CS480

Lexical Analysis

Ch 3 p83-113

February 11, 2013

How do they all fit together?

- Source Code
- Lexical Analyzer
- Parser
- Symbol Table

- Purpose of the Lexical analyzer?
- Purpose of the parser?
- Purpose of the ST?

- Why separate them?

CS310 Redux

- Alphabet
- String over some alphabet
- Empty string
- Language
  - Regular
  - non-regular
- Union
- Concatenation
- Kleene closure
  - Positive closure
- Regular expression

Notations?

Usage

- Use above to define valid identifiers
  - In C
  - In our lexer

Practice

- -?[0-9]$^+$
- Integer Numbers ? Real Numbers?
- All strings where every occurrence of a is followed immediately by a single occurrence of b over the alphabet {a,b}
  - RE/DFA? (or prove you can't!)
- Real numbers where the number of digits on the left of the decimal point is equal to the number of digits on the right.
  - RE? (or prove you can't!)

Definitions

- Token

- Pattern

- Lexeme

- How are these related?
- Which part of the complier deals with which?

In Action

- Our Lexer will return (token, value) tuples

```
sum = 2 + sum - num--;
```

Lexeme	Token	Value

Error Handling

- What errors can arise during lexing?

```
fi(vals == nums[i]);
// what else makes this hard to parse?

if (fi == 9) ;

if while ( x == 0 );
```

Error Handling

- Continue processing until a valid token is found
- Delete extraneous characters

- Insert missing characters
- Replace what appears to be incorrect characters
- If it makes sense, transpose two adjacent characters

What errors can the ***lexer*** produce?

Character Not In Grammar.
Missing Semicolon.
Missing Right Parenthesis.
Missing Left Parenthesis.
Missing Right Brace.
Missing Left Brace.
Missing Right Bracket.
Missing Left Bracket.
Identifier Expected.
Constant Expected.
Main Declaration Expected.
Invalid Declaration.
Read Past EOF.
Bad Expression.
Duplicate Identifier.
Undeclared Identifier.
Undeclared Function.

Identifier Not Right Type.
Undeclared Array.
Mismatched Parameters.
Unary Type Mismatch.
Addop Type Mismatch.
Mulop Type Mismatch.
Dereference Type Mismatch.
Assign Type Mismatch.
Invalid Identifier.
Constant Too Long.
Bad Statement.
Extra Tokens.
No More Tokens.
Cannot Open File.
Out Of Memory.
Missing Comma.

Lexer Implementation

- We might use (f)lex
- Write the code in a high level language using the I/O provided by the language
- Write the code in assembly managing the I/O explicitly

(f)lex

```
%{
  /* sample demonstration */
  /* identify is and are as verbs */
%}

%%
[\t ]+      /* ignore whitespace */
is |
are         { printf("%s: is a verb", yytext);
              return(VERB);}
[a-zA-Z]+   { printf("%s: is not a verb", yytext); }
%%

main()
{
  yylex();
}
```

quiz.h

```
double score (int correct);
char letter(double grade);
```

quiz.c

```
static const double MAX_Q = 9.0;
double score (int correct)
{
  Return correct / MAX_Q;
}
```

```
gcc -c quiz.c -o quiz.o -g
gcc -c main.c -o main.o -g
gcc -g main.o quiz.o -o quiz
```

main.c

```
#include <stdio.h>
#include “quiz.h”
int main()
{
  double g = score(4);
  printf(“%c\n”, letter(g));
  return 0;
}
```

Implementation

- How do you identify a token?

- Double buffers

- Transition Diagram (DFA)

Source