Cedar Full Language Syntax

¶3.3

1 module ::= DIRECTORY (n_d ?(: TYPE ?n_t)
(interface | implementation)

2 interface ::= n_m, !.. : ?CEDAR DEFINITIONS ?locks
?(IMPORTS ((n_{iv} : |) n_{it}), ...)
?•(SHARES n_s, ...) ~ ?•access[12] { ?open[7] (d | b); ... }

3 implementation ::= n_m : ?CEDAR ?safety
(PROGRAM ?drType[42] | MONITOR ?drType[42] ?locks)
?(IMPORTS ((n_{iv} : |) n_{it}), ...) ?(EXPORTS n_e, ...)
?•(SHARES n_s, ...) ~ ?•access[12] block .

4 safety ::= SAFE | UNSAFE --In 3, 41.

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, 17. •The ~~ may be written as :.

8 enable ::= ENABLE (enChoice | {enChoice; ...});

9 enChoice ::=(e, !.. | ANY) => s --In 7, 27.1.

10 body ::= (d | b); !.. ; s; ... | s; !.. --In 5, 17.

¶3.5

11 declaration ::= n, !.. : ?access varTC[40]
In 2, 10, 43. VAR, READONLY only for interface var.

12 access ::= PUBLIC | PRIVATE --In 2, 3, 11, 13, 50, 51, 53.

13 binding ::= n, !.. : ?access t ~ (
e | t_2 -- if t=TYPE-- | CODE |
„, ?TRUSTED MACHINE CODE { (e, ...); } |
?INLINE ?(ENTRY | INTERNAL) block[6])
In 2, 10. •The ~ may be written as =. •ENTRY or INTERNAL may be written before t. Block or MACHINE CODE only for proc types.

¶3.6

14 Statement ::= e_1_e_2 | e | block[6] | escape | loop | ...
--In 6, 10, 17, 19.

16 escape ::= GOTO n | GO TO n | EXIT | CONTINUE | •LOOP | •STATE |
•RETRY | •REJECT | (RETURN | RESUME) ?e | „„e

17 loop ::= ?iterator ?(WHILE e | UNTIL e)
DO ?•open[7] ?•enable[8] ?body[10]
?•(REPEAT (n, !.. => s); ...)

18 iterator ::= THROUGH e |
FOR (n : t | µn) (?DECREASING ?(TO e_2) | ?(BY e_3))
e is a subrange. In FOR n: t ..., n is readonly.

¶3.7

19 Expression ::= n | literal[57] | 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) | •STOP | 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 35.

20 prefixOp ::= @ (8) | (7) | (~ | NOT) (3) | ...

21 infixOp ::= * | / | MOD (6) | + | (5) | ...

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 | •NEW | •START |
•RESTART | „,TRANSFER WITH | „,RETURN WITH

25 subrange ::= ?typeName[37]
([e_1 .. e_2] | [(e_1 ... e_2)] | (e_1 ... e_2)) --In 19, 39, 48.

26 application ::= e [?argBinding ?applEn]

27 argBinding ::= (n ~ (e | | µTRASH)), !.. | (e | µTRASH), !..
In 19, 26. •The ~ may be written as :. •NULL may be written for TRASH.

27.1 applEn ::= ! enChoice[9]; ... -- In 19, 26.

¶3.8

28 if ::= IF e_1 THEN e_2 ?(ELSE e_3)

29 select ::= (?n_u, SELECT[: n_u, SELECT]e) FROM choice; ... endChoice
The ";" is "," in an expression, here and in 32 and 34.

30 choice ::= (?relOp[22] e_1), !..=> e_2

31 endChoice ::= ENDCASE ?(=> e_3) --In 29, 32, 34.

32 safeSelect ::=WITH e SELECT FROM safeChoice; ... end...

33 safeChoice ::= n : t => e_2

34 •withSelect ::= WITH (n_1 ~~ e_1 | • e_1) SELECT ?,e...
withChoice; ... endChoice[31] --•The ~~ may be ...

35 •withChoice ::= n_2 => e_2 | n_2, n_2, !.. => e_2

¶4

36 type ::= typeName | builtInType | typeCons

37 typeName ::= n_1 | typeName . n_2 | •typeName[e]
•n_2 typeName --In 19, ...
In 3, 18, 14.

¶4.2

38 builtInType ::= INT | REAL | TYPE | ATOM | CONDITION |
MONITORLOCK | •LONG CARDINAL | •,?LONG UNSPECIFIED |
•,MDSZone | µ ?,UNCOUNTED ZONE
See Table 4 2. TYPE only in a b or an interface's d.
In 3, 17.

39 typeCons ::= subrange[25] | paintedTC[40.1] | transfe...
arrayTC[44] | seqTC[45] | ,descriptorTC[45.1] | refTC[46]
,pointerTC[48] | •,relativeTC[49] | recordTC[50] | unio...
enumTC[54] | defaultTC[55]

¶4.3

40 varTC ::= (| READONLY | VAR) t | ANY
In 11, 45 48. ANY only in refTC. VAR only in interface decl.

40.1 paintedTC ::= typeName[37] PAINTED t

¶4.4

41 transferTC ::=?safety[4] xfer ?drType
xfer ::= PROCEDURE | PROC | PORT | PROGRAM |
PROCESS | SIGNAL | ERROR

42 drType ::= ?$fields_1$ RETURNS $fields_2$ | $fields_1$ --...

43 fields ::= [d[11], ...] | [t, ...] | ANY --In 42, ...

44 arrayTC ::= ?µPACKED ARRAY ?t_1 OF t_2

45 seqTC ::= ?µPACKED SEQUENCE tag[53] OF t
Legal only as last type in the fields of a recordTC or unio...

45.1 ,descriptorTC ::= ?LONG DESCRIPTOR FOR varTC[40]
varTC must be an array type.

46 refTC ::= REF ?varTC[40]

47 listTC ::= LIST OF varTC[40]

48 ,pointerTC ::= ?LONG ?ORDERED ?•BASE POINTER
•POINTER TO (?t_2 | varTC[40]) | •POINTER TO FRAME [n]
Subrange only in a relativeTC; no typeName[37] on it.

49 •,relativeTC ::= typeName[37] RELATIVE t

¶4.6

50 recordTC ::= ?access[12] (?MONITORED RECORD fields[43]
MACHINE DEPENDENT RECORD (mdFields | •fields[43]))

51 „mdFields ::= [((n pos), ... : ?•access[12] t),
51.1 „pos ::= e_1 (?(: e_2 .. e_3)) --In 51, 53.

52 unionTC ::= SELECT tag FROM
(n => (fields[43] | mdFields[51] | •NULL)), ...
•withSelect[34]
Legal only as last type in the fields of a recordTC or unio...

53 tag ::= (n ?pos[51.1] : ?•access[12] | µ,(COMPUTED|OVE...
(t | *) --In 44, 52. * only in unionTC.

¶4.7

54 enumTC ::= {n, ...} | MACHINE DEPENDENT ((?n)(...

55 defaultTC ::= t _ | t _ e TRASH | t _ µ TRASH
defaultTC legal only as the type in a decl in a body[9] or fi...
TYPE binding[13], or in NEW. Note the terminal |. •TRASH may be ...

¶3.26

56 name ::= letter (letter | digit)...

57 literal ::= num[0] | B | O ?num) -- INT literal. |
digit (digit | A | B | C | D | E | F) ... | H ?num -- Hex INT litera...
?num . num ?exponent | num exponent -- REAL liter...
' extendedChar | • digit " extendedChar ... | $ n •L...

58 exponent ::= E(+ |) num

59 num ::= digit ...

60 extendedChar ::= space | \ extension | anyCharN...

61 extension ::= $digit_1$ $digit_2$ $digit_3$ |
N | R | T | B | F | L | ' | " | \

Notation: item | item=choose one; ?item=zero or one item; terminals: SMALLCAPS; punctuation 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 opti...
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: •=obsolete; µ=efficiency hack; ,=unsafe; „=machine dependent; n[56] means n is defined in rule 56; n_2=n (the subscript is ...