Towards a Single Criterion for Identifying Program Unstructuredness*

T.H. Tse
Department of Computer Science
The University of Hong Kong
Pokfulam, Hong Kong
Email: thtse@cs.hku.hk

ABSTRACT

We introduce the concepts of fully embedded skeletons and partially overlapping skeletons in program flowgraphs. We show that only one simple criterion is necessary and sufficient for the identification of program unstructuredness. Namely, a program flowgraph is unstructured if and only if it contains partially overlapping skeletons.

1. INTRODUCTION

In [1], we proposed a formal approach for studying the properties of program flowgraphs. We found two conditions for the identification of unstructuredness. In this paper, we shall extend our theory to include fully embedded skeletons and partially overlapping skeletons. We will show that only one simple criterion is necessary and sufficient for the identification of unstructuredness in program flowgraphs.

2. DEFINING UNSTRUCTUREDNESS

A program flowgraph is defined as unstructured if and only if it contains at least one of the following:

(a) An Entry in the Middle of a Selection

Given a condition node $n$, the module $M_n$ is said to be a selection module if and only if neither of its branches** $B^+(n)$ or $B^-(n)$ contains $n$. A node $m$ in $M_n$ is defined as an entry node if and only if there exists some node $p$ outside $M_n$ such that $m$ is a successor of $p$, i.e., $m = s^p_p(p)$. A node $m$ is said to be an entry in the middle of a selection if and only if $m \neq n$ but is an entry node of a selection module $M_n$.

* Part of this research was done at the London School of Economics, University of London under a Commonwealth Academic Staff Scholarship. It was also supported in part by a University of Hong Kong Research Grant.

** The notations in this paper will follow those of [1].
(b) An Entry in the Middle of an Iteration

Given a condition node \( n \), the module \( M_n \) is said to be an iteration module if and only if one of the branches \( B_\alpha(n) \) contains \( n \). Given an iteration module \( M_n \), we define an entry in the middle of an iteration as an entry node \( m \) such that

(i) \( m \neq n \) or \( s_\alpha(n) \) or \( s_{-\alpha}(n) \), or

(ii) there exists some other entry node \( \neq m \).

(c) An Exit in the Middle of a Selection

A selection module \( M_n \) is said to have an exit in the middle if and only if its branches \( B_\alpha(n) \) and \( B_{-\alpha}(n) \) have a non-empty intersection.

(d) Multiple Exits in an Iteration

A decision node \( m \) is defined as an exit of an iteration module \( M_n \) if and only if:

(i) \( M_m = M_n \);

(ii) \( m \) is in one of the branches \( B_\gamma(m) \) but not in the opposite branch \( B_{-\gamma}(m) \).

An iteration module is said to have multiple exits if and only if it has more than one exit.

3. PARTIALLY OVERLAPPING SKELETONS

In [1], we attempted to relate program unstructuredness to the properties of skeletons in flowgraphs. We will extend our theory further in this paper. A skeleton \( q_\delta(v) \) is said to be fully embedded in another skeleton \( q_\gamma(u) \) if and only if \( q_\gamma(u) \) contains \( v \) as well as all the nodes of \( q_\delta(v) \). Two skeletons \( q_\gamma(u) \) and \( q_\delta(v) \) are said to be partially overlapping if and only if they are not fully embedded in one another but contain at least one common node \( m \) not equal to \( u \) or \( v \).

Lemma 3.1

A skeleton \( q_\delta(v) \) is fully embedded in another skeleton \( q_\gamma(u) \) if and only if \( q_\gamma(u) \) contains both \( v \) and \( s_\delta(v) \).

Proof:

If \( q_\delta(v) \) is fully embedded in \( q_\gamma(u) \), then the latter will of course contain both \( v \) and \( s_\delta(v) \). Conversely, suppose \( q_\gamma(u) \) contains both \( v \) and \( s_\delta(v) \). Then, there exists a sequence of nodes \( \langle w_0, w_1, ..., w_r \rangle \) such that

\[ w_0 = s_\gamma(u); \]
\[ w_i = s_\gamma(w_{i-1}) \text{ for } i = 1, 2, ..., r \text{ (if } s_\gamma(u) \neq s_\delta(v)); \]
\[ w_r = s_\delta(v). \]

For any node \( m \neq s_\delta(v) \) in \( q_\delta(v) \), there exists a sequence of nodes \( \langle w_r, w_{r+1}, ..., w_t \rangle \) such that

\[ w_r = s_\delta(v); \]
\[ w_i = s_\delta(w_{i-1}) \text{ for } i = r+1, r+2, ..., t; \]
\[ w_t = m. \]
Furthermore, \( m \leq s_{\gamma}(v) \leq s_{\gamma}(u) \). Hence, any \( m \) in \( q_{\delta}(v) \) must lie in \( q_{\gamma}(u) \). □

**Lemma 3.2**

If \( m \) is an entry node of \( M_{n'} \), then there exists a node \( u \) outside \( M_n \) such that \( m \) is in \( q_{\gamma}(u) \). Furthermore,

(a) \( m = s_{\gamma}(u) \), or
(b) \( m = s_{\gamma}(v) \) for some node \( v \) in \( q_{\gamma}(u) \) but outside \( M_n' \).

**Proof:**

Suppose \( m \) is an entry node of \( M_{n'} \). Then, there exists some node \( u_0 \) outside \( M_n \) such that \( m = s_{\gamma}(u_0) \). If \( u_0 \) is a condition node, then (a) follows immediately. If, on the other hand, \( u_0 \) is an action node, then \( s_{\gamma}(u_0) = m \), and there exists a node \( u_1 \) outside \( M_n \) such that one of its skeletons \( q_{\delta}(u_1) \) contains \( u_0' \). If \( s_{\gamma}(m) \leq s_{\gamma}(u_1) \), then (b) will follow. Otherwise, \( s_{\gamma}(u_1) = m \), and there exists another node \( u_2 \) outside \( M_n \) such that one of its skeletons \( q_{\gamma}(u_2) \) contains \( u_1 \). Proceeding in this way, since the program flowgraph is finite, we shall arrive at some \( u_r \) outside \( M_n \) such that one of its skeletons \( q_{\gamma}(u_r) \) contains \( m \), and \( m = s_{\gamma}(u_{r-1}) \) for some node \( u_{r-1} \) in \( q_{\gamma}(u_r) \) but outside \( M_n' \). □

We can now derive the main theorems of the paper, thus connecting unstructuredness with partially overlapping skeletons.

**Theorem 3.3**

A program flowgraph is unstructured if there exist partially overlapping skeletons.

**Proof:**

Suppose the skeletons \( q_{\gamma}(u) \) and \( q_{\gamma}(v) \) partially overlap at the node \( m \). By Lemma 3.1, we have two cases:

(a) \( v \notin q_{\gamma}(u) \). Suppose \( M_v \) is an iteration module. Since there exists an elementary path from \( m \) to \textbf{end} not passing through \( v \), by Lemma 5.1 of [1], there must be another exit. On the other hand, suppose \( M_v \) is a selection module. Since there exists an elementary path from \( u \notin M_v \) to \( m \in M_v \) not passing through \( v \), we must have an entry in the middle of a selection. Hence, we have unstructuredness in either case.

(b) \( v \in q_{\gamma}(u) \) and \( s_{\gamma}(v) \notin q_{\gamma}(u) \). Let \( w \) be the first node in \( q_{\gamma}(v) \) such that it is also in \( q_{\gamma}(u) \). Then, \( w \) is an entry node for \( M_v \). However, since \( w \) cannot be \( v \), \( s_{\gamma}(v) \) or \( s_{-\gamma}(v) \), it must be an entry in the middle of a selection or an entry in the middle of an iteration. In either case, we have unstructuredness. □

We will prove that the converse of the theorem is also true, resulting in a necessary and sufficient condition for unstructuredness:

**Theorem 3.4**

A program flowgraph is unstructured if and only if there exist partially overlapping skeletons.
Proof:

Suppose a program flowgraph is unstructured. There are three possible anomalies:

(a) An Entry in the Middle of a Selection or Iteration

Let \( m \) be an entry in the middle of a module \( M_n \). By Lemma 3.2, there exists a node \( u \) outside \( M_n \) such that one of its skeletons \( q(u) \) contains \( m \). Take the smallest submodule \( M \) in \( M_n \) such that \( m \) is also an entry in the middle of \( M \). We have four cases:

(i) \( m = s_\delta(v) \) and there is some other entry node \( p = s_{-\delta}(v) \). Assume that there is no partially overlapping skeleton. Then, \( v, s_\delta(v) \) and \( s_{-\delta}(v) \) are in \( q(u) \). Since \( s_{-\delta}(v) \leq v \) and \( s_{-\delta}(v) \leq v \), we must have either \( s_\delta(v) \leq s_{-\delta}(v) \leq v \), or \( s_{-\delta}(v) \leq s_\delta(v) \leq v \). Hence, all paths from \( s_\delta(v) \) to \( s_{-\delta}(v) \) (or vice versa) do not pass through \( v \), which is clearly a contradiction. Thus, we must have partially overlapping skeletons.

(ii) \( m = v \). Again, assume that there is no partially overlapping skeleton. Then, \( q(u) \) should contain both \( v \) and \( s_\delta(v) \). However, since \( s_{-\delta}(v) \leq v \), \( q(u) \) must be of the form \( \langle ..., s_\delta(v), ..., v, ... \rangle \). By definition, there exists a sequence of nodes \( \langle w_0, w_1, ..., w_t \rangle \) such that

\[
\begin{align*}
  w_0 &= s_\delta(v); \\
  w_i &= s_\delta(w_{i-1}) \quad \text{for } i = 1, 2, ..., t; \\
  w_t &= v = m.
\end{align*}
\]

Furthermore, all of these nodes are in both \( q(u) \) and \( M \). This contradicts the fact that, by Lemma 3.2, \( m = s_\gamma(w) \) for some node \( w \) in \( q(u) \) but outside \( M_n \). Hence, we must have partially overlapping skeletons.

(iii) \( m \neq v \) or \( s_\delta(v) \) or \( s_{-\delta}(v) \), and \( m \) is in \( q_\delta(v) \). Then, there exists a node \( w \) in \( q_\delta(v) \) such that \( m = s_\gamma(w) \). Assume that there is no partially overlapping skeleton. Then, both \( m \) and \( w \) are in \( q_\delta(v) \). The skeleton \( q_\delta(v) \) must therefore be of the form \( \langle ..., w, m, ... \rangle \). This contradicts the fact that \( m = s_\gamma(w) \) for some node \( w \) in \( q_\delta(v) \) but outside \( M_n \). Hence, we must have partially overlapping skeletons.

(iv) \( m \) is not in \( q_\delta(v) \). By definition, there exists a node \( w \) in \( M \) such that \( m \) is in \( q_\delta(w) \). Then, \( M \) contains two distinct entries: \( w \) and \( m \). This contradicts the fact that \( M \) is the smallest submodule in \( M_n \) such that \( m \) is an entry in the middle of \( M \). Hence, we must have partially overlapping skeletons.

(b) An Exit in the Middle of a Selection

Since \( B_\alpha(n) \) and \( B_{-\alpha}(n) \) have a non-empty intersection, there exists \( u \) in \( B_\alpha(n) \) and \( v \) in \( B_{-\alpha}(n) \) such that \( q_u(u) \) and \( q_v(v) \) contain a common node \( m \). By definition, we have partially overlapping skeletons.

(c) Multiple Exits in an Iteration

Let \( M_n \) be the iteration module and suppose that there are more than one exit. We have two cases:

(i) \( n \) is an entry node of \( M_n \). Then, there exists a node \( u \) outside \( M_n \) such that one of its skeletons \( q_u(u) \) contains \( n \). On the other hand, by Corollary 5.6 of [1], \( n \) is in neither of its own skeletons. Therefore, there exists another node \( v (\neq n) \) in \( M_n \) such that one of its skeletons \( q_v(v) \) contains \( n \). By the definition of skeletons, \( s_\gamma(n) \leq s_\gamma(v) \).
However, since \( v \) is in \( M_n \), \( s_q(v) \leq s_q(n) \). Hence, \( s_q(v) = s_q(n) \), and so \( v \) and \( n \) cannot both be in \( q_\gamma(u) \). Thus, we must have two partially overlapping skeletons.

(ii) \( n \) is not an entry node of \( M_n \). Unless we are having an entry in the middle of an iteration, the entry node must be \( s_\alpha(n) \). That is to say, there exists a node \( u \) outside \( M_n \) such that one of its skeletons \( q_\gamma(u) \) contains \( s_\alpha(n) \). If \( q_\gamma(u) \) does not contain \( n \), then we have two partially overlapping skeletons. If \( q_\gamma(u) \) contains \( n \), then, by arguments similar to (i), we also have two partially overlapping skeletons. □

Following the line of [2], fully embedded skeletons and partially overlapping skeletons can be located in \( O(N) \) time, where \( N \) is the total number of nodes. Hence, unstructuredness in program flowgraphs can be identified in \( O(N) \) time.

4. CONCLUSION

We have introduced the concepts of fully embedded skeletons and partially overlapping skeletons in program flowgraphs. We have shown that only one simple criterion is necessary and sufficient for the identification of program unstructuredness. Namely, a program flowgraph is unstructured if and only if it contains partially overlapping skeletons.

ACKNOWLEDGEMENTS

The author is grateful to R.K. Stamper of the London School of Economics, University of London and M.Y. Chan of the University of Hong Kong for their invaluable comments and suggestions.

REFERENCES
