A taste of Agda

Francesco Mazzoli <f@mazzo.li>

June 2013

Agda?

A functional programming language with a powerful type system.

We're going to define a simple language and a type checker.

If we have time, we will also prove a simple optimisation for said language correct.

`http://mazzo.li/posts/Lambda.html`

Credits

Credits:

- Conor McBride & James McKinna, *The View from the Left*, `http://strictlypositive.org/view.ps.gz`;
- and Adam Chlipala, POPL 2013—see also *Certified Programming with Dependent Types*, `http://adam.chlipala.net/cpdt/`.

Simply Typed λ-calculus

$$\begin{aligned}
\mathsf{Type} &::= \mathsf{nat} \mid \mathsf{Type} \Rightarrow \mathsf{Type} \\
\mathsf{Term} &::= x \\
&\mid \mathsf{Term}\ \mathsf{Term} \\
&\mid \lambda x : \mathsf{Type} \to \mathsf{Term} \\
&\mid n \\
&\mid \mathsf{Term} + \mathsf{Term}
\end{aligned}$$

de Bruijn indices

Humans usually use names to bind things. Convenient but tricky for computers.

Instead, we will use *de Bruijn indices*:

Named	**Nameless**
$\lambda x \to x$	$\lambda\ 0$
$\lambda x \to \lambda y \to x$	$\lambda\ \lambda\ 1$
$\lambda x \to \lambda y \to \lambda z \to x\ z\ (y\ z)$	$\lambda\ \lambda\ \lambda\ 2\ 0\ (1\ 0)$

Simply Typed, nameless λ-calculus

$$\begin{aligned}
\text{Type} &::= \text{nat} \mid \text{Type} \Rightarrow \text{Type} \\
\text{Term} &::= v \\
&\mid \text{Term}\ \text{Term} \\
&\mid \lambda \text{Type} \rightarrow \text{Term} \\
&\mid n \\
&\mid \text{Term} + \text{Term}
\end{aligned}$$

A simple type system

$$\frac{}{\Gamma \vdash \mathsf{var}\ v : \mathsf{lookup}\ v\ \Gamma}$$

$$\frac{}{\Gamma \vdash \mathsf{lit}\ n : \mathsf{nat}}$$

$$\frac{\Gamma \vdash t : \mathsf{nat} \qquad \Gamma \vdash u : \mathsf{nat}}{\Gamma \vdash t \oplus u : \mathsf{nat}}$$

$$\frac{\Gamma \vdash t : \sigma \Rightarrow \tau \qquad \Gamma \vdash u : \sigma}{\Gamma \vdash t \cdot u : \tau}$$

$$\frac{\sigma :: \Gamma \vdash t : \tau}{\Gamma \vdash \mathsf{lam}\ \sigma\ t : \sigma \Rightarrow \tau}$$