9/14: Up to Speed on Stirling & Spitters

Jay Kruer

September 13, 2021

Contents

1 Reminders of some notions from my first chapter

2 Two Yoneda embeddings and normalization by evaluation

2.1 Normalization by evaluation

Normalization by evaluation is a technique used to demonstrate normalization– the property that all terms have a normal form–for some lambda calculus. Andreas Abel renders the technique very clearly[1]: Normalization is the process of bringing an open term (with unknowns) to a special kind of fixed point. A similar notion is evaluation, the process of bringing a *closure*,

[1] `https://www.cse.chalmers.se/~abela/talkEAFIT2017.pdf`

namely an open term paired with a substitution closing it, to a canonical form comprised entirely of constructors.

Normalization by evaluation borrows an exist evaluator for some sufficiently expressive host language and uses it to normalize an open expression in the guest language we're interested in. The basic plot outline is this, each step paired with the relevant code from an instance[2] of normalization with the host being Standard ML and the guest being the simply typed lambda calculus:

1. Create in the host language an internal representation of the guest language syntax and type structure:

```
datatype ty = Basic of string
            | Arrow of ty * ty
            | Prod of ty * ty
```

2.

This is accomplished through a mutually recursive pair of operations called reification and reflection.

2.2 Reification, reflection

2.3 Just work from wikipedia for this part

2.4 The readback used in StirSpit

Note that I haven't yet seen exactly the readback operation (in terms of presheaves) planned yet. Their presentation is abysmally bad on this point actually... they delay explaining/defining an object of deep conceptual and technical import to the result until like 10 pages in. Maybe they suppose that the reader already has experience with normalization by evaluation, which I guess is kind of fair, but this was personally my first exposure to it in earnest.

3 A pickle: too much quotienting

The definition of the syntactic category (category of contexts and substitutions) given in the chapter I wrote a few weeks ago has for its equations

[2] `https://en.wikipedia.org/wiki/Normalisation_by_evaluation`

governing equality of morphisms those given by the "substitution lemma." It turns out that this identifies far too many terms for our uses. In particular, terms which are related by the various beta rules are identified, meaning that a normal form (a term for which no further beta reduction can be performed) is identified with its (manifestly not normal) beta-predecessor. The upshot is that we can't isolate the normal forms as a class of terms, which totally bungles our whole project of investigating which terms (all of them) of the simply typed lambda calculus have normal forms.

4 Unpickling ourselves: the category of renamings

5 The relative hom functor

Stirling & Spitters follow Fiore in defining the "relative hom functor", which they suggestively call $\mathfrak{Tm} : \mathrm{Cl}_\Sigma \to \mathrm{Ren}_\Sigma{}^{\mathrm{Set}}$. The suggestion hinted at by the name, that this functor defines a presheaf of open terms, turns out to be a (small?) lie. Let's look at what it actually does. $\mathfrak{Tm}$ is defined by adjusting the hom functor (i.e, the Yoneda embedding) by precomposition with the inclusion of the category of renamings into the category of (contexts and) substitutions. In particular, StirSpit define $\mathfrak{Tm}(\Delta) = \mathrm{Cl}_\Sigma[i(-), \Delta]$. In plain terms, $\mathfrak{Tm}(\Delta)$ takes a context in the category of renamings to the *substitutions on terms* **out of** Δ (recalling that the action of the category of contexts on its clones is contravariant). This can be (very loosely) construed as a presheaf of open terms. For a (renaming) context Γ, we have $\mathfrak{Tm}(\Delta)(\Gamma) = \mathrm{Cl}_\Sigma[i(\Gamma), \Delta]$, the context Γ just falls through and we get the substitutions $\gamma^* : \Delta \vdash \tau \to \Gamma \vdash \tau$ for arbitrary τ. In particular, any (possibly) open term $\Delta \vdash t : \tau$ is included in $\mathfrak{Tm}(\Delta, \tau)(\Delta)$ as the single substitution $[t/x]$. The reason I regard as misleading the suggestion that the relative hom functor defines a presheaf of open terms is that the morphisms in the syntactic category aren't just single substitutions, but also single omissions $\hat{x}$ and all the compositions of these two classes of maps.