Algebraic Subtyping for Algebraic Effects and Handlers

Student: Axel Faes
Promotor: Prof. dr. ir. Tom Schrijvers
Daily supervisor: Prof. dr. ir. Tom Schrijvers
Research group: Declarative Languages and Artificial Intelligence

1 Summary

This topic is situated in the area of Programming Language Theory, the formal study of programming language features and their properties. Specifically, you will design a novel type system and study its formal properties. This type system will be focused on algebraic effects and handlers.

2 Background

This thesis builds on two very recent developments in the area of programming language theory:

1. In his December 2016 PhD thesis, Stephen Dolan (University of Cambridge, UK), has 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.
2. "Algebraic effects and handlers" are a new formalism for formally modelling side-effects (e.g. mutable state or non-determinism) in programming languages, developed by Matija Pretnar (University of Ljubjana) and Gordon Plotkin (University of Edinburgh). This approach is gaining a lot of traction, not only as a formalism but also as a practical feature in actual programming languages (e.g. the Koka language developed by Microsoft Research). We are collaborating with Matija Pretnar on the efficient implementation of one such language, called Eff. Axel Faes has contributed to this collaboration during a project he did for the Honoursprogramme of the Faculty of Engineering Science.

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. We attribute this to the lack of the elegant properties of Dolan's type system. Indeed the existing type-&-effect systems are not only theoretically unsatisfactory, but they are also akward to implement and use in practice.

3 Research questions

- How can Dolan's elegant type system be extended with effect information?
- Which properties are preserved and which aren't preserved?
- What advantages are there to an type-&-effect system based on Dolan's elegant type system?

4 Goal

The goal of this thesis is to derive a type-&-effect system that extends Dolan's elegant type system with effect information. This type-&-effect system should inherit Dolan's harmonious combination

of subtyping (in our case induced by a lattice structure on the effect information) with parametric polymorphism and preserve 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). Afterwards this type-&-effect system

5 Approach

1. Study of the relevant literature and theoretical background.
2. Design of a type-&-effect system derived from Dolan's, that integrates effects.
3. Proving the desirable properties of the proposed type-&-effect system: type soundness, principal typing, ...
4. Time permitting: Design of a type inference algorithm that derives the principal types of programs without type annotations and proving its correctness.
5. Time permitting: Implementation of the algorithm and comparing it to other algorithms (such as row polymorphism based type-&-effect systems).

References

[1] Matija Pretnar. *An Introduction to Algebraic Effects and Handlers*. 2015. URL: http://www.eff-lang.org/handlers-tutorial.pdf.

[2] Stephen Dolan. *Algebraic Subtyping*. 2016. URL: https://www.cl.cam.ac.uk/~sd601/thesis.pdf.

[3] Stephen Dolan. *Prototype type inference engine*. 2016. URL: https://github.com/stedolan/mlsub.

[4] Stephen Dolan and Alan Mycroft. *Polymorphism, Subtyping, and Type Inference in MLsub*. 2016. URL: http://www.cl.cam.ac.uk/~sd601/lfcs_slides.pdf.

[5] Stephen Dolan and Alan Mycroft. "Polymorphism, Subtyping, and Type Inference in MLsub". In: *Proceedings of the 44th ACM SIGPLAN Symposium on Principles of Programming Languages*. POPL 2017. Paris, France: ACM, 2017, pp. 60–72. ISBN: 978-1-4503-4660-3. DOI: 10.1145/3009837.3009882. URL: http://doi.acm.org/10.1145/3009837.3009882.

[6] Gordon D. Plotkin and Matija Pretnar. "Handling Algebraic Effects". In: *Logical Methods in Computer Science* 9.4 (2013). DOI: 10.2168/LMCS-9(4:23)2013. URL: http://dx.doi.org/10.2168/LMCS-9(4:23)2013.

[7] *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, 2008. ISBN: 978-0-7695-3183-0. URL: http://ieeexplore.ieee.org/xpl/mostRecentIssue.jsp?punumber=4557886.

[8] Gordon D. Plotkin and Matija Pretnar. "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, 2008, pp. 118–129. ISBN: 978-0-7695-3183-0. DOI: 10.1109/LICS.2008.45. URL: http://dx.doi.org/10.1109/LICS.2008.45.