ISTITUTO DI ELABORAZIONE DELLA INFORMAZIONE DEL CNR

INTERPROCESS COMMUNICATIONS: DEADLOCK CONDITIONS

P. Ancilotti, M. Fusani, N. Lijtmaer

NOTA INTERNA B75-11

AGOSTO 1975

STAMPATA IN PROPRIO

1. INTRODUCTION

This work is concerned with the behaviour of systems composed by several asynchronous modules. In particular, systems whose modules communicate by means of *send* and *receive* primitives are considered and deadlock conditions, that may arise by using these message passing primitives, are analyzed. For this purpose we introduce a formal model based on computation schemata [1].

First of all we introduce sequential schemata and then we define cyclic sequential schemata in order to model cyclic sequential processes. Systems composed by a set of concurrent processes are modelled by means of parallel combination of cyclic sequential schemata.

On the ground of both processes structure and connections structure, systems are classified in three types: 1) time and data independent systems; 2) data dependent and time independent systems; 3) time and data dependent systems.

For the first two types of systems a set of properties are proved. In particular, for the first type of systems we prove that: If a deadlock condition arises in a particular computation, then the same deadlock condition will arise in every computation. For the second type of systems a similar argument can be proved, that is: If, given a set of input data, a deadlock condition arises in a particular computation, then, with the same input data, the same deadlock condition will arise in every computation.

2. COMPUTATION SCHEMATA: SOME DEFINITIONS

"A *computation schema*, or *schema*, represents the manner in which functional elements and decision elements are interconnected, and their actions sequenced, to define an algorithm" [1]. More precisely:

Definition 2.1: A *schema* is a triple S=(A,V,C), where:
A is a set of *actors*, V is a set of *variables*, and C is a set of *control sequences* (*control set* [2]).

The functional elements of a schema are called *operators* and the decision elements are called *deciders*. The set of operators and the set of deciders are denoted by O and D, respectively. These sets are disjoint. Both, operators and deciders, are called *actors*. Then, actors are agents capable of either transforming or testing values.

Each operator o evaluates some unspecified function f_o of m input variables and assigns values to n output variables:

$$f_o : \{\text{m-tuples}\} \rightarrow \{\text{n-tuples}\}$$

Operators having no input variables are allowed and are used to model constants.

A decider d tests some unspecified predicate p_d on m input variables

$$p_d : \{\text{m-tuples}\} \rightarrow \{true, false\}$$

Deciders must at least have one input value.

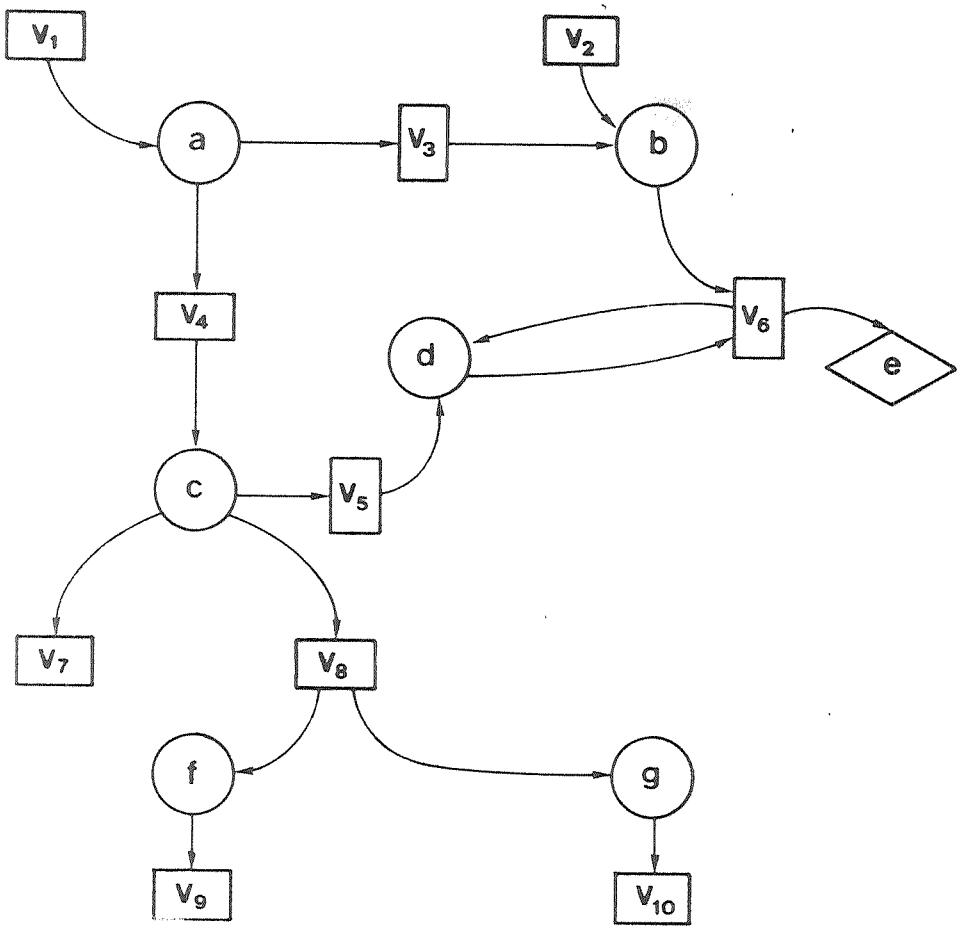
A *domain* X_a, which is a finite subset of V, is associated to each actor a. Similarly, a *range* Y_o, which is also a finite subset of V, is associated to each operator o.

A set I of variables ($I \subseteq V$) is called *schema input* if its intersection with the range of any operator is empty. Similarly, a set U of variables ($U \subseteq V$) is called *schema output* if its intersection with the domain of any actor is empty.

While variables of schemata may be denoted, generally, by single memory cells, each input or output variable of a schema may be denoted by a queue of memory cells. Each concecutive read operation uses up a single cell of the input queue. Similarly, for each writing a new cell is added to the output queue.

The interconnections between actors and variables may be represented by a

data flow graph . Figure 2.1 shows a data flow graph of a schema, where circles represent actors and boxes represent variables [1].



$O = \{a,b,c,d,e,f,g\}$

$I = \{v_1,v_2\}$

$X_a = \{v_1\}$, $Y_a = \{v_3,v_4\}$;

$X_c = \{v_4\}$, $Y_c = \{v_5,v_7,v_8\}$;

$X_e = \{v_6\}$;

$X_g = \{v_8\}$, $Y_g = \{v_{10}\}$

$D = \{e\}$

$V = \{v_7,v_9,v_{10}\}$

$X_b = \{v_2,v_3\}$, $Y_b = \{v_6\}$;

$X_d = \{v_5,v_6\}$, $Y_d = \{v_6\}$;

$X_f = \{v_8\}$, $Y_f = \{v_9\}$;

Figure 2.1

Actors, operators and deciders, will be considered the units of computational activity, as characterized by their external behaviour. Associated with an operator o there are an *initiation event* $\bar{o}$ and a *termination event* $\underline{o}$. Associated with each decider d, there are an initiation event $\bar{d}$ and either the true or the false termination event, denoted by d_T and d_F respectively.

A *control sequence* of the schema is a string $\sigma=abc...q...$ of actor initiation and termination events. The *control set* C represents all the allowed sequences of events.

Note that if $q=\underline{a}$ belongs to a control sequence σ, then an occurrence of $\bar{a}$ must be present in the prefix of σ that preceeds q.

The sequences in which operators and deciders are permitted to act, may be specified by a *precedence graph* [1]. Figure 2.2 shows a possible precedence graph of the schema whose data flow graph is represented in Figure 2.1.

Let us introduce some definitions needed to develop our model.

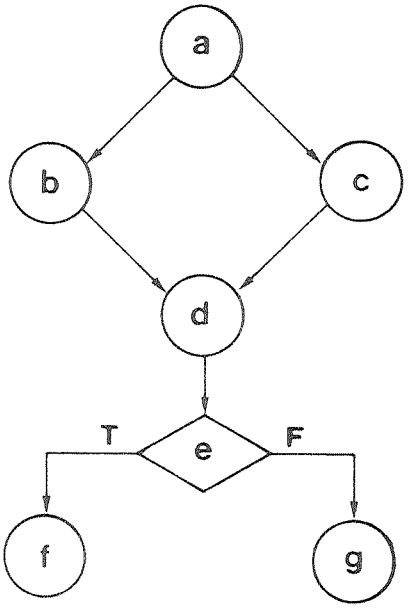
Definition 2.2: Given a control sequence $\sigma=ab...pq...$, if $\pi=ab...p$ is a prefix of σ, after the occurrences of the events in π, the only *feasible event* in σ is q.

If t(x) denotes the time of the occurrence of an event x, then the event q is feasible in the time interval from t(p) to t(q).

Definition 2.3: A schema S is *data independent* if the set of deciders is empty, otherwise S is *data dependent*.

A more detailed treatment about schemata is contained in [1,2] . In this paper we will be concerned with the control aspects of computation schemata. In particular, we want to emphasize that our results are concerned with *control* and *sequencing* of a set of cooperating processes, and not with their specific functions. For this purpose we refer to schemata as models of uninterpreted program modules. In this context any control sequence of a schema represents a particular process the program module may give rise to during execution while input/output.variables of the schema represent the external world with which the process exchanges data.

To convert a schema into a specification of a particular program module it is necessary to specify both the domains of the schema variables and a function or predicate for each operator or decider.



$$C \equiv \begin{cases} \sigma_1 = \bar{a}\ \underline{a}\ \bar{b}\ \underline{b}\ \bar{c}\ \underline{c}\ \bar{d}\ \underline{d}\ \bar{e}\ \underline{e_T}\ \bar{f}\ \underline{f} \\ \sigma_2 = \bar{a}\ \underline{a}\ \bar{b}\ \bar{c}\ \underline{b}\ \underline{c}\ \bar{d}\ \underline{d}\ \bar{e}\ \underline{e_T}\ \bar{f}\ \underline{f} \\ \sigma_3 = \bar{a}\ \underline{a}\ \bar{b}\ \bar{c}\ \underline{c}\ \underline{b}\ \bar{d}\ \underline{d}\ \bar{e}\ \underline{e_T}\ \bar{f}\ \underline{f} \\ \sigma_4 = \bar{a}\ \underline{a}\ \bar{c}\ \underline{c}\ \bar{b}\ \underline{b}\ \bar{d}\ \underline{d}\ \bar{e}\ \underline{e_T}\ \bar{f}\ \underline{f} \\ \sigma_5 = \bar{a}\ \underline{a}\ \bar{c}\ \bar{b}\ \underline{c}\ \underline{b}\ \bar{d}\ \underline{d}\ \bar{e}\ \underline{e_T}\ \bar{f}\ \underline{f} \\ \sigma_6 = \bar{a}\ \underline{a}\ \bar{c}\ \bar{b}\ \underline{b}\ \underline{c}\ \bar{d}\ \underline{d}\ \bar{e}\ \underline{e_T}\ \bar{f}\ \underline{f} \\ \sigma_7 = \bar{a}\ \underline{a}\ \bar{b}\ \underline{b}\ \bar{c}\ \underline{c}\ \bar{d}\ \underline{d}\ \bar{e}\ \underline{e_F}\ \bar{g}\ \underline{g} \\ \sigma_8 = \bar{a}\ \underline{a}\ \bar{b}\ \bar{c}\ \underline{b}\ \underline{c}\ \bar{d}\ \underline{d}\ \bar{e}\ \underline{e_F}\ \bar{g}\ \underline{g} \\ \sigma_9 = \bar{a}\ \underline{a}\ \bar{b}\ \bar{c}\ \underline{c}\ \underline{b}\ \bar{d}\ \underline{d}\ \bar{e}\ \underline{e_F}\ \bar{g}\ \underline{g} \\ \sigma_{10} = \bar{a}\ \underline{a}\ \bar{c}\ \underline{c}\ \bar{b}\ \underline{b}\ \bar{d}\ \underline{d}\ \bar{e}\ \underline{e_F}\ \bar{g}\ \underline{g} \\ \sigma_{11} = \bar{a}\ \underline{a}\ \bar{c}\ \bar{b}\ \underline{c}\ \underline{b}\ \bar{d}\ \underline{d}\ \bar{e}\ \underline{e_F}\ \bar{g}\ \underline{g} \\ \sigma_{12} = \bar{a}\ \underline{a}\ \bar{c}\ \bar{b}\ \underline{b}\ \underline{c}\ \bar{d}\ \underline{d}\ \bar{e}\ \underline{e_F}\ \bar{g}\ \underline{g} \end{cases}$$

Figure 2.2

Definition 2.4: An *interpretation* of a schema S is

i) for each variable $v \in V$, a value set $F(V)$;

ii) for each operator $o \in O$, a function $f_o : F(V_{x1}) \times F(V_{x2}) \times \ldots \times F(V_{xm}) \rightarrow F(V_{y1}) \times F(V_{y2}) \times \ldots \times F(V_{yn})$, where $X_o = \{V_{x1}, \ldots, V_{xm}\}$ and $Y_o = \{V_{y1}, \ldots, V_{yn}\}$;

iii) for each decider $d \in D$ e predicate $P_d : F(V_{x1}) \times F(V_{x2}) \times \ldots \times F(V_{xm}) \rightarrow \{true, false\}$, where $X_d = \{V_{x1}, \ldots, V_{xm}\}$.

Definition 2.5: Given an interpretation of a schema S, an *initial assignment* for S is a function z defined on V, such that $z(v) \in F(v)$ for each $v \in V$. To each v the value z(v) is assigned before the computation begins.

Our attention will be focused now on sequential processes. Parallel processes will be analized later as a combination of several processes which are themselves strictly sequential.

3. SEQUENTIAL SCHEMATA

A sequential schema is a schema characterized by an unique locus of control or, more precisely:

Definition 3.1: A *sequential schema* is a schema S=(A,V,C) where control sequences of C satisfy the following properties:

i) for each $\sigma \in C$, after the occurrence of $\bar{a}$, the feasible event is $\underline{a}^{(\circ)}$ and viceversa an event $\underline{a}$ is feasible only after the occurrence of $\bar{a}$;

ii) the feasible event after an occurrence of an event $\underline{a}$ is the same for all occurrences of $\underline{a}$ in all control sequences;

iii) all the control sequences of C begin with the same event.

The first part of this definition states that concurrent executions of two or more actors are not allowed; while the second one and the third one exclude "fork" and "join" actors for non trivial schemata.

Note that if a sequential schema is data independent then there is only one possible control sequence σ in C.

While Figure 3.1(a) shows the control graph of a data dependent sequential schema, Figure 3.1(b) shows the control graph of a data independent sequential schema.

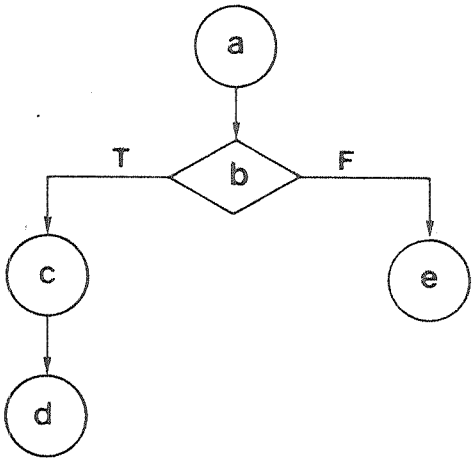
Definition 3.2: A *finite sequential schema* is a sequential schema in which the set of actors is finite.

While the control sequences of this type of schemata allow to model the behaviour of sequential processes, a new type of schema must be introduced to handle cyclic sequential processes.

Definition 3.3: Given a finite sequential schema S=(A,V,C), let us define a *cyclic sequential schema* $S^C=(A^C,V^C,C^C)$, where $A^C=A$, $V^C=V$ and $\sigma^C \in C$ iff $\sigma^C=\sigma_1\sigma_2 \ldots \sigma_i \ldots$ with $\sigma_i \in C$ for any integer $i>0$. S is called the *generating schema*

$(\circ)$ Note that if the actor is a decider d, then $\underline{a}$ may be d_T in some sequences and d_F in others.

of S^c. When a σ_i is composed by infinite events, then $\sigma^c=\sigma_1 \ldots \sigma_i$, that is, after a certain event, σ^c is composed by events of σ_i.



$\sigma_1 = \bar{a}\ \underline{a}\ \bar{b}\ \underline{b}_T\ \bar{c}\ \underline{c}\ \bar{d}\ \underline{d}$

$\sigma_2 = \bar{a}\ \underline{a}\ \bar{b}\ \underline{b}_F\ \bar{e}\ \underline{e}$

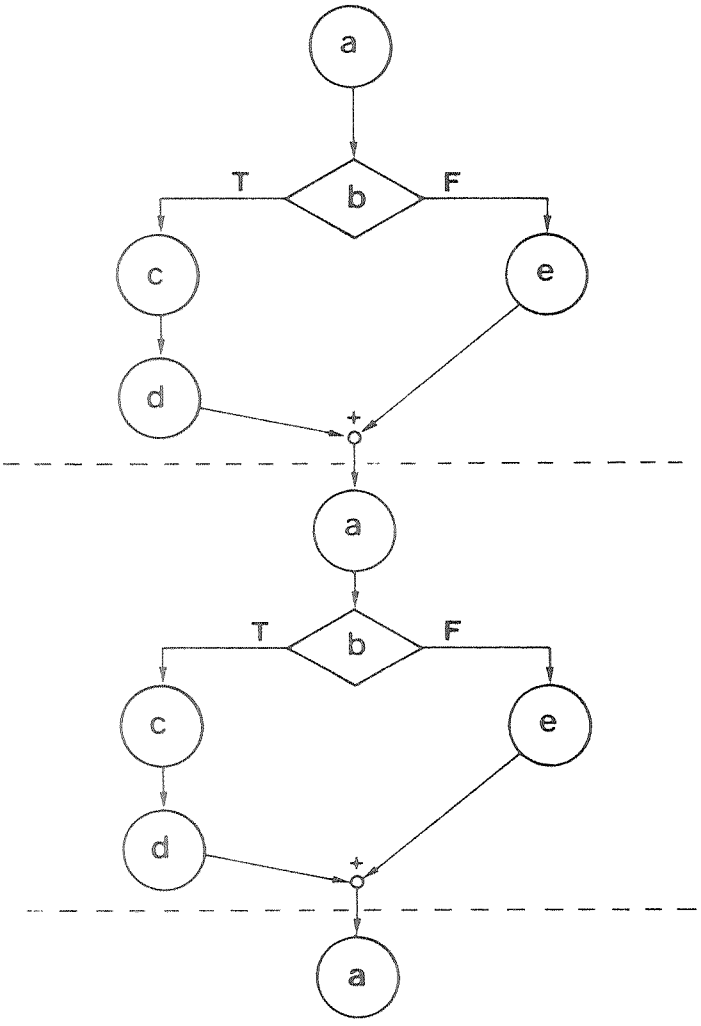
a)

$\sigma = \bar{a}\ \underline{a}\ \bar{b}\ \underline{b}\ \bar{c}\ \underline{c}\ \bar{d}\ \underline{d}$

b)

Figure 3.1

The control graph of a cyclic schema S^c is shown in Figure 3.2. This schema was generated by the schema whose control graph is represented in Figure 3.1(a).



$\sigma_1^c = \sigma_1\ \sigma_2\ \sigma_1\ \sigma_2\ \cdots$

$\sigma_2^c = \sigma_1\ \sigma_1\ \sigma_1\ \sigma_2\ \sigma_1\ \sigma_2\ \cdots$

.
.
.
.
.

Figure 3.2

If a cyclic sequential schema S^C is data independent then there is only one possible control sequence σ^C in C^C: $\sigma^C = \sigma\sigma\sigma\ldots\sigma\ldots$ where σ is the unique control sequence of S.

4. PARALLEL COMPOSITION OF CYCLIC SEQUENTIAL SCHEMATA

In a parallel schema a mechanism capable of describing concurrent, asynchronous activity is needed: This mechanism consists of allowing several initiations before a termination occurs, and of keeping track of such initiations.

Our attention will be focused now on schemata whose control sequences model sets of concurrent processes. They can be conceived as a combination of several processes which are themselves strictly sequential.

In this section we introduce the *directed parallel composition of cyclic sequential schemata* to model the behaviour, and to point out special properties, of systems composed of several independent modules. Modules are connected by a message buffer mechanism. A single module becomes, during execution, a cyclic sequential process.

In order to point out some properties related to concurrency, let us introduce:

i) A special type of variable, called *mailbox;*

ii) Two new types of actors, *send* and *receive*.

Definition 4.1: A *mailbox* is a pair (C,N) where C, called *counter*, is a memory cell that can assume only integer value between zero and n, and where N is a queue of n memory cells.

The integer n represents the *capacity* of the mailbox. Note that this capacity may be infinite. The integer w, the actual value of the counter, is referred to as the *state* of the mailbox.

Definition 4.2: *Send* and *receive* are actors such that:

i) the domain of any actors *send* (*receive*) is a set $X_s=\{x,m\}$ ($X_r=\{m\}$) where the variables x and m are a memory cell and a mailbox respectively. The range of any actor *send* (*receive*) is a set $Y_s=\{m\}$ ($Y_r=\{x,m\}$) where the mailbox m is the same in both the domain and the range;

ii) associated with each actor *send* (*receive*) there are an initiation event $\bar{s}$ ($\bar{r}$) and a termination event $\underline{s}$ ($\underline{r}$);

iii) given a schema S=(A,V,C), for each control sequence $\sigma=ab...pq... \in C$ where $q=\underline{s}$ ($q=\underline{r}$) is the termination event of an actor *send* (*receive*), q is *feasible* in σ only after the occurrence of all the events $ab...p$ and if the state of the mailbox m is such that $w<n$ ($w>o$). When $\underline{s}$ ($\underline{r}$) occurs, the state of m is

modified as follows:

w:=w+1 (w:=w-1).

The behaviour of an actor *send* is shown in the block diagram of Figure 4.1.

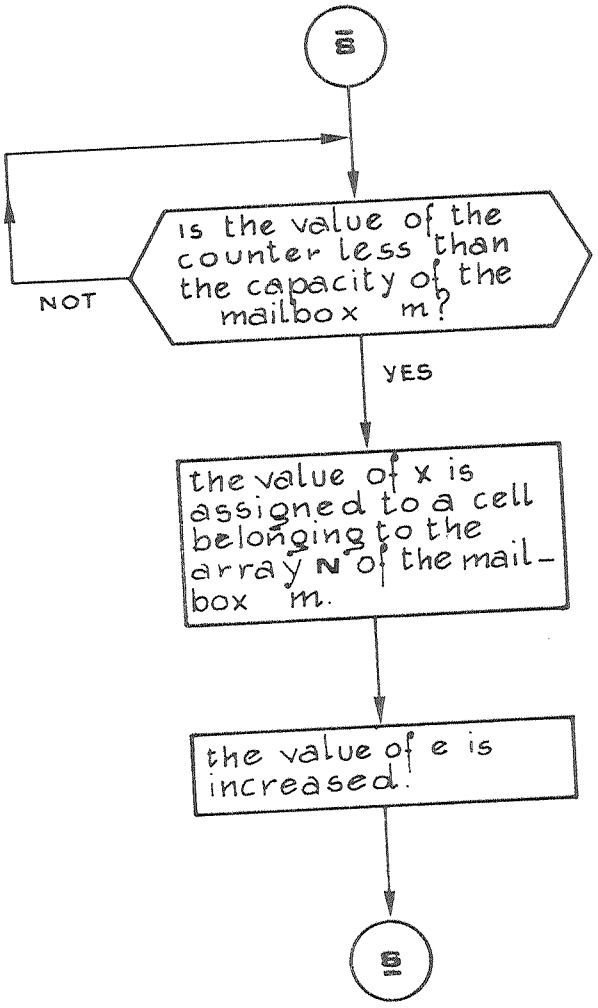


Figure 4.1

A similar diagram can be made for the actor *receive*. Note that *send* and *receive* are the only actors that are allowed to modify the state of a mailbox.

Definition 4.3: The *state* of a schema is the set of states of all the mailboxes of the schema.

Given an initial state and a control sequence we obtain a sequence of states of the schema. From now on the initial state is assumed to be such that for each mailbox w=0.

To define the parallel composition of n schemata $S_1, S_2, \ldots, S_n$ we introduce sets of mailboxes $M_{i,j}$ belonging to the range of actors *send* of the schema S_i and to the domain of actors *receive* of S_j. Mailbox $m_{i,j} \in M_{j,i}$ is at the same time an *output mailbox* of S_i and an *input mailbox* of S_j; $m_{i,j}$ is said to link directly S_i to S_j.

$M_{i,j}$ and $M_{j,i}$ are disjoint sets, possibly empty. If $M_{i,j}=M_{j,i}=\Phi$, then S_i and S_j are not directly connected.

Definition 4.4: A *Directed Parallel Composition* (DPC) of n cyclic sequential schemata $S_1, S_2, \ldots, S_n$ is a schema $S^P=(A^P,V^P,C^P)$ where:

i) $A^P=\bigcup_{i=1}^{n} A_i$ and for any pair (S_i,S_j) of component schemata $A_i \cap A_j=\Phi$;

ii) $V^P=\bigcup_{i=1}^{n} V_i$, and for any pair (S_i,S_j) of component schemata $V_i \cap V_j=M_{i,j} \cup M_{j,i}$;

iii) Any $\sigma^P \in C^P$ can be generated as follows: given an n-tuple $\sigma_1 \ldots \sigma_n$ where $\sigma_i \in C_i$ $(i=1,\ldots,n)$, for each prefix $\pi=abc\ldots p$ of σ^P, $\pi'=\pi q$ is a prefix of σ^P if q is the feasible event of a σ_i after the occurrence of the events of σ_i in π. Moreover, we assume that all the events that become feasible infinitely many times in some σ_i must occur in σ^P. As a final step in the construction of C^P, we erase all the above defined sequences of type $\ldots\bar{a}\ldots\bar{b}\ldots$ where a and b are *send* or *receive* actors operating on the same mailbox, and no occurrence of $\underline{a}$ exists between $\bar{a}$ and $\bar{b}$.

The last constraint is introduced in order to assure the indivisibility of *send* and *receive* [3].

Any control sequence σ^P of S^P can be obtained by "merging" control sequences, one for every component schema. However, it is necessary to observe the constraints among events imposed by point iii) of Definition 4.4.

For example, let S^P be the DPC of the schemata S_1 and S_2. Let $M_{1,2} = \{x\}$ and $M_{2,1} = \{y\}$ be mailboxes that connect S_1 and S_2, such that $n_x = n_y = 1$. Given the control sequences

$$\sigma_1 = \bar{a}\ \underline{a}\ \bar{b}\ \underline{b}\ \bar{r}_x\ \underline{r}_x\ \bar{c}\ \ \underline{c}\ \ \bar{s}_y\ \underline{s}_y \ldots$$

$$\sigma_2 = \bar{d}\ \underline{d}\ \bar{s}_x\ \underline{s}_x\ \bar{f}\ \ \underline{f}\ \bar{r}_y\ \underline{r}_y \ldots$$

where r_x, r_y, s_x, s_y denote the actors *receive* and *send* related to the mailboxes x and y respectively, then

$$\sigma'^P = \bar{a}\ \bar{d}\ \underline{a}\ \underline{d}\ \bar{b}\ \underline{b}\ \bar{s}_x\ \underline{s}_x\ \bar{r}_x\ \underline{r}_x\ \bar{r}_x\ \underline{r}_x \ldots$$

is not a control sequence for S^P since the event $\underline{r}_x$ is not feasible after the occurrence of the events in the prefix $\pi' = \bar{a}\ \bar{d}\ \underline{a}\ \underline{d}\ \bar{b}\ \underline{b}\ \bar{s}_x\ \underline{s}_x\ \bar{r}_x\ \underline{r}_x\ \bar{r}_x$. Similarly,

$$\sigma''^P = \bar{a}\ \bar{d}\ \underline{a}\ \underline{d}\ \bar{b}\ \underline{b}\ \bar{s}_x\ \bar{r}_x\ \underline{r}_x\ \underline{s}_x \ldots$$

is not a control sequence for S^P since the indivisibility constraint of *send* is violated.

From now on, given a σ^P of a DPC S^P, for each component schema S_i, we will denote by σ_i the control sequence of S_i that concurs to generate σ^P.

A DPC S^P can be represented by a *connection graph G*.

Definition 4.5: Given a DPC S^P of n cyclic sequential schemata $S_1\ S_2 \ldots S_n$, the *connection graph* is a bipartite graph whose nodes are component schemata and mailboxes. The arcs of the connection graph represent links among schemata through mailboxes.

Figure 4.2 shows the connection graph of a DPC of the schemata S_1, S_2, S_3, S_4, S_5.

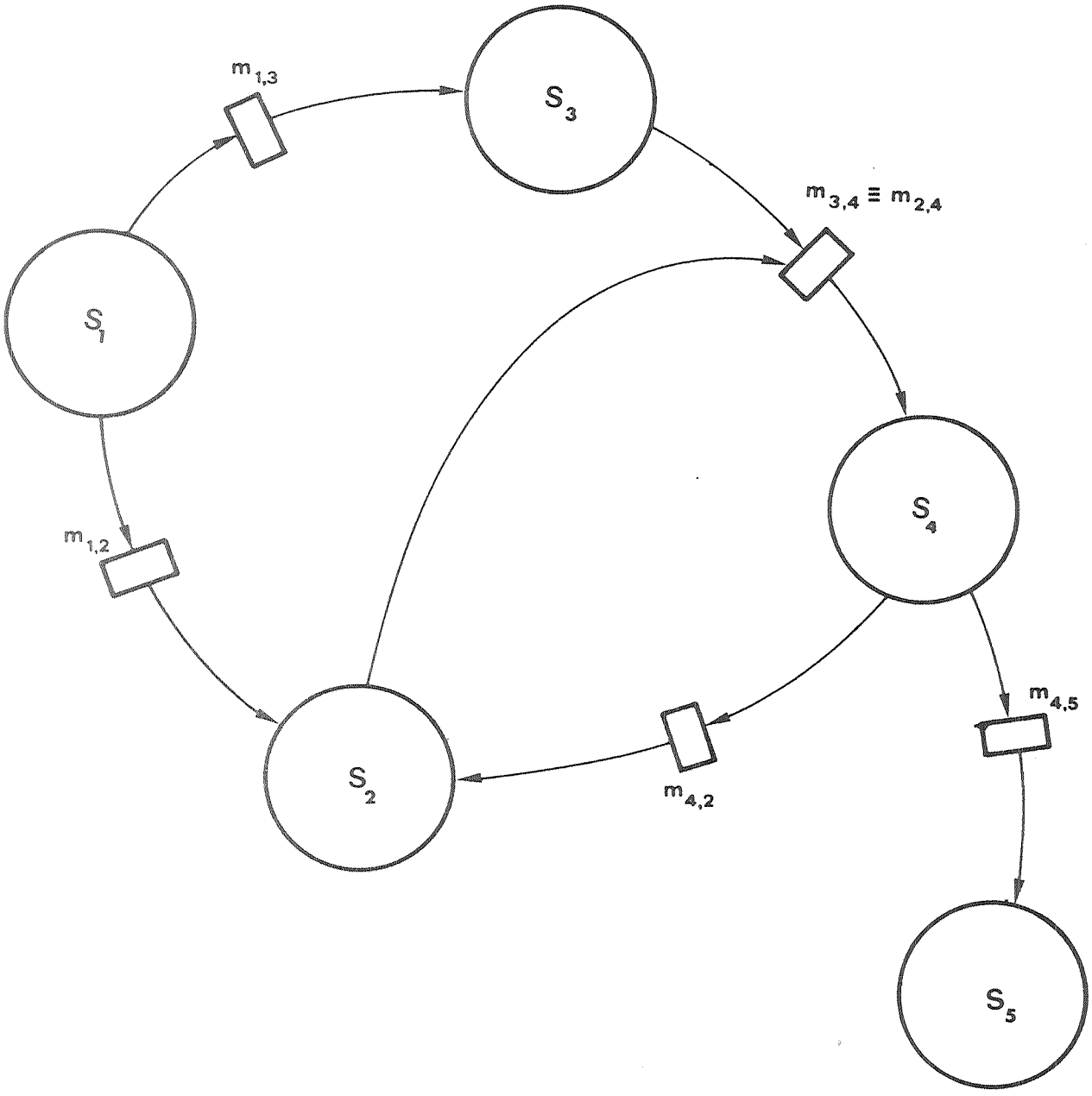


Figure 4.2

We introduce now some definitions on connection graphs.

Definitions 4.6: A *path* in the connection graph G of a DPC S^P is a sequence of schemata $S_1,\ldots,S_k$ linked through k-1 mailboxes:

$$m_{1,2},m_{2,3},\ldots,m_{k-1,k}$$

In Figure 4.2, S_1,S_3,S_4,S_5 is a path.

Definition 4.7: A *semipath* in G is a sequence of schemata $S_1,\ldots,S_k$ linked through k-1 mailboxes:

$$m_{1,2} \text{ or } m_{2,1},m_{2,3} \text{ or } m_{3,2},\ldots,m_{k-1,k} \text{ or } m_{k,k-1}\ .$$

In Figure 4.2, S_2,S_1,S_3 is a semipath.

Definition 4.8: A *semicycle* is a semipath $S_1,\ldots,S_k$, such that S_k coincides with S_1 and the two mailboxes linking any triple of consecutive schemata are different.

While S_2,S_1,S_3,S_4,S_2 is a semicycle, S_1,S_3,S_1 is not a semicycle (Figure 4.2).

Definition 4.9: A mailbox is called *one-to-one* if it links only two schemata.

For example, $m_{1,2},m_{1,3}m_{1,2},m_{4,5}$ are one-to-one mailboxes (Figure 4.2).

Definition 4.10: Given a DPC S^P of n cyclic sequential schemata $S_1,\ldots,S_n$, for each component schema S_i and for each mailbox m belonging to this schema let us denote by $\lambda\ (S_i,m)$ the function that gives *the set* $\Lambda_{i,m}$ *of schemata connected to* S_i *through* m.

If m is one-to-one mailbox, $\Lambda_{i,m}$ is composed by only one schema. For example, in Figure 4.2, $\lambda(S_4,m_{3,4})=\lambda(S_4,m_{2,4})=\{S_2,S_3\},\lambda(S_1,m_{1,3})=S_3$.

Definition 4.11: A DPC S^P is *well formed* if:
i) any mailbox links at least two schemata;
ii)the connection graph G is connected.

The DPC whose connection graph is shown in Figure 4.2 is well formed.

Note that in a well formed DPC S^P, for each component schema S_i the set $\Lambda_{i,m}=\lambda(S_i,m)$ is not empty, where m is a mailbox belonging to S_i.

From now on, any DPC will be tacitly considered well formed.

In order to point out some properties related to the control and sequencing of a set of cooperating processes, let us give the following definitions:

Definition 4.12: Given a control sequence σ^P of a DPC S^P, a *deadlock* occurs in σ^P if at least one component schema S_k and an event p belonging to both σ_k and σ^P exists such that no event that follows p in σ^P belongs to σ_k. Here σ_k is said a *deadlocked control sequence*. The first event q that follows p in σ_k is called the *blocking event* of σ_k related to σ^P.

Note that, for each deadlocked σ_k, the blocking event does not belong to σ^P: It is an unfeasible termination of an actor *send* or *receive* related to a mailbox m^* whose state is $w=n$ or $w=o$ respectively. In such case, we will denote briefly by Λ_k, the set Λ_{k,m^*} of schemata connected to S_k through m^*. This mailbox will be called the *critical mailbox* of S_k.

Definition 4.13: Given a control sequence σ^P of a DPC S^P, for any blocking event q, belonging to a deadlocked control sequence σ_i, if there is an event q' belonging to a control sequence σ_j such that the occurrence of q' in σ^P makes feasible q, then q' is called an *awaking event* of q.

Obviously, if σ_i is deadlocked in σ^P, then also σ_j is deadlocked.

Definition 4.14: A deadlock in a control sequence σ^P of a DPC S^P is called a *partial deadlock* if σ^P is infinite. Otherwise it is called a *total deadlock*.

In the following sections, distinctions among different kinds of DPC's will be pointed out in order to find relevant properties of systems under deadlock conditions.

5. DATA AND TIME INDEPENDENT SYSTEMS

Definition 5.1: A DPC S^P of n cyclic sequential schemata $S_1,\ldots,S_n$ is of *type 1* if all the component schemata are data independent and all the mailboxes are one-to-one.

Since they are data independent, any DPC S^P of type 1 is such that all its component schemata have only one control sequence. Furthermore, we can prove that:

Lemma 5.1: Given a DPC S^P of type 1, if a deadlock occurs in some σ^P, then for any blocking event q, belonging to a deadlocked control sequence σ_i, there exists one component schema $S_j = \Lambda_i$ whose control sequence σ_j contains the awaking event of q.

Proof: Let us suppose that the critical mailbox m of S_i is an output mailbox. Then, the blocking event q is an event s and the state of m is w=n . Since the DPC S^P is well formed, then there exists at least one component schema $S_j \in \Lambda_i$. As S^P is of type 1, then $\Lambda_i = S_j$. By definition 4.10, the mailbox m links S_i to S_j, then S_j contains at least one actor *receive* related to m. Since there is only one control sequence σ_j of S_j, it must contain an infinite number of events r related to m. Let p denote the last event of σ_i in σ^P. Let us consider the prefix of σ^P whose last event is p: there exists, in the prefix, a finite number, possibly zero, of events belonging to σ_j. Let p' denote the last of these events. The occurrence of the first event $q' = r$, successor of p' in σ_j, should modify the state of the mailbox m from w=n to w=n-1 and then it should make feasible q. Therefore q' is the awaking event of q. A similar argument can be developed if the blocking event q is an event r.

Q.E.D.

From Lemma 5.1 the following theorem arises:

Theorem 5.1: Given a σ^P of a DPC S^P of type 1, if the control sequence σ_i of the schema S_i is deadlocked, then the control sequence σ_j of the schema $S_j = \Lambda_i$ is deadlocked.

In other words, a *unique* control sequence *can not exist* in σ^P.

By Theorem 5.1, if σ_i is deadlocked in σ^P, then also σ_j is deadlocked, where $S_j=\lambda(S_i)=\Lambda_i$, and then also σ_k is deadlocked, where $S_k=\lambda(S_j)=\lambda(\lambda(S_i))$. This will be denoted, briefly, by:

$$S_k = \Lambda^2(S_i)$$

Then, we can state the following Lemma:

Lemma 5.2: Given a control sequence σ^P of a DPC S^P of type 1, if a deadlock occurs in σ^P, then, for each S_i whose control sequence σ_i is deadlocked, there is an infinite semipath, composed by schemata whose control sequences are deadlocked:

$$\Lambda_i^o, \Lambda_i, \Lambda_i^2, \ldots, \Lambda_i^j, \ldots$$

where $\Lambda_i^o=S_i$.

Definition 5.2: Under the hypothesis of Lemma 5.2, the semipath $\Lambda_i^o, \Lambda_i, \Lambda_i^2, \ldots, \Lambda_i^j, \ldots$ is called the *critical semipath* related to S_i.

Now we give a necessary condition for the occurrence of a deadlock, both a partial and a total deadlock, in a control sequence σ^P of a DPC of type 1. This condition is related to the connection topology of the component schemata.

Theorem 5.2: Given a control sequence σ^P of a DPC S^P of type 1, the existence of a semicycle in the connection graph G of S^P is a necessary condition for a deadlock in σ^P.

Proof: If a deadlock occurs in σ^P, then at least one component schema S_i exists such that the control sequence σ_i is deadlocked. Let us consider the critical semipath related to S_i:

$$\Lambda_i^o, \Lambda_i, \Lambda_i^2, \ldots, \Lambda_i^j, \ldots$$

Since the critical semipath is infinite and the number of component schemata in the DPC is finite, an integer $k \geqslant 0$ exists such that:

$$\Lambda_i^k = \Lambda_i^{k+h} \qquad (h \geqslant 2)$$

Let us prove that:

$$\Lambda_i^k, \Lambda_i^{k+1}, \ldots, \Lambda_i^{k+h}$$

is a semicycle. Let us consider the pair of consecutive schemata in the critical semipath:

$$S_a = \Lambda_i^{k+t}, \qquad S_b = \Lambda_i^{k+u}$$

(where $t=|j-1|_h$, $u=|j|_h$, $j \geqslant 1$) and let m_a and m_b be their critical mailboxes, respectively. By definition 4.8 it is sufficient to prove that $m_a \neq m_b$. On the contrary, if $m_a=m_b=m$, the same mailbox would be critical for both S_a and S_b. Since m is one-to-one, it should be an input mailbox for S_a (S_b) and an output mailbox for S_b (S_a) respectively. Then, the state of m should be $w=o$ and $w=n$ at the same time. This is a nonsense and then $m_a \neq m_b$.

Q.E.D.

From the proof of above theorem, the following corollary arises:

Corollary 5.1: Given a DPC S^P of type 1, each critical semipath contains at least a semicycle.

If a schema has the control sequence deadlocked, in addition to the schemata of its critical semipath, other schemata have their control sequences deadlocked. More precisely:

Lemma 5.4: Given a control sequence σ^P of a DPC S^P of type 1, if a deadlock occurs in σ^P, then, for any deadlocked control sequence σ_i, every schema S_j such that a path exists in the connection graph G from S_i to S_j has its control sequence deadlocked.

Proof: Let

$$S_i, S_{p_1}, \ldots, S_{p_h}, S_j$$

be a path from S_i to S_j. Then, an output mailbox m of S_i exists such that $\Lambda_{i,m} = S_{p_1}$. That is, σ_{p_1} must contain an infinite number of events r related to m. Let q be the last event of σ_i in σ^P and let $w=x$ be the state of m after $t(q)$. Let p denote the last event of σ_{p_1} in σ^P such that $t(p)<t(q)$. After the occurrences of at most x+1 events r that follow p in σ_{p_1}, the state of m becomes $w=o$ and then σ_{p_1} becomes deadlocked.

Since we can apply iteratively this argument to the other schemata of the path up to S_j, the theorem is proved.

Q.E.D.

In particular, if all the mailboxes of S^P have a finite capacity, then the following theorem can be proved:

Theorem 5.3: Given a control sequence σ^P of a DPC S^P of type 1, where all the mailboxes of S^P have a finite capacity, if a deadlock occurs in σ^P, then it is a total deadlock.

Proof: Let S_i be a schema whose control sequence σ_i is deadlocked. First of all, let us prove that any schema S_j such that there exists a path

$$S_j, S_{dh}, \ldots, S_{d1}, S_i$$

has its control sequence deadlocked. An input mailbox m of S_i exists such that $\Lambda_{i,m}=S_{d1}$. Then, σ_{d1} must contain an infinite number of events $\underline{s}$ related to m. Let q be the last event of σ_i in σ^P and $\mathbf{w=x}$ be the state of $\mathbf{m}$ after $t(q)$. Let p denote the last event of σ_{d1} in σ^P such that $t(p)<t(q)$. After the occurrence of at most n-x+1 (by hypothesis a finite number) events $\underline{s}$ that follow p in σ_{d1} the state of m becomes $\mathbf{w=n}$, then σ_{p1} becomes deadlocked.

Since we can apply iteratively this argument to the other schemata of the path up to S_j, the above proposition is proved.

From this result and from Lemma 5.2 it follows that if S_i has the control sequence deadlocked then each S_h such that a semipath exists between S_i and S_h, has the control sequence σ_h deadlocked.

By hypothesis, at least a deadlocked control sequence σ_i exists in S^P: Since the connection graph G is connected (that is, a semipath exists between S_i and each other component schema), if q is the last event of σ_i in σ^P, there exists at most a finite number of events that follow q in σ^P.

Q.E.D.

Let us state now, two important properties of any DPC S^P of type 1:

Theorem 5.4: Given a DPC S^P of type 1, if a partial deadlock exists in a particula control sequence σ^{P*}, then, denoting by $\Sigma=\{\sigma_{d1},\sigma_{d2},\ldots,\sigma_{di},\ldots,\sigma_{dk}\}$ the set of the deadlocked control sequences in σ^{P*} and by ν_i^* (i=1,2,...,k) the number of events of σ_{di} in σ^{P*}, for each σ^P of S^P:

$$\nu_i^* = \nu_i \quad (i=1,2,\ldots,k)$$

where ν_i is the number of events of σ_{di} in σ^P.

Proof: Let us suppose that a σ^P exists such that at least one $\sigma_{di} \in \Sigma$ has a number of events, in σ^P, $\nu_i > \nu_i^*$, and let us prove that this implies an absurdity.

Denoting by q the blocking event σ_{di} related to σ^{P*}, q must occur in σ^P. Let us consider the first two schemata of the critical semipath starting from S_{di}: S_{di} and S_{dj} (where $S_{dj}=\Lambda_{di}$). From Lemma 5.1, σ_{dj} contains the awaking event q' of q. By definition 4.13, the occurrence of q' must preceed the occurrence of q in σ^P (i.e. $t(q')<t(q)$). Then, denoting by p the blocking event of σ_{dj} related to σ^{P*}, since in σ^P $t(p)<t(q')$, it follows:

$$t(p) < t(q) \qquad (5.1)$$

Similarly, if we consider $S_{dh}=\Lambda_{dj}=\Lambda_{di}^2$, then, denoting by ℓ the blocking event of σ_{dh} related to σ^P:

$$t(\ell) < t(p) < t(q) \qquad (5.2)$$

We can apply the same argument, iterlatively, to the subsequent schemata of the critical semipath related to S_{di}. Since this semipath contains a semicycle (Corollary 5.1), then it will contain at least twice a certain schema S_{du}:

$$S_{di}, S_{dj}, S_{dh}, \ldots, S_{du}, \ldots, S_{du}, \ldots$$

Therefore, the argument that led to (5.1) and (5.2) gives the following relationship in σ^P:

$$\ldots<t(z)<\ldots<t(z)<\ldots<t(\ell)<t(p)<t(q) \qquad (5.3)$$

where z is the blocking event of σ_{du} related to σ^P. Since no event can preceed itself, (5.3) contains an absurdity, and then the hypothesis that σ_{di} has more events in σ^P than in σ^{P*} is false. The same conclusion could be obtained by assuming that $\nu_i<\nu_i^*$, exchanging in the above demostration σ^{P*} with σ^P.

Q.E.D.

Theorem 5.4 asserts that if a partial deadlock occurs in a particular control sequence of a DPC S^P of type 1, then the same partial deadlock occurs in all the control sequences of S^P. In the case of total deadlock, Theorem 5.4 implies the following corollary:

Corollary 5.2: Given a DPC S^P of type 1, if a particular control sequence σ^{P*} is finite and is composed of ν events, then every control sequence σ^P is finite and is composed by ν events.

Theorem 5.4 and Corollary 5.2 assert that for a DPC of type 1, a deadlock is independent of the particular control sequence. It only depends on both the topology of the connections among component schemata and the structure of each component schema. Modular systems modelled by DPC's of type 1 give rise, during execution, to a family of cooperating asynchronous processes. For these types of modular systems, a deadlock condition depends neither on the input data nor on the relative speeds of the component processes. For this reason we call this type of systems: *Data and time independent systems*.

6. DATA DEPENDENT AND TIME INDEPENDENT SYSTEMS

Definition 6.1: A DPC S^P of n cyclic sequential schemata is of *type 2* if all the mailboxes in S^P are one-to-one.

Any DPC of type 1 is a particular case of DPC's of type 2, where all the component schemata are data independent. When deciders are present, it is no longer true that for any cyclic sequential schema there is only one possible control sequence. Therefore, for a DPC of type 2, if a deadlock occurs in some control sequence σ^P, it is not true that for a blocking event q of a deadlocked control sequence σ_i there exists a S_j whose control sequence σ_j contains the awaking event of q. In particular, an unique deadlocked σ_i can exist in σ^P. Thus, results similar to the ones stated by Lemma 5.1, Lemma 5.2, Theorem 5.1 and Theorem 5.3 are no longer true for a DPC of type 2. For the same reason, also Theorem 5.2 is false, in general. However we can prove that it is still true only for total deadlocks.

Theorem 6.1: Given a control sequence σ^P of a DPC S^P of type 2, a semicycle in the connection graph G of S^P is a necessary condition for a total deadlock in σ^P.

Proof: Let S_{d1} be a component schema of S^P. By hypothesis, σ_{d1} is deadlocked in σ^P. Let us consider $S_{d2}=\Lambda_{d1}$: By hypothesis, σ_{d2} is also deadlocked. We can apply iteratively this argument to S_{d2}, and so on. In this way, we detect an infinite semipath whose schemata have their control sequences deadlocked in σ^P:

$$\Lambda^o_{d1},\Lambda_{d1},\Lambda^2_{d1},\ldots,\Lambda^i_{d1},\ldots$$

By applying the same proof of Theorem 5.2 it follows that the above semipath contains a semicycle.

Q.E.D.

Theorem 5.4 is a consequence of the characteristics of a DPC of type 1, that is, all the mailboxes are one-to-one and for each component schema there is only one control sequence. Here, Theorem 5.4 is no longer true. However, an analogue Theorem can be proved, if we choose a particular interpretation of actors *send* and *receive*; namely, values are written by *send* in the cells of the mailbox vector

N, one at a time, and they are read by *receive* exactly in the same order. In other words the vector N is a FIFO queue. For any DPC of type 2, let us denote by Ω the set of interpretations for which actors *send* and *receive* have the above specification.

We can prove that for any DPC of type 2, given an interpretation $\omega \in \Omega$ and an input assignment, there is only one possible control sequence for any componer schema:

Lemma 6.1: Given a DPC S^P of type 2, for each interpretation $\omega \in \Omega$ and for each input assignment, each component schema S_i has the same control sequence σ_i in all the control sequences of S^P.

Proof: Let us suppose that two control sequences σ_1^P and σ_2^P exist such that the component schema S_i has two different control sequences σ_{i1} and σ_{i2} in σ_1^P and σ_2^P respectively. Let us proof that this implies an absurdity.

Since the schemata are sequential, by definition 3.1 (iii) σ_{i1} and σ_{i2} have the same prefix $\pi=ab...p$. That is, p is the last event after which σ_{i1} and σ_{i2} are different. By definition 3.1 (i), p is the initiation event $\bar{d}$ of a decider d. This implies that, after the occurrence of the events of π, the values of some variables in the domain X(d) are different in σ_{i1} and σ_{i2}. Since σ_1^P and σ_2^P have, by hypothesis, the same input assignment, it can be proved that S_i must contain at least one input mailbox m from which different sequences of values are read in σ_{i1} and σ_{i2}. Let x be the first event r belonging to S_i that reads different values from m. It results, in σ_1^P and σ_2^P respectively:

$$t_1(x) < t_1(p)$$
$$t_2(x) < t_2(p)$$

Let us consider now the schema $S_j=\Lambda_{i,m}$. Since the interpretation belongs to Ω, it follows that σ_{j1} and σ_{j2} exist, in σ_1^P and σ_2^P respectively, such that different sequences of values are written in m. Let y be the first event s belonging to S_j that writes different values in m. It results, in σ_1^P and σ_2^P respectively:

$$t_1(y) < t_1(x) < t_1(p)$$
$$t_2(y) < t_2(x) < t_2(p)$$

As well as S_i, also S_j must have at least one input mailbox m' from which different sequences of values are read in σ_{j1} and σ_{j2}. Let z be the first event r belonging to S_j that reads different values from m'. It results, in σ_1^p and σ_2^p respectively:

$$t_1(z) < t_1(y) < t_1(x) < t_1(p) \qquad (6.1)$$
$$t_2(z) < t_2(y) < t_2(x) < t_2(p)$$

By applying iteratively the above argument to S_j, $\Lambda_{j,m'}$, and so on, we should find two endless chains of inequalities with their rightmost sides coinciding with (6.1):

$$\ldots < t_1(z) < t_1(y) < t_1(x) < t_1(p)$$
$$\ldots < t_2(z) < t_2(y) < t_2(x) < t_2(p)$$

This is an absurdity, because only a finite number of event can preceed z in any control sequence of S^p (see Definition 4.4 (iii)).

Q.E.D.

Now an proposition analogous to Theorem 5.4 can be stated:

Theorem 6.2: Given a DPC of type 2, if, for each interpretation $\omega \in \Omega$ and for each input assignment, a partial deadlock exists in a particular control sequence σ^{p*}, then, denoting by $\Sigma=\{\sigma_{d1},\ldots,\sigma_{di},\ldots,\sigma_{dk}\}$ the set of the deadlocked component control sequences in σ^{p*} and by ν_i^* (i=1,2,...,k) the number of events of σ_{di} in σ_*^p, under the same interpretation and the same input assignment, for each σ^p of S^p:

$$\nu_i^* = \nu_i \quad (i=1,2,\ldots,k)$$

where ν_i is the number of events of σ_{di} in σ^p.

Proof: Let us suppose that a σ^p exists such that one $\sigma_{di} \in \Sigma$ has a number of events, in σ^p, $\nu_i > \nu_i^*$ and let us prove that this implies an absurdity.

Denoting by q the blocking event of σ_{di} related to σ^{p*}, q must occur in σ^p. Let us consider $S_{dj}=\Lambda_i$. Its control sequence σ_{dj}, that by Lemma 6.1 is the same that occurs to form both σ^p and σ^{p*}, contains the awaking event of q. Denoting by p the blocking event of σ_{dj} related to σ^{p*}, it may be shown, as in Theorem 5.4, that, in σ^p,

$$t(p) < t(q)$$

We can apply iteratively this argument to S_{dj} and so on. In this way, we obtain an infinite semipath of schemata belonging to Σ. This semipath is analogous to the critical semipath in Theorem 5.4. From now on, the proof can go on in a simila way.

Q.E.D.

From Theorem 6.2 a proposition analogous to Corollary 5.2 can be stated:

Corollary 6.1: Given a DPC of type 2, if, for some input assignment, a particular control sequence σ^p_* is finite and composed by k events, then, for the same interpretation and for the same input assignment, every control sequence σ^p is finite and composed by the same number k of events.

Theorem 6.2 enables us to say that in any modular system modelled by a DPC of type 2 a total deadlock is independent of the relative speeds of the component processes. However, a total deadlock depends on the input data. In other words, if, for certain input data, a total deadlock occurs during a computation, then, for the same input data, the same deadlock condition occurs in any computation. For this reason we call this type of systems: *Data Dependent and Time Independent Systems*.

We want to emphasize that the above results hold only for an interpretation $\omega \in \Omega$. However, considering a mailbox as a FIFO queue is not a strong limitation.

A DPC of type 2 is a particular case of a DPC were all the mailboxes are one--to-one. If a DPC is not of type 2, Theorem 6.2 is no longer true. In other words, for a modular system modelled by a general type of DPC a deadlock depends on the input data *and* on the relative speeds of the component processes. We call this type of systems *Data and Time dependent*.

6. CONCLUSIONS

Deadlock conditions, that can arise in asynchronous systems, owing to interprocess communication activity, are analyzed. Systems are classified in three catagories, each of which has particular properties as far as deadlock conditions are considered.

Systems modelled by DPC's of type 2 are particurarly interesting. In fact they are sufficiently general. Furthermore, for any interpretation $\omega \in \Omega$, they belong to the class of Patil's β systems [4], that is they are functional systems.

7. REFERENCES

[1] J.B. Dennis: *Course Notes "Computation Structures"* Department of electrical Engineering, MIT 1967.

[2] J.P. Linderman: *"Productivity in Parallel Computation Schemata"*, MIT Project MAC-TR-111, December 1973.

[3] E.W. Dijkstra: *"Computing Sequential Processes"* Programming Languages, F. Genuys ed., Academic Press, 43-112, (1968).

[4] S.S. Patil: *"Closure Properties of interconnections of determinate systems"* Record - Project MAC Conference on Concurrent Systems and Parallel Computation, ACM, N.Y. 1970.