CONTROL NETS FOR ASYNCHRONOUS SYSTEMS, PART I

by

O. Herzog * and M. Yoeli **

Technical Report No. 74

May 1976

* University of Dortmund. His research was supported by the "Deutscher Akademischer Austauschdienst", Bonn, Fed. Rep. of Germany.

** Technion - Israel Institute of Technology, Haifa, Israel.
1. INTRODUCTION

Many digital systems may be considered as consisting of two parts: a data flow structure or device structure and a control structure \([1,2]\). The data flow structure consists of specific devices, such as registers, adders, counters, etc. The control structure supervises the activities and sequencing of these devices.

We assume all devices to operate asynchronously. Such a device is given a start or GO command \([2]\) (also referred to as ready signal \([1]\)) by the control structure to start its operation. Upon completion of its task the device returns a completion or DONE signal \([2]\) (acknowledge signal \([1]\)). In this paper we are not concerned with the various methods by which control commands and signals may be realized (see \([1,2]\)).

We assume that each device performs some specific task (e.g. addition). The extension of our considerations to devices with multi-task capabilities is straightforward. The sequence in which given tasks are to be initiated by a control structure is conveniently specified by means of a task flow chart \([2]\), an example of which is shown in Fig. 1. This chart involves parallel processing, namely tasks TA2 and TA3 are to be initiated concurrently after the completion of task TA1. Only after the completion of both task TA2 and task TA3, Task TA4 is to be started. The two-way branch symbol ("decision box") has the usual meaning. Thus, the truth value of the proposition \(P_1\) determines whether
task TA5 or task TA6 is to be carried out.

With the control structure of Fig. 1 we associate the control graph shown in Fig. 2. In such a graph tasks are represented by edges. The nodes represent basic control modules. E.g., a JOIN is a two-input one-output module, which issues a GO command after having received DONE signals on both of its input lines. One easily derives the performance of the other modules from a comparison of Fig. 2 with Fig. 1. In the next section we introduce a precise definition of a control graph and its dynamics. We also indicate the relationship between control graphs and Petri nets.

In this paper we consider control graphs which may be formed by means of the module types shown in Fig. 2, and study essential properties of such graphs. Formal definitions of these properties are given in the next section. Presently we introduce the relevant concepts informally.

We make the assumption that once a device is initiated (i.e. given a GO command), the control structure must wait for the DONE signal, before initiating the device a second time. If a control structure adheres to this rule, we call the corresponding control graph safe.

If a control structure is given the BEGIN command, we expect it to eventually reach a "final" state, i.e. to issue the END signal. A control
structure is deadlocked, if it is in a state, from which no final state can be reached (even if all deciders decide "favorably"!). A control graph is live if the corresponding control structure can never become deadlocked (even if all deciders decide "unfavorably"!).

After a control structure has issued the END signal, we expect it to return to its "initial" state, in which it is ready to accept another BEGIN command. If a control structure has this property, we call the corresponding control graph residue-free. In the sequel we turn our interest to control graphs which are well-formed, i.e. safe, live and residue-free.

There exists an extensive literature in which mathematical models of parallel computations are proposed and investigated (see [3] for a survey of this literature). However, these models differ considerably from the control graphs discussed in this paper. Modular control structures similar to ours, as well as more general ones, are studied in [1], but the emphasis in [1] is on methods of implementation. The present paper is closely related to [4], which considers well-formed control structures of parallel programs. Loop-free control graphs are investigated in [5].

II. CONTROL GRAPHS.

In this section we introduce the concepts of control graph and control net, and point out the relationship of control nets to Petri nets (for a brief introduction to Petri nets see [1,3]; for more formal and extensive discussions see e.g. [6,7,8]).
Definition 2.1 A control graph is a finite, directed graph, the nodes of which are partitioned into two disjoint sets A and B, and which satisfies the following conditions.

a) Multiple edges are admitted.

b) There exists exactly one node in A with indegree 0 and outdegree 1 (START).

c) There exists exactly one node in A with indegree 1 and outdegree 0 (HALT).

d) Any other node in A has either indegree 1 and outdegree 2 (FORK) or indegree 2 and outdegree 1 (JOIN).

e) Any node in B has either indegree 1 and outdegree 2 (DECIDER) or indegree 2 and outdegree 1 (UNION).

f) There exists a directed path from the START node to each node V ≠ START.

g) There exists a directed path from each node V ≠ HALT to the HALT node.

One easily verifies that a control graph cannot have self-loops (i.e. cycles of length 1).

One example of a control graph was given in Fig. 2. Fig. 3 shows another example of a control graph. A-type nodes are indicated by circles (⊙) and B-type nodes by squares (□).
Definition 2.2. Let G be a control graph. A marking m of G is a function \( m : E \rightarrow \mathbb{N} \), where E is the edge set of G and \( \mathbb{N} \) is the set of nonnegative integers. A control net is an ordered pair \( \langle G, m \rangle \), where G is a control graph and m is a marking of G.

Let e be an edge of the control net \( \langle G, m \rangle \). We refer to \( m(e) \) as the number of tokens on e. If \( m(e) > 0 \), we say e is marked (under m). In the graphical representation of control nets tokens are indicated by black dots. Fig. 4 shows examples of control nets having a common control graph.

Definition 2.3. Let \( \langle G, m \rangle \) be a control net. An A-type node \( V \neq \text{START} \) is enabled iff all incoming edges of V are marked. A B-type node is enabled iff at least one of its incoming edges is marked.

A node which is enabled may fire. The firing rules are as follows.

Definition 2.4. Assume the A-type node V of some control net is enabled. The firing of V decreases the marking of its incoming edges by 1, and increases the marking of its outgoing edges by 1. If a B-type node is enabled, its firing decreases the marking of one of its marked incoming edges by 1, and increases the marking of one of its outgoing edges by 1.
In Fig. 4(a), nodes F and D are enabled, whereas in Fig. 4(b) nodes J and D are enabled. The firing of F in Fig. 4(a) yields Fig. 4(b). The firing of D in Fig. 4(b) may yield either Fig. 4(c) or Fig. 4(d).

Control nets are easily represented as Petri nets. E.g., the Petri net of Fig. 5 may be regarded as a representation of the control net of Fig. 4(b). Clearly, any such Petri net representation of a control net belongs to the class of free-choice Petri nets [1,6].
III. WELL-FORMED CONTROL GRAPHS

In this section we define well-formed control graphs and state necessary and sufficient conditions for a control graph to be well-formed.

Let \( m \) and \( m' \) be markings of the control graph \( G \). We write \( m \rightarrow m' \) to indicate that the marking \( m' \) is obtainable from the marking \( m \) by firing node \( V \). We write \( m \rightarrow m' \) to state that \( m' \) is reachable from \( m \) by the successive firing of one or more nodes of \( G \). Also, we set

\[
[m] = \{ m' \mid m \rightarrow m' \} \cup \{m\},
\]

i.e. \([m]\) is the set of all markings reachable from \( m \), including \( m \).

**Definition 3.1** The initial marking \( m_I \) of a control graph \( G \) is defined as follows:

If \( e \) is the outgoing edge of the START node, then \( m_I(e) = 1 \),

else \( m_I(e) = 0 \).

A marking \( m \) of \( G \) is final, if the HALT node of \( G \) is enabled in \((G, m)\). We denote by \( M_F \) the set of all final markings of \( G \).
Definition 3.2 A control graph $G$ with edge set $E$ is safe, iff

$$(\forall m \in [m_I]) \ (\forall e \in E) : m(e) \leq 1 ,$$

i.e. the number of tokens on any edge $e$ cannot exceed 1, under any marking $m$ reachable from $m_I$.

Definition 3.3 A control graph $G$ is live, iff

$$(\forall m \in [m_I]) \ (\exists m' \in M_F) : m' \in [m] ,$$

i.e. if $m$ is reachable from $m_I$, then there exists a final marking reachable from $m$.

Definition 3.4 A control graph $G$ is residue-free iff

$$(\forall m \in [m_I] \cap M_F : \sum_{e \in E} m(e) = 1 ,$$

i.e. for any final marking $m$ reachable from $m_I$, the control net $\langle G, m \rangle$ contains exactly one token (namely on the incoming edge of the HALT node).
Definition 3.5 A control graph $G$ is well-formed, iff $G$ is safe, live, and residue-free.

The control graphs shown in Fig. 2 and Fig. 3 are both well-formed. A further example of a well-formed control graph is given in Fig. 6.

We shall be concerned with necessary and sufficient conditions for control graphs to be well-formed. In order to specify such conditions, we need the following two definitions. By path we always mean a directed path.

Definition 3.6 Let $F$ be a FORK and $J$ a JOIN of some control graph $G$. We say that $J$ belongs to $F$ iff there exist two edge-disjoint paths from $F$ to $J$.

Definition 3.7 Let $P_1$ and $P_2$ be edge-disjoint paths of some control graph $G$. $P_2$ is independent of $P_1$ iff the following condition is satisfied: For every path $P$ edge-disjoint to both $P_1$ and $P_2$, which starts in a node $V_1$ on $P_1$ and ends in a node $V_2$ on $P_2$, $V_1$ is a FORK and $V_2$ is a JOIN.
In Section IV we prove the following main result of this paper.

**THEOREM 1.** If a control graph \( G \) is well-formed then the following six conditions are satisfied by \( G \).

**CONDITION 1.** Let \( J \) be a \textit{JOIN} of \( G \), and \( P_1 \) a path from the \textit{START} node to \( J \). Then there exists a path \( P_2 \), edge-disjoint to \( P_1 \), which originates in some intermediate node of \( P_1 \) and terminates on \( J \).

**CONDITION 2.** Let \( F \) be a \textit{FORK} of \( G \), \( U \) a \textit{UNION}, and assume there exist two edge-disjoint paths \( P_1 \) and \( P_2 \) from \( F \) to \( U \). Then there exists a \textit{JOIN} \( J \) on \( P_1 \) or \( P_2 \) which belongs to \( F \).

**CONDITION 3.** Let \( D \) be a \textit{DECIDER} of \( G \), \( J \) a \textit{JOIN}, and assume there exist two edge-disjoint paths \( P_1 \) and \( P_2 \) from \( D \) to \( J \). Then there exists a \textit{FORK} \( F_1 \) on \( P_1 \) and a \textit{FORK} \( F_2 \) on \( P_2 \), such that \( J \) belongs to both \( F_1 \) and \( F_2 \).

**CONDITION 4.** Let \( F \) be a \textit{FORK} of \( G \), \( D \) a \textit{DECIDER}, and \( J \) a \textit{JOIN}. Let \( P \) be a path from the \textit{START} node to \( F \). Assume there exists a path \( P_1 \) from \( F \) to \( J \), a path \( P_2 \) from \( F \) to \( D \), and a path \( P_3 \) from \( D \) to \( J \).
Assume further that $P_1$ and $P_2$, as well as $P_1$ and $P_3$ are edge-disjoint, that $P$ is edge-disjoint to $P_1$, $P_2$ and $P_3$, and that $P_1$ is independent of $P_2$. Then there exists a path $P_4$ from $D$ to a node either on $P_2$ or $P_3$, such that $P_4$ is edge-disjoint to $P_3$.

**CONDITION 5.** Let $F$ be a FORK of $G$ on a cycle $C$. Then there exists a JOIN $J$ on $C$ which belongs to $F$.

**CONDITION 6.** Let $J$ be a JOIN of $G$ on a cycle $C$. Then there exists a FORK $F$ on $C$, such that $J$ belongs to $F$.

In Part II of this paper [8] we show that the above six conditions are also sufficient, i.e. if they are satisfied by a control graph $G$, then $G$ is well-formed.
IV. PROOF OF THEOREM 1

In this section we prove Theorem 1 of Section III. The proof is based on the following lemmata.

**LEMMA 1** Let $G$ be a well-formed control graph, and $J$ a JOIN of $G$. Let $m$ be a marking reachable from the initial marking $m_I$, such that $m(e) = 1$, where $e$ is an incoming edge of $J$. Then there exists a marking $m' \in [m]$ such that $J$ is enabled.

**PROOF** Assume that such a marking $m'$ does not exist. Since $G$ is live, there exists a final marking $m_F \in [m]$. It follows that $m_F(e) \geq 1$. Consequently $G$ is not residue-free, in contradiction to our assumption that $G$ is well-formed.

**LEMMA 2** Let $G$ be a well-formed control graph, $v$ a node of $G$, and $e$ an incoming edge of $v$. Assume that $e$ is marked under $m \in [m_I]$. Then there exists a marking $m' \in [m]$ such that $v$ is enabled.

**PROOF** If $v$ is not a JOIN, then $v$ is already enabled under $m$, i.e. $m' = m$. If $v$ is a JOIN, then Lemma 2 follows immediately from Lemma 1.
**LEMMA 3** Let $G$ be a well-formed control graph, and $P$ a path in $G$, consisting of the edge sequence $e_1, \ldots, e_n$. Assume $m(e_i) = 1$ for $m \in [m_1]$. Then there exists a marking $m' \in [m]$ such that $m'(e_n) = 1$.

**PROOF** We use induction on $n$. The case $n=1$ is trivial. For $n>1$, assume there exists a marking $m_{n-1} \in [m]$, such that $m_{n-1}(e_{n-1}) = 1$.

Let $v$ be the node between $e_{n-1}$ and $e_n$. By Lemma 2, there exists a marking $m_{n-1}' \in [m_{n-1}]$ such that $v$ is enabled. It follows that $m'(e_n) = 1$ for a marking $m'$, where $m_{n-1}' + m'$. Thus $m' \in [m]$.

**LEMMA 4** Let $G$ be a well-formed control graph. Then Condition 5 of Theorem 1 is satisfied.

**PROOF** Let $F$ be a FORK of $G$ on a cycle $C$. By Definition 2.1, condition f, there exists a path from the START node to $F$. Hence, by Lemma 3, $F$ is enabled under some marking $m \in [m_1]$. Let $m + m'$. Now assume $F$ has an outgoing edge $e$ not on $C$ which has no continuation entering $C$. Then $m''(e) = 1$. By Lemma 3, there exists a marking $m''' \in [m'']$ such that $F$ is again enabled. Clearly, we may assume that $m'''(e) = m''(e) = 1$. Now, let $m'' + m'''$. Then $m''''(e) > 1$, i.e. $G$ is not safe, in contradiction to our assumption that $G$ is well-formed. It follows that $e$ must have a continuation entering $C$, say at node $v$. If $v$ is a UNION, $G$ will again be not safe, hence $v$ is a JOIN. Thus Condition 5 of Theorem 1 is satisfied.
LEMMA 5 Let G be a well-formed control graph. Then Condition 1 of Theorem 1 is satisfied.

PROOF Let J be a JOIN of G, and P₁ a path from the START node to J. There must exist another such path P₂. If Condition 1 of Theorem 1 is not satisfied, then J must be on a cycle C, and no path can exist from an intermediate node of P₁ to C, which is edge-disjoint to P₁. Let e be the last edge of P₁. We may assume that e is not on C. By Lemma 3, m(e) = 1 for some marking m ∈ [m₁]. One easily verifies that C cannot be marked under m, nor under any marking reachable from m. It follows that there exists no marking m' ∈ [m] such that J is enabled. This contradicts Lemma 1. Hence Condition 1 of Theorem 1 must be satisfied.

LEMMA 6 Let G be a well-formed control graph. Then G satisfies Condition 3 of Theorem 1.

PROOF Let D be a DECIDER of G, J a JOIN, and assume the existence of two edge-disjoint paths P₁ and P₂ from D to J. Assume now that one of the paths, say P₁, contains no FORK F, such that J belongs to F. Without loss of generality we may also assume that the path P₁ contains no DECIDER D' ≠ D having the following property: there exists a path from D' to J which is edge-disjoint to P₁.
Let \( e \) be the incoming edge of \( D \). By Lemma 3 there exists a marking \( m \in [m] \) such that \( m(e) > 0 \). Since \( G \) is safe, \( m(e) = 1 \). Let \( e_1 \) and \( e_2 \) be the initial edges of \( P_1 \) and \( P_2 \), respectively. Then there exists a marking \( m' \) such that \( m \xrightarrow{D} m' \), \( m'(e_1) = 1 \), and \( m'(e_2) = 0 \).

Let us assume that there exists a path \( P_3 \), edge-disjoint to \( P_2 \), which meets \( P_2 \) in a node \( v \) and contains an edge \( e_3 \) marked under \( m' \). Let \( v \) be the first node on \( P_2 \) having this property. One easily verifies that \( P_3 \) is edge-disjoint to \( P_1 \) and that the part of \( P_2 \) up to \( v \) cannot be marked under \( m' \). Let now \( e_4 \) and \( e_5 \) be the incoming edges of \( v \) and \( P_2 \) and \( P_3 \), respectively. By Lemma 3, there exists a marking \( m'' \in [m'] \) such that \( m''(e_3) = 1 \). Clearly, \( m''(e_4) = 0 \). If \( v \) is a JOIN, one verifies that \( v \) cannot be enabled under any marking reachable from \( m'' \), in contradiction to Lemma 1. Hence \( v \) must be a UNION. But then there also exists a marking \( m''' \) such that \( m \xrightarrow{D} m''' \), \( m'''(e_1) = 0 \), and \( m'''(e_2) = 1 \), as well as a marking \( m'''' \in [m] \) such that \( m''''(e_4) = m''''(e_5) = 1 \). Thus \( G \) becomes unsafe. It follows that no path \( P_3 \) having the above properties exists.

Let \( e_6 \) and \( e_7 \) be the incoming edges of \( J \) on \( P_1 \) and \( P_2 \), respectively. By Lemma 3, there exists a marking \( m''''' \in [m'] \) such that \( m'''''(e_6) = 1 \). One verifies that \( J \) cannot be enabled under any marking in \([m''''']\), in contradiction to Lemma 1. It follows that there must exist a FORK \( F \) on path \( P_1 \) such that \( J \) belongs to \( F \). Hence \( G \) must satisfy Condition 3 of Theorem 1.
LEMMA 7  
Let $G$ be a well-formed control graph. Then Condition 6 of Theorem 1 is satisfied.

PROOF  
Let $J$ be a JOIN on a cycle $C$. Then there exists a path $P_1$ from the START node to $J$. We distinguish between the following two cases.

Case 1: The path $P_1$ enters the cycle $C$ at the node $J$. By Lemma 5, there exists a path $P_2$, edge-disjoint to $P_1$, which starts at some intermediate node $v$ of $P_1$, and terminates on $J$. If $v$ is a FORK, one easily verifies that $G$ is either not live or not residue-free. If $v$ is a DECIDER, Lemma 6 applies. If no FORK exists on $C$, to which $J$ belongs, such a FORK must exist on the part of $P_2$ which is outside $C$. But then one again verifies that $G$ is either not live or not residue-free.

Case 2: The path $P_1$ enters the cycle $C$ at some node $v \neq J$. By Lemma 5, there again exists a path $P_2$, edge-disjoint to $P_1$, which starts at some intermediate node of $P_1$, and terminates on $J$. If $P_2$ has no node $v' \neq J$ in common with $C$, then Case 1 applies. If $P_2$ has a node $v' \neq J$ in common with $C$, we may assume that the part of $P_2$ from $v'$ to $J$ is edge-disjoint to $C$. If $v'$ is a FORK, Condition 6 of Theorem 1 holds. If $v'$ is a DECIDER, Lemma 6 is applicable yielding again Condition 6 of Theorem 1.
LEMMA 8 Let $G$ be a well-formed control graph. Then Condition 2 of Theorem 1 is satisfied.

PROOF Let $F$ be a FORK of $G$, $U$ a UNION, and $P_1$, $P_2$ edge-disjoint paths from $F$ to $U$. Let $P$ be a path from the START node to $F$.

Since $G$ is safe, there must be a JOIN $J$ on $P_1$ or $P_2$, say on $P_1$, which "stops a token travelling along $P_1"$ or "absorbs a token from $P_2"$.

Without loss of generality we assume that no other FORK $F'$ exists on $P_1$ between $F$ and $J$, with two edge-disjoint paths between $F'$ and $U$.

In view of Lemmata 5 and 6 and our assumption, there exists a FORK $F_1$ on a path from the START node to $F$, such that $J$ belongs to $F_1$. If both paths from $F_1$ to $J$ are edge-disjoint with $P_2$, then $J$ will neither "stop a token on $P_1"$ nor "absorb a token from $P_2"$, contradicting our assumption about $J$.

Thus there exists a path from $F_1$ to $J$ which is not edge-disjoint to $P_2$. But then $J$ belongs to $F$, i.e. Condition 2 of Theorem 1 is satisfied.
LEMMA 9 If G is a well-formed control graph, then Condition 4 of Theorem 1 is satisfied.

PROOF Under the assumption of Condition 4 of Theorem 1, assume that no path $P_4$ exists, having the required properties. Let $e_1$ and $e_2$ be the outgoing edges of F on $P_1$ and $P_2$, respectively. There exists a marking $m \in [m_1]$ such that $m(e_1) = m(e_2) = 1$, but $P_3$ is not marked under $m$. Let $e_3$ and $e_4$ be the last edges of $P_1$ and $P_2$, respectively. In view of the independence of $P_1$ from $P_2$, there exists a marking $m' \in [m]$ such that $m'(e_3) = m'(e_4) = 1$. Let $m' \rightarrow m''$, where $m''(e_5) = 1$, $e_5$ being the outgoing edge of D not on $P_3$. Similarly to the proof of Lemma 6, one verifies that no path $P$ exists which is marked and meets $P_3$. It follows that $P_3$ cannot become marked, and consequently J cannot be enabled, in contradiction to Lemma 1. Hence Condition 4 of Theorem 1 must be satisfied.

We have shown, by means of Lemmata 4-9, that a well-formed control graph satisfies Conditions 1-6 of Theorem 1. This completes the proof of Theorem 1. In Part II of this paper [9] we show that any control graph satisfying Conditions 1-6 of Theorem 1 is well-formed.
REFERENCES


Fig. 1: Example of task flow chart.
Fig. 2: Control graph corresponding to Fig. 1.

Diagram:

- Start
- Fork
- Join
- Decider
- Union
- Tail
- Tail 6
- Tail 5
- Tail 4
- Tail 3
- Tail 2
- Tail 1
- Halt
Fig. 3 : Example of control graph.
Fig. 4: Example of control nets.
Fig. 5: A Petri net representation of Fig. 4(b).

Fig. 6: A well-formed control graph.