J Autom Reasoning (2012) 49:111–114
DOI 10.1007/s10817-011-9217-2

Preface: Theory and Applications of Abstraction, Substitution and Naming

Maribel Fernández · Christian Urban

Received: 1 February 2011 / Accepted: 1 February 2011 / Published online: 2 March 2011

Mathematical treatments of concrete syntax have always been a central concern in symbolic computation, programming language implementation and computer-aided reasoning. Formal treatments of abstract syntax have proved harder to develop, especially those addressing properties related to substituting structures for variables, binding of names and fresh name generation.

The λ-calculus, a paradigmatic calculus of functions, includes a binder—λ. It is therefore quite natural that some of the first treatments of abstract syntax with binders have been based on extensions of first-order syntax that use the λ-calculus as meta-language. Recently, other approaches to the formalisation of syntax with binders have been proposed, for instance, based on set theories with atoms such as Zermelo-Fraenkel sets with atoms, or Fraenkel-Mostowski's set theory.

The wealth of recent results and the opening of new, exciting, research directions in the area led to the introduction of several dedicated international workshops. The first International Workshop on Theory and Applications of Abstraction, Substitution and Naming (TASSN) took place in Edinburgh in 2007, followed by a second workshop in York in 2009, associated to ETAPS.

This special issue is devoted to papers describing recent advances in this area. It came out of a call for papers issued after the second TASSN workshop, with the aim of including a selection of extended versions of workshop papers in addition to opening it up to other authors. Seven articles were selected for this special issue, covering the following topics:

M. Fernández
Department of Informatics, King's College London, Strand, London WC2R 2LS, UK
e-mail: Maribel.Fernandez@kcl.ac.uk

C. Urban (✉)
Institute for Computer Science at the TU Munich (IV), Boltzmannstrasse 3,
85748 Garching, Germany
e-mail: urbanc@in.tum.de

1 New Formalisation Techniques for Languages Involving Binders

Nominal techniques for the representation of languages with binding rely on a syntactical distinction between bindable variables (called atoms in the terminology of nominal syntax) and non-bindable variables, called meta-variables or just variables in nominal syntax. The article by Lakin and Pitts, *Encoding abstract syntax without fresh names*, introduces a variant of nominal abstract syntax in which bindable names are represented by normal meta-variables as opposed to a separate class of atoms. Since bindable names are represented by normal meta-variables, distinct meta-variables can be instantiated with the same concrete name, and explicit constraints are used to control this kind of aliasing. The nominal meta-programming language alphaML, designed by the same authors, is based on this approach. In *Encoding abstract syntax without fresh names*, Lakin and Pitts recall the main features of the language and develop a theory of contextual equivalence for it. The central result of the paper is that abstract syntax trees involving binders can be encoded into alphaML in such a way that their alpha-equivalence corresponds with contextual equivalence of their encodings. Unlike previous results on contextual equivalence using standard nominal terms, this new encoding does not rely on the existence of globally fresh names and fresh name generation.

An alternative technique for the representation of languages with binding is presented by Pollack and Sato in the article *A canonical locally named representation of binding*. The approach is based on a variant of first-order syntax which uses different syntactic classes for locally bound variables and global (or free) variables. A key ingredient in Pollack and Sato's approach is the choice of names to obtain a canonical representation of terms. In previous work, the authors used a particular concrete function for canonically choosing the names of binders, whereas in *A canonical locally named representation of binding* they show that the properties of such a choice function can be defined in an abstract way, and they develop the metatheory of the representation.

2 Categorical Theories of Names and Binding

Denotational semantics is one method of specifying meaning of languages. A presheaf category is presented as a denotational semantics for the calculus of explicit fusions by Bonchi et al. in the article *A presheaf environment for the explicit fusion calculus*. They extend Turi and Plotkin's ideas on category theoretic modelling of structural operational semantics to a presheaf category over an index category. With this they can deal with naming aspects of the fusion calculus, a variant of the π-calculus including asynchronous broadcasting of name equivalences.

3 Properties of Substitutions in Calculi that Support Binding

A well-known technique to describe in an abstract way implementations of calculi that include binding operators is based on the definition of capture-avoiding substitutions explicitly. This requires an extended syntax, where substitutions are first-class citizens, and additional rules to specify the action of substitutions on terms. In

the article *On Explicit Substitution with Names*, Rose, Bloo and Lang describe the origins of the family of calculi of explicit substitution with variable names known as λx, and study the properties of preservation of strong normalisation, confluence, and confluence on terms that include meta-variables. Moreover, the article discusses the relationship with other versions of explicit substitution calculi using names and garbage collection rules.

4 Mechanised Meta-Theory of Calculi Involving Binding

Many proofs about calculi that support binding are arduous, but often this is due to the length of the proof rather than conceptual difficulties. Since there are usually many cases to check, the subtle ones may easily be overlooked amongst the many straightforward cases. Thus, interactive proof assistants can be a very useful tool in the study of the meta-theory of calculi involving binding.

To formalise a typed calculus or programming language in a proof assistant two approaches are available. In the first one, usually called extrinsic, one first defines a type that encodes untyped object expressions and then makes a separate definition of typing judgements over the untyped terms. In the second one, the intrinsic approach, only well-typed object expressions are defined (ill-typed expressions cannot be expressed). Intrinsic encodings naturally enforce the required properties, for instance, metalanguage operations on object expressions respect object types, but the downside is that the metalanguage types of intrinsic encodings and operations involve dependency. In the article *Strongly Typed Term Representations in Coq*, Benton, Kennedy, Hur and McBride describe intrinsic encodings of simply-typed and polymorphic languages using the Coq proof assistant. The Coq types encoding object-level variables and terms are indexed by both type and typing environment. In the simply typed case, the definitions are free of any use of type equality coercions. In the polymorphic case, some substitution operations still require type coercions but the treatment is simplified by the uniform use of heterogeneous equality.

When developing or checking proofs, adequacy is an important criterion for judging the correctness of formal reasoning. In the article *Formalizing adequacy for higher-order abstract syntax*, Cheney, Norris and Vestergaard investigate and formalise connections between nominal and higher-order abstract syntax techniques in Isabelle/HOL using the Nominal Datatype Package. The article clarifies some subtle issues that were overlooked in the past in informal proofs of adequacy, and proposes a library of useful results as well as a model proof of adequacy, which can be used as a basis for automating adequacy proofs.

5 Meta-Languages and Tools for the Formalisation of Systems that Involve Binding

Programming languages and theorem provers are the standard examples of systems that involve binding. To formalise their properties, Gacek, Miller and Nadathur propose, in *A two-level logic approach to reasoning about computations*, a specification logic that can explicitly deal with binding in object languages. To prove theorems about specifications written in this logic, a second logic, called the reasoning logic, is proposed. Since the specification logic includes various notions of binding, the

reasoning logic supports the encoding of binding structures as well as their associated notions of scope, free and bound variables, and capture-avoiding substitution. In addition, the reasoning logic includes mechanisms for constructing proofs by induction and co-induction. The article describes the specification and reasoning logics and shows how provability in the specification logic can be transparently encoded in the reasoning logic. It also describes the implementation of this approach in the interactive theorem prover Abella and gives examples to illustrate the advantages of Abella to reason about computations.