编译原理/Compiler
Lexical Analysis

周雅倩
复旦大学计算机科学技术学院
{zhouyaqian}@fudan.edu.cn
2018/10/11

复旦大学媒体计算研究所

References

- Partial contents are copied from the slides of Alex Aiken
  - http://www.stanford.edu/class/cs143/
  - https://www.coursera.org/course/compilers
- http://en.wikipedia.org/wiki/Comparison_of_parser_generators

2

复旦大学媒体计算研究所

Outline

- Specifying lexical structure using regular expressions
- Finite automata
  - Deterministic Finite Automata (DFAs)
  - Non-deterministic Finite Automata (NFAs)
- Implementation of regular expressions
  - RegExp => NFA => DFA => Tables

复旦大学媒体计算研究所

Outline

- 语言（Language）
- 正则表达式（Regular Expressions）
- 有限自动机（Finite Automata）

复旦大学媒体计算研究所

Definitions

- The lexical analyzer （词法分析器）produces a certain token （单词） wherever the input contains a string of characters in a certain set of strings.
- The set of strings is described by a rule（规则）.
- A pattern （模式） is associated with a token.
- A lexeme（词素） is a sequence of characters matching the pattern.

复旦大学媒体计算研究所

Token	Sample lexemes	Informal description of pattern
const	const	const
if	if	if
relation	<,<=,=,<>,>,>=	< or <= or = or <> or > or >=
id	pi, count, D2	Letter followed byletters or digits
num	3.1416,0, 6.02E23	Any numeric constant
literal	“core dumped”	Any characters between “and” except “”

复旦大学媒体计算研究所

Some Language aspects

- Fortran requires certain constructs in certain positions of input line, complicating lexical analysis.
- Modern languages – free-form input
  - Position in an input line is not important.

复旦大学媒体计算研究所

Some Language aspects

- Sometimes blanks are allowed within lexemes. For e.g. in Fortran X VEL and XVEL represent the same variable.

  e.g. DO 5 I = 1,25 is a Do- statement while
  DO 5 I = 1.25 is an assign statement.

复旦大学媒体计算研究所

Lexical Analysis in PL/I

- Most languages reserve keywords. Some languages do not, thus complicating lexical analysis.
  - e.g. PL/I

    IF THEN THEN THEN = ELSE; ELSE ELSE = THEN;
- PL/I Declarations:

  DECLARE (ARG_1,. . ., ARG_N)
- Can't tell whether DECLARE is a keyword or array reference until after the).
  - Requires arbitrary look ahead!

复旦大学媒体计算研究所

Lexical Analysis in C++

- Unfortunately, the problems continue today
- C++ template syntax:

  Foo<Bar>
- C++ stream syntax:

  cin >> var;
- But there is a conflict with nested templates:

  Foo<Bar<Bazz>>

复旦大学媒体计算研究所

Review

- The goal of lexical analysis is to
  - Partition the input string into lexemes
  - Identify the token of each lexeme
- Left-to-right scan => lookahead sometimes required

复旦大学媒体计算研究所

Attributes of Tokens

- If two or more lexemes match the pattern for a token then the lexical analyzer must provide additional information with the token.
- Additional information is placed in a symbol-table entry and the lexical analyzer passes a pointer/reference to this entry.
- E.g. The Fortran statement:

E = M * C ** 2

It has 7 tokens and associated attribute-values:

<id, reference to symbol-table entry for E>
<assign_op, >
< id, reference to symbol-table entry for M>
<mult_op, >
< id, reference to symbol-table entry for C>
<exp_op, >
<num, integer value 2>

复旦大学媒体计算研究所

Sample of objects in Lex

Token
-index (type) : int -stringVal : string(idl) -numVal : float(idl) -line : unsigned long(idl) -charBegin : unsigned short(idl) -charEnd

SymbolTable
+insert() : Token +delete() : Token +query() : Token

复旦大学媒体计算研究所

Errors in Lex

- Unmatched pattern
  - Simplest -> Panic mode
- Recovery and continue on
- Many times, lex has only localized view
- E.g. fi (a == f(x)) ……

复旦大学媒体计算研究所

Input Buffering

- Buffer pairs: The input buffer has two halves with *N* characters in each half.
- *N* might be the size of a disk block like 1024 or 4096.
- Mark the end of the input stream with a special character eof. Maintain two pointers into the buffer marking the beginning and end of the current lexeme.

```
E  =  M  *  |  C  *  *  2  eof
            ↑    ↑
            |    forward
      lexeme_start
```

复旦大学媒体计算研究所

Specifying Formal Languages

- Two Dual Notions
  - Generative approach (grammar or regular expression)
  - Recognition approach (automaton)
- Many theorems to transforms one approach automatically to another

复旦大学媒体计算研究所

正则表达式
REGULAR EXPRESSIONS

复旦大学媒体计算研究所

Language

- Language denotes any set of strings over alphabets (very broad definition)

Examples of Languages

- **Alphabet** = English characters
- **Language** = English sentences
  - Not every string of English characters is an English sentence
- **Alphabet** = ASCII
- **Language** = C programs
  - Note: ASCII character set is different from English character set

复旦大学媒体计算研究所

Specifying Tokens

- String is a finite sequence of symbols
  - E.g. tech is a string length of four
  - The empty string, denoted Є, is a special string length of zero
  - Prefix of s
  - Suffix of s
  - Substring of s
  - proper prefix, suffix, substring of s (x ≠ s, x ≠ Є)（真前缀，后缀，子串）
  - Subsequence of s

复旦大学媒体计算研究所

联结（Concatenation）

- If x and y are strings then the *concatenation* of x and y, written xy, is the string formed by appending y to x.
  - If x = dog and y = house then xy = doghouse.
- String Exponentiation: If x is a string then x^2 = xx, x^3 = xxx, etc. x^0 = ε.
  - If x = ba and y = na then xy2 = banana.

复旦大学媒体计算研究所

Specifying Tokens

- Abstract languages like the empty set {Є}, the set only empty strings
- Languages Operations
  - Union（并）
  - Concatenation（联结）
  - Closure（闭包）

复旦大学媒体计算研究所

Language Operations

- Union
  - L U M { s | s is in L or in M}
  - If L and M are languages then L ∪ M is the language containing all strings in L and all strings in M.
- Concatenation
  - {st | s is in L and t is in M}
  - If L and M are languages then LM is the language that contains concatenations of any string in L with any string in M.
- Kleene Closure
  - If L is a language then L^* = {Є} U L U LL U LLL U LLLL U ....
- Positive Closure（正闭包）
  - If L is a language then L^+ = L U LL U LLL U LLLL U ....

复旦大学媒体计算研究所

Language Operations

- For e.g. Let L = {A,B,….,Z,a,b,…,z} and let D = {0,1,2,….,9}
  - L∪D ?
  - LD?
- L^4 = LLLL is the set of all four-letter strings,
- L^* is the set of all strings of letters including the empty string, Є,
- $L(L \cup D)^*$ is the set of all strings of letters and digits that begin with a letter, and
- D^+ is the set of all strings of one or more digits.

复旦大学媒体计算研究所

复旦大学媒体计算研究所

Regular Expressions

- Regular expressions（正则表达式） are an important notation for specifying patterns.
  - Letter (letter| digit)* → ?
- Alphabet（字母表）: A finite set of symbols.
  - {0,1} is the binary alphabet.
  - ASCII and EBCDIC are two examples of computer alphabets.
- A *string*（字符串） over an alphabet is a finite sequence of symbols drawn from that alphabet.
  - 011011 is a string of length 6 over the binary alphabet.
  - The empty string denoted Є, is a special string of length zero.
- A *language*（语言） is any set of strings over some fixed alphabet.

复旦大学媒体计算研究所

Rules for Regular Expressions Over Alphabet Σ

- ε is a regular expression denoting {ε}, the set containing the empty string.
- If a is a symbol in Σ then a is a regular expression denoting {a}.
- If r and s are regular expressions denoting languages L(r) and L(s), respectively then:

  r|s is a regular expression denoting L(r) U L(s),

  rs is a regular expression denoting L(r)L(s),

  r* is a regular expression denoting (L(r))*.

复旦大学媒体计算研究所

Precedence

- The unary operator * has highest precedence
- Concatenation has second highest precedence
- | has the lowest precedence.
- All operators are left associative.

复旦大学媒体计算研究所

E.g. Pascal Identifiers

letter → A|B|….|Z|a|b|….|z

digit → 0|1|…|9

id → letter (letter | digit)*

复旦大学媒体计算研究所

E.g. Unsigned Numbers in Pascal

digits → digit digit*

opt_frac → .digits | Є

opt_exp → (E(+ | - | Є) digits) | Є

num → digits opt_frac opt_exp

复旦大学媒体计算研究所

Shorthand Notations

- If r is a regular expression then :
  - r+ means r r* and
  - r? means r | Є
  - . means any chars
  - a-z means chars between a and z

Recognition of Tokens

复旦大学媒体计算研究所

- Consider the language fragment:
  if → if
  then → then
  else → else
  relop → < | <= |= |<> | > | >=
  id → letter (letter | digit)*
  num → digit+(.digit+)?(E(+ | -)?digit+)?
- Assume lexemes are separated by white space. The regular expression for white space is ws.
  delim → blank | tab | newline
  ws → delim+
- The lexical analyzer does not return a token for ws. Rather, it finds a token following the whitespace and returns that to the parser.

有限自动机
FINITE AUTOMATA

复旦大学媒体计算研究所

Finite Automata

复旦大学媒体计算研究所

- Regular expressions = specification
- Finite automata = implementation

Finite Automata

复旦大学媒体计算研究所

- A mathematical model
  – state transition diagram
- Recognizer for a given language
- 5-tuple {Q, ∑ , δ, q_0, F}
  – Q is a finite set of states
  – ∑ is a finite set of input
  – *f* transition function Q x ∑
  – q_0 , δ, initial and final state respectively

Finite Automata

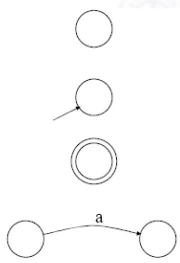
复旦大学媒体计算研究所

- Transitions
  $s1 \xrightarrow{a} s2$
- Is read
  In state s1 on input “a” go to state s2
- If end of input and in accepting state => accept
- Otherwise => reject

Finite Automata State Graphs

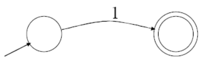
复旦大学媒体计算研究所

- A state
- The start state
- An accepting state
- A transition



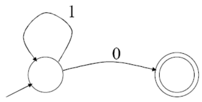
A Simple Example

- A finite automaton that accepts



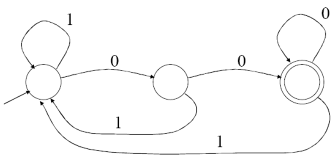
Another Simple Example

- A finite automaton accepting any number of 1's followed by a single 0
- Alphabet: {0,1}



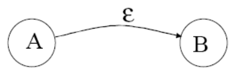
And Another Example

- Alphabet {0,1}
- What language does this recognize?



Epsilon Moves

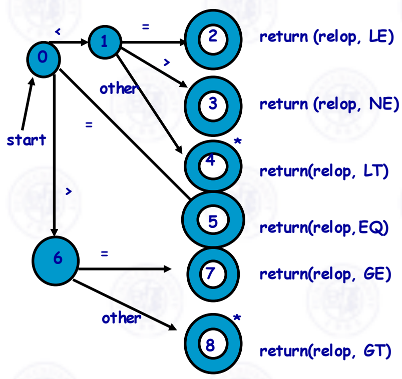
- Another kind of transition: ε-moves



- Machine can move from state A to state B without reading input

Finite Automata(FA)

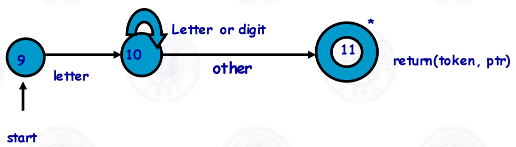
- NFA(Nondeterministic FA) vs. DFA(Deterministic FA)
  - Represented by a directed graph
  - NFA: But different rule applications may yield different final results
  - The same f(s, i) results in a different state
- DFA is a special case of NFA
  - No state has an ε transition
  - For each state s and input a, there is at most one edge labeled a leaving s.
  - Give examples (see the board)
- Conversion NFA -> DFA



Transition Diagrams

复旦大学媒体计算研究所

- Double circles mark *accepting states*; where a token has been found.
- Asterisks marks states where a character must be pushed back.
- E.g. Identifiers and keywords



复旦大学媒体计算研究所

Keywords

- 可以采用以下两种方式
- (1) write a separate transition diagram for each keyword
- (2) load the keywords in the symbol table before reading source (a field in the symbol table entry contains the token for the keyword, for non-keywords the field contains the id token).

复旦大学媒体计算研究所

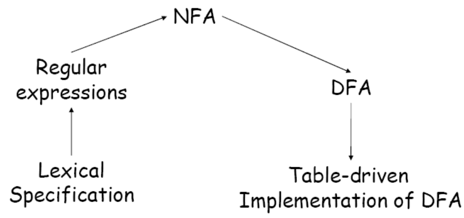
Outline

- Specifying lexical structure using regular expressions
- Finite automata
  - Deterministic Finite Automata (DFAs)
  - Non-deterministic Finite Automata (NFAs)
- Implementation of regular expressions
  - RegExp => NFA => DFA => Tables

复旦大学媒体计算研究所

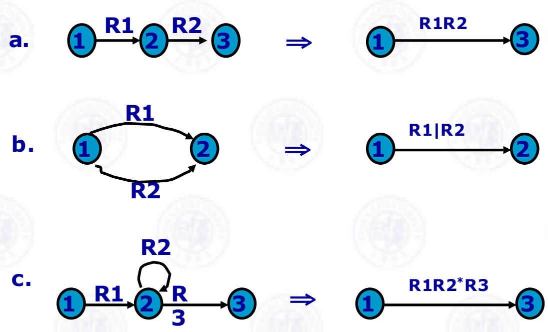
Regular Expressions to Finite Automata

- High-level sketch



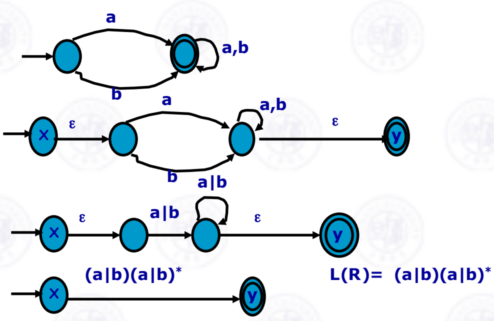
复旦大学媒体计算研究所

FA->RegExp



复旦大学媒体计算研究所

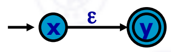
FA->RegExp



Regular Expressions to NFA (1)

复旦大学媒体计算研究所

- For ε,



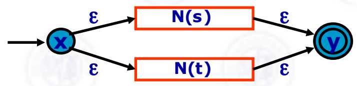
- For a,

Thompson's Construction

Regular Expressions to NFA (2)

复旦大学媒体计算研究所

- For s | t,

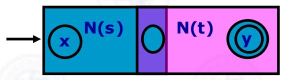


Thompson's Construction

Regular Expressions to NFA (3)

复旦大学媒体计算研究所

- For st,

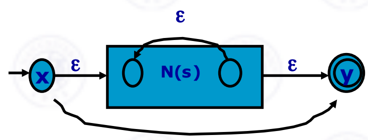


Thompson's Construction

Regular Expressions to NFA (4)

复旦大学媒体计算研究所

- For s*,



Thompson's Construction

Thompson's Construction

复旦大学媒体计算研究所

- N(r) has at most twice as many states as the number of symbols and operators in r.
- N(r) has exactly one start state and one accepting state.
- Each state of N(r) has either one outgoing transition on a symbol in Σor at most two outgoing ε-transitions.

Example of RegExp -> NFA conversion

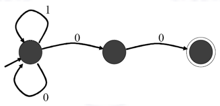
复旦大学媒体计算研究所

- (a|b)*abb

复旦大学媒体计算研究所

Acceptance of NFAs

- Rule: NFA accepts if it can get to a final state
- Weakness
  - An NFA can get into multiple states
- **Example**



Input: 1 0 0

复旦大学媒体计算研究所

How to judge?

- Backtracking
- Save all possible reaching states

复旦大学媒体计算研究所

ε-closure

- ε-closure(s)
  - Set of NFA states reachable from NFA state s on ε-transitions along.
- ε-closure(T)
  - Set of NFA states reachable from NFA state s in T on ε-transitions along.
- move(T,a)
  - Set of NFA states to which there is a transition on input symbol a from some NFA state s in T

复旦大学媒体计算研究所

Simulating the NFA

```
S=ε-closure({s0});
a = nextchar;
while (a!=eof)
{
        S= ε-closure(move(S,a));
        a=nextchar;
}
if (S ∩ F != Φ)
        return 1;
else
        return 0;
```

复旦大学媒体计算研究所

NFA to DFA

- NFAs and DFAs recognize the same set of languages (regular languages)
- DFAs are faster to execute
  - There are no choices to consider
- For a given language NFA can be simpler than DFA

复旦大学媒体计算研究所

NFA to DFA: The Trick

- Simulate the NFA
- Each state of DFA
  = a non-empty subset of states of the NFA
- Start state
  = the set of NFA states reachable through ε-moves from NFA start state
- Add a transition S $\xrightarrow{a}$ S' to DFA iff
  - S' is the set of NFA states reachable from any state in S after seeing the input a, considering ε-moves as well

复旦大学媒体计算研究所

Subset Construction

- NFA: N, DFA: D,
- Construct *Dstates* for D
- Construct a transition table *Dtran* for D.
- Algorithm:
  - Initially, ε-closure(s_0) is only state in Dstates and it is unmarked

复旦大学媒体计算研究所

算法

```
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;
        }
    }
```

复旦大学媒体计算研究所

NFA to DFA. Remark

- An NFA may be in many states at any time
- How many different states ?
- If there are N states, the NFA must be in some subset of those N states
- How many subsets are there?
  - $2^N - 1$ = finitely many

复旦大学媒体计算研究所

Implementation

- A DFA can be implemented by a 2D table T
  - One dimension is "states"
  - Other dimension is "input symbol"
- For every transition $S_i \xrightarrow{a} S_k$ define T[i,a] = k
- DFA "execution"
  - If in state Si and input a, read T[i,a] = k and skip to state S_k
  - Very efficient

复旦大学媒体计算研究所

Implementation (Cont.)

- NFA -> DFA conversion is at the heart of tools, such as flex
- But, DFAs can be huge
- In practice, flex-like tools trade off speed for space in the choice of NFA and DFA representations

Automaton	Space	Time
NFA	O(|r|)	O(|x|*|r|)
DFA	O($2^{\|r\|}$)	O(|x|)

|r| 正则表达式长度，|x|表示输入字符串的长度

复旦大学媒体计算研究所

DFA minimization

- DFA minimization is the task of transforming a given deterministic finite automaton (DFA) into an equivalent DFA that has minimum number of states.
- Here, two DFAs are called equivalent if they describe the same regular language.
- Several different algorithms accomplishing this task are known and described in standard textbooks on automata theory.

复旦大学媒体计算研究所

DFA minimization

- For each regular language that can be accepted by a DFA,
  - there exists a DFA with a minimum number of states (and thus a minimum programming effort to create and use)
  - and this DFA is unique (except that states can be given different names.)

Minimum DFA

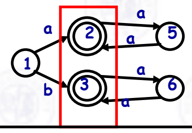
- There are three classes of states that can be removed/merged from the original DFA without affecting the language it accepts.
- Unreachable states are those states that are not reachable from the initial state of the DFA, for any input string.
- Dead states are those non-accepting states whose transitions for every input character terminate on themselves. These are also called Trap states because once entered there is no escape.
- Non-distinguishable states are those that cannot be distinguished from one another for any input string.

Minimize DFA state number

- Merging
- Split

Merging

- Two states s1 and s2 are equivalent when the machine starting in s1 accepts a string δ if and only if starting in s2 it accepts δ.
- How to find equivalent states?
  - trans[s1,c]=trans[s2,c]
  - not sufficiently general

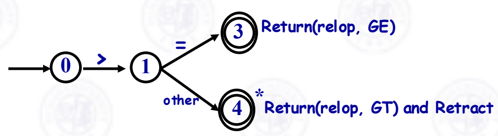


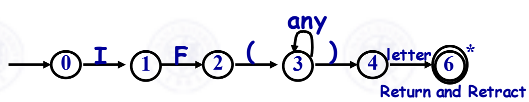
Split

- There are two major groups: non-final and final states.
- A group is split on a terminal if the states in the group have transitions on this terminal to different groups.
- If all the transitions are to the same group, then there is no split on this group based on this terminal.

```
do{
    for each group G,
     partition G if G contains distinguishable states.
   } while( the number of groups doesn't increase.)
```

Use FA to build scanner

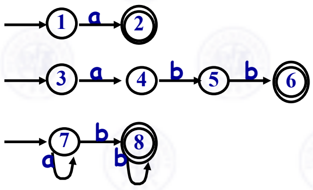




复旦大学媒体计算研究所

Example :

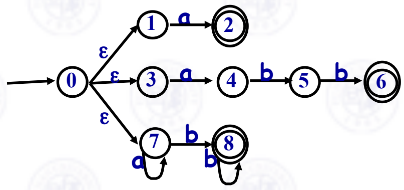
a | abb | a*b$^+$



复旦大学媒体计算研究所

- Max length match
- Muti－match select

Input: aaba; abb



复旦大学媒体计算研究所

scanner comparison

- Hand-coded scanner (like any ordinary program):
  - Programmer creates types, defines data & procedures, designs flow of control, implements in source language.
- Lex-generated scanner:
  - Programmer writes patterns
  - (Declarative, not procedural)
  - Lex implements flow of control
  - Must less hand-coding, but
  - code looks pretty alien, tricky to debugs

复旦大学媒体计算研究所

Lexical Analyzer Generator

- Lex, A Lexical Analyzer Generator
  - http://dinosaur.compilertools.net/
- Flex, A fast scanner generator
  - http://flex.sourceforge.net/
- JFLEX, The Fast Scanner Generator for Java
  - http://jflex.de/

复旦大学媒体计算研究所

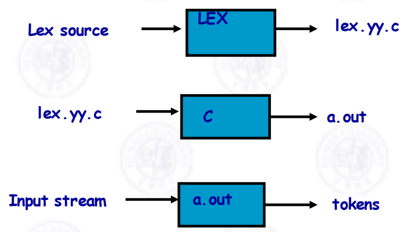
Testing Lexical Analyzer

- Create a suite of test source files to run through your analyzer rather than entering the source through the keyboard.
- Much faster
- More thorough
- Repeatable : You can make sure that correcting one bug in your analyzer doesn't introduce other bugs.
- Better documentation.

复旦大学媒体计算研究所

LEX

- A tool to generate a lexical analyzer from regular expressions.



复旦大学媒体计算研究所

what does lex do?

- Input: patterns written by programmer, describing tokens in language
- Process:
  - Reads patterns as regular expressions
  - Builds finite automaton to accept valid tokens
- Output: C code implementation of FA
- Compile and link C code, you've got a scanner

复旦大学媒体计算研究所

Structure of a Lex file

- The structure of a Lex file is divided up into three sections.

```
Definition section
%%
Rules section
%%
C code section
```

复旦大学媒体计算研究所

Regular Definitions

```
%%
delim       [\t\n]
ws          {delim}+
letter      [A-Za-z]
digit       [0-9]
id          {letter}({letter} | {digit})*
number      {digit} + (\.{digit} +) ? (E [+ \-] / {digit} + ) ?
%%
{ws}        {/*no action and no return*/}
if          {return (IF);}
then        {return (THEN);}
else        {return (ELSE);}
{id}        {yylval = install_id(); return(ID);}
{number}    {yylval = install_num(); return(NUM);}
```

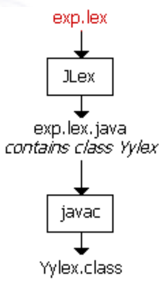
复旦大学媒体计算研究所

```
"<"         {yylval = LT; return(RELOP);}
"<="        {yylval = LE; return(RELOP);}
"="         {yylval = EQ; return(RELOP);}
"<>"        {yylval = NE; return(RELOP);}
">"         {yylval = GT; return(RELOP);}
">="        {yylval = GE; return(RELOP);}
%%
install_id() {/*procedure to install a lexeme into the symbol table and
return a pointer thereto*/}
install_num() {/*procedure to install a lexeme into the number table
and return a pointer thereto*/}
```

复旦大学媒体计算研究所

JLEX

- A tool to generate a lexical analyzer from regular expressions.
- based upon the Lex lexical analyzer generator model. JLex takes a specification file similar to that accepted by Lex, then creates a Java source file for the corresponding lexical analyzer



复旦大学媒体计算研究所

Sample of FLEX

```
/*** Definition section ***/
 %{
/* C code to be copied verbatim */
#include <stdio.h>
%}
 /* This tells flex to read only one input file */
%option noyywrap
%%

   /*** Rules section ***/
   /* [0-9]+ matches a string of one or more digits */
[0-9]+ {
        /* yytext is a string containing the matched text.
*/
        printf("Saw an integer: %s\n", yytext);
     }
.|\n   {  /* Ignore all other characters. */  }
```

```
%%
/*** C Code section ***/

int main(void)
{
   /* Call the lexer, then quit.
*/
   yylex();
   return 0;
}
```

复旦大学媒体计算研究所

Summary:

- Automata theory⇒ Programming practice
  - Regular expressions and automata theory prove you can write regular expressions, give them to a program like `lex`, which will generate a machine to accept exactly those expressions.

复旦大学媒体计算研究所

课后作业

- 2.3a
- 2.4b
- 2.5a
- 2.6
- 用正则表达式描述Java程序中的注释
- 用正则表达式描述C++程序中的注释