The Journal of Open Source Education

Mikrokosmos: an educational lambda calculus interpreter

Mario Román[1]

1 University of Granada

DOI: 10.21105/jose.00029

Software

- Review
- Repository
- Archive

Submitted: 05 August 2018
Published: 24 October 2018

Description

Mikrokosmos is an educational untyped and simply typed lambda-calculus interpreter. For students, it is a tool to learn lambda-calculus and intuitionistic logic by coding. For educators, it is a didactic resource, grounded in the theoretical implementation of a functional programming language, so that they can integrate it with other learning materials. Mikrokosmos can be used on three different environments: (1) as a *command line executable*, implementing a read-eval-print loop interpreter; (2) as a *Jupyter kernel*, executing code blocks from a Jupyter notebook (Jupyter Development Team, 2016); and (3) as a *Javascript web application*, that can be used in conjunction with web text editors such as Codemirror[1] to create an online programming environment.

Mikrokosmos provides a minimalist and unified syntax for both simply typed and untyped lambda-calculus. The same expressions can be used to define untyped and typed terms. This helps highlighting the differences between them, and avoids the added complexity of two different syntaxes. Moreover, the syntax is heavily inspired by Haskell so that it facilitates transferring ideas to a fully-fledged functional programming language. Mikrokosmos comes also bundled with a self-documenting standard library of common lambda-calculus combinators and data structures which can be consulted directly from the interpreter and ease the learning process, while demonstrating how to write basic programs on lambda-calculus.

The interpreter focuses on being portable, close to the theory and suited for learning and experimentation. For the untyped lambda-calculus, Mikrokosmos works as a Turing-complete programming language. Following Barendregt (1984), it implements the *leftmost evaluation strategy*, which finds a normal form whenever it exists, allowing infinite data structures and fixed point definitions. The evaluation can be visualized step-by-step, showing how the internal representation of a lambda term using De Bruijn indexes works (De Bruijn, 1972). Optional coloring of the output facilitates this purpose. With regards to combinatorial logic, a translation between lambda terms and SKI combinators is available. It follows a standard algorithm from Hindley & Seldin (2008) that allows the user to input and visualize results in both formats. On the typed side, Mikrokosmos facilitates the study of the Curry-Howard isomorphism; see Sørensen & Urzyczyn (2006) for a description of this correspondence. To this end, it implements implicit typing à la Curry: the most general possible type is automatically inferred for any term, providing a limited form of polymorphism. Given any typed lambda expression, Mikrokosmos can visualize the logical derivation tree it codifies in the style of Gentzen, sharing the notation of introductory articles and textbooks such as those by Wadler (2015) or Girard, Taylor, & Lafont (1989).

[1] http://codemirror.net/

JOSE
The Journal of Open Source Education

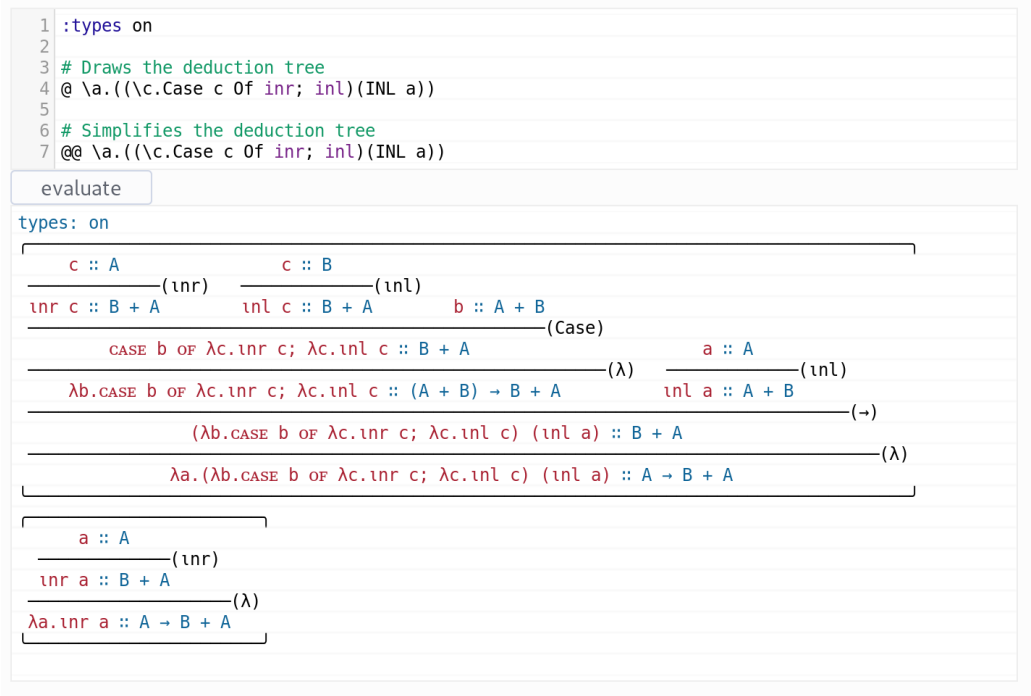

Figure 1. An example Gentzen diagram of a proof on intuitionistic logic.

The latest release of the interpreter can be installed via `stack` or `cabal` from Hackage[2] and the documentation can be found in https://mroman42.github.io/mikrokosmos/, where installation and usage instructions for the Jupyter kernel and the Javascript web application are also provided.

Statement of need

When teaching the logical structure of lambda-calculus, it is common to ask the students to evaluate lambda terms by hand. This approach seems necessary on the first stages of the learning process, but turns out to be very tedious while dealing with complex terms. Just as one would never teach C programming language by making students execute large programs by hand, it makes sense to provide an interpreter and a programming environment allowing the students to explore lambda-calculus without having to worry about the details and focusing on the big picture. Because of this, there is a need for a lambda-calculus interpreter. Untyped lambda-calculus interpreters are common, but they are usually developed as programming exercises, and only provide ascetic interpreters lacking any didactic features. Mikrokosmos addresses the need for an educational interpreter with multiple ways of visualizing the results, a clean and coherent syntax, libraries, and the possibility of integration in web pages and Jupyter notebooks.

The Curry-Howard isomorphism between simply-typed lambda calculus and propositional intuitionistic logic as described by Wadler (2015) is sometimes illustrated using the Haskell programming language or other similar functional languages. This approach, however, is theoretically questionable. On the one hand, Haskell implements a polymorphic lambda calculus corresponding to second-order intuitionistic logic, as depicted by Wadler (2007); but it is not total and makes every type inhabited, which is a fatal flaw for our purpose. On the other hand, it fails to represent a cartesian closed category due to its support for strict evaluation, partial functions and undefined values; as justified by Danielsson, Hughes, Jansson, & Gibbons (2006). Even if these deviations from the theory are justified

[2] https://hackage.haskell.org/package/mikrokosmos

by the needs of a real-world programming language, a student faced with Haskell as an example of the "Propositions as Types" paradigm may find it difficult to determine exactly what parts of the language should be ignored in order to retain a sound interpretation. Mikrokosmos, on the contrary, simply implements the internal language of a bicartesian closed category, following, for instance, Lambek & Scott (1989), and avoiding any spurious additions.

Simply-typed lambda calculus interpreters of this kind are rare; the existing software usually implements some particular primitive types, input-output, and tends not to use Curry typing. It is not designed to allow the study of the Curry-Howard correspondence.

Mikrokosmos has already been tested on the classroom and its web application can be easily deployed to create interactive tutorials suited for the needs of each specific course. In particular, Mikrokosmos and its interactive tutorial have already been used as a module for the course "Lógica y Programación" (Logic and Programming) taught by Pedro A. García-Sánchez at the University of Granada. The students were able to complete the exercises of the tutorial and to use the interpreter as an aid while completing other problem sets of the course.

Acknowledgments

This development was partially supported by the "Beca de Colaboración de Estudiantes en Departamentos Universitarios para el curso académico 2017-2018" from the Spanish Ministerio de Educación, Cultura y Deporte at the Departamento de Álgebra of the Universidad de Granada. The author would like to thank his advisor for this grant program, professor Pedro A. García-Sánchez, for his patient supervision, advice and encouragement.

References

Barendregt, H. (1984). *The lambda calculus: Its syntax and semantics*. Studies in logic and the foundations of mathematics. North-Holland. doi:10.1016/c2009-0-14341-6

Danielsson, N. A., Hughes, J., Jansson, P., & Gibbons, J. (2006). Fast and loose reasoning is morally correct. In *ACM SIGPLAN notices* (Vol. 41, pp. 206–217). ACM; Association for Computing Machinery (ACM). doi:10.1145/1111320.1111056

De Bruijn, N. G. (1972). Lambda calculus notation with nameless dummies, a tool for automatic formula manipulation, with application to the Church-Rosser theorem, *75*(5), 381–392. doi:10.1016/1385-7258(72)90034-0

Girard, J.-Y., Taylor, P., & Lafont, Y. (1989). *Proofs and types* (Vol. 7). Cambridge University Press Cambridge. doi:10.2307/2274726

Hindley, J. R., & Seldin, J. P. (2008). *Lambda-calculus and combinators: An introduction*. Cambridge University Press. doi:10.1017/cbo9780511809835

Jupyter Development Team. (2016). Jupyter Notebooks. A publishing format for reproducible computational workflows, 87–90. doi:10.3233/978-1-61499-649-1-87

Lambek, J., & Scott, P. J. (1989). *Introduction to higher-order categorical logic*. (J. L. Bell, Ed.)*The Journal of Symbolic Logic* (Vol. 54). Cambridge University Press. doi:10.2307/2274784

Sørensen, M. H., & Urzyczyn, P. (2006). *Lectures on the Curry-Howard isomorphism* (Vol. 149). Elsevier. doi:10.1016/s0049-237x(06)x8001-1

Wadler, P. (2007). The Girard–Reynolds isomorphism. *Theoretical Computer Science*, *375*(1-3), 201–226. doi:10.1016/j.tcs.2006.12.042

Wadler, P. (2015). Propositions as types. *Communications of the ACM*, *58*(12), 75–84. doi:10.1145/2699407