Towards Normalization by Evaluation for the \(\beta\eta\)-Calculus of Constructions

Andreas Abel

Project PI.R2, INRIA Rocquencourt and PPS, Paris
andre.as.abel@ifi.lmu.de

Abstract. We consider the Calculus of Constructions with typed beta-eta equality and an algorithm which computes long normal forms. The normalization algorithm evaluates terms into a semantic domain, and reifies the values back to terms in normal form. To show termination, we interpret types as partial equivalence relations between values and type constructors as operators on PERs. This models also yields consistency of the beta-eta-Calculus of Constructions. The model construction can be carried out directly in impredicative type theory, enabling a formalization in Coq.

1 Introduction

The proof assistant Coq \cite{INR08} based on intensional type theory is used for large verification projects in mathematics \cite{Gon04} and computer science \cite{Ler06}. However, to this day no complete meta theory of its logical core, the Calculus of Inductive Constructions (CIC) exists. The CIC is a dependent type theory with at least one impredicative base universe (\(\text{Set}\) or \(\text{Prop}\) or both) and an infinite cumulative hierarchy of predicative universes (\(\text{Type}_i\)) above this base. Inductive types with large (aka strong) eliminations exist at every level. The CIC is formulated with untyped equality, leading to complications in model constructions \cite{MW03} and in the treatment of \(\eta\)-equality. As \(\eta\)-reduction, the subject reduction property requires contravariant subtyping, which is especially hard to model (I am only aware of Miquel’s coherence space model \cite{Miq00}). And it cannot be formulated as \(\eta\)-expansion in an untyped setting. The lack of \(\eta\)-equality in Coq is an annoyance both for its implementers and its users.

Recently, formulations of CIC with typed equality, aka judgemental equality, have been considered since they admit simple set-theoretical models \cite{Bar09}. Judgemental equality also integrates \(\eta\)-equality nicely. On the downside, injectivity of the function space constructor \(\Pi\), crucial for the implementation of type checking, is notoriously difficult to establish. Goguen \cite{Gog94} has obtained injectivity of \(\Pi\) in the Extended Calculus of Constructions via his Typed Operational Semantics, a typed Kripke term model with standardizing reduction. In predicative Martin-Löf Type Theory, it is the byproduct of a PER model construction which also yields Normalization by Evaluation (NbE) \cite{ACD07}.

In this article, we investigate NbE for the Calculus of Constructions (CoC), a fragment of the CIC with just one impredicative and one predicative universe, with typed \(\beta\eta\)-equality. By constructing a PER model, we obtain termination and completeness for
Towards Normalization by Evaluation for the $\beta\eta$-Calculus of Constructions

NbE, the latter meaning that all judgmentally equal terms normalize to the same expression. As a consequence of the model, we obtain logical consistency of the $\beta\eta$-CoC.

The missing property of soundness of NbE, meaning that each term is judgmentally equal to its computed normal form, is implied by injectivity of $\Pi$ and vice versa. Decidability of typing also hinges on injectivity. This leaves two options to complete this work and obtain a sound and complete type checker for the CoC with $\eta$: Prove soundness of NbE by Kripke logical relations between syntax and semantics as in [ACP09], or obtain injectivity by syntactical means. Adams [Ada06] obtained injectivity for functional pure type systems with judgemental $\beta$-equality; his proof might extend to $\beta\eta$.

Overview. This article is organized as follows: In Section 2 we introduce CoC with typed equality and explicit substitutions. In Section 3 we define normalization by evaluation for CoC using partial applicative structures, and we specify a type inference algorithm. In Section 4 we recapitulate a simple method how to classify CoC expressions into terms, types, and kinds, a device which helps us to bootstrap the PER model construction in Section 5. Section 6 proves the rules of CoC sound wrt. our model, and as a corollary we obtain termination and completeness of NbE and consistency of CoC. Loose ends are listed in the conclusions (Section 7).

2 Syntax

We present the Calculus of Constructions (CoC) as a pure type system (PTS) with annotated $\lambda$-abstraction, typed equality and explicit substitutions.

1. Annotated $\lambda$-abstraction (as in $\lambda MN$) enables us to compute the type of a term from the type of its free variables and its semantics from the semantics of its free variables (see Section 6). Thus, a term makes already sense in a context alone, it does not need an ascribed type.

2. Typed equality is the natural choice in the presence of $\eta$ since untyped $\eta$-reduction is badly behaved in set-theoretical models and type-theoretical models without subtyping — this includes the semantics we are constructing. (Untyped $\eta$-expansion cannot be defined.)

3. Lambda calculi with explicit substitutions have more models than lambda calculi with substitution implemented as an operation. In particular, the model of closures in weak head normal form we will use in Section 3. Recent meta theoretic studies involving explicit substitutions include [Dan07, Cha09, Gra09, ACP09]. In the presence of explicit substitutions, variables are most naturally represented as de Bruijn indices [ACCL91].

2.1 Expressions and Typing

The CoC is a dependently typed lambda calculus with expressions on three levels: terms $t, u$, the data structures and programs of the language; the types $T, U$ of terms, generalized to a lambda-calculus of type constructors $\mathbb{1}$ and the kinds $\kappa, \iota$, the types of types.

---

$\mathbb{1}$ E.g., List is a type constructor which produces a type of homogeneous lists List $T$ for each element type $T$. 