CMPSC 4473: Theory of Programming Languages

Lexical Analyzer: As mentioned previously, in the lexical analysis phase of a compiler, a *lexical analyzer* reads input, one character at a time, and groups a sequence of one or more characters into an atomic unit called a *token*. Each token is passed to the parser which groups tokens together into a syntactic structure such as arithmetic expression. For example, the input

distance = 54.7 * time;

is read by a lexical analyzer which breaks it into the tokens

(distance, VAR)	(=, ASSIGN)	(54.7, CONST)
(*, ARITHOP)	(time, VAR)	(; TERM)

where VAR denotes a variable, ASSIGN denotes an assignment operator, ARITHOP denotes an arithmetical operator, CONST denotes a constant and TERM represents a terminal symbol. The tokens are passed to the parser, which groups them into syntactic structures. In this case

(54.7, CONST)   (*, ARITHOP)   (time,VAR)

form the syntactic structure (↑, EXPR), which denotes an expression, and

(distance, VAR)   (=, ASSIGN)   (↑, EXPR)

form the syntactic structure assignment statement. The symbol ↑ denotes a pointer to the syntactic structure expression.

Transition Diagram: A *transition diagram*, TD, that looks similar to a flowchart, can be useful for describing how a lexical analyzer groups symbols into a token.

Example 1: As an example, consider a TD that recognizes an *identifier* or *variable*. Assume a variable is a sequence of characters that begins with letter (lower or upper case) and is followed by zero or more characters which can be a letter or a digit. The end of an identifier is recognized when a character that is not a letter or digit is read (this character is not part of the identifier but serves to determine when an identifier ends).

It is convenient to use some set theory notation. Let L be the set of letters (lower or upper case) and let D be the set of digits. That is, let

$$L=\{a,b,c,\ldots,x,y,z,A,B,C,\ldots X,Y,Z\}$$

and let

$$D=\{0,1,2,3,4,5,6,7,8,9\} \quad .$$

The notation $\alpha\in X$ means element α is in set X and the notation $\alpha\notin X$ means element α is not in set X. For example, $\beta\in L$ means symbol β is a letter (lower or upper case) and $\gamma\notin D$ means symbol γ is not a digit.

TD for an Identifier: Figure 1, below, shows a TD that recognizes an identifier (variable).

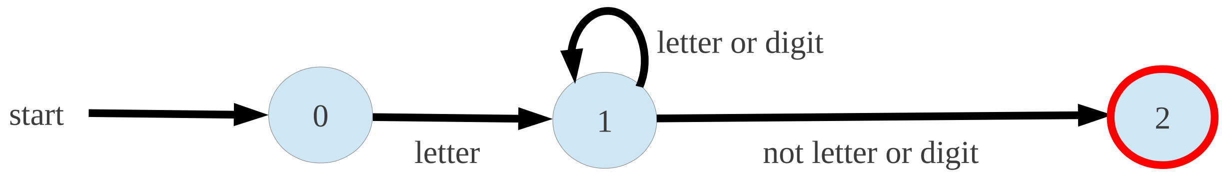

Figure 1: TD for an Identifier

Example 2: Consider a TD for a label that begins with an underscore '_' character, followed by 'L', followed by zero or more digits. Figure 2 shows a TD that recognizes a label.

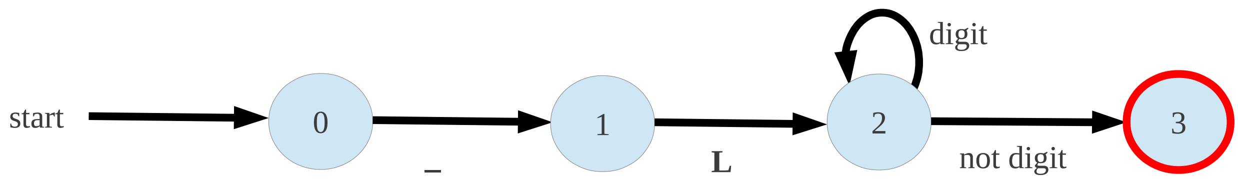

Figure 2: TD for a Label

Exercise 1: Consider a "vowel" identifier which starts with a lower case vowel and is followed by zero or more lower case vowels or digits. Let

$$V=\{a,e,i,o,u\}$$

be the set of lower case vowels. Draw the TD that recognizes vowel identifiers.

Exercise 2: Consider whole numbers that start with an *odd* digit, followed by zero or more digits and end with an *even* digit. Let

$$O=\{1,3,5,7,9\}$$

represent the set of odd digits and let

$$E=\{0,2,4,6,8\}$$

represent the set of even digits. As above, let D represent the set of digits. Draw the TD that recognizes these kind of whole numbers (assume the numbers are not preceded by a + or – sign).

Alphabet: A set of symbols is called an *alphabet*.

Example 3: The set of lower case vowels $V=\{a,e,i,o,u\}$ is an example of an alphabet.

Example 4: The set of binary digits $B=\{0,1\}$ is an example of an alphabet.

Exercise 3: Use set notation to specify the alphabet of symbols consisting of either an even digit or an upper case vowel.

Exercise 4: Use set notation to specify the alphabet of symbols consisting of lower case consonants.

String (word, sentence): A finite sequence of zero or more symbols from an alphabet is called a *string* or *word* or *sentence*. If x is a string the number of symbols in x is called the *length* of x and is denoted as $|x|$. The string consisting of zero symbols is called the *empty* string and the symbol ε is used to denote the empty string.

Example 5: The string 01001 can be produced from the alphabet B of binary digits or the alphabet D of digits but not from the alphabet E of even digits.

Exercise 5: Let $V=\{a,e,i,o,u\}$ be the set of lower case vowels, $E=\{0,2,4,6,8\}$ be the set of even digits, $O=\{1,3,5,7,9\}$ be the set of odd digits, $L=\{a,b,c,\ldots,x,y,z\}$ be the set of lower case letters, $U=\{A,B,C,\ldots X,Y,Z\}$ be the set of upper case letters and $B=\{0,1\}$ be the set of binary digits. Let x represent a character. If S is one of the sets V, E, O, L, U or B then x in S means x belongs to S and x not in S means x does not belong to S. Figure 3 is a *Transition Diagram*, TD using some of these sets.

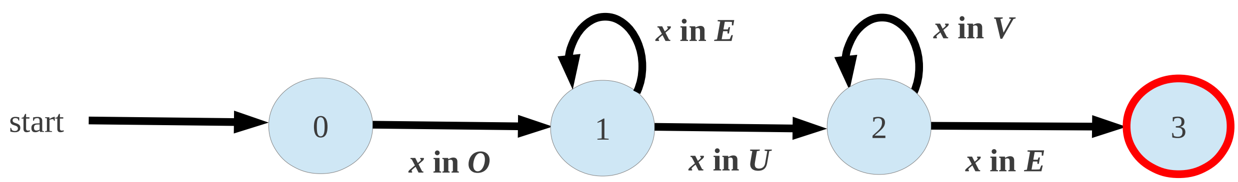

Figure 3: TD for Exercise 5

Determine which of the following strings is *recognized* by the TD in figure 3. A string is recognized if you can reach the *stop state*, state 3, in figure 3. If a string is not recognized, can you get "stuck" in a state that is not the stop state? If so, what *state* are you stuck in?

```
a) 9820Xae6
b) 8929Xae6
c) 5Wiu4
d) 76204Ru988
e) 3Y2
f) 966iua4
```

Language: A *language* L is a set of strings, including the *empty* set which is denoted by ∅.

Example 6: The set E of all English sentences is an example of a language. The set C of all valid C programs is a language. The set $L = \{ \varepsilon \}$ consisting of the empty string is a language.

Example 7: Consider the set of binary digits $B=\{0,1\}$ as an example of an alphabet, given in example 4. Set E consists of all computer "words" having exactly 32 bits. Each bit contains either the binary digit 0 or the binary digit 1. Set M consists of all computer "words" having *at most* 32 bits. Set L consists of all computer "words" having *at least* 32 bits. How many strings are in E, in M and in L?