GENERAL NET THEORY

C.A. Petri

Rapporteurs: Mr. E. Best
Dr. P.E. Lauer
Dr. M. Shields

Part 1

Those of you who attended this conference last year, may remember Anatol Holt's lecture 'Formal Methods in System Analysis'. My intention then, had been, this year, to supplement his lecture by three hours of concentrated blackboard mathematics, because I felt that nothing needed adding to Holt's lecture in terms of words and figures. But I now think that I ought to keep the mathematics to a minimum, both in order to give a general idea of the content of the theory, and to raise the entertainment value from negative to zero.

Net theory is a collection of new theoretical, mathematical and conceptional devices designed for the defining and solving of organisational problems. By that, I mean problems both inside and outside the computer; problems of specifying and implementing the co-operation of a large number of system components concurrently engaged in partially independent operations and activities. I should warn you that all my examples are exceedingly, perhaps excessively, simple. But my aim is to convey the basic ideas and some of the background and not to explain a full-fledged application example.

Net theory is not, of course, the work of one single person, and in connection with the material I am going to present I would like, in particular, to acknowledge the work of Anatol Holt, Fred Commoner, and my colleagues Hartmann Genrich and Kurt Lautenbach. To begin with, I think I should explain the purpose of net theory because it does not seem to be all that clearly understood. Possibly, the general impression is that its purpose is mainly descriptive. This, however, is not the case, but rather to supply us with

descriptive, deductive and conceptual devices:

- descriptive devices for demonstrating the structure of systems and of processes supported by a system, in terms of axiomatically introduced concepts;
- deductive devices for solving application problems such as:

  synchronisation problems, concurrency problems, problems involving mutual exclusion, conflict, arbitration, sequentialization, safety, problems of deadlock-avoidance and of endless-loop avoidance, problems in asynchronous switching logic, and last but not least, problems arising in an area, not generally known of as yet, called formal pragmatics, in which we are concerned with questions of the form 'What, precisely, do we do?', as opposed to formal semantics in which we are concerned with questions of the form 'What, precisely, does it mean?';
- conceptual devices producing precise concepts on many levels or for promoting the communication between the computer expert and other people; I see this as being a main point of General Net theory. We can communicate between ourselves very well, but it is difficult to explain computers to 'innocent' people. As a conceptual device, net theory should promote this communication and provide means for introducing new concepts, in a precise, but nevertheless, easily visualiseable way, hence the importance of graph-theoretical methods, and of the idea of the 'token game' played by many independent actors.

Net theory is not primarily a mathematical device, rather it is accompanied by a simple graphical means of expression consisting, at the basic level, of four symbols only.

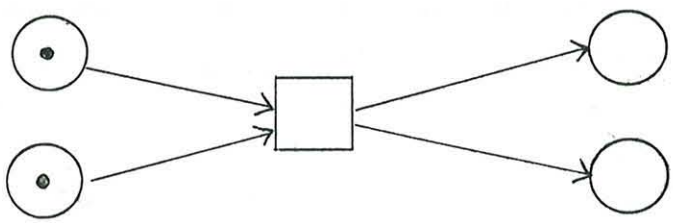


Figure 1

In this example (of a net), the symbols used are only squares, circles, arrows and little dots. For the time being you might think of it as describing a computer with two input files and two output files; the input files are present and connected to the computer while the output files do not yet exist.

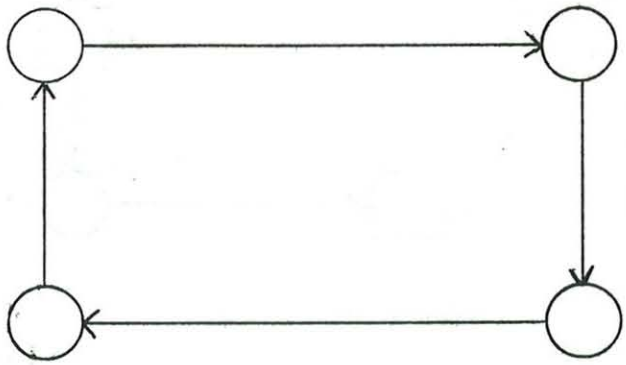


Figure 2a

To start at a very low level, we might compare the graphical way of expressing nets to more usual graphical tools; consider, for example, the simple state-transition diagram of Figures 2a and 2b describing a system of four states and the respective transitions. Instead of denoting the transitions by arrows, we introduce special symbols in order to bestow individual existence on them and to be able to append more than just two arrows to them, possibly leading to or from other states. We use a dot to denote the holding of a particular condition at a specific point of time:

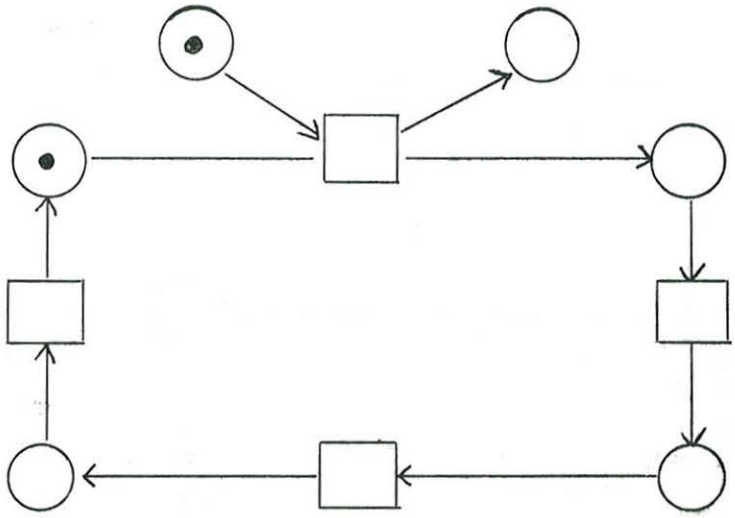


Figure 2b

Thus, in this notation, we have the following four types of symbols:

S: ◯ state elements (also called places if they can contain more than just one token);
S contains symbols denoting 'supply stocks', and items in these supply stocks are represented by tokens or markers (dots) on the state elements.

T: □ , transition elements, representing for example, processes; elementary events, of which processes are built; alterations in the holdings of conditions; transportations; transformations and so on.

F: ⟶ , flow relation, no longer denoting a transition itself but only the relation between a state and a transition; f might be read 'from ... to ...'

● , tokens, countable items, resources of any kind.

Thus, a net is defined to be a triple (S,T,F) where

$S \cap T = \emptyset$ (state elements and transition elements are disjoint sets),

$S \cup T$ = field (F) (there are neither unconnected state elements nor transition elements)

$F \neq \emptyset$ (nets cannot be empty)

$F \subseteq S \times T \cup T \times S$ (the flow relation holds only between state elements and transition elements or vice-versa.

A marked net is a net with a distribution of tokens over the places.

These concepts were introduced on the principle that we should take over primitive ideas from areas such as the production of goods, where we can observe and physically handle a distribution of items, rather than from abstract notions of computation, or of processes supposed to be occurring in people's minds, and in computers as a delegated type of mental activity.

Figure 3 shows several production activities, denoted by boxes, and several places in which resources can reside. We could imagine ourselves as being the worker in box A using one of the screws accessible to him (accessibility is denoted by arrows), one nut and one sheet of each type, and, with the help of the screwdriver (which he returns after use), producing one object of the type shown in place Z. No use is made of any of what are generally held to be 'primitive' concepts in computing such as assignment of values.

In this figure we can already observe the main phenomena with which Special Net Theory is concerned, for example, conflict and contact. All such activities can be described in terms of a simple game. I shall restrict myself to condition - event - nets, where the places can contain at most one token: the holding of the condition being denoted by the presence of a token. Everything which can be composed out of squares, circles and arrows in the manner of Figure 4, might be thought of as a gameboard. Each square and circle may accommodate respectively one player and one token. The players act independently except in so far as they are affected by the presence of tokens on the circles.

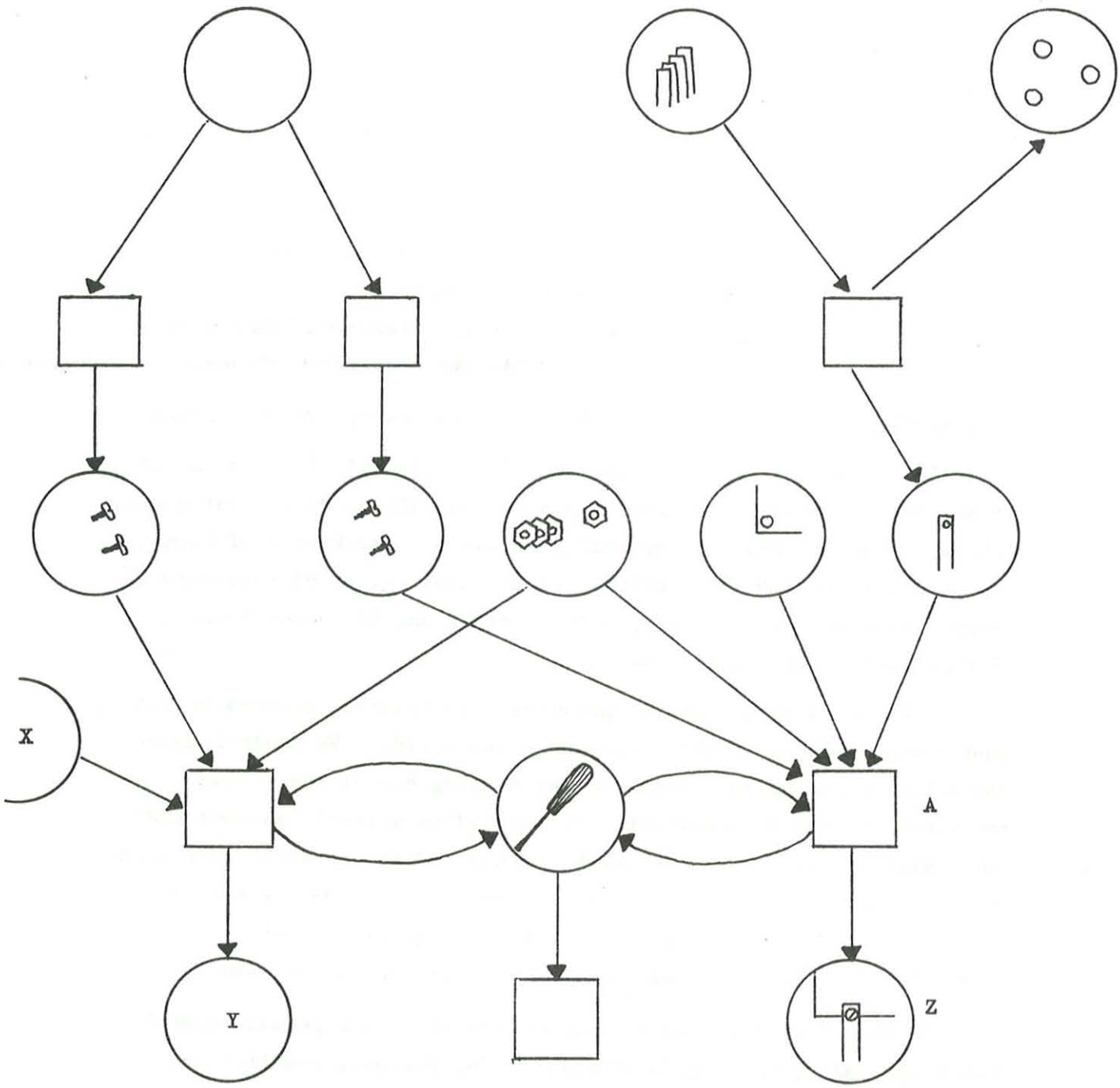


Figure 3

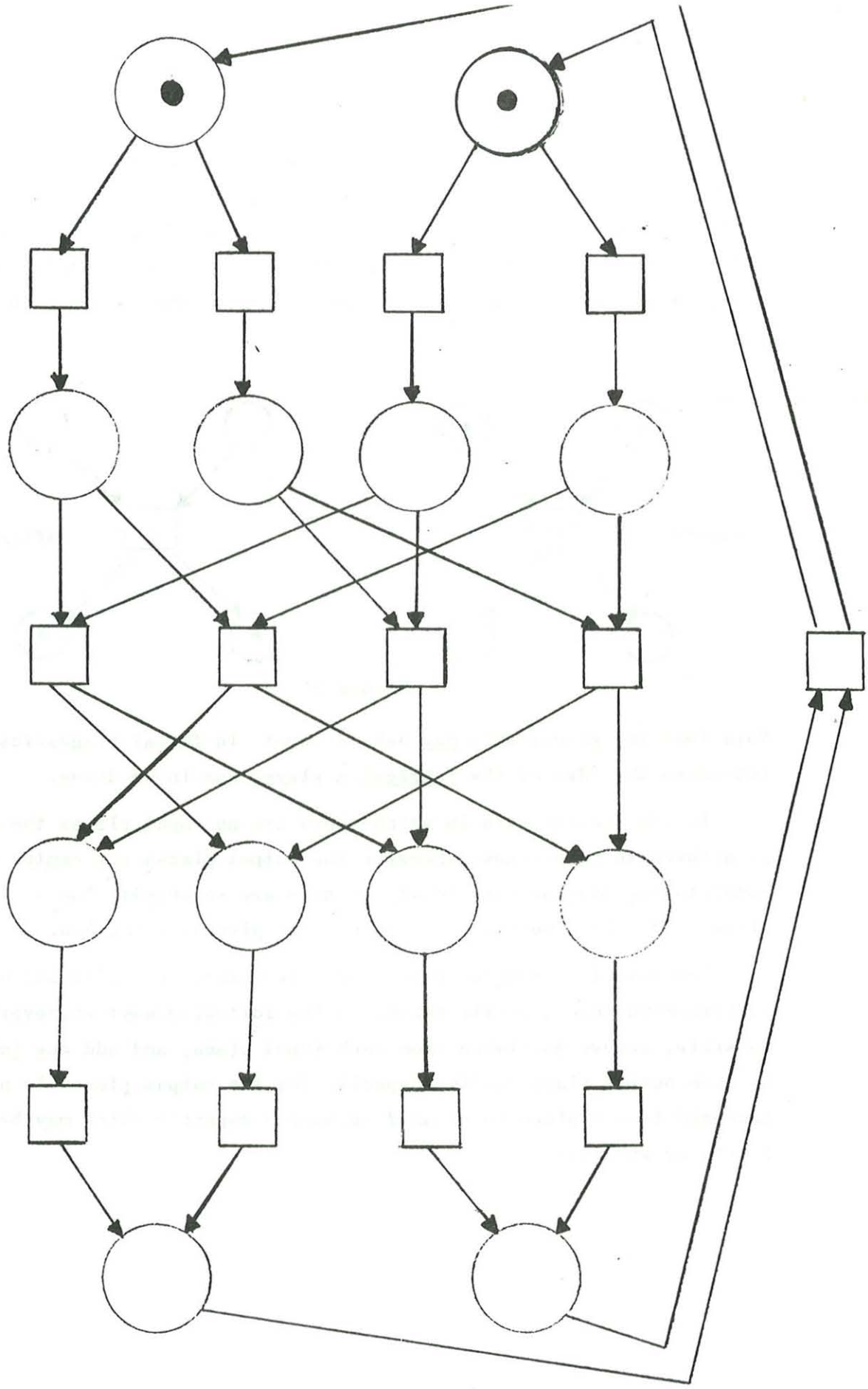


Figure 4

The rule for this game should be kept very simple. It is expressed by an axiom and I am going to explain it in every-day terms: a player such as x residing in the box in Figure 5 obeys the following rule: 'if all of your inputs are full and all of your outputs are empty, then you may empty your inputs and fill your outputs, to score 1'.

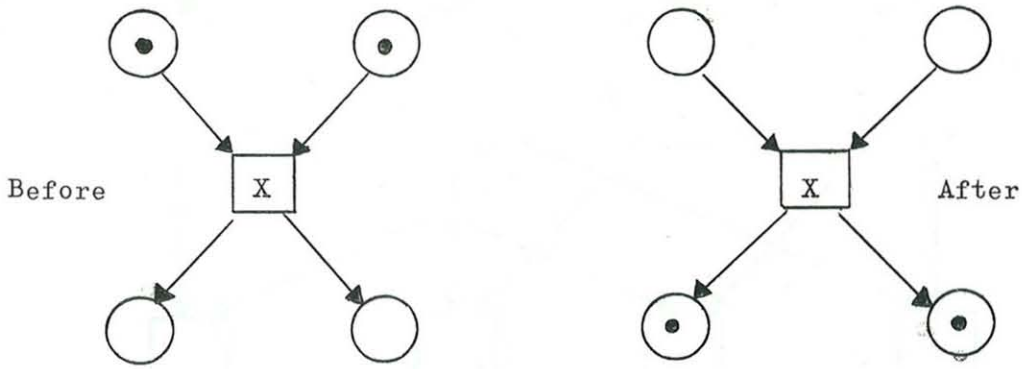


Figure 5

Note that the player only <u>may</u> make a move: In formal pragmatics we introduce the idea of the <u>interest</u> a player has in so doing.

In the special case in which there are no input places the player is allowed to make a move whenever the output places are empty, and consistently for the case in which there are no output places, he is allowed to make a move whenever the input places are marked.

The rule is extended to nets in which places are allowed to contain more than a single token, in the following way: whenever possible, remove one token from each input place, and add one token to each output place, if the capacity for the output places is not exceeded (every place is assumed to have a capacity which may be finite or infinite).

I would like to show two simple application examples. The first one appeared in 'Jurimentrics Journal' where the idea of such a game is used to define in a precise way something which could not be expressed in the ordinary language of the lawyers. In the example, it was not clear whether three particular activities had to be thought of as concurrent or as mutually exclusive. In the paper, it is shown by means of a gameboard, that the intention of whoever wrote the particular piece of law was to mutually exclude the three possibilities.

As an example from computer science, consider Figure 6; the part of the net shown in bold is supposed to be a gate. It appeared on the cover of a volume of CACM (vol. 16, No. 8, August 1973). The transition labelled ACT is supposed to be capable of firing only whenever there is a token on the place labelled GATE; this token can be brought into place by firing 'ON' and taken away by firing 'OFF'. When considered as a condition-event-net, however, the dotted additions show that this is not the correct basic specification of a gate: according to the rule, ACT would never be able to fire, since either one of its input places is unmarked or one of its output places is marked. This is thus rather a high-level description and we are still left with the task of decomposing it into basic specifications.

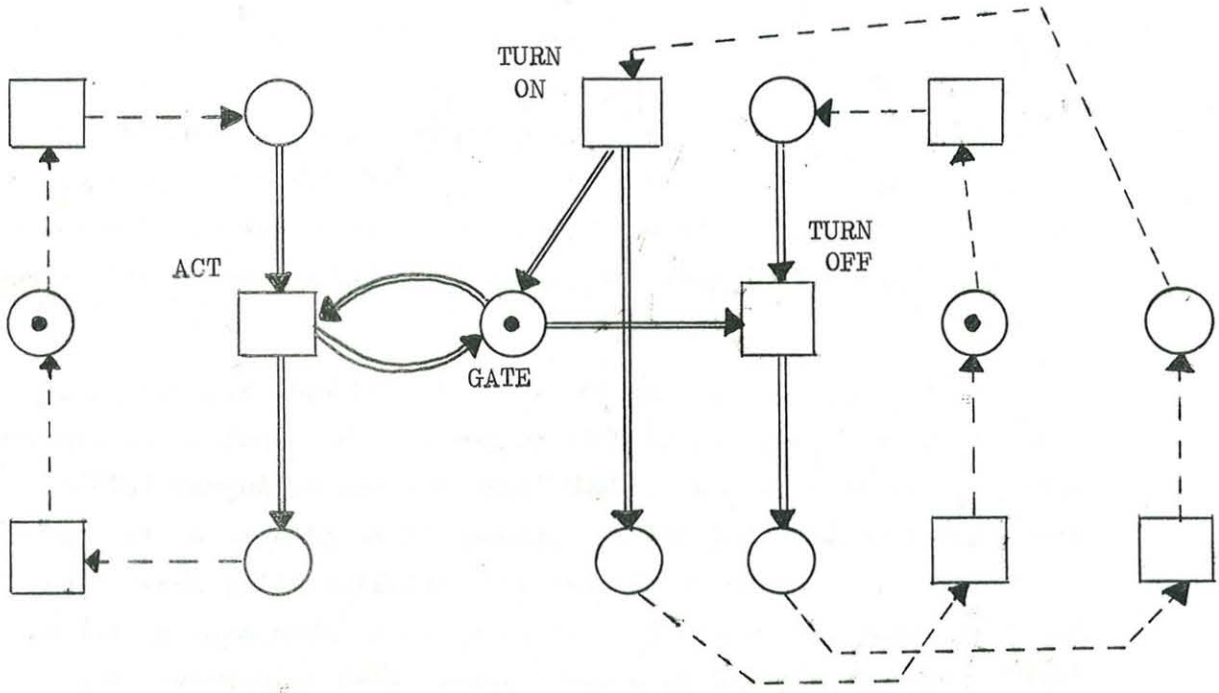


Figure 6

Most present applications are in Special Net Theory as opposed to General Net Theory because the former has been in existence for a longer time. Special Net Theory is precisely the theory of those games which I defined earlier with their associated rule. General Net Theory is the theory of transforming concepts and results of Special Net Theory into concepts and results on higher (and lower) levels of system description, by means of certain kinds of net-mappings.

Some present and intended applications of net theory go back to the year 1959 when SPLIT and WAIT were proposed as Programming Language primitives. A little theory was developed in this connection with the intention of showing that SPLIT and WAIT (being output and input of what I call transition) are the only language constructs which are theoretically needed. A late outcome of this piece of theory was

used for the ILLIAC IV Fortran Compiler, which used earlier results about liveness and safeness of nets.

By 1965 there was a full linear algebraic theory of Concurrent Processes (equivalent to Vector Addition Systems) and the period 1962-74 saw the development of results in switching logic (especially asynchronous) and of some applications in computer architecture. In 1974 Suhas Patil (MIT) constructed a direct implementation of transition nets in terms of TTL; this is supposed to be useful for process control. There are also a large number of papers dealing with Operation Systems problems (deadlocks, safety, looping, interrupts and so on) and with control problems applied in process control this year.

Since 1971 there have been applications in interdisciplinary communication in such areas as Physics, Logic, Chemistry and Jurisprudence. For example, the US Department of Justice has used Net Theory to define the precise meaning of pieces of legislation. Since 1972 a theory of communication and organisation is in the state of being developed. Results have been applied in the Mathematical Foundation of Computer Science, in Norms and Standards (for example, the language of Net Theory is used by a Standards Committee in West Germany for defining the basic concepts of Operating Systems), in education (mainly in teaching logic and Operating Systems); activities in my own institute have led to a formalisation of pragmatics, and, since 1974, to the origination of a theory of communication disciplines, which I shall discuss later.

What have all these things to do with computer science? I have already mentioned one purpose of net theory, that of making it possible to introduce concepts in a precise way. Some of the conceptual levels in the field of computing are demonstrated in Figure 7. Thus we have, to speak informally, high level concepts such as data bases, computer architecture, operating systems, files, tasks and so on. All these concepts are supposed to be firmly based and well-founded on the idea of algorithm; algorithms, in turn, are supposed

to be based on what one might call a 'cartesian product' of logic and time, which is an abstraction of AND-gates, delays, locks and so on which are themselves supposed to be correctly implemented by transistors, diodes, inverter-amplifiers and whatever else we have.

Interests (of individuals, groups)	Restrictions (physical, legal, economic...)
...	...

Offices, Agencies
Activities

Channels
Roles

(as in organisation, administration)

Data bases,	Computer Architecture
...	
Operating Systems	
Files, Tasks	
...	
(Algorithms:) if, do, :=, identifier	
(logic x time:) AND-gates, delays, locks... transistors, diodes, inv.-amplifiers...	

3. Information flow graphs: flux, influence (1967)
2. Transition nets: repetition, alternative action; synchrony, enlogy (1962-74)
1. Occurrence nets: partial order in time (1968)
0. Concurrency structure (1976)

Conceptual Levels above, within, and below computer science

Figure 7

We also have conceptual levels outside the computer which are certainly higher, such as the interests of groups or individuals, restrictions of legal, economic and other kinds, for which we have almost no formalism. On the slightly lower level of administration and organisation we have things such as offices and channels, activities and roles; the topics A. Holt talked about last year. The higher we look in this hierarchy the more diffuse and less formal and precise our descriptions tend to be.

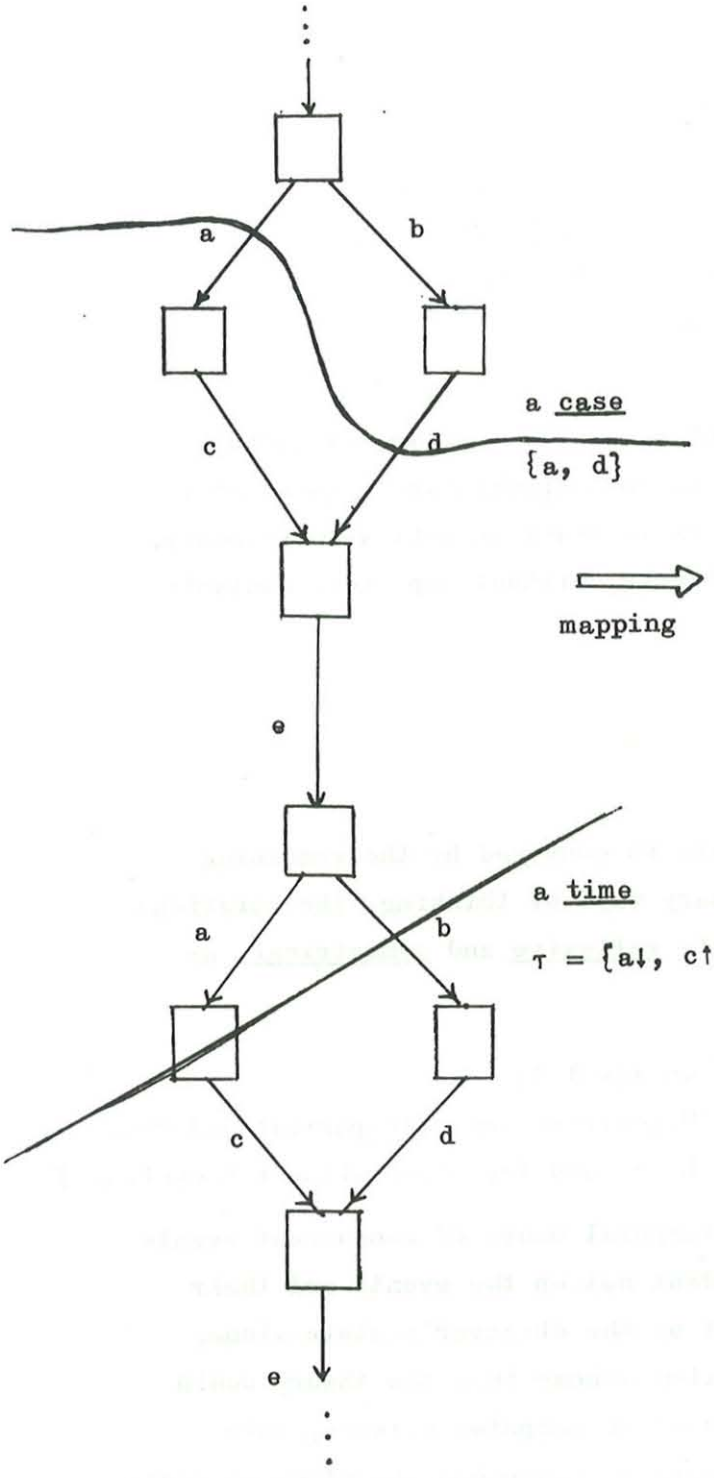
General Net Theory was conceived with the intention of achieving higher precision and the possibility of formal deduction in the upper areas of Figure 7 to the extent that such precision is welcome. In order to make progress in the direction of more precision and formal methods in the higher levels it turned out to be necessary to question the foundations of the supposedly lowest levels, below which nothing was thought to exist. I think that on the basis of results of General Net Theory we can certainly distinguish at least (I would rather like to say, precisely) four levels below the level of 'logic x time', and that, as a consequence, this marriage of logic and time is not possible because logic and time are sisters. Both propositional logic (or boolean algebra or whatever form you give it) and a concept of time (in so far as we have one) arise in stepping upwards from level 0 to level 2. I shall go into more details about levels 0-3 later, but for the moment, a short outline will suffice.

The lowest level 0 is only of theoretical importance; its use is in demonstrating that everything on the higher levels can be formally constructed using nothing other than the relation of concurrency. At level 1 we have a partial ordering in time of condition and event occurrences, first described by A. Holt in 1968. Level 2 is the level of transition nets, which I explained earlier in terms of a game; here we do have repetition and alternative action, and we have two structures which are of practical interest: namely the synchronic structure and the so-called enlogic structure containing all factually and logically valid statements about a transition net. And there

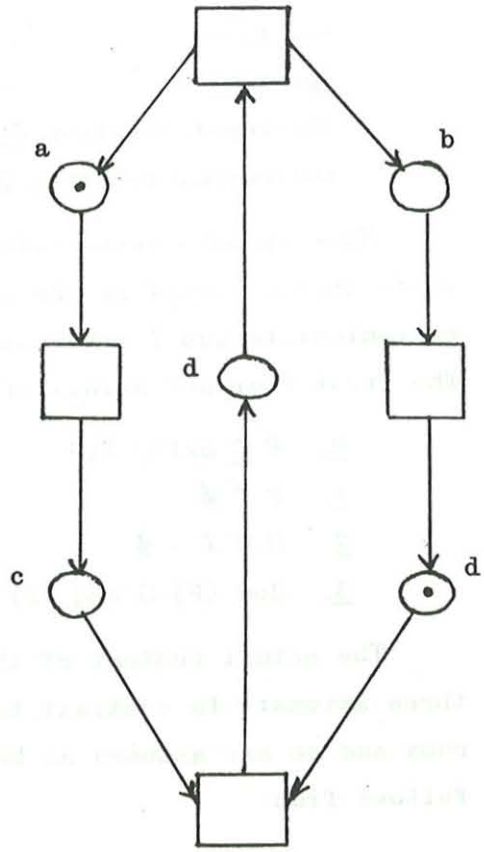
is another intermediate level, level 3, of information flow graphs; in 1967 two phenomena, which might be called flux and influence were detected. In most practical problems it is unlikely that one would need to carry an analysis through the whole range of levels, rather one will be moving between two adjacent levels. And this, I think, can better be demonstrated on the lower rather than the higher levels, for in the former we have full formality, while in the latter we would have the additional task of first achieving formality from informal descriptions. An example is shown in Figure 8: on the left-hand side we have a process described by an occurrence net; the occurrences might be thought of as activities performed by players who are never able to make more than one move. Thus every occurrence is unique. The states involved are omitted for clarity. A net of this type needs no marking; the marking could be deduced in the following way: everything not connected by a directed sequence of arrows is defined to be concurrent, and a maximal set of concurrent items can be thought of as being marked. Such a collection might be called a certain time, for instance the time τ shown, consisting of the ending of condition a, the beginning of condition c and the holding of condition b. A time containing only holdings of conditions might be called a case; for instance containing the holdings of conditions a and d, as shown in Figure 8. The labelling of the net indicates which occurrences are holdings of the same condition; this labelling implies a mapping (net morphism) yielding the net on the right-hand side, where the case $\{a,d\}$ is indicated by tokens.

By way of a transition to the more formal part of my lecture, I would like to give you an outline of General Net Theory. We have four primitive concepts.

state elements S,
transition elements T,
flow direction relation F and
coexistence relation coex.



Level 1, with intended mapping indicated by labels: A partial ordering of Event Occurrences



Level 2:
A transition net.

Figure 8

The latter becomes, at the lowest level, the concurrency relation co. We also have some defined concepts:

the set $X := S \cup T$ of elements under consideration,
the partitioning connecting relation $P := (F \cup \overleftarrow{F}) \cap (S \times T)$ indicating 'proximity' without reference to orientation,
the input relation $Z := P \cap F$ and
the output relation $Q := P \cap \overleftarrow{F}$.

Then we need seven axioms which are sufficient to set up the whole theory except in the case of certain applications, where it is convenient to add further assumptions in order to achieve more power. The first four are axioms of formal nets, without empirical content:

0. $F \subseteq S \times T \cup T \times S$
1. $F \neq \emptyset$
2. $S \cap T = \emptyset$
3. $\mathrm{dom}(F) \cup \mathrm{cod}(F) = X$

The actual content of the theory is conveyed by the remaining three axioms: In contrast to ordinary ways of thinking, the relations coex and co are assumed as being only reflexive and symmetrical, as follows from

4. $co = X \times X - (FF^* \cup \overleftrightarrow{FF}^*)$ on level 1.
("Level 1" is short for 'Occurrence net with partial ordering of occurrences generated by F, and therefore without F-cycles'.)

Coexistence is not transitive; the temporal order of concurrent events is known (from Physics) to be dependent not on the events and their objective relation to each other but on the observer's state alone. If we assumed transitivity in our axiom scheme then the theory would verge on the trivial. For the purposes of computer science, this intransitivity mainly reflects the possible independence of occurrences which are not actions of the same physical component in the computer or the system surrounding it.

For event structures we have two additional axioms (events being members of class T, the class of transition elements).

5. The Extensionality axiom

means, that the identity of an event is characterised only by those changes it effects in the world and by nothing else; not for instance, by side conditions on which the event may depend but the holding of which is not altered by the occurrence. For example, the truth value of the boolean expression involved in the execution of an 'if - statement' is ordinarily not changed by the execution of the 'if - statement'. Therefore the 'if - statement' is a complex structure, and not a primitive one in terms of General Net Theory. It must be decomposed in order to be operationally defined; we may then distinguish several types of 'if - statements' and tell precisely by what they are distinguished. The extenionality axiom means that every event is a coincidence class of changes; this implies that the intersection between F and $\overleftarrow{F}$ is empty ($F \cap \overleftarrow{F} = \emptyset$). Thus, in the Production Scheme example, the process in which a screwdriver is used but put back in its place is not an elementary event as defined here; again, it must be decomposed in order to define it precisely. Finally, we have:

6. the axiom of 'Relative indeterminism' (I might just as well have called it 'relative determinism'): If a transition net contains a conflict, then the system described by it possesses a non-empty environment. This axiom is not really necessary for developing the formalism but for defining the scope of the formalism. We do not really want to develop a theory of games but a theory of information flow, and in this connection I have the feeling that computer science is, compared to physics, in a kind of pre-Newtonian stage, since we do not know the natural laws of information flow. To remedy this defect slightly, axiom 6 is proposed; it says that information flow is the same thing as conflict resolution. Therefore the reverse of conflict resolution is the same thing as the reverse of information flow; this is quite an abstract and seemingly nonsensical

thing to say but it might encourage you to give the matter a little thought. A precise definition of conflict will be given later. This axiom concentrates the interest on conflict free nets, which are quite easily described. Nets with conflict or with higher order phenomena (confusion, skew events) are very difficult to deal with, without this axiom. Axiom 6 also means that whenever we encounter a case of conflict, a non-determinate situation, in the description of a system it is because the description of the environment is absent or not sufficiently detailed. Thus the existence of conflicts is traceable back to the cut between system and environment; it is only in our minds that we observe the conflict, because we have arbitrarily made the distinction between an 'inside' and an 'outside': if we describe the environment in more detail, then the conflict would disappear.

One mathematical description of General Net Theory is in terms of the category of nets. Of course nobody wanting to apply the theory needs to know about category theory, but things become very easy and concise indeed in terms of category theory. The objects of our category are all nets (tuples (S,T,F), (S,T,Z,Q) of (X,P,F), equivalently). Morphisms are mappings from net to net preserving the relations 'P ∪ identity' and 'F ∪ identity'; this mainly means that mappings are continuous in the sense of topology, and direction-preserving. Morphisms preserve those purely formal relations but transform the relation co (and whatever can be derived from this relation) into more complex relations in higher levels; these transformations are the objects of the formal study.

Part 2

I now want to go on to talk about General Net Theory, restricting myself, for reasons of time, to one level only, that of condition-event nets, the fundamentals of which I have already described. There are four basic phenomena that I propose to discuss and which I call concession, contact, conflict and confusion (Figure 9); the theory of these phenomena is the content of Net Theory on the level of transition nets. The tools of Net Theory I shall be talking about are linear algebra, applied to condition-event nets and to place-transition nets; the formalism of net completion, which leads to the subject of system invariants; and continuous mappings, which allow us to move between the different conceptual levels (Figure 7).

I won't go into details on this last point here, but I shall indicate the differences between net theory as a systems theory on this low level and other systems theories. Basically, the difference is that other systems theories do not set out to deal explicitly with concurrency. By concurrency I mean a binary relation in the set of conditions and events of a net; two elements are concurrent when they might be considered 'simultaneous' according to some frame of reference. This binary relation is certainly reflexive and symmetric but it is not transitive; to assume it to be so would be to idealise in a way which is not acceptable when we wish to describe a system with a high degree of precision.

On a higher level there is a difference between net theory and other systems theories in the treatment of measurement data, based on a distinction between 'analogue' and 'digital' and between 'continuous' and 'discrete' models; this distinction is not made in net theory (for example, a net of open intervals S and of points T can be associated with the continuum R of real numbers). On a lower level there is the distinction that we do not deal with side conditions of elementary events. This is usually considered an inadequacy of net theory because it seems to imply that we are incapable of talking about the conditional execution (= occurrence) of events. The fact

is, however, that conditional execution is a higher level concept, which may be precisely defined (in several ways with different meanings) in terms of lower level concepts involving only pre- and post-conditions but not side-conditions. To attempt to use side-conditions in order to discuss conditional execution would be meaningless in terms of transition nets of level 2. The absence of side-conditions is thus not an inadequacy of net theory but rather one of its main points.

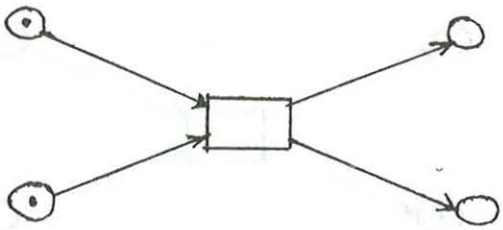
Returning to the axioms; zero to three say what a net is; they describe how the 'game board' must be constructed; they provide a language in which we may talk about the relation co (the concurrency relation), which constitutes the content of the theory. The extensionality axiom (five) restricts elementary processes - events - to consist of changes of conditions and nothing else; in this context we cannot meaningfully talk about side-conditions, i.e. conditions which are pre- and post-conditions of the same event. Finally we have the axiom of Relative Indeterminism, which says that conflict resolution is the same thing as information flow.

We are now in a position to consider the four fundamental phenomena in condition - event nets (Figure 9). A transition is said to have concession (Figure 9 (a)) when all its input places are marked and none of its output places are marked. A transition may fire in exactly this situation although it is not committed to do so. If we think for a moment of a player sitting in box 'A', it is possible that some other player might be capable of removing the token at p1 or p2. In other words 'A' may be able to lose concession without actually having fired.

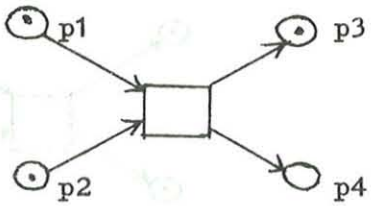
In the situation of contact (Figure 9 (b)) a transition has all its input places marked but has also some output places marked.

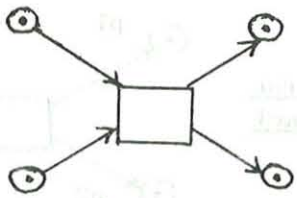
The fundamental situations in nets of conditions and events

a) Concession
('fireability')

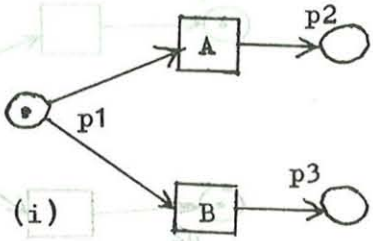


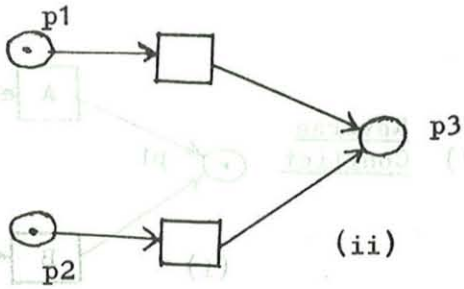
b) Contact



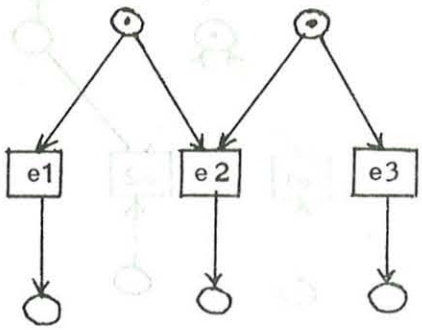


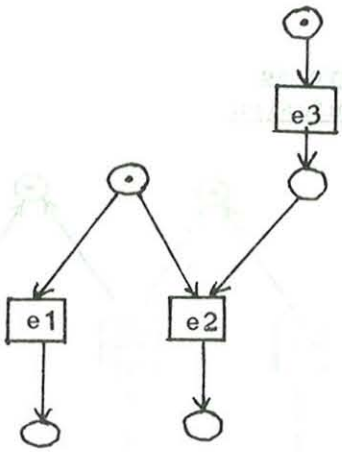
c) Conflict





d) Confusion



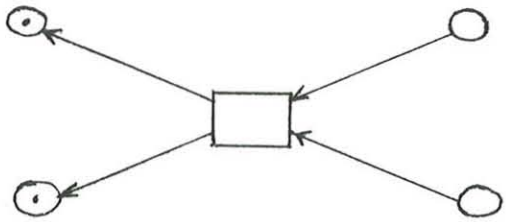


(i) 'symmetric'

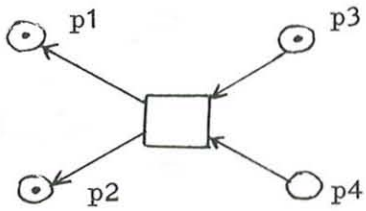
(ii) 'asymmetric'

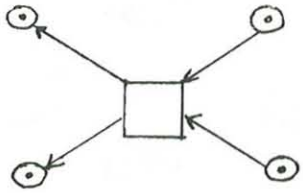
Figure 9

a') Reverse Concession

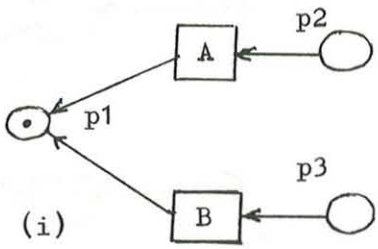


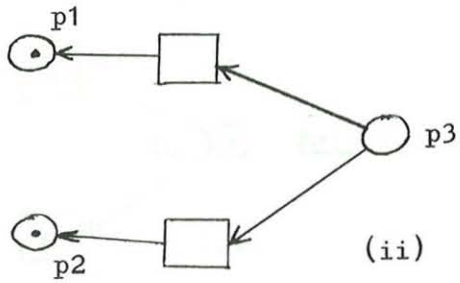
b') Reverse Contact



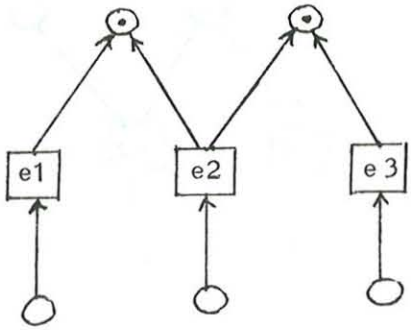


c') Reverse Conflict

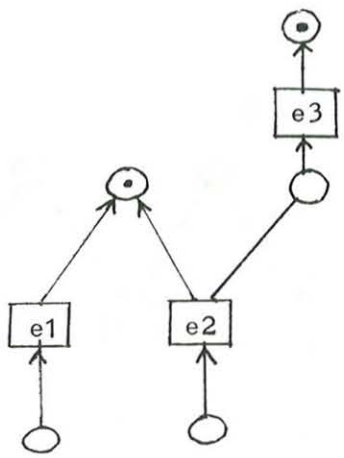




d') Reverse Confusion



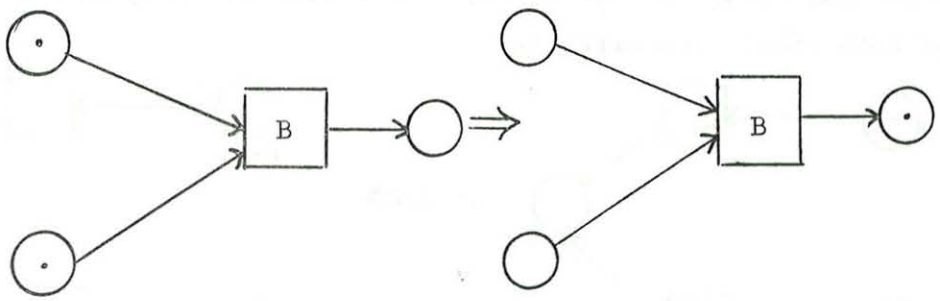
(i) 'symmetric'



(ii) 'asymmetric'

Figure 9 continued

It might appear at first sight that a transition in the situation of contact ought to have concession; after all, each of its pre-conditions hold so why should not the transition be able to fire? However, what would be the consequences of the firing of 'A' (Figure 9 (b))? Either p3 would as a result contain two tokens - which would be absurd, since p3 represents a condition, and what could its containing two tokens possibly mean? - or else the tokens on p1 and p2 would disappear and no new token would be placed on p3. But this would mean (by the extensionality axiom) that an event distinct from 'A' had taken place, namely an event B $\neq$ A, connecting p1 and p2 to p4 only:

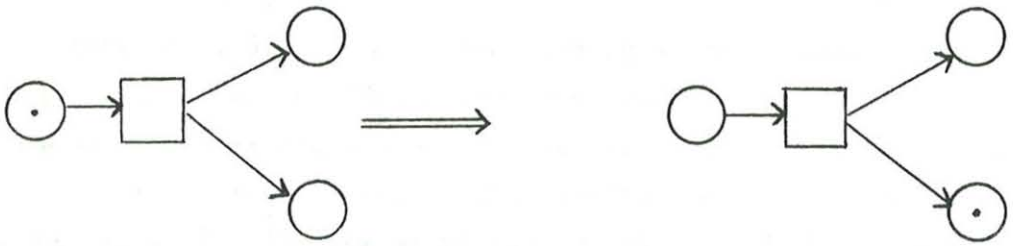


So if B is not explicitly contained in the set of events, the change effected by it cannot occur. There is no such thing as a partial occurrence of 'A'.

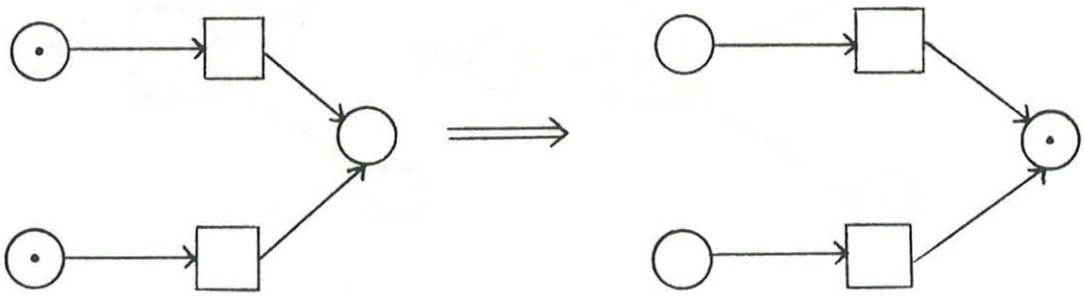
We conclude from the extensionality axiom that no transition in contact may fire, that is, contact and concession are mutually exclusive.

Thirdly we have the situation of <u>conflict</u> (Figure 9 (c)), in which two conceded events have either a common pre-condition or a common post-condition. As in the contact situation one might be tempted to argue that since each event has concession, each should be allowed to execute; as in the contact situation, we show that the simultaneous firing of the two events would be an infringement of

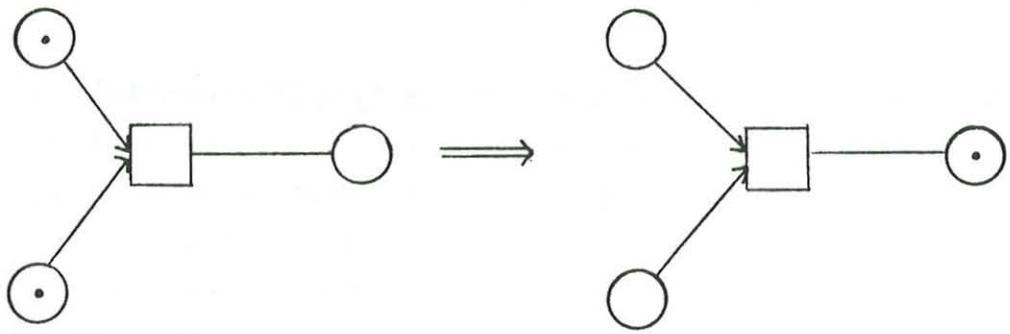
the extensionality axiom. Thus in (Figure 9 (c) (i)) for A and B to occur simultaneously would not be the same as the situation defined by (i) but rather an occurrence of an event $C \neq A,B$:



(ii) demonstrates a situation in which two events have a common post-condition. We may argue, similarly, that a hypothetical firing of both events concurrently:



infringes the extensionality axiom, since its corresponding change of state corresponds rather to an occurrence of the form



This is immediately obvious when we consider production schemes, where tokens represent material objects which may be transferred either one way or the other but not both. Conflicts have to be resolved, they cannot be ignored.

The final situation of interest is that of confusion (Figure 9 (d)). As you can see, there are two types: a symmetrical and an asymmetrical. . In the former, three events are involved all of which have concession and two pairs of which, (e1, e2) and (e2, e3), are in conflict. e1 and e3 are not in conflict, however; they are entirely independent and could, indeed, happen concurrently. In a situation of confusion it is possible that a conceded event may get out of conflict (i) or into conflict (ii) and retain concession. This is perhaps a little abstract but there is a simple example - the usual solution to the mutual exclusion problem - which I shall show you later and which does involve confusion. (Figure 14).

Axiom six (Relative indeterminism) suggests that in a confusion situation information flow is not objective because in such a case it would be up to the observer whether the occurrence of a particular event constituted a conflict resolution or not, that is to say whether information flow had taken place at all. Thus, in Figure 9 (d)(i) e1 is in conflict with e2. When e3 is executed, e1 gets out of conflict without either itself or its 'conflict partner' e2 having occurred; has there been a flow of information or not? This situation is indeed confused and unfortunately it comes up quite frequently. We should think of confusion within the context of axiom six as follows: when we encounter such a situation we may conclude that we have drawn the boundary between the system and its environment in an awkward manner and that we should draw it somewhere else in order to reduce confusion to mere conflict resolution. It follows that we really don't need to set up a theory of confusion - which would indeed be difficult to do. It has, in fact, been tried more than once and seems almost impossible.

Having described the four basic phenomena the theory of which is the content of net theory on the level of transition nets we now go on to look at the subject of algebraic descriptions (Figure 10).

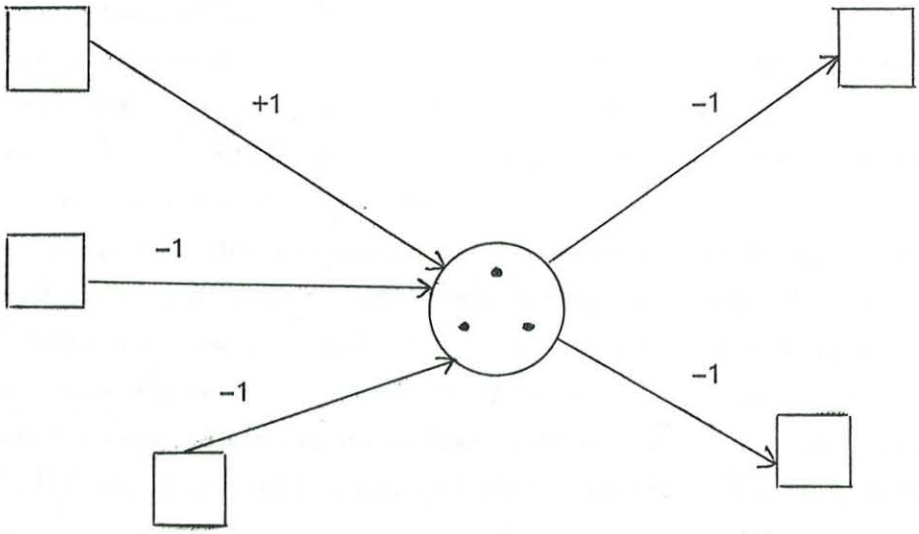


Figure 10

If the transition rule, for place-transition nets, is formulated as I have done earlier, then processes in nets may be described by a simple algebraic formalism. This means that what is possible in the theory of net processes is very clear indeed because the theory of linear integer equations and inequalities such as the ones that result from our algebraic formulation is old and well developed.

An algebraic description of processes in nets may be achieved in the following manner: to each state element assign a number s denoting the number of tokens residing in it - thus in the case in which a state element represents a condition, s must be either 0 or 1. We may also set up the connectivity matrix, C, of the graph of the net; in this matrix rows correspond to state elements and columns to transition elements and

$$C_{ij} = \begin{cases} -1 \text{ if } (p_i, t_j) \in Z \text{ (an arrow runs from } p_i \text{ to } t_j) \\ 1 \text{ if } (p_i, t_j) \in Q \text{ (an arrow runs from } t_j \text{ to } p_i) \\ 0 \text{ otherwise.} \end{cases}$$

The relation between the quantity changes of s, the number of tokens, 'tokens(p_i)', on each place p_i and the number of occurrences 'occ(t_k)' of each event t_k from the beginning to the end of any process is given by the formula

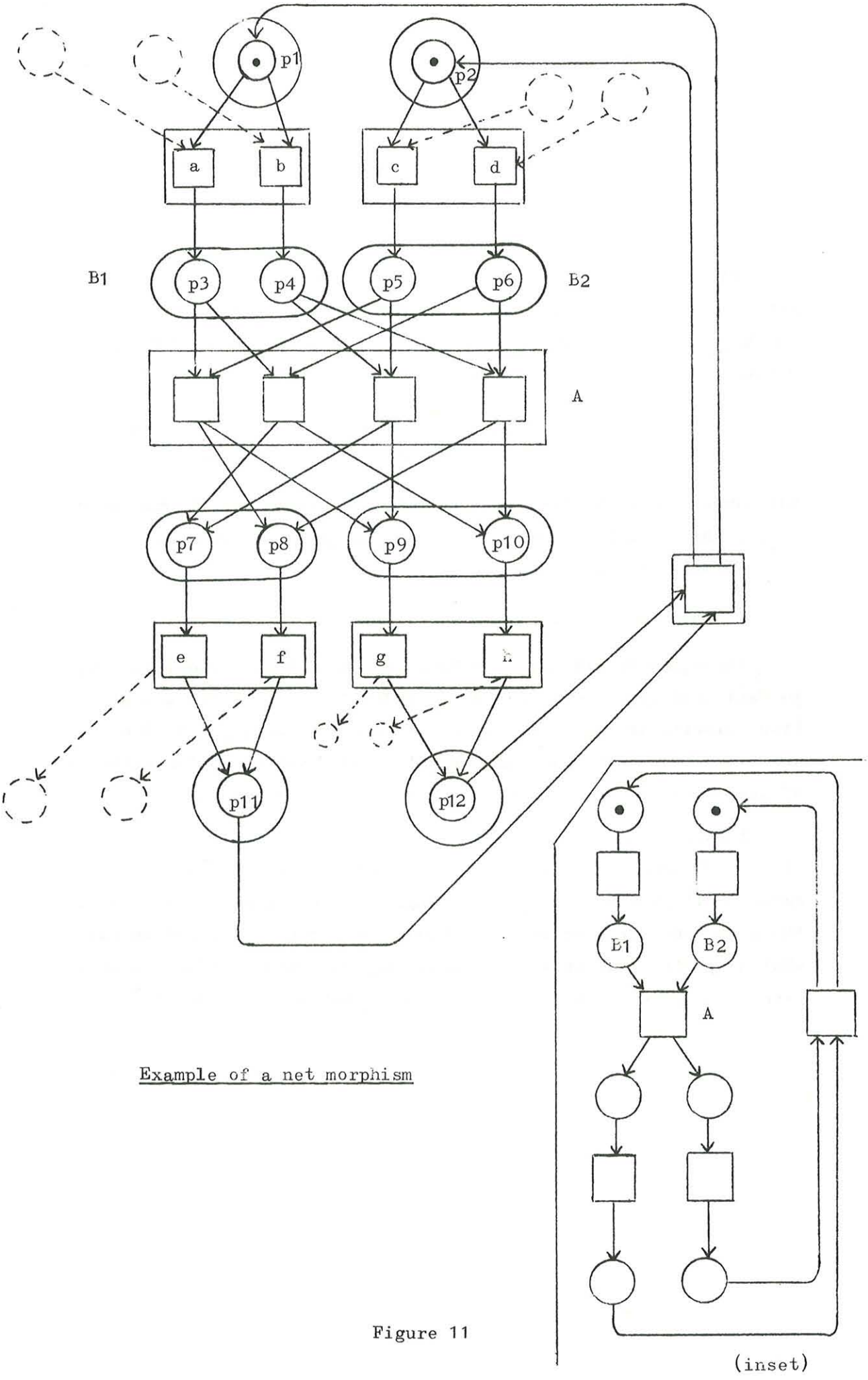
$$(1) \quad \Delta s = C.\Delta t \quad \text{with} \quad s = (\text{tokens}(p_1), \ldots, \text{tokens}(p_n))$$
$$t = (\text{occ}(t_1), \ldots, \text{occ}(t_m))$$

the number of tokens having to be bounded between 0 and some number $s_{i,max}$ which could be considered as the capacity of the i-th place, s_i. Written in vector form:-

$$(2) \quad 0 \leq s \leq s_{max}$$

The equality (1) may seem familiar from physics and there is, in fact, a definite connection except that here C is not a scalar (the velocity of light) but a matrix whose elements are 0, 1 or -1, and s and t are not real numbers but integer vectors. The entirety of the algebraic description is contained in (1) and (2).

I would now like to go back to an example I gave earlier (Figure 4) when I was explaining the nature of the 'gameboard'. We cannot tell just by looking at it what sort of thing can happen in the game, what meaning it might have, to what use it might be put, what properties, algebraic or otherwise, the game on this net will have if we give it the initial marking shown here (Figure 11).

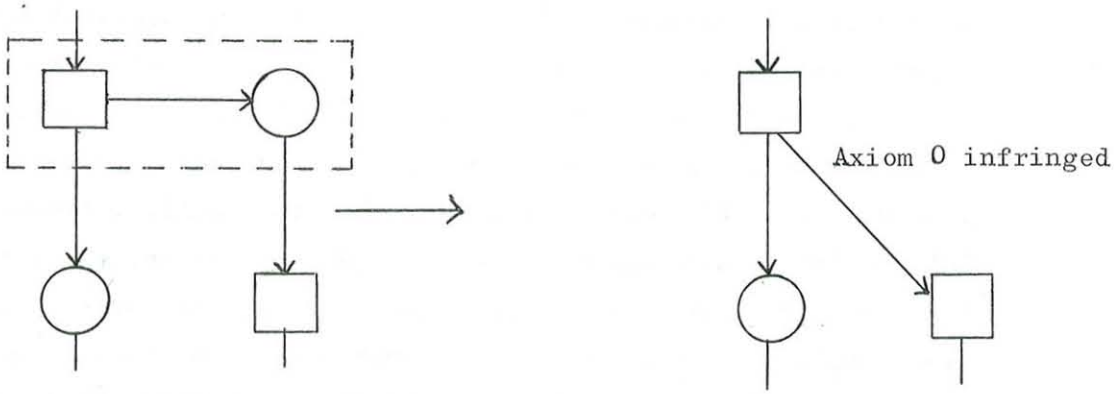


Example of a net morphism

Figure 11

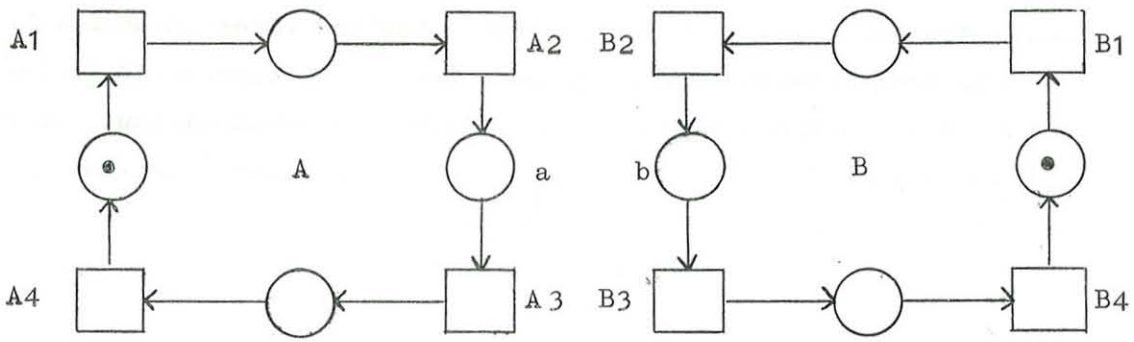
Now we can map this net onto the smaller net shown (Figure 11 - inset); the mapping might be indicated by drawing boxes or circular figures round net elements which are mapped onto the same element of the second net. This net is much simpler than the original; its conditions have only one input and output each. The two tokens can just wander around it like pointers of a clock and these two clocks or cycles are synchronised in such a way that no deadlock can take place. Now, if we go back from the second to the first net we can attach a meaning to box A (Figure 11) namely that it can be regarded as a switching element which computes the logical 'exclusive-or' of the two inputs B1 and B2. Resolution of the conflict between players 'a' and 'b' results in there being a token on either p3 or p4; This token distribution may be regarded as a bit; there is another bit on (p5, p6) and these two are combined by 'A' in such a way that (p7, p8) carries the exclusive-or of the input bits, while (p9, p10) reproduces the input bit on (p5, p6). This latter fact means that the construction has the special property that input can be recomputed from output. (If p9 is marked then p5 must have been marked before the exeuction of 'A'). Thus no information is gained or lost, merely recombined. The input and output of information to the system takes place only at the time of resolution of the conflicts associated with p1 and p2. The environment of the system would have to have additional inputs to a,b,c and d (Figure 11, dotted lines); the marking of these input places would show the bits input to the system from the outside. Similarly, there are resolutions of <u>reverse</u> conflict at e, f and g, h and in order that the information which tells us that a token on p11 had come from p7 as opposed to p8 (say) is not 'lost to the universe', the system's environment would have to have outputs from e and f (Figure 11, dotted lines) so that the bit pattern on p7, p8 can be reproduced.

We have given in Figure 11 an example of a net morphism. In general a morphism must not disrupt the net and must preserve direction. In other words if we draw a box round a number of net elements, in order that the net be not disrupted only transition type elements in the box may be connected to the outside of it - we cannot have something like this:-



A corresponding rule applies when we collect net elements in dotted circles (that is, map them onto a state element). These rules ensure that the image of a net under a net morphism is, mathematically speaking, a net.

I would now like to introduce an application problem. Suppose we have two systems A and B (Figure 12). Let us consider two simple tasks of system coordination: firstly to synchronise the two systems; and secondly to effect the mutual exclusion of the conditions 'a' and 'b'.

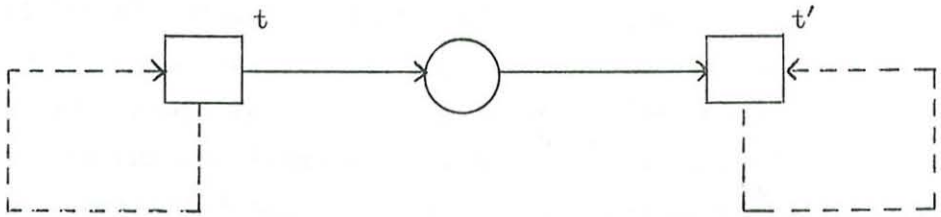


Task 1: Synchronise A and B

Task 2: Effect mutual exclusion of a and b

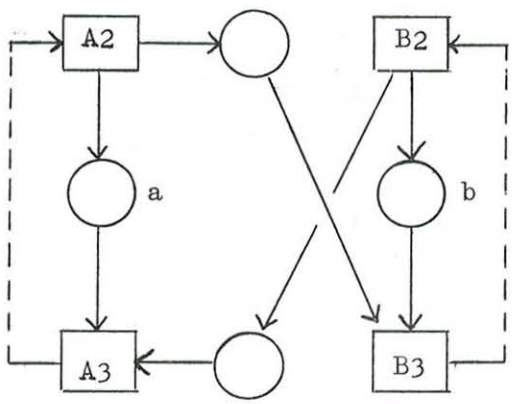
Figure 12

There are two parts to each task; the first is that of description and the second is that of solution. Thus, in the first task we must specify exactly what we mean by 'synchronise A and B'. This description might be something like this: select from each system a transition element and imagine a place connecting these two, thus:-

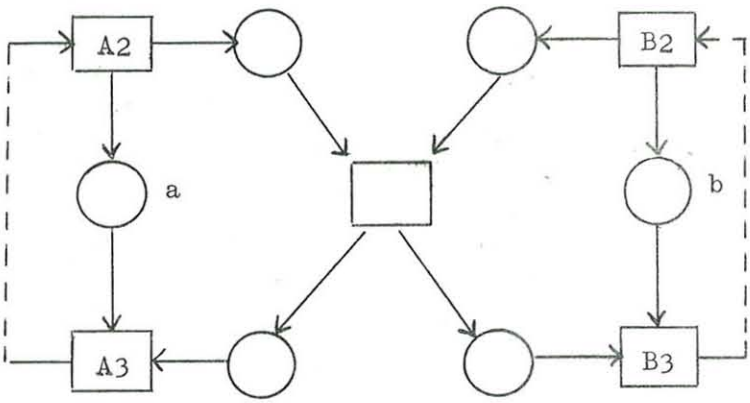


By synchronisation, I mean that this imaginary place would require only a finite capacity (s, say): t may fire at most s times more than t', for otherwise more than s tokens would accumulate on the imaginary place, infringing the capacity restriction. The smallest natural number s will be called the synchronic distance between t and t' - I shall talk more about this later.

That, then is a description of the first task, what about the solution? One way of achieving synchronisation is by attaching to the systems two state elements of unlimited capacity (Figure 13a);



a) Solution of task 1 (unsafe)



b) Solution of task 1 (safe)

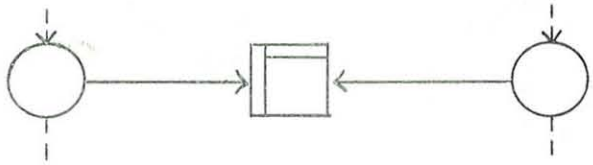
Figure 13

from the point of view of basic hardware considerations, this solution is considered unproblematic and is, in fact, used. This construction has the property that it is not safe, however. If the capacity bounds of these state elements are set to equal one, by defining them to be conditions, for example, then there is the possibility of a contact situation developing - by firing A1 and A2, B1 and B2 to put tokens on p1 and p2 and then firing A3, A4 and A1, after which A2, with tokens on p1 and p3, will be in a contact situation. Now, in a system with the possibility of contact we must take a close look at the technical implementation, and if in this implementation it is not possible to distinguish between one pulse and two pulses coming at the same time, the effects of which accumulate on the new places, then this lack of safeness leads to a system deadlock.

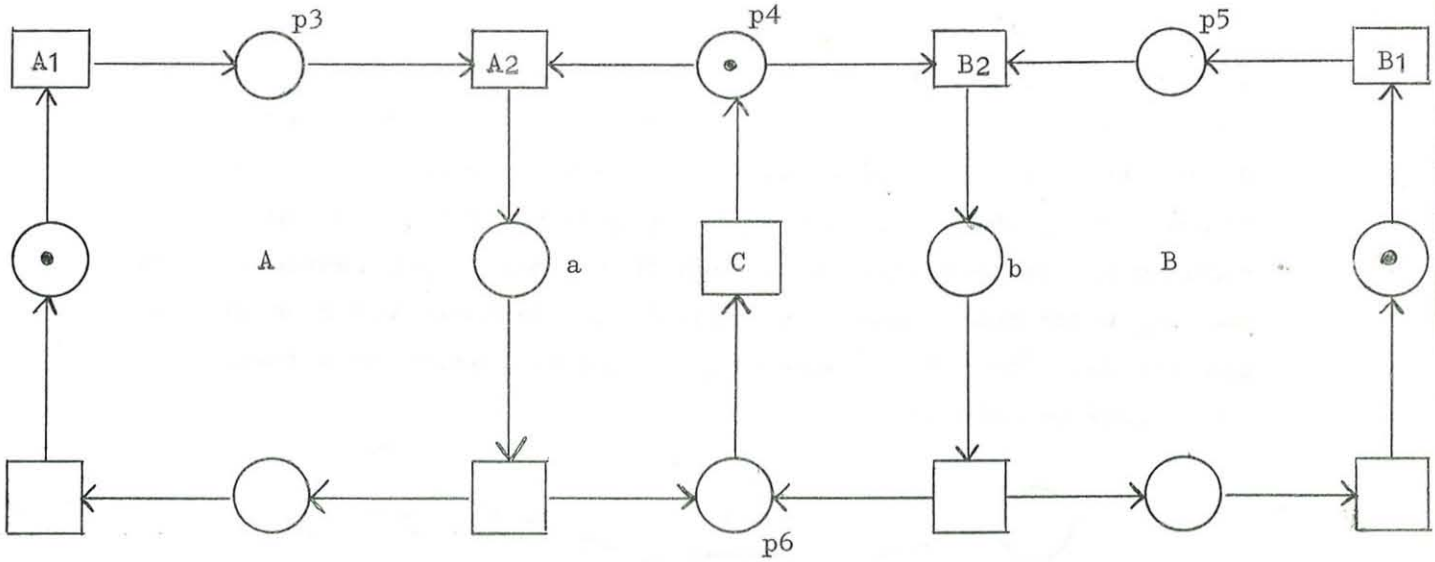
This solution, then, is unsafe and may be used only if the technical limitations for the duration of these processes and holdings of states are known.

There is a safe solution to this task: (Figure 13b) I won't go into details about the firing behaviour of this system except to point out that the synchronic distance between A2 and B2 is two: they may fire concurrently.

We now proceed to the second of the two tasks, that of effecting mutual exclusion of the states 'a' and 'b', and as in the first case we begin with a description of the task; 'a' and 'b' are in mutual exclusion if an imaginary transition connecting them, thus



is always dead (can never have concession).



Solution of task 2 (with confusion)

Figure 14

A solution to this problem is shown in Figure 14; there is the introduction of an additional piece of net with a conflict at p4. This is the way the problem is usually solved. Unfortunately this system has the possibility of confusion; after A1 and B1 have fired, for instance, places p3, p4 and p5 will all be marked giving a case of symmetrical confusion (cf. Figure 9). Thus something more needs to be said about the system. We have a solution, but a solution in which the boundary between the system and its environment has been ineptly drawn and in which it is not possible to trace information flow; if the net of this solution is regarded as a system then information flow through the system is not objective. But it is clear what goes on in it. If p3, p4 and p5 are all marked then it would depend on the environment (on chance, as it is often rashly formulated) whether A2 or B2 is executed first; they cannot occur concurrently by the extensionality axiom. After the critical section has been passed, a token will appear on p6 and then, via a firing of C, reappear on p4 allowing the composite system to continue operating indefinitely, possibly without any occurrence of B2, or of A2.

In order to talk more generally about the techniques we used in the description of the two tasks I have just been talking about, we must look at the main classes of T-elements of level-two nets. There are three types of transitional forms on this level: transitional forms which are such that a player can reside in them and make moves, transitional forms such that a player may reside in them but may never make a move and transitional forms in which no player may reside. That these three classes are exhaustive is a theorem in net theory on this level.

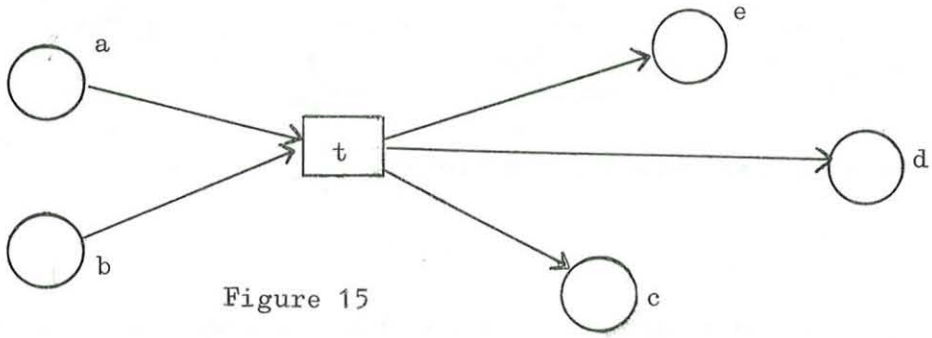


Figure 15

Consider the transition T of Figure 15. If this is a transitional form of the first kind (a process), then at some time 'a' and 'b' may hold and an occurrence of t causes the holding of 'a' and 'b' to be replaced by the holding of 'c', 'd' and 'e':-

Process: $a \wedge b$ is replaced by $c \wedge d \wedge e$

If t is a transitional form of the second type, then in all cases either t does not have all its inputs marked or is in contact. We could refer to t in this situation as a dead transition, but I would rather speak of it as a fact; this transitional form represents precisely a logical fact about the marking class of the whole net; that is, about everything which can be the case in the net. It represents the following proposition which holds in all cases:-

Fact: $a \wedge b$ implies $c \vee d \vee e$

This proposition says precisely that t never has concession. For t to have concession, we must have the following

$a \wedge b \wedge \neg c \wedge \neg d \wedge \neg e$

Thus, that t has never concession is equivalent to

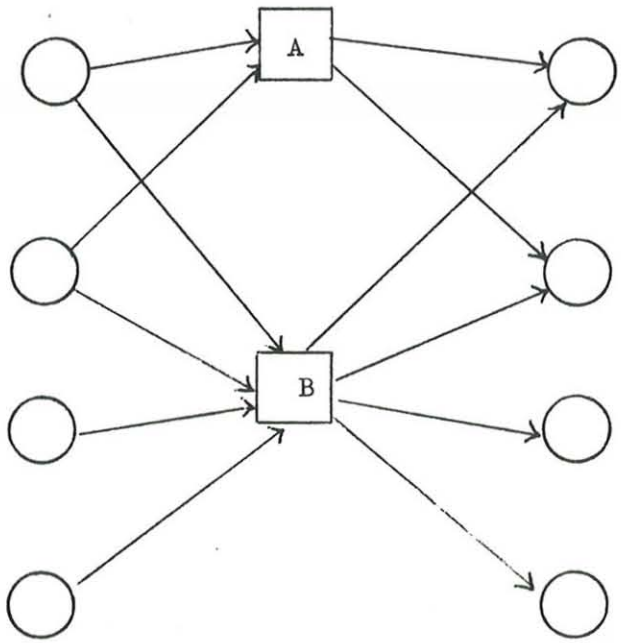
$\neg(a \wedge b \wedge \neg c \wedge \neg d \wedge \neg e)$

which is, of course, logically equivalent to the above implication.

In our description of the mutual exclusion problem we defined 'a' and 'b' to be mutually exclusive if a transition introduced into the net having both 'a' and 'b' as inputs were a fact (namely, that in all cases, $\neg a \vee \neg b$ holds).

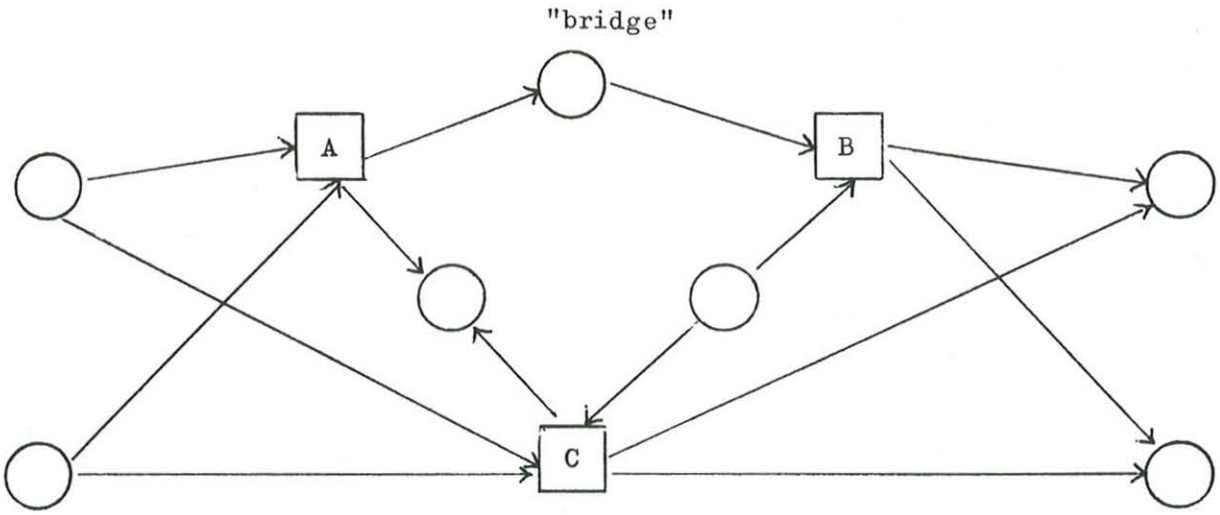
A violation is a transitional form in which no player should be allowed to reside. Unlike a fact, a violation would sometimes have concession, but it should not be allowed to fire as this would lead to a non-case; an example of a non-case would be a single object being in two states at the same time.

As I have said, there is a theorem about transitional forms on this level which says that the class of all T-elements consists of processes, facts and violations. How is this shown? Imagine we have the full class of all possible markings of some initially marked net (I shall not say how this class is constructed; we only need the concept here). A case is a maximal set of concurrently holding conditions and nothing else. Now, if we have such a marking class C we can tell for every possible transitional form which we can attach to the conditions shown in the net (in every possible way) whether this transitional form may carry cases to cases by substitution - call the class of such transitions CC - or whether it may carry cases to non-cases - these transitions constitute the class CN. We can define processes as those transitional forms belonging to CC and violations as those belonging to CN - CC. The theorem states that all other transitional forms are facts: facts = T - (CN ∪ CC). Well, this complete set of transitional forms which can be attached yields a certain super-net of the given net; and the super-net, together with the natural classification of its T-elements, is called the enlogic structure of the original net. We call it this because



Expansion Rule:

If precond(A) $\subset$ precond(B) and postcond(A) $\subset$ postcond(B) and A is a fact, then B is a fact.



Resolution Rule: A and B are facts → C is a fact (see text)

Figure 16

we can use it to deduce from the net axioms alone, without using theorems from logic, theorems such as the following: any transitional form whose set of preconditions (respectively postconditions) is contained within the set of preconditions (respectively postconditions) of some given fact will also be a fact. This is called the expansion rule. Another such theorem says that if we have a 'bridge' between two facts then we may deduce that the transitional form having the same inputs and outputs as the two facts taken together, except for the state element constituting the 'bridge', will also be a fact. We call this the resolution rule. It is known that the expansion and resolution rules are deductively complete. Thus we have here a form of propositional logic, in the guise of operations on nets, but which is applicable not only to the idealised product of static logical structure and time which I mentioned earlier as the supposed basis of algorithms but also to situations in which conditions are subject to concurrent, not necessarily sequentially controlled change. Adequate logical characterisation of such situations requires a logic of change which countenances a non-transitive (non-idealised) concurrency. We could use this simple calculus of facts to teach logic - and in fact this has been done. (Note that this understanding of logic is based upon the understanding of the "transition" rule of the token game alone).

To return to the use of level-two elements for specification: transitional forms classify into processes, facts and violations, (with many important subclasses not described here) and the entirety of state element forms are places with capacities of various sizes - from zero to infinity. Now for specification purposes we can see that places with capacity equal to zero denote coincidence between events, that those of capacity one denote conditions and that those of capacity n denote the synchronic distance n. For example, the input and output events of a buffer of capacity n have synchronic distance n. We can use these concepts - as before when describing tasks of synchronising two example systems, and of effecting mutual exclusion between states in them - to precisely define properties of the system by attaching

Basic uses of Level 2 Elements for Specification

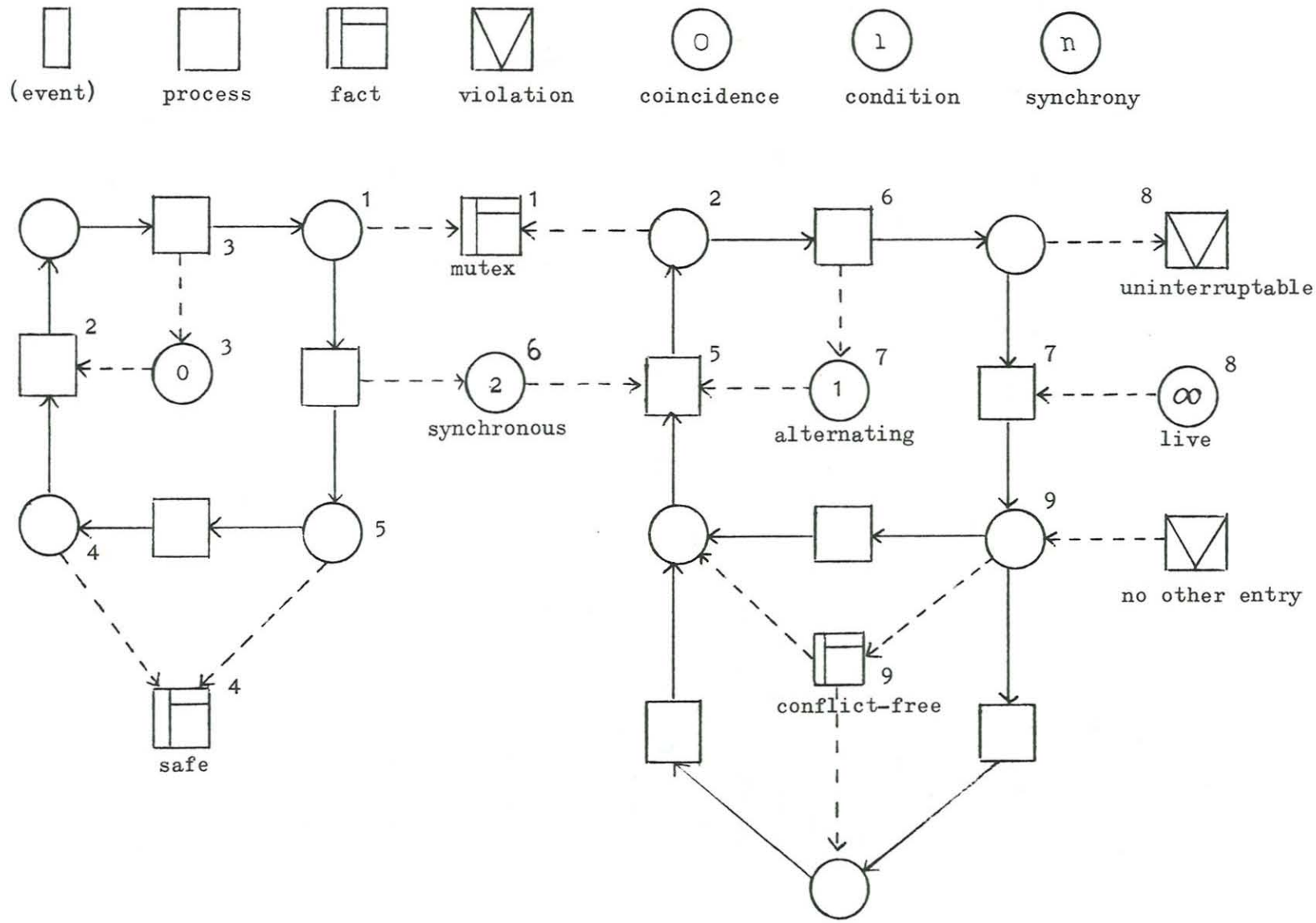


Figure 17

elements with specific properties to the system. In Figure 17, for example, we can specify the mutual exclusion of states 1 and 2 by appending fact 1; coincidence between transitions 2 and 3 using state 3; we can specify the safety of the net, the absence of a contact situation, by specifying, using a fact, that certain states can never be marked simultaneously (as, for example, states four and five and transition 4); we can specify concurrency or synchronicity in the usual sense by defining the capacity of an adjoined state element (as in the case of state 6) to be two; we can specify that processes five and six are alternating by defining the capacity of state seven to be one; we can specify that the process consisting of the two partial processes six and seven is uninterruptable by appending violation symbol eight; we can specify liveness in that area of the net, and likewise in all cycles, by appending state element eight with infinite capacity; we can specify absence of possible conflict by adjoining fact nine. Finally we can specify that this system, if regarded as a flow diagram for example, has no other entry except the entry shown at state element 9, by adjoining a violation symbol here, pointing to state element 9.