Nominal Techniques for Software Specification and Verification

Maribel Fernández
Department of Informatics, King’s College London, UK

Abstract
In this talk we discuss the nominal approach to the specification of languages with binders and some applications to programming languages and verification.

2012 ACM Subject Classification Theory of computation → Logic and verification; Theory of computation → Lambda calculus; Theory of computation → Equational logic and rewriting

Keywords and phrases Binding operator, Nominal Logic, Nominal Rewriting, Unification, Equational Theories, Type Systems

Digital Object Identifier 10.4230/LIPIcs.FSCD.2023.1

Category Invited Talk

Funding This work is partially funded by the Royal Society (International Exchanges, grant number IES/R2/212106).

Acknowledgements The work described in this abstract is the result of collaborations with several researchers, cited below. Special thanks to my PhD students and co-authors.

1 Overview

The nominal approach to the specification of languages with binding operators, introduced by Gabbay and Pitts [27, 21, 20], has its roots in nominal set theory [26]. Its user-friendly syntax and first-order presentation (indeed, nominal logic [25] is defined as a theory in first-order logic) makes formal reasoning about binding operators similar to conventional on-paper reasoning.

Nominal logic uses the well-understood concept of permutation groups acting on sets to provide a rigorous, first-order treatment of common informal practice to do with fresh and bound names. Nominal matching and nominal unification [34, 35] (which work modulo α-equivalence) are decidable and efficient algorithms exist [7, 8, 22, 9], which are the basis for efficient implementations of nominal rewriting [19, 17, 18].

A number of systems (such as Nominal Isabelle [33]) highlighted the benefits of the nominal approach, which gave rise to elegant formalisations of Gödel’s theorems [24] and the π-calculus [5] and to advances in programming language semantics [23]. However, there are still some obstacles to the inclusion of nominal features in programming languages and verification environments.

In this talk, I will present our current work towards incorporating nominal techniques into two widely-used rule-based first-order verification environments: the K specification framework [29] and the Maude programming language [11, 12].

An important component of rule-based programming and verification environments is the algorithm used to check equivalence of terms and to solve equations (unification). In practice, unification problems arise in the context of equational axioms (e.g., to take into account associative and commutative (AC) operators [32, 31, 13, 14, 6]). The first part of the talk will discuss notions of α-equivalence modulo associativity and commutativity axioms [1],
Nominal Techniques

extensions of nominal matching and unification to deal with AC operators [2], and the use of nominal narrowing [3] to deal with equational theories presented by convergent nominal rewriting rules.

Another important component of these environments is the type system. In the second part of the talk, I will discuss type systems for nominal languages (including polymorphic systems [15] and intersection systems [4]). Dependent type theories, the dominant approach to formalising programming languages, have been extended with nominal features [10, 28, 30]. A lambda-less nominal dependent type system is available [16] and we are currently working on a type checker for this system.

The talk is structured as follows: we will start with the definition of nominal logic (including the notions of fresh atoms and alpha-equivalence) followed by a brief introduction to nominal matching and unification. We will then define nominal rewriting, a generalisation of first-order rewriting that provides in-built support for alpha-equivalence following the nominal approach. Finally, we will discuss notions of nominal unification and rewriting modulo AC operators and briefly overview typed versions of nominal languages.

References


