Ligo Formal Description

October 29, 2019

Syntax

variables(x)
label (l)
constructor (c)

$declaration(d) =$

$\mid type\ x\ is\ te$ (Type declaration)

$\mid const\ x\ (:\ te)\ =\ e$ (Const declaration)

$typeexpression(te) =$

$\mid (*\ te_i)$ (Type tuple)

$\mid (\mid\ l_i\ of\ te_i)$ (Type sum)

$\mid \{\ l_i\ :\ te_i\ \}$ (Type record)

$\mid te1\ \to\ te2$ (function)

$\mid l$ (variable)

$\mid l\ (te_i)$ (type of built in function)

$expression(e) =$

$\mid value$ (values)

$\mid built_in$ (built-in function)

$\mid x$ (variables)

$\mid \lambda x\ .\ expr$ (lambda)

$\mid e1\ e2$ (application)

$\mid let\ x\ =\ e1\ in\ e2$ (let in)

$\mid (\ e_i\ )$ (tuple)

$\mid c\ e$ (constructor)

$\mid \{\ l_i\ =\ e_i\ \}$ (record)

$\mid [\ e1_i\ =\ e2_i\ ]$ (map)

$\mid [[\ e1_i\ =\ e2_i\ ]]$ (big map)

$\mid [\ e_i\ ]$ (list)

$\mid \{\ e_i\ \}$ (set)

$\mid e(.ai)$ (accessor)

$\mid \mathrm{e1}[e2]$ (look up)

$\mid match\ e\ with\ matching$ (matching)

$\mid e1;\ e2$ (sequence)

$\mid while\ e1\ do\ e2$ (loop)

$\mid x.(a_i)\ =\ e$ (assign)

$\mid SKIP$ (skip)

$\mid e\ as\ T$ (ascription)

$access(a) =$

$\mid int$ (for tuples)

$\mid string$ (for record)

$\mid e$ (for map)

$value(v) =$

$\mid literal$ (values of built-in types)

$\mid const\ v$ (values of construct types)

$\mid \lambda x\ .\ expr$ (lambda)

$literal =$

$\mid unit$ ()

$\mid bool$ ()

$\mid int$ ()

$\mid nat$ ()

$\mid mutez$ ()

$\mid string$ ()

$\mid bytes$ ()

$\mid address$ ()

$\mid timestamp$ ()

$\mid operation$ ()

$matching(m) =$

$\mid \{\ true\ \ =>\ \ e;\ \ false\ \ =>\ \ e; \}$ (match bool)

$\mid \{\ nil\ \ =>\ \ e;\ \ cons(hd :: tl)\ \ =>\ \ e; \}$ (match list)

$\mid \{\ none\ \ =>\ \ e;\ \ some(x)\ \ =>\ \ e; \}$ (match option)

$\mid (x_i)\ \ =>\ \ e$ (match tuple)

$\mid (const_i(x_i)\ \ =>\ \ e_i\ )$ (match variant)

$matchingvalue(mv) =$

$\mid \{\ true\ \ =>\ \ v;\ \ false\ \ =>\ \ v; \}$ (match bool value)

$\mid \{\ nil\ \ =>\ \ v;\ \ cons(hd :: tl)\ \ =>\ \ v; \}$ (match list value)

$\mid \{\ none\ \ =>\ \ v;\ \ some(x)\ \ =>\ \ v; \}$ (match option value)

$\mid (x_i)\ \ =>\ \ v$ (match tuple value)

$\mid (const_i(x_i)\ \ =>\ \ v_i\ )$ (match variant value)

Evaluation of expression

base

Values and variables are not evaluted

$$\frac{}{built_i n\ (e_i)\ \rightarrow\ built_i n_r esult\ (*\ evaluated\ depending\ on\ each\ case\ *)} \qquad \text{(E-BUILTIN)}$$

$$\frac{}{(\lambda x.e)\ v\ \rightarrow\ [\ x\ \rightarrow\ v\ ]\ e} \qquad \text{(E-LAMBDA)}$$

$$\frac{e1 \ \rightarrow \ e1'}{e1\ e2 \ \rightarrow \ e1'\ e2} \qquad \text{(E-APP1)}$$

$$\frac{e2 \ \rightarrow \ e2'}{v1\ e2 \ \rightarrow \ v1\ e2'} \qquad \text{(E-APP2)}$$

$$\frac{e1 \ \rightarrow \ e1'}{let\ x = e1\ in\ e2 \ \rightarrow \ let\ x = e1'\ in\ e2} \qquad \text{(E-LET)}$$

$$\overline{let\ x = v1\ in\ e2 \ \rightarrow \ [x \rightarrow v1]\ e2} \qquad \text{(E-LETIN)}$$

$$\frac{e1 \ \rightarrow \ e1'}{e1;\ e2 \ \rightarrow \ e1';\ e2} \qquad \text{(E-SEQ)}$$

$$\overline{unit;\ e2 \ \rightarrow \ e2} \qquad \text{(E-SEQNEXT)}$$

$$\frac{e1 \ \rightarrow \ e1'}{while\ e1\ then\ e2 \ \rightarrow \ while\ e1'\ then\ e2} \qquad \text{(E-LOOP)}$$

$$\overline{while\ true(= e1)\ then\ e2 \ \rightarrow \ e2;\ while\ e1\ then\ e2} \qquad \text{(E-LOOPTRUE)}$$

$$\overline{while\ false\ then\ e2 \ \rightarrow \ unit} \qquad \text{(E-LOOPFALSE)}$$

$$\overline{SKIP \ \rightarrow \ unit} \qquad \text{(E-SKIP)}$$

$$\frac{e \ \rightarrow \ e'}{e\ as\ T \ \rightarrow \ e'\ as\ T} \qquad \text{(E-ASCR1)}$$

$$\overline{v\ as\ T \ \rightarrow \ v} \qquad \text{(E-ASCR2)}$$

data structure

$$\frac{e \ \rightarrow \ e'}{c\ e \ \rightarrow \ c\ e'} \qquad \text{(E-CONST)}$$

$$\frac{e_j \ \rightarrow \ e_j'}{(v_i,\ e_j,\ e_k) \ \rightarrow \ (v_i,\ e_j',\ e_k)} \qquad \text{(E-TUPLES)}$$

$$\frac{e_j \ \rightarrow \ e_j'}{\{l_i = v_i,\ l_j = e_j,\ l_k = e_k\} \ \rightarrow \ \{l_i = v_i,\ l_j = e_j',\ l_k = e_k\}} \qquad \text{(E-RECORDS)}$$

$$\frac{e2_j \ \rightarrow \ e2_j'}{[e1_i = v_i,\ e1_j = e2_j,\ e1_k = e2_k] \ \rightarrow \ [e1_i = v_i,\ e1_j = e2_j',\ e1_k = e2_k]} \qquad \text{(E-MAP)}$$

$$\frac{e2_j \ \rightarrow \ e2_j'}{[[e1_i = v_i,\ e1_j = e2_j,\ e1_k = e2_k]] \ \rightarrow \ [[e1_i = v_i,\ e1_j = e2_j',\ e1_k = e2_k]]} \qquad \text{(E-BIGMAP)}$$

$$\frac{e_j \ \rightarrow \ e_j'}{[v_i,\ e_j,\ e_k] \ \rightarrow \ [v_i,\ e_j',\ e_k]} \qquad \text{(E-LIST)}$$

$$\frac{e_j \ \rightarrow \ e_j'}{\{v_i,\ e_j,\ e_k\} \ \rightarrow \ \{v_i,\ e_j',\ e_k\}} \qquad \text{(E-SET)}$$

$$\frac{e \ \rightarrow \ e'}{e(.a_i) \ \rightarrow \ e'(.a_i)} \qquad \text{(E-ACCESS)}$$

look up

$$\overline{(v_i)[j] \ \rightarrow \ v_j} \qquad \text{(E-LUPTUPLE)}$$

$$\overline{\{l_i = v_i\}[lj] \ \rightarrow \ v_j} \qquad \text{(E-LUPRECORD)}$$

$$\overline{[e_i = v_i][ej] \ \rightarrow \ v_j} \qquad \text{(E-LUPMAP)}$$

$$\overline{[[e_i = v_i]][ej] \ \rightarrow \ v_j} \qquad \text{(E-LUPBIGMAP)}$$

$$\overline{[v_i][j] \ \rightarrow \ v_j} \qquad \text{(E-LUPLIST)}$$

$$\overline{\{v_i\}[j] \ \rightarrow \ v_j} \qquad \text{(E-LUPSET)}$$

$$\frac{e \ \rightarrow \ e'}{x(.a_i) \ = \ e \ \rightarrow \ x(.a_i) \ = \ e'} \qquad \text{(E-ASSIGN)}$$

$$\overline{x(.a_i) \ = \ v \ \rightarrow \ x'(.a_i)\ with\ x'\ as\ x\ with\ field\ (.a_i)\ replace\ by\ v} \qquad \text{(E-ASSIGN2)}$$

matching

$$\frac{e \ \to \ e'}{match\ e\ with\ m \ \to \ match\ e'\ with\ m} \qquad \text{(E-MATCH1)}$$

$$\frac{m \ \to \ m'}{match\ v\ with\ m \ \to \ match\ v\ with\ m'} \qquad \text{(E-MATCH2)}$$

$$\frac{}{match\ v\ with\ mv \ \to \ v'\ if\ v \ => \ v'\ in\ mv} \qquad \text{(E-MATCH)}$$

$$\frac{e1 \ \to \ e1'}{\{\ true \ => \ e1;\ false \ => \ e2;\} \ \to \ \{\ true \ => \ e1';\ false \ => \ e2;\}} \qquad \text{(E-MAcTHBOOL1)}$$

$$\frac{e2 \ \to \ e2'}{\{\ true \ => \ v1;\ false \ => \ e2;\} \ \to \ \{\ true \ => \ v1;\ false \ => \ e2';\}} \qquad \text{(E-MAcTHBOOL2)}$$

$$\frac{e1 \ \to \ e1'}{\{\ nil \ => \ e1;\ cons(hd :: tl) \ => \ e2;\} \ \to \ \{\ nil \ => \ e1';\ cons(hd :: tl) \ => \ e2;\}} \qquad \text{(E-MATCHLIST1)}$$

$$\frac{e2 \ \to \ e2'}{\{\ nil \ => \ v1;\ cons(hd :: tl) \ => \ e2;\} \ \to \ \{\ nil \ => \ v1;\ cons(hd :: tl) \ => \ e2';\}} \qquad \text{(E-MATCHLIST2)}$$

$$\frac{e1 \ \to \ e1'}{\{\ none \ => \ e1;\ some(x) \ => \ e2;\} \ \to \ \{\ none \ => \ e1';\ some(x) \ => \ e2;\ \}} \qquad \text{(E-MATCHOPT1)}$$

$$\frac{e2 \ \to \ e2'}{\{\ none \ => \ v1;\ some(x) \ => \ e2;\} \ \to \ \{\ none \ => \ v1';\ some(x) \ => \ e2';\ \}} \qquad \text{(E-MATCHOPT2)}$$

$$\frac{e \ \to \ e'}{(x_i) \ => \ e \ \to \ (x_i) \ => \ e'} \qquad \text{(E-MATCHTUPLE)}$$

$$\frac{e_j \ \to \ e_j'}{(c_i(x_i) \ => \ v_i,\ c_j(x_j) \ => \ e_j,\ c_k(x_k) \ => \ e_k) \ \to \ (c_i(x_i) \ => \ v_i,\ c_j(x_j) => e_j',\ c_k(x_k) => e_k)}$$

(E-MATCHVARIANT)

Derive form

$$e1;\ e2\ is\ (\lambda x : Unit.e1)\ e2\ with\ x\ not\ a\ free\ variable\ in\ e1 \tag{1}$$

$$let\ x = e1\ in\ e2\ is\ (\lambda x : T1.e2)\ e1 \tag{2}$$