Zipper Contexts

Intro

What we do here is explore an interesting idea. In some algorithms for bidirectional typechecking of polymorphic lambda calculus, there appears a special thing called a mark. The idea is that the mark designates the place in the context up until which type variables can be generalized when forming universally quantified types.

Since such a mark is just another way of extending the context, besides $\Gamma, x : A$ (which corresponds to $\lambda x : A. e$) and Γ, α which corresponds to $\Lambda \alpha. e$, it might be worthwhile to explore what would happen if we had more ways of extending the context that correspond to terms?

The answer is interesting: contexts would turn into zippers!

Contexts as zippers

Contexts:
$\Gamma ::=$
$\quad \cdot \mid$
$\quad \Gamma, \lambda x. \circ, x : A \mid \Gamma, \circ\ e_2 \mid \Gamma, e_1\ \circ \mid$
$\quad \Gamma, (\circ, e_2) \mid \Gamma, (e_1, \circ) \mid \Gamma, \text{outl}\ \circ \mid \Gamma, \text{outr}\ \circ \mid$
$\quad \Gamma, \text{inl}\ \circ \mid \Gamma, \text{inr}\ \circ \mid \Gamma, \text{case}\ \circ\ \text{of}\ (f, g) \mid$
$\quad \Gamma, \text{case}\ e\ \text{of}\ (x. \circ, g), x : A \mid \Gamma, \text{case}\ e\ \text{of}\ (f, x. \circ), x : A \mid$
$\quad \Gamma, \mathbf{0}\text{-elim}\ \circ$

Contexts

To be clear, $(x : A) \in \Gamma$ if and only if either
$\Gamma = \Gamma_L, \lambda x.\, \circ, x : A, \Gamma_R$, or $\Gamma = \Gamma_L, \lambda x.\, \circ, x : A := e, \Gamma_R$, or
$\Gamma = \Gamma, \mathtt{case}\ e\ \mathtt{of}\ (x.\, \circ, y.\, e_2), x : A$

Terms

Terms:
$e ::=$
$\quad x \mid$
$\quad \lambda x.\, e \mid e_1\ e_2 \mid$
$\quad (e_1, e_2) \mid \text{outl } e \mid \text{outr } e \mid$
$\quad \text{inl } e \mid \text{inr } e \mid \text{case } e \text{ of } (e_1, e_2) \mid$
$\quad \text{unit} \mid \mathbf{0}\text{-elim } e$

Declarative typing – basics

$$\frac{(x : A) \in \Gamma}{\Gamma \vdash x : A} \textsc{Var}$$

Declarative typing – type-directed rules

$$\frac{\Gamma, \lambda x. \circ, x : A \vdash e : B}{\Gamma \vdash \lambda x.\, e : A \to B} \qquad \frac{\Gamma, \circ\; a \vdash f : A \to B \quad \Gamma, f \circ \vdash a : A}{\Gamma \vdash f\; a : B}$$

$$\frac{\Gamma, (\circ, b) \vdash a : A \quad \Gamma, (a, \circ) \vdash b : B}{\Gamma \vdash (a, b) : A \times B}$$

$$\frac{\Gamma, \mathtt{outl}\; \circ \vdash e : A \times B}{\Gamma \vdash \mathtt{outl}\; e : A} \qquad \frac{\Gamma, \mathtt{outr}\; \circ \vdash e : A \times B}{\Gamma \vdash \mathtt{outr}\; e : B}$$

$$\frac{\Gamma, \mathtt{inl}\; \circ \vdash e : A}{\Gamma \vdash \mathtt{inl}\; e : A + B} \qquad \frac{\Gamma, \mathtt{inr}\; \circ \vdash e : B}{\Gamma \vdash \mathtt{inr}\; e : A + B}$$

Declarative typing – type-directed rules

$$\frac{\begin{array}{c} \Gamma, \mathtt{case}\ \circ\ \mathtt{of}\ (x.\,e_1, y.\,e_2) \vdash e : A + B \\ \Gamma, \mathtt{case}\ e\ \mathtt{of}\ (x.\,\circ, y.\,e_2), x : A \vdash e_1 : C \\ \Gamma, \mathtt{case}\ e\ \mathtt{of}\ (x.\,e_1, y.\,\circ), y : B \vdash e_2 : C \end{array}}{\Gamma \vdash \mathtt{case}\ e\ \mathtt{of}\ (x.\,e_1, y.\,e_2) : C}$$

$$\frac{}{\Gamma \vdash \mathtt{unit} : \mathbf{1}} \qquad \frac{\Gamma, \mathbf{0}\text{-}\mathtt{elim}\ \circ \vdash e : \mathbf{0}}{\Gamma \vdash \mathbf{0}\text{-}\mathtt{elim}\ e : A}$$

What's the point?

What is the point of it? By now I don't really remember, but I think it was that we add definitions to the context, i.e. $\Gamma, x : A := e$, and then we turn the term-related context extensions into smart constructors, which can turn ordinary variable bindings into definitions.

$$\Gamma, \circ \ e, \lambda x. \circ, x : A \rightsquigarrow \Gamma, \circ \ e, \lambda x. \circ, x : A := e$$

$$\Gamma, \mathtt{case} \ (\mathtt{inl} \ e) \ \mathtt{of} \ (x. \circ, y. \ e_2), x : A \rightsquigarrow$$
$$\Gamma, \mathtt{case} \ (\mathtt{inl} \ e) \ \mathtt{of} \ (x. \circ, y. \ e_2), x : A := e$$

$$\Gamma, \mathtt{case} \ (\mathtt{inr} \ e) \ \mathtt{of} \ (x. \ e_1, y. \circ), y : B \rightsquigarrow$$
$$\Gamma, \mathtt{case} \ (\mathtt{inr} \ e) \ \mathtt{of} \ (x. \ e_1, y. \circ), y : B := e$$