Towards dependent combinator calculus

Thorsten Altenkirch Ambrus Kaposi Artjoms Šinkarovs
Tamás Végh

June 15, 2023

Why combinators?

- Eliminating variables.
- Eliminating schematic definitions.

deBruijn combinators

$$
\begin{aligned}
&\textbf{data}\ \mathsf{Tm} : \mathsf{Con} \to \mathsf{Ty} \to \mathsf{Type}\\
&\textbf{data}\ \mathsf{Tm}\ \textbf{where}\\
&\quad \mathsf{zero} : \mathsf{Tm}\ (\Gamma \triangleright \tau)\ \tau\\
&\quad \mathsf{suc} : \mathsf{Tm}\ \Gamma\ \sigma \to \mathsf{Tm}\ (\Gamma \triangleright \tau)\ \sigma\\
&\quad \mathsf{lam} : \mathsf{Tm}\ (\Gamma \triangleright \tau)\ \sigma \to \mathsf{Tm}\ \Gamma\ (\tau \Rightarrow \sigma)\\
&\quad \_\$\_ : \mathsf{Tm}\ (\tau \Rightarrow \sigma) \to \mathsf{Tm}\ \tau \to \mathsf{Tm}\ \sigma\\
&\quad \_[\_] : \mathsf{Tm}\ \Gamma\ \tau \to \mathsf{Sub}\ \Delta\ \Gamma \to \mathsf{Tm}\ \Delta\ \tau
\end{aligned}
$$

- Cheating, better define suc via substitution and app as inverse of lam.
- Add equations (QIIT)
- Generalizes to dependent types (Initial CwF).

Schönfinkel-Curry combinators

$$
\begin{array}{l}
\textbf{data}\ \mathsf{Tm} : \mathsf{Ty} \to \mathsf{Type}\ \textbf{where} \\
\quad \mathsf{K} \quad : \mathsf{Tm}\ (\tau \Rightarrow \sigma \Rightarrow \tau) \\
\quad \mathsf{S} \quad : \mathsf{Tm}\ ((\tau \Rightarrow \sigma \Rightarrow \delta) \Rightarrow (\tau \Rightarrow \sigma) \Rightarrow \tau \Rightarrow \delta) \\
\quad \_\$\_ : \mathsf{Tm}\ (\tau \Rightarrow \sigma) \to \mathsf{Tm}\ \tau \to \mathsf{Tm}\ \sigma \\
\quad \ldots \\
\mathsf{I} : \mathsf{Tm}\ (\tau \Rightarrow \tau) \\
\mathsf{I} = \mathsf{S}\ \$\ \mathsf{K}\ \$\ \mathsf{K}
\end{array}
$$

Bracket abstraction

$$\mathsf{lam} : \mathsf{Tm}\,(\Gamma \triangleright \tau)\,\sigma \to \mathsf{Tm}\,\Gamma\,(\tau \Rightarrow \sigma)$$

$$\begin{aligned}
\mathsf{lam}\ \mathsf{zero} &= \mathsf{I} \\
\mathsf{lam}\ (\mathsf{suc}\ \mathsf{t}) &= \mathsf{K}\ \$\ \mathsf{t} \\
\mathsf{lam}\ (\mathsf{t}\ \$\ \mathsf{u}) &= \mathsf{S}\ \$\ (\mathsf{lam}\ \mathsf{t})\ \$\ (\mathsf{lam}\ \mathsf{u}) \\
\mathsf{lam}\ \mathsf{K} &= \mathsf{K}\ \$\ \mathsf{K} \\
\mathsf{lam}\ \mathsf{S} &= \mathsf{K}\ \$\ \mathsf{S}
\end{aligned}$$

Extensionality ?

- Add equations

$$K\beta \ : \ \mathsf{K}\ \$\ \mathsf{x}\ \$\ \mathsf{y} \ \equiv \ \mathsf{x}$$
$$S\beta \ : \ \mathsf{S}\ \$\ \mathsf{x}\ \$\ \mathsf{y}\ \$\ \mathsf{z} \ \equiv \ \mathsf{x}\ \$\ \mathsf{z}\ \$\ (\mathsf{y}\ \$\ \mathsf{z})$$

- lam doesn't preserve equality!
- We need to add extensionality equations!
- Purely algebraic proof.

FSCD 23 paper

A.,Kaposi, Šinkarovs, Végh; *Combinatory logic and lambda calculus are equal, algebraically*

Dependent combinators

$$\mathsf{K} : \{\mathsf{A} : \mathsf{Set}\}\,\{\mathsf{B} : \mathsf{A} \to \mathsf{Set}\} \to (\mathsf{a} : \mathsf{A}) \to \mathsf{B}\ \mathsf{a} \to \mathsf{A}$$

$$\begin{aligned}\mathsf{S} : {} & \{\mathsf{A} : \mathsf{Set}\}\,\{\mathsf{B} : \mathsf{A} \to \mathsf{Set}\}\,\{\mathsf{C} : (\mathsf{a} : \mathsf{A}) \to \mathsf{B}\ \mathsf{a} \to \mathsf{Set}\} \\ & \to ((\mathsf{a} : \mathsf{A})\,(\mathsf{b} : \mathsf{B}\ \mathsf{a}) \to \mathsf{C}\ \mathsf{a}\ \mathsf{b}) \\ & \to (\mathsf{g} : (\mathsf{a} : \mathsf{A}) \to (\mathsf{B}\ \mathsf{a})) \\ & \to (\mathsf{a} : \mathsf{A}) \to \mathsf{C}\ \mathsf{a}\ (\mathsf{g}\ \mathsf{a})\end{aligned}$$

- Bracket abstraction fails: $\mathsf{lam}\ \mathsf{K} = \mathsf{K}\ \$\ \mathsf{K}$ because the type may still depend on the variable.

Dependent combinators with a universe

$$\mathsf{Ty} \;:\; \mathsf{Set}$$
$$\mathsf{Tm} \;:\; \mathsf{Ty} \;\to\; \mathsf{Set}$$
$$\mathsf{U} \;:\; \mathsf{Ty}$$
$$\Pi \;:\; (\mathsf{A} \;:\; \mathsf{Ty})\,(\mathsf{B} \;:\; \mathsf{Tm}\,\mathsf{A} \;\to\; \mathsf{Ty}) \;\to\; \mathsf{Ty}$$

with the equation $\mathsf{Tm}\,\mathsf{U} \;=\; \mathsf{Ty}$ $(\mathsf{Type} \;:\; \mathsf{Type})$

$$\mathsf{K} \;:\; (\mathsf{A} \;:\; \mathsf{Ty})\,(\mathsf{B} \;:\; \mathsf{Tm}\,\mathsf{A} \;\to\; \mathsf{Ty})$$
$$\to\; (\mathsf{a} \;:\; \mathsf{Tm}\,\mathsf{A}) \;\to\; \mathsf{Tm}\,(\mathsf{B}\,\mathsf{a}) \;\to\; \mathsf{Tm}\,\mathsf{A}$$
$$\mathsf{S} \;:$$

$$\lambda\,(\mathsf{x} \;:\; \mathsf{C}) \;\to\; \mathsf{K} \;=\; \mathsf{K}\;\mathsf{K}\text{-}\mathsf{Type}\;(\mathsf{K}\;\mathsf{C})\;\mathsf{K}$$

Deriving non-dependent combinators

- Can we derive the non-dependent combinator

$$K_0 \; : \; (\mathsf{A}\; \mathsf{B} \; : \; \mathsf{Ty}) \; \to \; \mathsf{Tm}\; \mathsf{A} \; \to \; \mathsf{Tm}\; \mathsf{B} \; \to \; \mathsf{Tm}\; \mathsf{A}$$

- The obvious definition ends up using K_0 again!

$$K_0\; \mathsf{A}\; \mathsf{B} \; = \; \mathsf{K}\; \mathsf{A}\; (K_0\; \mathsf{U}\; \mathsf{B})$$

- We can add both dependent and non-dependent combinators.
- But this seems a undesirable duplication.
- Alternative: just add I:

$$\mathsf{I} \; : \; (\mathsf{A} \; : \; \mathsf{Ty}) \; \to \; \mathsf{Tm}\; \mathsf{A} \; \to \; \mathsf{Tm}\; \mathsf{A}$$
$$K_0\; \mathsf{A}\; \mathsf{B} \; = \; \mathsf{K}\; (\mathsf{K}\; (\mathsf{K}\; (\mathsf{K}\; (\mathsf{I}\; \mathsf{u})\; \mathsf{u})\; \mathsf{A})\; \mathsf{B})$$

Where is Münchhausen?

- We want

$$\Pi \;:\; \mathsf{Tm}\,(\Pi\,\mathsf{U}\,(\lambda\,\mathsf{A} \;\to\; (\mathsf{A} \;\to\; \mathsf{U}) \;\to\; \mathsf{U}))$$

- This can be resolved using preterms (standard approach to very dependent types).
- We avoid preterms by defining Π in stages, using forward declarations.
- E.g. we define

$$\begin{aligned}
&_ \Rightarrow_0 _ \;:\; \mathsf{Ty} \;\to\; \mathsf{Ty} \;\to\; \mathsf{Ty}\\
&_\$\$_0_ \;:\; \mathsf{Tm}\,(\mathsf{A} \Rightarrow_0 \mathsf{B}) \;\to\; \mathsf{Tm}\,\mathsf{A} \;\to\; \mathsf{Tm}\,\mathsf{B}\\
&\Pi_0 \;:\; (\mathsf{A} : \mathsf{Ty})\,(\mathsf{B} : \mathsf{Tm}\,(\mathsf{A} \Rightarrow_0 \mathsf{U})) \;\to\; \mathsf{Ty}\\
&_\$_0_ \;:\; \mathsf{Tm}\,(\Pi_0\,\mathsf{A}\,\mathsf{B}) \;\to\; (\mathsf{a} : \mathsf{Tm}\,\mathsf{A}) \;\to\; \mathsf{Tm}\,(\mathsf{B}\,\$\$_0\,\mathsf{a})\\
&\mathsf{A} \Rightarrow_0 \mathsf{B} \;=\; \Pi_0\,\mathsf{A}\,(\mathsf{KK}_0\,\$\$_0\,\mathsf{B})
\end{aligned}$$

To do

- Complete the definition of combinatory models for dependent types.
- Add the extensionality axioms as for the simply typed case and show equivalence to CWFs with Π-types.
- Understand the semantics of using Münchhausen (forward declarations) in Type Theory.