Regular Grammars
**Regular Grammars**

*Grammar*

\[ G = \{N, \Sigma, R, S\} \]

- \( N \) = non-terminal symbols (upper case)
- \( \Sigma \) = terminal symbols (lower case)
- \( R \) = rules (see below)
- \( S \) = start symbol (non-terminal)

*Rules for Regular Grammar*

- \( T \rightarrow a \) (non-terminal goes to terminal)
- \( T_1 \rightarrow aT_2 \) (non-terminal goes to terminal followed by non-terminal)
Examples of Regular Grammars

Even Length Bit Strings
• \( S \rightarrow \varepsilon \)
• \( S \rightarrow 0T \)
• \( S \rightarrow 1T \)
• \( T \rightarrow 0S \)
• \( T \rightarrow 1S \)

Bit Strings Ending in 000
• \( S \rightarrow 0S \)
• \( S \rightarrow 1S \)
• \( S \rightarrow 0A \)
• \( A \rightarrow 0B \)
• \( B \rightarrow 0 \)
Examples of Finite State Machines for Regular Grammars

Even Length Bit Strings

Bit Strings Ending in 000
Correspondence Between Finite State Machines and Regular Grammars

Theorem: The regular grammars define exactly the regular languages.

Algorithm: Regular Grammar → Finite State Automata

1. Create a state for each non-terminal symbol
   • Let S correspond to the Start state
2. Create one accepting state #
3. For each Rule $A \rightarrow wB$, add a transition from state $A$ to state $B$
4. For each Rule $A \rightarrow w$, add a transition from $A$ to #
5. For each Rule $A \rightarrow \epsilon$, mark $A$ as an accepting state

Observations

1. Paths in FSA correspond to sequence of Rules in regular grammar.
2. FSA in state $T$ whenever $T$ is the current non terminal symbol.
Four Models for Regular Languages

1. Deterministic Finite State Automata

2. Non-Deterministic Finite State Automata

3. Regular Expressions

4. Regular Grammars