Motivating Example

- Token set:
  - assign -> :=
  - plus -> +
  - minus -> -
  - times -> *
  - div -> /
  - lparen -> (
  - rparen ->)
  - id -> letter(letter|digit)*
  - number -> digit digit*|digit*(.digit|digit.)digit*

1

Motivating Example

- What are the lexemes in the string "a_var := b * 3" ?
- What are the corresponding tokens ?
- How do you identify the tokens?

2

A naïve scanner algorithm

1. read characters one at a time (e.g., cur_char) with one look-ahead character lh_char
2. if cur_char is one of the one-character tokens { ( ) [ ] < > , ; = + −} then return the token
3. if cur_char is ':', check lh_char

   if lh_char is '=', then return token ":="

   else report error
3. if cur_char is a digit,

   then read any additional digits with at most one '.' and return number

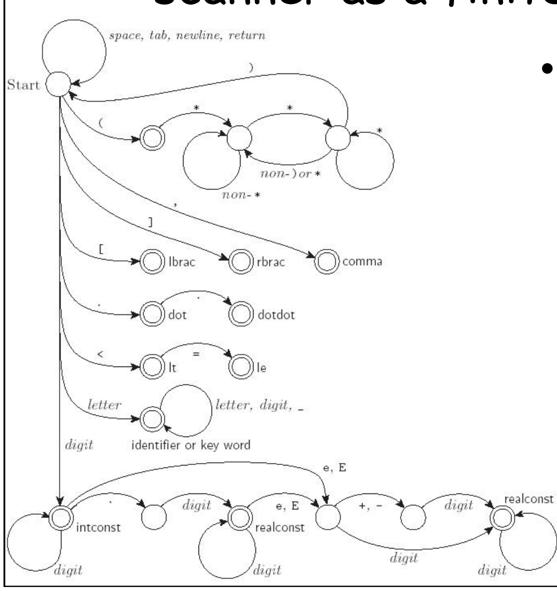
   else report error
4. if cur_char is a letter,

   then read any additional letters and digits and return id

… …

3

Pictorial representation of the scanner as a finite automaton



- This is a deterministic finite automaton (DFA)

4

Can we automate the scanner implementation?

- Yes. A scanner generator tool can generate the implementation from regular expressions with four steps
  - Convert regular expressions to Nondeterministic Finite Automata (NFA)
  - Convert NFA to Deterministic Finite Automata (DFA)
  - Minimize the DFA
  - Implement DFA as switch statements

5

Finite Automaton (FA)

- A simple idealized machine to recognize patterns in character sequences
- Its job is to accept or reject an input depending on whether the pattern defined by the FA occurs in the input

6

Revisit the Example

- Can an FA **M** accept a string **w**?
  - The machine starts in the start state
  - Given each character of **w**, the machine will transition from state to state according to predefined transition rules
  - **M** accepts **w** if the last input of **w** causes the machine to halt properly. Otherwise, it is said that **M** rejects **w**.

Finite Automaton & Regular Expression

- They have equivalent expressive power
  - For every regular expression RE, there is a corresponding FA that accepts the set of strings generated by RE
  - For every FA, there is a corresponding RE that generates the set of strings accepted by FA

Two types of FA

- Deterministic Finite Automaton (DFA)
  - Each transition is uniquely determined by its source state and input symbol
  - Reading an input symbol is required for each state transition
- Nondeterministic Finite Automaton (NFA)
  - For some state and input symbol, the next state may be one or more possible states
  - Epsilon transitions: arrows labeled by the empty string symbol ε

9

Deterministic Finite Automaton (DFA)

- 5-tuple, $(Q, \Sigma, \delta, q_0, F)$, consisting of
  - a finite set of states (Q)
  - a finite set of input symbols called the alphabet (Σ)
  - a transition function ($\delta : Q \times \Sigma \rightarrow Q$)
  - an initial or start state ($q_0 \in Q$)
  - a set of accept states ($F \subseteq Q$)

10

Deterministic Finite Automaton (DFA)

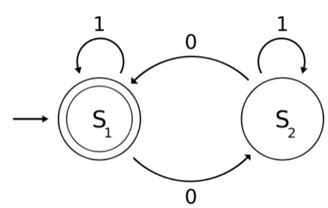
- Let $w = a_1a_2 \dots a_n$ be a string over the alphabet Σ. The automaton M accepts the string w if a sequence of states, $r_0, r_1, \dots, r_n$, exists in Q with the following conditions:
  - $r_0 = q_0$
  - $r_{i+1} = \delta(r_i, a_{i+1})$, for $i = 0, \dots, n-1$
  - $r_n \in F$.

11

DFA Example

- $M = (Q, \Sigma, \delta, q_0, F)$ where
  - $Q = \{S_1, S_2\}$,
  - $\Sigma = \{0, 1\}$,
  - $q_0 = S_1$,
  - $F = \{S_1\}$, and
  - δ is defined by the following state transition table:

	0	1
S_1	S_2	S_1
S_2	S_1	S_2

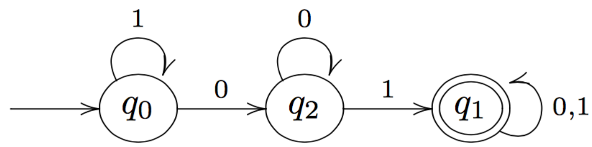


State transition diagram

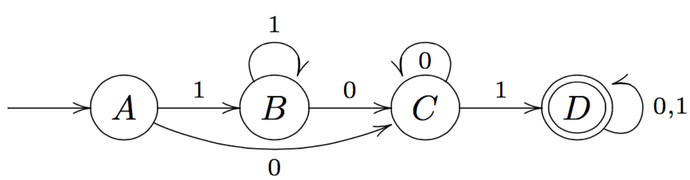
12

What is the transition diagram of the following transition table? [2]

	0	1
$\rightarrow q_0$	q_2	q_0
$*q_1$	q_1	q_1
q_2	q_2	q_1



What is the transition table of the following transition diagram?



	0	1
$\rightarrow$A	C	B
B	C	B
C	C	D
$*$D	D	D

Nondeterministic Finite Automaton (NFA)

- Let P(Q) denote the power set of Q
- 5-tuple, $(Q, \Sigma, \Delta, q_0, F)$, consisting of
  - a finite set of states Q
  - a finite set of input symbols Σ
  - a transition function $\Delta : Q \times \Sigma \rightarrow P(Q)$
  - an initial (or start) state $q_0 \in Q$
  - a set of accept states $(F \subseteq Q)$

15

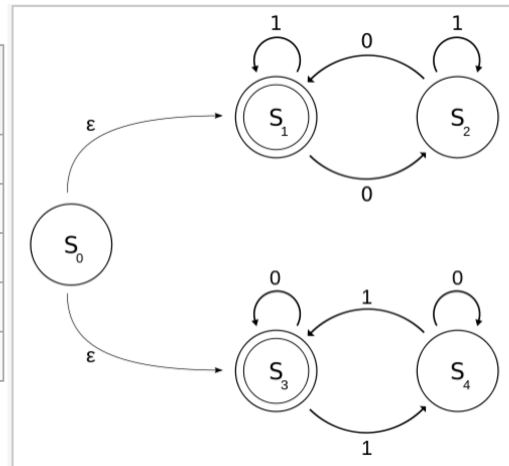
Nondeterministic Finite Automaton (NFA)

- Let $w = a_1a_2 \ldots a_n$ be a word over the alphabet Σ. The automaton *M* accepts the word *w* if a sequence of states, $r_0, r_1, \ldots, r_n$, exists in *Q* with the following conditions:
  - $r_0 = q_0$
  - $r_{i+1} \in \Delta(r_i, a_{i+1})$, for $i = 0, \ldots, n-1$
  - $r_n \in F$.

16

An NFA Example [3]

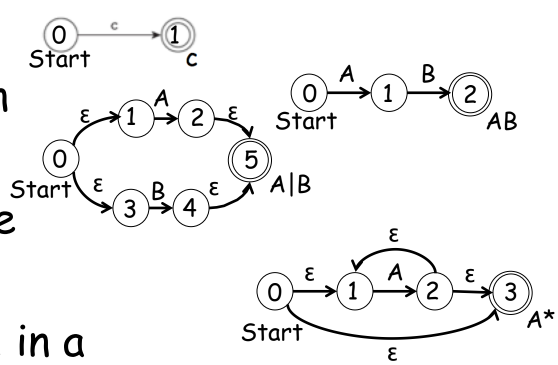
Input / State	0	1	ε
S_0	{}	{}	$\{S_1, S_3\}$
S_1	$\{S_2\}$	$\{S_1\}$	{}
S_2	$\{S_1\}$	$\{S_2\}$	{}
S_3	$\{S_3\}$	$\{S_4\}$	{}
S_4	$\{S_4\}$	$\{S_3\}$	{}



17

From a Regular Expression to an NFA

a) Base case
b) Concatenation
c) Alternation
d) Kleene closure

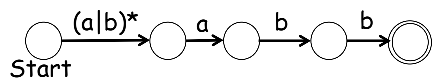


Note: Build NFA in a top-down manner by breaking components level-by-level

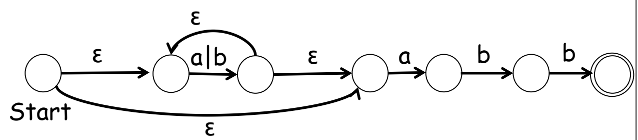
18

An Example

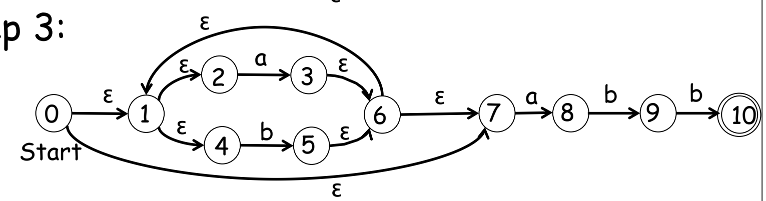
- (a|b)*abb
  - Step 1:



  - Step 2:

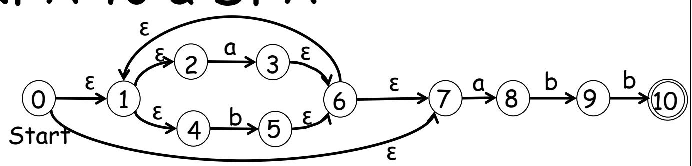


  - Step 3:



19

From an NFA to a DFA



- Insight
  - Remove ambiguous transition by merging states
  - Each DFA state corresponds to a set of equivalent NFA states
  - The NFA states within an equivalent set have ε-transitions among themselves, or they are reachable via the same input symbol from states in another equivalent set

20

From an NFA to a DFA

Define ε-closure(s) as the set of NFA states reachable from state s on ε-transitions alone.

Algorithm:

```
push ε-closure(start) on stack
while stack is not empty do
    pop an element T,
    mark T as processed
    for each input symbol a do
        find all states reachable from any state in T via a,
        also find the ε-closure of those newly found states,
        if the set is already processed
            get the assigned label U
        else
            assign a new label to the state set, such as U
            put U on stack
        Trans[T, a] := U, where T and U are new states in DFA, and their
transition is labeled with a
```