Available online at www.sciencedirect.com

ScienceDirect

Electronic Notes in Theoretical Computer Science 323 (2016) 57–74

Electronic Notes in Theoretical Computer Science

www.elsevier.com/locate/entcs

Completeness in PVS of a Nominal Unification Algorithm

Mauricio Ayala-Rincón [a,2] Maribel Fernández [b,3]
Ana Cristina Rocha-Oliveira[a,1]

[a] *Departamentos de Matemática e Ciência da Computação*
Universidade de Brasília
Brasília D.F., Brasil

[b] *Department of Informatics*
King's College London
London, UK

Abstract

Nominal systems are an alternative approach for the treatment of variables in computational systems. In the nominal approach variable bindings are represented using techniques that are close to first-order logical techniques, instead of using a higher-order metalanguage. Functional nominal computation can be modelled through *nominal rewriting*, in which α-equivalence, nominal matching and nominal unification play an important role. Nominal unification was initially studied by Urban, Pitts and Gabbay and then formalised by Urban in the proof assistant Isabelle/HOL and by Kumar and Norrish in HOL4. In this work, we present a new specification of nominal unification in the language of PVS and a formalisation of its completeness. This formalisation is based on a natural notion of nominal α-equivalence, avoiding in this way the use of the intermediate auxiliary weak α-relation considered in previous formalisations. Also, in our specification, instead of applying simplification rules to unification and freshness constraints, we recursively build solutions for the original problem through a straightforward functional specification, obtaining a formalisation that is closer to algorithmic implementations. This is possible by the independence of freshness contexts guaranteed by a series of technical lemmas.

Keywords: Nominal terms, binders, α-equivalence, nominal unification, PVS.

1 Introduction

When one introduces variable binders in a language, one thing to be considered immediately is α-equivalence. For instance, it must be possible to derive the equivalence between the formulas $\exists x : x > 1$ and $\exists y : y > 1$, despite the syntactical differences. Nominal theories treat binders in a way that is closer to informal practice, using variable names and freshness constraints instead of using indices as in

[1] Email: anacrismarie@gmail.com. Author supported by a Ph.D. scholarship from CAPES Brazil.

[2] Email: ayala@unb.br. Work partially supported by grant CNPq UNIVERSAL 476952/2013-1.

[3] Email: maribel.fernandez@kcl.ac.uk. Work partially supported by grant CsF PVE CAPES 146/2012.

http://dx.doi.org/10.1016/j.entcs.2016.06.005
1571-0661/

explicit substitutions *à la de Bruijn*. In nominal syntax, there are two kinds of variables: atoms, representing object-level variables, and meta-variables, or simply variables. Atoms can be abstracted but not substituted, whereas variables cannot be abstracted but can be substituted. The notion of substitution is first-order in the sense that it allows capture, but freshness constraints are taken into account. Notions such as rewriting (cf. [9]) and unification (cf. [18]) can be directly defined, without having to rely on involved notions such as β-reduction, as in the higher-order and explicit substitutions approaches (cf. [12,8,3]).

Nominal unification problems can be solved (modulo α-equivalence) with first-order substitutions that act over meta-variables, i.e., simply filling the holes marked with meta-variables $(X, Y, Z, \dots)$ and allowing capture of variable names $(a, b, c, i, k, \dots)$. This can be illustrated by the expressions

$$\sum_{k=0}^{7}\sum_{i=0}^{5}(i - X)^i \text{ and } \sum_{i=0}^{7}\sum_{k=0}^{5}(X - Y)^k,$$

which admit a most general unifier according to the algorithm in [18], with solution $[X \mapsto k][Y \mapsto i]$. Note that i and k are captured, because these names are bound or abstracted by the sum operator. In a higher-order unification approach, this solution would not be accepted because bound variable capture is forbidden.

On the other hand, the unification problem with the expressions

$$\sum_{i=0}^{5}(i - X)^i \text{ and } \sum_{k=0}^{5}(X - Y)^k$$

has no solution in the nominal setting. One could argue that a solution could be obtained instantiating $[X \mapsto i][Y \mapsto i]$ and renaming k as i. But this is not possible since i should be a "fresh" name in the scope of the second sum in order to proceed with this renaming, and the chosen substitution contradicts this condition. In other words, the meta-variable X should be instantiated uniformly throughout the problem. We can specify that a name is fresh for a term by writing a freshness constraint, for example, $i\#t$ states that the name i is fresh in the term t. In general, if two nominal terms are unifiable, the unifier is a pair consisting of a substitution and a set of freshness constraints.

Translations between nominal unification problems and higher-order pattern unification problems are given in [6,15].

Contribution

In this paper, we present a functional specification of a new nominal unification algorithm and formalise its correctness and completeness in the language of the higher-order proof assistant Prototype Verification System (PVS) [17]. PVS was chosen because it has a large library about term rewriting systems ([11]) and our `nominal unification` theory extends this background about rewriting.