A. Component-Based Semantics of Caml Light

A.1 Global names

$$\mathit{id}\,[\![GN]\!] = \textbf{id}('GN') \tag{1}$$

$$\mathit{id}\,[\![\texttt{prefix}\,ON]\!] = \textbf{id}('\text{prefix}\,ON') \tag{2}$$

$$\mathit{typeid}\,[\![LI]\!] = \textbf{typeid}('LI') \tag{3}$$

A.2 Constants

$$\mathit{value}\,[\![Int]\!] = Int \tag{4}$$

$$\mathit{value}\,[\![Float]\!] = Float \tag{5}$$

$$\mathit{value}\,[\![Char]\!] = \textbf{char}('CH') \tag{6}$$

$$\mathit{value}\,[\![String]\!] = String \tag{7}$$

$$\mathit{value}\,[\![K]\!] = \textbf{fold-poly}(\textbf{variant}('K', \textbf{tuple-emmpty})) \tag{8}$$

$$\mathit{value}\,[\![\texttt{false}]\!] = \textbf{false} \tag{9}$$

$$\mathit{value}\,[\![\texttt{true}]\!] = \textbf{true} \tag{10}$$

$$\mathit{value}\,[\![\texttt{[]}]\!] = \textbf{fold-poly}(\textbf{variant}('[]', \textbf{tuple-empty})) \tag{11}$$

$$\mathit{value}\,[\![\texttt{()}]\!] = \textbf{tuple-empty} \tag{12}$$

A.3 Patterns

$$\mathit{patt}\,[\![LI]\!] = \textbf{bind}(\mathit{id}\,[\![LI]\!]) \tag{13}$$

$$\mathit{patt}\,[\![\_]\!] = \textbf{any} \tag{14}$$

$$\mathit{patt}\,[\![P\ \texttt{as}\ LI]\!] = \textbf{patt-union}(\mathit{patt}\,[\![P]\!], \textbf{bind}(\mathit{id}\,[\![LI]\!])) \tag{15}$$

$$\mathit{patt}\,[\![\texttt{(}\,P\,\texttt{)}]\!] = \mathit{patt}\,[\![P]\!] \tag{16}$$

$$\mathit{patt}\,[\![\texttt{(}\,P : T\,\texttt{)}]\!] = \textbf{patt-at-type}(\mathit{patt}\,[\![P]\!], \mathit{type}\,[\![T]\!]) \tag{17}$$

$$\mathit{patt}\,[\![P1 \mid P2]\!] = \textbf{prefer-over}(\mathit{patt}\,[\![P1]\!], \mathit{patt}\,[\![P2]\!]) \tag{18}$$

$$\mathit{patt}\,[\![K]\!] = \textbf{unfold-variant-select}('K', \textbf{only}(\textbf{tuple-empty})) \tag{19}$$

$$\mathit{patt}\,[\![\texttt{[]}]\!] = \textbf{unfold-variant-select}('[]', \textbf{only}(\textbf{tuple-empty})) \tag{20}$$

$$\mathit{patt}\,[\![C]\!] = \textbf{only}() \tag{21}$$

$$\mathit{patt}\,[\![KP]\!] = \textbf{unfold-variant-select}('K', \mathit{patt}\,[\![P]\!]) \tag{22}$$

$$\mathit{patt}\,[\![P1\ \texttt{,}\ P2...]\!] = \mathit{patt\text{-}tuple}\,[\![P1, P2...]\!] \tag{23}$$

$$\mathit{patt\text{-}tuple}\,[\![P1]\!] = \textbf{invert tuple-prefix}(\mathit{patt}\,[\![P1]\!], \textbf{only}(\textbf{tuple-empty})) \tag{24}$$

$$\mathit{patt\text{-}tuple}\,[\![P1\ \texttt{,}\ P2...]\!] = \textbf{invert tuple-prefix}(pattP1, \mathit{patt\text{-}tuple}\,[\![P2...]\!]) \tag{25}$$

$$\mathit{patt}\,[\![\{\ L1\ \texttt{=}\ P1...\ \}]\!] = \textbf{loose record}(\mathit{patt\text{-}map}\,[\![L1\ \texttt{=}\ P1...]\!]) \tag{26}$$

$$\mathit{patt}\,[\![\{\ L1\ \texttt{=}\ P1...\ \texttt{;}\ \}]\!] = \mathit{patt}\,[\![\{\ L1\ \texttt{=}\ P1...\ \}]\!] \tag{27}$$

$$\mathit{patt\text{-}map}\,[\![L1\ \texttt{=}\ P1]\!] = \textbf{map1}('L1', \textbf{compose}(\mathit{patt}\,[\![P1]\!], \textbf{abs assigned-value})) \tag{28}$$

$$\mathit{patt\text{-}map}\,[\![L1\ \texttt{=}\ P1\ \texttt{;}\ L2\ \texttt{=}\ P2...]\!] = \textbf{map-union}(\mathit{patt\text{-}map}\,[\![L1\ \texttt{=}\ P1]\!], \mathit{patt\text{-}map}\,[\![L2\ \texttt{=}\ P2...]\!]) \tag{29}$$

$$\mathit{patt}\,[\![\texttt{[}\,P1\,\texttt{]}]\!] = \mathit{patt}\,[\![P1\ \texttt{::}\ \texttt{[]}]\!] \tag{30}$$

$$\mathit{patt}\,[\![\texttt{[}\,P1\ \texttt{;}\ P2...\,\texttt{]}]\!] = \mathit{patt}\,[\![P1\ \texttt{::}\ \texttt{[}\,P2...\,\texttt{]}]\!] \tag{31}$$

$$\mathit{patt}\,[\![\texttt{[}\,P1...\ \texttt{;}\,\texttt{]}]\!] = \mathit{patt}\,[\![\texttt{[}\,P1...\,\texttt{]}]\!] \tag{32}$$

$$\mathit{patt}\,[\![P1\ \texttt{::}\ P2]\!] = \textbf{unfold-variant-select}('\texttt{::}', \textbf{invert tuple2}(\mathit{patt}\,[\![P1]\!], \mathit{patt}\,[\![P2]\!])) \tag{33}$$

A.4 Expressions

$$\mathit{expr}\,[\![V]\!] = \textbf{instantiate-if-poly}(\textbf{bound-value}(\mathit{id}\,[\![V]\!])) \tag{34}$$

$$\mathit{expr}\,[\![C]\!] = \mathit{value}\,[\![C]\!] \tag{35}$$

$$\mathit{expr}\,[\![\texttt{(}\,E\,\texttt{)}]\!] = \mathit{expr}\,[\![E]\!] \tag{36}$$

$$\mathit{expr}\,[\![\texttt{begin}\,E\,\texttt{end}]\!] = \mathit{expr}\,[\![E]\!] \tag{37}$$

$$\mathit{expr}\,[\![\texttt{(}\,E : T\,\texttt{)}]\!] = \textbf{typed}(\mathit{expr}\,[\![E]\!], \mathit{type}\,[\![T]\!]) \tag{38}$$

$$\mathit{expr}\,[\![E1\ \texttt{,}\ E2...]\!] = \mathit{expr\text{-}tuple}\,[\![E1, E2...]\!] \tag{39}$$

$$\mathit{expr\text{-}tuple}\,[\![E1]\!] = \textbf{tuple-prefix}(\mathit{expr}\,[\![E1]\!], \textbf{tuple-empty}) \tag{40}$$

$$\mathit{expr\text{-}tuple}\,[\![E1\ \texttt{,}\ E2...]\!] = \textbf{tuple-prefix}(\mathit{expr}\,[\![E1]\!], \mathit{expr\text{-}tuple}\,[\![E2...]\!]) \tag{41}$$

$$\mathit{expr}\,[\![KE]\!] = \textbf{fold-poly}(\textbf{variant}('K', \mathit{expr}\,[\![E]\!])) \tag{42}$$

$$\mathit{expr}\,[\![E1\ \texttt{::}\ E2]\!] = \textbf{fold-poly}(\textbf{variant}('\texttt{::}', \textbf{tuple2}(\mathit{expr}\,[\![E1]\!], \mathit{expr}\,[\![E2]\!]))) \tag{43}$$

$$\mathit{expr}\,[\![\texttt{[}\,E1\,\texttt{]}]\!] = \mathit{expr}\,[\![E1\ \texttt{::}\ \texttt{[}\ \texttt{]}]\!] \tag{44}$$

$$\mathit{expr}\,[\![\texttt{[}\,E1\ \texttt{;}\ E2...\,\texttt{]}]\!] = \mathit{expr}\,[\![E1\ \texttt{::}\ \texttt{[}\,E2...\,\texttt{]}]\!] \tag{45}$$

$$\mathit{expr}[\![\texttt{[}\ E1...\ \texttt{;}\ \texttt{]}]\!] = \quad (46)$$
$$\quad \mathit{expr}[\![\texttt{[}\ E1...\texttt{]}]\!]$$

$$\mathit{expr}[\![\texttt{[|}\ \texttt{|]}]\!] = \quad (47)$$
$$\quad \textbf{array-empty}$$

$$\mathit{expr}[\![\texttt{[|}\ E1\ \texttt{|]}]\!] = \quad (48)$$
$$\quad \textbf{array1}(\textbf{alloc}(\mathit{expr}[\![E1]\!]))$$

$$\mathit{expr}[\![\texttt{[|}\ E1\ \texttt{;}\ E2...\ \texttt{|]}]\!] = \quad (49)$$
$$\quad \textbf{array-append}(\mathit{expr}[\![\texttt{[|}\ E1\ \texttt{|]}]\!], \mathit{expr}[\![\texttt{[|}\ E2...\ \texttt{|]}]\!])$$

$$\mathit{expr}[\![\texttt{[|}|E1\ \texttt{;}\ E2...\ \texttt{;}\ \texttt{|]}]\!] = \quad (50)$$
$$\quad \mathit{expr}[\![\texttt{[|}\ E1\ \texttt{;}\ E2...\ \texttt{|]}]\!]$$

$$\mathit{expr}[\![\{\ L1 = E1\ \}]\!] = \quad (51)$$
$$\quad \textbf{record1}('L1', \textbf{alloc}(\mathit{expr}[\![E1]\!]))$$

$$\mathit{expr}[\![\{\ L1 = E1\ \texttt{;}\ L2 = E2...\ \}]\!] = \quad (52)$$
$$\quad \textbf{record-union}(\mathit{expr}[\![\{\ L1 = E1\ \}]\!], \mathit{expr}[\![\{\ L2 = E2...\ \}]\!])$$

$$\mathit{expr}[\![L1 = E1...\ \texttt{;}]\!] = \quad (53)$$
$$\quad \mathit{expr}[\![L1 = E1...]\!]$$

$$\mathit{expr}[\![E1E2]\!] = \quad (54)$$
$$\quad \textbf{apply}(\mathit{expr}[\![E1]\!], \mathit{expr}[\![E2]\!])$$

$$\mathit{expr}[\![\texttt{-}\ E1]\!] = \quad (55)$$
$$\quad \mathit{expr}[\![\texttt{minus}\ E1]\!]$$

$$\mathit{expr}[\![\texttt{-.}\ E1]\!] = \quad (56)$$
$$\quad \mathit{expr}[\![\texttt{minus-float}\ E1]\!]$$

$$\mathit{expr}[\![\texttt{!}\ E1]\!] = \quad (57)$$
$$\quad \mathit{expr}[\![\texttt{prefix}\ \texttt{!}\ E1]\!]$$

$$\mathit{expr}[\![E1IOE2]\!] = \quad (58)$$
$$\quad \mathit{expr}[\![\texttt{prefix}\ IOE1E2]\!]$$

$$\mathit{expr}[\![E1\ \texttt{.}\ L]\!] = \quad (59)$$
$$\quad \textbf{assigned-value}(\textbf{record-select}(\mathit{expr}[\![E1]\!], 'L'))$$

$$\mathit{expr}[\![E1\ \texttt{.}\ L\ \texttt{<-}\ E2]\!] = \quad (60)$$
$$\quad \textbf{assign}(\textbf{record-select}(\mathit{expr}[\![E1]\!], 'L'), \mathit{expr}[\![E2]\!])$$

$$\mathit{expr}[\![E1\ \texttt{.(}\ E2\ \texttt{)}]\!] = \quad (61)$$
$$\quad \mathit{expr}[\![\texttt{vect\_item}\ E1E2]\!]$$

$$\mathit{expr}[\![E1\ \texttt{.(}\ E2\ \texttt{)}\ \texttt{<-}\ E3]\!] = \quad (62)$$
$$\quad \mathit{expr}[\![\texttt{vect\_assign}\ E1E2E3]\!]$$

$$\mathit{expr}[\![\texttt{not}\ E1]\!] = \quad (63)$$
$$\quad \textbf{not}(\mathit{expr}[\![E1]\!])$$

$$\mathit{expr}[\![E1\ \texttt{\&}\ E2]\!] = \quad (64)$$
$$\quad \textbf{if-true}(\mathit{expr}[\![E1]\!], \mathit{expr}[\![E2]\!], \textbf{false})$$

$$\mathit{expr}[\![E1\ \texttt{\&\&}\ E2]\!] = \quad (65)$$
$$\quad \mathit{expr}[\![E1\ \texttt{\&}\ E2]\!]$$

$$\mathit{expr}[\![E1\ \texttt{or}\ E2]\!] = \quad (66)$$
$$\quad \textbf{if-true}(\mathit{expr}[\![E1]\!], \textbf{true}, \mathit{expr}[\![E2]\!])$$

$$\mathit{expr}[\![E1\ \texttt{||}\ E2]\!] = \quad (67)$$
$$\quad \mathit{expr}[\![E1\ \texttt{or}\ E2]\!]$$

$$\mathit{expr}[\![\texttt{if}\ E1\ \texttt{then}\ E2]\!] = \quad (68)$$
$$\quad \mathit{expr}[\![\texttt{if}\ E1\ \texttt{then}\ E2\ \texttt{else ( )}]\!]$$

$$\mathit{expr}[\![\texttt{if}\ E1\ \texttt{then}\ E2\ \texttt{else}\ E3]\!] = \quad (69)$$
$$\quad \textbf{if-true}(\mathit{expr}[\![E1]\!], \mathit{expr}[\![E2]\!], \mathit{expr}[\![E3]\!])$$

$$\mathit{expr}[\![\texttt{while}\ E1\ \texttt{do}\ E2\ \texttt{done}]\!] = \quad (70)$$
$$\quad \textbf{seq}(\textbf{while-true}(\mathit{expr}[\![E1]\!], \textbf{effect}(\mathit{expr}[\![E2]\!])), \textbf{tuple-empty})$$

$$\mathit{expr}[\![\texttt{for}\ LI\ \texttt{=}\ E1\ \texttt{to}\ E2\ \texttt{do}\ E3\ \texttt{done}]\!] = \quad (71)$$
$$\quad \textbf{apply-to-each}(\textbf{abs}(\textbf{bind}(\mathit{id}[\![LI]\!]), \textbf{effect}(\mathit{expr}[\![E3]\!])),$$
$$\qquad \textbf{int-closed-interval}(\mathit{expr}[\![E1]\!], \mathit{expr}[\![E2]\!]))$$

$$\mathit{expr}[\![\texttt{for}\ LI\ \texttt{=}\ E1\ \texttt{downto}\ E2\ \texttt{do}\ E3\ \texttt{done}]\!] = \quad (72)$$
$$\quad \textbf{apply-to-each}(\textbf{abs}(\textbf{bind}(\mathit{id}[\![LI]\!]), \textbf{effect}(\mathit{expr}[\![E3]\!])),$$
$$\qquad \textbf{list-reverse}(\textbf{int-closed-interval}(\mathit{expr}[\![E2]\!], \mathit{expr}[\![E1]\!])))$$

$$\mathit{expr}[\![E1\ \texttt{;}\ E2]\!] = \quad (73)$$
$$\quad \textbf{seq}(\textbf{effect}(\mathit{expr}[\![E1]\!]), \mathit{expr}[\![E2]\!])$$

$$\mathit{expr}[\![E1\ \texttt{;}\ E2\ \texttt{;}]\!] = \quad (74)$$
$$\quad \mathit{expr}[\![E1; E2]\!]$$

$$\mathit{expr}[\![\texttt{match}\ E\ \texttt{with}\ SM]\!] = \quad (75)$$
$$\quad \textbf{apply}(\textbf{prefer-over}(\mathit{abs}[\![SM]\!],$$
$$\qquad \textbf{abs}(\textbf{throw}('Match\_failure'))), \mathit{expr}[\![E]\!])$$

$$\mathit{expr}[\![\texttt{function}\ SM]\!] = \quad (76)$$
$$\quad \textbf{close}(\textbf{prefer-over}(\mathit{abs}[\![SM]\!],$$
$$\qquad \textbf{abs}(\textbf{throw}('Match\_failure'))))$$

$$\mathit{expr}[\![\texttt{fun}\ Ps1\ \texttt{->}\ E\ \texttt{|}\ Ps2\ \texttt{->}\ E...]\!] = \quad (77)$$
$$\quad \textbf{prefer-over}(\mathit{expr}[\![\texttt{fun}\ Ps1\ \texttt{->}\ E]\!],$$
$$\qquad \mathit{expr}[\![\texttt{fun}\ Ps2\ \texttt{->}\ E...]\!])$$

$$\mathit{expr}[\![\texttt{fun}\ P1\ P2\ P3...\ \texttt{->}\ E]\!] = \quad (78)$$
$$\quad \textbf{curry}(\mathit{expr}[\![\texttt{fun (}P1\texttt{ , }P2\texttt{) -> fun }P3...\texttt{ -> }E]\!])$$

$$\mathit{expr}[\![\texttt{fun}\ P1\ P2\ \texttt{->}\ E]\!] = \quad (79)$$
$$\quad \textbf{curry}(\mathit{expr}[\![\texttt{fun (}P1\texttt{ , }P2\texttt{) -> }E]\!])$$

$$\mathit{expr}[\![\texttt{fun}\ P1\ \texttt{->}\ E]\!] = \quad (80)$$
$$\quad \mathit{expr}[\![\texttt{function}\ P1\ \texttt{->}\ E]\!]$$

$$\mathit{expr}[\![\texttt{try}\ E\ \texttt{with}\ SM]\!] = \quad (81)$$
$$\textbf{catch-else-rethrow}(\mathit{expr}[\![E]\!], \textbf{restrict-domain}(\mathit{abs}[\![SM]\!], \textbf{bound-type}(\textbf{typeid}('exn'))))$$

$$\mathit{expr}[\![VD\ \texttt{in}\ E]\!] = \quad (82)$$
$$\textbf{scope}(\mathit{decl}[\![VD]\!], \mathit{expr}[\![E]\!])$$

A.5 Pattern matching

$$\mathit{abs}[\![P1\ \texttt{->}\ E1]\!] = \quad (83)$$
$$\textbf{abs}(\mathit{patt}[\![P1]\!], \mathit{expr}[\![E1]\!])$$

$$\mathit{abs}[\![P1\ \texttt{->}\ E1\ \texttt{|}\ P2\ \texttt{->}\ E2...]\!] = \quad (84)$$
$$\textbf{prefer-over}(\mathit{abs}[\![P1\ \texttt{->}\ E1]\!], \mathit{abs}[\![P2\ \texttt{->}\ E2...]\!])$$

$$\mathit{abs}[\![\texttt{|}\ P1\ \texttt{->}\ E1...]\!] = \quad (85)$$
$$\mathit{abs}[\![P1\ \texttt{->}\ E1...]\!]$$

A.6 Value definitions

$$\mathit{decl}[\![\texttt{let rec}\ LB...]\!] = \quad (86)$$
$$\textbf{generalise-all}(\textbf{recursive-typed}(\mathit{bound\text{-}ids}[\![LB...]\!], \mathit{decl\text{-}mono}[\![LB...]\!]))$$

$$\mathit{decl}[\![\texttt{let}\ LB...]\!] = \quad (87)$$
$$\mathit{decl}[\![LB...]\!]$$

$$\mathit{decl}[\![LB1\ \texttt{and}\ LB2...]\!] = \quad (88)$$
$$\textbf{map-union}(\mathit{decl}[\![LB1]\!], \mathit{decl}[\![LB2...]\!])$$

$$\mathit{decl}[\![VP...\ \texttt{=}\ E]\!] = \quad (89)$$
$$\textbf{generalise-all}(\mathit{decl\text{-}mono}[\![VP... = E]\!])$$

$$\mathit{decl}[\![P\ \texttt{=}\ V]\!] = \quad (90)$$
$$\textbf{generalise-all}(\mathit{decl\text{-}mono}[\![P\ \texttt{=}\ V]\!])$$

$$\mathit{decl}[\![P\ \texttt{=}\ C]\!] = \quad (91)$$
$$\textbf{generalise-all}(\mathit{decl\text{-}mono}[\![P\ \texttt{=}\ C]\!])$$

$$\mathit{decl}[\![P\ \texttt{=}\ \texttt{function}\ SM]\!] = \quad (92)$$
$$\textbf{generalise-all}(\mathit{decl\text{-}mono}[\![P\ \texttt{=}\ \texttt{function}\ SM]\!])$$

$$\mathit{decl}[\![P\ \texttt{=}\ \texttt{fun}\ MM]\!] = \quad (93)$$
$$\textbf{generalise-all}(\mathit{decl\text{-}mono}[\![P\ \texttt{=}\ \texttt{fun}\ MM]\!])$$

$$\mathit{decl}[\![P\ \texttt{=}\ E]\!] = \quad (94)$$
$$\mathit{decl\text{-}mono}[\![P\ \texttt{=}\ E]\!]//default$$

$$\mathit{decl\text{-}mono}[\![LB1\ \texttt{and}\ LB2...]\!] = \quad (95)$$
$$\textbf{map-union}(\mathit{decl\text{-}mono}[\![LB1]\!], \mathit{decl\text{-}mono}[\![LB2...]\!])$$

$$\mathit{decl\text{-}mono}[\![P\ \texttt{=}\ E]\!] = \quad (96)$$
$$\textbf{match}(\mathit{expr}[\![E]\!], \textbf{prefer-over}(\mathit{patt}[\![P]\!], \textbf{abs}(\textbf{throw}('Match\_failure'))))$$

$$\mathit{decl\text{-}mono}[\![VP...\ \texttt{=}\ E]\!] = \quad (97)$$
$$\textbf{bind-value}(\mathit{id}[\![V]\!], \mathit{expr}[\![\texttt{fun}\ P... -> E]\!])$$

$$\mathit{bound\text{-}ids}[\![LB1\ \texttt{and}\ LB2...]\!] = \quad (98)$$
$$\textbf{map-union}(\mathit{bound\text{-}ids}[\![LB1]\!], \mathit{bound\text{-}ids}[\![LB2...]\!])$$

$$\mathit{bound\text{-}ids}[\![LI\ \texttt{=}\ E]\!] = \quad (99)$$
$$\textbf{map1}(\mathit{id}[\![LI]\!], \textbf{unknown-type})$$

$$\mathit{bound\text{-}ids}[\![\texttt{(}\ LI\ \texttt{:}\ TE\ \texttt{)}\ \texttt{=}\ E]\!] = \quad (100)$$
$$\textbf{map1}(\mathit{id}[\![LI]\!], \mathit{type}[\![TE]\!])$$

($\mathit{bound\text{-}ids}$ [[P = E]] otherwise unspecified)

$$\mathit{bound\text{-}ids}[\![VP...\ \texttt{=}\ E]\!] = \quad (101)$$
$$\textbf{map1}(\mathit{id}[\![V]\!], \textbf{unknown-type})$$

A.7 Types

$$\mathit{type}[\![LI]\!] = \quad (102)$$
$$\textbf{bound-type}(typeid[\![LI]\!])$$

$$\mathit{type}[\![TE1\ \texttt{->}\ TE2]\!] = \quad (103)$$
$$\textbf{depends}(\mathit{type}[\![TE1]\!], \mathit{type}[\![TE2]\!])$$

$$\mathit{type}[\![\texttt{(}\ TE\ \texttt{)}]\!] = \quad (104)$$
$$\mathit{type}[\![TE]\!]$$

$$\mathit{type}[\![TELI]\!] = \quad (105)$$
$$\textbf{instantiate-type}(\mathit{type}[\![LI]\!], \textbf{list1}(\mathit{type}[\![TE]\!]))$$

$$\mathit{type}[\![\texttt{(}\ TE1\ \texttt{,}\ TE2...\ \texttt{)}\ LI]\!] = \quad (106)$$
$$\textbf{instantiate-type}(\mathit{type}[\![LI]\!], \mathit{list}[\![TE1\ \texttt{,}\ TE2...]\!])$$

$$\mathit{type}[\![\texttt{'}\ LI]\!] = \quad (107)$$
$$\textbf{typevar}('LI')$$

$$\mathit{type}[\![TE1 * TE2]\!] = \quad (108)$$
$$\textbf{tuple2}(\mathit{type}[\![TE1]\!], \mathit{type}[\![TE2]\!])$$

$$\mathit{type}[\![TE1 * TE2 * TE3...]\!] = \quad (109)$$
$$\textbf{tuple-prefix}(\mathit{type}[\![TE1]\!], \mathit{type}[\![TE2 * TE3...]\!])$$

$$\mathit{list}[\![TE1\ \texttt{,}\ TE2]\!] = \quad (110)$$
$$\textbf{list2}(\mathit{type}[\![TE1]\!], \mathit{type}[\![TE2]\!])$$

$$\mathit{list}[\![TE1\ \texttt{,}\ TE2\ \texttt{,}\ TE3...]\!] = \quad (111)$$
$$\textbf{list-prefix}(\mathit{type}[\![TE1]\!], \mathit{list}[\![TE2, TE3...]\!])$$

A.8 Type definitions

$$\mathit{decl}[\![LI\ \texttt{==}\ TE]\!] = \quad (112)$$
$$\textbf{typedef}(\mathit{typeid}[\![LI]\!], \mathit{type}[\![TE]\!])$$

$$\mathit{decl}[\![TPS\ LI\ \texttt{==}\ TE]\!] = \quad (113)$$
$$\textbf{typedef}(\mathit{typeid}[\![LI]\!], \textbf{type-abs}(\mathit{typevar\text{-}list}[\![TPS]\!], \mathit{type}[\![TE]\!]))$$

$$\mathit{decl}[\![TPS\ LI\ \texttt{=}\ \{\ LD...\ \}]\!] = \quad (114)$$
$$\textbf{typedef}(\mathit{typeid}[\![LI]\!], \textbf{type-abs}(\mathit{typevar\text{-}list}[\![TPS]\!], \textbf{record}(\mathit{map}[\![LD...]\!])))$$

decl $[\![LI$ = $LD...]\!] =$ (115)
typedef(*typeid* $[\![LI]\!]$,
record(*map* $[\![LD...]\!]$))

decl $[\![TPS\ LI$ = $CD...]\!] =$ (116)
typedef(*typeid* $[\![LI]\!]$, **rectype**(*typeid* $[\![LI]\!]$,
type-abs(*typevar-list* $[\![TPS]\!]$, **variant**(*map* $[\![CD...]\!]$))))

decl $[\![LI$ = $CD...]\!] =$ (117)
typedef(*typeid* $[\![LI]\!]$, **rectype**(*typeid* $[\![LI]\!]$,
variant(*map* $[\![CD...]\!]$)))

typevar-list $[\![$ ' $LI]\!] =$ (118)
list1(**typevar**($'LI'$))

typevar-list $[\![$ (' LI)$]\!] =$ (119)
typevar-list $[\![$' $LI]\!]$

typevar-list $[\![$ (' $LI1$, ' $LI2...$)$]\!] =$ (120)
list-prefix($typevar('LI1')$, *typevar-list* $[\![('LI2...)]\!]$)

map $[\![LI : TE]\!] =$ (121)
map1($'LI'$, **variable**(*type* $[\![TE]\!]$))

map $[\![$`mutable` $LI : TE]\!] =$ (122)
map1($'LI'$, **variable**(*type* $[\![TE]\!]$))

map $[\![LD1$; $LD2...]\!] =$ (123)
map-union(*map* $[\![LD1]\!]$, *map* $[\![LD2...]\!]$)

map $[\![K]\!] =$ (124)
map1($'K'$, **tuple-empty**)

map $[\![K$ `of` $TE]\!] =$ (125)
map1($'K'$, *type* $[\![TE]\!]$)

map $[\![CD1$ | $CD2...]\!] =$ (126)
map-union(*map* $[\![CD1]\!]$, *map* $[\![CD2...]\!]$)

A.9 Module implementations

decl-impl $[\![IP$; ; $...]\!] =$ (127)
accum(*decl* $[\![IP]\!]$, *decl-impl* $[\![...]\!]$)

decl-impl $[\![]\!] =$ (128)
map-empty

decl $[\![E]\!] =$ (129)
seq(**print**(*expr* $[\![E]\!]$), **map-empty**)

decl $[\![$`type` $TD]\!] =$ (130)
decl $[\![TD]\!]$

decl $[\![$`type` $TD1$ `and` $TD2...]\!] =$ (131)
map-union(*decl* $[\![TD1]\!]$, *decl* $[\![$`type` $TD2...]\!]$)

decl $[\![$`exception` $CD]\!] =$ (132)
typedef(**typeid**($'exn'$), **variant-type-extend**
(**bound-type**(**typeid**($'exn'$)), *map* $[\![CD]\!]$))

decl $[\![$`exception` $CD1$ `and` $CD2...]\!] =$ (133)
accum(*decl* $[\![$`exception` $CD1]\!]$, *decl* $[\![$`exception` $CD2...]\!]$)

decl $[\![D]\!] =$ (134)
map-empty

prog $[\![IMPL]\!] =$ (135)
scope(**caml-light-library**, *decl-impl* $[\![IMPL]\!]$)