Cost Semantics and Verification

Robert Harper
Carnegie Mellon University

Guy E. Blelloch 60th Celebration
October 15, 2022

Coming to CMU

Guy and I have been colleagues since November, 1988.

We had, and have, a lot in common, both personally ...

- Life in the UK.
- Cycling.
- Politics.

... and professionally,

- Emphasis on the interplay between theory and practice in programming.
- Devotion to teaching and curriculum at all levels.

His influence on my thinking is enormous, and ever-growing. I will talk briefly about one example.

A Multi-Faceted Collaboration

Over the years we have

- Co-laborated on projects: eg, PSciCo.
- Co-advised students: Cheng, Acar, Spoonhower.
- Co-authored papers: eg, Cache-Efficient Functional Algorithms, ...
- Co-taught a course: Parallel Data Structures and Algorithms.

And I am a relatively minor player in Guy's extensive record of collaboration with people here and elsewhere!

Cost Semantics

Guy's work on cost semantics is particularly influential on me, and more broadly.

- Functional model of parallelism (starting with his scan-vector model).
- Parallelism is defined in terms of work and span aka cost graphs.
- Separation of abstraction from its realization: Brent-type theorems.

This all resonated with my interests and with methods in PL theory.

- Functional programs are far simpler and clearer.
- Types are very effective, because of Reynolds's theory of parametricity.
- Verification is much simpler, essentially equational.
- Adequacy theorems a la Plotkin relate mathematical to operational meaning.

Types for Behavior

Dependent types allow precise specifications of program behavior:

$$s : seq \to (s' : seq \times perm(s,s') \times sorted(s'))$$

The type of functions on seq's that return a sorted permutation of their inputs.

Equations can be used to verify (relative) correctness:

$$isort \doteq msort \in s : seq \to (s' : seq \times perm(s,s') \times sorted(s'))$$

Insertion sort and merge sort are extensionally equal.

$$isort(s) \doteq msort(s) \in (s' : seq \times perm(s,s') \times sorted(s'))$$

Types for Cost

[JWW Niu, Sterling, Grodin]

Can dependent type theory be extended to account for cost?

- Equations offer no intrinsic notion of cost (eg, steps).
- Account for sequential and parallel cost, other measures.

Costed specifications?

- $\mathit{isort} \in s : \mathit{seq} \xrightarrow{|s|^2} (s' : \mathit{seq} \times \mathit{perm}(s,s') \times \mathit{sorted}(s'))$
- $\mathit{msort} \in s : \mathit{seq} \xrightarrow{|s| \lg |s|} (s' : \mathit{seq} \times \mathit{perm}(s,s') \times \mathit{sorted}(s'))$

What does this mean? How can equal functions have different properties?

Types for Cost

Fundamental type theory to the rescue: Synthetic Tait Computability.

- Developed by Sterling in his remarkable 2021 Ph.D.
- Synthetic formulation: "everything is a computability relation."
- Phases separate syntax from semantics.

Abstract notion of cost:

- Instrument code: $step^{(n)}(e)$ (eg, count each comparison)
- Define $f \in A \xrightarrow{n} B$ to mean

$$x : A \to (y : B \times f(x) \doteq step^{(n)}(y)).$$

- (Must be validated separately.)

Equational reasoning is of the essence, including treating functional programs as functions.

Types for Cost

Use phases to distinguish extensional from intensional aspects:

- In the extensional phase $\mathit{step}^{(n)}(e) \doteq e$.
- Consequently,

$$\mathit{ext}\,\mathsf{true} \vdash \mathit{isort} \doteq \mathit{msort} \in \mathit{seq} \to \mathit{seq}.$$

- Standard type-theoretic methods available extensionally.

Absent restriction to extensional phase, cost matters:

- $\mathit{isort} \in s : \mathit{seq} \xrightarrow{|s|^2} \mathit{seq}$.
- $\mathit{msort} \in s : \mathit{seq} \xrightarrow{|s|\,\lg|s|} \mathit{seq}$.
- . . . and they are of course not equal!
- . . . despite function extensionality remaining valid.

What I'm Suppressing

To admit static analysis of cost, must enforce by-value evaluation.

- ∂CBPV type theory fundamental [Pedrot/Tabareaux].
- Distinguish values from computations.

To admit efficient algorithms, require arbitrary decompositions (eg, split a sequence).

- Must remain within total framework (no loops).
- Distinguish definitional formulation from cost decomposition.

Ongoing and Future Work

Empirical validation: mechanize Guy's undergraduate parallel algorithms course.

- Deterministic algorithms pose few additional challenges.
- Extend to probabilistic case? (Seems workable.)
- Tool of choice: Agda prover. (But others are conceivable.)

Validate cost accounting relative to execution model.

- Step count should reflect reality of execution!
- Generalize Plotkin's adequacy to account for cost and behavior.
- Notable similarity to Brent-type theorems in Guy's work.

See forthcoming Ph.D. of Yue Niu!

Happy Birthday, Guy!

Thanks for all of your influence and inspiration.