From the simply-typed lambda-calculus to cartesian closed categories

Andreas Abel

December 2017

Outline of the transformation.

1. Simply-typed lambda-calculus (named variables).
2. Explicit substitutions and judgmental equality.
3. Nameless (de Bruijn) presentation.
4. Removing the judgement for well-typed variables: 0 is the only variable, the others are represented using the weakening substitution.
5. Conflate contexts and types, substitutions and terms: we arrive at an internal language for Cartesian closed categories.

1 Exercises (Pen and Paper)

1.1 Simply-typed lambda-calculus with explicit substitutions

1. Write down the typing rules for simply-typed lambda-calculus.
2. Write down the typing rules for substitutions.
3. Write down the equality rules.
   a) β- and η-equality.
   b) Rules for the propagation of explicit substitutions into terms.
   c) Rules for the composition of substitutions.

1.2 Categories

1. Category of monoids.
   a) Define the category of (small) monoids (where the carrier of the monoid is a set).
   b) Show that the length function for lists is a monoid morphism.
2. Category of categories.
   a) Generalize the notion of monoid morphism to the concept of morphism between categories.
   b) Show that the (small) categories (where the objects form a set) form a category themselves. (This category is large in the sense that objects form a class.)

1.3 Products

1. Define the product in the category of monoids.
2. Let 1 denote the terminal object of some category $\mathcal{C}$. Show that the following span is a product of A and 1.

$$A \xleftarrow{\mathsf{id}} A \xrightarrow{!} 1$$

1.4 Cartesian closed categories

Recall $\mathsf{apply} : (A \Rightarrow B) \times A \longrightarrow B$ and $\mathsf{curry}\, f : C \longrightarrow (A \Rightarrow B)$ for $f : C \times A \longrightarrow B$.

1. Show $\mathsf{curry}\,\mathsf{apply} = \mathsf{id}$.
2. Show $A \cong (1 \Rightarrow A)$.

2 Exercises (Agda)

2.1 Simply-typed lambda-calculus in Agda

1. Represent simply-typed terms in Agda, using de Bruijn indices.
   a) Code simple types a and contexts Γ as data types.
   b) Define well-typed variables as indexed data type $\mathsf{Var}\,\Gamma\,a$.
   c) Define well-typed terms as indexed data type $\mathsf{Tm}\,\Gamma\,a$.
2. Add explicit substitutions via an indexed data type $\mathsf{Sub}\,\Gamma\,\Delta$.
3. Define an equality judgement as relation between well-typed terms: $\_\cong\_ : \mathsf{Tm}\,\Gamma\,a \to \mathsf{Tm}\,\Gamma\,a \to \mathsf{Set}$. Each rule is one constructor of this indexed data type.
4. Likewise, implement an equality judgement for well-typed substitutions.

2.2 CCCs in Agda

1. An E-category is a category with an equivalence relation on homsets. Define the notion of E-category in Agda.
   a) There is a Set of objects Ob.
   b) For each two objects $a, b : \mathsf{Ob}$ there is a Set of (homo)morphisms $\mathsf{Hom}\,a\,b$ from a to b.
   c) There is an equivalence relation on $\mathsf{Hom}\,a\,b$ (for each $a, b : \mathsf{Ob}$).
   d) There is an associative morphism composition $f \circ g : \mathsf{Hom}\,a\,c$ for each $f : \mathsf{Hom}\,b\,c$ and $g : \mathsf{Hom}\,a\,b$.
   e) Composition respects equality.
   f) There are morphisms $\mathsf{id}\,a$ for each $a : \mathsf{Ob}$ which are left and right units for composition.
2. Add products and terminal objects.
3. Add exponentials.

2.3 Interpretation of STLC in CCCs

1. Fix an arbitrary CCC.
2. Write an interpretation of types and contexts as objects in the CCC.
3. Write an interpretation of well-typed terms and substitutions as morphisms in the CCC.
4. Write an interpretation of judgmental equality as equality of morphisms in the CCC.

- First, prove each rule of judgmental equality as a theorem about morphisms in a CCC.
- Then, map the rules to these theorems.