Available online at www.sciencedirect.com

ScienceDirect

Electronic Notes in Theoretical Computer Science 312 (2015) 19–50

Electronic Notes in Theoretical Computer Science

www.elsevier.com/locate/entcs

A Dependent Type Theory with Abstractable Names

Andrew M. Pitts[a,1,2], Justus Matthiesen [a,3,4] and Jasper Derikx[b,5]

[a] *Computer Laboratory, University of Cambridge, Cambridge CB3 0FD, UK*

[b] *Radboud University, 6500 GL Nijmegen, Netherlands*

Abstract

This paper describes a version of Martin-Löf's dependent type theory extended with names and constructs for freshness and name-abstraction derived from the theory of nominal sets. We aim for a type theory for computing and proving (via a Curry-Howard correspondence) with syntactic structures which captures familiar, but informal, 'nameful' practices when dealing with binders.

Keywords: binding, dependent types, names, nominal sets

1 Introduction

We aim to develop a constructive version of nominal logic [15] as a dependent type theory. From a programming point of view we would like to combine Agda/Coq style theorem-proving (particularly inductively defined indexed families of types and dependent pattern-matching) with FreshML [21] style meta-programming for syntax with binding operations. Achieving these aims requires a constructive treatment of the nominal sets notion of *freshness* [16, Chapter 3]. Here we give one such treatment as an extension of Martin-Löf type theory.

The functional programming language FreshML is impure: it ensures freshness of names via generativity and (hence) avoids checking that a locally scoped name

[1] Partially supported by the UK EPSRC program grant EP/K008528/1, *Rigorous Engineering for Mainstream Systems* (REMS).

[2] Email: andrew.pitts@cl.cam.ac.uk

[3] Supported by the UK EPSRC leadership fellowship (Peter Sewell) grant EP/H005633/1, *Semantic Foundations for Real-World Systems*.

[4] Email: justus.matthiesen@cl.cam.ac.uk

[5] Email: jasperderikx@gmail.com

http://dx.doi.org/10.1016/j.entcs.2015.04.003

does not occur in the support of the meaning of the expression in which it is used. The original version of the language, ‘FreshML 2000’ [17], attempted to carry out such checks by inferring freshness information as part of the type system, but was found to be too restrictive in the context of a Turing-powerful language – the main difficulty being how to decide whether a name n is fresh for a (higher-order) function f, written $n \# f$. Within nominal sets [16], the definition of the freshness relation involves quantification over finite sets of names: $n \# f$ means that there exists a finite set of names supporting f that does not contain the name n. In practice, one often relies upon the fact that this relation is invariant under permuting names and uses the following sound method that reflects on a concrete, meta-theoretic version of freshness, *viz.* non-occurrence:

> To prove $n \# f$, pick a name n' that does not occur in the current context (that is, one that is meta-theoretically fresh) and prove $(n\ \ n') \cdot f = f$, which in the presence of function extensionality, is equivalent to showing $(\forall x)\ (n\ \ n') \cdot (f\,x) = f((n\ \ n') \cdot x)$. (As usual, $(n\ \ n') \cdot x$ denotes the result of transposing names n and n' in an element x of a nominal set.) Since $n' \# f$ holds by choice of n', applying the permutation $(n\ \ n')$ that swaps n and n' we get $n = (n\ \ n') \cdot n' \# (n\ \ n') \cdot f = f$, as required.

This proof principle was adopted by nominal algebra [9]/nominal equational logic [5] and emphasised particularly in Clouston’s thesis [4] and the recent work of Crole and Nebel [7], which both make freshness assertions

$$\Gamma \vdash n \,\#\, t : T$$

equivalent to equality judgements of the form

$$\Gamma[n' : N] \vdash (n\ \ n') \bullet t = t : T \tag{1}$$

where $(n\ \ n') \bullet \_$ is the (object-level) name-swapping operation, the context Γ contains hypotheses about freshness of names for free variables and $\Gamma[n : N]$[6] adds to Γ an extra freshness hypotheses for a (meta-theoretically) new name n of some sort N. Equality jugements, such as (1), will be axiomatized by the type theory introduced in Sect. 2.

We call this delegation of freshness to definitional equality *definitional freshness.* It means that equality judgements get intertwined with typing judgements in an extra way from what already happens in dependently typed systems. The advantage of this approach is that we can give ‘pure’ versions of locally scoped names and concretion of name-abstractions with a semantics just using nominal sets, rather than, for example, nominal restriction sets [16, section 9.1]. The next section describes such a dependent type theory with abstractable names. Since it is an extension of Martin-Löf’s Type Theory with many of the features of FreshML, we call it FreshMLTT. Section 3 describes the intended model of FreshMLTT; we

[6] Instead of using the ‘flattened’ contexts $\{\boldsymbol{n}_1 \# x_1 : T_1, \boldsymbol{n}_2 \# x_2 : T_2, \ldots\}$ from [24,9,5,4,7], here we will use ‘bunched’ ones, as in [19,18,3], because they fit better with the ‘telescopic’ nature of contexts in dependent type theory.