Algebraic subtyping for algebraic effects and handlers

AXEL FAES, KU Leuven
AMR HANY SALEH, KU Leuven
TOM SCHRIJVERS, KU Leuven

Algebraic effects and handlers benefit from a custom type-&-effect system, a type system that also tracks which effects can happen in a program. Several such type-&-effect systems have been proposed in the literature, but all are unsatisfactory. Recently, Stephen Dolan (University of Cambridge, UK) presented a novel type system that combines subtyping and parametric polymorphism in a particulary attractive and elegant fashion. A cornerstone of his design are the algebraic properties that the subtyping relation should respect. In this work, a type-&-effect system is derived that extends Dolan's elegant type system with effect information. This type-&-effect system inherits Dolan's harmonious combination of subtyping (in our case induced by a lattice structure on the effect information) with parametric polymorphism and preserves all of its desirable properties (both low-level algebraic properties and high-level meta-theoretical properties like type soundness and the existence of principal types).

Additional Key Words and Phrases: algebraic effect handler, algebraic subtyping, effects, optimised compilation

Contents

List of Figures

List of Tables

ACKNOWLEDGMENTS

I would like to thank Amr Hany Saleh for his continuous guidance and help.

REFERENCES

[1] Andrej Bauer and Matija Pretnar. 2014. An Effect System for Algebraic Effects and Handlers. *Logical Methods in Computer Science* 10, 4 (2014). https://doi.org/10.2168/LMCS-10(4:9)2014
[2] Andrej Bauer and Matija Pretnar. 2015. Programming with algebraic effects and handlers. *J. Log. Algebr. Meth. Program.* 84, 1 (2015), 108–123. https://doi.org/10.1016/j.jlamp.2014.02.001
[3] Edwin Brady. 2013. Programming and reasoning with algebraic effects and dependent types. In *Proceedings of the 18th ACM SIGPLAN International Conference on Functional Programming*. ACM, 133–144.
[4] Stephen Dolan and Alan Mycroft. 2017. Polymorphism, Subtyping, and Type Inference in MLsub. In *Proceedings of the 44th ACM SIGPLAN Symposium on Principles of Programming Languages (POPL 2017)*. ACM, New York, NY, USA, 60–72. https://doi.org/10.1145/3009837.3009882
[5] Stephen Dolan, Leo White, KC Sivaramakrishnan, Jeremy Yallop, and Anil Madhavapeddy. 2015. Effective Concurrency through Algebraic Effects. In *OCaml Workshop*.

Authors' addresses: Axel Faes, Department of Computer Science, KU Leuven, axel.faes@student.kuleuven.be; Amr Hany Saleh, Department of Computer Science, KU Leuven, amrhanyshehata.saleh@kuleuven.be; Tom Schrijvers, Department of Computer Science, KU Leuven, tom.schrijvers@kuleuven.be.

[6] Axel Faes and Tom Schrijvers. 2017. A core language with row-based effects for optimised compilation. In *Student Research Competition*. ICFP.

[7] Daniel Hillerström and Sam Lindley. 2016. Liberating Effects with Rows and Handlers. In *Proceedings of the 1st International Workshop on Type-Driven Development (TyDe 2016)*. ACM, New York, NY, USA, 15–27. https://doi.org/10.1145/2976022.2976033

[8] Daniel Hillerström, Sam Lindley, and KC Sivaramakrishnan. 2016. Compiling Links Effect Handlers to the OCaml Backend. In *OCaml Workshop*.

[9] J Roger Hindley and Jonathan P Seldin. 1986. *Introduction to Combinators and (lambda) Calculus*. Vol. 1. CUP Archive.

[10] Ohad Kammar, Sam Lindley, and Nicolas Oury. 2013. Handlers in action. In *Proceedings of the 18th ACM SIGPLAN International Conference on Functional programming (ICFP '14)*. ACM, 145–158.

[11] Ohad Kammar and Gordon D. Plotkin. 2012. Algebraic foundations for effect-dependent optimisations. In *Proceedings of the 39th Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*. ACM, 349–360.

[12] Oleg Kiselyov and KC Sivaramakrishnan. 2016. Eff Directly in OCaml. In *OCaml Workshop*.

[13] Daan Leijen. 2014. Koka: Programming with row polymorphic effect types. *arXiv preprint arXiv:1406.2061* (2014).

[14] Daan Leijen. 2017. Type Directed Compilation of Row-typed Algebraic Effects. In *Proceedings of the 44th ACM SIGPLAN Symposium on Principles of Programming Languages (POPL 2017)*. ACM, New York, NY, USA, 486–499. https://doi.org/10.1145/3009837.3009872

[15] Sam Lindley, Conor McBride, and Craig McLaughlin. 2017. Do Be Do Be Do. In *Proceedings of the 44th ACM SIGPLAN Symposium on Principles of Programming Languages (POPL 2017)*. ACM, New York, NY, USA, 500–514. https://doi.org/10.1145/3009837.3009897

[16] Benjamin C Pierce. 2002. *Types and programming languages*.

[17] Gordon D. Plotkin and John Power. 2003. Algebraic Operations and Generic Effects. *Applied Categorical Structures* 11, 1 (2003), 69–94.

[18] Gordon D. Plotkin and Matija Pretnar. 2008. A Logic for Algebraic Effects. In *Proceedings of the Twenty-Third Annual IEEE Symposium on Logic in Computer Science, LICS 2008, 24-27 June 2008, Pittsburgh, PA, USA*. IEEE Computer Society, 118–129.

[19] Gordon D. Plotkin and Matija Pretnar. 2013. Handling Algebraic Effects. *Logical Methods in Computer Science* 9, 4 (2013). https://doi.org/10.2168/LMCS-9(4:23)2013

[20] Matija Pretnar. 2014. Inferring Algebraic Effects. *Logical Methods in Computer Science* 10, 3 (2014). https://doi.org/10.2168/LMCS-10(3:21)2014

[21] Matija Pretnar. 2015. An introduction to algebraic effects and handlers. invited tutorial paper. *Electronic Notes in Theoretical Computer Science* 319 (2015), 19–35.

[22] Matija Pretnar. 2018. Eff. https://github.com/matijapretnar/eff. (2018).

[23] Matija Pretnar, Amr Hany Saleh, Axel Faes, and Tom Schrijvers. 2017. *Efficient compilation of algebraic effects and handlers*. Technical Report CW 708. KU Leuven Department of Computer Science.

[24] Didier Rémy. 1994. Theoretical Aspects of Object-oriented Programming. MIT Press, Cambridge, MA, USA, Chapter Type Inference for Records in Natural Extension of ML, 67–95. http://dl.acm.org/citation.cfm?id=186677.186689

[25] Amr Hany Saleh, Georgios Karachalias, Matija Pretnar, and Tom Schrijvers. 2018. Explicit Effect Subtyping. In *European Symposium on Programming*. Springer, 327–354.