Semantics for picoELLA

K. G. W. Goossens
Laboratory for foundations of computer science
Department of computer science
University of Edinburgh
The King's buildings
Edinburgh EH9 3JZ
Scotland, U.K.

June 1990

1 Preliminary Definitions

$$\begin{aligned} Env &= Name \rightarrow Value \\ BType &= \mathcal{F}(Cname) \\ Type &= Tname + BType + Type \times Type \\ TEnv &= (Name + CName) \rightarrow Type \\ TEnv' &= Tname \rightarrow Type \end{aligned}$$

We have $E \in Env; T \in TEnv; S \in TEnv'; \tau \in Type$. ϕ denotes the empty environment (for $TEnv$, $TEnv'$ and Env). $Dom : TEnv \rightarrow (Name + Cname)$ and $Dom : TEnv' \rightarrow Tname$ give the domain of a type environment. Similarly *Ran* delivers the range of $TEnv$ and $TEnv'$.

match is the formalisation of the matching function used in the operational semantics (rule 40). The arguments to *match* must be well-typed. c, c_1, c_2 are *const*s and ch_1 and ch_2 are *choose*rs (both defined in the next section).

$$\begin{aligned} match(c, ch_1|ch_2) &= match(c, ch_1) \vee match(c, ch_2) \\ match((c_1, c_2), (ch_1, ch_2)) &= match(c_1, ch_1) \wedge match(c_2, ch_2) \\ match(cname, cname') &= \begin{cases} tt & \text{if } cname \equiv cname' \\ ff & \text{otherwise} \end{cases} \\ match(?tname, cname) &= \perp_{Bool} \\ match(c, tname) &= tt \end{aligned}$$

$valueof : const \rightarrow Value$ is a bijection mapping a constant to a *value*. It is used to switch between a value and its syntactic representation.

2 Syntax

$$
\begin{array}{rcl}
program & ::= & \texttt{INPUT}\ name : tname\ \texttt{IN}\ expr \parallel \\
 & & \texttt{TYPE}\ tdecl\ \texttt{IN}\ program \\
tdecl & ::= & tname = type \\
type & ::= & btype \parallel ttype \\
btype & ::= & cname \parallel btype|btype \\
ttype & ::= & tname \parallel ttype * ttype \\
decl & ::= & \texttt{REC}\ name = expr \parallel name = expr \\
expr & ::= & const \parallel name \parallel \\
 & & expr[int] \parallel (expr, expr) \parallel \\
 & & \texttt{DELAY}\ (expr, const) \parallel \\
 & & \texttt{IF}\ expr\ \texttt{MATCHES}\ chooser\ \texttt{THEN}\ expr\ \texttt{ELSE}\ expr \parallel \\
 & & \texttt{LET}\ decl\ \texttt{IN}\ expr \\
chooser & ::= & tname \parallel cname \parallel (chooser, chooser) \parallel chooser|chooser \\
const & ::= & ?tname \parallel cname \parallel (const, const)
\end{array}
$$

3 Static semantics

3.1 Type Equivalence

$$\frac{}{S, T \vdash\ tname \simeq\ S(tname)} \qquad tname \in Dom(S) \tag{1}$$

$$\frac{}{S, T \vdash\ \tau \simeq\ \tau} \tag{2}$$

$$\frac{S, T \vdash\ \tau \simeq\ \tau'}{S, T \vdash\ \tau' \simeq\ \tau} \tag{3}$$

$$\frac{S, T \vdash\ \tau \simeq\ \tau' \qquad S, T \vdash\ \tau' \simeq\ \tau''}{S, T \vdash\ \tau \simeq\ \tau''} \tag{4}$$

$$\frac{S, T \vdash\ \tau \simeq\ \tau' \qquad S, T \vdash\ \tau'' \simeq\ \tau'''}{S, T \vdash\ (\tau, \tau'') \simeq\ (\tau', \tau''')} \tag{5}$$

$$\frac{S, T \vdash\ expr :\ \tau \qquad S, T \vdash\ \tau \simeq\ \tau'}{S, T \vdash\ expr :\ \tau'} \tag{6}$$

3.2 Type Definitions

$$\frac{S, T, tname \vdash\ type :\ T'}{S, T \vdash\ tname = type :\ S\{(tname, s)\}, T'} \qquad tname \notin Dom(S) \tag{7}$$

Where $s \equiv ttype$ if $type \equiv ttype$, and $s \equiv \{cname_1, \ldots, cname_N\}$ if $type \equiv cname_1|\ldots|cname_N$.

$$\frac{}{S, T, tname \vdash\ ttype :\ T} \qquad \text{All } tnames \text{ in } ttype \text{ must be in } Dom(S). \tag{8}$$

$$\frac{}{S, T, tname \vdash\ cname :\ T\{(cname, tname)\}} \qquad cname \notin Dom(T) \tag{9}$$

$$\frac{S, T, tname \vdash\ btype :\ T' \qquad S, T', tname \vdash\ btype' :\ T''}{S, T, tname \vdash\ btype|btype' :\ T''} \tag{10}$$

$$\frac{S, T \vdash\ expr :\ \tau}{S, T \vdash\ name = expr :\ T\{(name, \tau)\}} \tag{11}$$

$$\frac{S, T\{(name, \tau)\} \vdash\ expr :\ \tau}{S, T \vdash\ \texttt{REC}\ name = expr :\ T\{(name, \tau)\}} \qquad \tau \text{ corresponds to a } ttype \tag{12}$$

3.3 Programs

$$\frac{S, T\{(name, tname)\} \vdash\ expr :\ \tau}{S, T \vdash\ \texttt{INPUT}\ name : tname\ \texttt{IN}\ expr :\ \tau} \qquad tname \in Dom(S) \tag{13}$$

$$\frac{S, T \vdash\ tdecl :\ S', T' \qquad T', S' \vdash\ program :\ \tau}{S, T \vdash\ \texttt{TYPE}\ tdecl\ \texttt{IN}\ program :\ \tau} \tag{14}$$

$$\frac{}{S, T \vdash\ name \Rightarrow\ T(name)} \qquad name \in Dom(T) \tag{15}$$

$$\frac{S, T \vdash\ expr :\ \tau_1 * \tau_2}{S, T \vdash\ expr[i] :\ \tau_i} \qquad i = 1, 2 \tag{16}$$

$$\frac{S, T \vdash\ expr :\ \tau \qquad S, T \vdash\ expr' :\ \tau'}{S, T \vdash\ (expr, expr') :\ \tau * \tau'} \tag{17}$$

$$\frac{S, T \vdash\ expr :\ \tau \qquad S, T \vdash\ const :\ \tau}{S, T \vdash\ \texttt{DELAY}\ (expr, const) :\ \tau} \tag{18}$$

$$\frac{S, T \vdash\ expr :\ \tau \qquad S, T \vdash\ chooser :\ \tau \qquad S, T \vdash\ expr' :\ \tau' \qquad S, T \vdash\ expr'' :\ \tau'}{S, T \vdash\ \texttt{IF}\ expr\ \texttt{MATCHES}\ chooser\ \texttt{THEN}\ expr'\ \texttt{ELSE}\ expr'' :\ \tau'} \tag{19}$$

$$\frac{S, T \vdash\ decl :\ T' \qquad S, T' \vdash\ expr :\ \tau}{S, T \vdash\ \texttt{LET}\ decl\ \texttt{IN}\ expr :\ \tau} \tag{20}$$

$$\frac{}{S, T \vdash\ cname :\ T(cname)} \qquad cname \in Dom(T) \tag{21}$$

$$\frac{}{S, T \vdash\ ?tname :\ tname} \qquad tname \in Dom(S) \tag{22}$$

$$\frac{S, T \vdash\ const :\ \tau \qquad S, T \vdash\ const' :\ \tau'}{S, T \vdash\ (const, const') :\ \tau * \tau'} \tag{23}$$

$$\frac{}{S, T \vdash\ tname :\ tname} \qquad tname \in dom(S) \tag{24}$$

$$\frac{S, T \vdash\ chooser :\ \tau \qquad S, T \vdash\ chooser' :\ \tau}{S, T \vdash\ chooser | chooser' :\ \tau} \tag{25}$$

$$\frac{S, T \vdash\ chooser : \tau \qquad S, T \vdash\ chooser' : \tau'}{S, T \vdash\ (chooser, chooser') : \tau * \tau'} \tag{26}$$

4 Operational semantics

$$\frac{}{E \vdash\ program,\ nil \Rightarrow\ program,\ nil} \tag{27}$$

$$\frac{E \vdash\ program,\ h \Rightarrow\ p',\ h' \qquad E \vdash\ p',\ t \Rightarrow\ p'',\ t'}{E \vdash\ program,\ h :: t \Rightarrow\ p'', h' :: t'} \tag{28}$$

$$\frac{E\{(name, h)\} \vdash\ expr \Rightarrow\ expr',\ v}{E \vdash\ \texttt{INPUT}\ name : tname\ \texttt{IN}\ expr,\ h \Rightarrow\ \texttt{INPUT}\ name : tname\ \texttt{IN}\ expr',\ v} \tag{29}$$

$$\frac{E \vdash\ program,\ h \Rightarrow\ program',\ v}{E \vdash\ \texttt{TYPE}\ typedecl\ \texttt{IN}\ program,\ h \Rightarrow\ program', v} \tag{30}$$

$$\frac{E \vdash\ expr \Rightarrow\ expr', v}{E \vdash\ name = expr \Rightarrow\ name = expr', E\{(name, v)\}} \tag{31}$$

$$\frac{E, (name, ?type) \vdash\ \texttt{REC}\ name = expr \Rightarrow\ expr',\ v}{E \vdash\ \texttt{REC}\ name = expr \Rightarrow\ expr',\ E\{(name, v)\}} \qquad type \text{ is the type of } expr \tag{32}$$

$$\frac{E\{(name, v)\} \vdash\ expr \Rightarrow\ expr',\ v}{E, (name, v) \vdash\ \texttt{REC}\ name = expr \Rightarrow\ \texttt{REC}\ name = expr',\ v} \tag{33}$$

$$\frac{E\{(name, v)\} \vdash\ expr \Rightarrow\ expr', v' \qquad E, (name, v') \vdash\ \texttt{REC}\ name = expr \Rightarrow\ expr'',\ v''}{E, (name, v) \vdash\ \texttt{REC}\ name = expr \Rightarrow\ expr'',\ v''} \tag{34}$$

$$\frac{}{E \vdash\ name \Rightarrow\ name,\ E(name)} \tag{35}$$

$$\frac{E \vdash\ expr \Rightarrow\ (expr_1, expr_2),\ (v_1, v_2)}{E \vdash\ expr[i] \Rightarrow\ expr_i,\ v_i} \qquad i = 1, 2 \tag{36}$$

$$\frac{E \vdash\ expr_1 \Rightarrow\ expr_1', v_1 \qquad E \vdash\ expr_2 \Rightarrow\ expr_2', v_2}{E \vdash\ (expr_1, expr_2) \Rightarrow\ (expr_1', expr_2'),\ (v_1, v_2)} \tag{37}$$

$$\frac{E \vdash\ decl \Rightarrow\ decl', E' \qquad E' \vdash\ expr \Rightarrow\ expr', v}{E \vdash\ \texttt{LET}\ decl\ \texttt{IN}\ expr \Rightarrow\ \texttt{LET}\ decl'\ \texttt{IN}\ expr', v} \tag{38}$$

$$\frac{E \vdash\ expr \Rightarrow\ expr',\ v'}{E \vdash\ \texttt{DELAY}\ (expr,\ c) \Rightarrow\ \texttt{DELAY}\ (expr',\ valueof^{-1}(v')),\ valueof(c)} \tag{39}$$

$$\frac{E \vdash\ expr_0 \Rightarrow\ expr_0',\ v_0 \qquad E \vdash\ expr_1 \Rightarrow\ expr_1',\ v_1 \qquad E \vdash\ expr_2 \Rightarrow\ expr_2',\ v_2}{\begin{array}{l} E \vdash\ \texttt{IF}\ expr_0\ \texttt{MATCHES}\ chooser\ \texttt{THEN}\ expr_1\ \texttt{ELSE}\ expr_2 \\ \Rightarrow\ \texttt{IF}\ expr_0'\ \texttt{MATCHES}\ chooser\ \texttt{THEN}\ expr_1'\ \texttt{ELSE}\ expr_2',\ v \end{array}} \tag{40}$$

Where $type$ is the type of $expr_1$ and $expr_2$ in:

$$v \equiv \begin{cases} v_1 & match(v_0,\ chooser) = tt \\ v_2 & match(v_0,\ chooser) = ff \\ ?type & match(v_0,\ chooser) = \bot_{Bool} \end{cases}$$

$$\frac{}{E \vdash\ cname \Rightarrow\ cname,\ valueof(cname)} \tag{41}$$

$$\frac{E \vdash\ c_1 \Rightarrow\ c_1',\ v_1 \qquad E \vdash\ c_2 \Rightarrow\ c_2',\ v_2}{E \vdash\ (c_1, c_2) \Rightarrow\ (c_1', c_2'),\ (v_1, v_2)} \tag{42}$$

$$\frac{}{E \vdash ?tname \Rightarrow\ (?ttype_1, ?ttype_2),\ (?ttype_1, ?ttype_2)} \tag{43}$$

The type of $tname$ is $ttype_1 * ttype_2$.

$$\frac{}{E \vdash ?tname \Rightarrow ?tname,\ valueof(?tname)} \tag{44}$$

5 Examples

Example 1

```
TYPE foo = bar IN
        TYPE bar = foo IN
                INPUT wiz : foo IN
                        ( wiz, foo, bar, ?foo, ?bar )
```

This program illustrates that constructors and types may hide each other, and has type $foo * bar * foo * foo * bar$.

Example 2

```
TYPE foo = foo IN
        TYPE bar = bar IN
                INPUT in : foo IN
                        LET REC n =
                                LET REC n = n IN n
                        IN n
```

This program has type τ where τ corresponds to any $ttype$ (i.e. any tuple) that may be constructed from foo and bar. Thus τ may be foo, bar, (foo, foo), $((bar, foo), (foo, bar))$,... The program evaluates to the undefined value corresponding to the type: $?foo$, $?bar$, $(?foo, ?foo)$, $((?bar, ?foo), (?foo, ?bar))$,... Note that the free type variable τ originates in the inner `LET REC` and there is no choice for the type of the outer `LET REC`. Replacing the last occurrence of n by $n[2][2]$ would dissallow the first three types listed above.

Example 3

```
TYPE bool = true | false IN
        TYPE twobool = bool * bool IN
                INPUT in : twobool IN
                        IF in MATCHES twobool | (bool, true)
                        THEN ?twobool[1]
                        ELSE (?bool, in[2])[1]
```

This program has type $bool$ and evaluates to $?bool$. The derivation of the type depends on the expansion (once in $chooser$ and once in the type of $?twobool$) of $twobool$ to $bool * bool$. The evaluation depends on expanding $?twobool$ to $(?bool, ?bool)$ to allow indexing.