

Intersection types and Resource control in λ -calculus

Stéphane Lengrand¹ and Delia Kesner²

¹ LIX, Ecole Polytechnique Lengrand@LIX.Polytechnique.fr
² PPS, Université Paris 7 Kesner@PPS.Jussieu.fr

Webpage:

<http://www.lix.polytechnique.fr/~lengrand/Work/Sujets/TypeInter-en.html>

Keywords: λ -calculus, Typing system with intersection types, Explicit control of resources.

1 Background

The λ -calculus is a programming model based on functions. Its Turing-completeness comes from the freedom given to function constructions and function applications, but the λ -calculus thereby lacks some notions of control that are useful and natural both in programming and in mathematics.

Typing systems provide such control, for instance by allowing the application of a function f , of “type” $A \rightarrow B$, to an argument e only if the latter is indeed of “type” A .

Basic typing systems will thus forbid a function to be applied to itself $f(f)$, but they can be extended using *intersection types*: if f is of type $(A \rightarrow B) \cap A$ (i.e. both of type $A \rightarrow B$ and of type A), then the above application makes sense.

On the other hand, we introduced in [KL07] a version of λ -calculus with an explicit control of resources. An explicit component of this control is the *duplication operator*, which specifies the point in a program (and in its running) where some data (or more generally a *term*) needs to be duplicated. From this phenomenon come complexity issues, both in time and in space, and non-termination.

Intersection types allow some control over this phenomenon, since they give some information on the multiple usage of some data (c.f. $f(f)$). For instance it is known that some typing systems using intersection types characterise those programs that terminate, that is to say a program terminate if and only if it can be typed in the system.

2 Objectives of the internship

The main goal of the internship is to study the interaction between intersection types and the aforementioned duplication operator.

The first approach will be based on the following typing rule:

given a well-typed program p that uses some data x of type A and some data y of type B , the program $C_z^{x,y}(p)$ that duplicates the data z to provide x and y to p is well-typed if z is of type $A \cap B$.

The student will design a typing system based on this rule and will establish its basic properties (typing is an invariant of program runs, ...).

The student will establish the equivalence with more traditional intersection type systems, and will see how such an equivalence holds (or not) for some variants of his system.

Among such variants, a crucial step is made when *idempotence* of intersection is dropped, namely the property $A \cap A = A$. Such a step enhances types with some

information about the number of times some data is used, from which one hopes to derive properties of program complexity.

D. de Carvalho thus has a typing system using non-idempotent intersection, which allows to compute the exact number of computation steps in the run of a typed program. This result is part of a more general research area investigating variants of typed λ -calculus that guarantee some complexity properties, such as polynomial time. The student will connect these approaches with the explicit handling of data/term duplication offered by the duplication operator $\mathcal{C}_z^{x,y}(_)$.

Finally, the student will consider decision or semi-decision algorithms for the proposed typing systems and potentially implement them. He will look towards formalising the investigated theories in a proof assistant such as Coq.

3 Practical questions

The internship supervision will be mainly at LIX / Ecole Polytechnique with Stéphane Lengrand, but, according to convenience, also at the Laboratoire PPS (métro Chevaleret, 13ème), with Delia Kesner. This would put the student in contact with two research groups.

For this internship please contact Stéphane Lengrand at Lengrand@LIX.Polytechnique.fr.

References

- [KL07] D. Kesner and S. Lengrand. Resource operators for the λ -calculus. *Inform. and Comput.*, 205:419–473, 2007.