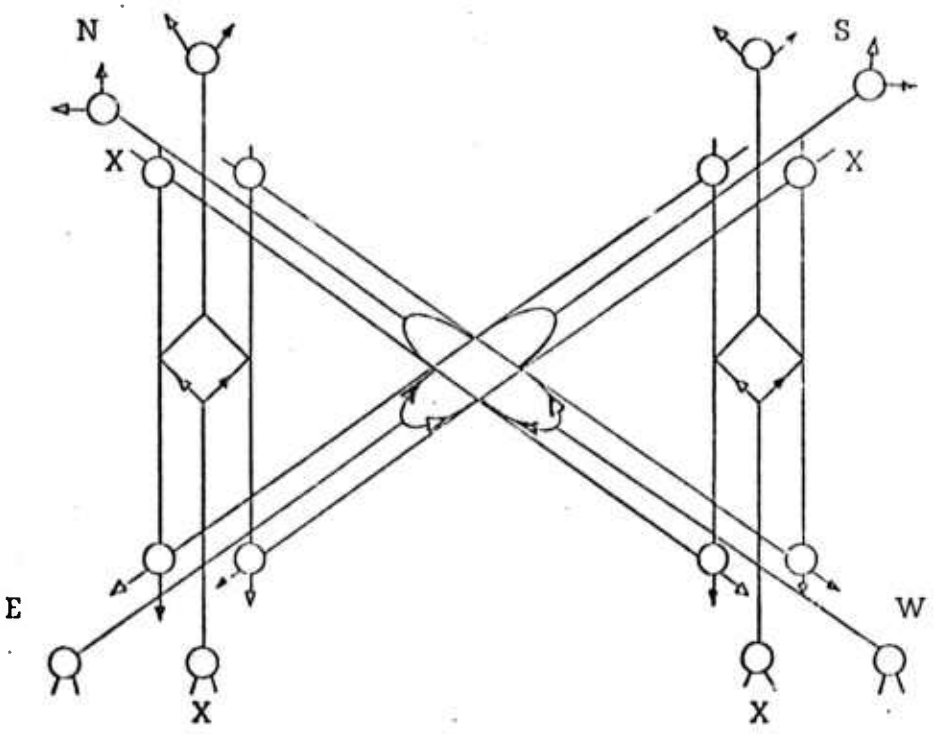
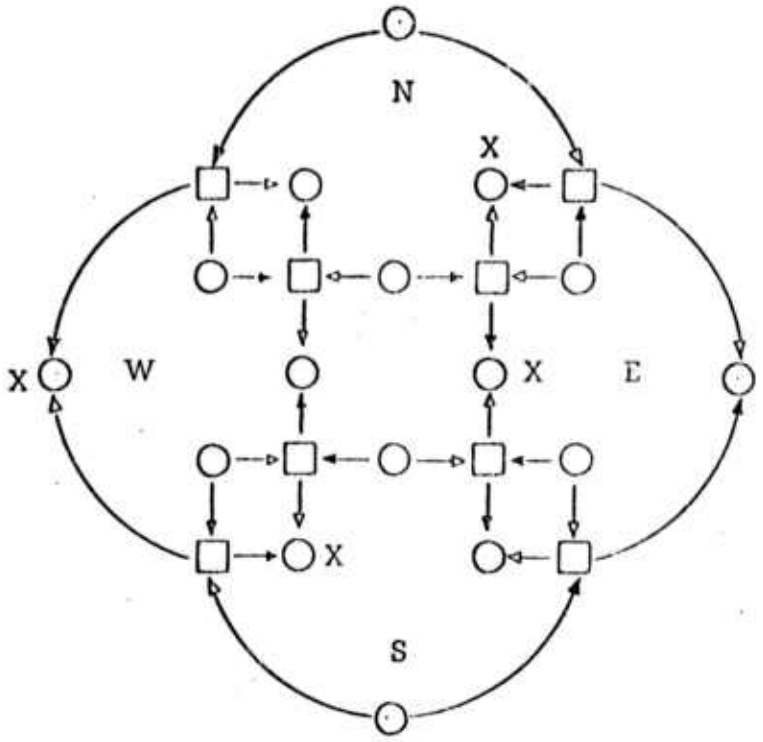


Figure 13



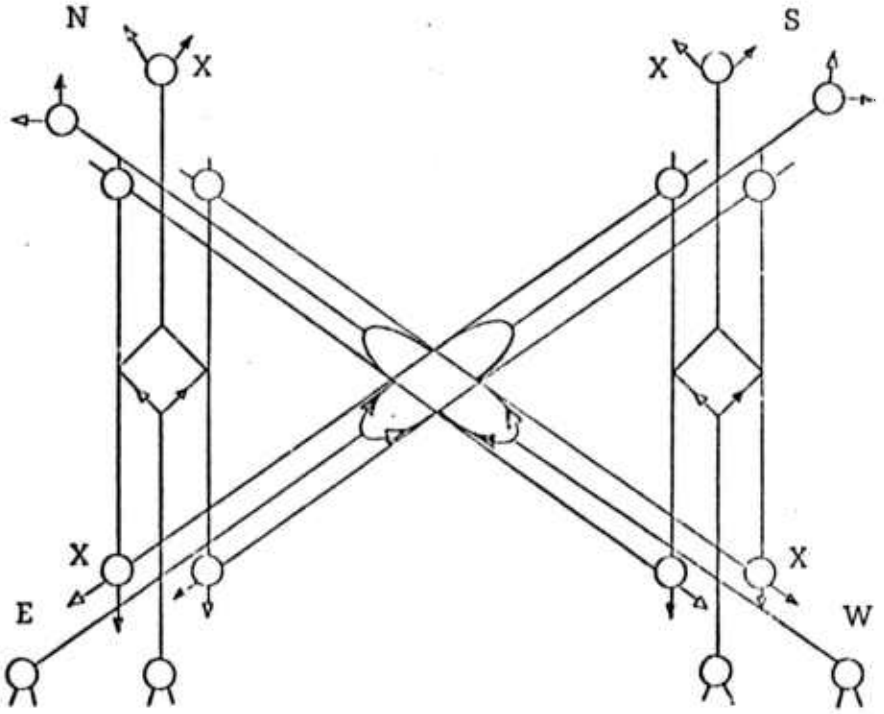


Figure 14

The Marked Graph Aspect

Earlier in this paper I stated that T-nets had a marked graph aspect. Though a complete exposition of this topic is beyond the scope of this paper, it is still possible to illustrate the basic idea with the help of the T-net representation of the Quine transfer.

In T-nets, the tokens in the underlying formalism lie not on states but, rather, on arcs between states and events (also, between events and states). In a Petri Net, events fire (occur) and in so doing transport tokens from input states to output states. In T-nets, both states and events can be thought of as firing, moving tokens from input arcs to output arcs. If two types of tokens are introduced; e.g., red and white, we can now crudely state a firing rule for these structures.

For an event firing: white tokens on both input arcs is a white firing. Both tokens are moved from input arcs to output arcs. Red tokens on both input arcs is a red firing. Both tokens are once again moved from input arcs to output arcs. One red token and one white token corresponds to system death (not live) in an ordinary Petri Net.

For a state firing: white tokens on both input arcs is a white firing, as in an event. Red tokens on both input arcs corresponds to a collision (not safe) in an ordinary Petri Net. One red and one white token corresponds to a choice situation in a Petri Net, the choice being which output arc gets the red token.

There is a conflict when the transfer component for which it is an input state has more than one possible outcome for the same input marking. IN A QUINE TRANSFER, THERE IS ONLY ONE POSSIBLE OUTCOME MARKING. Incomplete transfer components are the only ones in which a conflict exists. Such components represent a boundary of the system.

T-nets, with their regular structure, guarantee this marked graph aspect. In other words, a proper T-net model will be live and safe as a two-color marked graph with the above firing rule. If the color distinction is washed out and both events and states are thought of as marked graph events, the structure can then be thought of and analyzed using all the existing marked graph techniques. These will, of course, not answer questions which are color dependent; i.e., those aspects of the net that are choice dependent. For such questions, information flow analysis in the spirit of the vehicular flow model must be further developed.

The interested reader may mark up the Quine transfer with red and white tokens (using a red token to correspond to a token in the original net, and white tokens on all other arcs into the input states.

Thus, every input state will have tokens on both its input arcs; those states that had a token in the original net will have one red token and one white token, those input states with no tokens originally, will have two white tokens. For every one of the legitimate initial markings of the transfer, there will be one and only one firing pattern across the states, and then the events in the transfer.

In the formalism, in terms of components, the two color representation has some obvious consequences. Here, I mention only one:

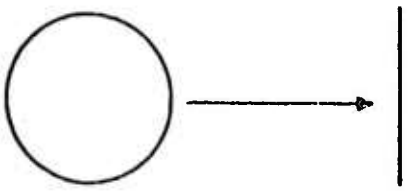
A state component, M_o, has exactly n tokens at all times, where n is equal to the number of mode paths in M_o. One of these tokens is red, the rest are white.

The high-level notation used for the vehicular flow model was actually based on the Quine transfer fired as a two-color marked graph. If one attempts to translate the model into an ordinary Petri Net, a synchronization problem arises. There is nothing in the net structure which in and of itself guarantees that all four parts at the transfer will exchange items. For instance, in a marking where E/W specifies that N meets E (and S meets W) simulation choice might perform the N/E exchange and proceed from there. Unless further synchronization is provided, E may once again arrive at the transfer, with another item. Such further synchronization is, of course, possible. In fact, it can be specified in terms of the higher level language by a suitable arrangement of role patterns for the communicating parts.

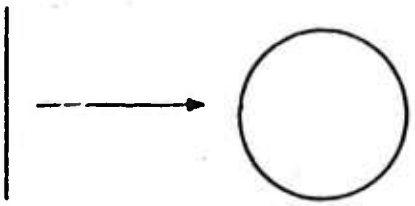
Appendix One

Petri Nets, Marked Graphs and State Machines

A Petri Net consists of states, which are represented by circles; events, which are presented by bars; and arrows, which connect the states and events with the following meanings:



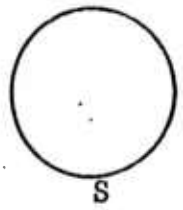
EVERY OCCURRENCE OF EVENT E ENDS ONE HOLDING OF STATE S



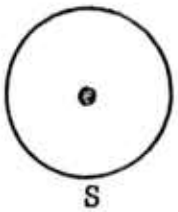
EVERY OCCURRENCE OF EVENT E BEGINS ONE HOLDING OF STATE S

We will sometimes use arrows to represent states or transitions which, if indicated explicitly, would have only one incoming arrow and one outgoing arrow. We also use a dot interchangeably with a bar to indicate an event. Thus:

If, in a Petri Net, we wish to represent a holding of a state S, we place a token on the corresponding state. The function which specifies the number of tokens on each state in a Petri Net is called the marking of the net.



no HOLDINGS OF S



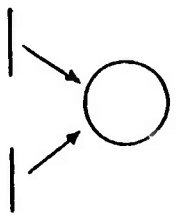
ONE HOLDING OF S

TWO HOLDINGS OF S

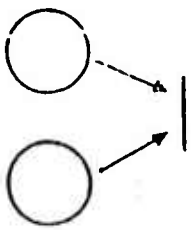
If every state which is input to an event has at least one holding, then an occurrence of that event can take place. This occurrence ends a certain set of holdings and begins a certain set of holdings. The occurrence of an event is represented in the Petri Net by removing one token from each of the states which point to the event and then adding one token to each of the states to which the event points.

Petri Nets which satisfy certain constraints have been examined extensively and considerable mathematics developed for them. The following four constraints are particularly important in this context. A Petri Net which satisfies 1 and 3 is called a marked graph. A Petri Net which satisfies 2 and 4 is called a state machine graph.

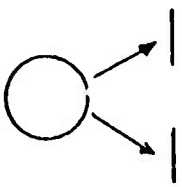
1: excludes



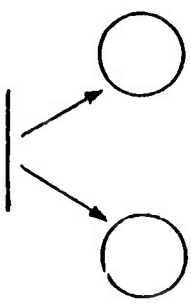
2: excludes



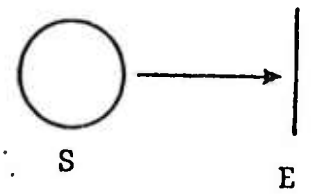
3: excludes



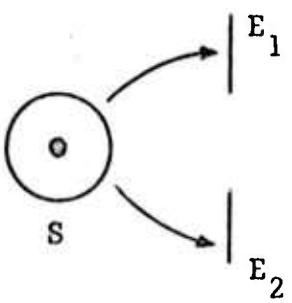
4: excludes



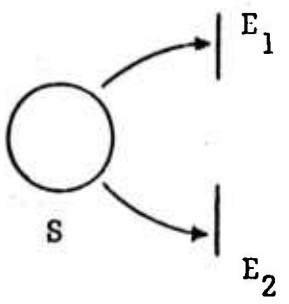
Following the meaning we have given to the structure



we see that

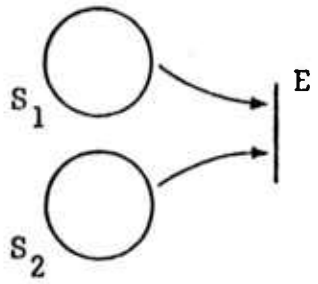


permits the occurrence of either event E_1 or event E_2, but not both. Thus the structure



in a Petri net represents a choice between two conflicting events.

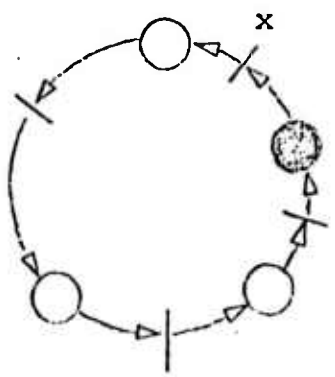
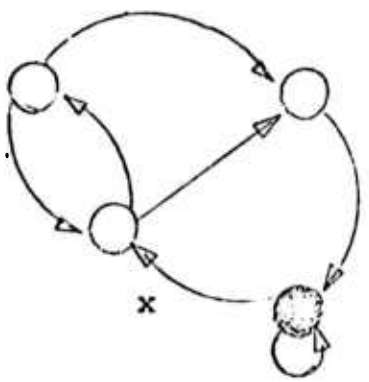
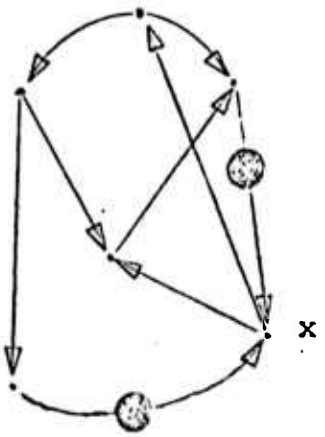
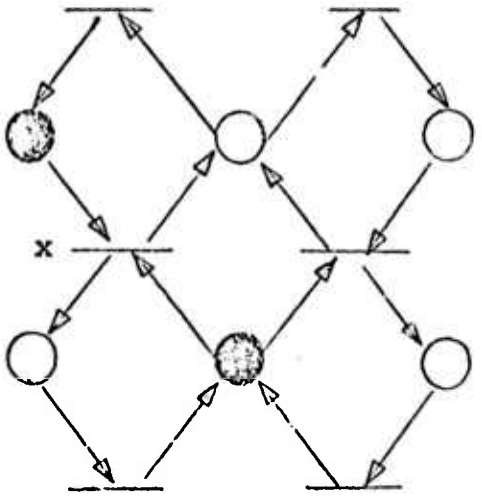
Following the meanings we have given we see



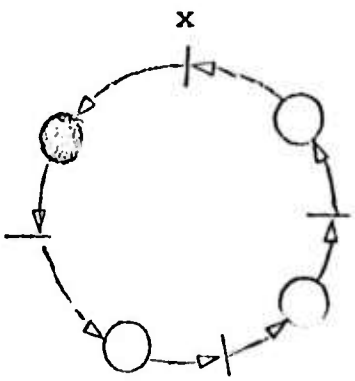
permits the occurrence of event E only when there are holdings of both S_1 and S_2 concurrently. Crudely stated, marked graphs allow concurrency but no conflict; state machines allow conflict but no concurrency.

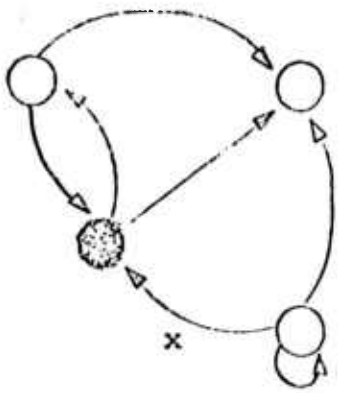
Using the shorthand conventions mentioned previously, every state in a marked graph may be represented by an arrow; every event in a state machine graph may be represented as an arrow.

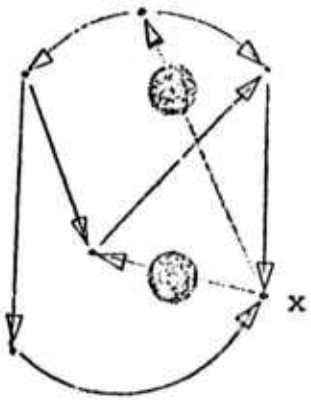
These special types of Petri Nets are illustrated in the following table:

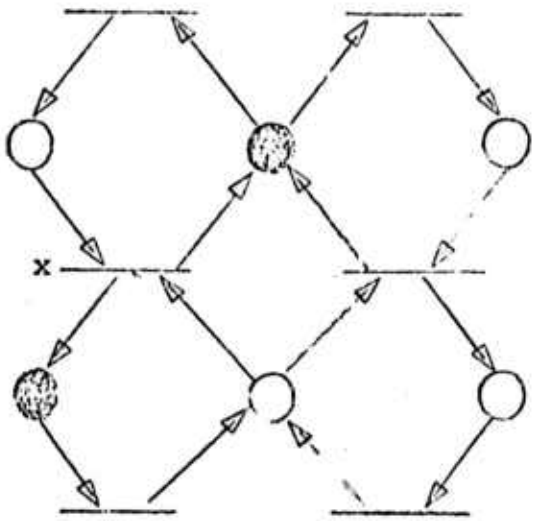
Is it a state machine graph? \ Is it a marked graph?	Yes	No
Yes		
No		

The marking or set of holdings change by the occurrence of events. Following the rules we have previously stated, all four of these Petri Nets have a new marking after the occurrence of the events marked X . Thus:









If M is a marking of a Petri Net and there exists a nonempty sequence of event occurrences possible starting at M, which results in another marking M', we say that M leads to M'. $\vec{M}$ designates the set of all markings that M leads to. $\overleftarrow{M}$ designates the set of all markings that lead to M. $\overleftrightarrow{M}$ is the union of $\vec{M}$ and $\overleftarrow{M}$ is called the marking class of M.

A marking M of a net is said to be live if, for any event e, there exists at least one sequence of event occurrences starting at M or any marking in $\vec{M}$, that includes e. This implies that if a marking is live, every event in the net can occur indefinitely many times.

A marking M is said to be safe if both M and every marking in $\vec{M}$ places at most one token on any state.

If a marked graph or state machine graph is strongly connected; i.e., there exists a directed path from any vertex to any vertex, then it has a live, safe marking class; a strongly connected state machine has only one live and safe marking class; any marking which places only one token on the net is a member of this class. A marked graph may have several marking classes. There exist procedures for enumerating and finding live, safe markings for a strongly connected graph.

In any marked graph, the number of tokens on any circuit can never be changed by event occurrences. This is so because any circuit must enter any vertex in the graph the same number of times that it exits from it. On the other hand, a vertex firing; i.e., event occurrence, takes one token from every entering arc and puts one token on every outgoing arc.

A circuit in a marked graph which has exactly one token on it is called a basic circuit. If we regard the events and states on a circuit as the elements of that circuit, then:

any two distinct elements on a basic circuit must "alternate".

Thus, if S_1 and S_2 are states on a basic circuit, after a holding of S_1 and before the next holding of S_1, there must be a holding of S_2.

A considerable amount of mathematics has been developed for marked graphs. This includes theorems and methods of calculation for liveness, achievability, maxima and minima vis a vis event occurrences and state holdings, safety, throughput rates, and so on. The interpretation of these results depends upon the semantics of the problem being modeled. For instance, the size of the largest marking could tell one how many processors would be required to perform a given cyclic task represented as a marked graph if no new timing restrictions are to result from the allocation of processors.

With marked graphs, one can explore the effects of various fixed cyclic schedules of allocation. An illustration of this is the modeling of production facilities, discussed in the literature (6). On the other hand, marked graphs cannot be used to represent and analyze the effect of resource pools. Nor can they model the effects of decisions with data dependent outcome or facilities in which each successive output requires the multiple traverse of various internal production cycles, some only a few times, and others many times. These all require the use of a different class of Petri Nets.

References

(1) Holt, A.W.:
Why We Need New Headware For Information Systems
(Paper presented at Schloss Birlinghoven, June 26, 1974)

(2) Shapiro, R.M.; Saint, H.:
The Representation of Algorithms
(Final Technical Report, RADC-TR-69-313, Vol. 2 (1969))

(3) Reference 2 above. Also refer to descriptions of the compiler developed by Massachusetts Computer Associates, Inc. for the ILLIAC IV

(4) Shapiro, R.M.; Saint, H.:
A New Approach to Optimization of Sequencing Decisions
(Annual Review in Automatic Programming 6, Part 5 (1970))
Also, Technical Report RADC-TR-68-305 (1963)

(5) See, for instance,
Holt, A.W.; Commoner, F.:
Events and Conditions
(Information Systems Theory Project, Applied Data Research, Inc. (1970)

(6) Holt, A.W.:
Production Schemata
(Applied Data Research, Inc. (1970))

(7) Holt, A.W.:
The Chinese Menu Axiom (1972)