Lexical Analysis - Scanner-Contd

66.648 Compiler Design Lecture 4(01/26/98)

Computer Science
Rensselaer Polytechnic

Lecture Outline

- **More on Lex**
- **Examples and Applications**
- **Administration**

LEX

Input to Lex consists of three parts, separated by lines beginning with %%:

first part
%%
pattern action
%%
third part

first and third parts are optional.

LEX- Contd

The first part contains the dimensions of certain tables internal to lex - also may contain definitions of text replacements. It can also contain global C code preceded by a line beginning %{ and ending with %}

The third part contains C code which is used as us. It usually contains functions which the second part uses.

The first separator (%%) is essential, whereas the second separator (%%) is not needed if the third part is empty.

Patterns

Letters, digits and some special characters represent themselves.

Period (.) represents any character other than line feed (\n)

Brackets ([and]) enclose a sequence of characters, called a character class. The class represents any one of its member or any single character not in the class, if the class starts with ^. Within the sequence, - between two characters denotes the inclusive range.

IF * follows one of the pattern parts, then the corresponding input may appear 0 or more times.

Patterns Counted

^ at the beginning of a pattern represents the beginning of an input line.

$ at the end of a pattern represents the end of the input line.

\ is used as escape character.

“ “ represent for a string of patterns.

Examples

```
“for”                              reserved word for
“--”                               decrement operator
[A-AA-z_][A-Sea-z0-9_]*            C identifiers
“/*”.*”*/”                         Single line comments
“//”.*                             C++ comments
[0-9][0-9]*                        Integer constants
“/*”([^*/]|[^*]”/”|”*”[^/])*”*/”   C Comments over many lin
\”([^”\n]|\\[“\n])*\”              Strings
```

Ambiguities

Lex always chooses the pattern which represents the longest possible input string.

If two patterns represent the same string, the first pattern in the list presented to lex is chosen.

use:
int
[a-z]+

Sample Lex Programs

```
1) %{
/* Remove uppercase letters . Commands to execute are
lex test.l and gcc lex.yy.c -ll -o test */
%}
%%
[A-Z]+          ;
2) %{
/* Line numbering */
%}
%%
^.*\n                    printf(“%d\t%s”,yylineno-1,yytext);
```

Sample Lex Programs contd

```
%{
/* unix utility wc simulated. counts chars words and lines*/
%}
int nchar,nword,nlines;
%%
\n        nchar++;nlines++;
[^ \t\n]+         {nword++;nchar+=yyleng; /*yyleng gives the
                   length of the pattern*/}
.                  nchar++;
%%
void  main(void)
{ yylex();
printf(“%d\t%d\t%d\n”,nchar,nword,nlines);
}
```

Applications

Pattern Matching Problem:
Given a pattern string p and a subject string s, find out whether p appears in s as a substring.
This is an important search problem.
See Exercises 3.26 and 3.27.
The trick is to avoid O(|p|*|s|) algorithm.

Applications-contd

Construct a DFA for the pattern. The back-transitions are constructed using failure functions.

e.g., pattern string is: a b a b a a.

Applications - Contd

Compute the edit distance between two given strings x and y.
The edit operations that are allowed : insert, delete and update.
(See exercise 3.35)
e.g., if two strings are rational and nation, the edit distance will be 3.

Applications - Contd

A Dynamic Programming algorithm can be used to compute edit distance.
Let D[i,j] be the edit distance between x_1,…x_i and y_1,…,y_j.
D[i,j]= min{ D[i-1,j-1]+replac(x_i,y_j),
D[i-1,j]+1,D[i,j-1]+1}

Administration

- We have finished Chapter 3 of Aho, Sethi and Ullman's book. Please read that chapter and chapter 1 which we covered in Lectures1 and 2.
- Work out the unstarred exercises of chapter 3.
- Lex and Yacc Manuals are handed out. Please read them.

First Project is in the web.

It consists of three parts.
1) To write a lex program
2) To write a YACC program.
3) To write five sample Java programs. They can be either applets or application programs

Comments andFeedback

- Please let me know if you have not found a project partner.
- A sample Java compiler is in the class home page.