Cedar Language Syntax

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[6] db[10]; !.. } .

3 implementation ::= n_m : ?CEDAR ?(SAFE | UNSAFE)
(PROGRAM drType[43] | MONITOR drType[43] ?locks)
?(IMPORTS ((n_{iv} : |) n_{it}), ...) ?(EXPORTS n_e, ...)
?•(SHARES n_s, ...) ~ ?•access[12] block .

4 locks ::= LOCKS e ?(USING n_u: t)

---

5 block ::= ?(CHECKED | UNCHECKED | TRUSTED)
{ ?open ?enable body ?(EXITS (n, !..=>s); ...) } --In 3, 13, 15.

6 open ::= OPEN (n ~~ e | e), !.. ;
In 2, 5, 17. •The ~~ may be written as :.

7 enable ::= ENABLE (enChoice | {enChoice; ...}); --In 5, 17.

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

9 body ::= db; ... s; ... --In 5, 17. A trailing ; is required after the db's.

10 db ::= d | b --In 2, 9.

---

11 declaration ::= n, !.. : ?access varTC[39]
In 10, 44. VAR, READONLY only for interface variable.

12 access ::= PUBLIC | PRIVATE --In 2, 3, 11, 13, 46, 47, 50.

13 binding ::= n, !.. : ?access t ~ (
e | t_2 -- t=TYPE-- | CODE |
„, ?TRUSTED MACHINE CODE { (e, ...); ...} |
?INLINE ?(ENTRY | INTERNAL) block[4])
In 10. •The ~ may be written as =. •ENTRY or INTERNAL may be written before t. Block or MACHINE CODE only for proc types.

---

14 Statement ::= e_1_e_2 | e | block[4] | control | loop | NULL
--In 5, 9, 17, 19.

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

17 loop ::= ?iterator ?(WHILE e | UNTIL e)
DO ?•open[6] ?•enable[7] body[9]
?•(REPEAT (n, !..=>s); ...) ENDLOOP

18 iterator ::= THROUGH e |
FOR (n : t | μn) (?DECREASING IN e | _ e_1 , e_2)
e is a subrange. In FOR n: t ..., n is readonly.

---

19 expression ::= n | literal[57] | (e) | application[26] | e . (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] | •withSelect[34]
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 [29].

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

21 infixOp ::= * | / | MOD (6) | + | (5) | relOp (4) | _ (0)

22 relOp ::= ?NOT (?~ (= | < | >) | <= | >= | # | IN) --In 19, 30.

23 builtIn ::= -- These are enumerated in Table 4 5.

24 funnyAppl ::= FORK | JOIN | SIGNAL | ERROR | •NEW |
•START | •RESTART | WAIT | NOTIFY | BROADCAST |
RETURN WITH ERROR| „,TRANSFER WITH | „,RETURN WITH

25 subrange ::= ?typeName[37]
([e_1 .. e_2] | [e_1 .. e_2) | (e_1 .. e_2] | (e_1 .. e_2)) --In 19, 39, 54.

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[8]; ... -- In 19, 26.

---

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 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; ... endChoice

33 safeChoice ::= n : t => e

34 •withSelect ::= WITH (n_1 ~~ e_1 | • e_1) SELECT ?,e FROM
withChoice; ... endChoice[30] --•The ~~ may be written as :.

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

---

36 type ::= typeName | builtInType | typeCons

37 typeName ::= n | typeName . n | typeName[e] |
n_3 ... typeName --In 25

38 builtInType ::= ?LONG (INTEGER | •CARDINAL) | INT |
TYPE | ATOM | MONITORLOCK | CONDITION | •,MDSZone |
μ ? UNCOUNTED ZONE | •, ?LONG UNSPECIFIED
See Table 4 2. TYPE only in a body's binding or an interface

39 typeCons ::= subrange[25] | defaultTC[40] | transferTC |
enumTC[45] | recordTC[46] | unionTC[49] | arrayTC[51] | seqTC |
refTC[52] | listTC[53] | ,pointerTC[54] | ,descriptorTC |
•,relativeTC[55.1]

39.1 varTC ::= (| READONLY | VAR) t | ANY
In 11, 44, 47, 52 55. ANY only in refTC. VAR only in interface

40 defaultTC ::= t _ | t _ e | μt _ e | TRASH | μt
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

41 transferTC ::=?(SAFE | UNSAFE) xfer drType

42 xfer ::= PROCEDURE | PROC | PORT | PROGRAM |
PROCESS | SIGNAL | ERROR

43 drType ::= ?fields_1 ?(RETURNS fields_2) --In 3, 41

44 fields ::= [d[11], ...] | [varTC[39], ...] | ANY --In

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

46 recordTC ::= ?access[12] (?MONITORED RECORD fields
„ MACHINE DEPENDENT RECORD (mdFields | •fields[44]))

47 „mdFields ::= [((n pos), ... : ?•access[12] varTC

48 „pos ::= (e_1 ?(: e_2 .. e_3)) --In 47, 50.

49 unionTC ::= SELECT tag FROM
(n, ... => (fields[44] | mdFields[47] | •NULL)), .
Legal only as last type in a recordTC or unionTC.

50 tag ::= (n ?„pos[48] : ?•access[12] | μ,COMPUTED | μ,C
(t | *) --In 49, 51.1. * only in unionT

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

51.1 seqTC ::= ?μPACKED SEQUENCE tag[50] OF t
Legal only as last type in a recordTC or unionTC.

52 refTC ::= REF ?varTC[39]

53 listTC ::= LIST OF varTC[39]

54 ,pointerTC ::= ?LONG ?•ORDERED ?•BASE POINTER
?•subrange[25] ?(TO varTC[39]) | •POINTER TO FRAME [n]
Subrange only in a relativeTC; no typeName[37] on it.

55 ,descriptorTC ::= ?LONG DESCRIPTOR FOR varTC[39]
varTC must be an array type.

55.1 •,relativeTC ::= typeName[37] RELATIVE t
t a pointer or descriptor type, typeName a base pointer typ

---

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

57 literal ::= num ?((D | B) ?num) | -- INT literal.
digit (digit | A | B | C | D | E | F)... (H) ?num | -- Hex INT lite
?num . num ?exponent | num exponent | -- REAL l
' extendedChar | • digit' ! extendedChar ... | $ n ?•L

58 exponent ::= (E | 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: SMALL CAPS, 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]; db=d | b; e=expression[19]; n=name[56]; s=statement[14]; t=type[36]. 15 Sept 82

Comments: •=obsolete; μ=efficiency hack; ,=unsafe; „=machine dependent; n[56] means n is defined in rule 56; n_2=n (the subscript is only for the des... CLR for the des...