GUARDED RECURSIVE TYPE THEORY VIA SIZED TYPES

Abstract. Lorem ipsum dolor sit amet, consectetur adipiscing elit. Praesent convallis orci arcu, eu mollis dolor. Aliquam eleifend suscipit lacinia. Maecenas quam mi, porta ut lacinia sed, convallis ac dui. Lorem ipsum dolor sit amet, consectetur adipiscing elit. Suspendisse potenti.

1. Introduction

[?]

2. Preliminaries

2.1. Sized Types.

2.2. Guarded Recursive Type Theory.

3. Presheaves

A presheaf is a functor from the category of sizes to Set.

```
record PSh : Set₁ where
  field
    Obj : Size → Set
    Mor : (i : Size) (j : Size< (↑ i))
      → Obj i → Obj j
    MorId : {i : Size} {x : Obj i}
      → Mor i i x ≡ x
    MorComp : {i : Size} {j : Size< (↑ i)} {k : Size< (↑ j)}
      → {x : Obj i}
      → Mor i k x ≡ Mor j k (Mor i j x)
```

Every type A in Set defines a constant presheaf whose action on i is given by A, for any size i. »»»> 1e8f49f03d642deca639953d3159a2a0ea1c6beb

Presheaves are the objects of a category whose morphisms are natural transformations. This is a cartesian closed category with finite coproducts. The terminal object is the constant presheaf on the unit type $\top$, i.e. Terminal $=$ Const $\top$. Given two presheaves P and Q, we write Prod P Q for their cartesian product. The action of Prod P Q on a size i is defined as follows:

```
ProdObj : (P Q : PSh) → Size → Set
ProdObj P Q i = PSh.Obj P i × PSh.Obj Q i
```

Coproducts are also defined in a similar pointwise way.

Given two presheaves P and Q, we write Exp P Q for their exponential. The action of Exp P Q on a size i is defined as follows:

```
ExpObj : (P Q : PSh) → Size → Set
ExpObj P Q i =
```

$$\Sigma\ ((j : \mathsf{Size{<}}\ (\uparrow i)) \to \mathsf{PSh.Obj}\ P\ j \to \mathsf{PSh.Obj}\ Q\ j)$$
$$\quad (\lambda\ f \to (j : \mathsf{Size{<}}\ (\uparrow i))\ (k : \mathsf{Size{<}}\ (\uparrow j))\ (x : \mathsf{PSh.Obj}\ P\ j)$$
$$\qquad \to \mathsf{PSh.Mor}\ Q\ j\ k\ (f\ j\ x) \equiv f\ k\ (\mathsf{PSh.Mor}\ P\ j\ k\ x))$$

4. The Model

4.1. Types, Contexts, Terms. A context in set is an element of Set. A context in tot is a preasheaf in PSh.

$$\mathsf{Ctx} : \mathsf{tag} \to \mathsf{Set}_1$$
$$\mathsf{Ctx}\ \mathsf{set} = \mathsf{Set}$$
$$\mathsf{Ctx}\ \mathsf{tot} = \mathsf{PSh}$$

Since we are modeling a simply typed calculus, types are interpreted in the same ways as contexts.

$$\mathsf{Ty} : \mathsf{tag} \to \mathsf{Set}_1$$
$$\mathsf{Ty}\ \mathsf{set} = \mathsf{Set}$$
$$\mathsf{Ty}\ \mathsf{tot} = \mathsf{PSh}$$

In set, a term of type A in context Γ is a function from Γ to A. In tot, a term of type A in context Γ is a natural transformation between the presheaves Γ and A.

$$\mathsf{Tm} : \{b : \mathsf{tag}\}\ (\Gamma : \mathsf{Ctx}\ b)\ (A : \mathsf{Ty}\ b) \to \mathsf{Set}$$
$$\mathsf{Tm}\ \{\mathsf{set}\}\ \Gamma\ A = \Gamma \to A$$
$$\mathsf{Tm}\ \{\mathsf{tot}\}\ \Gamma\ A =$$
$$\quad \Sigma\ ((i : \mathsf{Size}) \to \mathsf{PSh.Obj}\ \Gamma\ i \to \mathsf{PSh.Obj}\ A\ i)$$
$$\qquad (\lambda\ f \to (i : \mathsf{Size})\ (j : \mathsf{Size{<}}\ (\uparrow i))\ (x : \mathsf{PSh.Obj}\ \Gamma\ i)$$
$$\qquad\quad \to \mathsf{PSh.Mor}\ A\ i\ j\ (f\ i\ x) \equiv f\ j\ (\mathsf{PSh.Mor}\ \Gamma\ i\ j\ x))$$

Two types in set are judgementally equal if and only if they are isomorphic as elements of Set. Two types in tot are judgementally equal if and only if they are isomorphic as elements of PSh.

Two terms in set are judgementally equal if and only if they are propositionally equal as functions. Two terms in tot are judgementally equal if and only if they are propositionally equal as natural transformations.

$$\mathsf{def\text{-}eq} : \{b : \mathsf{tag}\}\ (\Gamma : \mathsf{Ctx}\ b)\ (A : \mathsf{Ty}\ b)\ (s\ t : \mathsf{Tm}\ \Gamma\ A) \to \mathsf{Set}$$
$$\mathsf{def\text{-}eq}\ \{\mathsf{set}\}\ \Gamma\ A\ s\ t = (x : \Gamma) \to s\ x \equiv t\ x$$
$$\mathsf{def\text{-}eq}\ \{\mathsf{tot}\}\ \Gamma\ A\ (s\ ,\ p)\ (t\ ,\ q) = (i : \mathsf{Size})\ (x : \mathsf{PSh.Obj}\ \Gamma\ i) \to s\ i\ x \equiv t\ i\ x$$

We write $\bullet$ for the empty context and $\Gamma\ ,,\ A$ for the extension of the context Γ with the type A.

4.2. Simple Types.

4.3. Later.

4.4. Clock Quantification.

4.5. Fix.

4.6. Inductive Types.

5. The Interpretation

6. Conclusion

Appendix A. Omitted Proofs