REDUCIBILITY OF SYNCHRONIZATION STRUCTURES

by

A. Ginzburg and M. Yoeli

Technical Report # 350
January 1985

* The research of the second author was supported by the Rome Air Development Center/AFSC, U.S. Air Force under grant AFSR-81-0152
REDUCIBILITY OF SYNCHRONIZATION STRUCTURES

A. Ginzburg and M. Yoeli

Dept. of Computer Science
Technion-Israel Institute of Technology
Haifa, Israel

and

Everyman's University
Tel Aviv, Israel

The research of the second author was supported by the Rome Air Development Center/AFSC, U.S. Air Force under grant AFOSR-81-0152.
ABSTRACT

This paper is concerned with synchronization structures (or "purely parallel" control structures), i.e. control structures with (2-way) forks and (2-way) joins as only control nodes. A synchronization structure is reducible iff it can be converted into an equivalent structured version, obtainable from a set of simpler ("primitive") structures. This paper derives necessary and sufficient conditions for a synchronization structure to be reducible and exhibits infinite families of irreducible synchronization structures.

Posets associated with synchronization structures, and a particular type of homomorphism ("structuring function") between posets play an important role in the paper.
1. INTRODUCTION

Structured programming has become an important methodology for the design of correct, easily understood computer programs [5]. The interest in structured programming has also led to theoretical results as to the comparative power of various sets of structuring primitives [6].

On the other hand, presently available technologies, particularly VLSI, have motivated considerable interest in multiprocessing and parallel programming [2]. 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 [1]; [3], [7].

The applicability and efficiency of structured programming evidently depend on an appropriate selection of control primitives. Consequently, many authors have been concerned with this selection problem (cf. [6]). On the other hand, the corresponding selection problem for parallel programming has received little attention so far.

An essential property of control primitives is their irreducibility (see [6]). In this paper we study irreducible parallel control structures, restricting our attention to structures without decision nodes ("synchronization" structures or "purely parallel" control structures). We introduce a suitable notion of reducibility and demonstrate the existence of infinite families of irreducible synchronization structures.
2. SYNCHRONIZATION GRAPHS

This section, which introduces the basic concept of synchronization graph, is a modified version of Section 2 of [8].

Definition 2.1 A synchronization graph (S-graph) is a finite, directed graph $\Gamma$, the nodes of which are partitioned into 5 types as shown in Fig. 1; furthermore, $\Gamma$ satisfies the following conditions:

a) Multiple edges are not admitted.

b) $\Gamma$ has exactly one START node $S$ and exactly one HALT node $H$.

c) Every node $v$ is reachable from $S$, i.e. there exists a (directed) path from $S$ to $v$.

d) The node $H$ is reachable from every node $v$.

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

Fig. 1 - Node types of S-graphs

Evidently an S-graph cannot have self-loops (i.e. cycles of length 1). Examples of S-graphs are shown in Fig. 2.

Definition 2.2 Let $\Gamma$ be an S-graph. A marking $m$ of $\Gamma$ is a function $m: E \to \omega$, where $E$ is the edge-set of $\Gamma$ and $\omega$ is the set of nonnegative integers. A marked S-graph is an ordered pair $(\Gamma, m)$ where $\Gamma$ is an S-graph and $m$ is a marking of $\Gamma$. 
Let $e$ be an edge of the marked $S$-graph $(\Gamma, m)$. We refer to the integer $m(e)$ as the number of tokens on $e$. If $m(e) > 0$, we say that $e$ is marked. In the graphical representation of marked $S$-graphs, tokens are indicated by dots ($\bullet$). Fig. 3 shows examples of marked $S$-graphs.
Definition 2.3 Let \((\Gamma, m)\) be a marked S-graph. A node of type OPERATION or FORK is enabled iff its inedge is marked. A JOIN node is enabled iff both its inedges are marked.

A node which is enabled may fire. The firing rules, illustrated in Fig. 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 an OPERATION node decreases the marking of its inedge by 1, and increases the marking of its outedge by 1.

![Diagram](image)

**Fig. 4 - Examples of "firings!"**

For example, node \(J\) in Fig. 3(a) is enabled. The firing of \(J\) yields the marked S-graph of Fig. 3(b).
3. SYNCHRONIZATION STRUCTURES

In this section we define \textit{synchronization structures} as "well-formed" S-graphs.

Let \( m \) and \( m' \) be markings of the S-graph \( \Gamma \). 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 obtainable from \( m \) by the successive firing of one or more nodes of \( \Gamma \). Furthermore we set
\[
[m] = \{ m' \mid m \rightarrow m' \} \cup \{ m \}.
\]
We shall refer to \([m]\) as the set of all markings \textit{reachable} from \( m \).

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

\textbf{Definition 3.1} The initial marking \( m_0 \) of an S-graph \( \Gamma \) is defined as follows:
\[
m_0(e_S) = 1 \quad \text{and} \quad m_0(e) = 0 \quad \text{for every} \quad e \neq e_S.
\]
A marking \( m \) of \( \Gamma \) is final iff \( m(e_H) > 0 \). We denote by \( M_F \) the set of all final markings of \( \Gamma \).

\textbf{Definition 3.2} An S-graph is \textit{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 \( m \).

\textbf{Definition 3.3} Let \( \Gamma \) be an S-graph and \( E \) its edge set. \( \Gamma \) is \textit{residue-free} iff
\[
(\forall m \in [m_0])\left[ m \in M_F \rightarrow \sum_{e \in E} m(e) = 1 \right],
\]
i.e. for any final marking \( m \) reachable from \( m_0 \), the marked S-graph \( (\Gamma, m) \) contains exactly one token (namely on \( e_H \)).

**Definition 3.4** An S-graph \( \Gamma \) is well-formed iff \( \Gamma \) is both terminating and residue-free.

The S-graph \( \Gamma_1 \) of Fig. 2(a) is well-formed, whereas the S-graph \( \Gamma_2 \) of Fig. 2(b) is not terminating. We shall refer to well-formed S-graphs as synchronization structures or S-structures.

**Definition 3.5** An S-graph \( \Gamma \) 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 result is an immediate consequence of Theorem 3.1 in [8].

**Proposition 3.1** Every S-structure is safe.

Furthermore, we prove

**Theorem 3.1** An S-graph is well-formed iff it is cycle-free.

**Proof** Let \( \Gamma \) be an S-graph containing a cycle. Then, by Proposition 4.2 of [8], \( \Gamma \) is not well-formed. Conversely, let \( \Gamma \) be an S-graph without cycles. Consider any directed path \( P \) from the START node \( S \) to the HALT node \( H \). The total number of tokens on \( P \) (its "token count") cannot change by any firing. For the initial marking \( m_0 \) of \( \Gamma \) every such path has token count 1. Let now \( m \in [m_0] \), and assume that some JOIN \( J \) has only one inedge, marked by \( m \), and is not firable under any marking \( m' \) reachable from \( m \). Assume further, that no other JOIN node between the START node \( S \) and node \( J \)
has this property (this assumption is justified since \( \Gamma \) is cycle-free).

Every directed path from \( J \) to \( H \) must be token-free (otherwise we would have a path from \( S \) through \( J \) to \( H \) with at least two tokens). Consider now a directed path from \( S \) to \( J \) via the other inedge of \( J \). It follows that this path must contain a token. This token can be "moved forward" to enable \( J \), except for the case when its "movement is blocked" by another JOIN node. However, this is impossible, in view of our assumption. Thus, node \( J \) becomes firable, in contradiction to our assumption about \( J \). Hence \( \Gamma \) does not contain any "blocking" JOIN node and must, therefore, be terminating. In view of the fact that every directed path from \( S \) to \( H \) has token count 1, \( \Gamma \) is also residue-free. Thus \( \Gamma \) is well-formed.

The above result could be proven alternatively by applying the theory of marked graphs (cf.[4]).

4. S-GRAPH LANGUAGES

Definition 4.1 Let \( \Gamma \) be an S-graph and \( V \) its set of vertices.

Let \( w \in V^+ \), i.e., \( w \) is a finite string of vertices, \( w = v_1 v_2 \ldots v_k \). 

\( w \) is called a firing sequence of \( \Gamma \) iff there exist markings \( m_1, m_2, \ldots, m_k \), such that

\[
\begin{align*}
  m_0 &\xrightarrow{v_1} m_1, \\
  m_1 &\xrightarrow{v_2} m_2, \ldots, \\
  m_{k-1} &\xrightarrow{v_k} m_k.
\end{align*}
\]

In this case we write \( m_0 \xrightarrow{w} m_k \).

Proposition 4.1 Let \( \Gamma \) be an S-structure and assume \( m_0 \xrightarrow{w} m \), where \( w \in V^+ \), and the marking \( m \in M_\Gamma \). Then \( w \) contains every
node of \( r \), except the START and HALT nodes, and each exactly once.

**Proof**. By Theorem 3.1, \( r \) is cycle-free. Let \( w = v_1v_2 \ldots v_k \) and assume
\[
m_0 \xrightarrow{v_1} m_1, \ldots, m_{k-1} \xrightarrow{v_k} m_k = m.
\]
Consider any directed path \( P \) from the START node \( S \) to the HALT node \( H \). The token count of \( P \) (see proof of Theorem 3.1) under the markings \( m_0, m_1, \ldots, m_k \) is constantly 1. It follows that the firing sequence \( w \) causes a single token to "travel along" the path \( P \).

Thus, every internal node on \( P \) will fire, and exactly once. Since every internal node is on some directed path from \( S \) to \( H \), the proposition follows. 

Let \( E \) denote the set of OPERATION nodes of an \( S \)-graph \( r \) with vertex set \( V \). We denote by \( \pi \) the projection \( \pi : V^* \to E^* \), i.e. for any \( w = v_1v_2 \ldots v_k \in V^* \), \( \pi(w) \) is obtained from \( w \) by replacing each \( v_i \in V - E \) by the empty word \( \lambda \).

**Definition 4.2**. Let \( r \) be an \( S \)-graph. Its *language* \( L(r) \) is defined by:
\[
L(r) = \{ \pi(w) \mid (\exists m \in M_r) m \xrightarrow{w} m \}.
\]
For example, for the \( S \)-graph \( r_1 \) of Fig. 2(a) we have
\[
L(r_1) = \{ abcd, abdc, acbd, bcad, badc \}.
\]

5. \( S \)-STRUCTURES AND POSETS

Every \( S \)-structure \( r \) defines, in a rather obvious way, a precedence relation on the operations represented by the OPERATION nodes of \( r \).

Accordingly we define:
Definition 5.1 Let $\Gamma$ be an S-structure and $\Sigma$ its set of nodes. With $\Gamma$ we associate the poset (partially ordered set) $G(\Gamma) = (\Sigma, \sqsubseteq)$, where $\sqsubseteq$ is a partial order relation on $\Sigma$:

$x \sqsubseteq y$ holds iff there exists a directed path in $\Gamma$ from node $x$ to node $y$, where $x \neq y$; $x \sqsubseteq y$ holds iff $x \sqsubseteq y$ or $x = y$.

For example, the poset $G(\Gamma_1)$, defined by the S-structure $\Gamma_1$ of Fig. 2(a) is shown (in the usual way of representing posets) in Fig. 5.

![Diagram](image)

Fig.5 - Poset $G(\Gamma_1)$ for the S-structure $\Gamma_1$ of Fig. 2(a).

Theorem 5.1 Let $G = (\Sigma, \sqsubseteq)$ be an arbitrary, finite poset. Then there exists an S-structure $\Gamma_G$ such that $G = G(\Gamma_G)$.

Proof We proceed by induction on $|\Sigma|$. The case $|\Sigma| = 1$ is trivial.

Assume now that the theorem holds for every poset with $|\Sigma| \leq n$ and let $G = (\Sigma, \sqsubseteq)$ be a poset with $|\Sigma| = n+1$. Since $G$ is finite, there exists at least one maximal element in $G$, say $a$. Let $b_1, \ldots, b_k$ (for $k \geq 0$) be the immediate successors of node $a$ in $G$. Denote by $G'$ the poset obtained from $G$ by deleting node $a$ together with its outedges. By the induction hypothesis there exists an $S$-structure $\Gamma'$ such that $G(\Gamma') = G'$. Fig. 6 illustrates how to obtain the S-structure $\Gamma$ from $\Gamma'$ (Fig. 6(a), (b) for $k = 0$, and Fig. 6(c), (d) for $k = 3$).
With any finite poset \( G = (\Sigma, \sqsubseteq) \) we associate a language \( L(G) \) as follows.

**Definition 5.2** Let \( G = (\Sigma, \sqsubseteq) \) be a finite poset. We define its language \( L(G) \subseteq \Sigma^* \) as the set of all permutations of all elements (letters) of \( \Sigma \) which preserve the partial order \( \sqsubseteq \) of \( G \). Namely, let \( \Sigma = \{\sigma_1, \ldots, \sigma_k\} \). Then

\[
\sigma_1 \sigma_2 \ldots \sigma_k \in L(G)
\]

iff \((\forall h)(\forall j) (\sigma_i \sqsubseteq \sigma_j \Rightarrow h_i < j)\).

Clearly if for two posets \( G_1 \) and \( G_2 \) \( L(G_1) = L(G_2) \), then \( G_1 \) and \( G_2 \) coincide.

For example, for the poset \( G(\Gamma_1) \) we have
\[
L(G(\Gamma_1)) = \{abcd, abdc, acbd, bacd, badc\}.
\]

Thus \( L(G(\Gamma_1)) = L(\Gamma_1) \).

This illustrates the following.

**Theorem 5.2** Let \( \Gamma \) be an S-structure. Then \( L(\Gamma) = L(G(\Gamma)) \).

**Proof** Assume \( w \in L(\Gamma) \). By Proposition 4.1, every letter of \( \Sigma \) appears in \( w \) exactly once. If \( x \sqsubseteq y \) in \( G(\Gamma) \), there exists a directed path in \( \Gamma \) from \( x \) to \( y \). Thus \( x \) must fire before \( y \). Consequently, \( x \) precedes \( y \) in \( w \). It follows that \( w \in L(G(\Gamma)) \).

Conversely, let \( w = a_1 a_2 \ldots a_k \) be in \( L(G(\Gamma)) \). We have to show the existence of a firing sequence \( \bar{w} \) in \( \Gamma \), such that \( m_0 \xrightarrow{\bar{w}} m \), where \( m \in M_\Gamma \) and \( \pi(\bar{w}) = w \).

Let \( a \) be an arbitrary OPERATION node in \( \Gamma \), and consider the subgraph \( \Gamma_a \) of \( \Gamma \) consisting of all paths from \( S \) to \( a \), including \( S \) and \( a \). Notice that if one inedge of a JOIN \( J \) belongs to \( \Gamma_a \),
so does the other. In case of a FORK $F$ it is possible, however, that only one outedge of $F$ belongs to $\Gamma_a$. In this case, this FORK is "short-circuited". $\Gamma_a$ is clearly cycle-free (since $\Gamma$ is cycle-free), hence $\Gamma_a$ is terminating. It follows that any fixing sequence in $\Gamma$ which contains all OPERATION nodes of $\Gamma_a$ except $a$ itself can be continued to a firing sequence in which the next OPERATION node is $a$.

This observation allows an inductive construction of a required firing sequence $\hat{w}$.

6. REDUCIBILITY OF S-STRUCTURES

The concept of reducibility plays an important role in the theory of structured programming (see [6]). In this section we introduce a suitable notion of reducibility, applicable to $S$-structures.

**Definition 6.1** Let $\Gamma_1$ and $\Gamma_2$ be $S$-structures, each containing more than one OPERATION node, and let $a$ be an OPERATION node of $\Gamma_1$. Assume $(\Sigma_1 \setminus \{a\}) \cap \Sigma_2 = \emptyset$. Then the refinement $\Gamma = \Gamma_1(a + \Gamma_2)$ of $\Gamma_1$ is the $S$-structure $\Gamma$ defined as indicated in Fig. 7.

The refinement of an $S$-structure (Definition 6.1) corresponds in an evident way to the "stepwise refinement" concept of structured programming (cf. [5]).

**Definition 6.2** An $S$-structure $\Gamma$ is *reducible*, iff there exists an $S$-structure $\Gamma'$ such that $L(\Gamma') = L(\Gamma)$ and $\Gamma'$ can be obtained as a refinement. Otherwise $\Gamma$ is *irreducible*. 
Fig. 7 - Illustrating the concept of refinement
(a) $\Sigma$-structure $\Gamma_1$
(b) $\Sigma$-structure $\Gamma_2$
(c) Refinement $\Gamma = \Gamma_1(a + \Gamma_2)$
Definition 6.3  Given posets $G$ and $H$, we say that a function $f$ from $G$ onto $H$ is an $H$-structuring of $G$, iff for every $g_1, g_2 \in G$:

1. $g_1 \sqsubseteq g_2 \implies f(g_1) \sqsubseteq f(g_2)$
2. $f(g_1) \sqsubseteq f(g_2) \implies g_1 \sqsubseteq g_2$.

We shall also say that $f$ is a structuring function of $G$. For the above-defined notion of refinement we have:

Let $\Gamma = \Gamma_1(a + \Gamma_2)$, and consider the function $f$ from $G(\Gamma)$ onto $G(\Gamma_1)$ defined by

$$(\forall \sigma \in \Sigma_1\{a\}) f(\sigma) = a$$
$$(\forall \sigma \in \Sigma_2) f(\sigma) = a.$$ 

Then $f$ is a $G(\Gamma_1)$-structuring of $G(\Gamma)$.

Consequently, we have

Proposition 6.1  If an $S$-structure $\Gamma$ is reducible, then there exists a non-trivial structuring function of $G(\Gamma)$.

Proof  Indeed, if for some $\Gamma'$ obtainable as refinement, $L(\Gamma') = L(\Gamma)$, then $L(G(\Gamma)) = L(G(\Gamma'))$, hence $G(\Gamma) = G(\Gamma')$. In view of the above observation, there exists a non-trivial structuring function of $G(\Gamma') = G(\Gamma)$. □

The converse of Proposition 6.1 is also valid. In order to obtain this result, we need the following lemma.

Lemma 6.1  Given an $S$-structure $\Gamma$ and an $H$-structuring $f: G(\Gamma) \to H$ of $G(\Gamma)$, let $h \in H$. Consider two elements $g_1, g_2 \in G(\Gamma)$ such that $f(g_1) = f(g_2) = h$. Let $g$ be an OPERATION node on a directed path from OPERATION node $g_1$ to OPERATION node $g_2$ in $\Gamma$. Then $f(g) = h$.

Proof  In $G(\Gamma)$ we have $g_1 \sqsubseteq g \sqsubseteq g_2$. Hence $h = f(g_1) \sqsubseteq f(g) \sqsubseteq f(g_2) = h$. Thus $f(g) = h$. □
Assume again the notation of Lemma 6.1. As usual, \( f^{-1}(h) \) denotes the set of nodes in \( G(\Gamma) \) mapped by \( f \) onto \( h \). Let \( h \) be the corresponding set of OPERATION nodes in \( \Gamma \), together with all directed paths between them. By Lemma 6.1, this set of OPERATION nodes contains all the nodes of \( \Gamma \) on the above paths.

Now consider all nodes in \( h \) which have an outedge not leading to another node in \( h \). Connect them by a suitable number of JOIN nodes to a new (HALT) node \( H_h \). Similarly, form a new (START) node \( S_h \) and connect it by a suitable number of FORK nodes to all nodes in \( h \) which have an inedge from some node outside of \( h \).

This construction yields an S-structure which we denote by \( \Gamma_h \). We use this construction in the following algorithm, which provides a constructive proof of the converse of Proposition 6.1.

**Reduction Algorithm**

Let \( \Gamma \) be an S-structure.

**Step 1** Construct \( G(\Gamma) \).

**Step 2** Find, if possible, a non-trivial structuring function \( f \) from \( G(\Gamma) \) onto some poset \( H \). If no such structuring function exists, then \( \Gamma \) is irreducible, by Proposition 6.1.

**Step 3** For every node \( h \) of \( H \) construct \( \Gamma_h \). Furthermore, construct \( \Gamma_H \) (see Theorem 5.1).

**Step 4** Let \( h_1, ..., h_m \) be the nodes of \( H \), listed in any order.

Generate successively the refinements

\[
\Gamma_1 = \Gamma_h(h_1 \circ \Gamma_{h_1}), ..., \Gamma_m = \Gamma_{m-1}(h_m \circ \Gamma_{h_m})
\]

**Theorem 6.1** Using the notation of the Reduction Algorithm, we have

\[
L(\Gamma) = L(\Gamma_m)
\]
Proof  It suffices to show that $G(r_m) = G(r)$. Indeed, if $a_1, a_2$ are OPERATION nodes belonging to the same $r_h$, then $a_1 \sqsubseteq a_2$ in $G(r_m)$ iff $a_1 \sqsubseteq a_2$ in $G(r)$, since $r_h$ is essentially a part of $r$.

Let now $a_1$ be an OPERATION node in $r_{h_i}$ and $a_2$ an OPERATION node in $r_{h_j}$, where $i \neq j$. If $a_1 \sqsubseteq a_2$ in $G(r)$, then $h_i \sqsubseteq h_j$ in $H$, hence also in $r_H$. Thus all nodes of $r_{h_i}$ precede all nodes of $r_{h_j}$. In particular, $a_1 \sqsubseteq a_2$ in $G(r_m)$. Conversely, if $a_1 \sqsubseteq a_2$ in $G(r_m)$, then $h_i \sqsubseteq h_j$ in $H$, hence, by (2) of Definition 6.3, $a_1 \sqsubseteq a_2$ in $G(r)$. □

Together with Proposition 6.1, we thus have

Theorem 6.2  An S-structure $r$ is reducible iff $G(r)$ has a non-trivial structuring function.

7. A UNIQUENESS THEOREM

Definition 7.1  A poset with more than one element is irreducible iff it has no non-trivial structuring function.

In this section we prove the following Uniqueness Theorem.

Theorem 7.1  Let $G$ be a poset, $H_1$ and $H_2$ irreducible posets, and $f_1: G \rightarrow H_1$, $f_2: G \rightarrow H_2$ structuring functions of $G$. Then $H_1$ and $H_2$ are isomorphic.

For the proof of this theorem, we need the following lemma.

Lemma 7.1  Let $G, H_1, H_2$ be posets, and $f_i: G \rightarrow H_i$ ($i = 1, 2$) structuring functions of $G$.

Assume $f_1(a) = f_1(b)$, but $f_2(a) \neq f_2(b)$. Define the function $f_3: G \rightarrow H_2$ as follows:
\[ f'_3 (b) = f_2 (a) \]
\[
(\forall g \in (G - \{b\})) f'_3 (g) = f_2 (g).
\]
Then \( f'_3 \) is an \( f'_3 (G) \)-structuring of \( G \).

**Proof** Let \( x \) be a node of \( G \).

(1) \( b \subseteq x \)
\[
\Rightarrow f'_1 (b) \subseteq f'_1 (x)
\[
\Rightarrow f'_1 (a) \subseteq f'_1 (x)
\[
\Rightarrow a \subseteq x
\]
\[
\Rightarrow f'_2 (a) \subseteq f'_2 (x)
\]
\[
\Rightarrow f'_3 (b) \subseteq f'_3 (x)
\]

(2) similarly for \( x \subseteq b \)

(3) \( f'_3 (b) \subseteq f'_3 (x) \)
\[
\Rightarrow f'_2 (a) \subseteq f'_2 (x)
\]
\[
\Rightarrow a \subseteq x
\]
\[
\Rightarrow f'_1 (a) \subseteq f'_1 (x)
\]
\[
\Rightarrow f'_2 (b) \subseteq f'_2 (x)
\]

(4) similarly for \( f'_3 (x) \subseteq f'_3 (b) \).

---

**Proof of Theorem 7.1** Assume first that \( |H_1| = |H_2| = 2 \).

If \( H_1 \) and \( H_2 \) are different, then, say
\[
H_1 = \bullet \bullet \quad \text{and} \quad H_2 = \begin{array}{c}
\bullet \\
\bullet \\
\circ \\
\circ
\end{array}
\]
Assume that for \( x \) and \( y \) in \( G \), \( f'_2 (x) = c \), and \( f'_2 (y) = d \). Then \( x \subseteq y \) and, therefore, \( f'_1 (x) \subseteq f'_1 (y) \). In view of the structure of \( H_1 \) we must have \( f'_1 (x) = f'_1 (y) \), say = \( a \). Let now \( z \) be an element of \( G \). If \( f'_2 (z) = c \), then \( z \subseteq y \). If \( f'_2 (z) = d \), then \( x \subseteq z \). In both cases, \( f'_1 (z) \) must be \( a \). Thus \( f'_1 \) is not onto \( H_1 \), in contradiction to our assumptions.
Now assume that, say, \(|H_2| > 2\), and that \(|H_2| \geq |H_1|\). Thus the partition \(\pi_1\) of \(G\) induced by \(H_1\) is not a proper refinement of the partition \(\pi_2\) of \(G\), induced by \(H_2\). If \(\pi_1 = \pi_2\), the theorem follows. If \(\pi_1 \neq \pi_2\), then there exist elements \(a, b\) in \(G\) such that \(f_1(a) \neq f_3(b)\), but \(f_2(a) \neq f_2(b)\). Applying Lemma 7.1, construct a structuring function \(f_3\) with \(f_3(a) = f_3(b)\). If no \(c \in G\) exists, such that \(f_2(b) = f_2(c)\), then \(|f_3(G)| = |H_2| - 1 \geq 2\). One easily verifies that \(f_3 \circ f_2^{-1}\) is a structuring function of \(H_2\) onto \(f_3(G)\), in contradiction to our assumption that \(H_2\) is irreducible. It follows that there exists an element \(c \in G\) such that \(f_2(b) = f_2(c)\).

Consequently, \(f_3(G) = H_2\).

If the partitions of \(G\) induced by \(f_1\) and \(f_3\) do not coincide, this construction can be continued, till a function \(f_k\) is reached, such that \(f_k\) is an \(H_2\)-structuring of \(G\), and the partitions induced by \(f_1\) and \(f_k\) coincide. It follows that \(H_1\) and \(H_2\) are isomorphic.

\[\square\]

8. FAMILIES OF IRREDUCIBLE S-STRUCTURES

In this section we exhibit infinite families of irreducible S-structures.

Consider the poset \(K_n\), \(n \geq 1\), shown in Fig. 8.

![Fig. 8 - Poset \(K_n\)]
Proposition 8.1  The poset $K_n$ (see Fig. 8), $n \geq 1$, is irreducible.

Proof. Assume $f$ is a structuring function of $K_n$, and say $f(3) = f(4)$. But $2 \subseteq 3$, hence $f(2) \subseteq f(3) = f(4)$. Since $f(2) \subseteq f(4)$ would imply $2 \subseteq 4$, which is not the case, we must have $f(2) = f(4)$. Continuing this argument both "to the left" and "to the right", we find that $f$ is trivial.

Assume now, say, $f(3) = f(6)$. As before, we have $2 \subseteq 3$, but $2$ and $6$ are not comparable. This leads to the conclusion that $f(2) = f(3) = f(6)$. Applying the previous argument, we conclude that $f$ is trivial. \hfill \Box

It follows that the $S$-structures $\Gamma_K$, $n \geq 1$, are all irreducible. $\Gamma_{K_3}$ is shown in Fig. 9.

\begin{figure}[h]
\centering
\includegraphics[width=0.5\textwidth]{fig9.png}
\caption{$\Gamma_{K_3}$}
\end{figure}
Another infinite family of irreducible $S$-structures $C_n$, $n \geq 1$, is shown in Fig. 10. The corresponding poset $G(C_n)$ is shown in Fig. 11.
The previous proof technique can be immediately applied to show that any structuring function of \( G(C_n) \) for \( n \geq 1 \) is trivial. Hence \( C_n \) is irreducible. (Indeed, notice that \( f(2) = f(2n-1) \) implies that \( f \) is trivial and that \( f(i) = f(j) \) for \( i \neq j \) easily yields \( f(2) = f(2n-1) \).)

9. FURTHER RESEARCH

Further research is in progress dealing with parallel control structures which also include decision and choice nodes. Furthermore, alternative reducibility criteria are also being investigated.
REFERENCES


