

Formalism for Datatype Extensions in Haskell

Tim Zakian

July 15, 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?

2 Typing Rules

Symbol Classes

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	$\langle \text{data constructor} \rangle$

Declarations

pgm	\rightarrow	$\overline{\text{decl}}; e$
decl	\rightarrow	data $T : \bar{\kappa} \rightarrow \star$ where $\frac{\kappa : \forall \bar{a} : \bar{\kappa}. \forall \bar{b} : \iota. \bar{\sigma} \rightarrow T \bar{a}}{\text{type } S_n : \bar{\kappa}^n \rightarrow \iota}$ $\text{axiom } C : \sigma_1 \sim \sigma_2$

Sorts and Kinds

δ	\rightarrow	$\text{TY} \mid \text{CO}$	Sorts
κ, ι	\rightarrow	$\star \mid \kappa_1 \rightarrow \kappa_2 \mid \sigma_1 \sim \sigma_2$	Kinds

Types and Coercions

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

Syntactic sugar

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

Terms

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

Environments

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