Towards a Relation-Algebraic Account of Control Flow Graph Analysis (Extended Abstract, Work in Progress)

Baltasar Trancón y Widemann

June 16, 2024

1 Introduction

Control flow graphs (CFGs) are an important data structure for the analysis and translation of programs [2, 8]. Their vertices are single instructions or atomic 'basic blocks' of code, often with the addition of some virtual nodes for structure, and their directed edges are potential (conditional or unconditional) jumps.

The discussion of CFGs and derived concepts is often only semiformal, i.e. in words rather than formulae, in applications [5, e.g.]. This terse and high-level style is effective for readers with solid expertise in the topic, but there are drawbacks: Verbal arguments are prone to overlooking technical details (example in section 2 below), and they may fail to convince or convey an adequate intuition in didactical scenarios.

The present paper describes the first steps of ongoing work to formalize CFG theory in the language of relation algebra. Besides the obvious advantages of a fully formal presentation, an axiomatic approach can also shed light on the meta-mathematics of a topic, i.e. on its implicit requirements.

2 Case Study: Dominance

While many algorithms work on the plain *adjacency* relation of CFGs, some (and in particular some sophisticated ones) are based on the derived *dominance* relation on CFG nodes [1]. The following definition is a prime example of the verbal style:

Definition 1. Let G be a directed graph with a distinguished *entry* node e. A node v *dominates* another node w, if and only if all paths in G from e to w pass through v.

A separate *post-dominance* relation is also considered, but it is merely the dominance relation of the converse graph.

To demonstrate that the devil may be in the details, consider the following statement: "Dominance is a partial order." This is intuitively appealing, and sometimes silently presupposed [4, e.g.], but not exactly true in general! In a reasonable formalization, one can prove that dominance is indeed reflexive and transitive, but antisymmetry fails unless an extra assumption is made. Namely, dominance is antisymmetric only if there is no *dead code*, i.e., all nodes are reachable from the entry node; unreachable nodes are vacuously dominated by every other node.

Unfortunately, whether unreachable nodes are allowed or forbidden in CFGs depends on the context; the rule may even change between phases of the same optimizing compiler, as evidenced by the existence of *dead code elimination* procedures [1].

More advanced constructs, such as the algorithm of Lengauer and Tarjan [6], require much more substantial investigation to unravel their formal inner workings [3, 7].

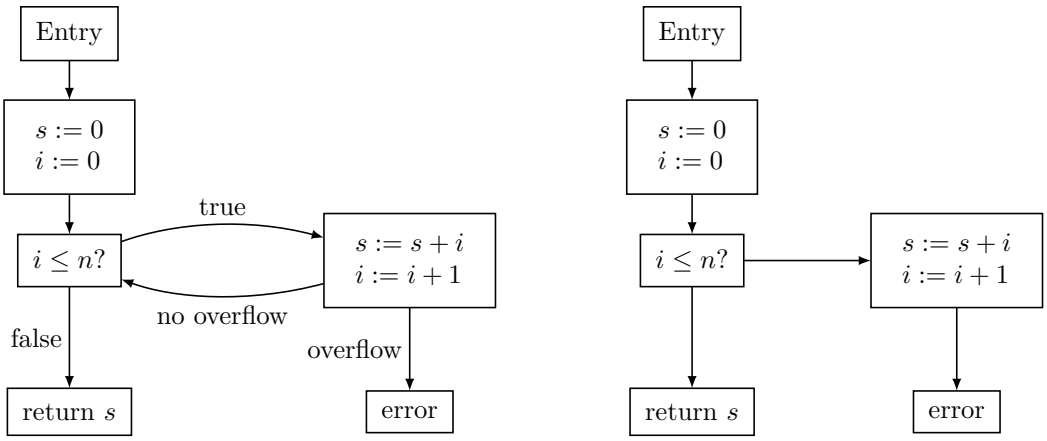

Figure 1: A Control Flow Graph (*Left*) and Its Dominance Spanning Tree (*Right*)

3 Which Theoretical Framework?

As a corollary of Occam's Razor, the axiomatic formalization of a topic should use the weakest system of axioms that does the job. Tarski's abstract relation algebra **RA** is the evident candidate, and we write $(\mathcal{L}, \sqcup, \sqcap, \bar{}, 0, 1, \cdot, I, \breve{})$ for the relational language. But since graph theory builds on actual relations on the set V of nodes, results may possibly be (only) theorems of the stronger representable relation algebra **RRA**. In the present work, our strategy is to make do with **RA** as far as possible, and state requirements that may go beyond explicitly. Firstly and most importantly, we require fixed points:

Requirement 2. *The relation algebra is complete as a Boolean lattice.*

Note that this is not a theorem of either **RA** or **RRA** at face value. In **RRA**, however, any relation algebra is, up to isomorphism, of the form $\mathcal{L} \subseteq \mathcal{P}(X \times X)$, and can hence be completed. Besides, since CFGs are generally finite, we may consider only finite relation algebras, which are necessarily complete.

Completeness gives us least and greatest fixed points of operators, quite fittingly, by Knaster's and Tarski's theorem.

Definition 3 (Least/Greatest Fixed Point)**.** Let $\Phi : \mathcal{L} \to \mathcal{L}$ be a unary operator acting on the relations. Let $\sqsubseteq$ denote the usual lattice order on relations. Then we call relations $\mu\,\Phi$ and $\nu\,\Phi$ *least* and *greatest fixed points* of Φ, respectively, if and only if:

$$\mu\,\Phi = \Phi(\mu\,\Phi) \qquad\qquad R = \Phi(R) \implies \mu\,\Phi \sqsubseteq R$$
$$\nu\,\Phi = \Phi(\nu\,\Phi) \qquad\qquad R = \Phi(R) \implies R \sqsubseteq \nu\,\Phi$$

Theorem 4 (Knaster–Tarski)**.** *Both $\mu\,\Phi$ and $\nu\,\Phi$ exist uniquely for all monotone operators Φ. Furthermore, the above conditions can be generalized to prefixed/postfixed points, respectively:*

$$\Phi(R) \sqsubseteq R \implies \mu\,\Phi \sqsubseteq R$$
$$R \sqsubseteq \Phi(R) \implies R \sqsubseteq \nu\,\Phi$$

Definition 5 ((Reflexive-)Transitive Closure)**.** The transitive closure R^+ of a relation R is the least fixed point of (left or right) composition with R. The reflexive-transitive closure R^* is derived in the usual way.

$$R^+ = \mu(R \cdot \_) = \mu(\_ \cdot R) \qquad\qquad R^* = R^+ \sqcup I$$

4 Control Flow Graphs

Definition 6 (Control Flow Graph)**.** A *control flow graph* (CFG) is a directed graph $G = (V, E, v_0)$, with simple edges that are fully specified by the adjacency relation $E \subseteq V \times V$, and a distinguished entry node $v_0 \in V$.

Statements about a CFG $G = (V, E, v_0)$ shall be formulated in terms of relations in $\mathcal{L} = \mathcal{P}(V \times V)$. Thus we use E as a constant. We also write I_W for the partial identity relation on $W \subseteq V$. In particular, we abbreviate $I_{\{v_0\}}$ to I_0. For example, we can state the no-dead-code property discussed above formally as follows:

Definition 7 (Entry-Connectedness)**.** A CFG $G = (V, E, v_0)$ is called *entry-connected* if and only if all nodes are reachable from the entry node:

$$I_0 \cdot E^* = I_0 \cdot 1$$

5 Dominance Formalized Classically

A classical formalization [2] specifies the dominance relation in a language of set-valued functions as the greatest solution of a system of equations.

Definition 8 (Successor/Predecessor)**.** The *successor* and *predecessors* functions $\mathrm{succ}, \mathrm{pred} : V \to \mathcal{P}(V)$ are the forward and reverse images of the adjacency relation, respectively.

$$\mathrm{succ}(v) = \{w \in V \mid (v, w) \in E\} \qquad \mathrm{pred}(v) = \{u \in V \mid (u, v) \in E\}$$

Definition 9 (Dominance Function)**.** The *dominance function* is the (pointwise) greatest function $\mathrm{Dom} : V \to \mathcal{P}(V)$ that satisfies the following equation:

$$\mathrm{Dom}(v) = \begin{cases} \{v\} & (v = v_0) \\ \{v\} \cup \bigcap_{p \in \mathrm{pred}(v)} \mathrm{Dom}(p) & (v \neq v_0) \end{cases}$$

6 Dominance Formalized Relationally

Definition 10 (Dominance Operator)**.** Let $G = (V, E, v_0)$ be a CFG. Write $V' = V \setminus \{v_0\}$ and $I' = I_{V'}$. Define an operator $\Delta : \mathcal{L} \to \mathcal{L}$ as follows:

$$\Delta(R) = I \sqcup \overline{\overline{R} \cdot \overline{E}} \cdot I'$$

Lemma 11. *Δ is monotone.*

Proof. Δ can be rewritten as a composition of five primitive operators, of which three are monotone and two are antitone:

$$\Delta = \left(I \sqcup \_\right) \circ \left(\_ \cdot I'\right) \circ {}^{-} \circ \left(\_ \cdot \overline{E}\right) \circ {}^{-}$$

□

Definition 12 (Dominance Relation)**.** The *dominance relation* is the greatest fixpoint of the operator Δ:

$$D = \nu\, \Delta$$

Theorem 13. *Definitions 9 and 12 are equivalent in the traditional interpretation:*

$$v \in \mathrm{Dom}(w) \iff (v, w) \in D$$

Proof. Let $D' = \{(v, w) \mid v \in \mathrm{Dom}(w)\}$ and show $D = D'$. Since D is defined as a unique greatest solution, it suffices if D' satisfies the same defining equation:

$$\begin{aligned}
(v,w) \in D' &\iff v \in \mathrm{Dom}(w) \\
&\iff (w = v_0 \wedge v = w) \vee \Big(w \neq v_0 \wedge \big(v = w \vee \forall p \in \mathrm{pred}(w).\, v \in \mathrm{Dom}(p)\big)\Big) \\
&\iff v = w \vee \big(w \neq v_0 \wedge \forall p \in \mathrm{pred}(w).\, v \in \mathrm{Dom}(p)\big) \\
&\iff v = w \vee (w \neq v_0 \wedge \forall p \in V.\, (p,w) \in E \wedge (v,p) \in D') \\
&\iff v = w \vee (w \neq v_0 \wedge \neg\exists p \in V.\, (p,w) \notin E \wedge (v,p) \notin D') \\
&\iff v = w \vee \big(w \neq v_0 \wedge \neg\exists p \in V.\, (p,w) \in \overline{E} \wedge (v,p) \in \overline{D'}\big) \\
&\iff v = w \vee \big(w \neq v_0 \wedge (v,w) \notin \overline{D'} \cdot \overline{E}\big) \\
&\iff v = w \vee \Big(w \neq v_0 \wedge (v,w) \in \overline{\overline{D'} \cdot \overline{E}}\Big) \\
&\iff v = w \vee \Big(\exists x \in V'.\, w = x \wedge (v,x) \in \overline{\overline{D'} \cdot \overline{E}}\Big) \\
&\iff (v,w) \in I \vee \Big(\exists x \in V.\, (w,x) \in I' \wedge (v,x) \in \overline{\overline{D'} \cdot \overline{E}}\Big) \\
&\iff (v,w) \in I \vee (v,w) \in \overline{\overline{D'} \cdot \overline{E}} \cdot I' \\
&\iff (v,w) \in I \sqcup \overline{\overline{D'} \cdot \overline{E}} \cdot I' \\
&\iff (v,w) \in \Delta(D')
\end{aligned}$$

□

7 Proving Properties

We shall evaluate the relational definition by demonstrating how to prove properties.

7.1 Reflexivity

Reflexivity is a very simple property; thus there is little actual need for a novel proof. However, it serves as a toy example to demonstrate succinctly the proof style that results from the relational approach to the topic.

Theorem 14. *D is reflexive:*

$$I \sqsubseteq D$$

Proof. Show that I is a postfixed point of Δ:

$$\Delta(I) = I \sqcup \cdots \sqsupseteq I$$

Hence the result follows immediately from Theorem 4. □

7.2 Transitivity

The reader is invited to give a formal proof of the transitivity of dominance, either by formalizing the notions of *path* and *passing through* in Definition 1, or by starting from Definition 9. Here, we shall give a novel equational proof in the relational style.

However, that proof requires a technical property that is not evidently a theorem of **RA**.

Requirement 15.

$$R \cdot \overline{\overline{S} \cdot T} \sqsubseteq \overline{\overline{R \cdot S} \cdot T}$$

Lemma 16. *Requirement 15 holds in* **RRA**.

Proof. By translation to set theory:

$$
\begin{aligned}
& (w,z) \in R \cdot \overline{\overline{S} \cdot T} \\
\iff & \exists x.\, (w,x) \in R \wedge (x,z) \in \overline{\overline{S} \cdot T} \\
\iff & \exists x.\, (w,x) \in R \wedge (x,z) \notin \overline{S} \cdot T \\
\iff & \exists x.\, (w,x) \in R \wedge \neg\big(\exists y.\, (x,y) \in \overline{S} \wedge (y,z) \in T\big) \\
\iff & \exists x.\, (w,x) \in R \wedge \neg\big(\exists y.\, (x,y) \notin S \wedge (y,z) \in T\big) \\
\iff & \exists x.\, (w,x) \in R \wedge \big(\forall y.\, (x,y) \in S \vee (y,z) \notin T\big) \\
\iff & \exists x. \forall y.\, (w,x) \in R \wedge \big((x,y) \in S \vee (y,z) \notin T\big) \\
\implies & \forall y. \exists x.\, (w,x) \in R \wedge \big((x,y) \in S \vee (y,z) \notin T\big) \\
\iff & \forall y. \exists x.\, \big((w,x) \in R \wedge (x,y) \in S\big) \vee \big((w,x) \in R \wedge (y,z) \notin T\big) \\
\implies & \forall y. \exists x.\, \big((w,x) \in R \wedge (x,y) \in S\big) \vee (y,z) \notin T \\
\iff & \forall y.\, \big(\exists x.\, (w,x) \in R \wedge (x,y) \in S\big) \vee (y,z) \notin T \\
\iff & \forall y.\, (w,y) \in R \cdot S \vee (y,z) \notin T \\
\iff & \neg\big(\exists y.\, (w,y) \notin R \cdot S \wedge (y,z) \in T\big) \\
\iff & \neg\big(\exists y.\, (w,y) \in \overline{R \cdot S} \wedge (y,z) \in T\big) \\
\iff & (w,z) \notin \overline{R \cdot S} \cdot T \\
\iff & (w,z) \in \overline{\overline{R \cdot S} \cdot T}
\end{aligned}
$$

□

Theorem 17. *D is transitive:*

$$D \cdot D \sqsubseteq D$$

Proof. Show that $D \cdot D$ is a postfixed point of Δ:

$$
\begin{aligned}
D \cdot D &= \Big(I \sqcup \overline{\overline{D} \cdot \overline{E}} \cdot I'\Big) \cdot \Big(I \sqcup \overline{\overline{D} \cdot \overline{E}} \cdot I'\Big) && \text{(Definition 12)} \\
&= I \cdot I \sqcup I \cdot \overline{\overline{D} \cdot \overline{E}} \cdot I' \sqcup \overline{\overline{D} \cdot \overline{E}} \cdot I' \cdot I \sqcup \overline{\overline{D} \cdot \overline{E}} \cdot I' \cdot \overline{\overline{D} \cdot \overline{E}} \cdot I' && \text{(distributive)} \\
&= I \cdot I \sqcup I \cdot \overline{\overline{D} \cdot \overline{E}} \cdot I' \sqcup I \cdot \overline{\overline{D} \cdot \overline{E}} \cdot I' \sqcup \overline{\overline{D} \cdot \overline{E}} \cdot I' \cdot \overline{\overline{D} \cdot \overline{E}} \cdot I' && (I \text{ shuffling}) \\
&= I \sqcup I \cdot \overline{\overline{D} \cdot \overline{E}} \cdot I' \sqcup \overline{\overline{D} \cdot \overline{E}} \cdot I' \cdot \overline{\overline{D} \cdot \overline{E}} \cdot I' && \text{(idempotents)} \\
&= I \sqcup \Big(I \sqcup \overline{\overline{D} \cdot \overline{E}} \cdot I'\Big) \cdot \overline{\overline{D} \cdot \overline{E}} \cdot I' && \text{(distributive)} \\
&= I \sqcup D \cdot \overline{\overline{D} \cdot \overline{E}} \cdot I' && \text{(Definition 12)} \\
&\sqsubseteq I \sqcup \overline{\overline{D} \cdot D \cdot \overline{E}} \cdot I' && \text{(Requirement 15)} \\
&= \Delta(D \cdot D) && \text{(Definition 10)}
\end{aligned}
$$

Hence the result follows immediately from Theorem 4. □

8 Conclusion

We have explored some first steps towards a relation-algebraic account of control flow graph analysis. In particular, we have given a concise relational definition of the dominance relation and an equational proof of its transitivity. Our approach relies on some requirements that are not (evidently) theorems of Tarski's relation algebra, most notably the unique existence of fixed points; their axiomatic status is currently uncertain. In the future, we intend to complete proving the relevant properties of the dominance relation in style, and to apply the approach to the formalization of algorithms for computing dominance relations, in particular that of Cooper et al. [4] and of Lengauer and Tarjan [6].

References

[1] A. V. Aho and J. D. Ullman. *Principles of Compiler Design*. The 'dragon book'. Pearson, 1977. ISBN: 0201000229.

[2] F. E. Allen. "Control flow analysis". In: *SIGPLAN Not.* 5.7 (1970), 1–19. ISSN: 0362-1340. DOI: 10.1145/390013.808479.

[3] S. Baziotis. *The Cryptic Proof of Semidominators in the Lengauer-Tarjan Algorithm*. 2022. URL: https://sbaziotis.com/compilers/semidominators-proof.html (visited on 06/16/2024).

[4] K. Cooper, T. Harvey, and K. Kennedy. *A Simple, Fast Dominance Algorithm*. CS Technical Report 06-33870. Rice University, 2006. URL: http://www.hipersoft.rice.edu/grads/publications/dom14.pdf.

[5] R. Cytron et al. "An efficient method of computing static single assignment form". In: *Proceedings of the 16th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*. ACM, 1989, 25–35. DOI: 10.1145/75277.75280.

[6] T. Lengauer and R. E. Tarjan. "A fast algorithm for finding dominators in a flowgraph". In: *ACM Transactions on Programming Languages and Systems (TOPLAS)* 1.1 (1979), pp. 121–141. DOI: 10.1145/357062.357071.

[7] J. Misra. *An Explanation of Lengauer–Tarjan Dominators Algorithm*. 2023. URL: https://www.cs.utexas.edu/users/misra/Lengauer+Tarjan.pdf (visited on 06/16/2024).

[8] R. T. Prosser. "Applications of Boolean matrices to the analysis of flow diagrams". In: *Proceedings of the Eastern Joint IRE-AIEE-ACM Computer Conference*. Boston, Massachusetts: Association for Computing Machinery, 1959, 133–138. DOI: 10.1145/1460299.1460314.