Lambda Semantics

Giuseppe Lomurno

1 TYPING RULES

$$\frac{n \in \mathbb{Z}}{\Gamma \vdash \textbf{IntE}\ n : \textbf{LInt}}\ \textsc{TyInt}$$

$$\frac{b \in \mathbb{B}}{\Gamma \vdash \textbf{BoolE}\ b : \textbf{LBool}}\ \textsc{TyBool}$$

$$\frac{}{\Gamma \vdash \textbf{UnitE} : \textbf{LUnit}}\ \textsc{TyUnit}$$

$$\frac{\Gamma \vdash e_1 : \tau_1 \qquad \Gamma \vdash e_2 : \tau_2}{\Gamma \vdash \textbf{Pair}\ e_1\ e_2 : \textbf{LProduct}\ \tau_1\ \tau_2}\ \textsc{TyProduct}$$

$$\frac{\Gamma \vdash e : \tau_1}{\Gamma \vdash \textbf{LeftE}\ e : \textbf{LSum}\ \tau_1\ \tau_2}\ \textsc{TyCoProductLeft}$$

$$\frac{\Gamma \vdash e : \tau_2}{\Gamma \vdash \textbf{RightE}\ e : \textbf{LSum}\ \tau_1\ \tau_2}\ \textsc{TyCoProductRight}$$

$$\frac{x : \tau \in \Gamma}{\Gamma \vdash \textbf{Var}\ x : \tau}\ \textsc{TyVar}$$

$$\frac{\Gamma, x : \tau_1 \vdash e : \tau_2}{\Gamma \vdash \textbf{Lambda}\ [\tau_1]\ e : \textbf{LArrow}\ \tau_1\ \tau_2}\ \textsc{TyLambda}$$

$$\frac{\Gamma \vdash e_1 : \textbf{LArrow}\ \tau_1\ \tau_2 \qquad \Gamma \vdash e_2 : \tau_1}{\Gamma \vdash \textbf{App}\ e_1\ e_2 : \tau_2}\ \textsc{TyApp}$$

$$\frac{\Gamma \vdash e : \textbf{LArrow}\ \tau\ \tau}{\Gamma \vdash \textbf{Fix}\ e : \tau}\ \textsc{TyFix}$$

$$\frac{\Gamma \vdash e_1 : \tau_1 \qquad \Gamma \vdash e_2 : \tau_1}{\Gamma \vdash \textbf{binop}\ e_1\ e_2 : \tau_2}\ \textsc{TyPrimBinOp}$$

$$\frac{\Gamma \vdash e : \tau_1}{\Gamma \vdash \textbf{op}\ e : \tau_2}\ \textsc{TyPrimOp}$$

$$\frac{\Gamma \vdash e_1 : \textbf{LBool} \qquad \Gamma \vdash e_2 : \tau \qquad \Gamma \vdash e_3 : \tau}{\Gamma \vdash \textbf{Cond}\ e_1\ e_2\ e_3 : \tau}\ \textsc{TyCond}$$

$$\frac{\Gamma \vdash e_1 : \textbf{LSum}\ \tau_1\ \tau_2 \qquad \Gamma \vdash e_2 : \textbf{LArrow}\ \tau_1\ \tau_3 \qquad \Gamma \vdash e_3 : \textbf{LArrow}\ \tau_2\ \tau_3}{\Gamma \vdash \textbf{Case}\ e_1\ e_2\ e_3 : \tau_3}\ \textsc{TyCase}$$

binop	τ_1	τ_2
PrimAdd	LInt	LInt
PrimSub	LInt	LInt
PrimMul	LInt	LInt
PrimDiv	LInt	LInt
PrimIntEq	LInt	LBool
PrimBoolEq	LBool	LBool
PrimAnd	LBool	LBool
PrimOr	LBool	LBool

op	τ_1	τ_2
PrimNeg	LInt	LInt
PrimNot	LBool	LBool
PrimFst	LProduct a b	a
PrimSnd	LProduct a b	b

Table 1: Arguments and result types of primitive functions.

2 BIG-STEP SEMANTICS

$$\frac{}{\textbf{IntE}\ v \Downarrow v}\ \textsc{SemInt}$$

$$\frac{}{\textbf{BoolE}\ v \Downarrow v}\ \textsc{SemBool}$$

$$\frac{}{\textbf{UnitE} \Downarrow ()}\ \textsc{SemUnit}$$

$$\frac{e_1 \Downarrow v_1 \qquad e_2 \Downarrow v_2}{\textbf{Pair}\ e_1\ e_2 \Downarrow (v_1, v_2)}\ \textsc{SemProduct}$$

$$\frac{e \Downarrow v}{\textbf{LeftE}\ e \Downarrow \textbf{Left}\ v}\ \textsc{SemCoProductLeft}$$

$$\frac{e \Downarrow v}{\textbf{RightE}\ e \Downarrow \textbf{Right}\ v}\ \textsc{SemCoProductRight}$$

$$\frac{}{\textbf{Lambda}\ \_\ e \Downarrow v \to [{}^{v}/_{x}]e}\ \textsc{SemLambda}$$

$$\frac{e_1 \Downarrow f : a \to [{}^{a}/_{x}]e \qquad e_2 \Downarrow v_2 \qquad f\ v_2 \Downarrow v}{\textbf{App}\ e_1\ e_2 \Downarrow v}\ \textsc{SemApp}$$

$$\frac{e \Downarrow f : a \to [{}^{a}/_{x}]e' \qquad f\ (\textbf{Fix}\ e) \Downarrow v}{\textbf{Fix}\ e \Downarrow v}\ \textsc{SemFix}$$

$$\frac{e_1 \Downarrow v_1 \qquad e_2 \Downarrow v_2 \qquad \textbf{binop}\ v_1\ v_2 = v}{\textbf{binop}\ e_1\ e_2 \Downarrow v}\ \textsc{SemPrimBinOp}$$

$$\frac{e \Downarrow v \qquad \textbf{op}\ v = v'}{\textbf{op}\ e \Downarrow v'}\ \textsc{SemPrimOp}$$

$$\frac{e_1 \Downarrow \textbf{True} \qquad e_2 \Downarrow v}{\textbf{Cond}\ e_1\ e_2\ e_3 \Downarrow v}\ \textsc{SemCondTrue}$$

$$\frac{e_1 \Downarrow \textbf{False} \qquad e_3 \Downarrow v}{\textbf{Cond}\ e_1\ e_2\ e_3 \Downarrow v}\ \textsc{SemCondFalse}$$

$$\frac{e_1 \Downarrow \textbf{Left}\ v \qquad e_2 \Downarrow f : a \to [{}^{a}/_{x}]e_2' \qquad f\ v \Downarrow v'}{\textbf{Case}\ e_1\ e_2\ e_3 \Downarrow v'}\ \textsc{SemCaseLeft}$$

$$\frac{e_1 \Downarrow \textbf{Right}\ v \qquad e_3 \Downarrow f : a \to [{}^{a}/_{x}]e_3' \qquad f\ v \Downarrow v'}{\textbf{Case}\ e_1\ e_2\ e_3 \Downarrow v'}\ \textsc{SemCaseRight}$$

Lambda	Haskell
LInt	Int
LBool	Bool
LUnit	()
LProduct a b	(a, b)
LSum a b	Either a b
LArrow	AST -> AST

Table 2: Concrete Haskell representation of Lambda types.