CSE302: Compiler Design

Instructor: Dr. Liang Cheng
Department of Computer Science and Engineering
P.C. Rossin College of Engineering & Applied Science
Lehigh University

February 15, 2007

Outline

- Recap
  - The lexical-analyzer generator Lex
- Implementing lexical-analyzer generators
- Summary and homework

Implementing Lexical-Analyzer Generators

- Regular expressions $\rightarrow$ Nondeterministic finite automata
- Nondeterministic finite automata $\rightarrow$ Deterministic finite automata
- Deterministic finite automata $\rightarrow$ A lexer

- Regular expressions $\rightarrow$ Deterministic finite automata
- Deterministic finite automata $\rightarrow$ A lexer

MYT Algorithm

- Constructing an NFA from a regular expression ***r*** by McNaughton-Yamada-Thompson algorithm
  - Organizing ***r*** into its constituent sub-expressions (parse tree)
    - Sub-expressions with no operators
    - Operators
  - Using basic rules to construct NFA for sub-expressions with no operators
  - Using inductive rules to construct larger NFA based on the constructed NFA for operations of sub-expressions

An Example: **(a|b)*abb**

Another Example

- Form the NFA for the regular expression ***letter*(*letter*|*digit*)***

Implementing Lexical-Analyzer Generators

- Regular expressions → Nondeterministic finite automata
- Nondeterministic finite automata → Deterministic finite automata
- Deterministic finite automata → A lexer

Conversion of NFA to DFA

- Subset construction algorithm
  - Input: An NFA *N*
  - Output: A DFA *D* accepting the same language as *N*
  - Algorithm: construct a transition table Dtran corresponding to D

```
Initially, ε-closure(s0) is the only state in Dstates, and it is unmarked;
while ( there is an unmarked state T in Dstates ) {
    mark T;
    for ( each input symbol a) {
      U = ε-closure(move(T,a));
      if ( U is not in Dstates ) add U as an unmarked state to Dstates;
      Dtran[T,a] = U;
    }
}
```

ε-closure(*s*) and ε-closure(T)

- ε-closure(*s*): a set of NFA states reachable from NFA state *s* on ε-transitions alone
- ε-closure(T): a set of NFA states reachable from some NFA state s in the set T on ε-transitions alone
  - $\cup_{s \text{ in } T}$ ε-closure(*s*)

```
push all states of T onto stack;
initialize ε-closure(T) to T;
while ( stack is not empty ) {
   pop t, the top element, off stack;
   for ( each state u with an edge from t to u labeled ε)
     if ( u is not in ε-closure(T) ) {
            add u to ε-closure(T);    push u onto stack;
     }
}
```

move(T,*a*)

- A set of NFA states to which there is a transition on input symbol a from some state *s* in T

Conversion of An NFA Accepting **(a|b)*abb** to A DFA

- Draw the state transition diagram

Another Example

- Convert the NFA for the regular expression ***letter*(*letter*|*digit*)*** * to a DFA

Simulation of An NFA

- An input string x terminated by **eof**. An NFA N with a start state s_0, accepting states F, and ε-closure() and move() functions.

S = ε-closure(s_0);
c = nextChar();
while (*c*! = **eof**) {
    S=ε-closure(move(S,*c*)); *c*=nextChar();
}
if (S $\cap$ F != $\varnothing$) **return** "yes";
else return "no";

Outline

- Recap
- Implementing lexical-analyzer generators
- Summary and homework

Flex

- Fast lexical analyzer generator

Conversion of NFA to DFA

- Subset construction algorithm
  - Input: An NFA *N*
  - Output: A DFA *D* accepting the same language as *N*
  - Algorithm: construct a transition table Dtran corresponding to D

```
Initially, ε-closure(s0) is the only state in Dstates, and it is unmarked;
while ( there is an unmarked state T in Dstates ) {
    mark T;
    for ( each input symbol a) {
      U = ε-closure(move(T,a));
      if ( U is not in Dstates ) add U as an unmarked state to Dstates;
      Dtran[T,a] = U;
    }
}
```

Reading Assignment

- For today's class
  - Sections 3.7 and 3.8
- For next Tuesday's class
  - Chapter 4

Homework (Due on 02/19 at 11:55 PM)

- 5.1. (10 points). Using flex and based on the Example 3.8 (pages 128-129 in the textbook), generate a lexer that scans the following input stream and outputs the following output stream.
  - Input stream: if i>0 then i=1 else i=0
  - Output stream: IF ID:i RELOP:GT NUMBER:0 THEN ID:i RELOP:EQ NUMBER:1 ELSE ID:i RELOP:EQ NUMBER:0
    Please provide a readme file explainig how you generate and test your lexer.
- 5.2. (10 points) Convert the NFA for the regular expression ***letter***(***letter***|***digit***)* to a DFA.