The Recursive Index Calculus and Its Translation From Rω

AH & JGM

November 13, 2023

1 Example translations of Rω terms and types

Record selection. In Rω,

$$\forall \rho : \mathsf{R}^\star,\ \ell : \mathsf{L},\ \tau : \star.\{\ell \triangleright \tau\} \lesssim \rho \Rightarrow \lfloor \ell \rfloor \to \Pi\rho \to \tau$$

translates to

$$\Pi(\rho : \mathrm{Row}\,\star).\Pi(\ell : \top).\Pi(\tau : \star).\{\ell \triangleright \tau\} \lesssim \rho.\Pi(\_ : \top).\Pi(i : \mathrm{Ix}\,\rho.1).\ \rho.2\ i$$

where

$$\begin{aligned} \mathrm{Row}\kappa :&= \Sigma(n : \mathrm{Nat}).\Pi(i : \mathrm{Ix}\,n).\kappa \\ \{\ell \triangleright \tau\} \lesssim \rho &= \Pi(i : \mathrm{Ix}\,\{\ell \triangleright \tau\}.1).\Sigma(j : \mathrm{Ix}\,\rho.1).\{\ell \triangleright \tau\}.1\ i \equiv \rho.2\ j \\ \{\ell \triangleright \tau\} &= (\mathrm{Suc}\,\mathrm{Zero} : \mathrm{Nat}, \lambda(i : \mathrm{Ix}\,(\mathrm{Suc}\,\mathrm{Zero})).\tau) \end{aligned}$$

Putting this all together:

$$
\begin{array}{l}
\Pi(\rho : (\Sigma(n : \mathrm{Nat}).\Pi(i : \mathrm{Ix}\, n).\star)).\\
\Pi(\ell : \top).\\
\Pi(\tau : \star).\\
\Pi(P :\\
\quad \Pi(i : \mathrm{Ix}\,(\mathrm{Suc}\,\mathrm{Zero} : \mathrm{Nat}, \lambda(i : \mathrm{Ix}\,(\mathrm{Suc}\,\mathrm{Zero})).\tau).1).\\
\quad \Sigma(j : \mathrm{Ix}\,\rho.1).\\
\quad (\mathrm{Suc}\,\mathrm{Zero} : \mathrm{Nat}, \lambda(i : \mathrm{Ix}\,(\mathrm{Suc}\,\mathrm{Zero})).\tau).2\ i \equiv \rho.2\ j)\\
\Pi(\_ : \top).\\
\Pi(i : \mathrm{Ix}\,\rho.1).\ \rho.2\ i
\end{array}
$$

which should normalize to

$$
\begin{array}{l}
\Pi(\rho : (\Sigma(n : \mathrm{Nat}).\Pi(i : \mathrm{Ix}\, n).\star)).\\
\Pi(\ell : \top).\\
\Pi(\tau : \star).\\
\Pi(P :\\
\quad \Pi(i : \mathrm{Ix}\,1).\\
\quad \Sigma(j : \mathrm{Ix}\,\rho.1).\\
\quad \tau \equiv \rho.2\ j).\\
\Pi(\_ : \top).\\
\Pi(i : \mathrm{Ix}\,\rho.1).\ \rho.2\ i
\end{array}
$$

2 Design considerations of the formal index calculus

2.1 The necessity of Set-in-Set

Denote the translation of Rω type τ to the index calculus as $[\![\tau]\!]$. The big stars of systems *R*&c. are record and variants. Consider how we might translate the former. We first establish the translation of kinds so that we may assert that translated types are in the meaning of their translated kinds. (This is a sensible verification.) The following seems reasonable.

$$
\begin{aligned}
[\![\star]\!]_\kappa &= \star \\
[\![\mathsf{L}]\!]_\kappa &= \top \\
[\![\kappa_1 \to \kappa_2]\!]_\kappa &= [\![\kappa_1]\!] \to [\![\kappa_2]\!] \\
[\![\mathsf{R}^\kappa]\!]_\kappa &= \exists i : Nat.(\mathrm{Fin}\, i \to [\![\kappa]\!])
\end{aligned}
$$

(Nevermind, for now, the complexity of existentially quantifying Nats within kinds.) Recall that records (in Rω) live in kind $\star$ but, as shown in our Agda denotation, should translate to functions which map finite indices to types. Below is a sensible type and translation for such an idea. (As $[\![\rho]\!]$ is a dependent product, let its first and second projections be defined as usual.)

$$
[\![\Pi\, \rho]\!] = \forall i : \mathrm{Nat}.\mathrm{Fin}\, [\![\rho]\!].1 \to [\![\rho]\!].2\, i
$$

Yet, as $\Pi\, \rho : \star$ (in Rω), we must conclude that the type $\forall i : \mathrm{Nat}.\mathrm{Fin}\, [\![\rho]\!].1 \to [\![\rho]\!].2\, i$ has kind $\star$ in μIx. I don't see a way around this. Further, it implies to me that we would like to flatten types and kinds and let types quantify over types. That is to say, we want a two-level stratification of terms and types only, omitting kinds. The next two subsections argue for this change.

2.2 Flattening kinds and types

Revisit now the translation of rows.

$$
[\![\mathsf{R}^\kappa]\!]_\kappa = \exists i : \mathrm{Nat}.(\mathrm{Fin}\, i \to [\![\kappa]\!])
$$

The existential quantification of $i : \mathrm{Nat}$ is ambiguous. We seem to need to either (i) permit the dependent, existential quantification over *type* Nat, or (ii) lift Nat indices to kinds and permit the existential quantification of indices in kinds. In the latter case, we would further need to represent the *application* of Fin to *index variable* i—and also, of course, need to represent existential quantification in kinds (and represent Fin!). This becomes cumbersome quickly, enough to beg the question: why distinguish types from kinds at all?

2.2.1 The inadequacy of ?

Described above by (i) is the approach advocated by **?**. **?** only permits types to be indexed by a declared set of index objects, each with index sort—for example, vectors are indexed by index objects with index sort Nat. The syntax of (some simple) indices is given below.

$$\begin{array}{lll} \text{Index Sorts} & \gamma & ::= \text{Nat} \mid \top \mid ... \\ \text{Index Objects} & \iota & ::= i \mid () \mid ... \end{array}$$

Rather than types we have *families of types*, each indexed by *index objects*. For example, an *intlist* might be a family type family indexed by Nat, representing its length. Consider the typing of vector concatenation.

$$\texttt{concat} : \Pi m : \text{Nat}.\Pi n : \text{Nat}.\mathit{intlist}\ m \to \mathit{intlist}\ n \to \mathit{intlist}\ (m + n)$$

This machinery is not sufficient to type many Rω terms. To illustrate, presume a base set of index sorts γ and index objects ι to be as defined below. (For convenience, let the index objects of ι inhabit both Nat and $\text{Fin}\,\iota$.)

$$\begin{array}{lll} \gamma & ::= & \text{Nat} \mid \text{Fin}\,\iota \\ \iota & ::= & i \mid 0 \mid \text{Suc}\,\iota \end{array}$$

Now consider an example translation of record concatenation from Rω to μIx. In Rω, we have:

$$\texttt{concat} : \forall(z_1\,z_2\,z_3 : \mathsf{R}^\star).z_1 \cdot z_2 \sim z_3 \Rightarrow \Pi\,z_1 \to \Pi\,z_2 \to \Pi\,z_3$$

This (hypothetically) translates to (the sketch I have in my head of) the μIx type:

$$
\begin{aligned}
\texttt{concat} : & \forall^i mnl : Nat. \\
& \forall(z_1 : \mathrm{Fin}\, m \to \star)(z_2 : \mathrm{Fin}\, n \to \star)(z_3 : \mathrm{Fin}\, l \to \star). \\
& [\![z_1 \cdot z_2 \; z_3]\!] \to \\
& (\forall(i : \mathrm{Fin}\, m) \to z_1\, i) \\
& (\forall(i : \mathrm{Fin}\, n) \to z_2\, i) \\
& \forall(i : \mathrm{Fin}\, l).z_3\, i
\end{aligned}
$$

Let $\forall^i$ denote the quantification of indices in types and $[\![z_1 \cdot z_2 \sim z_3]\!]$ denote the (yet-undefined) translation of Rω predicates to μIx types. The glaring incompatibility of **?** is that Rω is higher-order. So, it is not clear what the quantification of z_1 over $(\mathrm{Fin}\, m \to \star)$ means. Is $(\mathrm{Fin}\, m \to \star)$ a kind? **?** permits only the quantification over indices, and the use of those indices in types. The authors do not permit $F\omega$ (or just System F)-style quantification over type variables. And it is not clear how to lift their calculus to higher-order; I suspect, also, nontrivial.

A lot of our problems go away when committing to an impredicative, dependent, term-and-type-stratified type theory. So, this is what I suggest we do. Firstly, the higher-order nature of $F\omega$ is given for free, as there are no more kinds and thus all type-level quantification is over types. W.r.t. mechanizational ease, we substantially reduce the overlap in ASTs.

What we are left with is more less MLTT with (built-in) finite naturals. Call it $\lambda^{\Pi,\Sigma}$ for now.

2.3 Consistency in impredicative MLTT

Martin L of Type Theory is well known to be inconsistent without a predicative universe hierarchy **???**. The calculus of constructions, however, has two sorts—P, that of (impredicative) predicates, and T the type of P. This turns out to be consistent. It should be known why, but I do not know now.