Universal Algebra for Syntax with Bindings

Lorenzo Gheri

Middlesex University, London, lg571@live.mdx.ac.uk

Abstract: This is a short outline of my Ph.D. project plans. This is a work in progress which aims at a general theory of Syntax with Bindings, which will come together with a formalization in Isabelle/HOL and some applications. The major contributions will be dealing with coinduction, infinite objects and generalized notions of bindings.

1 Introduction

The central aim of the project we are here presenting is to develop a framework for specification of, and reasoning about, *formal systems* of relevance in computer science and mathematical logic. We call "formal system" any logical theory comprehending a syntax and some semantics; these structures are heavily employed in modelling the semantics of programming languages, type systems and logical deduction.

Notably, the area of programming languages has seen an explosion of formal systems aimed at capturing a wide range of computational and logical analysis aspects. For example, there is research in making programming languages more secure and specification-conforming, reflected in a variety of type systems (see Pierce's monograph [5]) and formal analysis tools, such as proof assistants , model checkers , J-Flow and other security tools, CSP-based tools etc. All this research relies on methodologies for rigorous, formal reasoning and computation.

Even though formal systems come in a wide variety, there are some fundamental mechanisms that seem common to all (or most) of them. These include the notions of *binding* and *scoping* aimed at dealing with locality: a program variable can be bound in a certain submodule (such as a function or a procedure) or a logic variable can be bound in the scope of a quantifier. Binding typically comes in pair with the notion of instantiation or substitution. For example, when a programming function is applied to a value, its execution proceeds by instantiating its formal parameter with that value and then evaluating the body of the function. In formal logic, this corresponds to the notion of substituting a term for a variable. Often, the operational behaviour of programs can be understood in terms of the interplay between bindings and instantiation/substitution. The laws that govern this interplay obey certain patterns that can be conceptualized in isolation from the particular language.

Another important aspect in formal systems is dealing with infinite behaviour. A finite syntactic object may produce an infinite semantic object by unfolding (as in operational semantics) or by interpretation (as in denotational semantics). The syntactic world is guided by bindings and substitution, whereas, in mirror, the semantic world is guided by functional objects and application. Likewise, the syntactic world can be specified *inductively*, whereas the semantic world is best describable *coinductively*.

Our work will contribute to the identification of such patterns and their presentation as a general theory, in the style of Universal Algebra, widely applicable for many formal systems. This work will be guided by concrete applications, including well-known challenges in formal reasoning and cases of notoriously difficult or tedious constructions. We plan to validate and illustrate our results on concrete case studies conducted in the proof assistant Isabelle/HOL [1].

2 The General Framework

The starting point of this project is that formal systems share a common syntactic structure. With the development of a unifying theory for syntax with bindings we aim at building a formalized framework, which can capture many formal systems and constructions as particular instances. In this section we sketch an embryonic version of the syntactic part of this framework.

First we need to be given a set of *sorts*, which are specified when the framework is instantiated. The intuition is that each object of the system is of a certain sort; if a sort is allowed to contain variables, it is said a *variable sorts*.

Example. In simply typed lambda calculus there are just two sorts, one for terms and one for types. As we need term variables, terms (but not types) are elements of a certain variable sort.

The next step is to populate our system. For every sort we give some elements, *terms*, which will form the respective *syntactic category*. Usually some auxiliary objects are given, as a base case; we call these the *context*. For example, the context must contain a countably infinite set of *variables* for every variable sort. The way we build every other element is with a set of *constructors*, given for the specific formal system. Every constructor comes with its *arity*; here we must also declare which are the *bound* variables in which term.

In short, we have a set, the syntactic category, for every sort, the elements of which are called *terms* and are defined inductively by means of constructors.

Example. Let us consider the constructors of the simply typed lambda calculus. The syntactic category Type of types has one constructor b for each type constant and another constructor $\to$ which takes two types and builds a type (the usual type of functions). The syntactic category Term of terms contains as a subset a countably infinite set of variables. Its constructor are:

- a constructor for every term constant c;
- a constructor for applications, which from two terms M and N builds a third one MN;
- a constructor for abstractions, which binds a variable x of a certain type σ in the term M, building the term $\lambda x : \sigma.M$.

As for inference systems, the theory of Nominal Logic has already taken care of inductive ones [6], by means of swapping, equivariance and freshness. For the coinductive case however, such a general theory still need to be developed.

To conclude, it is worth mentioning that from this presentation it is possible to move a first automatic step into the semantics of the formal system. First we need a *domain* (a set) for every sort, in which we will interpret the terms of that sort. If the sort is a variable sort we also need a *valuation*, namely a function that maps variables in elements of the appropriate domain. To conclude we must have a function for every constructor, in a way that the arity of the function matches the arity of the constructor. Once we are given these objects we can define the *interpretation* for every term by structural recursion, with a standard treatment for bindings.

3 Objectives

The central objective of our project is to develop a Universal Algebra for Syntax with Bindings, that improves on the state of art.

The first step will be to develop a solid and unitary theory, which addresses notoriously problematic features, especially substitution in terms modulo alpha equivalence, and integrates syntax and semantics. Here the progress will be in terms of generality, rigour and a formalization in Isabelle/HOL.

The universal theory I am proposing will also allow flexible bindings, including recursively specified bindings (e.g. *records* and *patterns* [2]).

While Nominal Logic [6] covers largely the theory for inductive objects, there are no such comprehensive works for coinductively defined objects involving bindings. Moreover our work will take care of codatatypes, and in general objects with no finite support. This means that for these it is not always possible to generate a fresh variable (i.e. different from every other in the object) just because their syntactic structure involves a finite number of variables. A significant example of these structure are infinite trees, as Bohm trees in Barendregt [3] or as the semantics naturally associated to infinite terms by repeatedly unfolding.

In short, here the main goal is to achieve a formal general treatment of infinite objects modulo α-equivalence and of corecursion and coinduction for binding structures.

Our project will also provide some general result for the denotational semantics of a generic formal system. In particular it will capture in a general way the substitution lemma and the fact that the interpretation of the term does not depend on the valuation for the non-free variables in that term.

Moreover, the framework is intended to grasp a more complex notion of bindings, for example the recursive binding of the System F (as in the POPLmark document [2] and in Pierce [5]). Here techniques from category theory will be heavily involved, such as Bounded Natural Functors [7].

In terms of applications, for example this framework will be suitable for a formalization of a real programming language. Moreover it will give a tool for a general treatment of adequacy in Higher Order Abstract Syntax, in contrast with the current literature characterized by different solutions to particular examples.

A deeper theory of binders could allow to faithfully represent the behaviour of actual bindings in languages such as Java, and also provide a mathematical model to the phenomenon of entanglement in quantum computing: we could imagine a quantum system and indeed a quantum program as a term of a certain syntax, in which the entanglement link between two particles is represented by an appropriate binding of two objects in the term.

References

[1] Isabelle home page, `https://isabelle.in.tum.de/`.

[2] Brian E. Aydemir, Aaron Bohannon, Matthew Fairbairn, J. Nathan Foster, Benjamin C. Pierce, Peter Sewell, Dimitrios Vytiniotis, Geoffrey Washburn, Stephanie Weirich, and Steve Zdancewic, *Mechanized Metatheory for the Masses: The POPLMark Challenge*, `http://www.seas.upenn.edu/~plclub/poplmark/`, 2005.

[3] Henk Barendregt, *The Lambda Calculus: its Syntax and Semantics*, revised ed., North-Holland, 1984.

[4] John C. Mitchell, *Foundations for Programming Languages*, MIT Press, 1996.

[5] Benjamin C. Pierce, *Types and Programming Languages*, MIT Press, 2002.

[6] Andrew M. Pitts, *Nominal Logic: A First Order Theory of Names and Binding*, Fourth International Symposium in Theoretical Aspects of Computer Software, TACS 2001, Sendai, Japan, October 29-31, 2001, Proceedings, volume 2215 of Lecture Notes in Computer Science, pages 219–242. Springer-Verlag, 2001.

[7] Dmitriy Traytel, Andrei Popescu, Jasmin Christian Blanchette, *Foundational, Compositional (Co)datatypes for Higher-Order Logic: Category Theory Applied to Theorem Proving*, LICS 2012, 596-605, IEEE.