Formalism for Datatype Extensions in Haskell

Tim Zakian

July 17, 2015

1 Language Definition

- Why does this not create open datatypes? How does it differ and how does it get around the problems found in open datatypes?
- Currently only have core $F_C(X)$ as presented in [1]. Need to extend this with the extendable data type section (i.e., the hard stuff...).

Symbol Classes

$$
\begin{array}{lll}
a, b, c, co & \rightarrow & \langle \text{type variable} \rangle \\
x, f & \rightarrow & \langle \text{term variable} \rangle \\
C & \rightarrow & \langle \text{coercion constant} \rangle \\
T & \rightarrow & \langle \text{value type constructor} \rangle \\
S_n & \rightarrow & \langle n-\text{ary type function} \rangle \\
K & \rightarrow & \langle \text{data constructor} \rangle
\end{array}
$$

Declarations

$$
\begin{array}{lll}
pgm & \rightarrow & \overline{decl}; e \\
decl & \rightarrow & \textbf{data}\ T\ :\ \overline{\kappa} \rightarrow \star\ \textbf{where} \\
 & & \overline{K : \forall\ \overline{a : \kappa}.\ \forall\ \overline{b : \iota}.\overline{\sigma} \rightarrow T\ \overline{a}} \\
 & | & \textbf{type}\ S_n\ :\ \overline{\kappa}^n \rightarrow \iota \\
 & | & \textbf{axiom}\ C\ :\ \sigma_1 \sim \sigma_2
\end{array}
$$

Sorts and Kinds

$$
\begin{array}{llll}
\delta & \rightarrow & TY \mid CO & \text{Sorts} \\
\kappa, \iota & \rightarrow & \star \mid \kappa_1 \rightarrow \kappa_2 \mid \sigma_1 \sim \sigma_2 & \text{Kinds}
\end{array}
$$

Types and Coercions

$$
\begin{array}{llll}
d & \rightarrow & a \mid T & \text{Atom of sort TY} \\
g & \rightarrow & c \mid C & \text{Atom of sort CO} \\
\varphi, \rho, \sigma, \tau, \nu, \gamma & \rightarrow & a \mid C \mid T \mid \varphi_1\ \varphi_2 \mid S_n\ \overline{\varphi}^n \mid \forall a : \kappa.\varphi & \\
 & | & \text{sym}\ \gamma \mid \gamma_1 \circ \gamma_2 \mid \gamma @ \varphi \mid \text{left}\ \gamma \mid \text{right}\ \gamma & \\
 & | & \gamma \sim \gamma \mid \text{rightc}\ \gamma \mid \text{leftc}\ \gamma \mid \gamma \blacktriangleright \gamma &
\end{array}
$$

Syntactic sugar

Types $\kappa \Rightarrow \sigma \ \equiv\ \forall\_ : \kappa.\sigma$

Terms

$$
\begin{array}{llll}
u & \rightarrow & x \mid K & \text{Variables and data consructors} \\
e & \rightarrow & u & \text{Term atoms} \\
 & | & \Lambda a : \kappa.e \mid e\varphi & \text{Type abstractions/application} \\
 & | & \lambda x : \sigma.e \mid e_1\ e_2 & \text{Term abstraction/application} \\
 & | & \textbf{let}\ x : \sigma = e_1\ \textbf{in}\ e_2 & \\
 & | & \textbf{case}\ e_1\ \textbf{of}\ \overline{p \rightarrow e_2} & \\
 & | & e \blacktriangleright \gamma & \text{Cast} \\
 & & & \\
p & \rightarrow & K\ \overline{b : \kappa}\ \overline{x : \sigma} & \text{Pattern}
\end{array}
$$

Environments

$\Gamma \quad \rightarrow \quad \epsilon \mid \Gamma, u : \sigma \mid \Gamma, d : \kappa \mid \Gamma, g : \kappa \mid \Gamma, S_n : \kappa$

A top-level environment binds only type constructors, T, S_n, data constructors K, and coercion constants C.

Figure 1: The core language for extensible data types – $F_C(X)$

$$\boxed{\Gamma \vdash_{TY} \sigma : \kappa}$$

$$\text{TyVar}\ \frac{(d:\kappa) \in \Gamma \quad \Gamma \vdash_k \kappa : TY}{\Gamma \vdash_{TY} d : \kappa} \qquad \text{TyApp}\ \frac{\Gamma \vdash_{TY} \sigma_1 : \kappa_1 \to \kappa_2 \quad \Gamma \vdash_{TY} \sigma_2 : \kappa_1}{\Gamma \vdash_{TY} \sigma_1\ \sigma_2 : \kappa_2}$$

$$\text{TySCon}\ \frac{(S_n : \overline{\kappa}^n \to \iota) \in \Gamma \quad \Gamma \vdash_{TY} \overline{\sigma : \kappa}^n}{\Gamma \vdash_{TY} S_n \overline{\sigma}^n : \iota} \qquad \text{TyAll}\ \frac{\Gamma, a : \kappa \vdash_{TY} \sigma : \star \quad \Gamma \vdash_k \kappa : \delta \quad a \notin fv(\Gamma)}{\Gamma \vdash_{TY} \forall a : \kappa . \sigma : \star}$$

$$\boxed{\Gamma \vdash_{CO} \gamma : \sigma \sim \tau}$$

$$\text{CoRefl}\ \frac{(a:\kappa) \in \Gamma \quad \Gamma \vdash_k \kappa : TY}{\Gamma \vdash_{CO} a : a \sim a} \qquad \text{CoVar}\ \frac{(g : \sigma \sim \tau) \in \Gamma}{\Gamma \vdash_{CO} g : \sigma \sim \tau} \qquad \text{CoAllT}\ \frac{\begin{array}{c}\Gamma, a : \kappa \vdash_{CO} \gamma : \sigma \sim \tau \\ \Gamma \vdash_k \kappa : TY \quad a \notin fv(\Gamma)\end{array}}{\Gamma \vdash_{CO} \forall a : \kappa . \gamma : \forall a : \kappa . \sigma \sim \forall a : \kappa . \tau}$$

$$\text{CoInstT}\ \frac{\begin{array}{c}\Gamma \vdash_{CO} \gamma : \forall a . \kappa . \sigma \sim \forall b : \kappa . \tau \\ \Gamma \vdash_{TY} \nu : \kappa\end{array}}{\Gamma \vdash_{CO} \gamma @ \nu : [\nu/a]\sigma \sim [\nu/b]\tau} \qquad \text{SComp}\ \frac{\Gamma \vdash_{CO} \overline{\gamma : \sigma \sim \tau}^n \quad \Gamma \vdash_{TY} S_n \overline{\sigma}^n : \kappa}{\Gamma \vdash_{CO} S_n \overline{\gamma}^n : S_n \overline{\sigma}^n \sim S_n \overline{\tau}^n} \qquad \text{Sym}\ \frac{\Gamma \vdash_{CO} \gamma : \sigma \sim \tau}{\Gamma \vdash_{CO} \gamma : \tau \sim \sigma}$$

$$\text{Trans}\ \frac{\begin{array}{c}\Gamma \vdash_{CO} \gamma_1 : \sigma_1 \sim \sigma_2 \\ \Gamma \vdash_{CO} \gamma_2 : \sigma_2 \sim \sigma_3\end{array}}{\Gamma \vdash_{CO} \gamma_1 \circ \gamma_2 : \sigma_1 \sim \sigma_3} \qquad \text{Comp}\ \frac{\begin{array}{c}\Gamma \vdash_{CO} \gamma_1 : \sigma_1 \sim \tau_1 \quad \Gamma \vdash_{CO} \gamma_2 : \sigma_2 \sim \tau_2 \\ \Gamma \vdash_{TY} \sigma_1\ \sigma_2 : \kappa\end{array}}{\Gamma \vdash_{CO} \gamma_1\ \gamma_2 : \sigma_1\ \sigma_2 \sim \tau_1\ \tau_2}$$

$$\text{Left}\ \frac{\Gamma \vdash_{CO} \gamma : \sigma_1\ \sigma_2 \sim \tau_1\ \tau_2}{\Gamma \vdash_{CO} \mathsf{left}\ \gamma : \sigma_1 \sim \tau_1} \qquad \text{Right}\ \frac{\Gamma \vdash_{CO} \gamma : \sigma_1\ \sigma_2 \sim \tau_1\ \tau_2}{\Gamma \vdash_{CO} \mathsf{right}\ \gamma : \sigma_1 \sim \tau_1}$$

$$\text{CompC}\ \frac{\begin{array}{c}\Gamma \vdash_{CO} \gamma : \kappa_1 \sim \kappa_2 \quad \Gamma \vdash_{CO} \gamma' : \sigma_1 \sim \sigma_2 \\ \Gamma \vdash_k \kappa_1 : CO\end{array}}{\Gamma \vdash_{CO} \gamma \Rightarrow \gamma' : (\kappa_1 \Rightarrow \sigma_1) \sim (\kappa_2 \Rightarrow \sigma_2)} \qquad \text{LeftC}\ \frac{\Gamma \vdash_{CO} \gamma : \kappa_1 \Rightarrow \sigma_1 \sim \kappa_2 \Rightarrow \sigma_2}{\Gamma \vdash_{CO} \mathsf{leftc}\ \gamma : \kappa_1 \sim \kappa_2}$$

$$\text{RightC}\ \frac{\Gamma \vdash_{CO} \gamma : \kappa_1 \Rightarrow \sigma_1 \sim \kappa_2 \Rightarrow \sigma_2}{\Gamma \vdash_{CO} \mathsf{rightc}\ \gamma : \sigma_1 \sim \sigma_2} \qquad (\sim)\ \frac{\Gamma \vdash_{CO} \gamma_1 : \sigma_1 \sim \tau_1 \quad \Gamma \vdash_{CO} \gamma_2 : \sigma_2 \sim \tau_2}{\Gamma \vdash_{CO} \gamma_1 \sim \gamma_2 : (\sigma_1 \sim \sigma_2) \sim (\tau_1 \sim \tau_2)}$$

$$\text{CastC}\ \frac{\Gamma \vdash_{CO} \gamma_1 : \kappa \quad \Gamma \vdash_{CO} \gamma_2 \kappa \sim \kappa'}{\Gamma \vdash_{CO} \gamma_1 \blacktriangleright \gamma_2}$$

$$\boxed{\Gamma \vdash_e e : \sigma}$$

$$\text{Var}\ \frac{(u : \sigma) \in \Gamma}{\Gamma \vdash_e u : \sigma} \qquad \text{Case}\ \frac{\Gamma \vdash_e e : \sigma \quad \overline{\Gamma \vdash_p p \to e : \sigma \to \tau}}{\Gamma \vdash_e \mathbf{case}\ e\ \mathbf{of}\ \overline{p \to e} : \tau} \qquad \text{Let}\ \frac{\Gamma \vdash_e e_1 : \sigma_1 \quad \Gamma, x : \sigma_1 \vdash_e e_2 : \sigma_2}{\Gamma \vdash_e \mathbf{let}\ x : \sigma_1 = e_1\ \mathbf{in}\ e_2 : \sigma_2}$$

$$\text{Cast}\ \frac{\Gamma \vdash_e e : \sigma \quad \Gamma \vdash_{CO} \gamma : \sigma \sim \tau}{\Gamma \vdash_e e \blacktriangleright \gamma : \tau} \qquad \text{Abs}\ \frac{\begin{array}{c}\Gamma \vdash_{TY} \sigma_x : \star \\ \Gamma, x : \sigma_x \vdash_e e : \sigma\end{array}}{\Gamma \vdash_e \lambda x : \sigma_x . e : \sigma_x \to \sigma} \qquad \text{App}\ \frac{\begin{array}{c}\Gamma \vdash_e e_1 : \sigma_2 \to \sigma_1 \\ \Gamma \vdash_e e2 : \sigma_2\end{array}}{\Gamma \vdash_e e_1\ e_2 : \sigma_1}$$

$$\text{AbsT}\ \frac{\Gamma a : \kappa \vdash_e e : \sigma \quad \Gamma \vdash_k \kappa : \delta \quad a \notin fv(\Gamma)}{\Gamma \vdash_e \Lambda a : \kappa . e : \forall a : \kappa . \sigma} \qquad \text{AppT}\ \frac{\Gamma \vdash_e e : \forall a : \kappa . \sigma \quad \Gamma \vdash_k \kappa : \delta \quad \Gamma \vdash_\delta \varphi : \kappa}{\Gamma \vdash_e e\ \varphi : \sigma[\varphi/a]}$$

$$\boxed{\Gamma \vdash_p p \to e : \sigma \to \tau}$$

$$\text{Alt}\ \frac{K : \forall \overline{a : \kappa} . \forall \overline{b : \iota} . \overline{\sigma} \to T\ \overline{a} \in \Gamma \quad \theta = [\overline{\nu/a}] \quad \Gamma, \overline{b : \theta(\iota), \overline{x : \theta(\sigma)}} \vdash_e e : \tau}{\Gamma \vdash_p K\ \overline{b : \theta(\iota)}\ \overline{x : \theta(\sigma)} \to e : T\ \overline{\nu} \to \tau}$$

Figure 2: Typing rules for the core language of $F_C(X)$

$$\boxed{\Gamma \vdash \mathsf{decl} : \Gamma'}$$

$$\text{Data}\ \frac{\overline{\Gamma \vdash_{TY} \sigma : \star} \qquad \Gamma \vdash_k \kappa : \mathsf{TY}}{\Gamma \vdash \textbf{data}\ T : \kappa\ \textbf{where}\ \overline{K : \sigma} : (T : \kappa, \overline{K : \sigma})}$$

$$\text{Type}\ \frac{\Gamma \vdash_k \kappa : \mathsf{TY}}{\Gamma \vdash (\textbf{type}\ S : \kappa) : (S : \kappa)} \qquad \text{Coerce}\ \frac{\Gamma \vdash_k \kappa : \mathsf{CO}}{\Gamma \vdash (\textbf{axiom}\ C : \kappa) : (C : \kappa)}$$

$$\boxed{\Gamma \vdash \mathsf{pgm} : \sigma}$$

$$\text{Pgm}\ \frac{\overline{\Gamma \vdash \mathsf{decl} : \Gamma_d} \qquad \Gamma = \Gamma_0, \overline{\Gamma_d} \qquad \Gamma \vdash_e e : \sigma}{\Gamma_0 \vdash \overline{\mathsf{decl}}; e : \sigma}$$

Figure 3: Environment extension rules for $F_C(X)$

2 Typing Rules

References

[1] Martin Sulzmann, Manuel MT Chakravarty, Simon Peyton Jones, and Kevin Donnelly. System f with type equality coercions. In *Proceedings of the 2007 ACM SIGPLAN international workshop on Types in languages design and implementation*, pages 53–66. ACM, 2007.