Thesis Synopsis

Datalogisk institut, Copenhagen University (DIKU)
Master's Thesis

Oleksandr Shturmov
oleks@oleks.info

June 5, 2013.

1 Working Title

Implicit guarantees of the computational complexity of programs.

2 Background

"Implicit" complexity theory is concerned with the design of programming languages, such that programs are guaranteed by construction to be bounded in their computational complexity. This is in contrast to "explicit" complexity theory, where we attempt to bound the computational complexity of already written programs.

Research in computational complexity has, so far, been predominantly focused on the explicit variant. The hope with the implicit variant is twofold [Baillot et al. (2006)]:

(a) provide for an alternative categorization of complexity classes, expressed in terms of logical principles, rather than e.g. the costs induced for a particular machine model; and

(b) provide for a tractable approach to static verification of program complexity.

In theory, we are concerned with the influence of various logical principles (e.g. elements of semantics and types) on the computational complexity of programs. In practice, we are concerned with (de)restricting the programmer, implying complexity guarantees.

3 Thesis Proposal

We consider designing a family of languages for complexity-bounded computation. We are concerned with the "feasible" complexity classes, i.e. P, and those contained in P.

We seek to incorporate this family into a single programming language, where the programmer can state, and be statically guaranteed, a computational complexity bound on a program section, e.g. the entire program.

We first consider a programming language where all computation is bounded to P, e.g. [Bellantoni & Cook (1992), Jones (1999)]. We then consider subsets of this language, bounded to subsets of P. We extend the original language with new constructs, if useful to our subsets, and syntactic sugar, if useful for programming.

The thesis proposal is thereby twofold:

(a) implicitly categorise a range of complexity classes under P; and

(b) devise a programming language, supporting bounded and effectful program sections.

The bounds are guaranteed trivially if we only allow for a no-op in a bounded section. So by "effectful", we mean that stating a higher bound enables more complex computation.

4 Learning Objectives

(a) Find and survey previous work in an area of research in computational complexity.

(b) Explain and contrast this work in a matter understandable to those unfamiliar with the research area, but familiar with computational complexity in general.

(c) Choose and define a range of computational complexity classes, using the conventional, explicit approach.

(d) Categorise the above computational complexity classes in an implicit matter, and prove them equivalent to their explicit counterparts.

(e) Devise a programming language, where the programmer can state, and be statically guaranteed, a computational complexity bound on parts of the program.

5 Tasks

The thesis consists of two parts: and introductory part, and a body of work.

The introductory part is expected to be completed within the first working month, and includes the following tasks:

(1) Survey previous work on implicit complexity theory and contrast it to the explicit variant.

(2) Choose and define a range of computational complexity classes, using the conventional, explicit approach.

(3) Devise a programming language, where the programmer can state, and be statically guaranteed, a computational complexity bound on a program section, e.g. the entire program.

The rest of the thesis is concerned with implicitly categorising the chosen complexity classes, and extending the language above, providing for effectful bounded program sections, arriving perhaps, at a language useful in practice. This includes the following tasks:

(4) Categorise the above computational complexity classes in an implicit manner.

(5) Prove the implcit categorisations equivalent to their explicit counterparts.

(6) Extend the language above, providing for bounded and effectful program sections, where a programmer is limited to a subset of the language, reducible to an implicit categorisation of the stated bound.

The last task is expected to first include providing for effectful program sections bounded to P. We then provide for more fine-grained, subpolynomial bounds.

References

[Baillot et al. (2006)] Patrick Baillot, Jean-Yves Marion, and Simona Ronchi Della Rocca. *Special Issue on Implicit Computational Complexity*. February 2006. Workshop on Implicit Computational Complexity - Geometry of Computation. ACM, Marseille, France.

[Bellantoni & Cook (1992)] Stephen Bellantoni and Stephen Cook. *A New Recursion-Theoretic Characterization of the Polytime Functions*. 1992. Computational Complexity. Vol. 2, pp. 97–110. Birkhäuser Verlag, Basel, Switzerland.

[Jones (1999)] Niel Jones. *LOGSPACE and PTIME characterized by programming languages*. October 1999. Theoretical Computer Science. Vol. 228, No. 1-2, pp. 151–174. Elsevier.