The Graphs of Structured Programming

Lucila Maria de Souza Bento\textsuperscript{a}, Davidson Rodrigo Boccardo\textsuperscript{b}, Raphael Carlos Santos Machado\textsuperscript{b}, Vinícius Gusmão Pereira de Sá\textsuperscript{a}, Jayme Luiz Szwarcfiter\textsuperscript{a}

\textsuperscript{a} Universidade Federal do Rio de Janeiro — Brasil
\textsuperscript{b} Instituto Nacional de Metrologia, Qualidade e Tecnologia — Brasil

Abstract

Control flow graphs represent the possible execution paths of a program and can be obtained by static analysis of software binaries. We give a formal characterization of the subclass of control flow graphs that correspond to structured code.

Keywords: structured code; control flow graph; graph classes; characterization

1 Introduction

Structured programming can be viewed as a paradigm for describing and implementing algorithms and computer programs. Roughly speaking, it consists of a top-down formulation in which the algorithm is broken into blocks. Such technique constrains the implementation to a set of allowed statements, supporting basically three control structures: \textit{sequence}, \textit{selection} and \textit{iteration}.

An early and important step towards structured programming was the seminal article by Dijkstra, “Go-to statement considered harmful” \cite{4}, followed by an extensive discussion in the literature (e.g., \cite{8,9,13}). The influence of

\textsuperscript{*} Work partially supported by Eletrobrás Distribuição Rondônia, DR/069/2012.
that new programming paradigm could also be noticed from the very outset on the development of algorithms, either explicitly, as in Henderson and Snowdon [7] and Knuth and Szwarcfiter [10], or implicitly, as in various graph algorithms by Tarjan, e.g. [12]. The main guidelines of structured programming can be found in [5].

A natural problem regarding structured programming is to recognize, directly from the program’s binary code, whether a given program is structured. A possible approach is the use of the control flow graph of the program [1,3], which is a directed graph representing the possible sequences of instructions along the execution of a program and can be obtained by static analysis tools after the disassembly of the code. Our problem then becomes a graph problem: given the control flow graph corresponding to some computer program, decide whether the program has been written obeying the rules of structured programming. Of course there are characterizations [6] and efficient recognition algorithms [11] for the well-known class of reducible graphs, a class which contains all control flow graphs for structured programs. However, the former class is much larger than the latter, and to our knowledge no attempt has been made so far to fully characterize and recognize the latter class.

This work brings about the latest results of the research these authors have been conducting with aims at solving software security problems through applications of graph theory and graph algorithms (see, for instance, [2]). We study the class of control flow graphs which correspond to structured programs. We give both a characterization of this class and a polynomial-time recognition algorithm, which is able to identify the maximal unstructured subgraph of the given input. In fact, we define two classes of graphs: the first one corresponds to classic, “pure” structured programs, restricted to the use of sequence, selection and iteration statements; the second one is more general and allows for two kinds of premature interruption of iterations (widely known as break and continue statements), besides the use of selection statements with divergent exits.

All graphs in this paper are directed. For a graph $G$, denote its vertex and edge sets by $V(G)$ and $E(G)$, respectively. For $v, w \in V(G)$, an edge from $v$ to $w$ is represented by $vw$. We say $vw$ is an out-edge of $v$ and an in-edge of $w$, and we let $N^+_G(v) = \{w \in V(G) \mid vw \in E(G)\}$ and $N^-_G(v) = \{w \in V(G) \mid wv \in E(G)\}$. A trivial graph contains solely one vertex. For vertices $v, w$ of a graph $G$, we say $v$ reaches $w$ when there is a path in $G$ from $v$ to $w$. A source of $G$ is a vertex that reaches all other vertices. A control flow graph $G$ contains at least one source, which is denoted $s(G)$. A vertex that reaches no other vertex is a sink of $G$. We represent by $T(G)$ the set of sinks of $G$. 
2 Structured program graphs

A statement graph is a directed graph $H$ satisfying the following:

(i) each vertex is labeled as either a regular ($R$) or an expansion ($X$) vertex;

(ii) $H$ belongs to one of the following classes:
   a) sequence graph (Fig. 1.1);
   b) if graph (Fig. 1.2), convergent-if-then-else graph (Fig. 1.3) and divergent-
      if-then-else graph (Fig. 1.6), also referred to as selection graphs;
   c) while graph (Fig. 1.4) and repeat graph (Fig. 1.5), also referred to as
      iteration graphs;
   d) break-while graph (Fig. 1.7) and break-repeat graph (Fig. 1.8);
   e) continue-while graph (Fig. 1.9) and continue-repeat graph (Fig. 1.10).

In particular, if $G$ is a statement graph, the selected source $s(G)$ corre-

sponds to a vertex labeled $v_1$ in Fig. 1, and is called the head of $G$, while each

sink is called a tail of $G$.

Let $G, H$ be vertex-disjoint flow graphs, and $v \in V(G)$, such that $|N_G^+(v)| = |T(H)| = 1$, or $|N_G^+(v)| = 0$ and $|T(H)| \geq 1$. The expansion of $v$ into $H$ is the

operation that replaces $v$ with $H$ in $G$, i.e., it constructs a new graph $G'$ in

Fig. 1. Statement graphs
the following way:

1. \( V(G') = [V(G) \setminus \{v\}] \cup V(H) \setminus T(H) \)
2. \( N_G^-(s(H)) = N_G^-(v) \)
3. If \(|N_G^+(v)| = 1\), let \(N_G^+(v) = \{z\}\), and identify the unique sink \(t(H)\) of \(H\)
4. \( N_G^+(t(H)) = N_G^+(v) \), for \(t(H) \in T(H)\)
5. \( N_G^-(w) = N_H^-(w) \setminus \{v\} \) and \(N_G^+(w) = N_H^+(w) \setminus \{v\} \), for \(w \in V(G') \setminus [V(H) \cup N_G^+(v)]\)
6. \( N_G^-(w) = N_H^-(w) \) and \(N_G^+(w) = N_H^+(w) \), for \(w \in V(H) \setminus [\{s(H) \cup T(H)\}]\)

Similarly, for a flow graph \(G'\) and a flow subgraph \(H\) of \(G'\), the contraction of \(H\) is the operation that contracts \(H\) into a single vertex \(v\).

**Definition 2.1** A structured program (SP) (respectively, extended structured program (ESP)) graph is recursively defined as a flow graph in which each vertex is either a regular \((R)\) or an expansion \((X)\) vertex, and such that a trivial graph is an SP (resp., ESP) graph, and any graph obtained from an SP (resp., ESP) graph by expanding an \(X\)-vertex into a statement graph from Fig. 1(1-5) (resp., from Fig. 1(1-10)) is also an SP (resp., a ESP) graph.

An induced subgraph \(H\) of a graph \(G\) is called basic if (i) \(H\) is isomorphic to some statement graph, and (ii) whenever \(H\) is the divergent-if-then-else graph, the sinks of \(H\) are also sinks of \(G\). We also say \(H\) is closed if (i) \(v \in v(H), v \notin \{s(H)\} \cup T(H) \Rightarrow N_H^+(v) = N_G^+(v) \) and \(N_H^-(v) = N_G^-(v)\); and (ii) \(N_H^+(s(H)) = N_G^+(s(H))\). A graph is called prime when it is non-trivial, basic and closed.

Note that when expanding some expansion vertex \(v\) into a non-trivial SP (ESP) graph, new expanding vertices are introduced. Note also that the expansion of \(v\) into a trivial vertex must change the label of \(v\) from \(X\) to \(R\). The following theorem characterizes SP (ESP) graphs.

**Theorem 2.2** A graph \(G\) is an SP (respectively, ESP) graph if and only if there is a sequence of graphs \(G_0, \ldots, G_k\), such that (i) \(G_0\) is the trivial graph, (ii) \(G_k = G\), and (iii) \(G_i\) is obtained from \(G_{i-1}\) by expanding some \(X\)-vertex of \(G_{i-1}\) into a statement graph whose vertices are labeled as in Fig. 1(1-5) (resp., as in Fig. 1(1-10)).

### 3 Recognition algorithm

Let \(\mathcal{H}(G)\) be the collection of prime subgraphs of graph \(G\). For \(H \in \mathcal{H}(G)\), denote by \(G[H \downarrow]\) the graph obtained from \(G\) by contracting \(H\). Call \(H\)
the contracting prime of $G$. Let $H, H' \in \mathcal{H}$, $H \neq H'$, and $vw$ an edge of $H'$. The image of $vw$ in $G[H \downarrow]$, denoted $I_{G[H \downarrow]}(vw)$, is the ordered pair of vertices of $G[H \downarrow]$ defined as follows: $I_{G[H \downarrow]}(vw) = vw$, if $vw \in E(G[H \downarrow])$; otherwise, $I_{G[H \downarrow]}(vw)$ equals either $s(H)w$ or $vt(H)$, according to whether or not $w \in V(G[H \downarrow])$. The image of any edge $vw \in E(H')$ is unique and can always be determined this way. More generally, the image of $H'$ in $G[H \downarrow]$ is the subgraph of $G[H \downarrow]$ defined by the subset of edges $\{e' \in E(G[H \downarrow]) \mid e' \in I_{G[H \downarrow]}(e), e \in E(H')\}$. Extending the definition, let $I_{G[H \downarrow]}(H) = \emptyset$.

Two prime subgraphs $H, H'$ of $G$ are independent when either $V(H) \cap V(H') = \emptyset$ or $V(H) \cap V(H') = \{v\}$, where $v$ is either the common tail of $H$ and $H'$, or $v$ is the tail of $H$ and the head of $H'$.

**Lemma 3.1** Let $G$ be an arbitrary graph and $H, H' \in \mathcal{H}(G)$, $H \neq H'$. Then $I_{G[H \downarrow]}(H') \in \mathcal{H}(G[H \downarrow])$, and $G[H \downarrow][I_{G[H \downarrow]}(H') \downarrow] = G[H' \downarrow][I_{G[H \downarrow]}(H) \downarrow]$.

A sequence of graphs $G_0, \ldots, G_k$ is a contractible sequence of $G$ when $G = G_0$, and $G_{i+1} = G_i[H_i \downarrow]$, for some prime subgraph $H_i \in \mathcal{H}(G_i)$, $i < k$. The contractible sequence $G_0, \ldots, G_k$ is maximal when $G_k$ does not contain prime subgraphs. In particular, if $G_k$ is the trivial graph then the sequence $G_0, \ldots, G_k$ is maximal. Let $S \equiv G_0, \ldots, G_k$ be a contractible sequence. Denote by $H_j$ the contracting prime of $G_j$, $j < k$. The concept of the image of a prime $H' \in \mathcal{H}(G_j)$ in $G_{j+1}$ is extended to any graph $G_q$ of $S$, $j + 1 \leq q \leq k$, as follows: $I_{G_q}(H') = I_{G_p}(I_{G_{q-1}}(H'))$, for $j + 1 \leq p \leq q$. The subgraph $I_{G_q}(H')$ is the image of $H'$ in $G_q$.

**Theorem 3.2** Let $G$ be an arbitrary graph, with $S \equiv G_0, \ldots, G_k$ and $S' \equiv G'_0, \ldots, G'_k$, two maximal contractible sequences of $G$. Then, $G_k$ and $G'_k$ are isomorphic. Furthermore, $k = k'$.

**Algorithm 1** Recognizing (extended) structured graphs

```
unmark all vertices of $G$
repeat
  choose any unmarked vertex $v \in V(G)$ and mark it
  if $v$ is the head of some prime subgraph $H$ of $G$ then
    contract $H$ into $v$
    unmark all marked vertices of the remaining graph
  if $G$ is the trivial graph then
  stop: $G$ is an SP graph
until all vertices are marked
stop: $G$ is not an SP graph
```
Corollary 3.3 Let $G$ be an arbitrary graph and $G_0, \ldots, G_k$ any contractible sequence of it. Then $G$ is an SP graph if and only if $G_k$ is a trivial graph.

Given the characterization of the class (Theorem 2.2) and the uniqueness of maximal contractible sequences (Theorem 3.2), the recognition algorithm becomes straightforward (see Algorithm 1). Observe that there can be can be $O(n)$ prime subgraphs. Each time one such subgraph is identified, all marked vertices that still remain in the graph become unmarked, therefore any vertex can be unmarked at most $O(n)$ times. In addition, verifying whether a vertex $v$ is the head of some prime subgraph can be done in $O(1)$ time. The overall complexity of the algorithm is thus $O(n^2)$.

References

Appendix: omitted proofs

Theorem 2.2

Proof: Suppose $G$ is an SP (ESP) graph. It follows from its definition that $G$ can be constructed by starting from a trivial graph $G_0$, whose vertex is labelled $X$, and iteratively replacing expansion vertices by statement graphs. Let $k$ be the number of iterations (expansions) performed in the construction of $G$.

At iteration $i$, denote by $G_i$ the graph obtained from $G_{i-1}$ by expanding some expansion vertex of $G_{i-1}$ into a statement graph $0 \leq i \leq k$. Then $G_k = G$ and the sequence $G_0, \ldots, G_k$ satisfies the conditions of the theorem.

Conversely, suppose that there exists a sequence of graphs $G_0, \ldots, G_k$ satisfying the conditions (i)-(iii) of the theorem. Conditions (i) and (iii) suffice to show that $G_i$ is an SP (ESP) graph, for all $0 \leq i \leq k$, and condition (ii) implies $G$ is an SP (ESP) graph. □

Lemma 3.1

Proof: First, observe that, since $H, H'$ are closed, $H_1 \cap H_2$ can only contain heads and tails of $H$ and $H'$. Again, because $H, H'$ are closed, we know that $N_{H_1}(s(H_1)) = N_{G}(s(H_1))$, and $N_{H_2}(s(H_1)) = N_{G}(s(H_2))$.

Now examine the alternatives for $|H \cap H'|$. If $|H \cap H'| \leq 1$, the above discussion implies that $H, H'$ must be independent. If $|H \cap H'| = 0$, $H, H'$ are disjoint, implying that $I_{G[H]}(H') = H'$ and $I_{G[H']}(H) = H$. Then (i) and (ii) are clearly valid. Consider $|H \cap H'| = \{v\}$. Since $H, H'$ have distinct heads, it follows that $v$ is either the common tail of $H, H'$ or the head of one of them, coinciding with the (unique) tail of the other one. In the first case, $H$ is replaced by edge $s(H_1)v$ in $G[H \downarrow]$, preserving $H'$ as a prime subgraph. If $v$ is the head of $H$ and the tail of $H'$, then $H$ is replaced by edge $vt(H)$, again preserving $H'$ as a prime subgraph of $G[H \downarrow]$. Finally, when $v$ is the tail of $H$ and the head of $H'$, it follows that $H$ becomes the edge $vt(H)$ in $G[H \downarrow]$, maintaining $H'$ as a prime subgraph once again. Consequently, $H' \in \mathcal{H}(H)$ in all cases, and (i) holds. To show that (ii) is true for $|H \cap H'| \leq 1$, proceed as follows. If $|H \cap H'| = 0$, then in both $G[H \downarrow][I_{G[H]}(H') \downarrow]$ and $G[H' \downarrow][I_{G[H']}\downarrow(H) \downarrow]$, the primes $H$ and $H'$ are replaced by the non-adjacent edges $s(H)t(H)$ and $s(H')t(H')$, respectively, implying that (ii) holds. When $H \cap H' = \{v\}$, again consider the above three possible alternatives. If $v$ is the common tail of $H$ and $H'$, then these subgraphs are contracted into edges $s(H)v$ and $s(H')v$, respectively, in both subgraphs $G[H \downarrow][I_{G[H]}(H') \downarrow]$ and
implying that (ii) holds. If $v$ is the head of $H$ and tail of $H'$, then, after the two contractions, $H$ and $H'$ become edges $vt(H)$ and $s(H')v$, respectively, also implying (ii). The case when $v$ is the tail of $H$ and head of $H'$ is analogous.

Finally, consider the alternative $|H \cap H'| > 1$, i.e., $|H \cap H'| = 2$. The only possibility here is that $H, H'$ are both sequence graphs, whereupon $H \cup H'$ induces a closed directed $P_4$ in $G$. Let $v_1, v_2, v_3, v_4$ be such a path and $v_1$ the head of $H$. Then $V(H) = \{v_1, v_2, v_3\}$ and $V(H') = \{v_2, v_3, v_4\}$. It follows that $G[H \downarrow]$ reduces $H$ to edge $v_1v_3$, and consequently $H'$ is reduced to $v_3v_4$. Thus, $I_{G[H \downarrow]}(H')$ is a closed $P_3$, namely $v_1, v_3, v_4$. The latter graph is clearly a prime of $G[H \downarrow]$, that is, $I_{G[H \downarrow]}(H') \in \mathcal{H}(G[H \downarrow])$, and (i) holds. To prove (ii), observe that $I_{G[H \downarrow]}(H)$ is a closed $P_3$ with vertices $v_1, v_2, v_4$, meaning that $H \cup H'$ becomes edge $v_1v_4$ in both $I_{G[H \downarrow]}(H')[H \downarrow]$ and $I_{G[H \downarrow]}(H)[H \downarrow]$. Therefore, $G[H \downarrow][I_{G[H \downarrow]}(H') \downarrow] = G[H \downarrow][I_{G[H \downarrow]}(H) \downarrow]$. □

**Theorem 3.2**

*Proof:* Without loss of generality, assume $k \leq k'$.

Denote by $H_i$ and $H'_i$ the contracting primes of $G_j$ and $G'_j$, respectively. That is, $G_{j+1} = G_j[H_j \downarrow]$ and $G'_{j+1} = G'_j[H'_j \downarrow]$. Let $i < k$ be the largest index such that $S$ and $S'$ coincide up to it. That is, $G_j = G'_j$, for $0 \leq j \leq i$, but $G_{i+1} \neq G'_{i+1}$. Since $G_0 = G'_0$, such an index exists, and we have $G_{i+1} = G_i[H_i \downarrow], G'_{i+1} = G'_i[H'_i \downarrow]$, and $H_i \neq H'_i$. By Lemma 2(i), $I_{G_{i+1}}(H_i) = G'_{i+1}$, and $\emptyset \neq I_{G'_j}(H_i) \in \mathcal{H}(G'_p)$, as long as $I_{G'_j}(H_i)$ does not become the contracting prime of $G'_p$, for some $p \geq i + 1$. Since $S$ is maximal, by Lemma 2, there must exist some index $\ell \geq i + 1$ such that $H'_\ell = I_{G'_\ell}(H_i)$. That is, the contracting prime of $G'_{\ell}$ is precisely the image of $H_i$ in $G'_{\ell}$. By Lemma 2(ii), we can swap the corresponding images of the contracting primes of $G'_{\ell-1}$ and $G'_{\ell}$, while preserving $G'_{\ell+1}$, i.e.,

$$G'_{\ell-1}[H'_{\ell-1} \downarrow][I_{G'_{\ell-1}[H'_{\ell-1}]}(H_i) \downarrow] = G'_{\ell-1}[I_{G'_{\ell}}(H_i) \downarrow][I_{G'_{\ell-1}[I_{G'_{\ell}}(H_i)]}(H'_\ell \downarrow)].$$

Define new contractible sequences $S(j), i \leq j \leq k - 1$, as follows. First, let $S(i) \equiv S$, and define $S(i + 1)$ initially coinciding with $S(i)$. Now let $S(i + 1)$, the corresponding image of $H_i$, become the contracting prime of $G'_{\ell-1}$, instead of $G'_{\ell}$, while the image of $H'_\ell$ becomes the contracting prime of $G'_{\ell-2}$. That is, the images of $H_i$ and $H'_\ell$ swap positions in $S(i - 1)$, implying that the graphs $G'_{\ell-1}$ and $G'_{\ell}$ are modified. However, by the above equality,
the graph $G'_{i+1}$ is preserved, and consequently all graphs $G'_j$, $j \geq \ell + 1$, are also preserved. However, the image of $H_i$ shifted one position to the left. Iteratively, proceed exchanging positions between the image corresponding to $H_i$ and the contractible prime on its left, until eventually $H_i$ itself reaches index $i$, that is, $H_i$ becomes the contractible prime of $G'_i$. Hence the new graph of $S(i+1)$ at index $i+1$ becomes $G'_i \downarrow H_i$, which is precisely the graph $G_i$. Consequently, $S$ and $S(i+1)$ coincide up to index $i+1$, while preserving the terminal graphs $G_k$ and $G'_k$. By iterating such a process for $i + 1, i + 2, \ldots, k - 1$, we conclude that $S(k-1)$ coincides with $S$ up to index $k - 1$. Let $G'_{k-1}(k-1)$ denote the graph of $S(k-1)$ at index $k - 1$. Then $G'_{k'-1}(k-1) = G_{k-1}$. However, $H_{k-1}$ is the unique prime of $G_{k-1}$, and $G_{k-1}[H_{k-1} \downarrow] = G_k$. Consequently, the same must hold for $G'_{k-1}(k-1)$, implying that $G_k = G'_k$ and $k = k'$. $\square$