Compiler Design
Lexical Analysis
Summary

conf. dr. ing. Ciprian-Bogdan Chirila

chirila@cs.upt.ro

http://www.cs.upt.ro/~chirila

Tokens

- Lexical analyzer
  - scans the source programs
  - produces as output sequence of tokens
- Tokens
  - name
  - lexical value

Lexemes

- sequence of input characters that the token represents

Buffering

- goals
  - to scan ahead on the input
  - to see where the next lexeme ends
  - to accelerate the process of scanning the input
- techniques
  - using a pair of buffer cyclically
  - ending each buffer content with a sentinel which warns of its end

Patterns

- description of token structure
- sequences of characters which can form the lexemes corresponding to the token
- language
  - set of words, strings of characters that match a given pattern

Regular Expressions

- expressions used to describe patterns
- built from
  - single characters
  - operators
    - union
    - concatenation
    - Kleene closure or any-number of

Regular Definitions

- complex collections of languages
- patterns that describe the tokens of a programming language
- sequence of statements that each define one variable to stand for regular expressions
- regular expression for one variable can use previously defined variables in its regular expression

Extended Regular-Expression Notation

- a number of additional operators
- short-hands in regular expressions
- to make easier to express patterns
- operators
  - + one or more of
  - ? zero or one of
  - character classes
    - the union of strings each consisting of one of the characters

Transition Diagrams

- expresses the behavior of the lexical analyzer
- states
  - representing something about the history of the characters seen during the current search
- arrows or transitions
  - from one state to another
  - indicates the possible next input characters that cause the lexical analyzer to make that change of state

Finite Automata

- formalization of transition diagrams
- include a start state
- one or more accepting states
- set of states
- input characters
- transitions among states
- accepting states
  - indicate that the lexeme for some token has been found
- can make transitions on
  - empty input
  - characters input

Deterministic Finite Automata

- special kind of finite automata
- has exactly one transition
  - out of each state
  - for each input symbol
- transitions on empty input are not allowed
- is easily simulated
- makes good implementation of a lexical analyzer
- similar to a transition diagram

Nondeterministic Finite Automata

- automata which are not DFA
- easier to design than DFA's
- possible architecture for lexical analyzer
  - to tabulate all the states
  - that NFA's for each of the possible patterns can be in
  - as we scan the input characters

Conversion Among Patterns Representation

- to convert any RE into NFA about the same size
- recognizing the same language
- any NFA can be converted to DFA for the same pattern
  - in the worst case the size of the automaton can grow exponentially
  - never encountered in common programming languages
- to convert NFA and DFA into RE

Lex

- patterns for tokens
  - regular expression notation
- lex and flex
  - family of software systems
  - lexical-analyzer generators
  - converts regular expressions into lexical analyzer ~ DFA

Minimization of Finite Automata

- for every DFA there is a minimum-state DFA accepting the same language
- the minimum-language DFA is unique except state naming

Bibliography

- Alfred V. Aho, Monica S. Lam, Ravi Sethi, Jeffrey D. Ullman – Compilers, Principles, Techniques and Tools, Second Edition, 2007