YaleNUSCollege

May 18, 2023

Emanuel Kieroński
Institute of Computer Science
University of Wrocław
Poland

Dear Prof. Kieroński:

Tomasz Drab's research is about reduction strategies in the lambda calculus and his thesis is that their implementation as abstract machines can be derived effectively. He considers a taxonomy (a "zoo") of reduction strategies – weak, full, without memoization, and with memoization – and he substantiates his thesis with a number of abstract machines, not by inventing them but by meticulously deriving them in a way that enables one to reason about their computational complexity, which is an achievement. His work is formalized with the Coq Proof Assistant, a guarantee of mathematical integrity.

Drab's dissertation is structured as follows, and true to Computer Science since Turing Award Edsger W. Dijkstra, it starts with a zeroth chapter:

- Chapter 0 provides a very broad overview of the domain as well as a lucid description of Drab's scholarly approach and academic journey, including an English-to-Polish glossary.
  Drab starts with an epistemological analysis of language and thought in the context of theoretical computer science. Like others before him, he wisely stops at the near teleology of mathematics, moving on instead to analyzing programming languages and computational thinking. His technical introduction starts at the bottom with a simple language of arithmetic expressions and then with the language of lambda terms which he methodically describes, from its syntax to its semantics and the implementations of these semantics. He then revisits the simple language of arithmetic expressions and describes the implementation of its semantics as an abstract machine before continuing with abstract machines for the language of lambda terms that implement various notions of reduction: weak (call by value, call by name, and call by need), full, without memoization, and with memoization. He briefly describes normalization by evaluation and the functional correspondence, and concludes on the computational complexity of abstract machines. All in all, Chapter 0 is not just encyclopedic: it also displays a non-trivial balance between intuition and logic that Henri Poincaré would have appreciated.

- Chapter I presents a taxonomy of reduction strategies for the lambda calculus that accounts for existing ones and that makes it possible to systematically explore new ones. I find it piercingly clever.
  Notably, it introduces a notion of phased strategies and demonstrate that they are a great enabler to explore new reduction strategies methodically.
- Chapter II presents an abstract machine for full applicative-order reduction, i.e., call by value where the bodies of the residual lambda abstractions are also reduced. The abstract machine is not invented – instead, it is systematically derived and it is situated in the taxonomy of existing abstract machines, with a formal correctness proof to boot.
  Notably, we are shown how to test whether two terms are convertible not by fully reducing them and then comparing their normal form, but instead by incrementally normalizing them and stopping as soon as a difference is spotted. This incremental strategy originates in the classical "samefringe" problem where we are asked whether two trees of payloads have the same list of payloads according to a given traversal strategy: since the two given trees in general do not have the same shape, they cannot be synchronously traversed, and therefore one first needs to map each of them to a list of payloads; since the two lists have the same shape, they can then be compared synchronously (and iteratively). Constructing these two lists eagerly, i.e., in a call-by-value way, is a source of inefficiency if they immediately differ: there was no need to construct their tails. An effective as well as more efficient solution is to map the two trees to two lazy lists that are constructed in a call-by-need way, so that their construction (and therefore the traversal of each tree) is stopped as soon as two payloads are observed to differ. As documented in Grégoire and Leroy at ICFP 2002, this incremental strategy is at the heart of the implementation of the "reflexivity" tactic in the Coq Proof Assistant, and a key reason why Coq is so efficient. This beautiful idea is elegantly rendered in this rich chapter.
- Chapter III perspicaciously expands on Chapter II based on Accattoli et al.'s "reasonable" notion of complexity, namely by relating the number of steps taken by the abstract machine with the number of β reductions in the given term.
  Notably, this relation beautifully hinges on an amortized analysis in the manner of Chris Okasaki.
- Chapter IV presents an abstract machine for full normal-order reduction with memoization, i.e., call by need where the bodies of the residual lambda abstractions are also reduced. Again, the abstract machine is not invented – instead, it is systematically derived and it is situated in the taxonomy of existing abstract machines, with a formal correctness proof to boot, a remarkable achievement.
  Notably, the complexity of this abstract machine is assessed.

Chapters I to IV are jointly authored and already peer-reviewed: they are of very high scientific quality, report substantial new results, and are written with great care and rigor. Chapter 0 is the prolegomenon of these chapters: it introduces their domain of discourse from the bottom up and unequivocally establishes Drab's conceptual, theoretical, practical, and pragmatic expertise. This encyclopedic chapter is aligned with the very high scientific quality and with the careful and rigorous narrative of the subsequent chapters, with a personal touch that many years from now, Drab will recognize as portraying the computer scientist as a young man.

Tomasz Drab's PhD dissertation is a commendable – and, to me, outstanding – contribution that speaks excellently of his education, his supervisors, and his academic institution.

Sincerely yours,

Olivier Danvy
Professor of Science (Computer Science)