Ligo Formal Description

Syntax

variables (x)

label (l)

constructor (c)

declaration (d) $=$

$\mid$ *type* x *is* te	(Type declaration)
$\mid$ *const* x $(:\ te)\ =\ e$	(Constant declaration)

type expression (te) $=$

$\mid (* \ te_i)$	(type of tuple)
$\mid (\mid l_i \ of \ te_i)$	(type of sum)
$\mid \{\ l_i\ :\ te_i\}$	(type of record)
$\mid te1\ \rightarrow\ te2$	(type of function)
$\mid l$	(type of 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)
$\mid e1[e2]$	(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)

$matching\ value\ (mv)\ =$

$\mid\{\ true\ =>\ v;\ false\ =>\ v;\}$	(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 are not evaluted

$x\ \rightarrow\ v$ (*corresponding value in the environment*)	(E-VARIABLE)
built in $(e_i)\ \rightarrow$ *built in result* ($*$ *evaluated depending on each case* $*$)	(E-BUILTIN)
$(\lambda x.e)\ v\ \rightarrow\ [\ x\ \rightarrow\ v\ ]\ e$	(E-LAMBDA)
$\dfrac{e1\ \rightarrow\ e1'}{e1\ e2\ \rightarrow\ e1'\ e2}$	(E-APP1)
$\dfrac{e2\ \rightarrow\ e2'}{v1\ e2\ \rightarrow\ v1\ e2'}$	(E-APP2)

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

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

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

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

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

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

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

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

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

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

data structure

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

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

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

$$\frac{e2_j \ \rightarrow \ e2'_j}{[e1_i = v_i,\ e1_j = e2_j,\ e1_k = e2_k] \ \rightarrow \ [e1_i = v_i,\ e1_j = e2'_j,\ e1_k = e2_k]} \quad \text{(E-MAP)}$$

$$\frac{e2_j \ \rightarrow \ e2'_j}{[[e1_i = v_i,\ e1_j = e2_j,\ e1_k = e2_k]] \ \rightarrow \ [[e1_i = v_i,\ e1_j = e2'_j,\ e1_k = e2_k]]} \quad \text{(E-BIGMAP)}$$

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

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

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

look up

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

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

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

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

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

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

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

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

matching

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

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

$$match\ v\ with\ mv \ \rightarrow \ v'\ if\ v \ => \ v'\ in\ mv \quad (\text{E-MATCH})$$

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

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

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

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

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

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

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

$$\frac{e_j \ \rightarrow \ e'_j}{(c_i(x_i) \ => \ v_i,\ c_j(x_j) \ => \ e_j,\ c_k(x_k) \ => \ e_k) \ \rightarrow \ (c_i(x_i) \ => \ v_i,\ c_j(x_j) => e'_j,\ c_k(x_k) => e_k)} \quad (\text{E-MATCHVARIANT})$$

Derive form

$e1;\ e2 \iff (\lambda x : Unit.e1)\ e2\ with\ x\ not\ a\ free\ variable\ in\ e1$

$let\ x = e1\ in\ e2 \iff (\lambda x : T1.e2)\ e1$