A metalanguage for cost-aware denotational semantics

Yue Niu
Computer Science Department
Carnegie Mellon University
Pittsburgh, USA
yuen@cs.cmu.edu

Robert Harper
Computer Science Department
Carnegie Mellon University
Pittsburgh, USA
rwh@cs.cmu.edu

omputer Science (LICS) | 979-8-3503-3587-3/23/$31.00 ©2023 IEEE | DOI: 10.1109/LICS56636.2023.10175777

***Abstract*—We present metalanguages for developing synthetic cost-aware denotational semantics of programming languages. Extending recent advances by Niu et al. in cost and behavioral verification in dependent type theory, we define two successively more expressive metalanguages for studying cost-aware metatheory. We construct synthetic denotational models of the simply-typed lambda calculus and Modernized Algol, a language with first-order store and while loops, and show that they satisfy a cost-aware generalization of the classic Plotkin-type computational adequacy theorem. Moreover, by developing our proofs in a synthetic language of phase-separated constructions of intension and extension, our results easily restrict to the corresponding extensional theorems. Consequently, our work provides a positive answer to the conjecture raised in op. cit. and contributes a framework for cost-aware programming, verification, and metatheory.**

***Index Terms*—types, semantics, cost analysis**

I. Introduction

Denotational semantics is a well-established method for obtaining an *equational theory* for program verification. Whereas the operational semantics of a programming language gives meaning to programs via closed, whole program computation, denotational semantics aims to assign a *compositional*

general context of equational theories. In particular, Niu et al. proposed a dependent type theory **calf** (**c**ost-**a**ware **l**ogical **f**ramework) that provides a rich specification language supporting both behavioral and cost verification of functional programs. That work formalizes a myriad of case studies of the cost analysis of algorithms in the framework and proves the consistency of **calf** via a model construction. As a type theory, **calf** can be thought of as the semantic domain of a denotational semantics in the sense that it furnishes an equational theory for program analysis. Moreover, as a cost analysis framework, **calf** does not stipulate a cost semantics for programs; instead, the users of the framework is responsible for specifying the cost model of the algorithms they define. This raises a natural question: how does one know if a cost model is reasonable relative to a given programming language? In the concluding remarks, the authors conjecture that the choice of a cost model with respect to an operational semantics may be justified by an *internal* adequacy theorem in the style of Plotkin.

In this paper, we substantiate this idea and develop extensions of **calf** that promote it to a metalanguage for *synthetic cost-aware denotational semantics*. To illustrate our approach, we first define **calf***, an extension of **calf** with universes and inductive types, which we use to define a computationally