and $TA(v_y) = TA(v_x) + (y_{1i} - x_{1i}) \times d_{1i} + (y_{1j} - x_{1j}) \times d_{1j}$

Definition 5.1-2: A diagonal in a mesh graph $MG$ is a 2-tuple $D=(V_D,k)$ where

1. $V_D$ is a set of ordered computation vertices from $MG$ and the ordering ranges from 1 to $|V_D|$ and if $v_x$ and $v_y$ are in $V_D$ such that $x_{1j}$ is less than $y_{1j}$ then the index of $v_x$ in the ordering is less than the index of $v_y$ in the ordering.

2. $k$ is such that
   a. either for every $v_x$ in $V_D$, $x_{1i} + x_{1j} = k$
   b. or for every $v_x$ in $V_D$, $x_{1i} - x_{1j} = k$

Let $D_1 = \{ D \text{ such that every } v_x \text{ in } V_D \text{ satisfies (2a)} \}$ and $D_r = \{ D \text{ such that every } v_x \text{ in } V_D \text{ satisfies (2b)} \}$. Augment $V_D$ of every $D$ in $D_1$ by a dummy source vertex labelled 1l, a dummy sink vertex labelled 11 and a set of dummy directed edges labelled 1l. Assign the index 0 to the source vertex and the index $|V_D|+1$ to the sink vertex. For every pair of adjacent vertices $v_x$ and $v_y$ in the ordering of all the vertices in the augmented $V_D$ if the index of $v_y$ is greater than the index of $v_x$ by 1 then direct a dummy edge labelled 1l from $v_x$ to $v_y$. Consequently every $D$ in $D_1$ is a major path labelled 1l. Similarly augment every $D$ in $D_r$ by a dummy source vertex labelled 1r, a dummy sink vertex labelled 1r and a

\[1\text{not part of the original graph}\]
set of dummy edges labelled \( lr \) and hence every \( D \) in \( D_r \) is a major path labelled \( lr \).

**Example 5.1-2:** Consider the mesh graph \( MG \) of example 5.1-1. \( D_r \) and \( D_1 \) in \( MG \) is shown in figure 5.1-2 below:

![Diagram](image)

Figure 5.1-2

In figure 5.1-2 the dummy source and sink vertices are denoted by \( 'O' \).

Let \( MG_{aug} \) denote the mesh graph \( MG \) augmented by the set of dummy edges, dummy source and sink vertices in \( D_1 \) and \( D_r \). Let \( A \) be the set of major paths in \( MG_{aug} \) such that for any pair of computation vertices \( v_x \) and \( v_y \) in any major path of \( A \), \( PA(v_x) = PA(v_y) \) and if \( v_x \) and \( v_y \) are in different major paths of \( A \) then \( PA(v_x) \neq PA(v_y) \). Let \( B = E_{11} \) if \( A = E_{11} \) and \( B = E_{1j} \) if \( A = E_{1j} \).

**Lemma 5.1-1:**

In any syntactically correct mapping of \( MG_{aug} \) the pair \( \langle A, B \rangle \) must be one of \( \{ \langle E_{1j}, E_{11} \rangle, \langle D_1, E_{11} \rangle, \langle D_r, E_{11} \rangle, \langle E_{11}, E_{1j} \rangle \} \).
Proof: Let x and y be the value assumed by n_{1i} and n_{1j} respectively. Now by lemma 4.0-1 n_{1i}=\emptyset and hence \langle x, y \rangle must be one of \{\langle 1, \emptyset \rangle, \langle 1, 1 \rangle, \langle 1, -1 \rangle, \langle \emptyset, 1 \rangle, \langle -1, \emptyset \rangle, \langle -1, 1 \rangle, \langle -1, -1 \rangle, \langle \emptyset, -1 \rangle \}. If M_1 is the mapping that results from \langle n_{1i}, n_{1j} \rangle=\langle x, y \rangle then the mapping M_2 that results from \langle n_{1i}, n_{1j} \rangle=\langle -x, -y \rangle is the same as M_1 with the array in M_1 reversed from end to end. Hence we need to consider only the four different mappings that result from \langle x, y \rangle assuming each of \{\langle 1, \emptyset \rangle, \langle 1, 1 \rangle, \langle 1, -1 \rangle, \langle \emptyset, 1 \rangle \}. By forming A for each of these four different values the lemma follows.

Let l_a and l_b denote the label of edges, source and sink vertices in A and B respectively. Clearly l_a must be one of \{l_i, l_j, l_l, l_r\} and l_b must be one of \{l_i, l_j\}.

**Lemma 5.1-2:**

In any syntactically correct mapping of MG_{aug}, n_{1b} must be 1 and n_{1a} must be \emptyset.

Proof: n_{1a} must be \emptyset follows from definition of A. From lemma 5.1-1 \langle A, B \rangle must be one of \{\langle E_{1j}, E_{1l} \rangle, \langle D_{1l}, E_{1l} \rangle, \langle D_r, E_{1l} \rangle, \langle E_{1l}, E_{1j} \rangle \}. It can be easily verified that B=E_{1l} if n_{1i}=1 and B=E_{1j} if n_{1j}=1 and hence n_{1b} must be 1.

Now any maximally-connected subgraph SG of MG_{aug} with L_{SG}=\{l_a, l_b\} is a mesh graph and consequently the relations r_{la} and r_{lb} impose a linear chain on B and A respectively and hence the major paths in A and B are also totally ordered. Now for any computation vertex v, x_{1b} and x_{1a} are
its horizontal and vertical co-ordinates. Also $x_{1b}$ and $x_{1a}$ are the indices of the major path in A and B respectively that pass through $v_x$.

Lemma 5.1-3:

If $v_x$, $v_y$, $v_w$ and $v_u$ are any four computation vertices in $MG_{aug}$ such that $y_{1b} - x_{1b} = w_{1b} - u_{1b}$ and if in any syntactically correct mapping of $MG_{aug}$ $TA(v_y) - TA(v_x) = TA(v_w) - TA(v_u)$ then $y_{1a} - x_{1a} = w_{1a} - u_{1a}$.

Proof: Immediate from proposition 5.1-2.

Lemma 5.1-4:

If $v_x$ and $v_y$ are any two computation vertices in $MG_{aug}$ such that $y_{1b} - x_{1b} = k * m$ and $y_{1a} - x_{1a} = k * n$ where $k$, $m$, $n$ are integers then in any syntactically correct mapping of $MG_{aug}$, $v_x <_{m,o}^k v_y$ where $o = (n * d_{1a} + m * d_{1b})$.

Proof: By proposition 5.1-2,

$PA(v_y) = (y_{1b} - x_{1b}) * n_{1b} + (y_{1a} - x_{1a}) * n_{1a} + PA(v_x)$
and $TA(v_y) = (y_{1b} - x_{1b}) * d_{1b} + (y_{1a} - x_{1a}) * d_{1a} + TA(v_x)$

By lemma 5.1-2 $n_{1a} = 0$ and $n_{1b} = 1$ and hence

$PA(v_y) = k * m + PA(v_x)$
and $TA(v_y) = k * o + TA(v_x)$

and hence the lemma.
5.2. Main Result

We will examine the syntactic properties of a class CL of program graphs such that there exists a syntactically correct mapping for every member of this class. In every program graph G in CL there exists at least a pair of labels li and lj such that the number of maximally-connected subgraphs SG with $L_{SG} = \{li, lj\}$ is 1. Hence all the major paths in $E_{li}$ and $E_{lj}$ are in this subgraph. By theorem 5.1-1 SG must be a mesh graph. Augment the subgraph SG with the two sets of diagonal paths $D_1$ and $D_r$. Define the sets A and B as for the augmented SG as done earlier in this section. Let $S_A = \{r_{lp}^{-1} | r_{lp} \text{ is a binary relation on the major paths in } B \text{ and } E_{lp} \text{ is in } H\}$. Let $S_B = \{r_{lp} | r_{lp} \text{ is a binary relation on major paths in } B \text{ and } E_{lp} \text{ is in } H\}$. By lemma 5.1-1 the pair $<A,B>$ must assume only one of $\{<E_{lj},E_{li}>, <D_1,E_{li}>, <D_r,E_{li}>, E_{lj},E_{lj}>\}$. Let us denote this set of four tuples as $H$.

Theorem 5.2-1:

There exists a syntactically correct mapping for any $G$ in CL iff there exists at least one tuple in $H$ assumed by $<A,B>$ such that each of the following condition is satisfied:

1. $S_A$ imposes a consistent order on the major paths in $A$ and the consistency constant of every relation in $S_A$ in the ordering imposed by $S_A$ is one of $\{1, -1, \emptyset\}$.

2. $S_B$ imposes a consistent order on the major paths in $B$.

---

1 refer definition 2.1-6, page 7
3. for any pair of computation vertices \( v_x \) and \( v_y \) in \( G \) and for any label if \( y_{lb} - x_{lb} = k^*a_{lp} \) and \( y_{la} - x_{la} = k^*b_{lp} \) where \( a_{lp} \) and \( b_{lp} \) are consistency constants of relations \( r_{lp} \) in the order imposed by \( S_A \) and \( S_B \) respectively then there must be a single major path labelled \( lp \) that passes through \( v_x \) and \( v_y \) such that the distance from \( v_x \) to \( v_y \) in the major path is \( k \).

4. for any relation \( r_{lp} \) in \( S_A \) if \( a_{lp} = \emptyset \) then there cannot exist a transitive edge labelled \( lp \).

Proof:

Necessity:

1. Consider any relation \( r_{lp} \) in \( S_A \). Now let \( v_x \) and \( v_y \) be any pair of computation vertices in any two major paths in \( A \) such that there is an edge labelled \( lp \) directed from \( v_x \) to \( v_y \). Now \( PA(v_y) = PA(v_x) + n_{lp} \) where \( n_{lp} \) is one of \( \{+1, -1, \emptyset\} \). Also \( PA(v_y) = y_{lb} \) and \( PA(v_x) = x_{lb} \) and hence \( y_{lb} - x_{lb} = n_{lp} \) and hence condition (1).

2. Again consider any relation \( r_{lp} \) in \( S_A \) and \( S_B \). Consider any two pair of computation vertices \( \langle v_x, v_y \rangle \) and \( \langle v_u, v_w \rangle \) such that there is an edge labelled \( lp \) directed from \( v_x \) to \( v_y \) and also another edge labelled \( lp \) directed from \( v_u \) to \( v_w \). Now \( d_{lp} \) is a constant and hence \( TA(v_w) - TA(v_u) = TA(v_y) - TA(v_x) \) and hence \( (y_{lb} - x_{lb}) * d_{lb} + (y_{la} - x_{la}) * d_{la} = (w_{lb} - u_{lb}) * d_{lb} + (w_{la} - u_{la}) * d_{la} \).

But by condition (1) \( y_{lb} - x_{lb} = w_{lb} - u_{lb} \) and hence
$y_{la} = x_{la} = w_{la} - u_{la}$ and hence condition (2).

3. We first show that $n_{lp} = a_{lp}$ and $d_{lp} = a_{lp} * d_{lb} + b_{lp} * d_{la}$.
Consider any pair of computation vertices $v_u$ and $v_w$ such that there is an edge labelled $lp$ directed from $v_u$ to $v_w$.
Now by condition (1), $w_{lb} - u_{lb} = a_{lp}$. Also $PA(v_w) = w_{lb}$ and $PA(v_u) = u_{lb}$ and hence $PA(v_w) - PA(v_u) = n_{lp} = a_{lp}$. Also $TA(v_w) - TA(v_u) = d_{lp} = (w_{lb} - u_{lb}) * d_{lb} + (w_{la} - u_{la}) * d_{la}$. From condition (2), $w_{la} - u_{la} = b_{lp}$ and hence $d_{lp} = a_{lp} * d_{lb} + b_{lp} * d_{la}$.

Now $n_{lb} = 1$ and $n_{la} = \emptyset$ and hence $PA(v_y) = PA(v_x) + k * a_{lp}$ and hence $PA(v_y) = PA(v_x) + k * n_{lp}$. From condition (2), $d_{lp} = a_{lp} * d_{lb} + b_{lp} * d_{la}$ and hence $TA(v_y) = TA(v_x) + k * d_{lp}$ and hence $v_x \overset{k}{\leftrightarrow}_{m, n} v_y$ where $m = n_{lp}$ and $n = d_{lp}$. If $n_{lp} = \emptyset$ then by definition of a syntactically correct mapping condition (3) follows. If $n_{lp} \neq \emptyset$ then by lemma 4.0-3 condition (3) follows.

4. Supposing there existed a $E_{lp}$ in $H$ such that $n_{lp} = \emptyset$. By condition (3) $E_{lp}$ must be identical to $A$ and hence $d_{lp} = d_{la}$. We show that if $E_{lp}$ exists then $d_{lb}$ must be some multiple of $d_{lp}$.

Consider two consecutively indexed major paths $q_1$ and $q_2$ in $E_{lp}$ as shown in figure 5.2-1 below:
Let PA(v_x) = p and hence PA(v_y) = p+1. Let TA(v_x) = t_1 and TA(v_y) = t_2 and hence \( t_2 = d_{lb} + (y_{la} - x_{la}) \cdot d_{lp} + t_1 \)\. Let \( y_{la} = x_{la} = k \) and hence \( t_2 = t_1 + k \cdot d_{lp} + d_{lb} \). The start time \( t_s < t_1 \). Besides there must be two tuples \( <v_{s1}, t_s> \) and \( <v_{s2}, t_s> \) in \( S_0 \) such that IOA\( <v_{s1}, t_s> = p \) and IOA\( <v_{s2}, t_s> = p + 1 \) and hence \( t_1 = k_1 \cdot d_{lp} = t_s = t_1 + k \cdot d_{lp} + d_{lb} - k_2 \cdot d_{lp} \) and hence \( d_{lb} = (k_2 - k_1 - k) \cdot d_{lp} \).

Next we will show that \( d_{lb} \) cannot be a integral multiple of \( d_{lp} \). Consider two consecutively indexed paths \( q_1 \) and \( q_2 \) in \( E_{lp} \) and a transitive edge \( e_a \) in \( q_1 \) as shown in figure 5.2-2 below:
Let \( u_{1a} - z_{1a} = k \) where \( k \) is some integer and hence \( w_{1a} - z_{1a} = k + 1 \). Let \( PA(v_u) = p \) and hence \( PA(v_w) = p \) and \( PA(v_z) = p + 1 \). Let \( TA(v_u) = t_u \), \( TA(v_w) = t_w \) and \( TA(v_z) = t_z \). In any syntactically correct mapping these times must be related by the following set of five equations:

a. \( t_z > t_u \)

b. \( t_w > t_z \)

c. \( t_z - t_u = k \cdot d_{1p} + d_{1b} \)

d. \( t_w - t_u = d_{1p} \)

e. \( t_w - t_z = (k + 1) \cdot d_{1p} - d_{1b} \)

Using the above equations we can show that \( k \cdot d_{1p} < d_{1b} < (k + 1) \cdot d_{1p} \) and hence \( d_{1b} \) cannot be an integral multiple of \( d_{1p} \).

Sufficiency:

Let \( \psi \) be the program transformed to \( G \). Let \(|A|=n \) and so the major paths in \( A \) are totally ordered with indices of the major paths in \( A \) ranging from \( \emptyset \) to \((n-1)\). Construct a linear-array \( L_A \) with \( \psi = \psi_p \), \( L_G = L_p \) and \(|PR|=n \) and construct a mapping as follows:

Set \( n_{1a} = \emptyset \) and \( n_{1b} = 1 \). For every \( r_{1p} \) in \( S_A \) set \( n_{1p} = a_{1p} \). If \( A = D_1 \) then set \( d_{1a} = 2 \) else set \( d_{1a} = 1 \). If the consistency constants of
all the relations in $S_b$ are $\not\prec \emptyset$ then set $d_{lb}=1$ else set $d_{lb}=d_{la}+l$ where $b_{lp}$ is the minimum among all the consistency constants of all the relations in $S_B$. For every $E_{lp}$ in $H$ set $d_{lp}=a_{lp}d_{lb}+b_{lp}d_{la}$. This completes assignment of $n_{lp}$ and $d_{lp}$ for every label $lp$ such that $E_{lp}$ is in $H$. Next for every $E_{lp}$ in $F$ if $E_{lp}$ in $H$ is identical to some $E_{lq}$ in $H$ then set $n_{lp}=n_{lq}$ and $d_{lp}=d_{lq}$. For every relation $E_{lp}$ in $S$ set $B_{lp}$ as empty. This completes assignment of $n_{lp}$ and $d_{lp}$ for every label in $G$. Next construct the functions $PA$ and $TA$. For every vertex $v_x$ in a path in $A$ whose index is $i$ set $PA(v_x)=i$. For every computation vertex $v_y$ set $TA(v_y) = t_0 + y_{lb}d_{lb} + y_{la}d_{la}$ where $t_0$ is the time at which the computation vertex whose co-ordinates are $<\emptyset, \emptyset>$ (i.e., the computation vertex on the major paths in $A$ and $B$ whose indices in the ordering of the major paths in $A$ and $B$ are $\emptyset$) is mapped. This completes the construction of the mapping. We next show that this mapping is syntactically correct.

$d_{la} \not\prec \emptyset$. Hence no two computation vertices in any major path in $A$ are simultaneously mapped onto the same processor. The computation vertices in distinct major paths in $A$ are mapped onto distinct processors. Hence no two computation vertices in $G$ are mapped simultaneously onto the same processor.

Next consider any two computation vertices $v_x$ and $v_y$ such that there is an edge labelled $lp$ directed from $v_x$ to $v_y$.

Now,

$$PA(v_y) = PA(v_x) + (y_{lb}-x_{lb})n_{lb} + (y_{la}-x_{la})n_{la}$$
$$= PA(v_x) + (y_{lb} - x_{lb})$$
$$= PA(v_x) + \Delta_{lp}$$
\[=PA(v_x) + n_{lp}\]

and \(n_{lp}\) is one of \(\{1, -1, 0\}\) as \(a_{lp}\) is one of \(\{1, -1, 0\}\).

Also
\[
\begin{align*}
    TA(v_y) &= TA(v_x) + (y_{lb} - x_{lb})^*a_{lp} + (y_{la} - x_{la})^*a_{la} \\
    &= TA(v_x) + a_{lp}^*d_{lb} + b_{lp}^*d_{la} \\
    &= TA(v_x) + d_{lp}
\end{align*}
\]

Next consider any two computation vertices \(v_x\) and \(v_y\) and any label \(lp\) such that \(y_{lb} - x_{lb} = k^*a_{lp}\) and \(y_{la} - x_{la} = k^*b_{lp}\). It can be verified that \(v_x \stackrel{k}{\leftarrow m, n} v_y\) where \(m = a_{lp}\) and \(n = d_{lp}\). If \(n_{lp} = 0\) then by condition (3) \(E_{lp}\) is identical to \(A\). In our construction no two distinct paths in \(A\) are mapped onto the same processor and hence no two distinct paths in \(E_{lp}\) are mapped onto the same processor. If \(n_{lp} \neq 0\) then by lemma 4.0-3 there must be a single path labelled \(l_{lp}\) passing through \(v_x\) and \(v_y\) which is condition (3). Hence no two source or sink vertices are simultaneously mapped onto the input port labelled \(lp\) of any processor in the array.

6. Semantic Issues

In the previous section we characterized a class \(CL\) of program graphs such that for every program graph in this class there exists a syntactically correct mapping. Recall that a syntactically correct mapping of a program graph \(G\) computes the program \(\Psi\) represented by \(G\) correctly iff the contents of every source and sink vertex in \(G\) that is mapped onto the input ports or output ports of processors more than once remains invariant. More formally in a syntactically correct mapping of \(G\) if there exists a pair of tuples \(v_{xm}, t_m\) and \(v_{xn}, t_n\) either in \(S_{C'}\) or \(S_{C'}\) then the contents of \(v_x\) at \(t_m\) must be the same as the contents of \(v_x\)
at $t_n$. For the contents to remain invariant the program $\psi$ represented by
the program graph $G$ which is the same as the function $\psi_P$ computed by
every processor in the linear array needs interpretation. However in a
syntactically correct mapping of $G$ if there does not exist any pair of
tuples $\langle v_x, t_m \rangle$ and $\langle v_x, t_n \rangle$ either in $S^*_G$ or $S^*_G$ such that $t_m \neq t_n$ then
clearly the program $\psi$ needs no interpretation. But we demonstrate in
theorem 6.1-1 that this sub-class of program graphs in CL is very
limited. We then examine some semantic properties required of programs
in CL for them to be correctly executable.

6.1. Uninterpreted Characterization

Herein we will examine the structure of program graphs in CL that can
be computed correctly without interpretation of the programs represented
by the program graphs. We establish a few preliminary results of a
general nature which we subsequently use to prove the main result,
.i.e., theorem 6.1-1.

**Lemma 6.1-1:**

If a syntactically correct mapping of $G$ (not necessarily in CL)
computes the program $\psi$ transformed to $G$ correctly without interpretation
of $\psi$ then for any label $l_p$ if $E_{l_p}$ is in $H$ and $n_{l_p} \neq 0$ then all the major
paths in $E_{l_p}$ must have the same path-length.

**Proof:** Let $q_1$ and $q_2$ be the two major paths in $E_{l_p}$ such that
the path-length of $q_1$ is greater than that of $q_2$. Let the path-
length of $q_1$ be $k$. Now $n_{l_p} \neq 0$ and hence there must be at least $k$
processors in the linear-array. Let $v_s$ and $v_f$ be the source and
sink vertices in \( q_2 \). Let \( v_x \) and \( v_y \) be the first\(^1\) and the last\(^2\) computation vertices respectively in \( q_2 \). Let \( PA(v_x)=p_1 \), \( PA(v_y)=p_2 \), \( TA(v_x)=t_1 \), \( TA(v_y)=t_2 \). Since the path-length of \( q_2 \) is less than the path-length of \( q_1 \) either one of the following cases must be true:

1. there exist at least two tuples \( <v_s,t_1-d_{1p}> \) and \( <v_s,t_1> \) in \( S_0G \) such that \( IOA<v_s,t_1-d_{1p}>=p_1 \) and \( IOA<v_s,t_1>=p_1-1 \)

2. there exist at least two tuples \( <v_f,t_2+d_{1p}> \) and \( <v_f,t_2+2d_{1p}> \) in \( SI_G \) such that \( IOA<v_f,t_2+d_{1p}>=p_2 \) and \( IOA<v_f,t_2+2d_{1p}>=p_2+1 \)

In both cases interpretation is needed.

Lemma 6.1-2:

If a syntactically correct mapping of \( G \) (not necessarily in \( CL \)) computes the program \( \Psi \) transformed to \( G \) correctly without interpretation of \( \Psi \) then for any pair of labels \( 1_p \) and \( 1_q \) if \( E_{1p} \) and \( E_{1q} \) are both in \( H \) then \( n_{1p}=1 \) \( \Rightarrow \) \( n_{1q}=\emptyset \) and \( n_{1q}=1 \) \( \Rightarrow \) \( n_{1p}=\emptyset \) (i.e., \( n_{1p} \) and \( n_{1q} \) cannot both be non-zero simultaneously).

Proof: Without loss of generality let \( n_{1p}=n_{1q}=1 \). By theorem 5.1-1 any maximally-connected subgraph \( SG \) with \( L_{SG}=(1_p,1_q) \) must be a mesh graph and hence the major paths labelled \( 1_p \) in \( SG \) are

\(^1\)the edge labelled \( 1_i \) from \( v_s \) is directed into \( v_x \)

\(^2\)the edge labelled \( 1_i \) from \( v_y \) is directed into \( v_f \)
totally ordered and similarly the major paths labelled 1q in SG are also totally ordered. Let \( q_r \) denote the major path with index \( \emptyset \) in \( E_{1p} \) and \( q_w \) be the major path with index \( \emptyset \) in \( E_{1q} \). Let \( v_x \) be a computation vertex in \( q_w \) and \( q_r \) as shown in figure 6.1-1 below:

![Figure 6.1-1](image)

Let \( v_y \) be a computation vertex in \( q_w \) and \( v_s \) a source vertex labelled \( 1p \). Let \( PA(v_x) = p \) and \( TA(v_x) = t \). \( n_{1p} = n_{1q} = 1 \) and hence \( PA(v_y) = p+1 \) and \( TA(v_y) = t + d_{1q} \). Consequently there must exist a tuple \( \langle v_s, t + d_{1q} \rangle \) and \( \langle v_s, t \rangle \) in \( SG \), such that \( IOA(v_s, t) = p \) and \( IOA(v_s, t + d_{1q}) = p+1 \) and hence interpretation is needed.

Let \( G \) be any program graph in \( CL \). Let \( SG \) be the only maximally-connected subgraph of \( G \) with \( L_{SG} = \{11, 1j\} \).

Lemma 6.1-3:

If a syntactically correct mapping of \( G \) in \( CL \) computes the program \( \psi \) transformed to \( G \) correctly without interpretation of \( \psi \) then there must be at most two sets of major paths in \( H \) (i.e., \( H \) can have atmost \( E_{11} \) and \( E_{1j} \)) and the path-lengths of the major paths in the same set must be identical to one another (i.e., the path-lengths of major paths in \( E_{11} \)
must be identical to one another and similarly the path-lengths of major paths in $E_{1j}$ must also be identical to one another).

Proof: We first show that $H$ must have at most $E_{1i}$ and $E_{1j}$. Suppose $H=\{E_{1i}, E_{1j}, E_{1p}\}$. By lemma 6.1-2 $n_{1i}$ and $n_{1j}$ cannot both be non-zero. Let $n_{1i} \neq 0$ and $n_{1j} = 0$. Also by lemma 6.1-2 $n_{1i}$ and $n_{1p}$ cannot both be non-zero and hence $n_{1p} = 0$. But by lemma 4.0-1 $n_{1j} = n_{1p} \neq 0$.

We next show that the path-lengths of the major paths in $E_{1i}$ are identical and the path lengths of the major paths in $E_{1j}$ are also identical. By lemma 6.1-2 $n_{1i}$ and $n_{1j}$ cannot both be non-zero. Without loss of generality let $n_{1i} \neq 0$ and so $n_{1j} = 0$. By lemma 6.1-1 the path lengths of the major paths in $E_{1i}$ are identical.

Suppose there exist major paths in $E_{1j}$ such that the path lengths are not same. Let $q_1$ and $q_2$ be two consecutively indexed major paths in the total ordering of the major paths in $E_{1j}$ whose path lengths differ. Without loss of generality let the path length of $q_1$ be greater than that of $q_2$ as shown in figure 6.1-2 below.

![Figure 6.1-2](image-url)
In figure 6.1-2 $v_f$ is a sink vertex labelled $11$ and $v_x$ and $v_y$ are computation vertices. Let $PA(v_x)=p$ and $TA(v_x)=t$. Now $n_{11}^0$ and $n_{1j}^0$. Without loss of generality let $n_{11}=1$ and hence $PA(v_y)=p+1$. Consequently there must exist two tuples $<v_f,t+d_{11}>$ and $<v_f,t+2*d_{11}>$ in $SI_G$ such that $IOA<v_f,t+d_{11}> = p$ and $IOA<v_f,t+2*d_{11}> = p+1$ and hence interpretation is needed.

Theorem 6.1-1:

A syntactically correct mapping of $G$ in CL computes the program $\psi$ transformed to $G$ correctly without interpretation of $\psi$ iff there is only one set of major paths in $E$ (i.e., $E$ is either $E_{11}$ or $E_{1j}$) and the path length of all the major paths in this set are identical to one another.

Proof: Necessity:

Suppose $H=E_{11}, E_{1j}$. Let $n_{11}=1$ and hence by lemma 6.1-2 $n_{1j}=0$. By lemma 6.1-1 the path lengths of all major paths in $E_{11}$ are the same and the path lengths of all major paths in $E_{1j}$ are the same. Now let $q_1$ and $q_2$ be two major paths in $E_{1j}$ indexed $\emptyset$ and $1$ respectively as shown in figure 6.2-2 below:

Figure 6.2-2
In figure 6.2-2 $v_x$ and $v_y$ are computation vertices. $v_w$ is a source vertex labelled $1j$. Let $PA(v_x)=p$ and $TA(v_x)=t$. Consequently $PA(v_y)=p+1$ and $TA(v_y)=t+d_{11}$. Now $t=t_s$ where $t_s$ is the start time. Hence $t+d_{11}>t_s$ and hence there must be at least the two tuples $<v_w, t_s>$ and $<v_w, t+d_{11}>$ in $SO'_G$ such that $IOA<v_w, t_s>=IOA<v_w, t+d_{11}>=p+1$ and this needs interpretation.

Sufficiency:

Let $\Psi$ be the program transformed to $G$. Assign unique indices to the major paths in $E_{1i}$. $(H=E_{1i})$ and hence there is no edge directed from a computation vertex in some major path in $E_{1i}$ to another computation vertex in some other distinct major path in $E_{1i}$ and hence the indices can be assigned unique indices arbitrarily). Let $E_{1i}={q_0, q_1, \ldots, q_m}$ where $q_i$ is the major path assigned index $i$. Let $v_{(r,w)}$ denote the computation vertex at a distance $w$ from the first computation vertex in the major path $q_r$. Let the the number of computation vertices in any major path be $n$. Construct a linear array LA with $\Psi=\Psi_p$, $|PR|_p=n$ and $L_G=L_p$. Construct a mapping $MP_G$ as follows:

For every $E_{lp}$ in $F$ set $n_{lp}=n_{1l}=1$. For every $E_{lp}$ in $S$ set $B_{lp}$ as empty. Define $PA(v_{(\emptyset, \emptyset)})=\emptyset$, $TA(v_{(\emptyset, \emptyset)})=t_\emptyset$, $PA(v_{(r,w)})=w$ and $TA(v_{(r,w)})=t_\emptyset+r+w$. It can be easily verified that such a mapping is syntactically correct and every tuple in $SO'_G$ and $SI'_G$ are mapped only onto the appropriate distinguished input ports and output ports.
6.2. Semantic Properties

From theorem 6.1-1 it is clear that the subclass of program graphs that can be computed without any semantic information is very limited. Herein we will examine three simple semantic properties of a program and demonstrate their significance in ensuring the correctness of computation of a syntactically correct mapping of any program graph in CL. The notations used are similar and hence have the same meaning as those used in definition 2.2-1 and definition 2.2-2 of a program and its transformation to a program graph.

Definition 6.2-1: \( \Psi_1 \) is an identity function iff for any \( x \) and \( z \) in \( Y \) if \( \Psi(x) = z \) then \( z_1 = x_1 \).

Example 6.2-1: In example 2.2-1 \( \Psi_2 \) and \( \Psi_3 \) are identity functions.

Example 6.2-2: Let \( W = (w_1, w_2, \ldots, w_{(k-1)}, w_{(i-1)}, w_{(i+1)}, \ldots, w_k) \) denote a \((k-1)\)-tuple where \( \nu \ j, 1 \leq j \leq k \) and \( j \neq i \), \( w_j \in Y_j \). \( W \) is an identity vector for \( \Psi_1 \) iff for any \( x_i \in Y_i \) and for any \( z \) in \( Y \) if \( \Psi(w_1, w_2, \ldots, w_{(i-1)}, x_i, w_{(i+1)}, \ldots, w_k) = z \) then \( z_1 = x_1 \).

Example 6.2-3: In example 2.2-1 \( w = (\emptyset, \emptyset) \) is an identity vector for \( 1 \).

Definition 6.2-2: Let \( w_i \in Y_i \), \( w_i \) is a fixpoint element for \( \Psi_1 \) iff for any \( z \) in \( Y \), any \( x_j \) in \( Y_j \) and \( \nu \ j \), \( 1 \leq j \leq k \) and \( j \neq i \), if \( \Psi(x_1, x_2, \ldots, x_{(i-1)}, w_i, x_{(i+1)}, \ldots, x_k) = z \) then \( z_1 = w_1 \).

Example 6.2-4: Let \( Y = Y_1 \times Y_2 \). For any \( x \) and \( z \) in \( Y \) if \( \Psi(x) = z \) then let \( z_1 = \min(x_1, x_2) \) and \( z_2 = \max(x_1, x_2) \).
\( \pm \infty \) is a fixpoint element for \( \min \) and \(-\infty\) is a fixpoint element for \( \max \).

Let \( \text{ID} = \bigcup_{1}^{\text{Y}_1} \bigcup_{1}^{\text{Y}_2} \ldots \bigcup_{1}^{\text{Y}_k} \) where \( \text{ID}_i \subseteq \text{Y}_i \) for \( 1 \leq i \leq k \). \( \text{ID}_i \) contains elements that are fixpoints of the function \( \psi_i \) or some elements from \( \text{Y}_i \) that are components of an identity vector for some \( \psi_j \), \( 1 \leq j \leq k \) and \( j \neq i \).

Let \( \text{MID} \) be a function from \( \text{ID} \) to a subset of \( T \) (\( T \) has the same significance as used in definition 3.0-1\(^1\)) such that if \( \text{MID}(x) = t \) and if \( x \) is in \( \text{ID}_i \) then the element \( x \) is mapped onto the distinguished input port for label \( l_i \) at time \( t \).

**Lemma 6.2-1**: If \( \text{MID}(x) = t \) and \( x \) is in \( \text{ID}_i \) and \( x \) is a fixpoint element of \( \psi_i \) then the element at the input port of processor indexed \( p+k*n_{l_i} \) at \( t+k*n_{d_{l_i}} \) is \( x \) where \( p \) is the processor index whose input port labelled \( l_i \) is the distinguished input port for label \( l_i \).

A transformation of a program \( \varphi \) to a program graph assigns to every element in each of the sets \( \text{Y}_1, \text{Y}_2, \ldots, \text{Y}_k \) either a labelled edge or a labelled source vertex or a labelled sink vertex. Consider a syntactically correct mapping of a program graph \( G \). For every source or sink vertex \( v_x \) let \( S_x = \langle v_x, t \rangle \mid \langle v_x, t \rangle \in \Sigma_G \) or \( \langle v_x, t \rangle \in \Sigma_0 \). In every \( S_x \) there is a tuple \( \langle v_x, t_{\text{max}} \rangle \) such that for every other tuple \( \langle v_x, t \rangle \) in \( S_x \) \( t < t_{\text{max}} \). Let \( \text{IOA}(v_x, t) = p \). Let \( \text{ID}(x, t) \) denote the elements at the other input ports of \( p \) at time \( t \). Let \( \text{ID}_x = \{ \text{ID}(x, t) \} \)

\(^1\) page 26
Theorem 6.2-1: A syntactically correct mapping of a program graph $G$ computes correctly if for any label $l_1$ one of the following holds:

1. $\psi_1$ is an identity function

2. for any source or sink vertex labelled the element in $Y_1$ that has been transformed to $v_x$ is a fixpoint for $\psi_1$

3. every $ID_{<x,t}$ in $ID_x$ is an identity vector for $\psi_1$

7. Illustration

In this section we illustrate the syntactic and semantic characterization developed in the previous sections by two examples. In the first example we consider the problem of multiplying a band matrix by a vector. We will use our characterization to synthesize the solution to this problem proposed in [5]. In the second example we consider the problem of sorting and we will again use our characterization to synthesize the rebound sorter [1].

Example 7.0-1: Multiplication of a Band-Matrix by a Vector

Herein we will use matrix $M$ and vector $X$ used in example 2.2-1. For the program graph (figure 2.2-2) in example 2.2-1 $H=\{E_{11}, E_{13}\}$, $F=\emptyset$ and $S=\{E_{12}\}$. The number of maximally-connected subgraphs $SG$ with $L_{SG}=\{11,13\}$ is 1 and hence the program graph is in $CL$. Augment $SG$ by the set $D_{1}$ of diagonal paths. The resulting augmented program graph is shown in figure 7.0-1 below:
In figure 7.0-1 the dashed lines denote the diagonal paths. $D_1 = \{A_0, A_1, A_2, A_3\}$ and $E_{11} = \{B_0, B_1, B_2, B_3, B_4, B_5\}$. Let $A = D_1$ and hence $B = E_{11}$. So $S_A = \{E_{11}, E_{13}\}$ and $S_B = \{D_1, E_{13}\}$. It can be verified that this program graph satisfies theorem 5.2-1. The consistency constants $a_{11}$ and $a_{13}$ for $E_{11}$ and $E_{13}$ in $S_A$ are 1 and -1 respectively. The consistency constants $b_{11}$ and $b_{13}$ for $D_1$ and $E_{13}$ in $S_B$ are both 1.

We demonstrate a mapping of this program graph using the construction used in the sufficiency of theorem 5.2-1. $n_{1a} = \emptyset$ and $n_{1b} = 1$. $B$ is $E_{11}$ and hence $n_{11} = 1$. Set $n_{13} = a_{13} = 1$. $A$ is $D_1$ and hence set $d_{1a} = 2$. Set $d_{1b} = 1$ and hence $d_{11} = 1$. $d_{13} = d_{1b}a_{13} + d_{1a}b_{13}$ and hence $d_{13} = 1$. Set $t_0 = \emptyset$. Map every computation vertex in $A_1$ and $B_j$ onto processor $i$ at time $i + 2j$. The mapping is shown in figure 7.0-2 below:
In Figure 7.0-2 we have omitted most of the source and sink vertices from the program graph for clarity. $I_{11}$ and $O_{11}$ are the distinguished input port and output port for label 11, $I_{12}$ and $O_{12}$ are the distinguished input port and output port respectively for label 12. and $I_{13}$ and $O_{13}$ are the distinguished input ports and output ports respectively for label 13. All the computation vertices in any shaded box are mapped onto the same processor to which the shaded box is connected. The time at which a computation vertex is mapped is shown by the side of the computation vertex. $v_{s3}$ is the only source vertex labelled 12 that gets mapped onto the input ports labelled 12 more than once. However, $v_3$ is an identity function and hence the contents of $v_{s3}$ remains invariant till it reaches processor 2. $v_{s1}$, $v_{s2}$, $v_{f1}$ and $v_{f2}$ are the only source and sink vertices labelled 11 that get mapped onto input ports labelled 11 of processors more than once. Now $v_1$ has the
vector \( \langle x, \emptyset \rangle \) as its identity vector where \( x \) can be any element from \( Y_3 \). Map the element \( \emptyset \) onto port \( I_{12} \) of processor \( \emptyset \) at times \( \emptyset \) and 2 and onto port \( I_{12} \) of processor 1 at time 1. This ensures that the contents of \( v_{s1} \) and \( v_{s2} \) remain invariant till it reaches processor 2 and 1 respectively. Similarly map the element \( \emptyset \) onto the input port \( I_{12} \) of processor 2 at time 12 and input port \( I_{12} \) of processor 3 at times 11 and 13. This ensures that the contents of \( v_{f1} \) and \( v_{f2} \) remain invariant till they reach \( 0_{11} \).

Example 7.0-2: Sorting

We wish to sort the set of elements \( \{2, 10, 5, 6\} \). A program graph that performs sorting is shown in figure 7.0-3 below:

![Program Graph for Sorting](image)

Figure 7.0-3: Program Graph for Sorting

In figure 7.0-3 'o' denote computation vertices and '\( 0 \)' denote either source or sink vertices. Each computation vertex computes the minimum and maximum of the two input elements denoted by the incoming horizontal and vertical edges. The outgoing horizontal and vertical edges denote the minimum and maximum respectively of the two input elements computed by the computation vertex. The horizontal edges are labelled 11 and the
source and sink vertices connected to horizontal edges are all labelled 11. The vertical edges are labelled 12 and the source and sink vertices connected to vertical edges are all labelled 12. The set of horizontal source vertices is \{ih_1, ih_2, ih_3, ih_4\} and the set of horizontal sink vertices is \{oh_1, oh_2, oh_3, oh_4\}. Similarly the set of vertical source vertices is \{iv_1, iv_2, iv_3, iv_4\} and the set of vertical sink vertices is \{ov_1, ov_2, ov_3, ov_4\}. The source vertices iv_1, iv_2, iv_3 and iv_4 are initialized to 2, 1\emptyset, 6 and 5 respectively. The source vertices ih_1, ih_2, ih_3 and ih_4 are all initialized to \infty. It can be verified that oh_1, oh_2, oh_3 and oh_4 are 2, 5, 6 and 1\emptyset respectively.

\(E_{11}=\{\text{set of horizontal paths}\}\) and \(E_{12}=\{\text{set of vertical paths}\}\). \(H=\{E_{11}, E_{12}\}, F=\{\emptyset\}\) and \(S=\{\emptyset\}\). The number of maximally-connected subgraphs SG with \(L_{SG}=\{11, 12\}\) is 1 and hence the program graph is in \(CL\). Augment SG by the set \(D_1\) of diagonal paths. The resulting program graph is shown in figure 7.\(\emptyset-4\) below:

![Figure 7.\(\emptyset-4\): Sorting Graph](image-url)

In figure 7.\(\emptyset-4\) the dashed lines denote the diagonal paths. \(D_1=\{A_\emptyset, A_1, A_2, A_3\}\) and \(E_{11}=\{B_\emptyset, B_1, B_2, B_3\}\). Let \(A=D_1\) and hence \(B=E_{11}\). So
$S_A = \{E_{11}, E_{12}\}$ and $S_B = \{D_1, E_{12}\}$. It can be verified that this program graph satisfies theorem 5.2-1. The consistency constants $a_{11}$ and $a_{12}$ for $E_{11}$ and $E_{12}$ in $S_A$ are 1 and -1 respectively. The consistency constants $b_{11}$ and $b_{12}$ for $D_1$ and $E_{12}$ in $S_B$ are both 1.

We next construct a mapping of this program graph. Set $n_{1a} = 0$ and $n_{1b} = 1$. $B$ is $E_{11}$ and hence $n_{11} = 1$. Set $n_{12} = a_{12} = 1$. $A$ is $D_1$ and hence set $d_{1a} = 2$. Set $d_{1b} = 1$ and hence $d_{11} = 1$. $d_{12} = d_{1b} \cdot a_{12} + d_{1a} \cdot b_{12}$ and hence $d_{12} = 1$. Set $t_0 = 7$. Map every computation vertex in $A_i$ and $B_j$ onto processor $i$ at time $7 + i + 2 \cdot j$. The mapping is shown in figure 7.0-5 below:

![Figure 7.0-5](image)

The time at which a computation vertex is mapped onto a processor is shown by the side of the computation vertex. We have to ensure that the contents of $iv_1$, $iv_2$ and $iv_3$ remain unchanged till they reach processors $\emptyset$, 1 and 2 respectively. $\psi_1$ is the min function and $\psi_2$ is the max function. $\infty$ is the identity element for 1 and $-\infty$ is the identity
element for max function. Map $-\infty$ onto $I_{11}$ at times 1, 3 and 5. Similarly to ensure invariance of the contents of $oh_1$, $oh_2$ and $oh_3$ map $\infty$ onto $I_{12}$ at times 12, 14 and 16. This mapping is a synthesis of the rebound sorter [1].

8. Conclusions

In this paper we characterized a class of uniform graphs that are correctly computable on a model of linear array processors for VLSI. We illustrated our characterization by synthesizing published algorithms for two important computational problems. Obviously not all program graphs belong to this class. In [7] the characterization of arbitrary program graphs is being explored.
REFERENCES

[1] Chen, T.C., Lum, V.Y. and Tung, C.
The Rebound Sorter: An Efficient Sort Engine for Large Files.
In Proceedings of the 4th International Conference on Very Large

A Survey of Highly Parallel Computing.

Structure Of Parallel Algorithms.
Technical Report CMU-CS-79-143, computer science

Technical Report CMU-CS-79-151, Computer Science Dept., Carnegie--

Systolic Arrays for VLSI.
Technical Report CMU-CS-79-103, Computer Science Dept., Carnegie--

Why Systolic Architectures.

[7] Ramakrishnan, I.V.
Characterization of Programs Computable on a Model of VLSI Array
Processors (in preparation).

[8] Sutherland, I.E. and Mead, C.A.
Microelectronics and Computer Science.