Zipper Contexts

Wojciech Kołowski

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 contextup 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 worthwile 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:

$$
\begin{aligned}
\Gamma ::= \\
& \cdot \mid \\
& \Gamma, \lambda x.\circ, x : A \mid \Gamma, \circ\ e_2 \mid \Gamma, e_1\ \circ \mid \\
& \Gamma, (\circ, e_2) \mid \Gamma, (e_1, \circ) \mid \Gamma, \texttt{outl}\ \circ \mid \Gamma, \texttt{outr}\ \circ \mid \\
& \Gamma, \texttt{inl}\ \circ \mid \Gamma, \texttt{inr}\ \circ \mid \Gamma, \texttt{case}\ \circ\ \texttt{of}\ (f, g) \mid \\
& \Gamma, \texttt{case}\ e\ \texttt{of}\ (x.\circ, g), x : A \mid \Gamma, \texttt{case}\ e\ \texttt{of}\ (f, x.\circ), x : A \mid \\
& \Gamma, \texttt{exfalso}\ \circ
\end{aligned}
$$

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

Terms

Terms:

$e ::=$

$x \mid$

$\lambda x.\, e \mid e_1\ e_2 \mid$

$(e_1, e_2) \mid \texttt{outl}\ e \mid \texttt{outr}\ e \mid$

$\texttt{inl}\ e \mid \texttt{inr}\ e \mid \texttt{case}\ e\ \texttt{of}\ (e_1, e_2) \mid$

$\texttt{unit} \mid \texttt{exfalso}\ e$

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

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

$$\frac{}{\Gamma \vdash \texttt{unit} : \mathbf{1}} \qquad \frac{\Gamma, \texttt{exfalso}\ \circ \vdash e : \mathbf{0}}{\Gamma \vdash \texttt{exfalso}\ e : A}$$

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.

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

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

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