Practical LADL

Automatic typing systems for formally specified languages

Research grant proposal
Martin Lundfall, dapphub, martin@dapp.org

August 1, 2018

Abstract

In [SM16], Stay and Meredith proposes a general method for generating typing systems for programming languages under the slogan of *logic as a distributive law*. This grant proposal outlines the project of implementing this procedure as a tool to become part of the K framework, making the algorithm available for any language defined in K.

1 Theory

The idea behind *logic as a distributive law* (LADL) is that given a formal semantics of a programming language L, specifying its term structure and rewrite rules, and a language of *collections*, C, specifying the algebra of some abstract container (such as sets, bags, trees or lists), we can combine the two to get a type language $T = L + C$. The language T features the term constructors of both L and C combined, and given a translation of terms built using a constructor from L applied to terms from C to terms built using a constructor of C applied to terms from L (in other words a distributive natural transformation $\delta : C \circ L \implies L \circ C$), we can produce a type checker which interprets the types of T as a collection of terms of L.

The procedure is presented using the theory of enriched lawvere theories to represent operational semantics, a framework which, while providing very succinct descriptions and promising mathematical results [WB18], glosses over much of the technical details associated with the practice of formalizing programming languages.

One of the most advanced tool sets for programming language formalization and analysis is the $\mathbb{K}$ framework [SPY+16]. It was developed on practical grounds to provide a convenient formalism for represent programming languages with an emphasis on universality, modularity, non-determinism, concurrency. The developers of $\mathbb{K}$ felt like neither of the two traditional theoretical frameworks, structural operational semantics, and rewrite theory were suitable for this task. [**?**] Only recently has the authors of K begun to identify its underlying theory as *Matching logic* [Ros17]. Matching logic, borrowing ideas from pattern matching provides a unified framework in which one can easily deal with concepts like variable binding, fixed points, unification/anti-unification, rewriting, reachability and contexts.

The aim of this project is to translate LADL into a foundation more suitable for direct implementation in the K framework, so that the algorithm can be implemented in $\mathbb{K}$. Whether the target of the translation will be matching logic, a flavor of rewrite theory or structural operational semantics, the goal is to make sure that the procedure will be available to languages with a formal $\mathbb{K}$ definition.

The implementation will be built using the new, haskell backend for the K framework, since it has a stronger theoretical foundation than the current java backend, and since the functional nature of haskell is more suitable for the ideas involved in implementing LADL. [Lab18].

Since this project aims to yield a concrete algorithm for generating type languages for arbitrary languages, we will need to use both practical and theoretical methods for development. For example, besides translating the theory of LADL to matching logic, we will continuously work with a number of example languages, such as the algebra of monoids, lambda calculus, rho calculus, and the folklore imperative language IMP to practically see how to automate the procedure, or where problems arise.

References

[Lab18] Formal Systems Laboratory, *The semantics of k*, work in progress, 2018.

[Ros17] Grigore Rosu, *Matching Logic*, Logical Methods in Computer Science **Volume 13, Issue 4** (2017), doi:10.23638/LMCS-13(4:28)2017, https://lmcs.episciences.org/4153.

[SM16] Mike Stay and Lucius Gregory Meredith, *Logic as a distributive law*, preprint, 2016, arXiv:1610.02247.

[SPY+16] Andrei Stefănescu, Daejun Park, Shijiao Yuwen, Yilong Li, and Grigore Roşu, *Semantics-based program verifiers for all languages*, ACM SIGPLAN Notices, vol. 51, ACM, 2016, pp. 74–91.

[WB18] Christian Williams and John C. Baez, *Operation through enrichment*, work in progress, 2018.