Generalising Control Dependence

Sebastian Danicic[a] Richard W. Barraclough[b] Mark Harman[c] Ákos Kiss[d] Michael R. Laurence[e]

[a] *Dept. of Computing, Goldsmiths College, University of London, London SE14 6NW, UK.*

[b] *@UK PLC 5 Jupiter House, Calleva Park, Aldermaston, Reading, Berkshire, RG7 8NN, UK*

[c] *CREST, Software Systems Engineering Group, Department of Computer Science, University College London, Malet Place, London, WC1E 6BT, UK.*

[d] *Dept. of Software Engineering, Institute of Informatics, University of Szeged, 6720 Szeged, Hungary.*

[e] *Dept. of Computer Science, The University of Sheffield, Western Bank, Sheffield S10 2TN, UK.*

Abstract

Weak and strong projection, properties which capture the underlying semantics of control dependence, are defined. Weak and strong commitment-closedness, generalisations of non–termination sensitive and insensitive control dependence to arbitrary finite directed graphs, are introduced and shown to satisfy these desirable semantic properties. Low polynomial–time algorithms for computing these generalised forms of control dependence are given.

Our formulation is attractively simple and, because of its generality, widely applicable to both existing and future notions of control dependence. To demonstrate this, it is proved that all published forms of control dependence can be implemented using weak or strong commitment-closedness, thereby satisfying either the weak or strong semantics. A by-product of this research has, thus, been to classify all previous forms of control dependence into just two: weak and strong.

1 Introduction

Control dependence is the relationship that exists between two vertices of a control flow graph (CFG) representing a program when one vertex determines whether or not the other can be executed. Informally, vertex v is said to control vertex w if v computes a value which determines whether w is executed or avoided. This fundamental concept in program analysis has been studied since the 1970s [16] and yet still produces new and surprising results. For example, recently [3,29,30], it was demonstrated that standard definitions of control dependence [17], in use for over two decades, were unsuitable for capturing control dependence in a wide class of reactive systems.

Control dependence is central to many program analysis and transformation techniques. For instance, it underpins work on program slicing [15,19,22,33], goto elimination [28] and compiler optimisations [17]. This paper focuses on the use of control dependence in program slicing [11,21,32], though the definitions and results concerning control dependence that the paper introduces also apply to other applications. The aim of program slicing is, given a chosen set of variables and chosen points in the program, to find a set of all statements which may affect the values of the variables at those points.

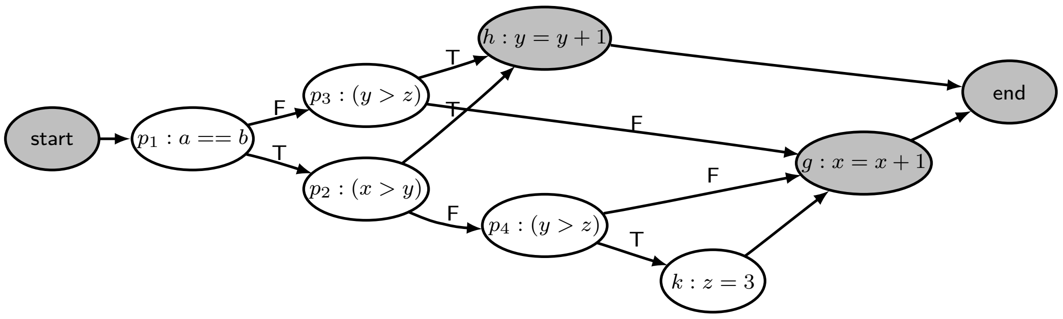
Slicing algorithms conventionally use two relations between statements in a program, or, more precisely, vertices in its control flow graph (CFG). These are *data dependence* and *control dependence*. Statement s is data dependent on statement t if t assigns a value to a variable v, say, which is referenced in s and there is a path from t to s with no intervening assignments to v.

Control dependence in program slicing can be understood by considering the program fragment represented by graph $G_{1(a)}$ in Figure 1. Suppose we are interested in finding out which program statements contribute to the final values of variables x and y. The set of vertices of interest is therefore $\{g, h\}$ because these are the only vertices that correspond to statements that change the values of x and y. The vertices $\{\mathsf{start}, \mathsf{end}\}$ are also added since these are traditionally required in a CFG. This gives a starting set $\{g, h, \mathsf{start}, \mathsf{end}\}$.

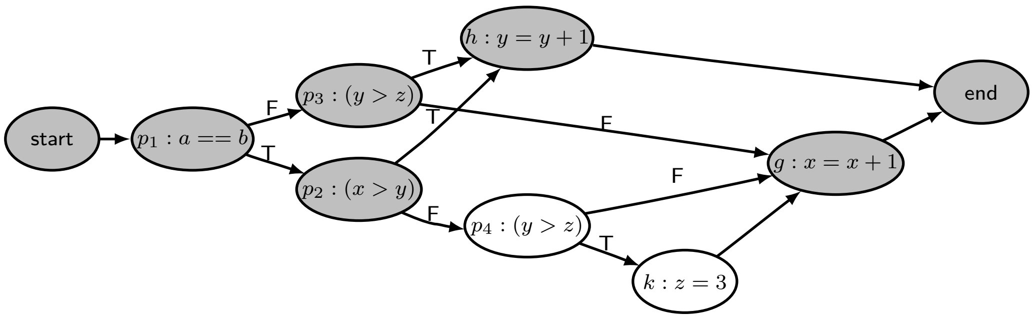
We now see that predicates p_2 and p_3 both *control* which of g and h are executed, and in turn p_1 controls which of p_2 and p_3 is executed. To compute the vertices which control $\{g, h\}$, we thus build up a *closure* to finally arrive at the set $\{\mathsf{start}, g, h, p_2, p_3, p_1, \mathsf{end}\}$ which is *closed* under control dependence. These vertices are then reconnected to produce the slice $G_{1(c)}$. Vertices p_4 and k have been 'sliced away' and a new edge from p_2 to g has appeared.

Control dependence has a long history, throughout which authors have sought to capture the property for certain classes of program graphs of interest. The first authors to consider control dependence are widely regarded to be Denning and Denning in their seminal work on secure information flow [16], a topic which remains highly relevant to this day.

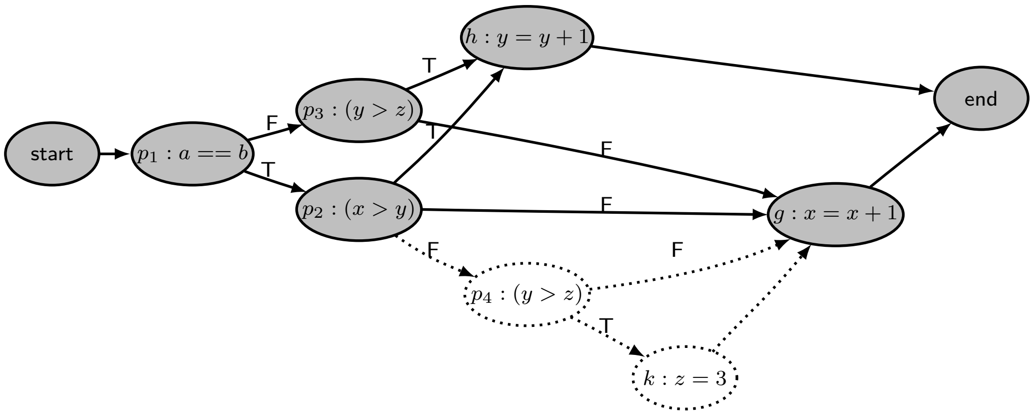
Weiser [34], was the first to express the Dennings' concept graph–theoretically in order to support slice construction. Subsequently, Ottenstein and Ottenstein [26] showed how Weiser's slicing could be captured as a graph reachability problem. Ferrante et al. [17] further developed these notions into a formal



(a) To slice $G_{1(a)}$ with respect to $\{\mathsf{start}, h, g, \mathsf{end}\}$...



(b) ...first, the statements (vertices) in the slice are computed: p_2 and p_3 are added because they control h and g and then p_1 is added because it controls p_2 and p_3. This gives the set $\{\mathsf{start}, p_1, p_2, p_3, h, g, \mathsf{end}\}$ which is closed under control dependence.



(c) These statements are then rewired to produce the slice $G_{1(c)}$ above. Vertices p_4 and k have been 'sliced away' and a new edge from p_2 to g has appeared.

Fig. 1. The use of control dependence in program slicing

characterisation of the program dependence graph [1].

In the 1990s, a generalisation of control dependence was defined by Bilardi and Pingali [6]. This generalisation is achieved by abstracting the notion of dominance to any set of paths. Generalised control dependence is thus 'parameterised' by this set. Instantiating different sets of paths yields different forms of control dependence. This, in effect, provides a framework for ex-

[1] Similar ideas are mentioned in [2,25,31,35].

pressing different forms of control dependence. The value of the framework was demonstrated by using it to express those forms of control dependence known at the time including the weak control dependence of Podgurski and Clarke [27].

The program dependence graph, later extended to handle procedures as the System Dependence Graph [23], has formed the basis of many analyses, such as program slicing, since its introduction in the 1980s. However, more recently, building on the work of Podgurski and Clarke [27], Ranganath et al. [29,30] and Amtoft [3] developed new notions of control dependence for reactive systems. Ranganath et al. [30] showed that the definitions used up until then were inadequate to handle (increasingly prevalent) reactive systems, in which programs react to inputs continuously without termination. Such reactive programs are deliberately written to non–terminate. These programs, thus, have graphs that contain vertices from which `end` is not reachable.

The wide range of applications of control dependence and its fundamental nature make it attractive to seek a simple, general characterisation that captures all previous definitions and which is also readily understood and from which it is easy to prove results for application areas. This paper introduces semantic definitions of control dependence, using a simple graph theoretic formulation, unhindered by specific restrictions on graph properties, such as constraints on the connectivity of the graph or presence or absence of certain structural features, such as special vertices.

1.1 Contributions of this paper

In this paper we develop a coherent theory of control dependence. There are four main contributions:

(1) We give a semantics for non–termination insensitive and non–termination sensitive control dependence by defining properties which must exist between graphs and graphs induced by subsets closed under control dependence in its different forms. We call these properties *weak* and *strong* projection.
(2) We introduce weak and strong commitment-closedness: generalisations of non–termination insensitive and non–termination sensitive control dependence which we prove satisfy our semantics. Unlike conventional control dependence, these are defined not just for traditional control flow graphs but also for more general structures.
(3) In program slicing, we require slices to be as small as possible. With this in mind, we give low order polynomial worst–case time complexity algorithms for computing the unique minimal weak and strong commitment-

closed sets. These algorithms are functionally equivalent to previous ones but more generally applicable.

(4) We believe that weak and strong projection capture the essence of control dependence. We demonstrate this by showing that all forms of control dependence in the literature can be implemented using weak and strong commitment-closedness. In so doing we provide a classification of all previous forms of control dependence into just two: weak and strong.

1.2 Overview of the paper structure and results

This section provides an overview of the technical contributions of the paper and the structure within which they are presented in the remainder of the paper.

In Section 2, we define a generalised form of CFG upon which the rest of our theory is built. Our CFGs are finite, directed, labelled graphs. They are simple in the sense that we do not need variables, assignments and expressions (these are only needed for data–dependence). The edges are labelled with subsets of $\{\mathsf{T}, \mathsf{F}\}$. Our CFGs are deterministic in the sense that edges from the same vertex must be disjointly labelled. Many of the definitions of control dependence in the literature impose constraints upon the types of graph for which they are defined. Our graphs encompass all those previously considered in the literature. They need not have a special end vertex which, when reached, represents successful termination. If it exists, the end vertex need not be reachable from all vertices. We allow all vertices to have out–degree zero and we allow predicates to have out–degree one. Leaving such 'incomplete' vertices corresponds to our program failing which we think of as reaching a state of silent non–termination. Intuitively, this can be thought of as the program appearing to do nothing but never returning to the operating system prompt. Incomplete vertices give rise to finite yet 'non–terminating' paths. Using the language of process algebra [24], we imagine the program infinitely engaging in (silent) τ–actions after reaching an incomplete vertex. It will be shown that graphs with incomplete vertices arise naturally in constructing minimal strong projections from deliberately non–terminating CFGs. Finally, our CFGs do not require a special start vertex.

We define a number of useful graph–theoretic concepts including V'–intervals and V'–paths, where V' is a set of vertices. A V'–interval is a path whose initial and final elements are both in V' but the intermediate ones are not. A V'–path is a path of length at least two whose final element lies in V', its first element may be in V', but none of its intermediate elements are in V'.

In Section 3, we describe the 'rewiring' problem: given a CFG G and a subset V'

of the vertices of G (representing the statements in the slice of G), how do we connect the elements of V' and relabel the new edges in a 'sensible' way? We call this the graph *induced* by V' from G. Rewiring is achieved by connecting vertices v_1 and v_2 in G' if and only if there is a V'–interval connecting them [2] in G. Edge labels are formed by taking the union of 'corresponding' edges in the original.

In Section 4, we define weak and strong projections. These are semantic relations which exist between graphs and the graphs induced from them by subsets of vertices closed under termination insensitive and termination sensitive control dependence respectively.

These projections are defined in terms of *walks*. A walk is very similar to a path, but elements which are predicates also include the boolean value in $\{\mathsf{T}, \mathsf{F}\}$ representing the 'choice' that was taken at that predicate. A weak projection is a graph where every walk of the original, when restricted to the vertices of the projection, is a walk of the projection. This is analogous to the situation that arises in conventional slicing where if the original executes n times a statement that is also in the slice then the slice, when executed from the same initial state, also executes the statement n times. As is well known, in conventional slicing we may execute this statement more times in the slice than in the original program because, for example, non–terminating loops may have been sliced away. This is also the case with weak projections.

The graph induced from a CFG G by V' is not necessarily, itself, a CFG. In general, it may contain non–predicate vertices of out–degree greater than one, predicates of out–degree greater than two, or non–disjoint edge labellings. We prove (Proposition 18) that being a weak projection is no more than a by–product resulting from ensuring that the induced graph is indeed a CFG. In other words, if the induced graph from a CFG is a CFG then it must be a weak projection of the original too.

Weak projection captures the behaviour of slices produced using the weak forms of control dependence (see Figure 2.) A *stronger* semantics is, however, required for slices produced using the strong forms of control dependence. With this aim, in Section 4.3, we define a strong projection. In the case of strong projection, every *maximal* walk of the original, when restricted to the vertices of the projection, gives rise to a *maximal* walk of the projection. Moreover, every walk of the projection arises in this way. For strong projection the number of times a walk passes through a vertex in the projection must be *equal* to the number of times the corresponding walk in the original passes though the vertex.

[2] This is not the first paper to define rewiring in this way. Earlier work [1,20] uses very similar definitions.

Weak Control Dependence	Strong Control Dependence
$\xrightarrow{\text{W-controls}}$ [34]	
$\xrightarrow{\text{F-controls}}$ [17]	
$\xrightarrow{\text{WOD}}$ [3]	
	$\xrightarrow{\text{PC-weak}}$ [27]
	$\xrightarrow{\text{NTSCD}}$ and $\xrightarrow{\text{DOD}}$ [30]

Fig. 2. Table showing the different forms of weak and strong control dependence.

In Section 4.4, we observe that both weak and strong projections either may or may not preserve the termination conditions of the original. This is also true of the so–called non-termination sensitive control dependence of Ranganath et al. [30]. Termination and walk preservation are orthogonal conditions. The weak projection of a CFG G may terminate when G does not. Strong projections, on the other hand, are always non-termination preserving. If a weak or strong projection of a CFG contains end then it termination preserving. Strong projections containing end of a CFG G, thus, perfectly preserve the termination and non-termination of G.

In Section 5, we develop a theory of *weak and strong commitment–closedness*, generalisations of non-termination sensitive and insensitive control dependence. We prove that these are properties of vertex sets which are necessary and sufficient to induce graphs which are weak and strong projections. Weak and strong commitment–closedness are used both in the production of algorithms for producing minimal weak and strong projections and also in the proofs that classify the previous forms of control dependence as either weak or strong.

In Sections 5.1 and 5.2, we define weak and strong commitment–closedness. In Section 5.3, we investigate graphs induced by weakly commitment–closed sets. The main result of Section 5.3 is Theorem 41 which states that the following three statements are all equivalent:

- The graph induced by V' from G is a CFG.
- V' is weakly commitment–closed in G.
- The graph induced by V' from G is a weak projection of G.

In Section 5.4, we investigate graphs induced by strongly commitment–closed sets. The main result of Section 5.4 is Theorem 45 which states that the graph induced by V' from G is a strong projection if and only if V' is strongly commitment–closed in G.

In Section 5.5, we prove Theorems 50 and 54 which state that given any set V' of vertices in a CFG, there are unique minimal weakly and strongly commitment–closed sets containing V'. These are the sets that are closed under control dependence which are required for slicing. This proves that for any vertex subset V' of CFG G, unique minimal weak and strong projections (slices) containing V' exist.

In Section 6, algorithms for computing minimal weak and strongly commitment–closed sets containing V' are defined and proved correct. This demonstrates that minimal weak and strong projections (slices) containing V' are computable. Furthermore, we show that these algorithms have worst–case time complexity $O(|G|^4)$, where $|G| = |V| + |E|$. For CFGs, since the maximum out–degree is two, we have $O(|G|) = O(|V|)$, giving a worst–case time complexity of $O(|V|^4)$. This is of a very similar order to the worst–case time complexity of the algorithms for computing the new control dependences of Ranganath et al. which is $O(|V|^4 log|V|)$. It is likely that the efficiency of these algorithms can be improved, but this is a topic for future work and is beyond the scope of this paper.

In Section 7, we categorise the weak forms of control dependence in the literature.

They are:

- $\xrightarrow{\text{W-controls}}$: the control dependence of Weiser [34],
- $\xrightarrow{\text{F-controls}}$: the control dependence of Ferrante et al. [17] and
- $\xrightarrow{\text{WOD}}$: the weak order dependence of Amtoft [3].

The results of this section give the relationship between sets closed under the weak forms of control dependence mentioned above and weakly commitment–closed sets.

From these, we prove our main result, Theorem 62, which states that, indeed, all weak forms of control dependence in the literature induce weak projections.

In Section 8, we categorise the strong forms of control dependence in the literature. We call them *strong* because, as we show in this section, vertex sets closed under them induce strong projections. They are:

- the combination of $\xrightarrow{\text{NTSCD}}$ and $\xrightarrow{\text{DOD}}$ of Ranganath et al. [30].
- $\xrightarrow{\text{PC-weak}}$, the weak control dependence of Podgurski and Clarke [27].

The results of this section give the relationship between sets closed under the strong forms of control dependence mentioned above and strongly commitment–closed sets. From these, we can prove our main result of this section, Theorem 80, which shows that, indeed, both strong forms of control dependence in the literature induce strong projections. Sections 7 and 8, as well as semantically characterising current forms of control dependence, justify our claim that weak and strong projection capture the essence of control dependence.

In Section 9, we conclude and give directions for future work.

2 Generalised CFGs for control dependence

Our graphs, since we are exploring only control dependence, do not need the variables, assignments and expressions which would be required for data–dependence. We can therefore have a very simple definition of a CFG. Our CFGs are finite directed graphs whose vertices are either predicates or non–predicates. Non–predicates have out–degree of at most one and predicates of out–degree of at most two. The edges emerging from predicates are labelled with subsets of $\{\mathsf{T}, \mathsf{F}\}$. This allows for a predicate with one edge labelled $\{\mathsf{T}, \mathsf{F}\}$ to represent a predicate where both branches go to the same destination. The labellings of the edges from each predicate must be disjoint. In other words, our CFGs are deterministic.

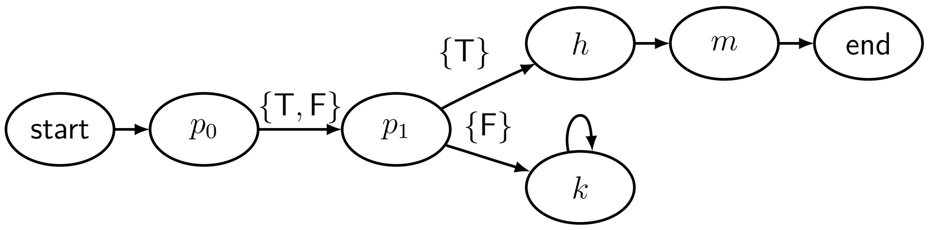
There is at most one special non–predicate vertex called end of out–degree zero. Reaching end corresponds to termination. Unlike conventional CFGs, we allow other non–predicates to have out–degree zero and predicates (incomplete) to have edges whose union of labels is not $\{\mathsf{T}, \mathsf{F}\}$. Reaching such vertices corresponds to a program silently non–terminating. We imagine programs which reach such vertices apparently not performing any actions but also not returning to the 'operating system prompt'. This situation arises naturally when slicing away infinite loops when preserving termination properties. Unlike, conventional CFGs we do not insist on a special start vertex.

Definition 1 (CFGs) *A* control flow graph *(*CFG*) is a triple $G = (V, E, \beta)$ where (V, E) is a finite directed graph and the vertex set V is partitioned as $V = P \cup N$ (predicates and non–predicates) with $P \cap N = \emptyset$, and $\beta : E \to \mathcal{P}(\{\mathsf{T}, \mathsf{F}\})$ is the edge labelling function.*

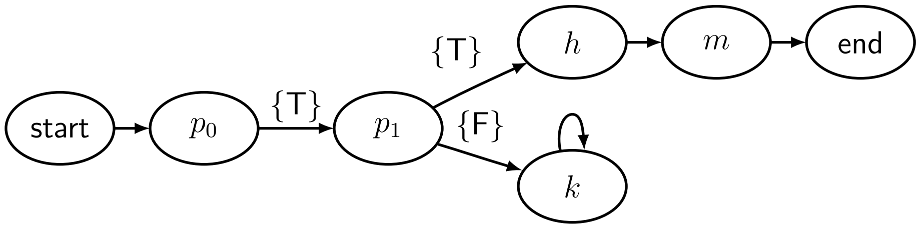
(1)
- *If $x \in P$ then the out–degree of x is at most 2.*
- *If $x \in N$ then the out–degree of x is at most 1.*
- *There is at most one end vertex. It has out–degree 0. ($\mathsf{end} \in N$ is the only vertex which represents normal termination.)*

(2) The edges are labelled by β where:
- *If $x \in P$ and $(x, y) \in E$ then $\beta(x, y) \neq \emptyset$.*
- *If $x \in N$ and $(x, y) \in E$ then $\beta(x, y) = \emptyset$.*
  (For clarity we omit the label $\emptyset$ from our diagrams.)

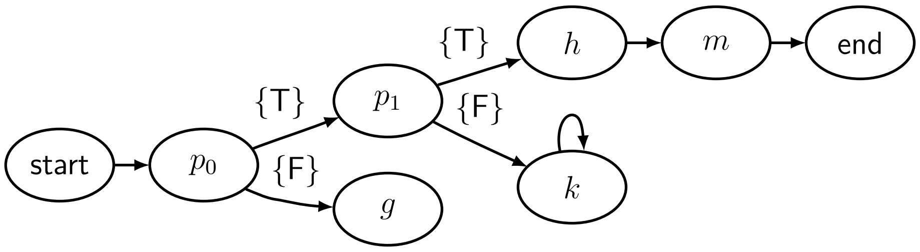
(3) Let p be a predicate. If $(p, y) \in E$ and $(p, z) \in E$ with $y \neq z$ then



(a) $G_{3(a)}$: a CFG with a complete predicate p_0 of out–degree 1.



(b) $G_{3(b)}$: A CFG with an incomplete predicate p_0.



(c) $G_{3(c)}$: a CFG with a final non–end, non–predicate vertex g.

Fig. 3. Examples of CFGs: In (a) the predicate p_0, although of out–degree 1 is complete. Its successor is independent of evaluating p_0. In (b) the predicates are $\{p_0, p_1\}$. Vertex p_0 is incomplete since the union of the labels of its branches is not $\{\mathsf{T}, \mathsf{F}\}$. If predicate p_0 evaluates to F, a state representing silent non–termination is reached. In (c) the predicates are $\{p_0, p_1\}$ and both are complete. The other non–end vertices are non–predicates and have out–degree 1 except for g which is a final non–end vertex. After executing g, again all programs represented by $G_{3(c)}$ are deemed to silently non–terminate.

$\beta(p, y) \cap \beta(p, z) = \emptyset$. *(In other words, our* CFG*s are deterministic.)*

See Figure 3 for examples of CFGs.

Definition 2 (Complete predicates of a CFG) *A predicate is complete if and only if the union of the labels of its outgoing edges is* $\{\mathsf{T}, \mathsf{F}\}$.

Definition 3 (Complete CFGs) *A* CFG *is complete if and only if all its predicates are complete.*

Definition 4 (Final vertices of a CFG) *A final vertex is either a non–predicate vertex of out–degree 0 or an incomplete predicate.*

2.1 Specialised classes of graphs and CFGs

The previous forms of control dependence variously make references to a unique start or end of the program or CFG. The following definition gives notation for the required classes of specialised graphs and CFGs.

Definition 5 ($\{\mathsf{start}, \mathsf{end}\}$–CFGs and graphs) *Let $G = (V, E)$ be a finite directed graph.*

(1) If G has a unique distinguished vertex $\mathsf{start} \in V$ and every $v \in V$ is reachable from start then G is a $\{\mathsf{start}\}$–graph.
(2) If G has a unique distinguished vertex $\mathsf{end} \in V$ that is reachable from every vertex $v \in V$ then G is an $\{\mathsf{end}\}$–graph.
(3) If G is a $\{\mathsf{start}\}$–graph and G is also an $\{\mathsf{end}\}$–graph then G is a $\{\mathsf{start}, \mathsf{end}\}$–graph.

If $G = (V, E, \beta)$ is a CFG and the graph (V, E) is a $\{\mathsf{start}\}$–graph then we call G a $\{\mathsf{start}\}$–CFG, etc.

2.2 Useful graph–theoretic definitions

Definition 6 (Paths of a graph) *A path in a graph $G = (V, E)$ is a sequence of vertices $v_1, \ldots, v_i, v_{i+1}, \ldots$ with $(v_i, v_{i+1}) \in E$ for all i.*

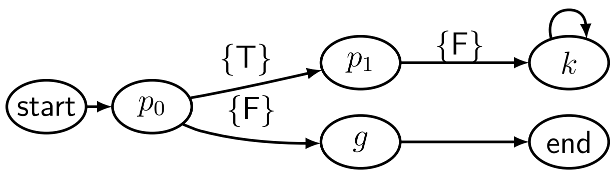
Note that paths can be empty (of length zero), of length one (consisting of a single vertex), or even infinite.

Definition 7 (Proper paths) *A path is* proper *if its initial and final vertices are distinct.*

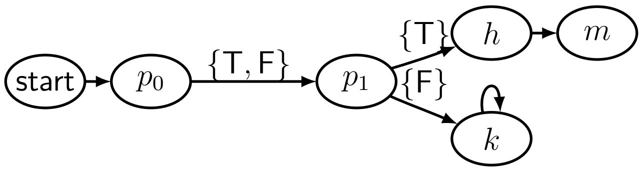
Definition 8 (Prefixes) *A prefix of a path π is a path ρ such that there exists a path σ with $\pi = \rho\sigma$ (the concatenation of ρ and σ). Note π is a prefix of itself. Write $\rho \sqsubseteq \pi$. If $\rho \sqsubseteq \pi$ and $\rho \neq \pi$, ρ is called a 'proper' prefix of π.*

Definition 9 (V'–intervals and V'–paths) *Let $G = (V, E)$ be a graph and let $V' \subseteq V$.*

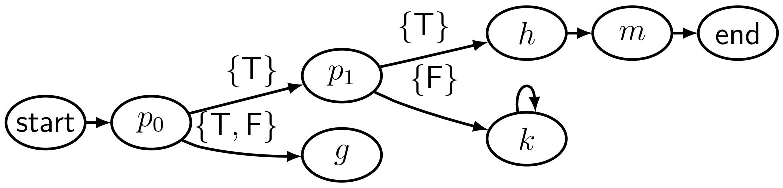
- *A V'–interval is a finite path of length > 1 in G where only the first and last elements are in V'.*
- *An $[l, m]$ V'–interval is a V'–interval that starts at $l \in V'$ and ends at $m \in V'$.*
- *A V'–path [18] is a finite path $v_1 \ldots v_m$ in G where $m > 1$, $v_m \in V'$ and $1 < i < m \Rightarrow v_i \notin V'$.*



(a) $G_{4(a)}$: Predicate p_1 is not complete.



(b) $G_{4(b)}$: A CFG without end.



(c) $G_{4(c)}$: Not a CFG (non–deterministic).

Fig. 4. (a) and (b) are CFGs but (c) is not. In (a) the final vertices are p_1 and end. In (b) the only final vertex is m and the absence of end means that all complete paths are non–terminating.

A V'–path is a path whose last element is in V' and whose first element may be in V' but none of the other elements are in V'. A V'–interval is a V'–path but not necessarily the converse [3].

Definition 10 (Complete Paths of a CFG) *A complete path is either an infinite path or a finite path whose last vertex is final.*

2.3 Examples

In Figure 3(c) the only terminating paths are those whose final vertex is end. The complete paths ending at g although finite are considered non–terminating because $g \neq$ end. Complete paths through k are infinite and hence non–terminating.

$G_{4(a)}$ in Figure 4(a) is an example of a CFG where a predicate (p_1) is incomplete. Therefore p_1 is a final vertex and there are complete paths of $G_{4(a)}$ which end at p_1. These complete paths correspond to the situation where the predicate expression at p_1 evaluates to T.

[3] Ranganath et al. [30] define a similar concept: the *first observable elements* from v, written $\text{obs}^1_{may}(v)$ is the set of *first* elements at the end of a V'–path from v.

$G_{4(b)}$ in Figure 4(b) is an example of a CFG without an end vertex. All complete paths of $G_{4(b)}$, including the finite ones, are thus non–terminating.

$G_{4(c)}$ in Figure 4(c) is not a CFG. If p_0 evaluates to T there is a choice of which edge to follow.

3 The induced graph

Given a program and a slicing criterion, a slicing algorithm produces a subset of the statements of the program, containing the slicing criterion. Normally, the slice produced is a valid and executable program. For a CFG $G = (V, E, \beta)$ the slicing criterion is a set $V' \subseteq V$ of the vertices and a slicing algorithm will produce a possibly larger subset $V' \subseteq V'' \subseteq V$. The problem now is to make a valid CFG on V''.

In this section we show how to define a graph on such a subset by 'rewiring' the edges of the original CFG – this is called the *induced graph*.

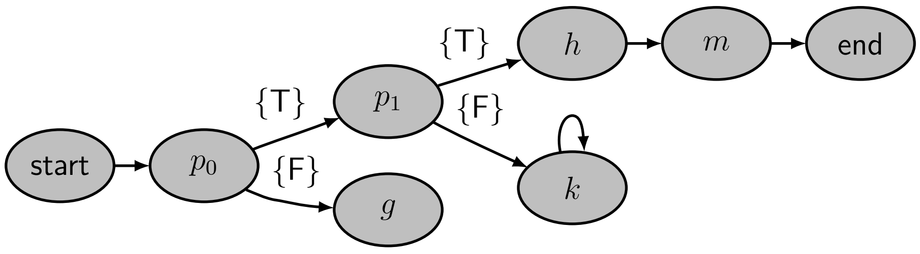
For any CFG $G = (V, E, \beta)$ and $V' \subseteq V$, we can construct a new graph on V' by connecting $x \in V'$ to $y \in V'$ if and only if there is a $[x, y]$ V'–interval (Definition 9). The new edge $(x, y) \in E'$ is labelled by the union of the labels of the edges $(x, x') \in E$ where x' is a successor of x from which there is a path in G to y.

Definition 11 (The induced graph) *Let $G = (V, E, \beta)$ be a CFG and let $V' \subseteq V$. The graph induced by V' from G has edge set $E' \subseteq V' \times V'$ where $(x, y) \in E'$ if and only if there is a V'–interval $x, \ldots, y$ in G. In the graph induced by V' from G ,*

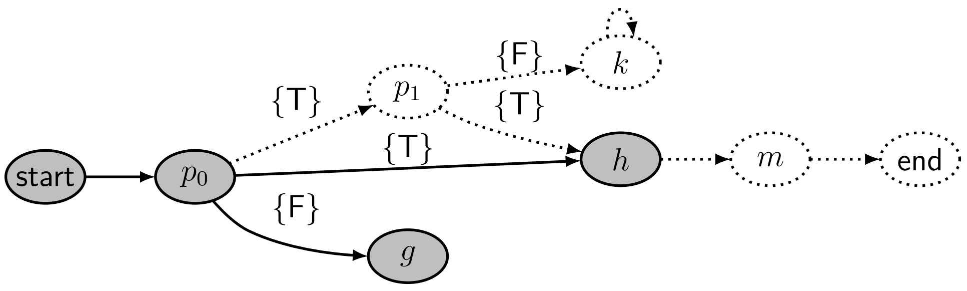
$$\beta'(x, y) = \bigcup_{x' \in K} \beta(x, x')$$
$$\textit{where } K = \{x' \in V \mid (x, x', \ldots, y) \textit{ is a } V'\textit{–interval}\}.$$

The predicates and non–predicates of the graph induced by V' from G are deemed to be $V' \cap P$ and $V' \cap N$, where P and N are the predicates and non–predicates of G respectively.

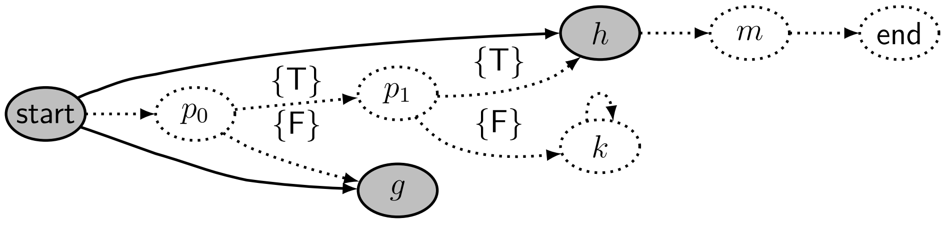
In general, the induced graph is not a CFG because the rewiring may increase the out–degree of vertices, and may destroy the necessary disjointness property of edge labels in a CFG. Figure 5 gives three examples of induced graphs, one of which is a CFG and two of which are not. Figure 5(c) shows the graph induced by $\{\mathsf{start}, h, g\}$ from G_5. It has an edge (start, h) because there is a $\{\mathsf{start}, h\}$–interval, $\mathsf{start}p_0p_1h$, in G_5. Similarly, it has an edge (start, g). Clearly the graph induced by $\{\mathsf{start}, h, g\}$ from G_5 is not a CFG as it has a non–predicate vertex



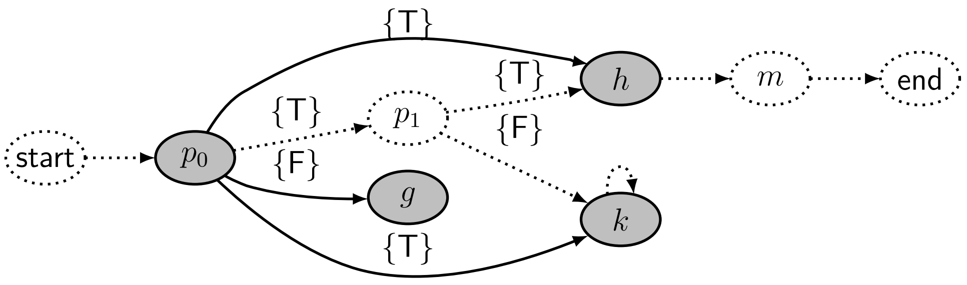
(a) CFG G_5.



(b) The graph induced by $\{\mathsf{start}, p_0, h, g\}$ from G_5 is a CFG.



(c) The graph induced by $\{\mathsf{start}, h, g\}$ from G_5 is not a CFG because it has a non–predicate vertex of out–degree greater than one.



(d) The graph induced by $\{p_0, h, g, k\}$ from G_5 is not a CFG because it has a predicate vertex with non–disjoint edge labels.

Fig. 5. Inducing graphs from G_5 and different sets of vertices (Definition 11). Dotted edges and vertices represent those that are removed in producing the induced graph. Solid edges and labels represent those that remain in the induced graph.

start of out–degree greater than 1. Similarly, the graph induced by $\{p_0, h, g, k\}$ from G_5 is not a CFG because it has a predicate vertex p_0 with non–disjoint edge labels.

In the next section we will show that for the graph induced by V' from G to be a CFG, V' must be *weakly commitment–closed* in G.

4 Weak and strong projections: a semantics of control dependence

In this section, we define weak and strong projections; properties that are preserved between graphs and graphs induced by sets closed under control dependence. The concepts of weak and strong projection are a semantics of non–termination insensitive and non–termination sensitive control dependence respectively. Later, in Sections 7 and 8, we demonstrate this by showing that all forms of control dependence in the literature are special cases of weak and strong projections. This provides strong evidence for our belief that these projections capture the underlying intention and essence of control dependence.

The semantic relationship between programs and their slices has been well studied [5,8–10,12–14]. Weak projection expresses a property analogous to the property of conventional slicing where if the program executes statement s n times then the slice, when executed from the same initial state, also executes s at least n times (if, of course, s is in the slice). In conventional slicing, s may execute more times in the slice than in the original program because, for example, non–terminating loops that prevent s being reached may have been sliced away. This is also true of weak projections. Strong projections, on the other hand, are analogous to the form of slice where for all initial states, s will execute *exactly* the same number of times in the slice as in the original.

We define weak and strong projections in terms of *walks* and give necessary and sufficient conditions for subsets of vertices of a CFG to induce weak and strong projections respectively. It turns out that the graph induced by V' from G is a weak projection of V' if and only if V' is weakly commitment–closed in G (Theorem 41). We go on to define a stronger condition, Strong commitment–closedness, and prove an analogous result, Theorem 45, for strong projections.

4.1 Walks of a CFG

Conventional semantic slices are defined in terms of program executions, so we first define an analogous concept for CFGs using *walks*. A walk of a CFG is similar to a graph–theoretic path of a CFG, except that predicate vertices are replaced with a pair (p, B) where $\mathsf{B} \in \{\mathsf{T}, \mathsf{F}\}$ to represent evaluation of p. Notice, we are not defining a particular start vertex. Our paths and walks can start at *any* vertex in the graph.

Definition 12 (Elements) *Let* $G = (V, E, \beta)$ *be a* CFG*. An* element w *is either a vertex* $v \in N \subseteq V$*, or a pair* (p, B) *where* $p \in P \subseteq V$ *and* $\mathsf{B} \in \{\mathsf{T}, \mathsf{F}\}$*. Write* $\bar{w}$ *for the vertex component of an element and* $\bar{\bar{w}}$ *for the second (boolean) component of the pair when it exists.*

Definition 13 (Walks) *Let $G = (V, E, \beta)$ be a* CFG. *A walk ω in G is a sequence $w_1, w_2, \ldots, w_i, \ldots$ of elements where:*

(1) $\bar{\omega} = \bar{w}_1, \bar{w}_2, \ldots, \bar{w}_i, \ldots$ *is a path in G; and*
(2) *if w_i, w_{i+1} are consecutive elements of ω and $\bar{w}_i$ is a predicate vertex then* $\bar{\bar{w}}_i \in \beta(\bar{w}_i, \bar{w}_{i+1})$.

For example the CFG $G_{4(b)}$ in Figure 4 has a path $\pi_1 = p_0, p_1, h, m$ and there are two walks:

$$\begin{aligned} \omega_1 &= (p_0, \mathsf{T}), (p_1, \mathsf{T}), h, m \\ \text{and } \omega_2 &= (p_0, \mathsf{F}), (p_1, \mathsf{T}), h, m \end{aligned}$$

which give rise to path π_1.

Observe that $\mathsf{start}, (p_0, \mathsf{F})$ is a walk of $G_{3(a)}$ in Figure 3 although p_0 does not have a false branch. This walk cannot go any further. It is an example of a finite maximal walk caused by incomplete predicates.

Definition 14 ($\overrightarrow{G}$) *Let G be a* CFG. *$\overrightarrow{G}$ is the set of all walks in G.*

4.2 Weak projections of a CFG

Restricting a path to a set of vertices means removing all the vertices not in the set.

Definition 15 (Path Restriction) *Let $G = (V, E)$ be a graph, let $V' \subseteq V$, and let π be a path in G. $\pi{\downarrow}V'$ is the subsequence of π obtained by removing all vertices v of π where $v \notin V'$. We say $\pi{\downarrow}V'$ is the* restriction *of π to V'.*

Analogously, restricting a walk to a set of vertices means removing all the elements whose vertex component is not in the set.

Definition 16 (Walk restriction) *Let $G = (V, E, \beta)$ be a* CFG, *let $V' \subseteq V$, and let ω be a walk in G. Define $\omega{\downarrow}V'$ to be the subsequence of ω obtained by removing all elements ω_i of ω where $\bar{\omega}_i \notin V'$. We say $\omega{\downarrow}V'$ is the* restriction *of ω to V'.*

Note that $\overline{\omega{\downarrow}V'} = \bar{\omega}{\downarrow}V'$.

Definition 17 (Weak projections) *Given a* CFG *$G = (V, E, \beta)$, a* CFG *$G' = (V', E', \beta')$ $(V' \subseteq V)$ is a* weak projection *of G if and only if and every walk of G when restricted to V', is a walk of G'.* i.e.,

$$\omega \in \overrightarrow{G} \implies \omega{\downarrow}V' \in \overrightarrow{G'}.$$

Figure 6 gives some examples of weak projections. Here $G_{6(b)}$ is a weak projection of $G_{6(a)}$. The walks of $G_{6(a)}$ are the 46 segments[4] of the three walks:

$$\begin{gathered}\mathsf{start}, (p_0, \mathsf{T}), (p_1, \mathsf{F}), k, \mathsf{end} \\ \mathsf{start}, (p_0, \mathsf{T}), (p_1, \mathsf{T}), h, m, \mathsf{end} \\ \mathsf{start}, (p_0, \mathsf{F}), g, \mathsf{end}\end{gathered}$$

and the walks of $G_{6(b)}$ are all 12 segments of the two walks:

$$\begin{gathered}\mathsf{start}, (p_0, \mathsf{T}), h \\ \mathsf{start}, (p_0, \mathsf{F}), g.\end{gathered}$$

Every walk of $G_{6(a)}$ when restricted to the vertices $\{\mathsf{start}, p_0, h, g\}$ of $G_{6(b)}$ is a walk of $G_{6(b)}$. Similarly, $G_{6(c)}$ is a weak projection of $G_{6(a)}$.

Proposition 18 *Let $G = (V, E, \beta)$ be a* CFG *and $V' \subseteq V$. If the graph induced by V' from G is a* CFG *then the graph induced by V' from G is a weak projection of G.*

PROOF. Let

$$\begin{aligned}\omega = \omega_1, \ldots, \omega_i, \omega_{i+1}, \ldots &\text{ be a walk of } G \\ \text{and write } \omega{\downarrow}V' = \omega_{n_1}, \ldots, \omega_{n_i}, \omega_{n_{i+1}}, \ldots\end{aligned}$$

where $1 < n_1 < n_2 < \cdots$. Then

$$\bar{\omega}_{n_i}, \bar{\omega}_{n_i+1}, \ldots, \bar{\omega}_{n_{i+1}}$$

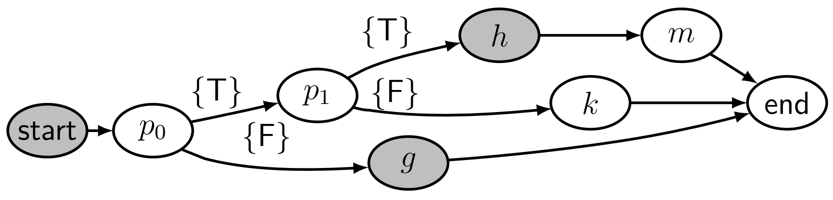
is a V'–interval, and hence by Definition 11

$$\bar{\omega}_{n_1}, \ldots, \bar{\omega}_{n_i}, \bar{\omega}_{n_{i+1}}, \ldots$$

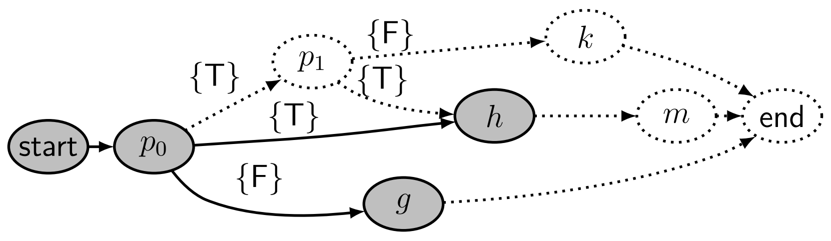
is a path in the graph induced by V' from G. Finally, again by Definition 11 we have $\beta(\bar{\omega}_{n_i}, \bar{\omega}_{n_i+1}) \subseteq \beta'(\bar{\omega}_{n_i}, \bar{\omega}_{n_{i+1}})$ and so $\omega{\downarrow}V'$ is a walk of the graph induced by V' from G.

This shows that the mere act of ensuring that the induced graph is well–formed will also ensure it satisfies the semantic property of being a weak projection.

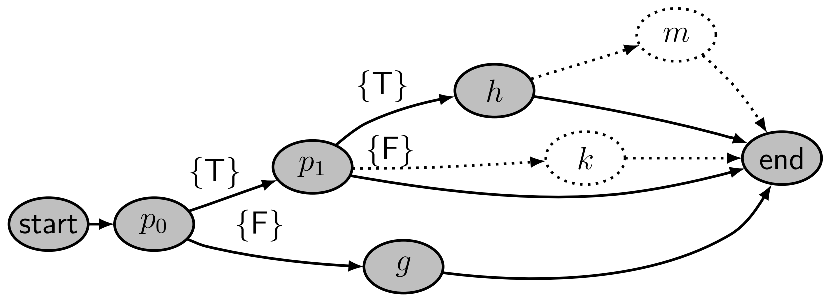
[4] A segment of a sequence is contiguous sequence of elements of the sequence. For example the sequence $\{u, v, w\}$ has segments $\{u, v, w\}$, $\{u, v\}$, $\{v, w\}$, $\{u\}$, $\{v\}$, and $\{w\}$ but not $\{u, w\}$.



(a) $G_{6(a)}$.



(b) $G_{6(b)}$ is a weak projection of $G_{6(a)}$.



(c) $G_{6(c)}$ is also a weak projection of $G_{6(a)}$.

Fig. 6. Two weak projections. Any walk of $G_{6(a)}$ when restricted to the vertices of $G_{6(b)}$ is a walk in $G_{6(b)}$. Ditto $G_{6(c)}$.

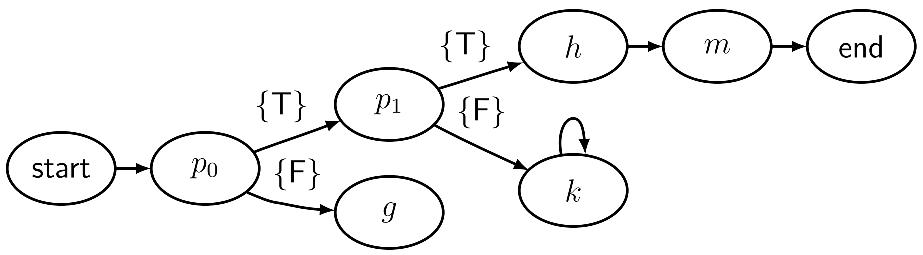


Fig. 7. G_7 has 13 maximal walks (Definition 19): 3 end at g, 4 end with an infinite sequence of ks, and 6 end at end.

4.3 Strong projections of a CFG

A strong projection is a weak projection where also maximal walks project onto maximal walks.

Definition 19 (Maximal walks) *A maximal walk of G is a walk which is not a proper prefix of a walk of G.*

For example the CFG G_7 in Figure 7 has thirteen maximal walks. There are nine finite maximal walks:

$$\begin{array}{l}
\mathsf{start}, (p_0, \mathsf{F}), g \\
(p_0, \mathsf{F}), g \\
g \\
\mathsf{start}, (p_0, \mathsf{T}), (p_1, \mathsf{T}), h, m, \mathsf{end} \\
(p_0, \mathsf{T}), (p_1, \mathsf{T}), h, m, \mathsf{end} \\
(p_1, \mathsf{T}), h, m, \mathsf{end} \\
h, m, \mathsf{end} \\
m, \mathsf{end} \\
\mathsf{end}
\end{array}$$

and four infinite maximal walks:

$$\begin{array}{l}
\mathsf{start}, (p_0, \mathsf{T}), (p_1, \mathsf{F}), k, k, \ldots \\
(p_0, \mathsf{T}), (p_1, \mathsf{F}), k, k, \ldots \\
(p_1, \mathsf{F}), k, k, \ldots \\
k, k, \ldots.
\end{array}$$

The six walks ending with end correspond to executions which terminate normally. The other walks correspond to non–terminating programs, the finite ones to 'silently' non–terminating executions.

Proposition 20 *Let G be a* CFG*. If ω is a maximal walk of G then $\bar{\omega}$ is a complete path of G.*

PROOF. If ω is infinite then $\bar{\omega}$ is infinite and thus is complete. If ω is finite then suppose that $\bar{\omega}$ is not complete. Let $\bar{\omega}$ end at v, say, and since $\bar{\omega}$ is not complete there exists and edge $(v, w) \in E$. But then ω can be extended to w, contradicting the maximality of ω.

The converse is true only for CFGs where all predicates are complete.

Proposition 21 *Let G be a* CFG *and let ω be a walk in G. If $\bar{\omega}$ is complete and all predicates in G are complete then ω is maximal.*

PROOF. If $\bar{\omega}$ is infinite then ω is infinite and thus is maximal. If $\bar{\omega}$ is finite then it ends at a final vertex. Now, all predicate vertices are complete, therefore all final vertices in G must have out–degree 0, and not 1. Thus $\bar{\omega}$ cannot be a prefix of any extending path and hence ω is maximal.

The converse of Proposition 20 is not true for CFGs that contain incomplete predicates. For example in $G_{4(a)}$ in Figure 4(a) the two walks

$$\omega_1 = (p_0, \mathsf{T}), (p_1, \mathsf{T})$$
$$\omega_2 = (p_0, \mathsf{T}), (p_1, \mathsf{F})$$

have $\bar{\omega}_1 = \bar{\omega}_2 = p_0, p_1$ which is complete because it ends at the final vertex p_1. However, ω_1 is maximal but ω_2 is not. Nevertheless, as this example implies, for every complete path π there exists a maximal walk ω with $\bar{\omega} = \pi$.

Proposition 22 *Let G be a* CFG*. If π is a complete path of G then there exists a maximal walk ω of G such that $\bar{\omega} = \pi$.*

PROOF. Follows immediately from the definitions of maximal walks and complete paths.

There may exist more than one maximal walk with the same complete path, for example where an edge is labelled $\{\mathsf{T}, \mathsf{F}\}$.

Definition 23 (Strong projections) *Let $G = (V, E, \beta)$ be a* CFG *and $V \subseteq V'$. A* CFG *$G' = (V', E', \beta')$ is a* strong projection *of* CFG *if and only if all maximal walks of G when restricted to V' give maximal walks of G'.* i.e.,

$$\omega \in \overrightarrow{G} \textit{ is maximal} \implies \omega{\downarrow}V' \in \overrightarrow{G'} \textit{ and is maximal.}$$

So, for every walk in a strong projection, the number of times we visit a vertex in the projection is *identical* to the number of times we visit it in the corresponding walk in the original. Strong projections have surprising property: *every* walk in a strong projection is the restriction of a walk of the original.

Lemma 24 *A strong projection is a weak projection.*

PROOF. This follows immediately from the fact that every walk is the prefix of a maximal walk.

Lemma 25 *Let the* CFG $G' = (V', E', \beta')$ *be a strong projection of the* CFG $G = (V, E, \beta)$. *For all* $(x, y) \in E'$ *there exists an* $[x, y]$ V'*–interval in* G.

PROOF. Assume that $(x, y) \in E'$.

(1) If x is a predicate then without loss of generality we can assume that $\mathsf{T} \in \beta'(x, y)$. Let w be a maximal walk of G starting from (x, T).

Since G' is a strong projection of G, $w{\downarrow}V'$ is maximal in G'. The walk w will reach V' after (x, T), because otherwise the $w{\downarrow}V'$ is just (x, T), which cannot be maximal since $\mathsf{T} \in \beta'(x, y)$. (This is where the proof would break down for weak projections.) Therefore let v' be the first V' vertex after (x, T) in w. Since $w{\downarrow}V'$ is a walk of G', there must be an edge in $(x, v') \in E'$ with $\mathsf{T} \in \beta'(x, v')$. Since G' is a CFG we must have $y = v'$. Hence there is an $[x, y]$ V'–interval in G as required.

Moreover, it follows that if G' contains a walk with first element (x, T) and with second element having vertex component y, then G contains a walk with first element (x, T), with a later element having vertex component y, and with no intermediate element having a vertex component in V'. Similarly with F instead of T. (This stronger result is needed in order to use Lemma 25 for Proposition 26.)

(2) If x is a non–predicate then as above, let w be a maximal walk of G starting from x. Using the same argument as above, w will reach V' after x and by the uniqueness of the next element of non–predicate vertices we must have that y is the unique next element after x in $w{\downarrow}V'$. Hence there is an $[x, y]$ V'–interval in G as required.

Proposition 26 *Let the* CFG $G' = (V', E', \beta')$ *be a strong projection of the* CFG $G = (V, E, \beta)$. *For all walks* ω' *of* G' *there exists a walk* ω *of* G *such that* $\omega{\downarrow}V' = \omega'$.

PROOF. Follows from Lemma 25 by induction on the length of a finite prefix of a walk.

In fact every path in a strong projection arises as the slice of a path in the original, *i.e.,* 'the path of the strong projection is (exactly) the slice of the path.' This is different from the weak case where the restriction of a path may be only a prefix of a path in the induced graph.

Corollary 27 *Let the* CFG $G' = (V', E', \beta')$ *be a strong projection of the* CFG $G = (V, E, \beta)$. *For all complete paths* π' *of* G' *there exists a complete path* π *of* G *such that* $\pi{\downarrow}V' = \pi'$.

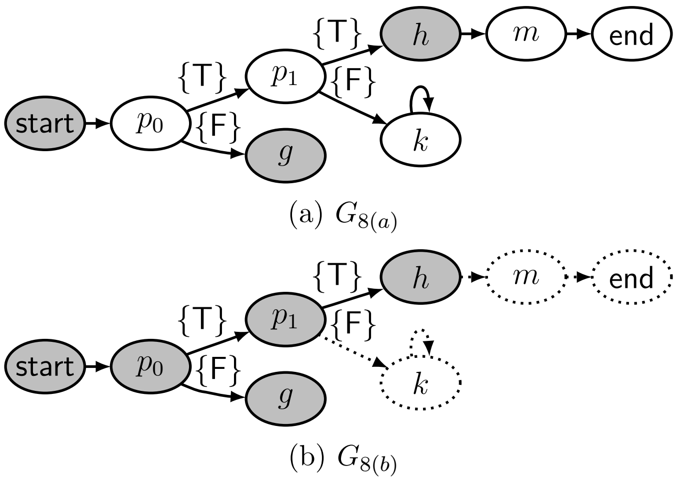


Fig. 8. Strong projections not containing end do not necessarily preserve termination conditions: $G_{8(b)}$ (removed vertices and edges shown dotted) is a strong projection of $G_{8(a)}$ The terminating complete path: start p_0 p_1 h m end in the original induces the path: start p_0 p_1 h in the induced graph. This path is non–terminating since it ends at a non–end non–predicate of out–degree zero.

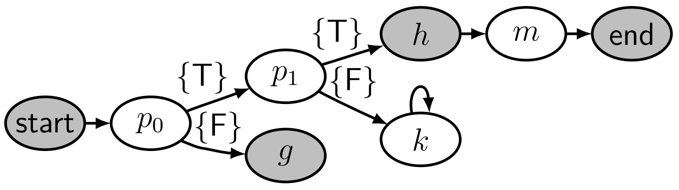
PROOF. If π' is a complete path of G' then by Proposition 22 there is a maximal walk $\omega' \in \overrightarrow{G'}$ with $\bar{\omega}' = \pi'$. By Proposition 26 there exists $\omega \in \overrightarrow{G}$ with $\omega{\downarrow}V' = \omega'$ If ω is not maximal then simply take any maximal walk μ extending ω. G' is a strong projection therefore $\bar{\mu}{\downarrow}G' = \pi'$ for otherwise π' is not maximal.

4.4 Weak and strong projections and non–termination

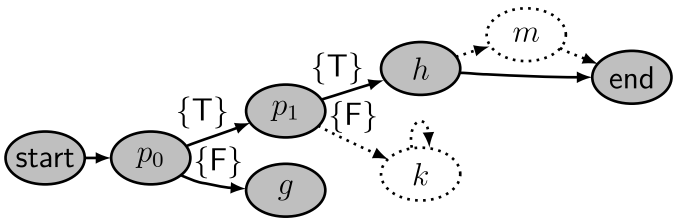
Weak and strong projection of a CFG G may both non-terminate when G terminates. To see this in the weak case, consider $G_6(b)$ which is a weak projection of $G_6(a)$ in Figure 6. The terminating walk start, $(p_0, \mathsf{F}), g,$ end restricts to the non-terminating walk start, $(p_0, \mathsf{F}), g$. This walk is non-terminating because it ends in a final non–end vertex g in $G_6(b)$.

Similarly, to see this in the strong consider $G_{8(b)}$ in Figure 8. $G_{8(b)}$ is a strong projection of $G_{8(a)}$ The terminating complete path: start p_0 p_1 h m end in the original induces the path: start p_0 p_1 h in the induced graph. This path is non–terminating since it ends at a non–end non–predicate of out–degree zero [5] .

[5] In this example the smallest set containing $\{\mathsf{start}, g, h\}$ closed under $\xrightarrow{\text{NTSCD}}$ and $\xrightarrow{\text{DOD}}$ of Ranganath et al.(defined later) is also $\{\mathsf{start}, g, h, p_0, p_1, \mathsf{end}\}$ inducing $G_{8(b)}$, showing that their so called ‘non–termination sensitive control dependence’ does not always preserve termination conditions either.

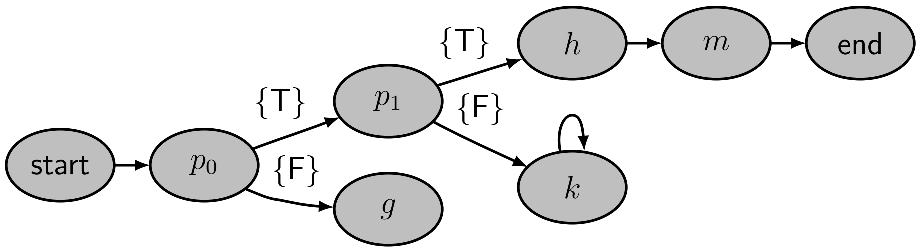


(a) $G_{9(a)}$

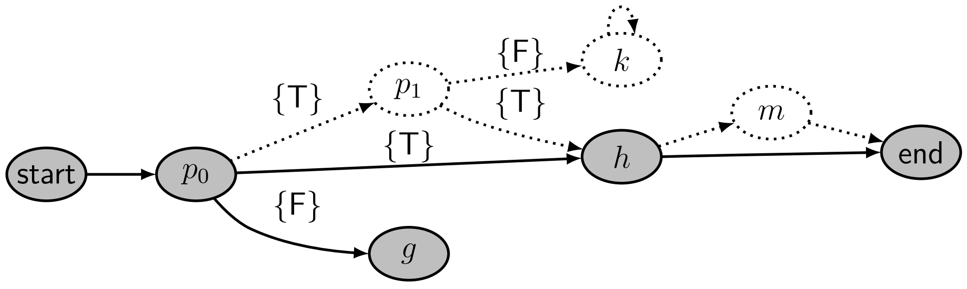


(b) $G_{9(b)}$

Fig. 9. Termination behaviour is preserved only when end is included in the strong projection. $G_{9(b)}$ is a strong projection of $G_{9(a)}$. Both have the same termination conditions.



(a) CFG $G_{10(a)}$.



(b) $G_{10(b)}$

Fig. 10. $G_{10(b)}$ is a weak projection of $G_{10(a)}$ but it does not preserve non-termination. Predicate vertex p_1, which can lead to non-termination in $G_{10(a)}$, does not exist in $G_{10(b)}$.

The weak projection of a CFG G may terminate when G does not. To see this, consider Figure 10. $G_{10(b)}$ is a weak projection of $G_{10(a)}$ but it does not preserve non-termination. Predicate vertex p_1, which can lead to non-termination in $G_{10(a)}$, does not exist in $G_{10(b)}$.

If a weak or strong projection contains end then it cannot introduce non–termination. This is stated formally in Lemma 28. A terminating walk is a finite walk whose final element is end.

Lemma 28 *Let $G = (V, E, \beta)$ and $G' = (V', E', \beta')$ be* CFG*s containing $\{\mathsf{end}\}$ and let G' be a weak projection of G. If ω is a terminating walk of G then $\omega{\downarrow}V'$ is a terminating walk of G'.*

PROOF. This follows immediately from the fact the $\mathsf{end} \in V'$.

Lemma 29 *Let $G = (V, E, \beta)$ and $G' = (V', E', \beta')$ be* CFG*s and let G' be a strong projection of G. If ω is a non-terminating walk of G then $\omega{\downarrow}V'$ is a non-terminating walk of G'.*

PROOF. Suppose $\omega{\downarrow}V'$ is a terminating walk of G'. Then by definition, the final element of $\omega{\downarrow}V'$ is end. Now, since $V' \subseteq V$ we must have $\mathsf{end} \in V$. So end must be an element of ω and hence ω is terminating. Contradiction.

Since a strong projection is a weak projection, Lemmas 28 and 29 guarantee that strong projections containing end of a CFG, g, preserve the termination conditions of g. See Figure 9 for an example.

5 Weak and strong commitment–closedness: a generalisation of non–termination sensitive and non–termination insensitive control dependence

In this section we develop a theory of *weak and strong commitment–closedness.* These are properties of vertex sets which are necessary and sufficient to induce graphs which are weak and strong projections. Weak and strong commitment–closedness are used both in the production of algorithms for producing minimal weak and strong projections and also in the proofs that classify the previous forms of control dependence as either weak or strong. An advantage of weak and strong commitment–closedness is that they are defined for *any* directed graph not just CFGs.

5.1 Weak commitment–closedness

Informally, at this stage, a set is weakly commitment–closed if and only if it is closed under non–termination sensitive control dependence. Before we can define weak commitment–closedness, we need a preliminary definition:

Definition 30 (V'–weakly committing vertices) *Let G be a directed graph. A vertex v is V'–weakly committing in G if all V'–paths from v have the same end point. In other words, there is at most one element of V' that is 'first–reachable' from v.*

For example, in Figure 5 the predicate p_1 is $\{\mathsf{start}, h, g\}$–weakly committing in G_5 since the only first–reachable vertex in $\{\mathsf{start}, h, g\}$ from p_1 is h. Vertex p_0, on the other hand, is not $\{\mathsf{start}, h, g\}$–weakly committing in G_5 since both h and g are first reachable from p_0.

Definition 31 (Weak commitment–closedness) *Let G be a directed graph and let $V' \subseteq V$. V' is weakly commitment–closed in G if and only if all vertices not in V' that are reachable from V' are V'–weakly committing in G.*

In Figure 5, $\{\mathsf{start}, h, g\}$ is not weakly commitment–closed in G_5 because p_0 is reachable from $\{\mathsf{start}, h, g\}$ but p_0 is not $\{\mathsf{start}, h, g\}$–weakly committing since h and g are both first reachable from p_0. However, $\{\mathsf{start}, p_0, h, g\}$ is weakly commitment–closed in G_5 because all of the vertices of G_5 that are not in $\{\mathsf{start}, p_0, h, g\}$ and are reachable from $\{\mathsf{start}, p_0, h, g\}$ are $\{\mathsf{start}, p_0, h, g\}$–weakly committing. Later, we will show that sets closed under all weak forms of control dependence in the literature are weakly commitment–closed.

5.2 Strong commitment–closedness

Informally a set is weakly commitment–closed if and only if it is closed under non–termination sensitive control dependence.

Definition 32 (V'–strongly committing vertices) *Let $G = (V, E, \beta)$ be a* CFG *and let $V' \subseteq V$. A vertex v is V'–strongly committing G if and only if it is V'–weakly committing in G and all complete paths in G from v contain an element of V'.*

This means that *all* paths from v re–enter V' (and do so at the same vertex) whereas if v is only V'–weakly committing in G then some paths from v in G may never re-enter V'.

Definition 33 (V'–avoiding vertices) *Let $G = (V, E, \beta)$ be a* CFG *and let $V' \subseteq V$. A vertex v is V'–avoiding in G if and only if no vertex in V' is reachable in G from v.*

Definition 34 (Strong commitment–closedness) *Let $G = (V, E, \beta)$ be a* CFG *and let $V' \subseteq V$. V' is strongly commitment–closed in G if and only if every vertex in $V \backslash V'$ that is reachable in G from V' is V'–strongly committing or V'–avoiding in G.*

In Figure 7, the set $\{start, g, h, \mathsf{end}\}$ is not strongly commitment–closed in G_7 because p_1 is reachable from $\{start, g, h, \mathsf{end}\}$ but is neither $\{start, g, h, \mathsf{end}\}$–strongly committing or $\{start, g, h, \mathsf{end}\}$–avoiding. (Similarly p_0.)

In Figure 7, the set $\{start, p_0, p_1, h, \mathsf{end}\}$ is strongly commitment–closed in $G_9(b)$ because k and g are $\{start, p_0, p_1, h, \mathsf{end}\}$–avoiding and m is $\{start, p_0, p_1, h, \mathsf{end}\}$–strongly committing. Later, we will show that sets closed under all strong forms of control dependence in the literature are strongly commitment–closed.

5.3 Graphs induced by weakly commitment–closed sets

We now develop the theory of graphs induced by weakly commitment–closed sets. We show that for any subset V' of the vertices of a CFG, G, V' being weakly commitment–closed in G is both necessary and sufficient for the induced graph from V' not only to be well formed, but also to be a weak projection.

Lemma 35 *Let $G = (V, E, \beta)$ be a CFG, $V' \subseteq V$, and $v \in V'$. If V' is weakly commitment–closed in G then the out–degree of v in the graph induced by V' from G is at most the out–degree of v in G.*

PROOF.

Let $v_1, \ldots, v_n$ be all the successors of v in the graph induced by V'. Thus there are V'-paths, $v\gamma_i$, in G ending with v_i for each $i \leqslant n$. Let w_i be the first vertex of γ_i. We must have $i \neq j$ implies $w_i \neq w_j$, since if $w_i = w_j$ then either $w_i \notin V'$ in which case w_i would not be V'-weakly committing in G although it is reachable from V', implying that V' is not weakly commitment-closed in G, or $w_i \in V'$ and so $v_i = v_j$, contrary to our assumption. Thus v has n successors $w_1, \ldots, w_n$ in G, proving the result.

Proposition 36 *Let $G = (V, E, \beta)$ be a CFG and $V' \subseteq V$. If V' is weakly commitment–closed in G then the graph induced by V' from G is a CFG.*

PROOF. Let the graph induced by V' from G be (V', E', β'). By Lemma 35, the graph induced by G on V' satisfies conditions (1) and (2) of Definition 1. Suppose that part (3) of Definition 1 does not hold, then there exist edges $(x, y) \in E'$ and $(x, z) \in E'$ with $y \neq z$ but $\beta'(x, y) \cap \beta'(x, z) \neq \emptyset$. Assume without loss of generality that $\mathsf{T} \in \beta'(x, y) \cap \beta'(x, z)$. By Definition 11, there exists y_1 and z_1 in V such that $\mathsf{T} \in \beta(x, y_1)$ and $\mathsf{T} \in \beta(x, z_1)$ and V'–intervals $x, y_1 \ldots y$ and $x, z_1 \ldots z$. Since G is a CFG, and hence edges from the same predicate must be disjointly labelled, we must have $y_1 = z_1$. But the V'–paths $y_1 \dots y$ and $z_1 \dots z$ then contradict the hypothesis that V' is weakly commitment–closed in G.

Proposition 37 *Let $G = (V, E, \beta)$ be a* CFG *and $V' \subseteq V$. If the graph induced by V' from G is a* CFG *then V' is weakly commitment–closed in G.*

PROOF. Suppose not, then there exists $v \notin V'$ reachable from V' but not V'–weakly committing in G. Therefore exists $v' \in V'$ and V'–intervals

$$v', v'', \dots, v, \dots, l_1 \quad \text{and} \quad v', v'', \dots, v, \dots, l_2$$

with $l_1 \neq l_2$. Let the graph induced by V' from G be $(V', E'\beta')$. By Definition 11, the graph induced by V' from G will contain the edges $\{v', l_1\}$ and $\{v', l_2\}$. If $v' \in P$ (*i.e.,* it is a predicate) then the two edges would not have disjoint labelling, since $\beta'(v', l_1) \cap \beta'(v', l_2) \supseteq \beta(v', v'') \neq \emptyset$. If $v' \in N$ (*i.e.,* it is a non–predicate), then in the induced graph a non–predicate would have two successors. Both cases contradict the fact that the graph induced by V' from G is a CFG.

The following three straightforward results show when a subset of the vertices includes the necessary distinguished vertices that the graph induced on that subset again belongs to the same restricted class.

Proposition 38 *Let $G = (V, E, \beta)$ be a $\{\mathsf{start}\}$–*CFG*. V' is weakly commitment–closed in G and $\mathsf{start} \in V'$ if and only if the graph induced by V' from G is a $\{\mathsf{start}\}$–*CFG*.*

PROOF. This follows immediately from Propositions 36 and 37, and the fact that if $\mathsf{start} \in V'$, then every vertex in V' is reachable from start in the graph induced by V' from G, which is a consequence of Definition 11 and the analogous assertion in G.

Proposition 39 *Let $G = (V, E, \beta)$ be an $\{\mathsf{end}\}$–*CFG*. V' is weakly commitment–closed in G and $\mathsf{end} \in V'$ if and only if the graph induced by V' from G is an $\{\mathsf{end}\}$–*CFG*.*

PROOF. Similar to the proof of Proposition 38.

Proposition 40 *Let $G = (V, E, \beta)$ be a $\{\mathsf{start}, \mathsf{end}\}$–*CFG*. V' is weakly commitment–closed in G and $\{\mathsf{start}, \mathsf{end}\} \subseteq V'$ if and only if the graph induced by V' from G is a $\{\mathsf{start}, \mathsf{end}\}$–*CFG*.*

PROOF. This is an immediate consequence of Propositions 38 and 39.

We have proven that if V' is weakly commitment–closed in G then the graph induced from V' is a well formed CFG and conversely if the graph induced from $V' \subseteq V$ is a well–formed CFG then V' is weakly commitment–closed in G. A set being weakly commitment–closed in G is thus an equivalent to the graph induced by it from G being a well–formed CFG. In Section 7, we prove that weak commitment–closedness generalises the property that V' is closed under each of the weak forms of control dependence in the literature.

Theorem 41 *Let $G = (V, E, \beta)$ be a CFG and $V' \subseteq V$. The following are equivalent.*

(1) The graph induced by V' from G is a CFG.
(2) V' is weakly commitment–closed in G.
(3) The graph induced by V' from G is a weak projection of G.

PROOF. (1) $\implies$ (2) by Proposition 37.
(2) $\implies$ (3) by Proposition 36 and Proposition 18.
(3) $\implies$ (1) because by definition a weak projection is a CFG.

We have thus shown that for any subset V' of the vertices of a CFG, G, V' being weakly commitment–closed in G is both necessary and sufficient for the induced graph from V' not only to be well formed, but also to be a weak projection.

Recall from Theorem 41 that the graph induced on a subset V' of the set of vertices of a CFG $G = (V, E, \beta)$ is a weak projection of G if and only if V' is weakly commitment–closed in G. In the next section we define the corresponding property of *strong* commitment–closedness for strong projections.

5.4 Graphs induced by strongly commitment closed sets

In this section, we investigate graphs induced by strongly commitment–closed sets. The main result of this section is that graph induced by V' from G is a strong projection of G if and only if V' is strongly commitment–closed in G.

Lemma 42 *If V' is strongly commitment–closed in G then V' is weakly-commitment–closed in G.*

PROOF. Result follows from the fact that a strongly committing vertex is weakly committing (Definition 32) and that a V'–avoiding vertex is vacuously V'–weakly committing.

An example of strong commitment–closedness in terms of CFGs can be seen in Figure 8. The smallest strongly commitment–closed set containing V' is $\{\mathsf{start}, p_0, p_1, g, h\}$. Unlike in the weak case (see $G_{6(b)}$ in Figure 6), vertex p_1 is included, since k is avoiding and h is strongly committing.

Proposition 43 *Let $G = (V, E, \beta)$ be a* CFG *and let $V' \subseteq V$. If the graph induced by V' from G is a strong projection of G and hence a* CFG*, then V' is strongly commitment–closed in G.*

PROOF. By Proposition 37, V' is weakly commitment–closed in G. Suppose $v \notin V'$ is reachable from V' but v is not V'–avoiding and v is not V'–strongly committing. Since v is reachable from V' there is a path $\rho = v_1 \dots v_{n-1}v_n$ $(v_n = v)$ from $v_1 \in V'$ to v with $v_2, \dots, v_n \in V \backslash V'$. Since v is not V'–avoiding in G, there is a V'–path $v\mu k$ from v to some vertex $k \in V'$. Since v is V'–weakly committing in G but not V'–strongly committing in G there exists a V'–avoiding path $vv_{n+1}v_{n+2}\dots$ say.

Now, $\rho v_{n+1}v_{n+2}\dots$ is a complete path and so by Proposition 22 there exists a maximal walk $\omega = \omega_1\omega_2\dots$ where $\bar{\omega}_i = v_i$. However, $\omega{\downarrow}V' = \omega_1$ and the existence of the V'–interval $\rho\mu k$ means that by Definition 11 there is an edge (v_1, k) in the graph induced by V' from G and if v_1 is a predicate then $\bar{\bar{\omega}}_1 \in \beta(v_1, v_2) \subseteq \beta'(v_1, k)$. Hence ω_1 is a prefix of any walk $\omega_1 l \dots$, where $\bar{l} = k$, *i.e.,* it is not maximal which contradicts that the graph induced by V' from G is a strong projection of G.

Proposition 44 *Let $G = (V, E, \beta)$ be a* CFG *and $V' \subseteq V$. If V' is strongly commitment–closed in G then the graph induced by V' from G is a strong projection of G.*

PROOF. By Proposition 36, the graph induced by V' from G is a CFG and then, by Proposition 18 the graph induced by V' from G is a weak projection of G. Suppose it is not a strong projection. Then there exists a maximal walk ω of G such that $\omega{\downarrow}V'$ is a proper prefix of a maximal walk of the graph induced by V' from G. This means that $\omega{\downarrow}V'$ is finite, so write

$$\omega{\downarrow}V' = \omega_1, \dots, \omega_n.$$

Since $w_1 \dots w_n$ is not maximal in the graph induced by V' from G, there exists a walk $w_1 \dots w_n w'$ in the graph induced by V' from G. There must exist a predicate $p \notin V'$ on a path between w_n and w' in G from which some paths reach $w' \in V'$ and from which at least one path never re–enters V', since if all paths in G re–enter V' from w_n then w cannot be maximal in the graph induced by V' from G. By definition, p is not strongly V' avoiding or not strongly committing and hence V' is not strongly commitment–closed in G.

Theorem 45 *Let $G = (V, E, \beta)$ be a* CFG *and $V' \subseteq V$. The graph induced by V' from G is a strong projection of G if and only if V' is strongly commitment–closed in G.*

PROOF. ($\Rightarrow$) Proposition 43.
($\Leftarrow$) Proposition 44.

In Section 4, we defined weak and strong projections and in this section we have given necessary and sufficient conditions for a subset, V', of vertices of a CFG to induce weak and strong projections respectively. The conditions are that V' is weakly commitment–closed in G and V' is strongly commitment–closed in G respectively. In this section we prove that for any subset of vertices V' of a CFG, there are unique minimal sets V'' and V''' containing V' such that V'' is weakly commitment–closed in G and V''' is strongly commitment–closed in G. This implies that for any subset of vertices V' of a CFG, there are unique minimal sets V'' and V''' containing V' such that the graphs induced from G by V'' and V''' are weak and strong projections of G respectively.

5.5 Existence and uniqueness of minimal weakly commitment–closed sets

Good slices are small slices. Given a CFG $G = (V, E, \beta)$ and a slicing criterion $V' \subseteq V$ we want to find the smallest subset V'' of V containing V' such that the graph induced from G by V'' is a weak projection of G. By Theorem 41, V'' will be the smallest weakly commitment–closed set containing V'. In this subsection, we prove that such a set exists and is unique.

In order to do this we need the concept of a *weakly deciding* vertex. Informally, a vertex is weakly deciding over a set V' if it *decides* between any two vertices in V'. It is called weak because the choice does not guarantee reaching an element in V'. Having made the choice, however, there will be at least one path that reaches the vertex in V'. The choice guarantees that the *other* interesting vertex will definitely not be reached first.

Definition 46 (Weakly deciding vertices) *Let $G = (V, E)$ be a finite directed graph and let $V' \subseteq V$. A vertex $v \in V$ is V'–weakly deciding in G if and only if there exist two finite proper V'–paths in G that both start at v and* *have no other common vertex. We write* $\mathsf{WD}_G(V')$ *for the set of all* V'*–weakly deciding vertices in* G*.*

It is possible for a vertex to be neither V'–weakly committing in G, nor V'–weakly deciding in G. To see this, consider vertex start of $G_{6(a)}$ in Figure 6. It is not $\{\mathsf{start}, h, g\}$–weakly committing in $G_{6(a)}$ since h and g are both first reachable from start, nor $\{\mathsf{start}, h, g\}$–weakly deciding in $G_{6(a)}$ since all proper V'–paths from start contain p_0 and are hence not disjoint.

There now follows a lemma which shows that V' is weakly commitment–closed if and only if all V'–weakly deciding vertices that are reachable from V' are in V'.

Lemma 47 (Weak commitment–closedness in terms of WD) *Let* $G = (V, E)$ *be a finite directed graph and let* $V' \subseteq V$*.* V' *is weakly commitment–closed in G if and only if all* V'*–weakly deciding vertices in G that are reachable from* V' *are in* V'*.*

PROOF. Suppose that V' is weakly commitment–closed in G and let $v \in \mathsf{WD}_G(V')\backslash V'$. Since v is V'–weakly deciding in G there must exist V'–paths $v...v'$ and $v...v''$ which share no common vertex after their common initial vertex v. Therefore v is not V'–weakly committing and because V' is weakly commitment–closed in G, v cannot be reachable from V'.

Conversely, let $v \notin V'$ be reachable from V' but not V'–weakly committing, so there exist V'–paths $v...v'$ and $v...v''$ for $v' \neq v''$. If w is the last common vertex on these paths then w is reachable from V' and $w...v'$ and $w...v''$ share no common vertex after w and $w \neq v', w \neq v''$. Hence w is V'–weakly deciding in G.

There now follow a useful result stating that WD is monotonic.

Lemma 48 (WD is monotonic) *Let* $G = (V, E)$ *be a finite directed graph and* $V_1 \subseteq V_2 \subseteq V$*, then*

$$\mathsf{WD}_G(V_1) \subseteq \mathsf{WD}_G(V_2).$$

PROOF. Suppose there is a vertex $v \in \mathsf{WD}_G(V_1)\backslash \mathsf{WD}_G(V_2)$. Clearly there are proper $[v, m_i]$ V_1–paths γ_i in G, for vertices $m_1, m_2 \in V_1 \subseteq V_2$, which share only v as a common vertex. Let δ_i be the prefix of γ_i ending at the first vertex in V_2 for each i; since $v \notin \mathsf{WD}_G(V_2)$, each δ_i is a proper path from v and the δ_i also share only v as a common vertex. Thus $v \in \mathsf{WD}_G(V_2)$, giving a contradiction.

Theorem 49 *Let $G = (V, E)$ be a finite directed graph and let $V' \subseteq V$. Suppose that V' is not weakly commitment–closed in G. Then there is an edge (p, r) in G such that p is reachable in G from V' with*

(1) $|\Theta(G, V', r)| = 1$ *and*
(2) $|\Theta(G, V', p)| \geqslant 2$.

Furthermore, for any edge (p, r) satisfying (1) and (2), the vertex p lies in every weakly commitment–closed subset of V containing V'.

PROOF. Since V' is not weakly commitment–closed, there is a vertex $p \in V \backslash V'$ that is reachable in G from V' and is not V'–weakly committing. Hence there is a path $p_0 = p, p_1 = r, \ldots, p_m \in V'$ in G. Choose p so that m is minimal. Thus r is V'–weakly committing and $|\Theta(G, V', r)| = 1$ and $|\Theta(G, V', p)| \geqslant 2$ follow from this. Now assume that an edge (p, r) is in G and satisfies the conditions given and p is reachable in G from V', but there is a weakly commitment–closed set $W \supseteq V'$ not containing p. Since $|\Theta(G, V', r)| = 1$, there is a path $p_0 = p, p_1 = r, \ldots, p_m \in \Theta(G, V', r)$ in G, for some $m \geqslant 1$. In addition, $|\Theta(G, V', p)| \geqslant 2$ and so there is also a path $q_0 = p, q_1, \ldots, q_n \in V' \backslash \{p_m\}$ in G for some $n \geqslant 1$. Since $\Theta(G, V', p_i) = \{p_m\}$ for each $i \geqslant 1$, no $p_i = q_j$ unless $i = j = 0$. Thus $p \in \mathsf{WD}_G(V') \subseteq \mathsf{WD}_G(W)$ by Lemma 48, and so W is not weakly commitment–closed, giving a contradiction.

Thus we have shown if an edge (p, r) in V satisfies $|\Theta(G, V', r)| = 1$ and $|\Theta(G, V', p)| \geqslant 2$, then p lies in every weakly commitment–closed superset of V'.

Theorem 50 *Let $G = (V, E)$ be a finite directed graph and let $V' \subseteq V$. There exists a unique minimal weakly commitment–closed subset of V that contains V'.*

PROOF. We now prove the uniqueness of minimal weakly commitment–closed supersets of V' by induction on $|V| - |V'|$. We may assume that V' is not weakly commitment–closed, hence by Theorem 49, there is an edge (p, r) in V that satisfies $|\Theta(G, V', r)| = 1$ and $|\Theta(G, V', p)| \geqslant 2$, and such that p lies in every weakly commitment–closed superset of V'. If $V' \cup \{p\}$ is weakly commitment–closed, then the uniqueness result follows immediately. Otherwise, by the inductive hypothesis, there is a unique minimal weakly commitment–closed superset of $V' \cup \{p\}$, and since every weakly commitment–closed superset of V' must contain p, the result follows.

We now consider the analogous problem of proving the existence of the minimal strongly commitment–closed superset of a given set. We prove this to be well–defined in Theorem 53. In order to do this, we first need to define the function Γ which gives the set of vertices lying on complete paths not passing through V'.

Definition 51 (Γ) *Let $G = (V, E)$ be a finite directed graph and let $V' \subseteq V$. We define $\Gamma(G, V')$ to be the set of all $x \in V$ that lie on a complete path in G which does not pass through V'.*

Definition 52 (Θ) *Let $G = (V, E)$ be a finite directed graph and let $V' \subseteq V$. Let H be the* CFG *obtained from G by deleting all edges (v', v) with $v' \in V'$. For any $u \in V$, we define $\Theta(G, V', u)$ to be the set of vertices in V' that are reachable in H from u.*

Note that the elements in V' reachable from elements in $V \backslash V'$ in H are the elements in V' *first* reachable from elements in $V \backslash V'$ in G.

Theorem 53 *Let $G = (V, E)$ be a finite directed graph and let $V' \subseteq V$. If V' is not strongly commitment–closed then there is an edge (p, r) in G with:*

(1) $p \in V \backslash V'$.
(2) p is reachable in G from V'.
(3) $|\Theta(G, V', r)| = 1$.
(4) $r \notin \Gamma(G, V')$.
(5) Either $|\Theta(G, V', p)| \geqslant 2$ or $p \in \Gamma(G, V')$.

Furthermore, for any edge (p, r) in G satisfying these conditions, the vertex p lies in every strongly commitment–closed subset of V containing V'. From this, we show it follows that there is a unique minimal strongly commitment–closed superset of V'.

PROOF. Since V' is not strongly commitment–closed, there is a vertex $p \in V \backslash V'$ that is reachable in G from V' that is neither V'–strongly committing nor V'–avoiding in G. Hence there is a path $p_0 = p, p_1 = r, \ldots, p_m \in V'$ in G. Choose p so that m is minimal. Thus r is V'–strongly committing and the conditions involving the functions Θ and Γ follow from this. Now assume that there is an edge (p, r) in G that satisfies the conditions given, but that there is a strongly commitment–closed set $W \supseteq V'$ not containing p. Since $|\Theta(G, V', r)| = 1$, there is a path $p_0 = p, p_1 = r, \ldots, p_m \in \Theta(G, V', r)$ in G, for some $m \geqslant 1$. From the condition on p, either of two possibilities may occur.

- $|\Theta(G, V', p)| \geqslant 2$ and so there is also a path $q_0 = p, q_1, \ldots, q_n \in V' \backslash \{p_m\}$ in G for some $n \geqslant 1$. Since $\Theta(G, V', p_i) = \{p_m\}$ for each $i \geqslant 1$, no $p_i = q_j$ unless $i = j = 0$. Since $\{p_m, q_n\} \subseteq V' \subseteq W$, but $p = p_0 = q_0 \notin W$, there exist minimal $k, l \geqslant 1$ such that $p_k \in W$ and $q_l \in W$, and so $p \in \mathsf{WD}_G(W)$, and so W is not weakly commitment–closed, and hence not strongly commitment–closed, giving a contradiction.
- $p \in \Gamma(G, V')$ and so there is a complete path $q_0 = p, q_1, \ldots$ in G such that every $q_j \notin V'$, and since $r = p_1 \notin \Gamma(G, V')$, no $p_i = q_j$ unless $i = j = 0$. Let $k \geqslant 1$ be minimal such that $p_k \in W$. If every $q_i \notin W$, then p is not W–strongly committing, giving a contradiction, and if $q_k \in W$ for minimal $l \geqslant 1$, then there is a $[p, p_k]$ W–path and a $[p, q_l]$ W–path in G, and so W is not weakly commitment–closed, and hence not strongly commitment–closed, giving a contradiction.

Thus we have shown if an edge (p, r) in V satisfies conditions (1)–(5), then p lies in every strongly commitment-closed superset of V'.

Theorem 54 *Let $G = (V, E)$ be a finite directed graph and let $V' \subseteq V$. There exists a unique minimal strongly commitment–closed superset of V' in G.*

PROOF. We now prove the uniqueness of minimal strongly commitment–closed supersets of V' by induction on $|V| - |V'|$. We may assume that V' is not strongly commitment–closed, hence there is an edge (p, r) in V that satisfies conditions (1)–(5) of Theorem 53, and such that p lies in every strongly commitment–closed superset of V'. If $V' \cup \{p\}$ is strongly commitment–closed, then the uniqueness result follows immediately. Otherwise, by the inductive hypothesis, there is a unique minimal strongly commitment–closed superset of $V' \cup \{p\}$, and since every strongly commitment–closed superset of V' must contain p, the result follows.

In this section we have proved that for any subset of vertices V' of a CFG, there are unique minimal weak and strong commitment closed sets containing V' and hence unique minimal weak and strong projections containing V'. In Section 6, we give low polynomial algorithms for computing these sets.

6 Algorithms for computing minimal weakly and strongly commitment–closed sets

In this section we define algorithms for computing the minimal weakly and strongly commitment–closed supersets of a set. (We showed that these sets exist in the previous section.) Since weakly and strongly commitment–closedness is a necessary and sufficient condition for the induced graph to be a weak/strong projection, in effect, we have algorithms for producing minimal weak/strong projections (slices).

Formally, given a CFG $G = (V, E, \beta)$ and a subset $V' \subseteq V$, we wish to compute the minimal superset of V' which is weakly or strongly commitment–closed. As we will show in this section, this can be done with worst–case time complexity $O(|G|^4)$, where $|G| = |V| + |E|$. For CFGs, vertices have a maximum out–degree of 2, so $O(|E|) = O(|V|)$ giving $O(|G|) = O(|V|)$. This gives our algorithms a worst–case time complexity $O(|V|^4)$. This is of a very similar order to the worst–case time complexity of the algorithms for computing the new control dependences of Ranganath et al. which they give as $O(|V|^3 \times lg(|V|) \times \sum T_n) = O(|V|^4 \times lg(|V|))$. (Since they define T_n to be the number of successors of vertex, n so $O(\sum T_n) = O|V|)$).

The algorithms we present here are not intended to replace current algorithms for computing sets closed under previous known forms of control dependence, since for CFGs where the previous forms apply the results will be identical. This follows immediately from the results of Sections 7 and 8.

We are not convinced, however, that in practice, our algorithms are any more efficient (or less) than those of Ranganath et al.. We are merely demonstrating the existence of low polynomial–time algorithms for computing the required sets for the more general form of CFGs used in this paper. We suspect, however, that improvements in the efficiency of the algorithms presented here exist. This will be the subject of future work.

Definition 55 *Let $G = (V, E)$ be a graph. We define $|G| = |V| + |E|$.*

The computation of $\Theta(G, V', u)$ (Definition 52) has time complexity $O(|G|^2)$, since removing the appropriate edges from G to obtain H takes linear time, and the subsequent reachability problem has time complexity $O(|G|^2)$.

For CFGs, since the maximum out–degree for each vertex is two, we have $O(|G|) = O(|V|)$. The computation of $\Theta(G, V', u)$, thus has time complexity $O(|V|^2)$.

6.1 *An algorithm to compute the minimal weakly commitment–closed superset of V' in G*

Let $G = (V, E, \beta)$ be a CFG and let $V' \subseteq V$. We require an algorithm for computing the minimal weakly commitment–closed superset of V' in G. Our algorithm is indicated by Theorem 49, which gives a condition stated in terms of the function Θ on vertices that must be added to the set V' in order to obtain a weakly commitment–closed set. We now give an algorithm for computing the minimal weakly commitment–closed superset of V' in G which we prove has time–complexity $O(|V|^4)$. Let $G = (V, E, \beta)$ be a CFG and let $V' \subseteq V$. To compute the minimal weakly commitment–closed superset of V' in G proceed as follows:

Algorithm 56 *(1) Assign $X = V'$.*
(2) Choose any edge (p, v) in G with p reachable from V' and such that $|\Theta(G, X, v)| = 1$ and $|\Theta(G, X, p)| \geqslant 2$ hold, and assign $X = X \cup \{p\}$. If no such edge (p, v) exists then STOP.
(3) GOTO 2.

Theorem 57 *Algorithm 56 has time complexity $O(|V|^4)$, and the value of the set X when STOP is reached is the minimal weakly commitment–closed superset of V' in G.*

PROOF. Step (2) cannot execute twice with the same value of p and is therefore executed at most $|V|$ times. For each execution of (2), not more than $|G|$ edges are tested, and for each testing, the time taken is determined by Θ and thus is bounded by $O(|V|^2)$, proving the total $O(|V|^4)$ time complexity. By Theorem 49, X is the unique smallest weakly commitment–closed set in G containing V' when STOP is reached.

Later in the paper we will prove that weak commitment–closedness subsumes all the previous definitions of weak control dependence in the literature. In other words, the problem of computing the set of vertices that transitively control a set of vertices (whichever definition in the literature we use) can be reduced to computing weak commitment–closed sets. Furthermore weak commitment–closedness is more general, in the sense that it is defined for graphs for which previous definitions of control dependence are not defined.

6.2 An algorithm to compute the minimal strongly commitment–closed superset of V' in G

We now consider the analogous problem of computing the minimal strongly commitment–closed superset of a given set. First we need an algorithm for computing $\Gamma(G, V')$ (Definition 51): the set of all $x \in V$ that lie on a complete path in G which does not pass through V'.

Algorithm 58 (Algorithm for computing $\Gamma(G, V')$) *(1) Assign $X = V'$.*
(2) Choose any edge (y, x) in G with $x \in X$, $y \notin X$ and y not an incomplete predicate vertex. If no such edge exists, then STOP.

(3) Delete the edge (y, x) from G. If y is a predicate, convert it to a non–predicate (so in future iterations it is not considered incomplete).
(4) If there are no remaining edges (y, z) with $z \notin X$, then assign $X = X \cup \{y\}$.
(5) GOTO (2).

Theorem 59 *In Algorithm 58, the final value of $V \backslash X$ is precisely $\Gamma(G, V')$, and the algorithm has time complexity $O(|G|^2) = O(|V|^2)$.*

PROOF. Each execution of (3) except the last deletes an edge from G, hence the number of iterations is bounded by $O(|G|)$. Also, lines (2)–(4) have time complexity bounded by $O(|G|)$, proving the total time complexity bound given. For any execution of (2) and for the current value of X just before this execution, the edges deleted from G and the vertices added to X are not used in any complete path using only the vertices in $V \backslash X$. Thus the set of all such complete paths does not change throughout the whole execution, even as X changes. Since at the end of the execution there are no edges from $V \backslash X$ to X, the set $\Gamma(G, V')$ of vertices occurring on these paths is $V \backslash X$, proving the Theorem.

Let $G = (V, E)$ be a finite directed graph and let $V' \subseteq V$. To compute the minimal strongly commitment–closed superset of V', now proceed as follows:

Algorithm 60 *(1) Assign $X = V'$.*
(2) Find an edge (p, r) in G such that p is reachable in G from X and satisfying:
(a) $|\Theta(G, X, r)| = 1$ and
(b) $r \notin \Gamma(G, X)$ and
(c) $|\Theta(G, X, p)| \geqslant 2$ or $p \in \Gamma(G, X)$.
If no such edge exists, then STOP, else assign $X = X \cup \{p\}$.
(3) GOTO (2).

Theorem 61 *Algorithm 60 computes unique minimal strongly commitment–closed superset of V' and has worst–case time complexity $O(|V|^4)$.*

PROOF. Both parts of the proof are similar to their analogues in the proof of Theorem 57, with Theorem 53 used in place of Theorem 49, and with Theorem 59 used to bound the time complexity.

Although, it is beyond the scope of this paper, it is worth mentioning that the worst–case $O(|V|^4)$ time complexity bound of Algorithms 56 and 60 can quite possibly be improved upon.

Of course, faster algorithms can be obtained by placing additional restrictions on the graph G. However, as weak and strong commitment–closeness are generalisations of existing forms of control dependence the existing faster algorithms can be used instead in such simpler cases.

7 The weak forms of control dependence

7.1 Summary

In the literature, there are three distinct forms of control dependence which we call *weak* because, as we show in this section, vertex sets closed under them induce weak projections. They are:

- $\xrightarrow{\text{W-controls}}$: the control dependence of Weiser [34],
- $\xrightarrow{\text{F-controls}}$: the control dependence of Ferrante et al. [17], and
- $\xrightarrow{\text{WOD}}$: Amtoft's weak order dependence [3].

There is also Podgurski and Clarkes' *strong control dependence* [27], but this is merely a paraphrasing of Weiser's.

The main results of this section give the relationship between sets closed under the weak forms of control dependence mentioned above and weakly commitment–closed sets. These can be summarised as follows:

Lemma 74 Let $G = (V, E)$ be an $\{\mathsf{end}\}$–graph. V' is closed under $\xrightarrow{\text{F-controls}}$ if and only if V' is closed under $\xrightarrow{\text{W-controls}}$.

Lemma 75 Let $G = (V, E)$ be a $\{\mathsf{start}, \mathsf{end}\}$–graph with $\{\mathsf{start}, \mathsf{end}\} \subseteq V' \subseteq V$. If V' is weakly commitment–closed in G then V' is closed under $\xrightarrow{\text{W-controls}}$.

Lemma 77 Let $G = (V, E)$ be an $\{\mathsf{end}\}$–graph with $V' \subseteq V$. If V' is closed under $\xrightarrow{\text{W-controls}}$ then V' is weakly commitment–closed in G.

Lemma 78 Let $G = (V, E)$ be a finite directed graph with $V' \subseteq V$. If V' is closed under $\xrightarrow{\text{WOD}}$ then V' is weakly commitment–closed in G.

Lemma 79 Let $G = (V, E)$ be a $\{\mathsf{start}\}$–graph with $\mathsf{start} \in V' \subseteq V$. If V' is weakly commitment–closed in G then V' is closed under $\xrightarrow{\text{WOD}}$.

From these, we can prove our main result which shows that, indeed, all weak forms of control dependence in the literature induce weak projections. These forms of control dependence have thus been *semantically* characterised for the first time. The characterisation is as follows:

Theorem 62 (Main theorem for weak control dependence)

(1) If G is a $\{\mathsf{start}, \mathsf{end}\}$–CFG with $\{\mathsf{start}, \mathsf{end}\} \subseteq V'$, then V' is closed under $\xrightarrow{\text{W-controls}}$ if and only if the induced graph induced by V' from G is a weak projection of G.
(2) If G is a $\{\mathsf{start}, \mathsf{end}\}$–CFG with $\{\mathsf{start}, \mathsf{end}\} \subseteq V'$, then V' is closed under $\xrightarrow{\text{F-controls}}$ if and only if the induced graph induced by V' from G is a weak projection of G.
(3) If G is a $\{\mathsf{start}\}$–CFG with $\mathsf{start} \in V'$ then V' is closed under $\xrightarrow{\text{WOD}}$ if and only if the induced graph induced by V' from G is a weak projection of G.

PROOF. (1) follows from Lemmas 74, 75, 77 and Theorem 41. (2) follows from (1) and Lemma 74 and, (3) follows from Lemmas 78 and 79 together with Theorem 41.

It can be seen from Theorem 62, that each form of weak control dependence requires a restriction to the CFG for it to be characterised by a weak projection. As Theorem 41 showed, weak commitment closedness, on the other hand, requires no such restriction and can thus be more generally applied.

7.2 Weiser's control dependence

In order to define Weiser's control dependence, we first need *forward domination.*

Definition 63 (Forward domination) *Let $G = (V, E)$ be an $\{\mathsf{end}\}$–graph and let $v, w \in V$. If every path from v to* end *passes through w then w* forward dominates *v.*

Forward domination is the terminology of Podgurski and Clarke. Weiser calls it *inverse domination.* Ferrante et al. [17] call it *post domination.*

$$\begin{tikzcd}
 & & w_1 \arrow[rd] & \\
 & v_1 \arrow[ru] \arrow[r] & w_2 \arrow[r] & \mathsf{end} \\
v \arrow[ru] \arrow[rd] & & & \\
 & v_2 \arrow[rruu, bend right] & &
\end{tikzcd}$$

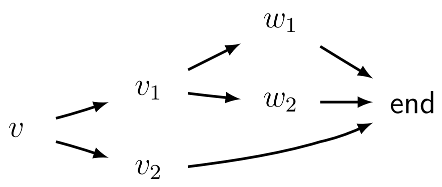


Fig. 11. Ferrante et al. control dependence is not transitive: $v \xrightarrow{\text{F-controls}} v_1$ and $v_1 \xrightarrow{\text{F-controls}} w_1$ but $v \xrightarrow{\text{F-controls}} w_1$ is false.

It is well known [27, Theorem 1] that if $v \neq \mathsf{end}$ is a vertex in a CFG, then the set of all vertices that forward dominate v always occur in the same order on any path from v to end. We call the first such vertex apart from v the *nearest forward dominator* of v.

Definition 64 (ND) *Let $G = (V, E)$ be an $\{\mathsf{end}\}$–graph and $v \in V$. $\mathsf{ND}(v)$ is the set of vertices which lie on a path from v to its nearest forward dominator b, excluding v and b themselves.*

Implicit in this is Weiser's definition of control dependence:

Definition 65 ($\xrightarrow{\textbf{W-controls}}$) *Let $G = (V, E)$ be an $\{\mathsf{end}\}$–graph and $v, w \in V$, then $v \xrightarrow{\text{W-controls}} w$ if and only if $w \in \mathsf{ND}(v)$.*

Note that in v cannot control itself using Weiser's definition.

7.3 The control dependence of Ferrante et al.

In defining the *program dependence graph*, control dependence was once again redefined [17].

Definition 66 ($\xrightarrow{\textbf{F-controls}}$) *Let $G = (V, E)$ be an $\{\mathsf{end}\}$–graph, then $v \xrightarrow{\text{F-controls}} w$ if and only if v is not forward dominated by w, and there exists a path π from v to w such that for all vertices z occurring on π apart from v is forward dominated by w.*

Ferrante et al. control dependence is not transitive, as is demonstrated by the example in Figure 11 where $v \xrightarrow{\text{F-controls}} v_1$ and $v_1 \xrightarrow{\text{F-controls}} w_1$ but v does not control w_1.

Lemma 67, in effect, allows an alternative definition of Ferrante et al. control dependence, which is often used:

Lemma 67 *Let $G = (V, E)$ be an $\{\mathsf{end}\}$–graph and $v, w \in V$. The vertex $v \xrightarrow{\text{F-controls}} w$ if and only if v has immediate successors v_1, v_2 such that w forward dominates v_1 but not v_2.*

PROOF. Suppose that w is Ferrante control dependent on v. Then there is a path π from v to w on which all vertices except v are forward dominated by w, and there is a path $v\mu\mathsf{end}$ which does not pass through w. Let v_1 be the second vertex of π then $v\mu\mathsf{end}$ also does not pass through v_1 because w forward dominates v_1. Therefore the first vertex of μ must be some $v_2 \neq v_1$ and so v has immediate successors v_1, v_2 with w forward dominating v_1 but not v_2, as required.

Conversely, let v have immediate successors v_1, v_2 such that w forward dominates v_1 but not v_2. Since w forward dominates v_1, and end is reachable from v_1, there is a path $v_1\rho\mathsf{end}$ on which w occurs, and which hence has a prefix $v_1\nu w$ on which w does not occur except at the end. Thus every vertex on $v_1\nu$ is forward dominated by w. However since w does not forward dominate v_2, and there is an edge (v, v_2) and $w \neq v$, w does not forward dominate v either, and thus $v \xrightarrow{\text{F-controls}} w$ as required.

7.4 Amtoft's weak–order dependence

Amtoft [3,30] observes that the traditional CFG is not well adapted to handle modern programming constructs which may intentionally fail to terminate, *e.g.,* reactive systems. Amtoft addresses this problem by define weak order dependence ($\xrightarrow{\text{WOD}}$) as an extension of Ferrante et al. control dependence to handle CFGs which are not necessarily $\{\mathsf{end}\}$–graphs.

Definition 68 ($\xrightarrow{\textbf{WOD}}$) *Let $G = (V, E)$ be a finite directed graph with $v, b, c \in V$. Then $v \xrightarrow{\text{WOD}} b, c$ if and only if:*

(1) There is a path from v to b not containing c.
(2) There is a path from v to c not containing b.
(3) v has an immediate successor a such that either
- *b is reachable from a, and all paths from a to c contain b; or*
- *c is reachable from a, and all paths from a to b contain c.*

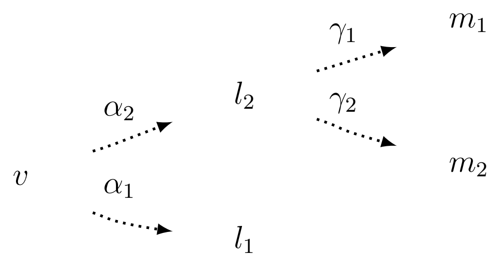


Fig. 12. Lemma 69: If v weakly decides between an element of V' and an element of $\mathsf{WD}_G(V')\backslash V'$ then $v \in \mathsf{WD}_G(V')$. PROOF: There exist two finite proper $\{l_1, l_2\}$–paths, $v\alpha_1 l_1$ and $v\alpha_2 l_2$ in G that both start at v and have no other common vertex. Choose $l_1 \in V'$ and $l_2 \in \mathsf{WD}_G(V')\backslash V'$ such that $|\alpha_1| + |\alpha_2|$ is minimal where $\alpha_1 l_1$ and $\alpha_2 l_2$ are disjoint. By definition of WD, there are two finite proper V'–paths $l_2\gamma_1 m_1$ and $l_2\gamma_2 m_2$ in G with $\{m_1, m_2\} \subseteq V'$ such that $\gamma_1 m_1$ and $\gamma_2 m_2$ are disjoint. Furthermore we can assume that l_2 is not in either γ_1 or γ_2 (since if it is we can simply create smaller paths by removing the cycles.)

7.5 The relationship between controlling predicates and weakly deciding predicates

There is a strong connection between sets closed under the different forms of weak control dependence and the sets closed under WD_G, *i.e.,* those sets V' for which $\mathsf{WD}_G(V') \subseteq V'$. In order to prove the main results of this section we first prove an important property of WD_G which we refer to as *idempotence.* This means that:

$$\mathsf{WD}_G(V' \cup \mathsf{WD}_G(V')) \subseteq V' \cup \mathsf{WD}_G(V').$$

This is equivalent to saying $V' \cup \mathsf{WD}_G(V')$ is closed with respect to WD_G *i.e.,* if we take the set of vertices that are weakly deciding on $V' \cup \mathsf{WD}_G(V')$ we will not get any new elements. We need a preliminary lemma which we use to prove idempotence of WD_G.

Lemma 69 *Let $l_1 \in V'$ and $l_2 \in \mathsf{WD}_G(V')\backslash V'$. Then*

$$\mathsf{WD}_G(\{l_1, l_2\}) \subseteq \mathsf{WD}_G(V').$$

PROOF. There exist two finite proper $\{l_1, l_2\}$–paths, $v\alpha_1 l_1$ and $v\alpha_2 l_2$ in G that both start at v and have no other common vertex. Choose $l_1 \in V'$ and $l_2 \in \mathsf{WD}_G(V')\backslash V'$ such that $|\alpha_1| + |\alpha_2|$ is minimal where $\alpha_1 l_1$ and $\alpha_2 l_2$ are disjoint. By definition of WD, there are two finite proper V'–paths $l_2\gamma_1 m_1$ and $l_2\gamma_2 m_2$ in G with $\{m_1, m_2\} \subseteq V'$ such that $\gamma_1 m_1$ and $\gamma_2 m_2$ are disjoint. (See

$$\begin{tikzcd}
 & & & w \arrow[r, dotted, "\tau"] & m_1 \\
 & w \arrow[r, dotted, "\nu"] & l_2 \arrow[ur, dotted, "\sigma"] \arrow[dr, dotted, "\gamma_2"] & & \\
v \arrow[ur, dotted, "\mu"] \arrow[dr, dotted, "\alpha_1"] & & & m_2 & \\
 & l_1 & & &
\end{tikzcd}$$

Fig. 13. Proof of Lemma 69 Part (1):Without loss of generality, assume γ_1 contains an element of α_2. Let w be the last vertex in α_2 which also occurs in γ_1. Thus we may write $\alpha_2 = \mu w \nu$ and $\gamma_1 = \sigma w \tau$, where no vertex in ν occurs in γ_1.

$$\begin{tikzcd}
 & & m_1 & \\
 & w \arrow[ur, dotted, "\tau"] \arrow[dr, dotted, "\nu"] & & \\
v \arrow[ur, dotted, "\mu"] \arrow[dr, dotted, "\alpha_1"] & & l_2 \arrow[r, dotted, "\gamma_2"] & m_2 \\
 & l_1 & &
\end{tikzcd}$$

Fig. 14. Proof of Lemma 69 Part (1): The paths τ and ν are disjoint since no vertex in ν occurs on γ_1 and τ is inside γ_1. Also τ and γ_2 are disjoint because τ is inside γ_1, and γ_1 and γ_2 are disjoint. Finally $l_2 \notin \tau$ because τ is inside γ_1.

$$\begin{tikzcd}
 & z \arrow[r, dotted] & l_1 & \\
v \arrow[ur, dotted, "\mu"] \arrow[dr, dotted, "\alpha_2"] & & & \\
 & l_2 \arrow[r, dotted] \arrow[dr, dotted, "\gamma_2"] & z \arrow[r, dotted, "\tau"] & m_1 \\
 & & m_2 &
\end{tikzcd}$$

Fig. 15. Proof of Lemma 69 Part 2(b): Now assume that α_1 and one of the γ_i have a common vertex. Let z be the first vertex in α_1 that shares an element with the γ_i. Again, without loss of generality, assume $z \in \gamma_1$. *i.e.,* $\alpha_1 = \mu z \nu$ and $\gamma_1 = \sigma z \tau$, where no vertex in μ occurs in either of the γ_i.

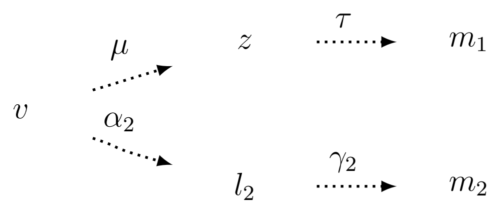


Fig. 16. Proof of Lemma 69, part 2 (b): Part (a) has shown that $z\tau$, does not share a vertex with $\alpha_2 l_2 \gamma_2$, (since $z\tau$ is a path within γ_1, $l_2 \notin \gamma_2$ and the γ_i are disjoint). Also, because of the choice of z, the only vertex in shared between $\alpha_2 l_2 \gamma_2$ and μ, is v. Therefore $v \in \mathsf{WD}_G(V')$ witnessed by the paths $\mu z \tau$ and $\alpha_2 l_2 \gamma_2$, as required.

Figure 12). Furthermore we can assume that l_2 is not in either γ_1 or γ_2 (since if it is we can simply create smaller paths by removing the cycles).

(1) We first prove that both the γ_i are disjoint from α_2. Suppose not, then without loss of generality assume that γ_1 contains an element of α_2. Let w be the last vertex in α_2 which also occurs in γ_1. Thus we may write $\alpha_2 = \mu w \nu$ and $\gamma_1 = \sigma w \tau$, where no vertex in ν occurs in γ_1 (see Figure 13).

From Figure 14, we see that the paths τ and ν are disjoint since no vertex in ν occurs on γ_1 and τ is inside γ_1. Also τ and γ_2 are disjoint because τ is inside γ_1, and γ_1 and γ_2 are disjoint. Finally $l_2 \notin \tau$ because τ is inside γ_1.

So $w \in \mathsf{WD}_G(\{m_1, m_2\})$ witnessed by the paths $w\tau m_1$ and $w\nu l_2 \gamma_2 m_2$. m_1 and m_2 are in V' so by Lemma 48, $w \in \mathsf{WD}_G(V')$. Also γ_1 is disjoint from V' so $w \notin V'$ since $w \in \gamma_1$. So $w \in WD(V', G) \backslash V'$ just like l_2.

So could have chosen w instead of l_2 with the required property and since the length of μ is less than the length of α_2 contradicting our choice of l_2. So both the γ_i are disjoint from α_2.

(2) We now prove that $v \in \mathsf{WD}_G(V')$.

(a) If α_1 and neither of the paths γ_i have a common vertex, then, again by Lemma 48, since $\{l_1, m_1\} \subseteq V'$, $v \in \mathsf{WD}_G(V')$ witnessed by the paths $v\alpha_1 l_1$ and $v\alpha_2 l_2 \gamma_1 m_1$ as required where, without lost of generality, we assume that $l_1 \neq m_1$.

(b) Now assume that α_1 and one of the γ_i have a common vertex. Let z be the first vertex in α_1 that shares an element with the γ_i. Again, without loss of generality, assume $z \in \gamma_1$. *i.e.*, $\alpha_1 = \mu z \nu$ and $\gamma_1 = \sigma z \tau$, where no vertex in μ occurs in either of the γ_i (see Figure 15).

Part (a) has shown that $z\tau$, does not share a vertex with $\alpha_2 l_2 \gamma_2$, (since $z\tau$ is a path within γ_1, $l_2 \notin \gamma_2$ and the γ_i are disjoint). Also, because of the choice of z, the only vertex in shared between $\alpha_2 l_2 \gamma_2$ and μ, is v. Therefore $v \in \mathsf{WD}_G(V')$ witnessed by the paths $\mu z \tau$ and $\alpha_2 l_2 \gamma_2$, (see Figure 16) as required.

Lemma 70 (WD_G is idempotent) *Let G be a* CFG *with vertex set V and let $V' \subseteq V$. Write $M = V' \cup \mathsf{WD}_G(V')$. Then $\mathsf{WD}_G(M) \subseteq M$.*

PROOF. Let $v \in \mathsf{WD}_G(M)\backslash V'$. Thus there are $[v, l_i]$ M–paths α_i in G for $i = 1, 2$ with each $l_i \in M$ which do not have a common vertex apart from the initial one, v. We may assume that the sum of the lengths of the paths α_i is minimal. We will show that $v \in \mathsf{WD}_G(V')$, thus proving the Lemma. We consider three cases:

(1) If both $l_1, l_2 \in V'$ then clearly $v \in \mathsf{WD}_G(V')$, as required.
(2) If $l_1 \in V'$ and $l_2 \in \mathsf{WD}_G(V')\backslash V'$ then the result follows immediately from Lemma 69.
(3) If both $l_1, l_2 \in \mathsf{WD}_G(V')\backslash V'$ (see Figure 17) then we prove first that α_2 and γ_{11} are disjoint.

Suppose not and let w be the last element of α_2 which shares an element with γ_{11}. Therefore w weakly decides between m_{11} and l_2 and hence by Lemma 69, $w \in \mathsf{WD}_G(V')$. This contradicts that α_2 is an M–path. Therefore α_2 and γ_{11} are disjoint which means that v weakly decides between m_{11} and l_2 and hence, again by Lemma 69, $v \in \mathsf{WD}_G(V')$ as required.

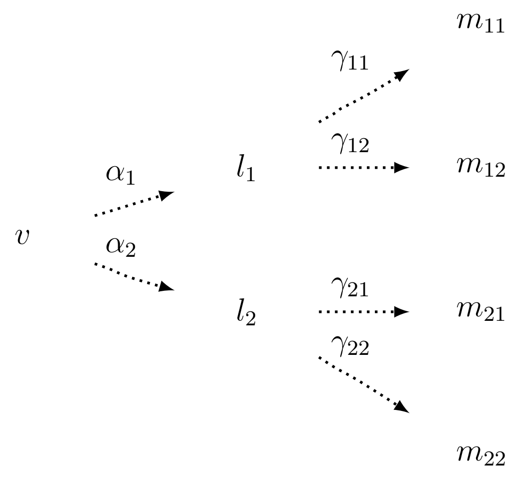


Fig. 17. Proof of Lemma 70.

Having proved that WD_G is idempotent, we are now in a position to prove the main results of this section.

7.6 *Characterisation of sets closed under different weak forms of control dependence*

The following theorem in essence has already appeared [7]. We restate and prove it here for completeness.

Lemma 71 *Let $G = (V, E)$ be an $\{\mathsf{end}\}$–graph and let $p, v \in V$ with $p \neq v$, then $p \xrightarrow{\text{F-controls}^*} v$ if and only if p is $\{v, \mathsf{end}\}$–weakly deciding in G.*

PROOF. Suppose that $p \xrightarrow{\text{F-controls}^*} v$. Thus there is a sequence $v = p_0, \ldots, p_m = p$ with $m \geqslant 1$ such that $p_i \xrightarrow{\text{F-controls}} p_{i-1}$. We prove $p \in \mathsf{WD}_G(\{v, \mathsf{end}\})$ by induction on m.

If $m = 1$ then $p \xrightarrow{\text{F-controls}} v$. Since end is reachable from every vertex, by Theorem 67, p has immediate successors p_1, p_2 such that there is a $[p_1, v]$ path $\alpha_1 v$ containing v only at the end and a $[p_2, v]$ path $\alpha_2\mathsf{end}$ not containing v, and every path from p_1 (and hence from any vertex in α_1) to end passes through v. Thus no vertex occurs on both paths $\alpha_1 v$ and $\alpha_2\mathsf{end}$, hence $p \in \mathsf{WD}_G(\{v, \mathsf{end}\})$ follows.

If $m > 1$, the inductive hypothesis gives us

$$p_{m-1} \in \mathsf{WD}_G(\{v, \mathsf{end}\}),$$

and since $p_m \xrightarrow{\text{F-controls}} p_{m-1}$, we have just shown that

$$p = p_m \in \mathsf{WD}_G(\{p_{m-1}, \mathsf{end}\})$$

follows. Thus by Lemma 48, we may replace $\{p_{m-1}, \mathsf{end}\}$ by its superset

$$\mathsf{WD}_G(\{v, \mathsf{end}\}) \cup \{v, \mathsf{end}\}$$

to get

$$\begin{aligned} p &\in \mathsf{WD}_G(\mathsf{WD}_G(\{v, \mathsf{end}\}) \cup \{v, \mathsf{end}\}) \\ &\subseteq \mathsf{WD}_G(\{v, \mathsf{end}\}) \cup \{v, \mathsf{end}\} \end{aligned}$$

by Lemma 70. Since clearly $p \notin \{v, \mathsf{end}\}$, the result $p \in \mathsf{WD}_G(\{v, \mathsf{end}\})$ follows, as required.

Conversely, suppose that $w \in \mathsf{WD}_G(\{v, \mathsf{end}\})$ then there are paths $w\alpha_1 v$ and $w\alpha_2\mathsf{end}$ in V which are disjoint other than w. We will prove $w \xrightarrow{\text{F-controls}^*} v$ by induction on the length of the path α_1. If end is not reachable in $V \backslash \{v\}$ from any vertex on α_1, then $w \xrightarrow{\text{F-controls}} v$ is immediate. Thus we may write

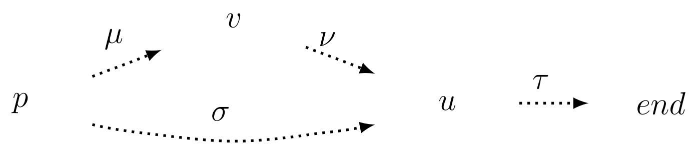


Fig. 18. Lemma 72.

$\alpha_1 = \mu u \nu$ where there is a path $u\rho\mathsf{end}$ in V that does not pass through v. Assume u is the last vertex on α_1 from which end is reachable in $V\backslash\{v\}$, and so ν has no vertex in common with α_2. Therefore ρ and ν have no common vertex, and so $u \in \mathsf{WD}_G(\{v, \mathsf{end}\})$ and hence by induction $u \xrightarrow{\text{F-controls}^*} v$.

Finally, $w \in \mathsf{WD}_G(\{u, \mathsf{end}\})$ and hence by induction $w \xrightarrow{\text{F-controls}^*} u$ and thus $w \xrightarrow{\text{F-controls}^*} v$.

Lemma 72 *Let $G = (V, E)$ be an $\{\mathsf{end}\}$–graph, then $p \xrightarrow{\text{W-controls}} v$ if and only if p is $\{v, \mathsf{end}\}$–weakly deciding in G.*

PROOF. Let u be the nearest forward dominator of p in G. Suppose that $v \in \mathsf{ND}(p)$. Then there is a path $p\mu v \nu u$ on which u does not occur except at the end, and so v does not forward dominate p, hence there is also a path $v\sigma u$ not passing through v (see Figure 18).

If every path from p to u shares a vertex with μ then v forward dominates u and so u is not p's nearest forward dominator. Now, u forward dominates every vertex occurring on μv because otherwise u would not forward dominate p. Since end is reachable from u there is a path $u\tau\mathsf{end}$ with $\tau\mathsf{end}$ not passing through u or, hence, any vertex forward dominated by u. Thus the paths $p\mu v$ and $p\sigma u \tau\mathsf{end}$ have no common vertex except for p, and so p is $\{v, \mathsf{end}\}$–weakly deciding in G..

Conversely, suppose that p is $\{v, \mathsf{end}\}$–weakly deciding in G. Then there are paths $p\mu\mathsf{end}$ and $p\nu v$ which share only p as a common vertex. There is also a path $v\pi\mathsf{end}$. Clearly u occurs on both paths $p\mu\mathsf{end}$ and $p\nu v\pi\mathsf{end}$, and the 'non-sharing' property implies that u occurs on $\pi\mathsf{end}$, proving $v \in \mathsf{ND}(p)$.

Lemma 73 *Let $G = (V, E)$ be an $\{\mathsf{end}\}$–graph and let $p, v \in V$ with $p \neq v$, then $p \xrightarrow{\text{F-controls}^*} v$ if and only if $p \xrightarrow{\text{W-controls}} v$.*

PROOF. By Theorem 71 $p \xrightarrow{\text{F-controls}^*} v$ if and only if p is $\{v, \mathsf{end}\}$–weakly deciding in G. By Lemma 72 p is $\{v, \mathsf{end}\}$–weakly deciding in G if and only if

$$\begin{tikzcd} & v_1 \arrow[dr] & \\ v \arrow[ur] \arrow[dr] & & \mathsf{end} \\ & v_2 \arrow[ur] & \end{tikzcd}$$

Fig. 19. Necessity that $\mathsf{start} \in V'$ in Theorem 75: Here v controls both v_1 and v_2, so $\{v_1, v_2\}$ is not closed under control dependence, but $\{v_1, v_2\}$ is weakly commitment–closed in G

$p \xrightarrow{\text{W-controls}} v$.

From Lemma 73, it now follows immediately that:

Lemma 74 *Let $G = (V, E)$ be an $\{\mathsf{end}\}$–graph and let $V' \subseteq V$. V' is closed under $\xrightarrow{\text{F-controls}}$ if and only if V' is closed under $\xrightarrow{\text{W-controls}}$.*

7.7 Sets closed under weak forms of control dependence and weak commitment

Lemma 75 *Let $G = (V, E)$ be a $\{\mathsf{start}, \mathsf{end}\}$–graph and let $\{\mathsf{start}, \mathsf{end}\} \subseteq V' \subseteq V$.*
If V' is weakly commitment–closed in G then V' is closed under $\xrightarrow{\text{F-controls}}$ (and by Lemma 74, $\xrightarrow{\text{W-controls}}$).

PROOF. Suppose not, then, by Theorem 67, there are vertices v, w with $v \in V \backslash V'$ and $w \in V'$ such that $v \xrightarrow{\text{F-controls}} w$. Thus v has distinct immediate successors z_1, z_2 such that there are paths $z_1\rho_1 w$ and $z_2\rho_2\mathsf{end}$, where w does not occur on either ρ_i and end is not reachable in $V\backslash\{w\}$ from z_1. Thus v is $\{\mathsf{end}, w\}$–weakly deciding in G. By Lemma 48 v is V'–weakly deciding in G. But v is reachable from $\mathsf{start} \in V'$ therefore by Lemma 47, V' is not weakly commitment–closed in G.

Note that the condition that $\mathsf{start} \in V'$ really is necessary, as the example in Figure 19 shows.

Furthermore, the condition that $\mathsf{end} \in V'$ cannot be dispensed with. In Figure 20, let $V' = \{\mathsf{start}, w\}$. V' is weakly commitment–closed in G but v controls w so V' is not closed under control dependence.

$$\begin{tikzcd}\mathsf{start} \arrow[r] & v \arrow[rr] \arrow[dr] & & \mathsf{end} \\ & & w \arrow[ur] & \end{tikzcd}$$

Fig. 20. Necessity that $\mathsf{end} \in V'$ in Theorem 75: Here if $V' = \{\mathsf{start}, w\}$ then V' is weakly commitment–closed in G but v controls w, so V' is not closed under control dependence.

Lemma 76 *Let $G = (V, E)$ be an $\{\mathsf{end}\}$–graph and let $V' \subseteq V$. Suppose that $w \in V \backslash V' \cap \mathsf{WD}_G(V')$. Then there exists $w_1 \in V \backslash V'$ and $v \in V'$ such that w_1 is $\{v, \mathsf{end}\}$–weakly deciding in G.*

PROOF. Since $w \in \mathsf{WD}_G(V')$, for $i \in \{1, 2\}$ there are proper V'–paths $w\rho_i z_i$ for vertices z_i such that no vertex lies on both paths $\rho_i z_i$. There is also a path $z_1\sigma\mathsf{end}$ through G which does not pass more than once through z_1. By replacing z_1 by z_2 if necessary, and using a suffix of $\sigma\mathsf{end}$, we may assume also that $\sigma\mathsf{end}$ does not pass through z_2. If no vertex on $\sigma\mathsf{end}$ also occurs on ρ_2, then the conclusion follows for $w = w_1$ and $v = z_2$ by examining the paths $w\rho_1 z_1\sigma\mathsf{end}$ and $w\rho_2 z_2$. Otherwise we may write $\rho_2 = \alpha w_1 \beta$ and $\sigma = \tau w_1 \omega$, where $w_1 \in V$ and no vertex occurs on both β and ω, and the conclusion again follows for $v = z_2$ by examining the paths $w_1\beta z_2$ and $w_1\omega\mathsf{end}$.

Lemma 77 *Let $G = (V, E)$ be an $\{\mathsf{end}\}$–graph and let $V' \subseteq V$.*
If V' closed under $\xrightarrow{\text{F-controls}}$ then V' is weakly commitment–closed in G.

PROOF. Suppose that V' is not weakly commitment–closed in G, then by Lemma 47 $V \backslash V'$ contains an element of $\mathsf{WD}_G(V')$. By Lemma 76 there exists $\bar{w} \in V \backslash V'$ and $v \in V'$ such that $z \in \mathsf{WD}_G(\{v, \mathsf{end}\})$, and hence by Theorem 71 $z \xrightarrow{\text{F-controls}^*} v$. Hence V' is not closed under $\xrightarrow{\text{F-controls}}$.

Lemma 78 *Let $G = (V, E)$ be a finite directed graph and let $V' \subseteq V$.*
If V' is closed under $\xrightarrow{\text{WOD}}$ then V' is weakly commitment–closed in G.

PROOF. Suppose not, then there exists $x \in V'$ which has an immediate successor $y \in V \backslash V'$ which is not weakly–committing. Therefore there exist proper V'–paths $y\alpha_1 w_1$ and $y\alpha_2 w_2$ for distinct vertices $w_1 \neq w_2$. Let Ω be the set of all vertices from which w_2 is reachable in $V \backslash \{w_1\}$; thus $y \in \Omega$, but $w_1 \notin \Omega$. Thus we may write $y\alpha_1 w_1 = \beta v \gamma w_1$, where $v \in \Omega$ and no vertex in γw_1 lies in Ω. Hence $v \xrightarrow{\text{WOD}} w_1, w_2$ which contradicts that V' is closed under weak order dependence.

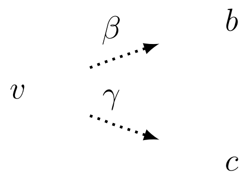


Fig. 21. Theorem 79.

Lemma 79 *Let $G = (V, E)$ be a $\{\mathsf{start}\}$–graph with $\mathsf{start} \in V'$.
If V' is weakly commitment–closed in G then V' is closed under $\xrightarrow{\text{WOD}}$.*

PROOF. Suppose that $b, c \in V'$ and $v \xrightarrow{\text{WOD}} b, c$ holds for $v \in V$. We will assume that $v \notin V'$ and deduce a contradiction. From the definition of weak order dependence and by interchanging b and c if necessary, there are paths $v\beta b$ and $v\gamma c$ such that b does not occur on γc, nor c on βb, and all paths from the first vertex of βb to c pass through b before reaching c, which implies that no vertex occurs on both βb and γc. See Figure 21. So $v \in \mathsf{WD}_G(V')\backslash V'$. Vertex v is reachable from $\mathsf{start} \in V'$ contradicting the assumption that V' is weakly commitment–closed in G (by Lemma 47).

Note that the condition that $\mathsf{start} \in V'$ really is necessary: in Figure 19 $v \xrightarrow{\text{WOD}} v_1, v_2$ so $\{v_1, v_2\}$ is not closed under $\xrightarrow{\text{WOD}}$ but $\{v_1, v_2\}$ is weakly commitment–closed in G.

We have shown that all weak forms of control dependence in the literature are essentially the same: vertex sets closed under them all induce weak projections. We may call any relation on vertex sets that has this property *weak control dependence*. So $\xrightarrow{\text{W-controls}}$, $\xrightarrow{\text{F-controls}}$ and $\xrightarrow{\text{WOD}}$ are all examples of weak control dependence. In the next section, we turn our attention to the strong forms of control dependence.

8 The strong forms of control dependence

8.1 Summary

In the literature, there are two distinct forms of control dependence which we call *strong* because, as we show in this section, vertex sets closed under them induce strong projections. These are:

- the combination of $\xrightarrow{\text{NTSCD}}$ and $\xrightarrow{\text{DOD}}$ of Ranganath et al. [30].
- $\xrightarrow{\text{PC-weak}}$, the weak control dependence of Podgurski and Clarke [27].

The main results of this section give the relationship between sets closed under the strong forms of control dependence mentioned above and strongly commitment–closed sets. These can be summarised as follows:

Lemma 87 Let $G = (V, E, \beta)$ be a complete CFG. If $V' \subseteq V$ is closed under both $\xrightarrow{\text{NTSCD}}$ and $\xrightarrow{\text{DOD}}$ then V' is strongly commitment–closed in G.

Lemma 88 Let $G = (V, E)$ be $\{\mathsf{start}\}$–graph and $\mathsf{start} \in V' \subseteq V$. If V' is strongly commitment–closed in G then V' is closed under both $\xrightarrow{\text{NTSCD}}$ and $\xrightarrow{\text{DOD}}$.

Lemma 90 Let $G = (V, E, \beta)$ be a complete $\{\mathsf{end}\}$–CFG and $V' \subseteq V$. If V' is closed under $\xrightarrow{\text{PC-weak}}$ then V' is strongly commitment–closed in G.

Lemma 91 Let $G == (V, E)$ be a $\{\mathsf{start}, \mathsf{end}\}$–graph with $\mathsf{start} \in V' \subseteq V$. If V' is strongly commitment–closed in G then V' is closed under $\xrightarrow{\text{PC-weak}}$.

From these, we can prove our main result which shows that, indeed, both strong forms of control dependence in the literature induce strong projections. These forms of control dependence have thus been *semantically* characterised for the first time. The characterisation is as follows:

Theorem 80 (Main theorem for strong control dependence)

(1) If G is a complete $\{\mathsf{start}\}$–CFG with $\mathsf{start} \in V'$ then V' is closed under $\xrightarrow{\text{DOD}}$ and $\xrightarrow{\text{NTSCD}}$ if and only if the induced graph induced by V' from G is a strong projection of G.
(2) If G is a complete $\{\mathsf{start}, \mathsf{end}\}$–CFG with $\mathsf{start} \in V'$ then V' is closed under $\xrightarrow{\text{PC-weak}}$ if and only if the induced graph induced by V' from G is a strong projection of G.

PROOF. (1) follows from Lemmas 87, 88 and Theorem 45 and

(2) follows from Lemmas 90, 91 and Theorem 45.

It can be seen from Theorem 80, that each form of strong control dependence requires a restriction to the CFG for it to be characterised by a strong projection. As Theorem 45 showed, strong commitment–closedness, on the other hand, requires no such restriction and can thus be more generally applied.

Ranganath et al. [30] observe that, despite being widely used, existing definitions and approaches to calculating control dependence are difficult to apply directly to modern program structures which make substantial use of exception processing and which may deliberately run indefinitely. A major motivation of Ranganath's work is that traditional forms of control require the `end` to be reachable from every vertex. They rightly claim that this is not a suitable restriction for such programs which are designed to non–terminate. So they, like us, allow CFGs where `end` is not necessarily reachable from every vertex.

For these sort of programs, they argue that the slice should non–terminate in all initial states when the original does. In order to compute this strong form of slice, Ranganath et al. define two new control dependence relations:

- Non–termination–sensitive control dependence, $\xrightarrow{\text{NTSCD}}$.
- Decisive order dependence (sometimes referred to as *direct order dependence*), $\xrightarrow{\text{DOD}}$.

In the definitions below, reproduced from their work [30], the term *maximal path* refers to a path that either is infinite or ends at `end`. The definitions still make sense for the more general class of graphs under investigation in this paper where a *maximal path* is taken as the path $\bar{\omega}$ of a maximal walk ω (see Definitions 13 and 19). By Proposition 21 both of these notions of *maximal path* coincide in CFGs where all predicates are complete.

Definition 81 **($\xrightarrow{\text{DOD}}$)** *Let $G = (V, E)$ be a finite directed graph, then $v \xrightarrow{\text{DOD}} b, c$ if and only if:*

(1) All maximal paths from v contain both b and c.
(2) v has an immediate successor from which all maximal paths contain b before any occurrence of c.
(3) v has an immediate successor from which all maximal paths contain c before any occurrence of b.

Definition 82 **($\xrightarrow{\text{NTSCD}}$)** *Let $G = (V, E)$ be a finite directed graph, then $v \xrightarrow{\text{NTSCD}} w$ if and only if:*

(1) v has at least two immediate successors.
(2) w occurs on all maximal paths from one of these immediate successors.
(3) there is a maximal path from another immediate successor which does not contain w.

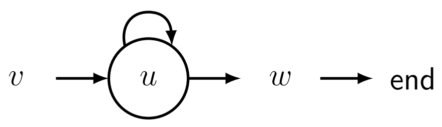

$$\begin{tikzcd} v \arrow[r] & u \arrow[loop above] \arrow[r] & w \arrow[r] & \mathsf{end} \end{tikzcd}$$

Fig. 22. Vertex w forward dominates v but does not strongly forward dominate v.

8.3 Podgurski–Clarke weak control dependence

Definition 83 (Strong forward domination) *Let G be an $\{\mathsf{end}\}$–graph. A vertex w* strongly forward dominates *a vertex v if and only if w forward dominates v and there exists an $n \in \mathbb{N}$ such that every path of length n from v contains w.*

Strong forward domination is a properly stronger condition than forward domination as can be seen by the diagram in Figure 22.

Definition 84 ($\xrightarrow{\textbf{PC-weak}}$) *Let $G = (V, E)$ be an $\{\mathsf{end}\}$–graph, then $v \xrightarrow{\text{PC-weak}} u$ if and only if:*

(1) v has at least two immediate successors w_1 and w_2.
(2) u strongly forward dominates w_1 but does not strongly forward dominate w_2.

In $\{\mathsf{end}\}$–graphs $\xrightarrow{\text{PC-weak}}$ and $\xrightarrow{\text{NTSCD}}$ are equivalent. Ranganath et al. [30] prove a similar result (Theorem 3 (Coincidence Properties, II)). We shall show in Lemma 85 that $\xrightarrow{\text{NTSCD}}$ (by itself) is equivalent to Podgurski–Clarke weak control dependence in $\{\mathsf{end}\}$–graphs.

Lemma 85 *Let $G = (V, E)$ be an $\{\mathsf{end}\}$–graph. Then for all $u, v \in V$, $v \xrightarrow{\text{PC-weak}} u$ if and only if $v \xrightarrow{\text{NTSCD}} u$.*

PROOF. Observe that since end is reachable from every vertex in V, a path in G is maximal if and only if it is either infinite or reaches end.

We prove that for $u, w \in V$, u strongly forward dominates w if and only if every maximal path from w passes through u.

With this equivalence part 2 of Definition 82 becomes equivalent to the condition of w_1 in Definition 84. The contra–positive form of this equivalence makes part 3 of Definition 82 equivalent to the condition on w_2 in Definition 84.

Suppose that u strongly forward dominates v. Then by Definition 83, u strongly forward dominates an immediate successor w of v. Then from Definition 84, every maximal path from w passes through u.

Conversely, assume that every maximal path from w passes through u. Clearly u forward dominates w, but in addition every path $w\rho$ of length $|V|+1$ must pass through u, since $w\rho$ must pass more than once through at least one vertex, and so if $w\rho$ does not pass through u, then an infinite path from w exists which also does not pass through u, giving a contradiction, hence u strongly forward dominates w.

Lemma 86 *Let $G = (V, E)$ be a finite directed graph and let $v \in V$. Let $d_1, \ldots, d_n \in V$ be all the immediate successors of v, and assume that $n \geqslant 2$. Let $e_1, \ldots, e_n \in V$ and assume that for each $i \leqslant n$, all maximal paths from d_i pass through e_i, and do so before passing through any e_j for $j \neq i$. Then either there exist e_k, e_l such that $v \xrightarrow{\text{DOD}} e_k, e_l$ or there exists e_l such that $v \xrightarrow{\text{NTSCD}} e_l$.*

PROOF. If every vertex e_i occurs on every maximal path starting at any of the vertices d_j, then $v \xrightarrow{\text{DOD}} e_i, e_j$ for every $i \neq j$ follows. Otherwise, there exist i, j such that e_i does not occur on every maximal path starting at d_j, whereas necessarily e_i occurs on every maximal path starting at d_i. Hence $v \xrightarrow{\text{NTSCD}} e_i$ follows.

8.4 The relationship between strongly commitment–closed sets and sets closed under both $\xrightarrow{\text{NTSCD}}$ and $\xrightarrow{\text{DOD}}$

Lemma 87 *Let $G = (V, E, \beta)$ be a complete* CFG*. V' is closed under both $\xrightarrow{\text{NTSCD}}$ and $\xrightarrow{\text{DOD}}$ then V' is strongly commitment–closed in G.*

PROOF. For any $x \in V \backslash V'$, we first define S_x to be the set of all vertices in $V \backslash V'$ which are reachable from x via a path which only contains vertices in $V \backslash V'$. Observe that if (x, y) is an edge with $x, y \in V \backslash V'$, then $S_x \supseteq S_y$, with strict inclusion if there is no maximal path starting at x and not passing through V', since in that case $x \in S_x \backslash S_y$.

Suppose that V' is not strongly commitment–closed in G. Then there is a vertex $v \in V \backslash V'$ which is not V'–avoiding and is not strongly committing. Choose v satisfying this condition such that $|S_v|$ is minimal and among such vertices having the minimal value for $|S_v|$, the distance in G from v to the set V' (which is defined, since v is not V'–avoiding) is minimal. Let $d_1, \ldots, d_n \in V$ be all the immediate successors of v, where we assume that d_1 is closer to V' than v is.

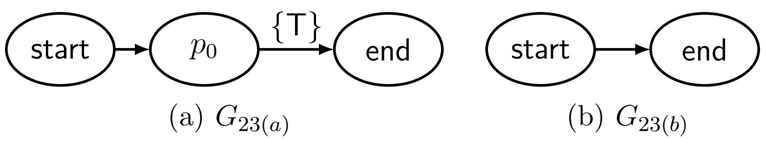


Fig. 23. The smallest set closed under $\xrightarrow{\text{NTSCD}}$ and $\xrightarrow{\text{DOD}}$ in $G_{23(a)}$ containing $\{\mathsf{start}, \mathsf{end}\}$ is $\{\mathsf{start}, \mathsf{end}\}$ but the induced graph, $G_{23(b)}$, is not a strong projection of $G_{23(a)}$ because the maximal walk, $\mathsf{start}, (p_0, \mathsf{F})$, of $G_{23(a)}$ restricts to the walk start of $G_{23(b)}$ which is not maximal in $G_{23(b)}$. Alternatively, we can see that $\{\mathsf{start}, \mathsf{end}\}$ is not strongly commitment closed in $G_{23(a)}$ since p_0 is reachable from $\{\mathsf{start}, \mathsf{end}\}$ in $G_{23(a)}$ but p_0 is neither $\{\mathsf{start}, \mathsf{end}\}$–strongly committing nor $\{\mathsf{start}, \mathsf{end}\}$–avoiding in $G_{23(a)}$.

Since v is not V'–avoiding, there is a V'–path $(v, d_1, \ldots)$ in G with endpoint $w \in V'$, and since all predicate vertices in V are complete and v is not strongly committing, there is a complete and hence maximal path ρ starting at v which does not pass through a vertex in V'. Clearly d_1 is not V'–avoiding, and so by the minimality assumption on v, the condition on d_1 and the fact that $d_1 \notin V' \Rightarrow S_{d_1} \subseteq S_v$ holds, d_1 is strongly committing and hence all maximal paths starting at d_1 pass through w, whereas ρ does not. Thus $v \xrightarrow{\text{NTSCD}} w$ holds, hence V' is not closed under $\xrightarrow{\text{NTSCD}}$.

Thus we may assume that every maximal path starting at v (or hence at any vertex d_i) passes through a vertex in V'. Thus $d_i \notin V' \Rightarrow S_{d_i} \subseteq S_v \backslash \{v\}$ holds for each i, and so each d_i is strongly committing and thus there are vertices $e_i \in V'$ such that for all $i \leqslant n$, either $d_i = e_i$ or every V'–path starting at d_i ends at e_i, and at least one such V'–path exists. Since v is not strongly committing, it is not weakly–committing, and so $n \geqslant 2$. Thus by Lemma 86, V' is not closed under both $\xrightarrow{\text{NTSCD}}$ and $\xrightarrow{\text{DOD}}$.

Figure 23 gives an example where the graph has incomplete predicates. V' is closed under both $\xrightarrow{\text{NTSCD}}$ and $\xrightarrow{\text{DOD}}$ but V' is not strongly commitment–closed in G and hence the induced graph is not a strong projection. This shows that we cannot drop the ‘complete predicates’ condition of Theorem 87.

Lemma 88 *Let $G = (V, E)$ be* start*–graph and* $\mathsf{start} \in V'$*. If V' is strongly commitment–closed in G then V' is closed under both $\xrightarrow{\text{NTSCD}}$ and $\xrightarrow{\text{DOD}}$.*

PROOF. We consider two cases.

- First assume that V' is not closed under $\xrightarrow{\text{DOD}}$. Then $v \xrightarrow{\text{DOD}} b_1, b_2$ for some $b_1, b_2 \in V'$ and $v \in V \backslash V'$. Thus v has immediate successors x_1, x_2 such that for each i, all maximal paths from x_i contain b_i before b_{3-i}. By definition of $\xrightarrow{\text{DOD}}$, there are paths $x_i \nu_i$ on which b_{3-i} does not occur, and on which

b_i occurs at the end, but not before; and by conditions (2) and (3) of the $\xrightarrow{\text{DOD}}$ definition, no vertex occurs on both paths. Hence each path $x_i\nu_i$ has a prefix $\tau_i a_i$ which is a V'–path for distinct $a_i \in V'$, and so $v \in \mathsf{WD}_G(V')$. Now, v is reachable from $\mathsf{start} \in V'$, so by Lemma 47 V' is not weakly commitment–closed, and hence not strongly commitment–closed.

- Suppose instead that V' is not closed under $\xrightarrow{\text{NTSCD}}$. Thus $v \xrightarrow{\text{NTSCD}} w$ holds for some $w \in V'$ and $v \in V \backslash V'$. Hence v has immediate successors x_1, x_2 such that all maximal paths from x_1 contain w, but there is a maximal and hence complete path $x_2\mu$ not passing through w. Thus v is not strongly committing. Since v is clearly not V'–avoiding, and is reachable from start, V' is not strongly commitment–closed.

8.5 The relationship between strongly commitment–closed sets and sets closed under Podgurski–Clarke weak control dependence

Let $G = (V, E, \beta)$ be a $\{\mathsf{start}, \mathsf{end}\}$–CFG and $V' \subseteq V$. In this section we prove that V' is strongly commitment–closed in G if and only if V' is closed under Podgurski–Clarke weak control dependence.

Theorems 91 and 90 show that closure under $\xrightarrow{\text{PC-weak}}$ is equivalent to strong commitment–closedness for vertex sets containing start, provided that end is reachable from all vertices.

Lemma 89 *Let $G = (V, E)$ be an $\{\mathsf{end}\}$–CFG, then for all $v, b, c \in V$, $v \xrightarrow{\text{DOD}} b, c$ never holds.*

PROOF. Suppose that $v \xrightarrow{\text{DOD}} b, c$ holds. After interchanging b and c if necessary, there is a path $b\rho\mathsf{end}$ which does not pass through c. From condition (2) of Definition 81 there is a path $v\sigma b$ which also does not pass through c. However this contradicts condition (1) of Definition 81.

Lemma 90 *Let $G = (V, E)$ be a complete $\{\mathsf{end}\}$–CFG and $V' \subseteq V$. If V' is closed under $\xrightarrow{\text{PC-weak}}$ then V' is strongly commitment–closed in G.*

PROOF. This follows immediately from Lemma 89 and Theorems 85 and 87.

Lemma 91 *Let $G = (V, E)$ be a $\{\mathsf{start}, \mathsf{end}\}$–graph and let $\mathsf{start} \in V' \subseteq V$. If V' is strongly commitment–closed in G then V' is closed under $\xrightarrow{\text{PC-weak}}$.*

PROOF. Suppose V' is not closed under $\xrightarrow{\text{PC-weak}}$. Thus for some $u \in V'$ and $v \in V \backslash V'$, $v \xrightarrow{\text{PC-weak}} u$ holds. Thus by Theorem 85 and the fact that end is reachable from u, for vertices w_i there is a path $vw_1\rho_1 u$ and a maximal and hence complete path $vw_2\rho_2$ which does not pass through u, and every maximal path starting at a vertex on $w_1\rho_1 u$ passes through u. Clearly v is not V'–avoiding. Let x_1 be first element of V' to occur on $vw1\rho1$. If $vw_2\rho_2$ does not pass through V', then v is not V'–strongly committing. On the other hand, suppose that x_2 is the first element of V' to occur on $vw_2\rho_2$, then $x_1 \neq x_2$, since otherwise there would be a maximal path from x_1 not passing through u. Again, we have shown that v is not V'–strongly committing. Thus, since $\mathsf{start} \in V'$, V' is not strongly commitment–closed, proving the theorem.

We have shown that both strong forms of control dependence in the literature are essentially the same: vertex sets closed under them all induce strong projections. We may call any relation on vertex sets that has the property *strong control dependence.* So $\xrightarrow{\text{PC-weak}}$ and the combination of $\xrightarrow{\text{NTSCD}}$ and $\xrightarrow{\text{DOD}}$ are both examples of strong control dependence.

9 Conclusions and Future Work

Authors have previously expressed control dependence as a relation between the vertices of a CFG. In an attempt to capture the *intention* of control dependence, we, on the other hand, define relations between CFGs and show that all previous forms of control dependence induce graphs which, indeed, satisfy these relations. Weak and strong projection can, thus, be thought of as a specification or a semantics of control dependence rather than an implementation. Furthermore, by introducing weak and strong commitment-closedness, we have generalised control dependence and algorithms which compute sets closed under it.

We believe these very natural relations can be considered as correctness criteria for future definitions of control dependence on more general structures and that authors of such new definitions will have a proof obligation based on them. The work we present here also has practical implications: we have defined reasonably efficient algorithms which can be used for slicing more general structures than those considered previously. Weak and strong commitment-closedness also generalises further. Future research will include the following work:

(1) We will investigate the applicability of the theory to more general structures. Clearly, since the concept of a walk generalises to arbitrary finite

labelled graphs, so do weak and strong projections. This may be useful, for example, in defining control dependence in graphs representing non–deterministic programs where non-predicate vertices may have out–degree greater than one and predicates may have non-disjoint edge labels.

(2) We will investigate the theoretical validity and practicability of combining the algorithms for the minimal weakly and strongly commitment–closed supersets of V' in G and described in this paper with data dependence to form weak and strong semantic slices of arbitrarily unstructured programs.

(3) Improvements to the algorithms for computing the minimal weakly and strongly commitment–closed supersets of V' in G will be investigated. It is believed that an $O(V^3)$ worst–case time complexity algorithm may exist.

(4) We will investigate the application our generalised notions of control dependence to other structures for example extended finite state machines [4].

References

[1] A. Abadi, R. Ettinger, Y. A. Feldman, Improving slice accuracy by compression of data and control flow paths, in: Proceedings of the the 7th joint meeting of the European software engineering conference and the ACM SIGSOFT symposium on The foundations of software engineering, ESEC/FSE '09, ACM, New York, NY, USA, 2009, pp. 223–232.
URL `http://doi.acm.org/10.1145/1595696.1595729`

[2] J. R. Allen, K. Kennedy, C. Porterfield, J. Warren, Conversion of control dependence to data dependence, in: POPL '83: Proceedings of the 10th ACM SIGACT-SIGPLAN symposium on Principles of programming languages, ACM, New York, NY, USA, 1983, pp. 177–189.

[3] T. Amtoft, Slicing for modern program structures: a theory for eliminating irrelevant loops, Information Processing Letters 106 (2008) 45–51, an abridged version of Technical report 2007-3 (with title: 'Correctness of Practical Slicing for Modern Program Structures'), Department of Computing and Information Sciences, Kansas State University, May 2007.

[4] K. Androutsopoulos, D. Clark, M. Harman, Z. Li, L. Tratt, Control dependence for extended finite state machines, in: Fundamental Approaches to Software Engineering (FASE '09), vol. 5503, Springer LNCS, York, UK, 2009, pp. 216–230.

[5] R. Barraclough, D. Binkley, S. Danicic, M. Harman, R. Hierons, Ákos Kiss, M. Laurence, A trajectory-based strict semantics for program slicing, Theoretical Computer ScienceAvailable online doi:10.1016/j.tcs.2009.10.025.

[6] G. Bilardi, K. Pingali, A framework for generalized control dependence, in: PLDI '96: Proceedings of the ACM SIGPLAN 1996 conference on Programming language design and implementation, ACM, New York, NY, USA, 1996, pp. 291–300.

[7] G. Bilardi, K. Pingali, Algorithms for computing the static single assignment form, J. ACM 50 (3) (2003) 375–425.

[8] D. Binkley, S. Danicic, T. Gyimóthy, M. Harman, Ákos Kiss, B. Korel, Theoretical foundations of dynamic program slicing, Theoretical Computer Science 360 (1) (2006) 23–41.

[9] D. Binkley, S. Danicic, T. Gyimóthy, M. Harman, A. Kiss, L. Ouarbya, Formalizing executable dynamic and forward slicing, in: 4th International Workshop on Source Code Analysis and Manipulation (SCAM 04), IEEE Computer Society Press, Los Alamitos, California, USA, 2004, pp. 43–52.

[10] D. W. Binkley, S. Danicic, M. Harman, J. Howroyd, L. Ouarbya, A formal relationship between program slicing and partial evaluation, Formal Aspects of Computing 18 (2) (2006) 103–119.

[11] D. W. Binkley, M. Harman, A survey of empirical results on program slicing, Advances in Computers 62 (2004) 105–178.

[12] S. Danicic, D. Binkley, T. Gyimóthy, M. Harman, Ákos Kiss, B. Korel, Minimal slicing and the relationships between forms of slicing, in: 5th IEEE International Workshop on Source Code Analysis and Manipulation (SCAM 05), IEEE Computer Society Press, Los Alamitos, California, USA, 2005, pp. 45–54, best paper award winner.

[13] S. Danicic, D. Binkley, T. Gyimóthy, M. Harman, Ákos Kiss, B. Korel, A formalisation of the relationship between forms of program slicing, Science of Computer Programming 62 (3) (2006) 228–252.

[14] S. Danicic, M. Harman, J. Howroyd, L. Ouarbya, A lazy semantics for program slicing, in: 1st. International Workshop on Programming Language Interference and Dependence, Verona, Italy, 2004.
URL `http://profs.sci.univr.it/ mastroen/noninterference.html`

[15] M. Daoudi, S. Danicic, J. Howroyd, M. Harman, C. Fox, L. Ouarbya, M. Ward, ConSUS: A scalable approach to conditioned slicing, in: IEEE Working Conference on Reverse Engineering (WCRE 2002), IEEE Computer Society Press, Los Alamitos, California, USA, 2002, pp. 109–118, invited for special issue of the Journal of Systems and Software as best paper from WCRE 2002.

[16] D. E. Denning, P. J. Denning, Certification of programs for secure information flow, Commun. ACM 20 (7) (1977) 504–513.

[17] J. Ferrante, K. J. Ottenstein, J. D. Warren, The program dependence graph and its use in optimization, ACM Transactions on Programming Languages and Systems 9 (3) (1987) 319–349.

[18] T. Gallai, Maximum-minimum sätze und verallgemeinerte faktoren von graphen, Acta Math. Acad. Sci. Hungar. 12 (1961) 131–173.

[19] M. Harman, D. W. Binkley, S. Danicic, Amorphous program slicing, Journal of Systems and Software 68 (1) (2003) 45–64.

[20] M. Harman, S. Danicic, A new algorithm for slicing unstructured programs, Journal of Software Maintenance and Evolution 10 (6) (1998) 415–441.

[21] M. Harman, R. M. Hierons, An overview of program slicing, Software Focus 2 (3) (2001) 85–92.

[22] S. Horwitz, T. Reps, D. W. Binkley, Interprocedural slicing using dependence graphs, in: ACM SIGPLAN Conference on Programming Language Design and Implementation, Atlanta, Georgia, 1988, pp. 25–46, proceedings in *SIGPLAN Notices*, 23(7), pp.35–46, 1988.

[23] S. Horwitz, T. Reps, D. W. Binkley, Interprocedural slicing using dependence graphs, ACM Transactions on Programming Languages and Systems 12 (1) (1990) 26–61.

[24] R. Milner, A Calculus of Communicating Systems, Springer-Verlag New York, Inc., Secaucus, NJ, USA, 1982.

[25] I. A. Natour, On the control dependence in the program dependence graph, in: CSC '88: Proceedings of the 1988 ACM sixteenth annual conference on Computer science, ACM, New York, NY, USA, 1988, pp. 510–519.

[26] K. J. Ottenstein, L. M. Ottenstein, The program dependence graph in software development environments, Proceedings of the ACM SIGSOFT/SIGPLAN Software Engineering Symposium on Practical Software Development Environmt, SIGPLAN Notices 19 (5) (1984) 177–184.

[27] A. Podgurski, L. Clarke, A formal model of program dependences and its implications for software testing, debugging, and maintenance, IEEE Transactions on Software Engineering 16 (9) (1990) 965–79.

[28] L. Ramshaw, Eliminating goto's while preserving program structure, Journal of the ACM 35 (4) (1988) 893–920.

[29] V. P. Ranganath, T. Amtoft, A. Banerjee, M. B. Dwyer, J. Hatcliff, A new foundation for control-dependence and slicing for modern program structures, in: European Symposium on Programming, 2005, pp. 77–93.

[30] V. P. Ranganath, T. Amtoft, A. Banerjee, J. Hatcliff, M. B. Dwyer, A new foundation for control dependence and slicing for modern program structures, ACM Transactions on Programming Languages and Systems (2007) 29 (5).

[31] S. Sinha, M. J. Harrold, G. Rothermel, Interprocedural control dependence, ACM Trans. Softw. Eng. Methodol. 10 (2) (2001) 209–254.

[32] F. Tip, A survey of program slicing techniques, Journal of Programming Languages 3 (3) (1995) 121–189.

[33] M. Weiser, Program slices: Formal, psychological, and practical investigations of an automatic program abstraction method, Ph.D. thesis, University of Michigan, Ann Arbor, MI (1979).

[34] M. Weiser, Program slicing, in: 5^{th} International Conference on Software Engineering, San Diego, CA, 1981, pp. 439–449.

[35] M. Weiss, The transitive closure of control dependence: the iterated join, ACM Lett. Program. Lang. Syst. 1 (2) (1992) 178–190.