Thesis Synopsis

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

Oleksandr Shturmov
oleks@oleks.info

June 3, 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 building a programming language where the programmer can state a bound on the computational complexity of a program section, and have those bounds statically guaranteed.

We proceed by limiting the programmer to a subset of the language within a bounded section, where the subset is reducible to an implicit categorisation of the stated bound.

The thesis proposal is thereby twofold:

(a) implicitly categorise a range of interesting complexity classes; and

(b) devise a programming language, supporting effectful program sections, bounded to the above classes.

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 should enable more complex computation.

4 Learning Objectives

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

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

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

(d) Categorise the above computational complexity classes in an implicit matter.

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

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

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 covers the learning obejctives (a)–(c) above.

The rest of the thesis is concerned with implicitly categorising the chosen complexity classes, and extending the language, enabling the programmer to write effectful bounded sections, arriving perhaps, at a language useful in practice. This covers the learning objectives (d)–(f).

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 (GEOCAL'06). Marseille, France.