REDUCIBILITY OF PARALLEL CONTROL STRUCTURES

by

Irit Bodo* and Michael Yoeli**

Technical Report #283
June 1983

* This paper is part of the first author's M.Sc. Thesis, Dept. of Computer Science, Technion.

** This research was supported by the Fund for the Promotion of Research at the Technion, under grant 121-589.
ABSTRACT

This paper is concerned with parallel control structures containing (2-way) forks, (2-way) joins and decision nodes as control nodes. A parallel control structure is reducible iff it can be converted into an equivalent structured version, obtainable from a set of simpler ("primitive") structures. A control structure is "purely parallel", if it contains forks and joins, but no decision nodes. It is "purely sequential" if it is without forks and joins. It is "pure", if it is either purely parallel or purely sequential. This paper investigates the reducibility of parallel control structures with respect to sets of pure control primitives.
1. INTRODUCTION

Structured programming has become an important methodology for the design of correct, easily understood computer programs [DA-DI-HO]. The arguments in favor of a structured approach to sequential programming evidently also apply to parallel programming. Consequently, efforts have been devoted to various aspects of structured parallel programming [CO-LE], [YOE], [BA-YO-ST], [GI-YO].

An important aspect of structured programming is the appropriate selection of control primitives. An essential property of control primitives is their irreducibility (see [LE-MA]).

The paper [GI-YO] is concerned with the irreducibility of "purely parallel" control structures, i.e. parallel control structures (cf. [YO-GI]), containing (2-way) forks, (2-way) joins, but not decision nodes. The irreducibility of conventional, "purely sequential" control structures has been extensively studied in the literature (cf. [LE-MA]).

In this paper we study the irreducibility of "mixed" control structures, i.e. parallel control structures, containing forks and joins as well as decision nodes. In particular, we investigate their irreducibility with respect to sets of "pure" control primitives (a control structure is "pure", if it is either "purely parallel" or "purely sequential").
2. PARALLEL CONTROL GRAPHS

In this section we introduce the basic concept of parallel control graph (PCG). This section is a modified version of Section 2 of [YO-GI].

Definition 2.1 A parallel control graph (PCG) is a finite, directed graph $G$ with the following properties.

1. Each node of $G$ is of one of the seven types shown in Fig. 2.1.
2. Multiple edges are not admitted.
3. $G$ has exactly one START node $S$ and exactly one HALT node $H$.
4. There exists a directed path from $S$ to every other node $v$ of $G$.
5. There exists a directed path from every node $v \neq H$ of $G$ to the node $H$.

Evidently a PCG cannot have self-loops (i.e. cycles of length 1). Examples of PCGs are shown in Fig.2.2.

We shall refer to nodes of type FORK, JOIN, DECIDER, and UNION as control nodes. A PCG with DECIDER and UNION nodes as only control nodes is purely sequential. Similarly, a PCG with FORK and JOIN nodes as only control nodes is purely parallel.

Definition 2.2 Let $G$ be a PCG. A marking $m$ of $G$ is a function $m: E \to \omega$, where $E$ is the edge set of $G$ and $\omega$ is the set of non-negative integers. A marked PCG is an ordered pair $(G,m)$, where $G$ is a PCG and $m$ is a marking of $G$. 

<table>
<thead>
<tr>
<th>NODE TYPE</th>
<th>INDEGREE</th>
<th>OUTDEGREE</th>
<th>GRAPHICAL REPRESENTATION</th>
</tr>
</thead>
<tbody>
<tr>
<td>START</td>
<td>0</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>HALT</td>
<td>1</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>FORK</td>
<td>1</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>JOIN</td>
<td>2</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>DECIDER</td>
<td>1</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>UNION</td>
<td>2</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>OPERATION</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
</tbody>
</table>

Figure 2.1 - Node types of parallel control graph.
Figure 2.2 - Examples of PGCs.
Let $e$ be an edge of the marked PCG $(G, m)$. We refer to $m(e)$ as the number of tokens on $e$. If $m(e) > 0$, we say that $e$ is marked. In the graphical representation of marked PCGs, tokens are indicated by dots (●). Fig. 2.3 shows examples of marked PCGs.

Figure 2.3 - Examples of marked PCGs.
Definition 2.3 Let \((G, m)\) be a marked PCG. A node of type OPERATION or DECIDER or FORK is enabled iff its inedge is marked. A JOIN node is enabled iff both its inedges are marked. A UNION node is enabled iff at least one of its inedges is marked. A node which is enabled may fire.

The firing rules, illustrated in Fig. 2.4, are as follows:

Definition 2.4

a) The firing of a FORK node decreases the marking of its inedge by 1 and increases the marking of both its outedges by 1.

b) The firing of a JOIN node decreases the markings of both its inedges by 1, and increases the marking of its outedge by 1.

c) The firing of a DECIDER node decreases the marking of its inedge by 1, and increases the marking of either one of its outedges by 1.

d) The firing of a UNION node decreases the marking of one of its marked inedges by 1, and increases the marking of its outedge by 1.

e) The firing of an OPERATION node decreases the marking of its inedge by 1 and increases the marking of its outedge by 1.

For example, node \(J\) in Fig. 2.3(a) is enabled. The firing of \(J\) yields the marked PCG of Fig. 2.3(b).

Marked PCGs can, of course, also be defined in terms of Petri nets (cf. [YOE]).
<table>
<thead>
<tr>
<th>NODE TYPE</th>
<th>BEFORE FIRING</th>
<th>AFTER FIRING</th>
</tr>
</thead>
<tbody>
<tr>
<td>FORK (F)</td>
<td><img src="image1" alt="Diagram" /></td>
<td><img src="image2" alt="Diagram" /></td>
</tr>
<tr>
<td>JOIN (J)</td>
<td><img src="image3" alt="Diagram" /></td>
<td><img src="image4" alt="Diagram" /></td>
</tr>
<tr>
<td>DECIDER (D)</td>
<td><img src="image5" alt="Diagram" /></td>
<td><img src="image6" alt="Diagram" /></td>
</tr>
<tr>
<td>UNION (U)</td>
<td><img src="image7" alt="Diagram" /></td>
<td><img src="image8" alt="Diagram" /></td>
</tr>
<tr>
<td>OPERATION (OP)</td>
<td><img src="image9" alt="Diagram" /></td>
<td><img src="image10" alt="Diagram" /></td>
</tr>
</tbody>
</table>

Figure 2.4 - Examples of "firings"
3. WELL-FORMED PARALLEL CONTROL GRAPHS

In this section we define well-formed PCGs, (similarly to Section 3 of [YO-GI]). Let \( m \) and \( m' \) be markings of the PCG \( G \).

We write \( m \vdash m' \) 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 \( G \). Furthermore, we set

\[
[m] = \{m' | m \vdash m'\} \cup \{m\}.
\]

We shall refer to \([m]\) as the set of all markings reachable from \( m \).

We denote by \( e_S \) the outedge of the START node \( S \), and by \( e_H \) the inedge of the HALT node \( H \).

**Definition 3.1** The initial marking \( m_0 \) of a PCG \( G \) is defined as follows:

\[
m_0(e_S) = 1 \text{ and } m_0(e) = 0 \text{ for every } e \neq e_S.
\]

A marking \( m \) of \( G \) is final iff \( m(e_H) > 0 \). We denote by \( M_F \) the set of all final markings of \( G \).

Let \( G \) be the PCG shown in Fig. 2.3, \( m_a \) its marking shown in Fig. 2.3(a) and \( m_b \) the marking shown in Fig. 2.3(b). Then \( m_a \in [m_0] \), \( m_b \in [m_0] \), and \( m_b \in M_F \).

**Definition 3.2** A PCG \( G \) is terminating iff \( (\forall m \in [m_0]) ([m] \cap M_F \neq \emptyset) \) i.e. if \( m \) is reachable from \( m_0 \), then there exists a final marking reachable from \( m \).

By deadlock we mean a marking \( m \) such that \([m] \cap M_F = \emptyset\), i.e. no final marking is reachable from \( m \). Thus, \( G \) is terminating iff no deadlock is reachable from \( m_0 \).
One easily verifies that the PCGs of Figs. 2.2(a), 2.2(b) and 2.3 are terminating, whereas the graph of Fig. 2.2(c) is not terminating.

**Definition 3.3** Let \( G \) be a PCG and \( E \) its edge set, \( G \) is residue-free iff

\[
(\forall m \in [m_0]) \left[ \sum_{e \in E} m(e) = 1 \right],
\]

i.e. for any final marking \( m \) reachable from \( m_0 \), the marked PCG \((G,m)\) contains exactly one token (namely on \( e_H \)).

**Definition 3.4** A PCG \( G \) with edge set \( E \) is safe iff

\[
(\forall m \in [m_0])(\forall e \in E)m(e) \leq 1,
\]

i.e. the number of tokens on any edge \( e \) cannot exceed 1, under any marking \( m \) reachable from \( m_0 \).

The following proposition is an immediate consequence of Theorem 3.1 of [YO-GI].

**Proposition 3.1** Every well-formed PCG is safe.

**Proposition 3.2** If a well-formed PCG \( G \) contains a JOIN node, then \( G \) also contains a FORK node.

**Proof:** Assume \( G \) is well-formed and contains a JOIN node \( J \), but no FORK node. There exists a directed path \( P \) from the START node \( S \) to \( J \). Without loss of generality, assume \( P \) contains no other JOIN node between \( S \) and \( J \). Hence there exists a marking \( m \in [m_0] \), such that the inedge \( e \) of \( J \) on \( P \) is marked under \( m \).
Since G contains no FORK, the total number of tokens under m must be exactly one. It follows that J can never fire, contradicting our assumption that G is well-formed.
4. PARALLEL CONTROL STRUCTURES

A parallel control structure (PCS) is a suitably labeled PCG [YOE]. A synchronization structure, as defined in [GI-YO], is a special case of a PCS.

Definition 4.1 A parallel control structure (PCS) \( \Gamma \) consists of the following:

1. A PCG \( G(\Gamma) \)
2. A finite alphabet \( \Sigma \) of operation letters. Every OPERATION node of \( G(\Gamma) \) is labeled by a letter of \( \Sigma \).
3. A finite alphabet \( \Pi \) of predicate letters. Every DECIDER node \( D \) of \( G(\Gamma) \) is labeled by a letter of \( \Pi \). Furthermore, one outgoing edge of \( D \) is labeled \( T \) (true), and the other edge \( F \) (false).

An example of a PCS is shown in Fig. 4.1.

\[ \Sigma = \{a, b, c\} \]
\[ \Pi = \{p\} \]

Figure 4.1 - Example of PCS(\( \Gamma_1 \))
A PCS $\Gamma$ is well-formed iff $G(\Gamma)$ is well-formed.

**Definition 4.2** Let $G$ be a PCG. A node sequence

$$(v_1, v_2, \ldots, v_n)$$

is a firing sequence of $G$ iff there exist markings $(m_1, m_2, \ldots, m_n)$ of $G$ such that

$$m_{i-1} \xrightarrow{v_i} m_i \text{ for } 1 \leq i \leq n,$$

where $m_0$ is the initial marking of $G$ and $m_n$ is final (i.e., $m_n \in M_F$).

**Definition 4.3** Let $\Gamma$ be a PCS. We denote by $\overline{\Pi}$ the set of negated predicate letters, i.e.

$$\overline{\Pi} = \{\overline{p} | p \in \Pi\}.$$

Let $\alpha = (v_1, v_2, \ldots, v_n)$ be a firing sequence of $G(\Gamma)$ and $(m_1, m_2, \ldots, m_n)$ the corresponding sequence of markings. We associate with every $v_i$ in $\alpha$ a symbol $\tilde{v}_i$ in $\tilde{\Sigma} \cup \{\lambda\}$, where $\tilde{\Sigma} = \Sigma \cup \Pi \cup \overline{\Pi}$ and $\lambda$ denotes the empty sequence in accordance with the following rules:

(a) if $v_i$ is a FORK or a JOIN or a UNION, then $\tilde{v}_i = \lambda$.

(b) if $v_i$ is an OPERATION node, then $\tilde{v}_i = \sigma$, where $\sigma \in \Sigma$ is the label of $v_i$ in $\Gamma$.

(c) if $v_i$ is a DECIDER with label $p \in \Pi$, outedge $e_1$ labeled $T$ and outedge $e_2$ labeled $F$, then $\tilde{v}_i = p$ if $m_i(e_1) = m_{i-1}(e_1) + 1$, else $\tilde{v}_i = \overline{p}$.

We set $\tilde{\alpha} = \tilde{v}_1 \tilde{v}_2 \ldots \tilde{v}_n$. Thus $\overline{\varphi} \in (\tilde{\Sigma})^*$. 
Definition 4.4 Let $\Gamma$ be a PCS. With $\Gamma$ we associate the language $L(\Gamma) \subseteq (\Sigma)^*$ defined as follows:

$$L(\Gamma) = \{ \tilde{a} \mid \text{a is a firing sequence of } G(\Gamma) \}.$$ 

For example, for the PCS $\Gamma_1$ of Fig. 4.1 we have

$$L(\Gamma_1) = \{pab, pba, \tilde{p}c\}.$$ 

If $L(\Gamma) = L(\Gamma')$, $\Gamma$ and $\Gamma'$ are said to be $L$-equivalent.

Proposition 4.1 Let $\Gamma$ be a well-formed PCS. Then $L(\Gamma)$ is regular.

Proof This follows from Proposition 3.1, stating that every well-formed PCG is safe. Thus the set of markings reachable from the initial marking is finite. Hence, there exists a finite automaton $A$ such that $L(A) = L(\Gamma)$.

On the other hand, the language $L(\Gamma)$ of a PCS $\Gamma$ which is not well-formed, is not necessarily regular. For example, we have for the PCS $\Gamma_2$ of Fig. 4.2: $L(\Gamma_2) = \{ (pa)^n p (\phi)^m q \mid 0 \leq m \leq n \}$. This language is clearly not regular.
Figure 4.2 - A PCS \( T_2 \) whose language \( L(T_2) \) is not regular

Since the languages of well-formed PCSs are regular, it is decidable whether two well-formed PCSs are \( L \)-equivalent.
5. COMPOSITION OF PARALLEL CONTROL STRUCTURES

Structured programs are obtained by "successive composition", using a given set of basic ("primitive") control structures [DA-DI-HO], [LE-MA]. In the following definition we extend this concept of "composition" to PCGs (cf. [YOE]).

Definition 5.1 Let $G_1$ and $G_2$ be disjoint PCGs and $v$ an OPERATION node of $G_1$. We define the composition $G_1(v + G_2)$ to be the PCG $G$ obtained by substituting $G_2$ for $v$ in $G_1$, as indicated in Fig. 5.1.

For example, the PCG $G(T_1)$ of the PCS $T_1$ shown in Fig. 4.1, can be obtained as composition $G_1(v + G_2)$ where $G_1$ and $G_2$ are shown in Fig. 5.2.

![Figure 5.1 - Illustrating the concept of composition](image_url)
Proposition 5.1 Let $G_1$ and $G_2$ be disjoint PCGs, and $v$ an OPERATION node of $G_1$. Then their composition $G = G_1(v \cdot G_2)$ is well-formed iff $G_1$ and $G_2$ are well-formed.

Proof: If $G_1$ and $G_2$ are terminating and residue-free, then $G$ is clearly also terminating and residue-free. If $G_1$ is not well-formed, evidently $G$ is not well-formed. Assume now that $G_2$ is not terminating. In view of Lemma 3.2 of [YO-GI], there exists a marking of $G$, reachable from its initial marking, such that the start edge of $G_2$ in $G$ is marked under $m$. It follows that $G$ is either not terminating or not residue-free.

Similarly, if $G_2$ is not residue-free, it follows that $G$ is not residue-free.

The concept of "reducibility" plays an important role in the theory of structured programming (cf. [LE-MA]).
Definition 5.2 Let $\Delta$ be a set of well-formed PCG's, $\Delta = \{G_1, G_2, \ldots\}$, and $\Gamma$ a PCS. $\Gamma$ is reducible with respect to $\Delta$ iff there exists a PCS $\Gamma'$, such that

1. $L(\Gamma') = L(\Gamma)$
2. $G(\Gamma')$ can be obtained by successive compositions of PCGs in $\Delta$.

Fig. 5.3 shows well-known primitive "D-structures" (D for Dijkstra, see [LE-MA]). Our next proposition proves that they are sufficient for the composition of every well-formed PCS without cycles and without fork nodes.

Proposition 5.2 Let $\Gamma$ be a well-formed, cycle-free, purely sequential PCS. Then $\Gamma$ is reducible w.r.t. $\{D_0, D_1, D_2\}$, where the $D_i$'s are shown in Fig. 5.3.

Figure 5.3 - Primitive, cycle-free D-structures

Proof: If $G(\Gamma)$ contains no control node, the proposition is trivial.
We now assume that \( G(\Gamma) \) contains at least one control node. Since \( G(\Gamma) \) is cycle-free, the first control node of \( G(\Gamma) \) cannot be a UNION. Thus, the first control node of \( G(\Gamma) \) is a DECIDER.

We now use induction on the number \( k \) of DECIDER nodes in \( G(\Gamma) \). If \( k = 1 \), then \( G(\Gamma) \) must be as shown in Fig. 5.4.

![Diagram showing the structure of \( G(\Gamma) \) with at least one control node](image)

Figure 5.4

In this case, the proposition evidently holds. Assume now that the proposition holds for PCSs with \( i \leq k \) DECIDER nodes and that \( \Gamma \) contains \( k + 1 \) DECIDER nodes.

One easily verifies that \( \Gamma \) is equivalent to a PCS \( \Gamma' \), where \( G(\Gamma') \) is shown in Fig. 5.5.

The subgraphs \( \hat{G}_1 \) and \( \hat{G}_2 \) of Fig. 5.5 contain \( i \leq k \) DECIDER nodes.
represents a (possibly empty) sequence of OPERATION nodes.

Figure 5.5

Hence the induction hypothesis is applicable to $\hat{G}_1$ and $\hat{G}_2$, and the proposition follows.

Fig. 5.6 illustrates the application of Proposition 5.2.
Figure 5.6 - (a) a PCS \( \Gamma \)
(b) a structured version of \( \Gamma \).
Böhm and Jacopini [BO-JA] introduced additional primitive (purely sequential) control structures $\Omega_k, k \geq 1$ (see [LE-MA]). We prove their "irreducibility", in the following sense.

**Proposition 5.3** Let $\Omega_k, k \geq 1$, be the well-formed PCSs shown in Fig. 5.7. Then $\Omega_k, k > 1$, is not reducible w.r.t. $\Delta = \{D_0, D_1, D_2\} \cup \{G(\alpha_i) \mid 1 \leq i \leq k\}$.

**Proof.** Let $L = L(\Omega_k)$. Then $L \subseteq \tilde{\Sigma}^*$, where $\tilde{\Sigma} = \Sigma \cup \Pi \cup \Pi^\dagger$

$\Sigma = \{a_1, \ldots, a_k\}$, $\Pi = \{p_1, \ldots, p_k\}$. We define $\eta: \Pi \cup \Pi^\dagger \cup \{\lambda\}$ as follows:

$$\begin{align*}
(\forall \sigma \in \Sigma) \eta(\sigma) &= \lambda \\
(\forall p \in \Pi) \eta(p) &= p \\
(\forall \tilde{p} \in \Pi^\dagger) \eta(\tilde{p}) &= \tilde{p}.
\end{align*}$$

We set $\eta(L) = \{\eta(w) \mid w \in L\}$. Then $\eta(L)$ can be represented by the regular expression

$$E = (p_1 p_2 \ldots p_k)^* (\tilde{p}_1 + p_1 \tilde{p}_2 + \ldots + p_1 p_2 \ldots \tilde{p}_{k-1} p_k).$$

Assume now that $\Omega_k$ is reducible with respect to $\Delta$. Thus, there exists a PCS $\Gamma$, such that $L(\Gamma) = L(\Omega_k) = L$, and $G(\Gamma)$ can be obtained by a composition sequence

$$G_0, G_1, \ldots, G_i, G_{i+1}, \ldots, G_r = G(\Gamma),$$

where $G_0 \in \Delta$, $G_{i+1} = G_i(\nu + G_i^r)$ and $G_i^r \in \Delta$, for $0 \leq i \leq r-1$.

Clearly, there exists a nonnegative integer $h$, such that $G_h^r \in \Delta - \{D_0, D_1\}$, whereas every $G_i^r$ for $i > h$ is either $D_0$ or $D_1$.  

---

Technion - Computer Science Department - Technical Report CS0283 - 1983
Consider now the case $G_h' = D_2$.

Let $p_h$ be the label in $\Gamma$ of the DECIDER node of $G_h'$. Then $x p_h y \in \eta(L)$ implies $x\check{p}_h y \in \eta(L)$. Now, every word in $\eta(L)$ contains exactly one negative predicate (see Fig. 5.7(b)). Evidently, this condition cannot be satisfied by both $x p_h y$ as well as $x\check{p}_h y$.

It follows that $G_h' \in \{G(\Omega_i) | 1 \leq i \leq k\}$. Say, $G_h' = G(\Omega_j)$, where $j < k$. Then $\eta(L(\Gamma))$ must contain a word $w = x(q_1 q_2 \ldots q_j)(q_1 q_2 \ldots q_j)y$, where $\{q_1, q_2, \ldots, q_j\} \subseteq \Pi \cup \check{\Pi}$. Since no word in $\eta(L)$ contains more than one negated predicate, we must have $\{q_1, q_2, \ldots, q_j\} \subseteq \Pi$.

However, in view of the form of $E$, no word in $\eta(L)$ can contain a substring $q_1 q_2 \ldots q_j$ for $j < k$. Thus, $L(\Gamma) \neq L$. 
Consequently, our assumption about the reducibility of \( \Omega_k \) with respect to \( \Delta \) cannot hold.

An alternative proof of Proposition 5.3, for the case \( k = 2 \), is due to Knuth and Floyd [KN-FL].
6. THE IRREDUCIBILITY OF "MIXED" PARALLEL CONTROL STRUCTURES

In [YOE] the following conjecture was formulated.

Conjecture Let $\Delta_s$ be a set of PCGs such that every well-formed, purely sequential PCS is reducible w.r.t. $\Delta_s$. Similarly, let $\Delta_p$ be a set of PCGs such that every well-formed, purely parallel PCS is reducible w.r.t. $\Delta_p$. Then every well-formed PCS is reducible w.r.t. $\Delta_s \cup \Delta_p$.

In this section we show that the above conjecture does not hold.

Theorem 6.1 Let $T^{(8)}$ be the PCS shown in Fig. 6.1. Let $\Delta_p$ be an arbitrary set of well-formed purely parallel PCGs and $\Delta_s$ an arbitrary set of well-formed purely sequential PCGs. Then $T^{(8)}$ is irreducible w.r.t. $\Delta = \Delta_p \cup \Delta_s$.

![Diagram of PCS $T^{(8)}$](image)

Figure 6.1 - PCS $T^{(8)}$.
For the proof of this theorem we need the following lemmata.

**Lemma 6.1** Let $\Gamma$ and $\Gamma'$ be well-formed $L$-equivalent PCSs. If $\Gamma$ is cycle-free, then so is $\Gamma'$.

**Proof:** Let $\Gamma$ be cycle-free. Then $L(\Gamma)$ is finite. Assume $\Gamma'$ contains a cycle $C$. Then $C$ contains a DECIDER $D$, by Prop. 4.2 of [YO-GI]. For every $n \geq 1$, there exists a firing sequence of $G(\Gamma)$ in which $D$ is enabled $n$ times. It follows that $L(\Gamma')$ is infinite. Hence $\Gamma'$ is cycle-free.

**Lemma 6.2** Let $G$ be a well-formed, cycle-free PCG, and $v$ any node of $G$. Assume there exists a directed path $P$ from the START node $S$ to $v$, not containing a DECIDER as intermediate node.

Then every firing sequence of $G$ contains the node $v$ exactly once.

**Proof:** The token-count of $P$ (i.e. the number of tokens on $P$) under the initial marking $m_0$ is 1. This token-count cannot decrease by the firing of any intermediate node of $P$, since only a DECIDER node could remove a token from $P$. Since $G$ is residue-free, any firing sequence of $G$ must contain $v$.

Assume now that some firing sequence of $G$ contains $v$ twice. Since $G$ is cycle-free, the marking enabling $v$ for the second time must include at least two tokens, one on the inedge $e$ of $v$, and the other on a directed path between $e$ and the HALT node $H$. In view of Lemma 5.1 of [YO-GI], $G$ must contain a cycle, in contradiction to our assumption.
Lemma 6.3  Let G be a well-formed, purely parallel PCG. Then every firing sequence of G contains every internal node (i.e. every node except START and HALT) exactly once.

Proof: By Proposition 4.2 of [YO-GI], every cycle of a well-formed PCG contains a DECIDER node. Hence G is cycle-free, and Lemma 6.2 is applicable to every internal node. □

(This lemma coincides with Proposition 4.1 of [GI-YO].)

Lemma 6.4  Let Γ be a well-formed, cycle-free PCS, containing a single DECIDER node. Let Γ' be a well-formed PCS, L-equivalent to Γ. Then Γ' also contains a single DECIDER node.

Proof: Evidently, Γ' contains at least one DECIDER node. Assume now that G(Γ') contains two DECIDER nodes D₁ and D₂. If there exist directed paths P₁ and P₂, from the START node S to D₁ and D₂, respectively, not containing any DECIDER node as intermediate node, then, by Lemma 6.2, every firing sequence of G contains both D₁ and D₂.

If no such paths exist, then there exists a directed path from S containing at least two DECIDER nodes. In both cases there exists a firing sequence of G, containing two DECIDER nodes. It follows that L(Γ') contains a word with two predicate letters. However, in view of Lemma 6.2, L(Γ) cannot contain such a word. Since L(Γ) = L(Γ'), G(Γ') cannot contain more than one DECIDER node. □
Let G be a well-formed, cycle-free PCG. We define a binary
relation \( \preceq \) on the set \( \text{OP}(G) \) of OPERATION nodes of G as follows:

\[ a \preceq b \text{ iff in every firing sequence of } G \text{ containing both } \]
\[ a \text{ and } b, \text{ a precedes } b. \]

(Notice that since G is cycle-free, any firing sequence cannot
contain an OPERATION node twice.)

**Lemma 6.5** Let \( G_1 \) and \( G_2 \) be well-formed, cycle-free PCG's
and let \( G = G_1(v + G_2) \). Then G satisfies the following conditions:

(a) If for some \( v_1 \in \text{OP}(G_1) \), there exists \( v_2 \in \text{OP}(G_2) \), such that
\[ v_1 \preceq v_2, \text{ then } (\forall v_2 \in \text{OP}(G_2))(v_1 \preceq v_2). \]

(b) Same as (a), but \( \preceq \) replaced by \( \prec \).

**Proof:**

(a) One easily verifies, that under our assumptions, the firing
of \( v_1 \) must always precede the firing of the first control node in
the \( G_2 \)-subgraph of G. Therefore, we must have \( v_1 \ preprocess v \) in \( G_1 \).
Consequently, \( v_1 \preceq v_2 \) holds in G for every \( v_2 \in \text{OP}(G_2) \).

(b) Same as (a).

**Proof of Theorem 6.1** Assume \( \Gamma^{(8)} \) is reducible with respect
to \( \Delta \) i.e. there exists a well-formed PCS \( \Gamma' \), such that

(1) \( L(\Gamma') = L(\Gamma) \), and

(2) \( G(\Gamma') \) can be obtained by successive compositions of PCG's in \( \Delta \).

In view of Lemmata 6.1 and 6.4, \( \Gamma' \) is cycle-free, and contains
a single DECIDER node. Therefore, only one component of \( \Gamma' \) may
contain a DECIDER node. This component, say C, must evidently be
of the form shown in Fig. 5.4.
Hence \( G(\Gamma') \) can be obtained as a composition \( G(\Gamma') = G_1(v_1 \circ C') \), where \( C' \) is derived from \( C \) by successive compositions using PCGs in \( \Delta \) and \( G_1 \) is a well-formed, purely parallel PCG or \( G(\Gamma') \) is \( C' \).

**Remark** Since no word in \( L(\Gamma^{(8)}) \) contains any letter twice, and in view of Lemma 6.3, applied to \( G_1 \), no two OPERATION nodes of \( \Gamma' \) outside \( C' \) can have the same label. This holds, similarly, for two OPERATION nodes, one inside \( C' \) and the other one outside \( C' \).

In \( L = L(\Gamma^{(8)}) \), \( p \) always precedes the letters \( b, c \) and \( d \), whereas \( \tilde{p} \) never precedes these letters. It follows that all OPERATION nodes labeled \( b, c \) and \( d \) in \( \Gamma' \) must be nodes of \( C' \).

In the sequel, we identify the OPERATION nodes of \( G(\Gamma^{(8)}) \), with their labels in \( \Gamma^{(8)} \).

We have \( c \in e \) in \( G(\Gamma^{(8)}) \), but \( d \in e \) does not hold; hence by Lemma 6.5 and the above Remark, no occurrence of \( e \) can be outside \( C' \) in \( \Gamma' \).

Similarly, \( d \in f \), but \( c \in f \) does not hold, hence \( f \) cannot appear outside \( C' \) in \( \Gamma' \). Thus, both \( e \) and \( f \) are inside \( C' \) in \( \Gamma' \). Furthermore, \( e \in g \), but \( f \in g \) does not hold, hence also \( g \) must be inside \( C' \) in \( \Gamma' \). Now, \( a \in g \), but \( a \in b \) does not hold.

It follows that \( a \) must be inside \( C' \) in \( \Gamma' \).

But in this case there are two possibilities:

1) \( a \) would have to be always preceded by either \( p \) or \( \tilde{p} \), in contradiction to the fact that \( a\tilde{p}efg \in L \).

2) \( a \) would always precede either \( p \) or \( \tilde{p} \), in contradiction to the fact that \( \tilde{p}ae fg \in L \).

Consequently, our assumption that \( \Gamma \) is reducible with respect to \( \Delta \) cannot hold. \( \square \)
7. DERIVATIVES AND THEIR APPLICATIONS

In this section we formulate and prove a condition which has to be satisfied by every PCS reducible w.r.t. a set $\Delta = \Delta_p \cup \Delta_s$, where $\Delta_p$ is a set of well-formed, purely parallel PCGs, and $\Delta_s$ is a set of well-formed, purely sequential PCGs.

For this purpose we introduce the following concept of "derivative".

**Definition 7.1** Let $\Sigma$ and $\Pi$ be finite, disjoint alphabets, and $L$ a language over $\tilde{\Sigma} = \Sigma \cup \Pi \cup \bar{\Pi}$, where $\bar{\Pi} = \{\bar{p} \mid p \in \Pi\}$.

Let $\eta: \tilde{\Sigma} \rightarrow \tilde{\Sigma} \cup \Pi \cup \{\lambda\}$ be the projection defined in the proof of Proposition 5.3. We denote by $E$ the equivalence relation on $L$ defined by

$$wEw' \text{ iff } \eta(w) \text{ is a permutation of } \eta(w').$$

Then every equivalence class in $L/E$ is a derivative of $L$.

**Theorem 7.1** Let $\Delta_p$ be a set of purely parallel, well-formed PCGs, and $\Delta_s$ a set of purely sequential, well-formed PCGs. Suppose a well-formed, PCS $\Gamma$ without repetitive labelling of deciders is reducible w.r.t. $\Delta = \Delta_p \cup \Delta_s$.

Let $H$ be any derivative of $L(\Gamma)$. Then there exists a purely parallel, well-formed PCS $\Gamma_H$ with $\tilde{\Sigma} = \Sigma \cup \Pi \cup \bar{\Pi}$ as its alphabet of operation letters, such that $L(\Gamma_H) = H$ and $\Gamma_H$ is reducible w.r.t. $\Delta_p$.

**Proof:** (a) We first prove the theorem for the case that $\Gamma$ is cycle-free.

In view of Proposition 5.2, we may assume that $\Delta_s = \{D_0, D_1, D_2\}$. We use induction on the number of DECIDER nodes in $\Gamma$. 

Technion - Computer Science Department - Technical Report CS0283 - 1983
The case \( k = 0 \) is trivial.

Assume now that the theorem holds for any cycle-free PCS which contains less than \( k \) DECIDER nodes, and that \( \Gamma \) contains \( k > 0 \) DECIDER nodes.

Let \( D \) be a DECIDER node of \( \Gamma \), and \( U \) the corresponding UNION node.

Suppose \( D \) is labeled by \( p \in \Pi \). Let \( \Gamma_1 \) (resp. \( \Gamma_2 \)) be the PCS obtained from \( \Gamma \) as follows:

1. omit the part of \( \Gamma \) between the outedge of \( D \) labeled \( F \) (resp. \( T \)) and the node \( U \),
2. replace node \( D \) by an OPERATION node labeled \( p \) (resp. \( \hat{p} \)),
3. eliminate node \( U \) by "short-circuiting" it.

Evidently, both \( \Gamma_1 \) and \( \Gamma_2 \) are cycle-free and reducible w.r.t. \( \Delta \), and contain less than \( k \) DECIDER nodes. In view of the induction hypothesis, the theorem holds for both \( \Gamma_1 \) and \( \Gamma_2 \). Since every derivative of \( L(\Gamma) \) is either a derivative of \( L(\Gamma_1) \) or a derivative of \( L(\Gamma_2) \), the theorem also holds for \( \Gamma \). This completes part (a) of the proof.

(b) We now prove the theorem for the case that \( \Gamma \) contains cycles.

Similarly to part (a), we may assume that \( \Sigma_S = \{D_0, D_1, D_2\} \cup \Delta'_S \), where all the PCGs in \( \Delta'_S \) contain loops.

We shall use induction on the number \( k \) of DECIDER nodes, which are on some cycle of \( \Gamma \). The case \( k = 0 \) corresponds to part (a) of the proof. Assume now that the theorem holds for any PCS, provided the number of DECIDER nodes on cycles is less than \( k \), and that \( \Gamma \) contains \( k > 0 \) DECIDER nodes on cycles.

Since \( \Gamma \) is reducible w.r.t. \( \Delta_p \cup \{D_0, D_1, D_2\} \cup \Delta'_S \), there exist PCGs \( G_0, G_1, G_2, \ldots, G_r \) such that \( G_1 = G_0(G_j \ast G_j^\hat{\cdot}, \ldots, G_{j+1} = G_j(G_j^\hat{\cdot} \ast G_j^\hat{\cdot}), \ldots, G_r = G_r(G_r^\hat{\cdot} \ast G_r^\hat{\cdot}) \) where \( G_r = G(\Gamma) \), \( G_0 \in \Delta_s \cup \Delta_p \), \( G_i \in \Delta_s \cup \Delta_p \), ...
for \( 0 \leq i < r \), \( \hat{G}_j \in \Delta' \), and \( \hat{G}_i \in \Delta' \) for \( i > j \).

Let \( \hat{G}_j \) be the PCS obtained from \( \hat{G}_j \) as follows.

1) If \( v \) is an OPERATION node of \( \hat{G}_j \), then the node \( v \) is labeled by the letter \( v \) in \( \hat{G}_j \).

2) If \( D \) is a DECIDER node of \( \hat{G}_j \), and \( D \) is labeled by \( p(D) \) in \( \hat{G}_j \), then \( D \) is labeled by \( p(D) \) in \( \hat{G}_j \). Furthermore, the outedges of \( D \) in \( \hat{G}_j \) are labeled in accordance with the corresponding labeling of \( \Gamma \). Let \( L = L(\hat{G}_j) \). With any word \( w = a_1a_2\ldots a_m \) in \( L \) we associate the PCG \( G_w \) shown below:

![Diagram](attachment:image)

Given any \( w \in L \), let \( G_{j+1}^w = G_j(v_j + G_w) \). Starting from \( G_{j+1}^w \), we generate a composition series ending, say, in \( G^w \), which corresponds in the evident way to the composition series

\[ G_{j+1}, \ldots, G_{r-1}, G_r. \]

Let \( \Gamma^w \) be the PCS obtained from \( G^w \) by applying a labeling which corresponds to the labeling of \( \Gamma \).

Clearly, the induction hypothesis is applicable to \( \Gamma^w \), for every \( w \in L \). Hence the theorem holds for every \( \Gamma^w \). Since every derivative of \( L(\Gamma) \) is a derivative of \( L(\Gamma^w) \) for some \( w \in L \), the theorem also holds for \( \Gamma \). This completes part (b) of our theorem.