Type Normalization in R$\omega\mu$

ALEX HUBERS, The University of Iowa, USA

1 INTRODUCTION

We describe the normalization-by-evaluation (NBE) of types in R$\omega\mu$. Types are normalized modulo β- and η-equivalence—that is, to $\beta\eta$-long forms. Because the type system of R$\omega\mu$ is a strict extension of System Fω, type level computation for arrow kinds is isomorphic to reduction of arrow types in the STLC. Novel to this report are the reductions of Π, Σ, and label bound terms.

2 SYNTAX OF KINDS

Our formalization of R$\omega\mu$ types is *intrinsic*, meaning we define the syntax of *typing* and *kinding judgments*, foregoing any description of untyped syntax. The syntax of types is indexed by kinding environments and kinds, defined below.

```
data Kind : Set where
  ★     : Kind
  L     : Kind
  _‘→_ : Kind → Kind → Kind
  R[_] : Kind → Kind

infixr 5 _‘→_
```

The kind system of R$\omega\mu$ defines $\star$ as the type of types; L as the type of labels; $(\to)$ as the type of type operators; and $R[\kappa]$ as the type of *rows* containing types at kind κ. As shorthand, we write $R^n[\kappa]$ to denote n repeated applications of R to the type κ–e.g., $R^3[\kappa]$ is shorthand for $R[R[R[\kappa]]]$.

The syntax of kinding environments is given below. Kinding environments are isomorphic to lists of kinds.

```
data KEnv : Set where
  ε : KEnv
  _,,_ : KEnv → Kind → KEnv
```

Let the metavariables Δ and κ range over kinding environments and kinds, respectively. Correspondingly, we define *generalized variables* in Agda at these names. The syntax of intrinsically well-scoped De-Bruijn-indexed variables is given below.

```
private
  variable
    Δ Δ₁ Δ₂ Δ₃ : KEnv
    κ κ₁ κ₂ : Kind

data KVar : KEnv → Kind → Set where
  Z : KVar (Δ ,, κ) κ
  S : KVar Δ κ₁ → KVar (Δ ,, κ₂) κ₁
```

Author's address: Alex Hubers, Department of Computer Science, The University of Iowa, 14 MacLean Hall, Iowa City, Iowa, USA, alexander-hubers@uiowa.edu.

The kind variable x is indexed by kinding environment Δ and kind κ to specify that x has kind κ in kinding environment Δ.

3 SYNTAX OF TYPES

R$\omega\mu$ is a qualified type system with predicates of the form $\rho_1 \lesssim \rho_2$ and $\rho_1 \cdot \rho_2 \sim \rho_3$ for row-kinded types ρ_1, ρ_2, and ρ_3. Because predicates occur in types and types occur in predicates, the syntax of well-kinded types and well-kinded predicates are mutually recursive. The syntax for each is given below; we describe (in this order) the syntactic components belonging to the STLC, System Fω, qualified types, and system Rω.

$\text{data Pred } (\Delta : \text{KEnv}) : \text{Kind} \to \text{Set}$

$\text{data Type } \Delta : \text{Kind} \to \text{Set}$

$\text{data Type } \Delta \text{ where}$

$\quad \text{`} :$

$\qquad (\alpha : \text{KVar } \Delta\ \kappa) \to$

$\qquad \text{Type } \Delta\ \kappa$

$\quad \text{`}\lambda :$

$\qquad (\tau : \text{Type } (\Delta \text{ ,, } \kappa_1)\ \kappa_2) \to$

$\qquad \text{Type } \Delta\ (\kappa_1 \text{ `}\!\to \kappa_2)$

$\quad \_\cdot\_ :$

$\qquad (\tau_1 : \text{Type } \Delta\ (\kappa_1 \text{ `}\!\to \kappa_2)) \to$

$\qquad (\tau_2 : \text{Type } \Delta\ \kappa_1) \to$

$\qquad \text{Type } \Delta\ \kappa_2$

$\quad \_\text{`}\!\to\_ :$

$\qquad\quad (\tau_1 : \text{Type } \Delta\ \star) \to$

$\qquad\quad (\tau_2 : \text{Type } \Delta\ \star) \to$

$\qquad\quad \text{Type } \Delta\ \star$

Description, description, blah.

$\text{data Pred } \Delta \text{ where}$