Compiler Design
Lexical Analysis
Finite Automata

conf. dr. ing. Ciprian-Bogdan Chirila

chirila@cs.upt.ro

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

Outline

- Nondeterministic Finite Automata
- Transition Tables
- Acceptance of Input Strings by Automata
- Deterministic Finite Automata

Finite Automata

- lexical rules -> finite automata ->lexical analyzer
- Recognizers of each possible input string
  - Answer yes or no
- Two flavors:
  - Nondeterministic Finite Automata (NFA)
    - No restrictions on the labels of their edges
    - A symbol can label several edges out of the same state
    - The empty string ε is a valid label
  - Deterministic Finite Automata (DFA)
    - for each state and each symbol only one edge is leaving that state
- NFA and DFA recognize the same languages
- Regular Languages
  - regular expressions can describe

Nondeterministic Finite Automata

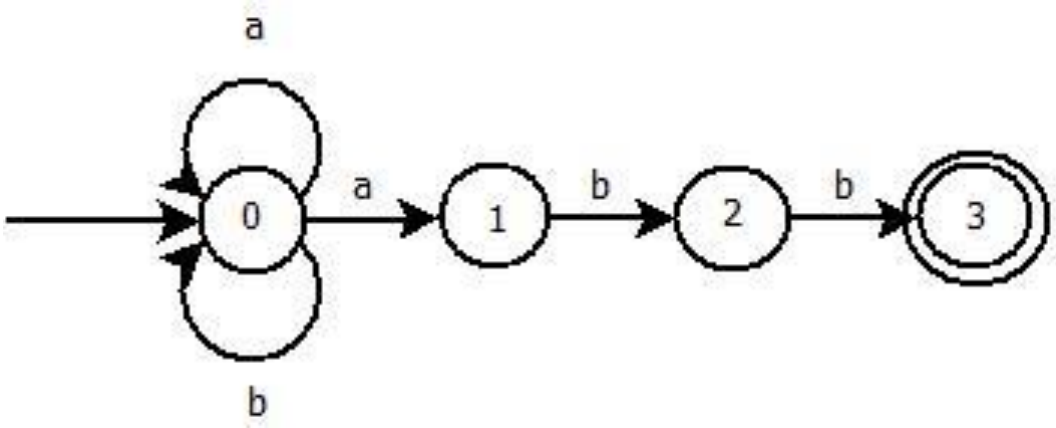
- A finite set of states S
- A set of input symbols Σ
  - the input alphabet
  - ε is not in Σ
- A transition function
  - for each state and each symbol gives a set of next states
- A state s_0 from S
  - start (initial) state
- A set of states F
  - subset of S
  - accepting (final) states

Finite Automata Representation

- Transition graph
  - nodes are states
  - labeled edges -> transition function
  - $s \xrightarrow{a} t$
  - graph ~ transition diagram
    - the same symbol can label edges from one state to several different states
    - an edge can be labeled by ε in addition to symbols from the input alphabet

Example

- (a|b)*abb



- a nondeterministic finite automaton
  - start state 0
  - accepting state 3
  - the only strings getting in the accepting state are ending in “abb”

Transition Tables

- rows correspond to states
- columns correspond to input symbols and ε
- if the transition function has no information about that state-input pair the value in the table is ø

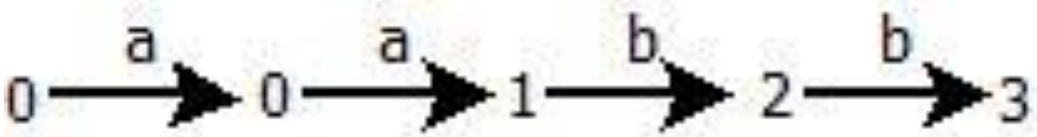
state	a	b	ε
0	{0,1}	{0}	ø
1	ø	{2}	ø
2	ø	{3}	ø
3	ø	ø	ø

Acceptance of Input Strings by Automata

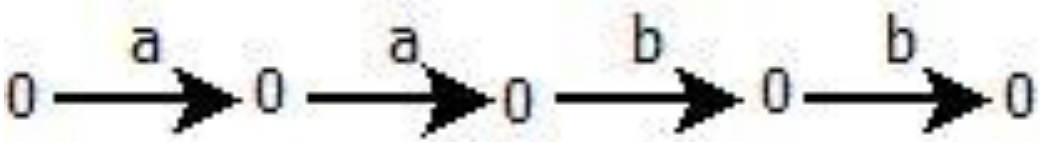
- a string x is accepted by a NFA iff there is one path in the transition graph
  - from the start state
  - to one accepting states
- the ε labels across the path are ignored
- the language defined / accepted by a NFA
  - set of strings labeling some path from start to accepting state
- notation L(A) – language accepted by automaton A

Example 1

- label *aabb* is covered by path from state 0 to 3



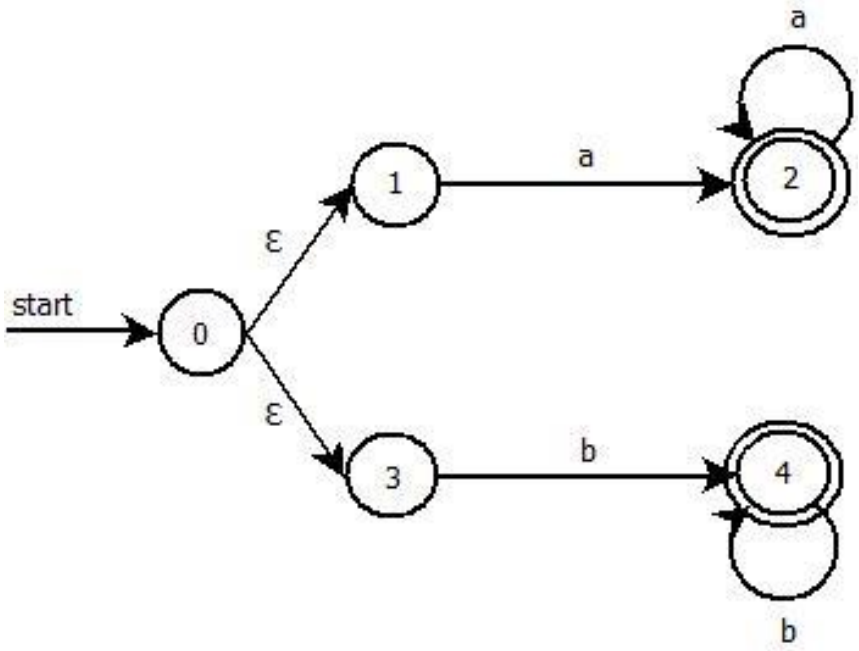
- the same *aabb* label may lead to different states



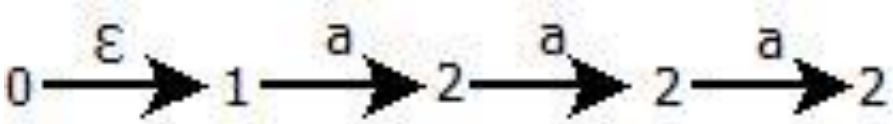
  - this path leads to state 0 which is not accepting
  - A NFA accepts a string as long exists a path from the start state to an accepting state
  - a path leading to non-accepting state is irrelevant

Example 2

- L(aa*|bb*)



- string *aaa* is accepted



Deterministic Finite Automata

- DFA – deterministic finite automaton
  - is a NFA where
    - there are no moves on input
    - for each state **s** and input symbol **a** there is only one edge out of **s** labeled **a**
  - no more sets in the transition table
- NFA – abstract representation of an algorithm
- DFA – concrete algorithm for string recognition

Simulating a DFA

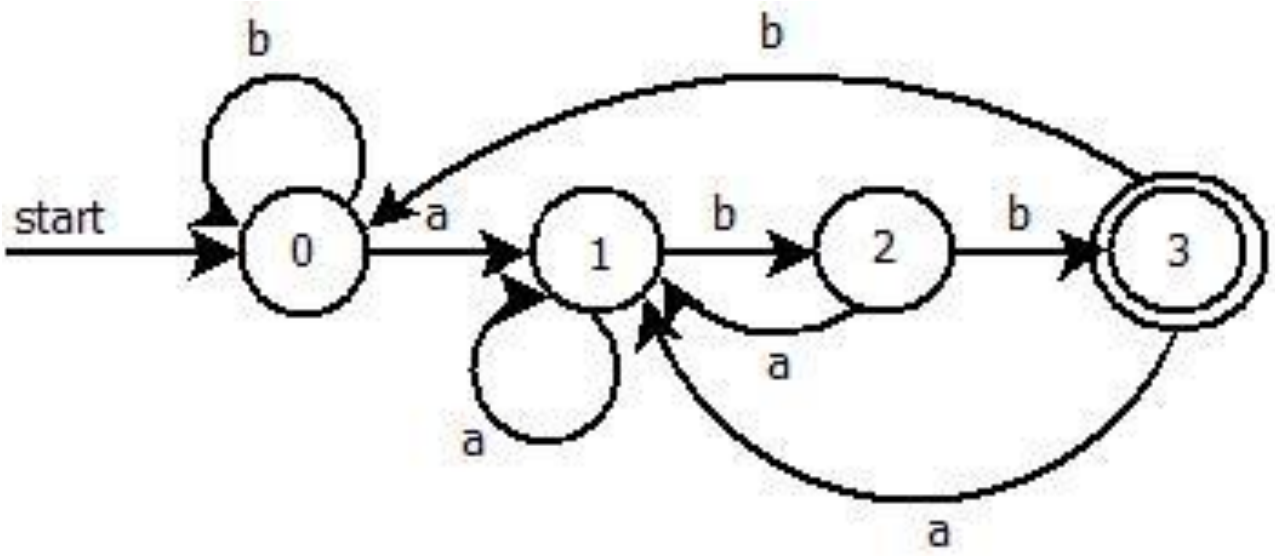
- Input
  - An input string x terminated by eof character
  - DFA *D*
    - start state s_0
    - accepting states *F*
    - transition function *move*
- Output
  - yes - if D accepts
  - no - otherwise
- Method
  - function *move(s,c)* – gives the state to which is an edge from state *s* on input *c*
  - function *nextChar* – returns the next character of the input string *x*

Algorithm: Simulating an DFA

```
s=s0;
c=nextChar();
while(c!=eof)
{
  s=move(s,c);
  c=nextChar();
}
if(s is in F) return “yes”;
else return “no”;
```

Example

- DFA accepting (a|b)*abb



Bibliography

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