Reasonable Space for the λ-Calculus, Logarithmically[*]

Beniamino Accattoli[1], Ugo Dal Lago[2], and Gabriele Vanoni[2]

[1] Inria & École Polytechnique
beniamino.accattoli@inria.fr
[2] Università di Bologna & Inria
{ugo.dallago,gabriele.vanoni2}@unibo.it

Abstract

Can the λ-calculus be considered as a reasonable computational model? Can we use it for measuring the time and space consumption of algorithms? While the literature contains positive answers about time, much less is known about space. This paper presents a new reasonable space cost model for the λ-calculus, based on a variant over the Krivine abstract machine. For the first time, this cost model is able to account for logarithmic space. Moreover, we study the time behavior of our machine.

Programming Languages and Computational Complexity. While the first years of *theoretical* computer science were devoted to discover *what* is computable, later the focus shifted to *how* things are computed. The *extensional* view, was substituted by an *intensional* one. In particular, the use of resources needed to execute an *algorithm/program* was investigated, and it is still now a very active research area. The attention is mainly devoted to the time and space (*i.e.* memory) consumption of algorithms. Traditionally, the analysis is done considering Turing machines (TMs), where it is very clear what time and space are:

- *time:* the number of transitions of the TM;
- *space:* the maximum number of cells written during the computation of the TM.

This way, the definition of complexity classes, such as L, P, NP, PSPACE, seems to depend on the details of the machine model. To overcome this problem, Slot and van Emde Boas [4] coined the *invariance thesis*, which states:

Reasonable machine models simulate each other with polynomially bounded overhead in time and (multiplicative) constant overhead in space.

This way, complexity classes such as L, P, NP, PSPACE become machine independent, once one knows that a machine model is reasonable. For example, all variants of TMs turn out to satisfy the invariance thesis. The question now is if high-level programming languages have reasonable cost models too. The reader can think about programming styles and features such as *object orientation*, *higher-order functions*, *logic programming*, *first-class list manipulation*. It is not at all obvious how to measure time and space complexity when a programming language features these characteristics. We focus on the functional paradigm, taking the λ-calculus as its natural model.

The Time of the λ-Calculus. The λ-calculus comes with just one computational rule, which is β-reduction. We are interested in cost models where time is the number of of β-steps. The simulation of TMs into the λ-calculus is not problematic, since there is an encoding of TMs into the λ-calculus, due to Accattoli and Dal Lago [2], for which Turing machines are simulated within a linear overhead.

[*]This is an abstract of a paper accepted at LICS 2022.

The delicate part of showing that the number of β-steps is a reasonable time cost model lies in the simulation of the λ-calculus strategy into TMs, or another reasonable model, typically random access machines (RAMs). The difficulty stems from a degeneracy, called *size explosion* in the literature. The point is that the size of a term can grow exponentially during the evaluation (thus also requiring exponential time to be written down). To circumvent the exponential explosion in space, λ-terms are usually evaluated *up to sharing*, that is, in calculi with sharing constructs or abstract machines, that compute shared representations of the results. These representations can be exponentially smaller than the results themselves: explosiveness is then encapsulated in the sharing unfolding process (which itself has to satisfy some reasonable properties). Using this machinery, one is able to give the required polynomially bounded simulation of the λ-calculus into RAMs. [1] is a survey on the subject.

The Space of the λ-Calculus. The natural space cost model for the λ-calculus is the maximum size of λ-terms belonging to the reduction sequence. The problem is that this cost model *cannot* accommodate sub-linear space complexity, and thus cannot reflect, *e.g.*, algorithms in L. This is because if space is the maximum size of terms in an evaluation sequence, the first of which contains the input, then space simply *cannot be* sub-linear. How could we account for *logarithmic* reasonable space? One needs, as it is done with TMs, *log-sensitivity*, that is, a distinction between an immutable *input space*, which is not counted for space complexity (because otherwise the complexity would be at least linear), and a (possibly smaller) mutable *work space*, that is counted. Moreover, logarithmic space usually requires manipulating *pointers* to the input (which are of logarithmic size) rather than pieces of the input (which can be linear). Log-sensitivity thus seems to clash with the natural approach based on the rewriting of λ-terms, which does not distinguish between input and work space and that manipulates actual sub-terms rather than pointers.

Our Contribution. We solve the problem mentioned above by defining a variant of Krivine's abstract machine [3]. We prove that its space consumption is a *reasonable* cost model for the λ-calculus, accommodating also logarithmic complexity, this way solving a long standing open problem in the theory of the λ-calculus. Moreover, we are able to give to our machine a reasonable *time* cost model.

References

[1] Beniamino Accattoli. (In)Efficiency and Reasonable Cost Models. In *Proc. of 12th LSFA*, volume 338 of *ENTCS*, pages 23–43. Elsevier, 2017.

[2] Ugo Dal Lago and Beniamino Accattoli. Encoding turing machines into the deterministic lambda-calculus. *CoRR*, abs/1711.10078, 2017.

[3] Jean-Louis Krivine. A Call-by-name Lambda-calculus Machine. *Higher Order Symbol. Comput.*, 20(3):199–207, 2007.

[4] Cees F. Slot and Peter van Emde Boas. The problem of space invariance for sequential machines. *Inf. Comput.*, 77(2):93–122, 1988.