CS480
Translators
Finishing Lexical Analysis
Chap. 3

Lexical-Analyzer Generator

What does Lex do?

- A program that takes regular expressions and action pairs, and builds a recognizer.
- Why actions?
Using Lex

1. **LEX** makes NO assumptions about tokens, must recognize ALL of input, including spaces, comments even illegal symbols (\' default action).

2. **Multiple rules can match the same input.**

   Ambiguity is resolved using the following two rules:
   - A) choose pattern which matches the longest input string
   - B) if two patterns match the same size string, choose that which was listed first.

```c
/* definitions of special constants */
t, le, eq, ge, lt, gt,
if, then, else, do, number, help */

/* regular definitions */
#define \n ( \n\n )
#define \a ( \a-\n )
#define \d ( [0-9] )
#define \l ( \l(letter)(\d(digit)*)
#define \n ( digit+)\.\+digit+[0-9]*\d[\l]*/

int installID() { /* function to install the lexeme, whose first character is pointed to by yylval, and whose length is yyleng, into the symbol table and return a pointer thereto */
}

int installNum() { /* similar to installID, but puts numerical constants into a separate table */
}
```

Figure 3.23: Lex program for the tokens of Fig. 3.12
Quiz #3

• Convert the NFA to DFA

• Write a finite state automata that will recognize any string consisting of a and b characters where the number of a’s is even (or zero) and the number of b’s is odd.