Cedar Safe Language Syntax

¶3.3
1 module ::=DIRECTORY (n_d ?(: TYPE n_t) ?(USING [n_u, ...]) ;
(interface | implementation)
2 interface ::= n_m, !.. : CEDAR DEFINITIONS ?locks
?(IMPORTS ((n_{iv} : |) n_{it}), ...) ~ { ?open[7] (d | b); !.. } .
3 implementation ::= n_m : CEDAR
(PROGRAM ?drType[42] | MONITOR ?drType[42] ?locks)
?(IMPORTS ((n_{iv} : |) n_{it}), ...) ?(EXPORTS n_e, ...) ~ block .
5 locks ::= LOCKS e ?(USING n_u: t)

¶3.4
6 block ::= ?(CHECKED | UNCHECKED | TRUSTED)
{ ?open ?enable ?body ?(EXITS (n, !..=>s); ...) }
7 open ::= OPEN (n ~~ e | e), !.. ; --In 2, 5.
8 enable ::= ENABLE { enChoice; ... };
9 enChoice ::=(e, !.. | ANY) => s --In 7, 27.1.
10 body ::= (d | b); !.. ; s; ... | s; !.. --In 6, 27.1.

¶3.5
11 declaration ::= n, !.. : ?(PUBLIC | PRIVATE) varTC
13 binding ::= n, !.. : ?(PUBLIC | PRIVATE) t ~ (
e | t_2 -- t=TYPE-- | CODE | ?INLINE ?(ENTRY|INTERNAL) block)

¶3.6
14 Statement ::= e_1 _ e_2 | e | block[6] | escape | loop | NULL
16 escape ::=GOTO n | EXIT | CONTINUE | (RETURN|RESUME) ?e
17 loop ::= ?iterator ?(WHILE e | UNTIL e)
DO ?body[10] ?(REPEAT FINISHED=>s) ENDLOOP
18 iterator ::= THROUGH e |
FOR n : t (?DECREASING IN e | e, e)
e is a subrange. n is readonly.

¶3.7
19 expression ::= n | literal | application[26] |
(e | typeName[37]) . (9) n |
prefixOp e | e_1 infixOp e_2 | e_1 AND (2) e_2 | e_1 OR (1) e_2 |
e ^ (9) | ERROR | [argBinding[27]] |
builtIn [e, ... ?applEn[27.1]] |
funnyAppl e ?([?argBinding[27] ?applEn[27.1]]) |
s | subrange[25] | if[28] | select[29] | safeSelect[32]
Precedence is in bold in rules 19-21. All operators associate to the left except _, which associates to the right. Application has highest precedence. Subrange only after IN or THROUGH. s only in if [28] and select choices[30 33].
20 prefixOp ::= @ (8) | # (7) | (~ | NOT) (3)
21 infixOp ::= * | / | MOD (6) | + | & (5) | relOp (4) | _ (0)
22 relOp ::= ?NOT (?~ (= | < | >) | <= | >= | # | IN) --In 21, 30.
23 builtIn ::= -- These are enumerated in Table 4 5.
24 funnyAppl ::= FORK | JOIN | WAIT | NOTIFY | BROADCAST |
SIGNAL | ERROR | RETURN WITH ERROR
25 subrange ::= ?typeName[37] (e_1 .. e_2) -In 19, 39.
26 application ::= e [?argBinding ?applEn]
27 argBinding ::= (n ~ ?e), !.. | (?e), !.. --In 19, 26.
27.1 applEn ::= ! enChoice[9]; ... -- In 19, 26.

¶3.8
28 if ::= IF e_1 THEN e_2 ?(ELSE e_3)
29 select ::= SELECT e FROM choice; ... endChoice
The ";" is "," in an expression, here and in 32.
30 choice ::= (?relOp[22] e_1), !..=> e_2
31 endChoice ::= ENDCASE ?(=> e_3) --In 29, 32, 34.
32 safeSelect ::=WITH e SELECT FROM safeChoice; ... endChoice[31]
33 safeChoice ::= n : t => e_2

¶3.2
56 name ::= letter (letter | digit)...
57 literal ::= num | ?B D | digit (DIGIT | HEX) ... | H
?num . num ?exponent | num exponent |
' extendedChar | " extendedChar ... " | $ n
58 exponent ::=?(E + | -) num
59 num ::= digit !..
60 extendedChar ::= space | \ extension | anyCharNot'"Or\
61 extension ::= $digit_1$ $digit_2$ OR $digit_3$ | FNU | ' | " | \

¶4.1
type ::= typeName | builtInType | typeCons
¶4.2
typeName ::= n | typeName . n | typeName[e] --I
¶4.3
builtInType ::= INT | REAL | TYPE | ATOM | CONDITION | MONITORLOCK
See Table 4 2. TYPE only in a b or an interface's d.
typeCons ::= subrange[25] | paintedTC[40.1] | transfe
arrayTC[44] | seqTC[45] | refTC[46] | listTC[47] |
recordTC[50] | unionTC[52] | enumTC[54] | defaultTC[55]
varTC ::= (| READONLY | VAR) t | ANY
In 11, 46, 47. ANY only in refTC. VAR only in interface decl
paintedTC ::= typeName[37] PAINTED t
In 3, 13, 14.
¶4.4
transferTC ::=?(SAFE | UNSAFE) xfer ?drType
xfer ::=PROC | PORT | PROCESS | SIGNAL | ERROR | PRO
drType ::= ?$fields_1$ RETURNS $fields_2$ | $fields_1$ --
fields ::= [d[11], ...] | [t, ...] | ANY --In
arrayTC ::= ARRAY t_1 OF t_2
--In 2, 10, 43.
seqTC ::= SEQUENCE n : t_1 OF t_2 -- Only as last type
--In 2, 10.
¶4.5
refTC ::= REF ?varTC[40]
listTC ::= LIST OF varTC[40]
¶4.6
recordTC ::= ?MONITORED RECORD fields[43]
unionTC ::= SELECT n : (t | *) FROM (n => fields[4
ENDCASE -- Only as last type
¶4.7
enumTC ::= {n, ...} | MACHINE DEPENDENT {(?n)(...
¶4.11
defaultTC ::= t _ | t _ e
Only as t in a decl in body[9] or field [43] (n: t _ e),in a TYP

Notation: item | item=choose one; ?item=zero or one item; terminals: SMALLCAPS, underlined other than bold ()?| (parens are terminals only in rules 19, item s ...=zero or more items, separated by s; item s !..=one or more items, separated by s; with s=";", a trailing ";" is optio
Abbreviated non-terminals: b=binding[13]; d=declaration[11]; e=expression[19]; n=name[56]; s=statement[14]; t=type[36]. 16 Feb 83
Comments: n[56] means n is defined in rule 56; n_2=n (the subscript is only for the desugaring). CLRMSafeGram.press