ON STRUCTURED PARALLEL CONTROL SCHEMATA

by

Z. Barzilai, M. Yoeli and E. Strasbourger

Technical Report #151

April 1979
ABSTRACT

This research deals with the structuring of control networks for parallel systems. Sufficient conditions are established for a well-formed parallel control schema to be convertible into a strongly equivalent schema structured by means of the primitives COMPOSITION, IFTHENELSE, DOWHILE and PARDO.

Keywords: Parallel systems, structured programming, flowcharts.
1. INTRODUCTION

Structured programming has become an important methodology for the design of correct computer programs. The interest in structured programming has also led to significant theoretical results as to the possibilities of structuring a given flowchart ([KN-FL], [KOS], [MIL], [LE-MA], [IND], [ELG]).

On the other hand, considerable interest presently exists as to the design of parallel computation and control structures ([BR-AL], [PA-DE], [YO-BR], [VAL], [CO-VA]). The arguments in favor of a structured approach to sequential programming evidently also apply to the correct design of parallel systems, especially in view of the additional difficulties of system verification due to the concurrencies involved. Consequently, a number of researchers have devoted efforts to various aspects of structured parallel systems and structured parallel programming ([PAT], [YOE], [BA-ST-YO], [ME-MA], [CO-MA], [CO-LE]).

In this paper, based on [HE-YO], [YOE], and [BA-ST-YO], we first introduce the concept of parallel control schema as extension of flowchart (schema). We then suitably extend the concept of D-chart [KOS], and state sufficient conditions for "well-formed" parallel control schemata to be "equivalent" to structured schemata. Although there is some similarity between our work and [CO-LE], there are major differences as to the formulation of the structuring problem. Consequently the structuring algorithms are essentially different.
2. PARALLEL CONTROL SCHEMATA

Many digital systems may be considered as consisting of two parts: a device structure and a control structure [PAT], [BR-YO], [YO-BR]. The device 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 GO command by the control structure to start its operation. Upon completion of its task the device returns a DONE signal. Furthermore, the device structure provides status information to the control structure, by means of suitable level signals.

We assume that each device (or operational unit) performs some specific task (e.g. addition). In order to specify the sequence in which the various operational units are to perform their task, we introduce a suitable extension of the concept of flowchart, namely parallel control schema (PCS).

A parallel control schema (PCS) $S$ consists of the following:

1. A finite, directed graph $G(S)$, the nodes of which are partitioned into seven types, as indicated in Figure 1.

2. A finite alphabet $\Sigma$ of operation letters. Every OPERATION node of $G(S)$ is labeled by a letter of $\Sigma$.

3. A finite alphabet $\Pi$ of predicate letters. Every DECIDER node $D$ of $G(S)$ is labeled by a letter of $\Pi$. Furthermore, one outgoing edge of $D$ is labeled $T$ (true), and the other edge $F$ (false).
<table>
<thead>
<tr>
<th>NODE TYPE</th>
<th>INDEGREE</th>
<th>OUTDEGREE</th>
<th>GRAPHICAL REPRESENTATION</th>
</tr>
</thead>
<tbody>
<tr>
<td>START (ST)</td>
<td>0</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>HALT (H)</td>
<td>1</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>DECIDER (D)</td>
<td>1</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>UNION (U)</td>
<td>2</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>FORK (F)</td>
<td>1</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>JOIN (J)</td>
<td>2</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>OPERATION (OP)</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
</tbody>
</table>

Figure 1: Node types of parallel control schemata

Every PCS $S$ satisfies the following conditions:

(a) $S$ has exactly one START node and exactly one HALT node.

(b) If $v$ is a node of $S$, and $v \neq$ START, then there exists a directed path from START to $v$.

(c) If $v$ is a node of $S$, and $v \neq$ HALT, then there exists a directed path from $v$ to HALT.

(d) $S$ has no multiple edges, i.e. there exists at most one edge from any node $v_1$ to any other node $v_2$. 
One easily verifies that $S$ contains no self-loops (i.e. cycles of length 1). The concept of PCS introduced here is a slight modification of the concept of control net defined in [HE-YO]. An example of a PCS is shown in Figure 2.

![Diagram of PCS](image)

**Figure 2**: Example of PCS

A formal definition of the control dynamics of a PCS is given below. Informally, DECIDER and UNION nodes correspond to predicate and collecting nodes of conventional flowcharts [MIL] respectively. Thus, a DECIDER
directs control from its in-edge to one of its two out-edges. A UNION
passes control from one of its in-edges to its out-edge. On the other
hand, a FORK passes control from its in-edge to both of its out-edges.
A JOIN passes control to its out-edge, provided control signals appear on
both its in-edges. To start the activity of a PCS, the START node issues
a control signal. The activity terminates, whenever a control signal arrives
at the HALT node.

For any operation letter $\sigma_k \in \Sigma$ representing a device (operational
unit) of the device structure, we denote by $\tilde{\sigma}_k$ the initiation of the
corresponding operation, and by $\bar{\sigma}_k$ its termination. Then a possible
activity of the PCS of Figure 2 is represented by the activity sequence

\[ \tilde{\sigma}_2 \tilde{\sigma}_1 \bar{\sigma}_2 \tilde{p} \tilde{o}_3 \bar{\sigma}_1 \tilde{\sigma}_2 \bar{\sigma}_1 \bar{\sigma}_2 \bar{p} \, . \]

In such an activity sequence $\bar{p}$ represents the outcome F (false) of testing
the predicate $p$, whereas $\tilde{p}$ represents the outcome T (true).

With any given PCS $S$ we associate its language $L(S)$, consisting
of all possible (finite and infinite) activity sequences of $S$. Two PCSs
$S_1$ and $S_2$ are (strongly) equivalent iff $L(S_1) = L(S_2)$.

We shall now provide a precise formulation of these concepts. Let $S$
bé a PCS, $E$ the set of its edges, and $A$ the set of its operational nodes.
A marking $m$ of $S$ is a function $m : E \cup A \to \{0,1\}$. The initial marking
$m_0$ of $S$ is specified as follows:

1) $m_0(a) = 0$, for every $a \in A$
2) $m_0(e_o) = 1$, where $e_o$ is the out-edge of the START node.
3) $m_0(e) = 0$, for every edge $e \neq e_o$. 

Technion - Computer Science Department - Technical Report CS0151 - 1979
The final marking $m_f$ of $S$ is specified as follows:

1) $m_f(a) = 0$, for every $a \in A$.
2) $m_f(e_f) = 1$, where $e_f$ is the in-edge of the HALT node.
3) $m_f(e) = 0$, for every edge $e \neq e_f$.

If $m(e) = 1$, we say that $e$ is marked under $m$.

Graphically, a marked edge $e$ is indicated by placing a token (\(\bullet\)) on $e$. Similarly, a marked operational node $v$ is represented by placing a token inside the square representing $v$.

A node of a PCS of type D, U, F, J, OP is firable under a marking $m$, iff the firability conditions shown in Figure 3 are satisfied. Figure 3 also shows the outcome of any firing, as well as the activity symbol associated with such a firing. We use $\lambda$ to denote the empty string.

Let $S$ be a PCS and $m$ a marking of $S$. We write $m[x]m'$ to state that the marking $m'$ is obtained from $m$ by firing a firable node with $x$ as associated activity symbol. If there exists a finite firing sequence

$$m_1[x_1]m_2[x_2]m_3, \ldots, m_{k-1}[x_{k-1}]m_k,$$

we write $m_1[w]m_k$, where $w = x_1x_2 \ldots x_{k-1}$.

We say that $m_k$ is reachable from $m_1$ iff $m_k = m_1$ or $m_1[w]m_k$ for some $w$.

Consider now an infinite firing sequence of $S$,

$$m_1[x_1]m_2, \ldots, m_1[x_i]m_{i+1}, \ldots$$

We write $m_1[\alpha]$, where $\alpha$ denotes the infinite sequence

$$\alpha = x_1x_2 \ldots x_i \ldots$$
<table>
<thead>
<tr>
<th>NODE TYPE</th>
<th>FIRABILITY CONDITIONS</th>
<th>OUTCOME OF FIRING</th>
<th>ASSOCIATED ACTIVITY SYMBOL</th>
</tr>
</thead>
<tbody>
<tr>
<td>FORK (F)</td>
<td><img src="image1" alt="Diagram" /></td>
<td><img src="image2" alt="Diagram" /></td>
<td>λ</td>
</tr>
<tr>
<td>JOIN (J)</td>
<td><img src="image3" alt="Diagram" /></td>
<td><img src="image4" alt="Diagram" /></td>
<td>λ</td>
</tr>
<tr>
<td>DECIDER (D)</td>
<td><img src="image5" alt="Diagram" /></td>
<td><img src="image6" alt="Diagram" /></td>
<td>p, p̄</td>
</tr>
<tr>
<td>UNION (U)</td>
<td><img src="image7" alt="Diagram" /></td>
<td><img src="image8" alt="Diagram" /></td>
<td>λ</td>
</tr>
<tr>
<td>OPERATION (OP)</td>
<td><img src="image9" alt="Diagram" /></td>
<td><img src="image10" alt="Diagram" /></td>
<td>σ, ̄σ</td>
</tr>
</tbody>
</table>

Figure 3: Firing rules for PCS nodes
For any given PCS \( S \), its language \( L(S) \) is defined as follows:

\[
L(S) \triangleq \{ w \mid m_0(w) > m_f \} \cup \{ \alpha \mid m_0(\alpha) \}
\]

For the PCS \( S \) of Fig. 2, we obtain

\[
L(S) = \left[ \left( \sigma_1 \| \sigma_2 \right) \bar{p} \bar{q} \right]^* \left( \sigma_1 \| \sigma_2 \right) \bar{p} \bar{q} \cup \left[ \left( \sigma_1 \| \sigma_2 \right) \bar{p} \bar{q} \right]^\omega
\]

where

\[
\sigma_1 \| \sigma_2 \triangleq \{ \sigma_1 \sigma_2, \sigma_1 \bar{p}, \sigma_2 \bar{q}, \sigma_2 \bar{q} \sigma_1 \bar{p} \bar{q} \sigma_1 \}
\]

and \( W^\omega \) denotes the set of all infinite strings over \( W \).

The language \( L(S) \) represents, in a rather evident way, the input-output behavior of the control structure represented by the PCS \( S \). Thus, if two PCSs \( S_1 \) and \( S_2 \) are equivalent, i.e. \( L(S_1) = L(S_2) \), the corresponding control structures will have the same input-output behavior, and therefore the same "degree of parallelism" (note that this is not the case in [CO-LE]!).

Alternatively, the firing rules of a PCS \( S \) and its language \( L(S) \) may be specified by means of Petri nets, as shown in [YOE].
3. STRUCTURED PARALLEL CONTROL SCHEMATA

We call the graph $G(S)$ of any PCS $S$ a Parallel Control Graph (PCG). Let $G_1$ and $G_2$ be PCGs and $v$ an OPERATION node of $G_1$. We define $G_1(v + G_2)$ to be the PCG $G$ obtained by substituting $G_2$ for $v$ in $G_1$, as indicated in Figure 4.

![Diagram showing the substitution of PCGs](image)

**Figure 4**: Substitution of PCGs
Let \( \Delta \) denote the set of primitive PCGs shown in Figure 5.

A PCS \( S \) is \( \Delta \)-structured iff its graph \( G(S) \) can be obtained from one of the \( \Delta \)-primitives of Fig. 5 by successive substitutions of OP-nodes by \( \Delta \)-primitives. A PCS \( S \) is \( \Delta \)-structurable iff it is equivalent to a PCS \( S' \) which is \( \Delta \)-structured. We say that \( S' \) is a structured version of \( S \).

In the sequel we establish (non-trivial) sufficient conditions for a PCS \( S \) to be \( \Delta \)-structurable, extending known results on the structurability of (sequential) flowcharts by means of D-charts ([Kos], [Mil], [Ba,Ma],[Tan]).
4. MAIN THEOREM

The main theorem of this paper applies to PCSs which are "well-formed" in the sense to be specified below (cf. [HE-YO]).

A PCS is properly terminating iff the following condition is satisfied:
The final marking $m_F$ is reachable from every marking $m$ which is reachable from the initial marking $m_0$.

A PCS is residue-free iff the following condition is satisfied:
If $m$ is a marking reachable from $m_0$ and $m(e_F) = 1$, then $m = m_F$.

A PCS is well-formed iff it is properly terminating and residue-free.

Clearly, both the property of proper termination as well as the property "residue free" are desirable properties of PCSs. Consequently we restrict our considerations to well-formed PCSs.

Let $C$ be a cycle of a PCS and $v$ a node of $C$. The node $v$ is an exit of $C$ iff there exists a path from $v$ to HALT, edge-disjoint with $C$. A PCS has the single-exit property iff it contains no cycle with more than one exit. It has been shown ([KOS], [IND]) that a flowchart with a single START and a single HALT is structurable by D-charts iff it has the single-exit property. Hence we restrict our considerations to well-formed PCSs which have the single-exit property.

Consider now the PCS shown in Figure 6. It can be shown that this PCS is not $\Delta$-structurable:

We say that a JOIN $J$ belongs to a FORK $F$ iff there exist two edge-disjoint paths from $F$ to $J$. A PCS is bridge-free iff it contains no FORK $F$ such that more than one JOIN belongs to $F$. The PCS of Figure 6 is not bridge-free, since both $J_1$ and $J_2$ belong to $F_1$. 
We are now in a position to formulate our

MAIN THEOREM. Let $S$ be a well-formed PCS which has the single-exit property and is bridge-free. Then $S$ is $\Delta$-structurable.
5. AUXILIARY DEFINITIONS AND LEMMATA

In this section we list various auxiliary definitions and lemmata to be used in the following section, where a proof of the main theorem is given. The lemmata of this section will be proven elsewhere [GI-YO] (cf. also [HE-YO]).

**Lemma 5.1** Any exit of a cycle of a well-formed PCS is a DECIDER.

**Lemma 5.2** Let $F$ be a FORK of a well-formed PCS and $P$ a path from $F$ to HALT. Then there exists a JOIN on $P$ which belongs to $F$.

**Lemma 5.3** Let $F$ be a FORK and $U$ a UNION of a well-formed PCS, and assume there exist two edge-disjoint paths $P_1$ and $P_2$ from $F$ to $U$. Then there exists a JOIN on $P_1$ or $P_2$ which belongs to $F$.

**Lemma 5.4** Let $D$ be a DECIDER and $J$ a JOIN of a well-formed PCS, and assume there exist two edge-disjoint paths $P_1$ and $P_2$ from $D$ to $J$. Then there exists a FORK $F_1$ on $P_1$ and a FORK $F_2$ on $P_2$, such that $J$ belongs to both $F_1$ and $F_2$. Furthermore, there exists a UNION $U_1$ on $P_1$ between $F_1$ and $J$, as well as a UNION $U_2$ on $P_2$ between $F_2$ and $J$, together with paths from $F_1$ to $U_2$ and from $F_2$ to $U_1$.

A Control Node of a PCS is any node of type DECIDER, UNION, FORK, JOIN.

The Beginning Node of a PCS is the first control node, encountered on the path leaving the START node.

**Lemma 5.5** The beginning node of a well-formed PCS cannot be a JOIN.

The Ending Node of a PCS is the last control node on the path leading to the HALT node.
Lemma 5.6  The ending node of a well-formed PCS cannot be a FORK.

Lemma 5.7  Let $S$ be a well-formed PCS, having a DECIDER $D$ as its ending node. Let $J$ be a JOIN of $S$. Assume there exists a path $P_1$ from START to $J$ which does not contain $D$. Then there also exists a path $P_2$ from START to $J$ which contains the inedge of $J$ not in $P_1$, and does not pass through $D$.

Lemma 5.8  Let $S$ be a well-formed PCS, $C$ a cycle of $S$, $D$ an exit of $C$, and $J$ a JOIN on $C$. Then there exists a cycle $C'$ which contains $D$ and the inedge of $J$ which is not on $C$.

Let $S$ be a PCS and $S'$ the subsystem of $S$ obtained by removing from $S$ the START node together with its outedge, as well as the HALT node, together with its inedge. We shall refer to $S'$ as a partial PCS and to $S$ as the extension of $S'$.

We call any PCS $S$ which satisfies the conditions of our theorem i.e. $S$ is well-formed, bridge-free and has the single-exit property, a Dijkstra-type Parallel Schema (DPS).

6. PROOF OF MAIN THEOREM

We can now reformulate our main theorem as follows:

**MAIN THEOREM.** Any DPS is $\Delta$-structurable.

We prove the Main Theorem by induction on the number $n$ of control nodes.

For $n = 2$ the Theorem is obvious.
We now assume that \( n > 2 \) and that the Theorem holds for any number \( n' < n \) of control nodes.

We distinguish the following five cases:

Case A. The beginning node is a DECIDER.

Case B. The ending node is a DECIDER.

Case C. The beginning node is a UNION.

Case D. The beginning node is a FORK and the ending node is a JOIN.

Case E. The beginning node is a FORK and the ending node is a UNION.

In view of Lemmata 5.5 and 5.6, Cases A-E cover all possibilities. Case B evidently overlaps with other cases.

Case A. Let \( P \) be a DPS with a DECIDER \( D \) as beginning node. Let \( e_A \) and \( e_B \) be the outedges of \( D \). Let \( A \) be the set of all nodes of \( G(P) \) reachable from \( e_A \), and \( B \) the set of all nodes of \( G(P) \) reachable from \( e_B \).

Assume \( e = (v_B, v_A) \) is an edge of \( G(P) \), such that \( v_A \in A \) and \( v_B \in B-A \). We shall refer to \( v_A \) as an entry node of \( A \). In view of Lemma 5.4, \( v_A \) must be a UNION.

Let \( G_A \) be the subgraph of \( G(P) \) spanned by the node set \( (\{D\} \cup A) - \{\text{HALT}\} \).

Let \( G_B \) be the subgraph of \( G(P) \) spanned by the node set \( (\{D\} \cup B) - \{\text{HALT}\} \).

It is easily verified that the DPS \( P \) is equivalent to the DPS \( P' \) shown in Figure 7, where \( P' \) satisfies the following conditions:

1. \( P_A \) and \( P_B \) are partial DPSs.
(2) $G(P_A)$ is isomorphic to the graph obtained from the subgraph $G_A$ by "shortcircuiting" all entry nodes of $A$.

(3) $G(P_B)$ is isomorphic to the graph obtained from the subgraph $G_B$ by "shortcircuiting" all entry nodes of $B$.

Clearly the extensions of $P_A$ and $P_B$ have fewer nodes than $P$ hence the induction hypothesis applies. It follows that $P$ is $\Delta$-structurable.

---

**Figure 7:** DPS $P'$ of Case A.
Example 1. Consider the non-structured DPS $P$ of Figure 8. This DPS is an example of Case A, i.e. the beginning node is a DECIDER.

We obtain

$A = \{s_2, F_1, s_4, s_5, U_1, U_2, s_8, s_9, J, s_{10}, H\}$

$B = \{s_3, F_2, s_6, s_7, U_1, U_2, s_8, s_9, J, s_{10}, H\}$.

The entry nodes of $A$ as well as of $B$ are both $U_1$ and $U_2$.

---

**Figure 8:** A non-structured DPS $P$ beginning with DECIDER.
The $\Delta$-structured version of $P$ is shown in Figure 9.

Figure 9: A structured version of $P$ of Figure 8.
Case B. Let $P$ be a DPS with a DECIDER $D$ as ending node. Let $e$ be the outedge of $D$ not terminating in the HALT node. We denote by $A$ the set of nodes of $G(P)$ reachable from the START node without passing the DECIDER $D$. Let $B$ be the set of all nodes belonging to some cycle containing $D$, but excluding $D$.

In view of Lemmata 5.7 and 5.8, $A$ and $B$ cannot contain JOIN nodes with a single inedge.

Let $G_A$ be the subgraph of $G(P)$ spanned by $A$, and $G_B$ the subgraph of $G(P)$ spanned by $B$.

It is now easily seen that the DPS $P$ is equivalent to the DPS $S'$ shown in Figure 10, where $P'$ satisfies the following conditions:

1. $P_A$ and $P_B$ are partial DPSs.
2. $G(P_A)$ is isomorphic to $G_A$ with all single-entry UNIONs short-circuited.
3. $G(P_B)$ is isomorphic to $G_B$ with all single-entry UNIONs short-circuited.

The extensions of $P_A$ and $P_B$ again have fewer nodes than $P$. In view of the induction hypothesis, it follows that $P$ is $\Delta$-structurable.

Figure 10: DPS $P'$ of Case B.
Example 2. An example of a non-structured DPS, illustrating Case B, is shown in Figure 11. We have

\[ A = \{0_1, F_1, 0_2, 0_3, U_1, U_2, 0_4, 0_5, J, 0_6\} \]

\[ B = \{0_8, F_2, 0_9, 0_{10}, U_1, U_2, 0_4, 0_5, J, 0_6\} \]

Figure 11: A non-structured DPS P ending with DECIDER.
The A-structured version of P is shown in Figure 12.

Figure 12: A structured version of P of Figure 11.
Case C. Let $P$ be a DPS with a UNION $U$ as beginning node.

Evidently $U$ must be on some cycle $C$. Let $v$ be an exit of $C$. Since $P$ is a DPS, the node $v$ is the only exit of $C$. By Lemma 5.1 the node $v$ must be a DECIDER.

One easily sees that there exists a cycle $C'$ which contains $U$, and has a single exit, which is a DECIDER $D$, having an outedge $e$ from which $C'$ is not reachable. Thus the removal of edge $e$ from $G(P)$ splits $G(P)$ into two node-disjoint subgraphs $G_S$ and $G_H$, where $G_S$ contains the START and $G_H$ the HALT node of $G(P)$. If $G_H$ contains only the HALT node, this case reduces to Case B. Otherwise, $P$ must have the structure shown in Figure 13, where $P_S$ and $P_H$ satisfy the following conditions:

1. $P_S$ and $P_H$ are partial DPSs, the extensions of which have fewer nodes than $P$.
2. $G(P_S)$ is isomorphic to $G_S$ less START node and its outedge.
3. $G(P_H)$ is isomorphic to $G_H$ less HALT node and its inedge.

In view of our induction hypothesis, $P$ is $\Delta$-structurable.

Figure 13: DPS $P$ of Case C.
Case D. Let $P$ be a DPS with a FORK $F$ as beginning node and a JOIN $J$ as ending node. We distinguish between two cases.

Subcase D-1: $J$ belongs to $F$.

Let $e_A$ and $e_B$ be the out edges of $F$. Let $A$ and $B$ be the sets of nodes of $G(P)$ excluding $J$ and HALT, reachable from $e_A$ and $e_B$, respectively. Let $e$ be an edge $(v_A, v_B)$, where $v_A \in A$ and $v_B \in B$. If $v_B$ is a JOIN $J'$, then $J'$ belongs to $F$, i.e. $P$ is not bridge-free, contradicting our assumption. If $v_B$ is a UNION, the assumptions of Lemma 5.3 are satisfied. Hence there again exists a JOIN $J' \neq J$ which belongs to $F$, leading again to a contradiction. Thus no such edge $e$ can exist. Due to symmetry there exists no edge from $B$ to $A$. It follows that $P$ has the structure shown in Figure 14. In view of the induction hypothesis, $P$ is $\Delta$-structurable.

Subcase D-2. $J$ does not belong to $F$.

By Lemma 5.2 there exists a JOIN $J'$ which belongs to $F$. Let $A$ be defined as follows:

$A = \{v \mid v$ is a node of $G(P)$ and there exists a path from $v$ to $J'\} \cup \{J'\}$.

Let $B$ be the set of the remaining nodes of $G(P)$. Note that $J \in B$.

Clearly no edge exists from $B$ to $A$. Now let $e = (v_A, v_B)$ be an edge from $A$ to $B$.

Assume $v_A \neq J'$.

There exists a path $p$ from $F$ to $v_A$. Assume that $p$ does not pass through $J$. By Lemma 5.2, the path $(p, e)$ has an extension passing
through $J'$. This contradicts our assumption that $v_B \in B$. Hence $p$
must pass through $J'$, Thus there exists a cycle $C$ containing $v_A$ and
$J'$, where $v_A$ is an exit of $C$. Hence, by Lemma 5.1, $v_A$ is a DECIDER.

It follows that either $v_A = J'$ or $v_A$ is a DECIDER. It is now easily seen that exactly one edge exists from $A$ to $B$. Namely, the existence of two or more edges between $A$ and $B$ will imply two or more exits from the cycle $C$. On the other hand, clearly at least one such edge exists.

It follows that $P$ has the structure shown in Figure 15, where $P_A$
and $P_B$ are defined in the evident way. Arguments similar to the previous
ones show that $P$ is $\Delta$-structurable.

Note that the assumption that $P$ of Figure 15 is a DPS immediately yields the result that both $P_A$ and $P_B$ are partial DPSs.

Case E. Let $P$ be a DPS with a FORK $F$ as beginning node and a UNION $U$ as ending node.

By Lemma 5.2, $P$ contains a JOIN $J$ which belongs to $F$. Thus the arguments used in Subcase D-2 become applicable, i.e. $P$ is $\Delta$-structurable.
7. OPEN PROBLEMS

In this section we indicate various open problems, related to the main result of this paper.

A) Let us say that a PCS $S$ is contention-free iff the following condition is satisfied:

No pair of operational nodes labeled by the same letter $\sigma \in \Sigma$ can ever become marked simultaneously. Evidently, PCSs which contain contentions require arbitration networks in their implementation.

We claim that our structuring algorithm does not introduce contention, in the evident sense.

B) We may define the "degree of parallelism" of a PCS $S$ as the maximal number of tokens which may occur simultaneously, under any marking reachable from the initial marking. We claim that our structuring algorithm preserves the degree of parallelism. This is evidently not true for the algorithm of [CO-LE].

C) We intend to reformulate our structuring algorithm in such a way that it becomes applicable to arbitrary PCSs. If the PCS is a DPS, the algorithm is to yield its structured version. Otherwise, the algorithm is to yield the result that the given PCS is not a DPS.

D) Further research is required in order to establish a suitable converse of our Main Theorem. This necessitates the formulation of a weaker type of equivalence. For example, activity sequences $pq\sigma$ and $qp\sigma$ should be considered interchangeable. A suitable converse theorem would generalize available results on the structuring of (sequential) flowcharts ([KOS], [IND]).
E) Further research is intended as to suitable extensions of the set $\Delta$ of primitive PCGs introduced in this paper. Such extensions will, of course, broaden the class of structurable PCSs beyond the class of DPSs studied in this paper. Considerable efforts have been devoted so far to the corresponding aspects of sequential flowcharts ([MIL], [LE-MA], [ELG]).
REFERENCES


REFERENCES (cont'd)


