Parul® University

NAAC A++

Regular grammars and equivalence with finite automata

Chapter - 2: Regular languages and finite automata

Prof. Riddhi Atulkumar Mehta
Assistant Professor
Department of Computer Science and Engineering

Parul® University

NAAC A++

Content

INDEX

Introduction

- Regular grammars are the simplest type of formal grammars in the Chomsky hierarchy
- They generate regular languages
- These languages can be recognized by Finite Automata (FA)

- This section explores:
  - ✅ Definition of regular grammars
  - ✅ Examples
  - ✅ Equivalence with finite automata

What is a Grammar?

A grammar G is defined as a 4-tuple:

G = (V, **Σ**, P, S)

Where:

- V: Set of variables (non-terminals)
- Σ: Set of terminals
- P: Set of productions/rules
- S: Start symbol ($S \in V$)

What is a Regular Grammar?

A grammar is regular if all production rules are of one of the following forms:

✅ Right Linear Grammar:

- $A \to aB$
- $A \to a$
- $A \to \varepsilon$

✅ Left Linear Grammar:

- $A \to Ba$
- $A \to a$
- $A \to \varepsilon$

Where $A, B \in V$ and $a \in \Sigma$

Example of a Regular Grammar

Grammar G:

- V = {S, A}
- Σ = {0, 1}
- P:
  - S → 0S
  - S → 1A
  - A → 0A
  - A → 1A
  - A → ε
- Start Symbol = S

Language Generated: Strings that start with any number of 0’s or a single 1, followed by any combination of 0’s and 1’s

Regular Grammar ➡ Finite Automaton

Steps:

1. For each production of the form A → aB, add transition from state A to state B on input a
2. For each A → a, add transition from state A to final state on input a
3. For each A → ε, mark A as a final state

Regular Grammar ➡ Finite Automaton Example

Eliminating Epsilon Productions: A Step-by-Step Approach:

S → a, aA | bB

A → aA | aS

B → cS

S → ε

B → ε

Step 1: Identifying Epsilon Productions

First, we identify the epsilon productions in our grammar. In this case, they are –

S → ε

B → ε

Regular Grammar ➡ Finite Automaton Example

Step 2: Generating Non-Epsilon Productions

We list all the productions that do not involve epsilon. These are the productions that form the basis of our epsilon-free grammar –

S → a, aA | bB

A → aA | aS

B → cS

Regular Grammar ➞ Finite Automaton Example

Step 3: Replacing Epsilon Productions

Now, we systematically replace all occurrences of non-terminals with epsilon productions in the right-hand sides of our productions.

- In 'A → aA | aS', replacing 'S' with 'ε' yields 'A → aA | ε'. Since 'ε' is the empty string, we can simplify this to 'A → aA'.
- In 'B → cS', replacing 'S' with 'ε' yields 'B → c'.
- In 'S → aA | bB', replacing 'B' with 'ε' yields 'S → aA | b'.
- The final productions will be like,

S → a, aA | b

A → aA | aS | a

B → cS | c

Regular Grammar ➡ Finite Automaton Example

Step 4: Handling the Start Symbol

- The start symbol 'S' has an epsilon production. This means that the start symbol can derive the empty string, which is often acceptable in finite automata.
- However, if the start symbol has an epsilon production, we need to add a new start symbol ('S1' in our example) that inherits all the productions of the original start symbol, including the epsilon production. This new start symbol ensures that the empty string can be accepted by the automata.

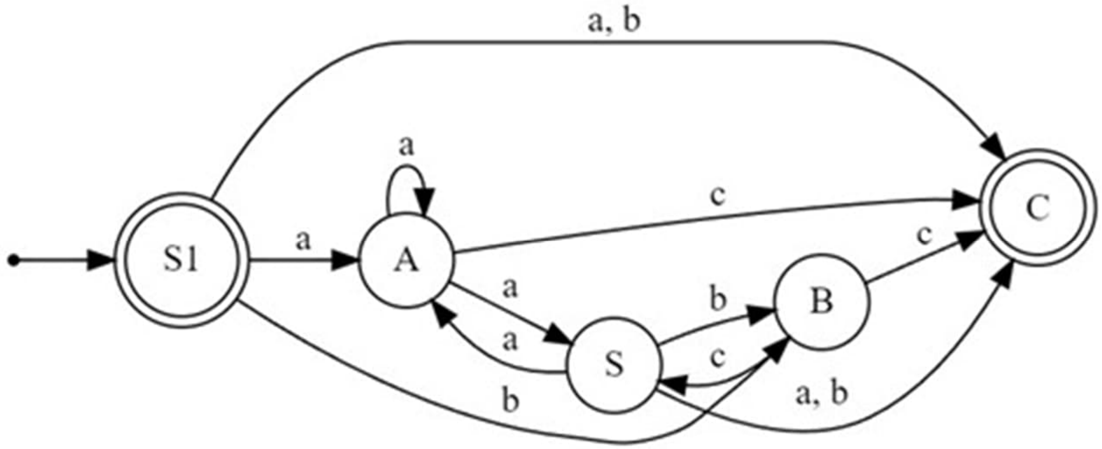
$S1 \rightarrow a \mid aA \mid b \mid bB \mid \varepsilon$

$S \rightarrow a, aA \mid b$

$A \rightarrow aA \mid aS \mid a$

$B \rightarrow cS \mid c$

Regular Grammar ➡ Finite Automaton Example

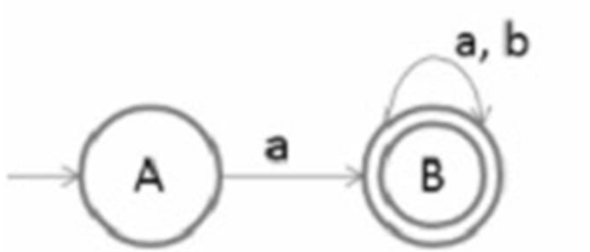


Finite Automaton ➞ Regular Grammar

Steps:

1. For each transition from state A to B on input a, add production A → aB
2. For each final state F, add production F → ε

Let's consider a Finite automaton (FA) as given below –



Finite Automaton ➞ Regular Grammar Example

Pick the start state A and output is on symbol ‘a’ going to state B

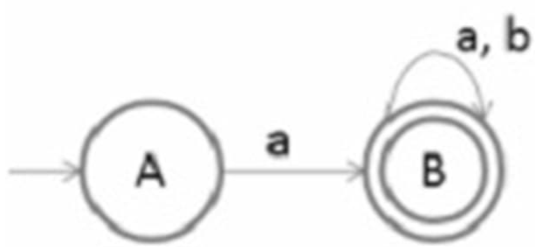
$A \to aB$

Now we will pick state B and then we will go on each output

i.e $B \to aB$

$B \to bB$

$B \to \varepsilon$



Therefore,

Final grammar is as follows –

$A \to aB$

$B \to aB/bB/\varepsilon$

Summary

- Regular grammars are defined by linear productions
- They generate regular languages
- Regular grammars and finite automata are equivalent in power
- Conversion between grammar and automaton is systematic
- Useful in real-world applications like compilers and text scanners

Parul® University | NAAC GRADE A++

https://paruluniversity.ac.in/