Εργασία Μεταγλωττιστών
Τμήμα Α2

Ομάδα 15

29 Μαρτίου 2021

Ομάδα 15

Αναλυτικά τα μέλη:

Διονύσης Νικολόπουλος AM : 18390126

Θανάσης Αναγνωστόπουλος AM : 18390043

Αριστείδης Αναγνωστόπουλος AM : 16124

Σπυρίδων Φλώρος AM : 141084

Αναλυτικά οι ρόλοι:

Γενικός Συντονιστής: Διονύσης Νικολόπουλος

Υπεύθυνος Τμήματος Εργασίας Α2: Διονύσης Νικολόπουλος

Η εργασία αυτή πραγματοποιήθηκε με χρήση LaTeX

Περιεχόμενα

1 Εισαγωγή

Τι πραγματεύεται η εργασία

Στην εργαστηριακή εργασία αυτή ασχοληθήκαμε με την δημιουργία ενός ντετερμινιστικού αυτόματου πεπερασμένων καταστάσεων.

Ο σκοπός του αυτομάτου

Το αυτόματο αυτό έχει ως σκοπό να λειτουργεί σαν Λεκτικός Αναλυτής της γλώσσας Uni-C, ενός παρακλαδιού της γλώσσας C, απο το Πανεπιστήμιο Δυτικής Αττικής.

Βήματα Υλοποίησης

Πριν πραγματοποιηθεί φυσικά η κωδικοποίηση για την προσσέγγιση του προβλήματος αυτού, έγινε η περιγραφή της γλώσσας Uni-C σε μορφή BNF.

Η κωδικοποίηση του έχει γίνει καταρχήν με την βοήθεια του προγράμματος *FSM*, που επίσης είναι προγραμματισμένο πάνω στην γλώσσα C.

Ο σχεδιασμός των διαγραμμάτων έχει γίνει με το πρόγραμμα *JFLAP*.

Εξήγηση για το τι ακολουθεί

Σε αυτό το PDF αρχικά θα παρατεθεί η περιγραφή της Uni-C σε BNF.

Έπειτα, θα παρατεθούν επίσης και τα διαγράμματα του εναίου αυτόματου που βοηθούν στην κωδικοποίηση του αλλά και στην σύλληψη νοητά του τι περιμένουμε να κάνει το αυτόματό μας, πως θα το υλοποιήσουμε τεχνικά.

Στην συνέχεια, θα ακολουθήσει ο κώδικας του αυτόματου τμηματικά με εξηγήσεις της λειτουργίας του.

Τελος, εφόσον ο κώδικας παραδωθεί και ολοκληρωμένα, θα παρατεθούν ορισμένοι ελέγχοι πραγματικής λειτουργίας του αυτόματου σε εικόνες.

2 Περιγραφή σύνταξης της Uni-C σε BNF

Πριν την υλοποίηση του αυτομάτου σε εκτελέσιμο κώδικα, όπως αναφέραμε παραπάνω, έγινε η ανάλυση της σύνταξης της γλώσσας Uni-C σε μορφή BNF (Backus-Naur Form ή Μορφή Μπάκους-Ναούρ).
Οπότε, με την βοήθεια του περιβάλλοντος λεκτικού επεξεργαστή Emacs, γράψαμε τον εξής συμβολισμό.

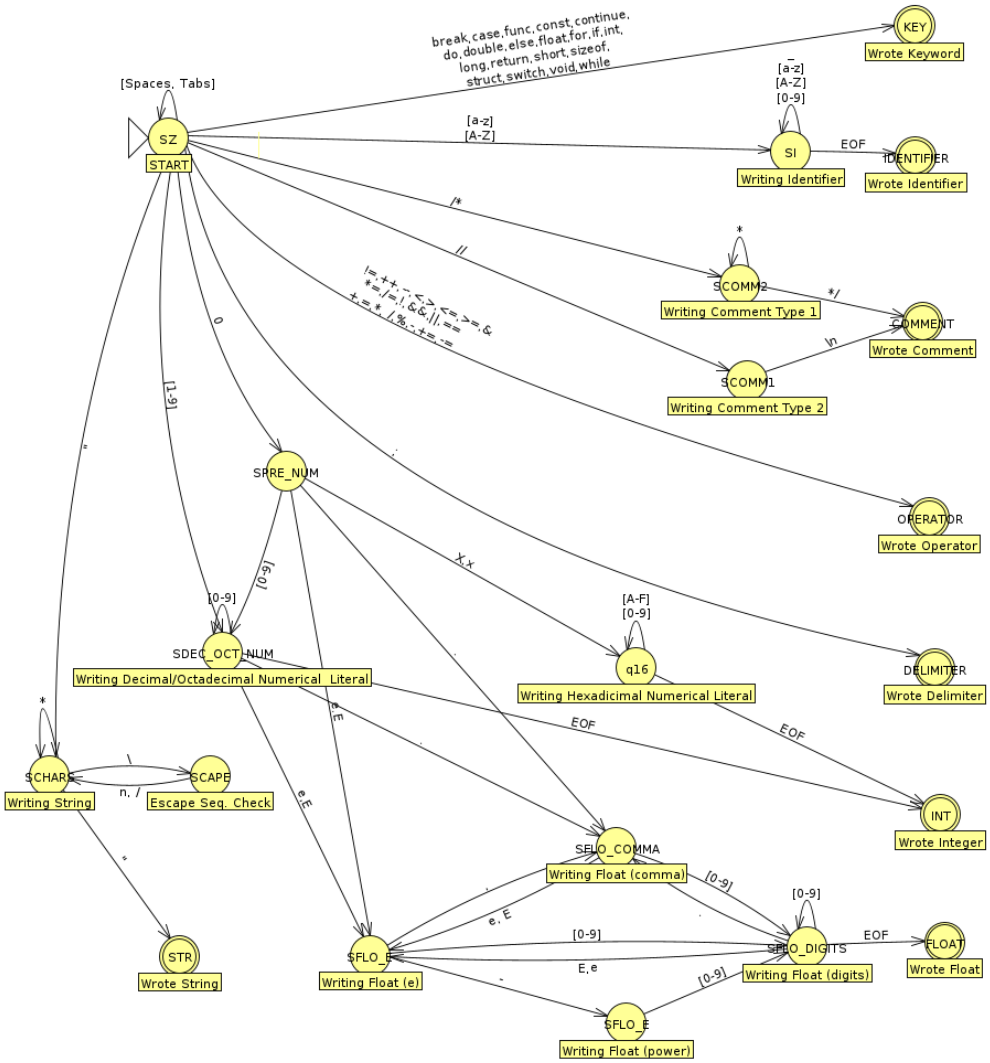
```
1  symbol         ::= "|" | " " | "!" | "#" | "$" | "%" | "&" | "(" | ")
       " | "*" | "+" | "," | "-" | "." | "/" | ":" | "\;" | ">" | "=" |
       "<" | "?" | "@" | "[" | "\" | "]" | "^" | "_" | "`" | "{" | "}" |
        "~"
2  letter         ::= "A" | "B" | "C" | "D" | "E" | "F" | "G" | "H" | "I
       " | "J" | "K" | "L" | "M" | "N" | "O" | "P" | "Q" | "R" | "S" | "
       T" | "U" | "V" | "W" | "X" | "Y" | "Z" | "a" | "b" | "c" | "d" |
       "e" | "f" | "g" | "h" | "i" | "j" | "k" | "l" | "m" | "n" | "o" |
        "p" | "q" | "r" | "s" | "t" | "u" | "v" | "w" | "x" | "y" | "z"
3  digit          ::= "0" | "1" | "2" | "3" | "4" | "5" | "6" | "7" |
       "8" | "9"
4  character      ::= letter | symbol | digit
5  character1     ::= character | "'"
6  character2     ::= character | '"'
7  text1          ::= "" | character1 text1
8  text2          ::= '' | character2 text2
9  literal        ::= '"' text1 '"' | "'" text2 "'"
10 identifier     ::= character text1
11 operator       ::= "+" | "-" | "*" | "/" | "%" | "=" | "+=" | "-=" |
       "*=" | "" | "" | "--" | "=" | "=" | "&" | "==" | "!=" | "++" |
       "&&" | "||"
12 number1        ::= digit number1 | "'"
13 number2        ::= digit number2 | '"'
14 number         ::= number1 | number2
15 integer        ::= number | "0X" number | "0x" number | "0"
16 float          ::= number "." number | number "e" number | number "e"
        number "-" number
17
```

3 Διαγράμματα Ενιαίου Αυτόματου

3.1 Γενικευμένο Διάγραμμα Ενιαίου Αυτόματου

Εδώ είναι το γενικευμένο ενιαίο αυτόματο που σχεδιάσαμε:

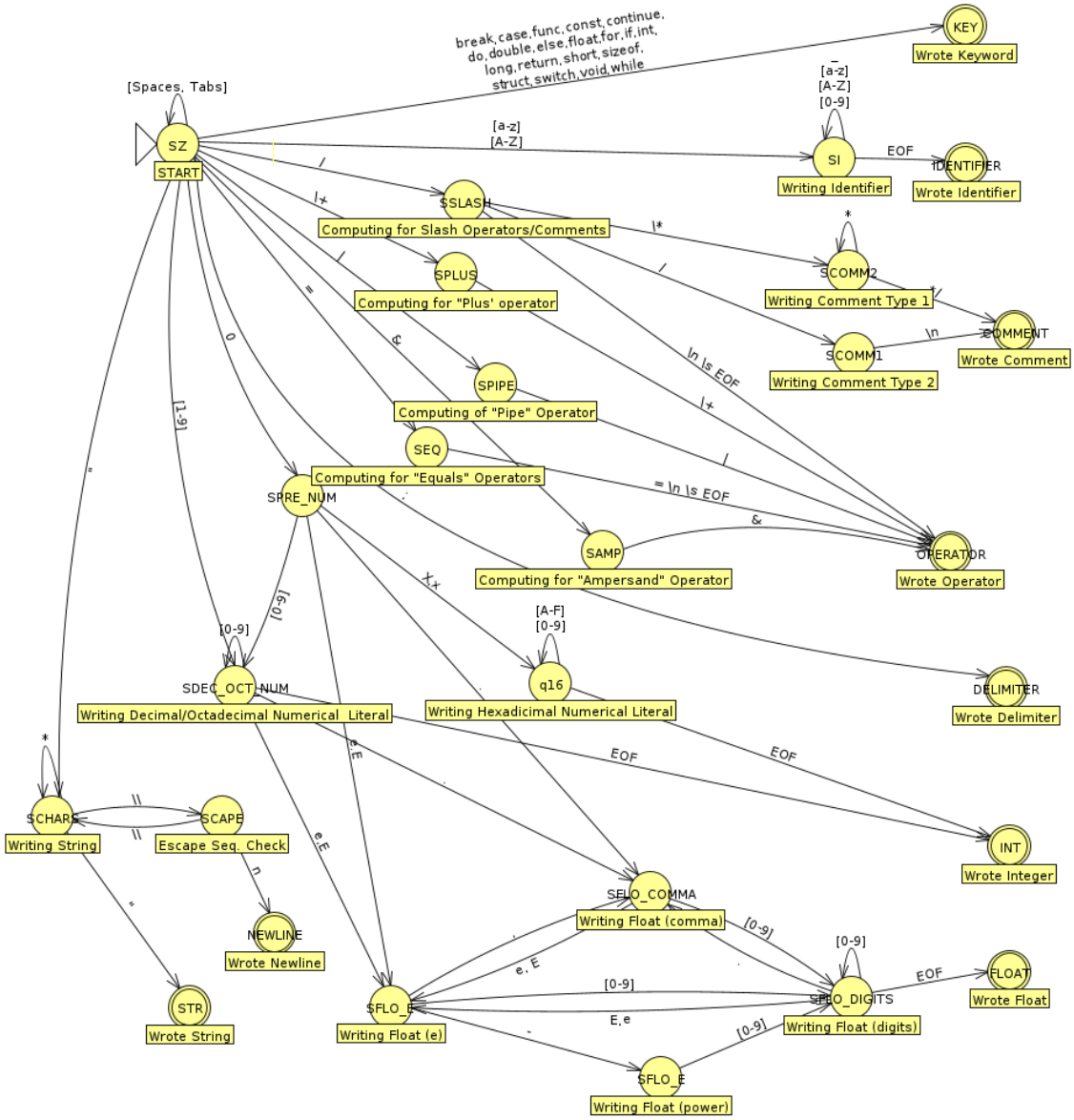
Σχήμα 1: Γενικευμένο Αυτόματο



3.2 Λεπτομερές Διάγραμμα Ενιαίου Αυτόματου

Εδώ είναι το λεπτομερές ενιαίο αυτόματο που σχεδιάσαμε:

Σχήμα 2: Λεπτομερές Αυτόματο



4 Εξήγηση Κώδικα σε FSM

Το πρώτο κομμάτι κώδικα σε FSM είναι, ασφαλώς, η αρχική μας κατάσταση:

```
START=SZ
SZ: % -> OPERATOR
    ! < > \- = \* -> SEQ
    \+ -> SPLUS
    & -> SAMP
    ; -> DELIMITER
    | -> SPIPE
    a-z A-Z _ -> SI
    \n \s -> SZ
    / -> SSLASH
    " -> SCHARS
    0 -> SPRE_NUM
    1-9 -> SDEC_OCT_NUM
    * -> BAD

```

Εδώ, βλέπουμε ότι ο αναλυτής μας περιμένει τον πρώτο, ουσιαστικά χαρακτήρα απο τον χρήστη, ο οποίος θα έχει μεγάλη σημασία στο να καθορίσουμε ¨τι πάει να γράψει ο χρήστης¨.

Βλέπουμε πολλές καταστάσεις, των οποίων η φιλοσοφία εξηγείται ως εξής.

- Οι καταστάσεις που το όνομα τους αρχίζει με "S" είναι για τον αναλυτή μας οι ¨μεταβατικές' μας καταστάσεις, οι καταστάσεις στις οποίες δεν έιναι απόλυτα σίγουρο το τι θέλει να γράψει ο χρήστης, αλλά είναι σίγουροι, με βάση κανόνες και ντετερμινιστικά οι ¨δρόμοι' που μπορεί να ακολουθήσει ο χρήστης.
- Οι υπόλοιπες καταστάσεις, καταστάσεις όπως η DELIMITER που βλέπουμε εδώ, εκφράζουν επιστροφή του αναγνωριστικού (και άρα επιτυχής αναγνώριση) της λεκτικής μονάδας (πχ εδώ το token (λεκτική μονάδα) είναι DELIMETER) καθώς ο αναλυτής μας είναι σίγουρος, *με βάση κανόνες που ακολουθεί ντετερμινιστικά*, ότι η λέξη η οποία ο χρήστης εισήγαγε είναι ενός συγκεκριμένου τύπου.

Συνεχίζοντας, βλέπουμε 2 καταστάσεις που αφορούν την αναγνώριση συμβολοσειράς, στις οποίες είναι δυνατόν να μεταβεί το αυτόματο, αν ο χρήστης εισάγει σε ορισμένες καταστάσεις τον χαρακτήρα ".

Αυτό ισχύει για την αρχική κατάσταση, όπως είδαμε παραπάνω, αλλά και για τις καταστάσεις IDENTIFIER, INT, FLOAT, αφού είναι πιθανό ο χρήστης να γράφει μια συμβολοσειρά έπειτα απο τους τύπους λέξεων αυτούς.

Η κατάσταση SCHARS συγκεκριμένα είναι αυτή στην οποία δίνονται οι χαρακτήρες της συμβολοσειράς, και αφήνεται ένα ενδεχόμενο είτε να τερματισθεί η συμβολοσειρά με το να εισαχθεί ξανά ο χαρακτήρας " (και να μεταβεί στην κατάσταση STR), είτε να να εισαχθεί ο χαρακτήρας \.

Στην τελευταία περίπτωση, ο αναλυτής έρχεται στην κατάσταση SCAPE, την οποία περιμένει να εισαχθέι είτε ο χαρακτήρας n, οπότε να επιστραφεί η κατάσταση NEWLINE, είτε να εισαχθεί ο χαρακτήρας \ξανά, οπότε να επιστρέψει στην κατάσταση SCHARS, είτε να κάνει λάθος και να εισαχθεί οποιοσδήποτε άλλος χαρακτήρας, οπότε να παέι στην κατάσταση BAD.

Στην κατάσταση NEWLINE, οποιοσδήποτε χαρακτήρας μας πάει πίσω στην κατάσταση SCHARS, ενώ ο χαρακτήρας " στην κατάσταση STR

```
1  SCHARS: * -> SCHARS
2          " -> STR
3          \\ -> SCAPE
4
5  SCAPE:  n -> NEWLINE
6          \\ -> SCHARS
7          * -> BAD
8
9  NEWLINE: " -> STRING
10           * -> SCHARS
11
12 STR:         \s \n -> SZ
13
```

Με παρόμοια λογική βλέπουμε διάφορους χαρακτήρες που είναι *δυνητικά* τελεστές (ο πρώτος είναι δυνητικά και σχόλιο).

```
SSLASH: / -> SCOMM1
        = \n \s -> OPERATOR
        \* -> SCOMM2
         * -> BAD

SEQ:     = \n \s -> OPERATOR
         * -> BAD

SPLUS:   = \+  -> OPERATOR
         * -> BAD

SAMP:    & \s  -> OPERATOR
         * -> BAD

SPIPE:   |     -> OPERATOR
         * -> BAD

```

Επίσης, εδώ βλέπουμε, με παρόμοιο συλλογισμό με τις καταστάσεις για τις συμβολοσειρές, και την επεξεργασία για τα σχόλια (με τελική κατάσταση την κατάσταση COMMENT)

```
SCOMM1:     * -> SCOMM1
           \n -> SZ

SCOMM2:    \* -> SCOMM21
           * -> SCOMM2

SCOMM21:    / -> COMMENT
            * -> SCOMM2

COMMENT:    \n \s -> SZ

```

Παρατηρούμε και την κατάσταση SI, η οποία δέχεται τους επιτρεπτούς χαρακτήρες ως κομμάτια της πιθανής λέξης ενός identifier, που με τον χαρακτηρα ;, με κενό, ή με τέλος αρχείου γίνεται η μεταβολή του αναλυτή στην κατάσταση IDENTIFIER.

```
SI:      a-z A-Z 0-9 -> SI
         ;           -> IDENTIFIER
         \s          -> IDENTIFIER
         EOF         -> IDENTIFIER
```

```
5           * -> BAD
6
```

Εδώ βλέπουμε τις ¨τελικές' καταστάσεις στις οποίες φτάνει ο αναλυτής οταν κρίνει μια λέξη να πληρεί τις προϋποθέσεις για να χαρακτηρισθεί ως μια συγκεκριμένη λεκτική μονάδα.
Παρατηρούμε ότι, με βάση την γλώσσα Uni-C, οι λεκτικές μονάδες είναι δυνατό να ακολουθηθούν απο συγκεκριμένες λεκτικές μονάδες (παράδειγμα οι τελεστές μονάχα από ένα νούμερο ή identifier, για αυτό οι μοναδικές επιτρεπτές καταστάσεις που μεταβαίνει ο αναλυτής μετά απο την κατάσταση OPERATOR είναι η ενδιάμεση κατάσταση έιτε αριθμού έιτε identifier)

```
1  OPERATOR:    a-z A-Z _ -> SI
2               0 -> SPRE_NUM
3               1-9 -> SDEC_OCT_NUM
4               " -> SCHARS
5               * -> BAD
6
7  DELIMITER:   * -> SZ
8  INT:         \s \n -> SZ
9               A-Z a-z _ -> SI
10              % -> OPERATOR
11              ! < > \- = \* -> SEQ
12              \+ -> SPLUS
13              & -> SAMP
14              ; -> DELIMITER
15              | -> SPIPE
16              / -> SSLASH
17              0 -> SPRE_NUM
18              1-9 -> SDEC_OCT_NUM
19              * -> BAD
20              " -> SCHARS
21
22 FLOAT:       \s \n -> SZ
23              A-Z a-z _ -> SI
24              % -> OPERATOR
25              ! < > \- = \* -> SEQ
26              \+ -> SPLUS
27              & -> SAMP
28              ; -> DELIMITER
29              | -> SPIPE
30              / -> SSLASH
31              0 -> SPRE_NUM
32              1-9 -> SDEC_OCT_NUM
```

```
33              * -> BAD
34
35  IDENTIFIER: \s \n -> SZ
36              A-Z a-z _ -> SI
37              % -> OPERATOR
38              ! < > \- = \* -> SEQ
39              \+ -> SPLUS
40              & -> SAMP
41              ; -> DELIMITER
42              | -> SPIPE
43              / -> SSLASH
44              0 -> SPRE_NUM
45              1-9 -> SDEC_OCT_NUM
46              * -> BAD
47              " -> SCHARS
48
49
50
```

Τέλος, έχουμε τις ενδιάμεσες καταστάσεις με τις οποίες ο αναλυτής κρίνει αν ο αριθμός που εισάγεται είναι πραγματικός ή ακέραιος.
Με βάση λοιπόν την σύνταξη της Uni-C, διαχωρίζονται αυτές οι δυό κατηγορίες.
(πχ. ακέραιοι: 0, 954, 4235 πχ. πραγματικοί: 3.14 2.43E31, 35.3e10, 34e12, 0e0, 0.2222)

```
SPRE_NUM:    x X -> SHEX
             0-9 -> SDEC_OCT_NUM
             \. ->  SFLO1
             e E -> SFLO2
             * -> BAD

SDEC_OCT_NUM:   0-9      -> SDEC_OCT_NUM
                \.       -> SFLO_COMMA
                e E      -> SFLO_E
                \s \n ; -> FLOAT
                * -> BAD

SFLO_COMMA:     \s \n ; -> FLOAT
                e E      -> SFLO_E
                0-9      -> SFLO_DIGITS
                * -> BAD

SFLO_DIGITS:    0-9      -> SFLO_DIGITS
                \n \s ; -> FLOAT
                e E      -> SFLO_E
                \.       -> SFLO_COMMA
                * -> BAD

SFLO_E:         \.        -> SFLO_COMMA
                \-        -> SFLO_POWER
                0-9      -> SFLO_DIGITS
                * -> BAD

SFLO_POWER:     0-9      -> SFLO_DIGITS
                * -> BAD

SHEX:        A-F      -> SHEX
             0-9      -> SHEX
             \s \n ; -> INT
             * -> BAD

```

5 Ολοκληρωμένος Κώδικας σε FSM

Ο κώδικας του αυτοματού μας μπορεί να βρεθεί ως αρχείο και στον φάκελο *Source_Code* που βρήκεται μέσα στον κύριο φάκελο του παραδοτέου.
Ακολουθεί ο ολοκληρωμένος κώδικας σε FSM του αυτόματού μας:

```
START=SZ
SZ: % -> OPERATOR
    ! < > \- = \* -> SEQ
    \+ -> SPLUS
    & -> SAMP
    ; -> DELIMITER
    | -> SPIPE
    a-z A-Z _ -> SI
    \n \s -> SZ
    / -> SSLASH
    " -> SCHARS
    0 -> SPRE_NUM
    1-9 -> SDEC_OCT_NUM
    * -> BAD


SCHARS: * -> SCHARS
        " -> STR
        \\ -> SCAPE

SCAPE:  n -> NEWLINE
        \\ -> SCHARS
        * -> BAD

NEWLINE: " -> STRING
         * -> SCHARS

SSLASH: / -> SCOMM1
        = \n \s -> OPERATOR
        \* -> SCOMM2

SEQ:     = \n \s -> OPERATOR
        * -> BAD

SPLUS:   = \+  -> OPERATOR
        * -> BAD

SAMP:    & \s  -> OPERATOR
```

```
39          * -> BAD
40
41  SPIPE:    |      -> OPERATOR
42          * -> BAD
43
44  SI:       a-z A-Z 0-9 -> SI
45            ;           -> IDENTIFIER
46            \s          -> IDENTIFIER
47            EOF         -> IDENTIFIER
48          * -> BAD
49
50  SCOMM1:     * -> SCOMM1
51             \n -> SZ
52
53  SCOMM2:     \* -> SCOMM21
54              * -> SCOMM2
55
56  SCOMM21:    / -> COMMENT
57              * -> SCOMM2
58
59  COMMENT:    \n \s -> SZ
60
61  OPERATOR:   a-z A-Z _ -> SI
62              0 -> SPRE_NUM
63              1-9 -> SDEC_OCT_NUM
64              " -> SCHARS
65              * -> BAD
66
67  DELIMITER:  * -> SZ
68
69  SPRE_NUM:   x X -> SHEX
70              0-9 -> SDEC_OCT_NUM
71              \. ->  SFLO1
72              e E -> SFLO2
73              * -> BAD
74
75  SDEC_OCT_NUM:   0-9     -> SDEC_OCT_NUM
76                  \.      -> SFLO_COMMA
77                  e E     -> SFLO_E
78                  \s \n ; -> FLOAT
79                  * -> BAD
80
81  SFLO_COMMA:     \s \n ; -> FLOAT
82                  e E     -> SFLO_E
```

```
                0-9     -> SFLO_DIGITS
                * -> BAD

SFLO_DIGITS:    0-9     -> SFLO_DIGITS
                \n \s ; -> FLOAT
                e E     -> SFLO_E
                \.      -> SFLO_COMMA
                * -> BAD

SFLO_E:         \.       -> SFLO_COMMA
                \-       -> SFLO_POWER
                0-9     -> SFLO_DIGITS
                * -> BAD

SFLO_POWER:     0-9     -> SFLO_DIGITS
                * -> BAD

SHEX:       A-F     -> SHEX
            0-9     -> SHEX
            \s \n ; -> INT
            * -> BAD

INT:        \s \n -> SZ
            A-Z a-z _ -> SI
            % -> OPERATOR
            ! < > \- = \* -> SEQ
            \+ -> SPLUS
            & -> SAMP
            ; -> DELIMITER
            | -> SPIPE
            / -> SSLASH
            0 -> SPRE_NUM
            1-9 -> SDEC_OCT_NUM
            * -> BAD
            " -> SCHARS

FLOAT:      \s \n -> SZ
            A-Z a-z _ -> SI
            % -> OPERATOR
            ! < > \- = \* -> SEQ
            \+ -> SPLUS
            & -> SAMP
            ; -> DELIMITER
            | -> SPIPE
```

```
127             / -> SSLASH
128             0 -> SPRE_NUM
129             1-9 -> SDEC_OCT_NUM
130             * -> BAD
131
132 IDENTIFIER: \s \n -> SZ
133             A-Z a-z _ -> SI
134             % -> OPERATOR
135             ! < > \- = \* -> SEQ
136             \+ -> SPLUS
137             & -> SAMP
138             ; -> DELIMITER
139             | -> SPIPE
140             / -> SSLASH
141             0 -> SPRE_NUM
142             1-9 -> SDEC_OCT_NUM
143             * -> BAD
144             " -> SCHARS
145
146 STR:        \s \n -> SZ
147
148 BAD(OK): \n -> SZ
149 GOOD(OK): \n -> GOOD
150
151
```