Kripke Resource Models of a Dependently-Typed, Bunched λ-Calculus (Extended Abstract)

Samin Ishtiaq and David J. Pym

Department of Computer Science
Queen Mary & Westfield College
University of London
London E1 4NS
{si,pym}@dcs.qmw.ac.uk

Abstract. The λΛ-calculus is a dependent type theory with both linear and intuitionistic dependent function spaces. It can be seen to arise in two ways. Firstly, in logical frameworks, where it is the language of the RLF logical framework and can uniformly represent linear and other relevant logics. Secondly, it is a presentation of the proof-objects of BI, the logic of bunched implications. BI is a logic which directly combines linear and intuitionistic implication and, in its predicate version, has both linear and intuitionistic quantifiers. The λΛ-calculus is the dependent type theory which generalizes both implications and quantifiers. In this paper, we describe the categorical semantics of the λΛ-calculus. This is given by Kripke resource models, which are monoid-indexed sets of functorial Kripke models. We describe a class of concrete, set-theoretic models. The models are given by the category of families of sets, parametrized over a small monoidal category, in which the intuitionistic dependent function space is described in the established way, but the linear dependent function space is described using Day’s tensor product.

1 Introduction

A long-standing problem has been to combine type-dependency and linearity. In [13], we introduced the λΛ-calculus, a first-order dependent type theory with a full linear dependent function space, as well as the usual intuitionistic dependent function space. The λΛ-calculus can be seen to arise in two ways. Firstly, in logical frameworks [9,18], in which it provides a language that is a suitable basis for a framework capable of properly representing linear and other relevant logics. Secondly, from the logic of bunched implications, BI [15,19], in which the antecedents of sequents are structured not as lists but as bunches, which have two combining operations, “;”, which admits Weakening and Contraction, and “,”, which does not. The λΛ-calculus stands in propositions-as-types correspondence with a fragment of BI [14,12].

The purpose of this paper is to present the categorical semantics of the λΛ-calculus. This is given by Kripke resource models, which are monoid-indexed sets of functorial Kripke models. The indexing element can be seen as the resource able to realize the structure it indexes. We work with indexed categories rather than, for example, with Cartmell’s contextual categories [4], as the indexed approach allows a better separation of the evident conceptual issues.
Kripke resource models generalize, as we might expect, the functorial Kripke models of the \(\lambda\Pi\)-calculus \([18]\). These consist of a functor \(\mathcal{F}:[\mathcal{V}, [C^\text{op}, \mathsf{Cat}]]\), where \(\mathcal{V}\) is a Kripke world structure, \(\mathcal{C}\) is a category with a \((\times, 1)\) cartesian monoidal structure on it and \([C^\text{op}, \mathsf{Cat}]\) is a strict indexed category. The intuitionistic dependent function space \(\Pi\) is modelled as right adjoint to the weakening functor \(p^*;\mathcal{F}(\mathcal{W})(D) \rightarrow \mathcal{F}(\mathcal{W})(D \times A)\).

In the \(\lambda\Lambda\)-calculus, we have two kinds of context extension operators, so we require \(\mathcal{C}\) to have two kinds of monoidal structure on it, \((\otimes, I)\) and \((\times, 1)\). The intuitionistic dependent function space \(\Pi\) can be modelled, as usual, using the right adjoint to projection. However, there is no similar projection functor corresponding to \(\Lambda\). For this, we must require the existence of the natural isomorphism \(\text{Hom}_{\mathcal{F}_\Lambda}(\mathcal{W}(\Delta(A))(1, B)) \cong \text{Hom}_{\mathcal{F}_\Lambda}(\mathcal{W}(D))(1, A B(0))\), where \(D \otimes A\) is defined in the \(r + r^2\)-indexed model. This is sufficient to define the function space.

While the \(\lambda\Lambda\)-calculus has familiar soundness and, via a term model, completeness theorems, it is important to ask if there is a natural class of models. For the \(\lambda\Pi\)-calculus, for instance, the most intuitive concrete model is that of families of sets, \(\mathsf{Fam}\). This can be viewed as an indexed category \(\mathsf{Fam}:[Ctx^{op}, \mathsf{Cat}]\). The base, \(Ctx\), is a small set-theoretic category whose objects are sets and morphisms are set-theoretic functions. For each \(D \in \text{obj}(\mathsf{Fam})\), \(\mathsf{Fam}(D) = \{ y \in B(x) \mid x \in D \}\). The fibre is just a discrete category whose objects are the elements of \(B(x)\). If \(f \in \mathsf{Fam}(C, D)\), then \(\mathsf{Fam}(f)\) just re-indexes the set over \(D\) to one over \(C\). As there is little structure required in the fibre, the description of families of sets can also be given sheaf-theoretically, as \(\mathsf{Fam}:[Ctx^{op}, \mathsf{Set}]\), each \(\mathsf{Fam}(D)\) being considered as a discrete category. Using Day’s construction \([7]\), we obtain a corresponding class of set-theoretic models, parametrized on a small monoidal category, for the \(\lambda\Lambda\)-calculus. That is, we describe a families of sets model in \(\mathsf{BIFam}:[C, [Ctx^{op}, \mathsf{Set}]]\), where \(C\) is some small monoidal category.

## 2 The \(\lambda\Lambda\)-Calculus

A detailed account of the \(\lambda\Lambda\)-calculus and the RLF logical framework is given in \([13]\). The work there develops ideas originally presented in \([17]\).

The \(\lambda\Lambda\)-calculus is a first-order dependent type theory with both linear and intuitionistic function types. The calculus is used for deriving typing judgements. There are three entities in the \(\lambda\Lambda\)-calculus: objects, types and families of types, and kinds. Objects (denoted by \(M, N\)) are classified by types. Families of types (denoted by \(A, B\)) may be thought of as functions which map objects to types. Kinds (denoted by \(K\)) classify families. In particular, there is a kind \(Type\) which classifies the types. We will use \(U, V\) to denote any of the entities. The abstract syntax of the \(\lambda\Lambda\)-calculus is given by the following grammar:

\[
\begin{align*}
K &::= \text{Type} \mid A x : A. K \mid A x ! A. K \\
A &::= a \mid A x : A. B \mid A x ! A. B \mid \lambda x : A. B \mid \lambda x! A. B \mid A M \mid A k B \\
M &::= c \mid x \mid \lambda x : A. M \mid \lambda x! A. M \mid M N \mid \langle M, N \rangle \mid \pi_0(M) \mid \pi_1(M)
\end{align*}
\]

We write \(x \in A\) to range over both linear \((x : A)\) and intuitionistic \((x! A)\) variable declarations. The \(\lambda\) and \(\Lambda\) bind the variable \(x\). The object \(\lambda x : A. M\) is an inhabitant of the linear dependent function type \(Ax : A. B\). The object \(\lambda x! A. M\) is an inhabitant of the