Multi Types, Evaluation Lengths, and Normal Forms (Work in Progress)

Beniamino Accattoli[1], Delia Kesner[2], and Stéphane Graham-Lengrand[3]

1 **INRIA & LIX, École Polytechnique, France,** beniamino.accattoli@inria.fr
3 **CNRS, INRIA & LIX, École Polytechnique, France,** lengrand@lix.polytechnique.fr
2 **IRIF, CNRS and Université Paris-Diderot, France,** kesner@irif.fr

Abstract

Multi types—aka non-idempotent intersection types—have been used to obtain quantitative bounds on higher-order programs, as pionereed by De Carvalho. Notably, they bound at the same time the number of evaluation steps *and* the size of the result. Recent results show that the number of steps can be taken as a reasonable time complexity measure. At the same time, however, these results show that multi types provide quite lax complexity bounds, because the size of the result can be exponentially bigger than the number of steps.

Starting from this observation, we refine a technique introduced by Bernadet & Lengrand to study the maximal strategy, in two directions. First, our type systems give *separate* information about evaluation lengths and result sizes. Second, we provide *exact* bounds for various evaluation strategies, both in the λ-calculus and in the linear substitution calculus, to stress the modularity of the approach.

Our works aims at both revisiting the literature and extending it with new results. Concerning the literature, it unifies De Carvalho and Bernadet & Lengrand via a uniform technique and a complexity-based perspective. The two main novelties, instead, are exact bounds for the leftmost strategy—the only strong strategy known to provide a reasonable complexity measure—and the discovery that the computing device hidden behind multi types is *exactly* the linear substitution calculus.

Digital Object Identifier 10.4230/LIPIcs.DICE.2018.

Background

Intersection types and multi types. Most type systems for the λ-calculus ensure termination. One of the cornerstones of the theory of λ-calculus is that intersection types *characterise* termination: not only typed programs terminate, but all terminating programs are typable as well [8, 9, 17]. In fact, the λ-calculus comes with different notions of evaluation (*e.g.* call-by-name, call-by-value, to weak / head / full normal form, etc) and different notions of termination (weakly or strongly normalizing) and, accordingly, with different systems of intersection types.

Intersection types are a flexible tool and, even when one fixes notions of evaluation and termination, the type system can be formulated in various ways. A flavor that became quite fashionable in the last 10 years is that of *non-idempotent* intersection types [13, 10] (a survey can be found in [7]), where the intersection $A \cap A$ is not equivalent to A. Non-idempotent intersections can be seen as multi-sets [6], which is why, to ease the language, we prefer to call them *multi types* rather than *non-idempotent intersection types*.

Multi types have two main features:

Developments in Implicit Computational Complexity.

Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

1. *Bounds on evaluation lengths*: they go beyond simply characterising termination, as typing derivations provide bounds on the length of evaluation (*i.e.*, on the number of β-steps). Therefore, they give intensional insights on programs, and seem to provide a tool for complexity guarantees about programs.
2. *Linear logic interpretation*: multi types are deeply linked to linear logic. The relational model of linear logic (often considered as a sort of canonical model of linear logic) is based on multi-sets and multi types can be seen as a syntactic presentation of the relational model of the λ-calculus induced by the interpretation into linear logic. Morally, multi types are collections of linear typings, and their bounding capability comes from the fact that linear resources have to be used exactly once.

These two facts together have a potential, fascinating consequence: they suggest that denotational semantics may provide abstract tools for complexity analyses, and in a way that is theoretically solid, being grounded on linear logic.

Reasonable cost models. Various works in the literature explore the bounding power of multi types [14, 15, 16]. They apply a measure to typing derivations and show that the measure provides a bound on the length of the evaluation of the typed term. A further step, is to show that for some carefully chosen derivations the measure gives the *exact* number of β-steps [11, 3, 4] (or cut-elimination steps in proof nets [12]). A criticism often raised against these works is, or rather was, that the number of β-steps of the bounded evaluation strategies might not be a reasonable cost model, that is, it might not be a reliable complexity measure. Such a criticism was however light. Not because it is not a serious point, but because no reasonable cost models for the λ-calculus were known at the time (only solutions to very special cases were known)—the problem, essentially, was not specific to multi types but, more generally, it concerned the whole theory of λ-calculus.

The understanding of cost models for the λ-calculus made significant progress in the last few years. Since the nineties, it is known that the number of steps for *weak* strategies (*i.e.* not reducing under abstraction) is a reasonable cost model [5] (where *reasonable* precisely means polynomially related to the cost model of Turing machines). It is only in 2014, however, that a solution for the general case has been obtained: the length of leftmost evaluation to normal form is a reasonable cost model, as shown by Accattoli and Dal Lago [1]. As it often happens with long-standing problems, progress required new techniques and a new understanding of the problem. In this work we essentially update the study of the bounding power of multi types with the insights coming from the study of reasonable cost models. In particular, we try to answer to the just mentioned criticism and to provide new elements to the question of whether denotational semantics can really be used as a tool for complexity analyses.

Size explosion and lax bounds. The answer to the criticism, is now available, and yet it is somewhat unexpected. It is true, indeed, that the number of β-steps for the considered strategies is a reasonable cost model. Therefore, the historical criticism to the information provided by multi types was not justified. On the other hand, the study of cost models made clear that the information contained in multi typings is too generous, because they also bound the size of the normal form. To explain this point properly, let us make a step back.

The skepticism about taking the number of β-steps as a reliable complexity measure comes from the *size explosion problem*, that is, the fact that in the λ-calculus the size of terms can grow exponentially with the number of β-steps. Precisely, in the λ-calculus there are families of terms $\{t_n\}_{n\in\mathbb{N}}$ where t_n has size linear in n, it evaluates to normal form in n β-steps, and produces a result p_n of size $\Omega(2^n)$, *i.e.* exponential in n. Moreover, the size explosion problem is extremely robust, as there are families for which the size explosion is independent of the evaluation strategy. The difficulty in proving that the length of a

given strategy provides a reasonable cost model is precisely the fact that one needs a way to simulate the strategy (with some graph rewriting, explicit substitutions, or abstract machine mechanism) without explicitly computing the full normal form, that is huge and would be too expensive, but only a compact representation of it.

Now, multi typings do bound the number of β-steps of reasonable strategies, but they always also bound the size of the normal form. Therefore, even minimal typings provide a bound that in some cases is exponentially worse than the number of β-steps. And there is even worse: even *the minimum type itself*, not only the typing derivation, bounds the size of the normal form. Note that there is no contradiction with the literature: previous work used to apply *measures* to typings, and the role of these measures, among other things, was to subtract the size of the normal form from the size of the typing.

Our observation is that the typing themselves are in fact much bigger, and so the widespread point of view for which multi types—and so the relational model of linear logic—faithfully capture evaluation lengths, or even the complexity, is misleading.

Contributions

The tightening technique. Our starting point is the refinement of a technique introduced by Bernadet & Lengrand in a technical report [2]. Our type systems are capable of bounding *separately* the number of β-steps and the size of the normal form. The key point is the use of some *tight* type constants with dedicated typing rules, that isolate the part of the typing derivation contributing to the normal form. Every typing derivation provides a bound, but *tight derivations*—that are derivations whose concluding sequent has only tight constants—provide exact separate bounds for evaluation lengths and normal forms.

It is natural to wonder how natural the tightening technique is—a malicious reader may indeed suspect that we are cooking up an ad-hoc way of measuring evaluation lengths, betraying the *linear-logic-in-disguise* spirit of multi types. To remove any doubt, we show that our tight typings are actually isomorphic to minimal multi typings withouth tight constants. Said differently, the tightening technique turns out to be a way of characterizing minimal typings. Let us point out that, in the literature, there are characterizations of minimal typings only for normal forms, and they extend to non-normal terms only *indirectly*, that is, by subject expansion of those for normal forms. Our approach, instead, provides a *direct* description, for any typable term.

New results: (head and) leftmost evaluation. Our first application of the *tightening* technique is to the head and leftmost evaluation strategies. The head case is the simplest possible one. The leftmost case is the natural iteration of the head one, and the only known strong strategy whose number of steps provides a reasonable cost model. Multi types characterising leftmost normalizing terms have been studied by Kesner and Ventura, but somewhat surprisingly the number of steps taken by the leftmost strategy has not been measured via multi-types before—therefore, this is a new result.

Literature: de Carvalho's work. The study of the head and the leftmost strategies, at first sight, seems to be a minor reformulation of de Carvalho's results about measuring via multi types the length of executions of Krivine abstract machine (shortened KAM)—implementing weak head evaluation—and of the iterated KAM—that implements leftmost evaluation [11]. The study of cost models is here enlightening: de Carvalho's iterated KAM does implement leftmost evaluation, but the overhead of the machine (that is taken into account by de Carvalho) is exponential in the number of β-steps, while we only measure the number of β-steps, thus providing a much more parsimonious measure, and the only

meaningful one (according to the current understanding of reasonable cost models for the λ-calculus).

Literature: Bernadet & Lengrand. We also apply the technique to the maximal strategy, that is, the strategy taking the maximum number of steps to normal form, if any, and diverging otherwise. The maximal strategy has been measured by Bernadet & Lengrand in [3], and it is also the case for which a preliminary form of the thightening technique was developed in [2].

With respect to their work, here we provide a simpler technical development. On the one hand, because we employ a λ-calculus with a memory operator, in the style of Klop. On the other hand, because the study of the other cases inspires a more uniform and smooth development. We also show how to obtain the same results without the memory operator, and discuss the technical challenges posed by the maximal case.

New results: the linear substitution calculus. Last, we apply the tightening technique to linear head evaluation (lh for short), formulated in the linear substitution calculus (LSC). The literature contains a characterization of lh-normalizable terms due to Kesner and Ventura [14], and de Carvalho's measure of the executions of the KAM can be interpreted as a measure of lh-evaluations. What we show however is stronger, and somewhat unexpected.

To bound lh-evaluation, in fact, we do not have to change almost anything. The result for the exact bounds for head evaluation takes into account, for the bounds, only the number of abstraction and application typing rules. For *linear* head evaluation, we simply need to count also the axioms, that is, the rules typing variable occurrences, nothing else. It turns out then that the length of a linear head evaluation plus the size of the linear head normal form is *exactly* the size of the tight typing. Said differently, multi typings simply code evaluations in the LSC.

Let us stress it once more, differently. We do not have to adapt multi types to the LSC, as for instance de Carvalho does to deal with the KAM. It actually is the other way around. As they are, multi typings naturally measure evaluations in the LSC. To measure evaluations in the λ-calculus, instead, one has to forget the role of the axioms. One may say that multi types then fit better the LSC than the λ-calculus. The best way to put it, probably, is that the LSC is the computing device behind multi types.

Acknowledgements. This work has been partially funded by the ANR JCJC grant COCA HOLA (ANR-16-CE40-004-01).

References

1 B. Accattoli and U. D. Lago. (Leftmost-Outermost) Beta-Reduction is Invariant, Indeed. *LMCS*, 12(1), 2016.

2 A. Bernadet and S. Graham-Lengrand. A big-step operational semantics via non-idempotent intersection types, 2013.

3 A. Bernadet and S. Lengrand. Complexity of strongly normalising λ-terms via non-idempotent intersection types. In *FOSSACS 2011*, pages 88–107, 2011.

4 A. Bernadet and S. Lengrand. Non-idempotent intersection types and strong normalisation. *Logical Methods in Computer Science*, 9(4), 2013.

5 G. E. Blelloch and J. Greiner. Parallelism in sequential functional languages. In *FPCA*, pages 226–237, 1995.

6 A. Bucciarelli, D. Kesner, and S. R. D. Rocca. The inhabitation problem for non-idempotent intersection types. In *IFIP TCS 2014*, pages 341–354, 2014.

7 A. Bucciarelli, D. Kesner, and D. Ventura. Non-idempotent intersection types for the lambda-calculus. *Logic Journal of the IGPL*, 25(4):431–464, 2017.

8 M. Coppo and M. Dezani-Ciancaglini. A new type assignment for lambda-terms. *Archive for Mathematical Logic*, 19:139–156, 1978.

9 M. Coppo and M. Dezani-Ciancaglini. An extension of the basic functionality theory for the λ-calculus. *Notre Dame Journal of Formal Logic*, 4:685–693, 1980.

10 D. de Carvalho. *Sémantiques de la logique linéaire et temps de calcul.* These de doctorat, Université Aix-Marseille II, 2007.

11 D. de Carvalho. Execution time of lambda-terms via denotational semantics and intersection types. *CoRR*, abs/0905.4251, 2009.

12 D. de Carvalho, M. Pagani, and L. Tortora de Falco. A semantic measure of the execution time in linear logic. *Theor. Comput. Sci.*, 412(20):1884–1902, 2011.

13 P. Gardner. Discovering needed reductions using type theory. In *TACS '94*, pages 555–574, 1994.

14 D. Kesner and D. Ventura. Quantitative types for the linear substitution calculus. In *IFIP TCS 2014*, pages 296–310, 2014.

15 D. Kesner and D. Ventura. A resource aware computational interpretation for herbelin's syntax. In *ICTAC 2015*, pages 388–403, 2015.

16 D. Kesner and P. Vial. Types as resources for classical natural deduction. In *FSCD 2017*, pages 24:1–24:17, 2017.

17 J.-L. Krivine. λ-*calcul, types et modèles.* Masson, 1990.