Chapter 3 Regular Languages and Regular Grammar

Recursive Definitions
Regular Expressions
RE and RL
Regular Grammars

4 Representations of RL

- FA: DFA, NFA
- TG: Transition Graph
- **RE: Regular Expression**
- **RG: Regular Grammar**

Recursive Definition: 3 Steps

- Rule 1. Specify basic objects in the set.
- Rule 2. Give rules for constructing more objects in the set.
- Declare that no other object in the set
- Example: EVEN in recursive definition
  - Rule 1. 2 is in EVEN
  - Rule 2. If x is in EVEN then so is x + 2.
  - Rule 3. Those are the only elements in EVEN.

Recursive definition

- Give recursive definition for the following languages
  - ODD PALINDROM
  - EVEN PALINDROM
  - PALINDROM
  - Set ODD = {1, 3, 5, …}

3.1 Regular Expression

- A RE is a concise way of expressing a pattern in a series of characters.
- Regular Language (RL) – a language that can be defined by regular expression.
- RE is one of 4 representations for RL:
  - **RE – Regular Expression**
  - FA – Finite Automata
  - TG – Transition Graph
  - RG – Regular Gramma

Formal Definition of RE

- Definition 3.1 (page 72)
  - Let Σ be a give alphabet, then
    - 1. $\varnothing$, λ, $a \in \Sigma$ are regular expressions (RE). These are called primitive RE.
    - 2. If r_1 and r_2 are RE, so are $r_1 + r_2$, $r_1 . r_2$, r_1^* and (r_1).
    - 3. A string is a RE if and only if it can be derived from rule 1 and rule 2.

Languages Associated with RE

- Definition 3.2 (page 72)
  - The language L (r) denoted by any RE r is defined by the following rules.

1. $\varnothing$ is a regular expression denoting the empty set
2. λ is a RE denoting $\{\lambda\}$
3. For every $a \in \Sigma$, a is a RE denoting $\{a\}$
4. $L(r_1 + r_2) = L(r_1) \cup L(r_2)$
5. $L(r_1 \,.\, r_2) = L(r_1)\, L(r_2)$
6. $L((r_1)) = L(r_1)$
7. $L(r_1^{*}\ ) = (L(r_1))^*$

RE Examples - 1

- Example 3.2 (page 73)
  - $L(a^*.(a+b)) = L(a^*)L(a+b)$
  - $= \{\lambda, a, aa, aaa, \ldots\}\{a, b\}$
  - $= \{a, aa, aaa, \ldots, b, ab, aab, \ldots\}$
- In-class exercise: try exhibit the following RL in set notation and NFA
  - $L((a+b).a^*)$
  - $L(aa^*)$
  - $L((a+b)^*)$

RE Examples - 2

- Example 3.3 $r = (a+b)^*(a+bb)$, L =?
- Example 3.4 $r = (aa)^* (bb)^*b$, L =?
- For $\Sigma = \{0, 1\}$, give a RE such that
  - $L(r) = \{w \in \Sigma^*$: w has at least one pair of consecutive zeros$\}$
  - $L(r) = \{w \in \Sigma^*$: w has no pair of consecutive zeros$\}$

RE Examples - 3

- Example 3.5
  - $r = (0+1)^* 00(0+1)^*$
- Example 3.6
  - $r = (1 + 01)^*(0+\lambda)$

RE Examples - 4

- Exhibit L = { ? } and nfa
  - X*
  - XX*
  - a*b*
  - (ab)*
  - X(XX)*
  - b*ab*a(a+b)*
  - (a+b)(a+b)(a+b)

RE Examples - 5

- Informal notations: a^+, a^3
- RE with * always denoting a language that contains λ
- RE to English descriptions:
  - $(a+b)^*$
  - $(a+b)^*a(a+b)^*$
  - $(a+b)^*aaa$
  - $a(a+b)^*b$

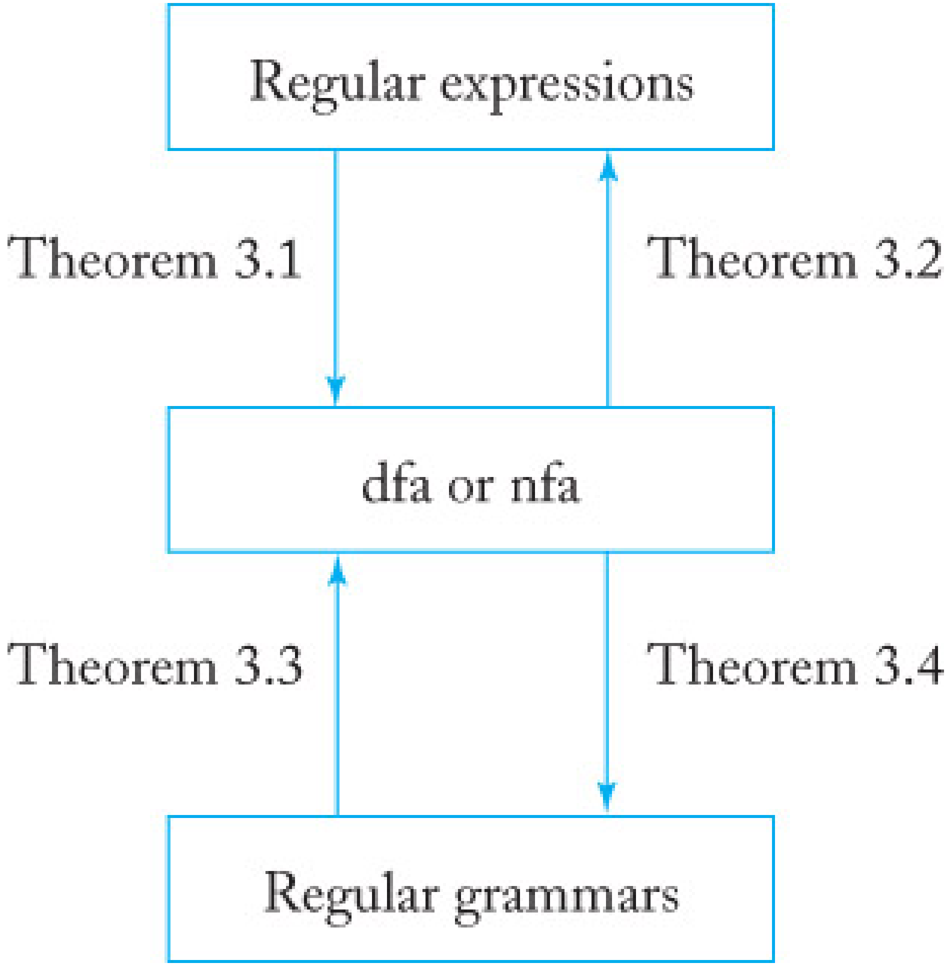
RE Examples - 6

- An application of RE:
  - Letter(Letter +Digit)*
    - A set of strings that beginning with a letter
  - Letter = A+B+…+Z+a+b+…+z
  - Digit = 0+1+2+…+9

3.2 Connection Between Regular Expressions and Regular Languages

- RE denote Regular languages
- Kleen's Theorem:
  - L(FA) = L(TG) = L(RE)
- Theorem 3.1
  - L(RE) $\subset$ L(TG)
- Theorem 3.2
  - L(FA) $\subset$ L(RE)

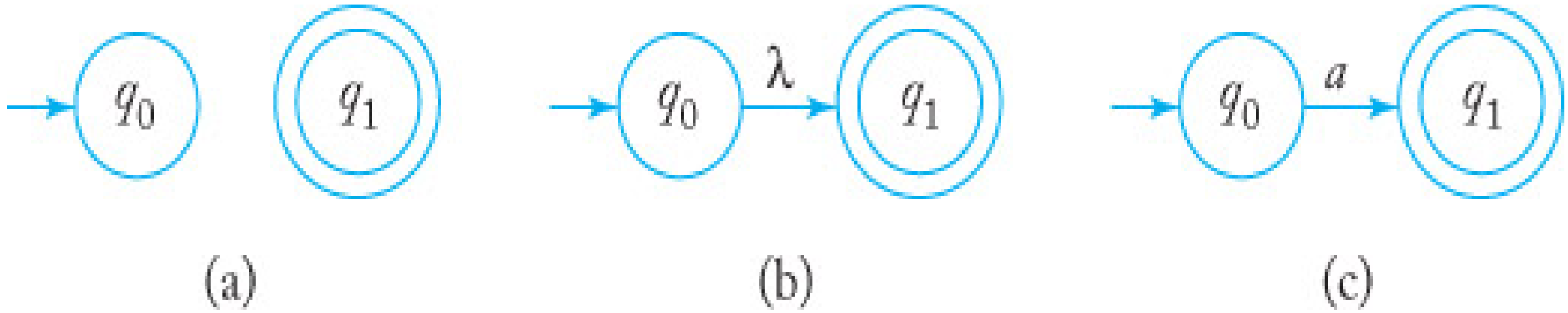
Four Theorems in this Chapter



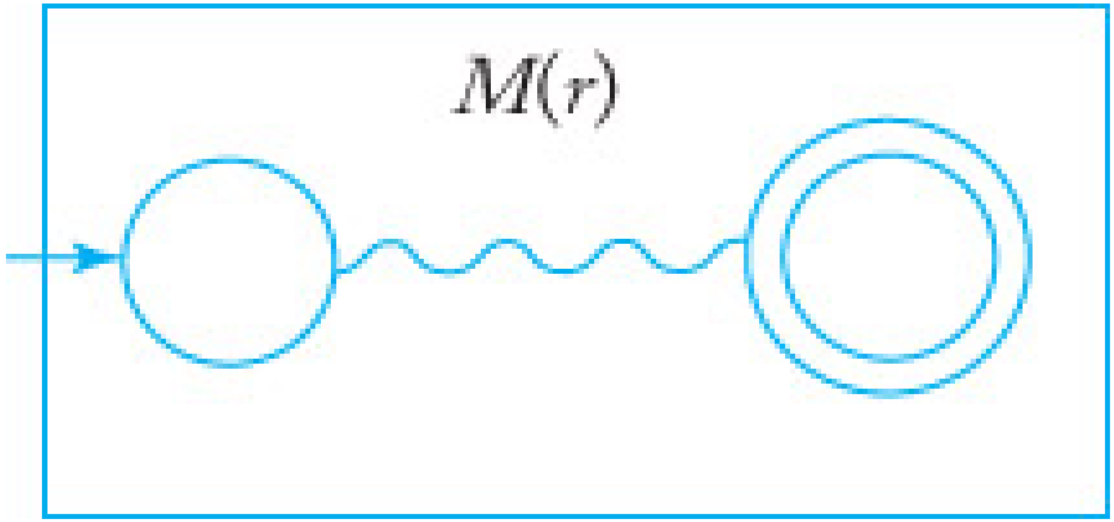
RE $\Rightarrow$ FA

- Theorem 3.1 RE $\Rightarrow$ NFA
- Proof – constructive proof based on recursive definition of RE
  - Step 1: construct nfa for each basic element defined in rule 1
  - Step 2: construct nfa for each operation defined in rule 2

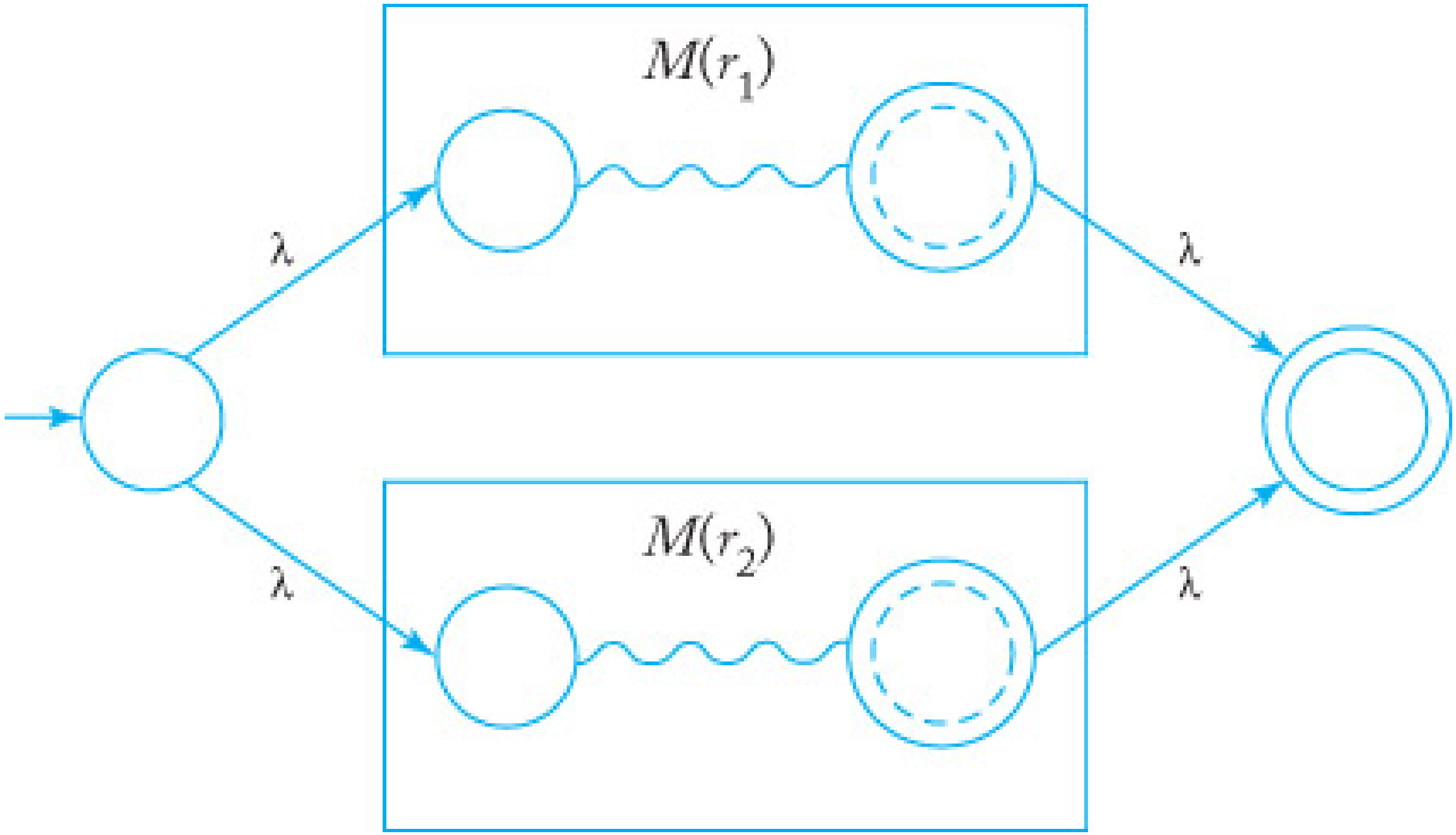
RE $\Rightarrow$ FA or TG proof step 1: an nfa for basic elements of RE



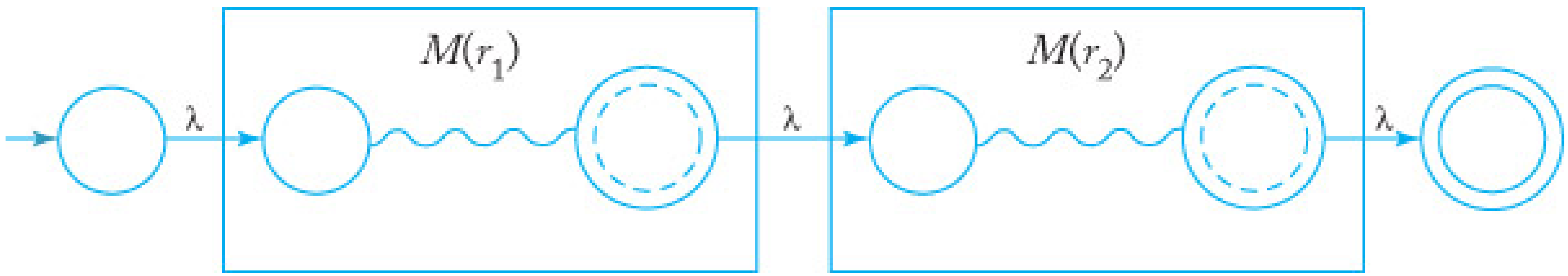
Schematic representation of an nfa accepting L(r)



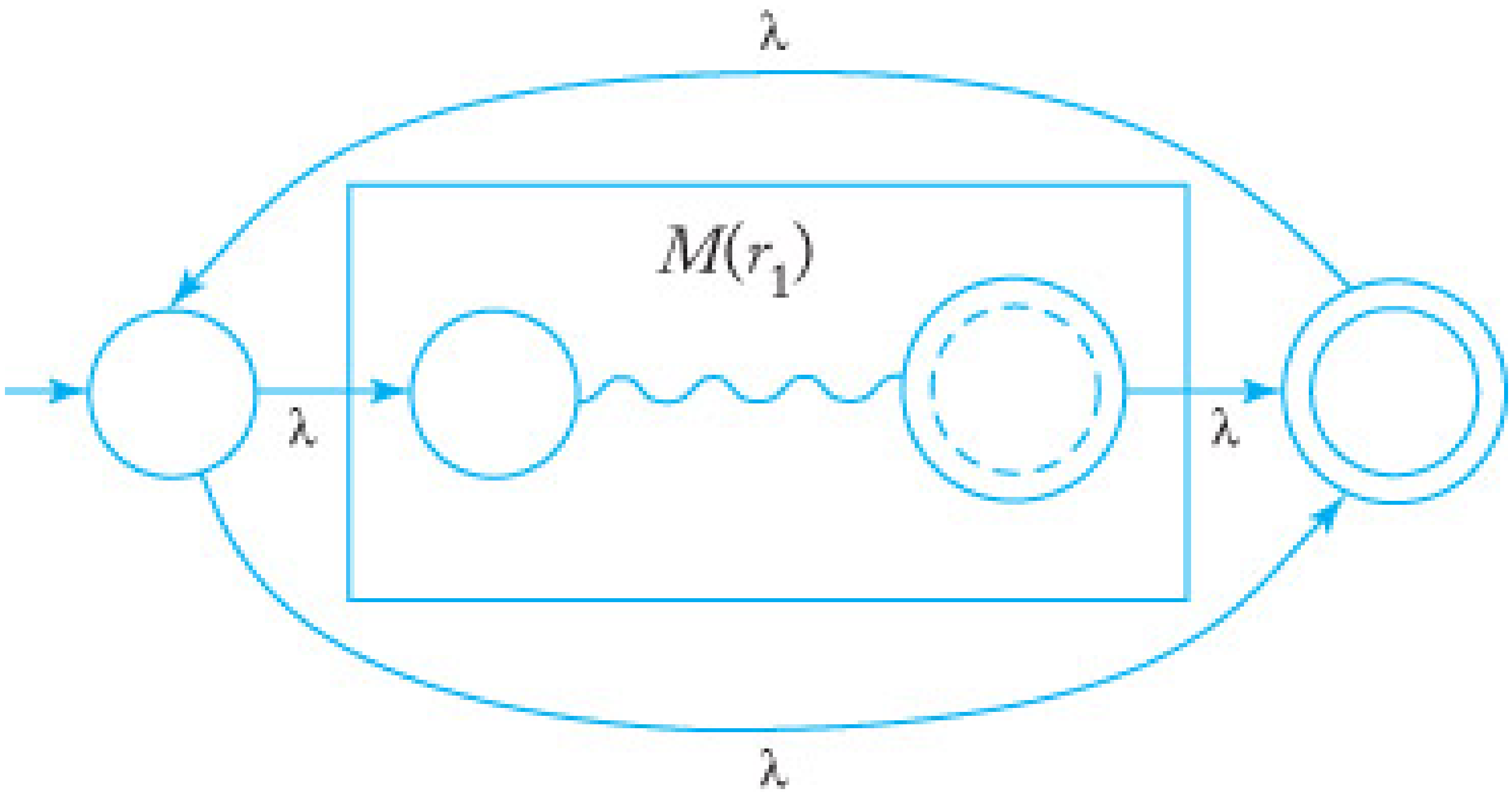
RE $\Rightarrow$ FA proof Step 2-1: nfa for $L(r_1 + r_2)$



RE ⇒ FA proof Step 2-2: nfa for $L(r_1 r_2)$



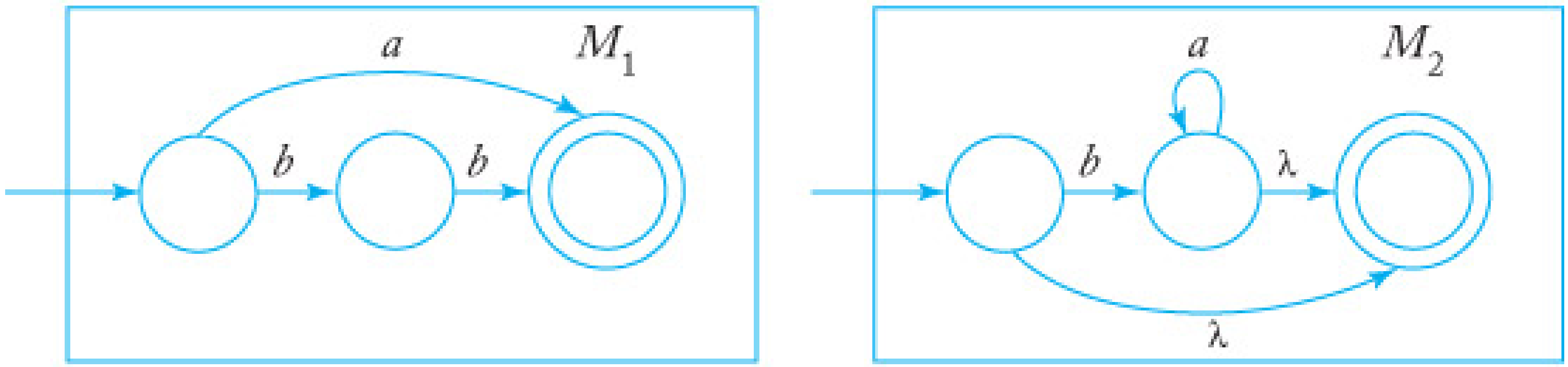
RE $\Rightarrow$ FA proof Step 2-3: nfa for $L(r_1{*})$



RE $\Rightarrow$ FA Example 3.7

- Find an nfa that accepts L(r), where
  - $r = (a + bb)^*(ba^* + \lambda)$
- We first construct nfa for
  - $r_1 = (a+bb)$ and
  - $r_2 = (ba^* + \lambda)$
- Then putting them together according to Theorem 3.1 for r
  - $r_1^* \, r_2$

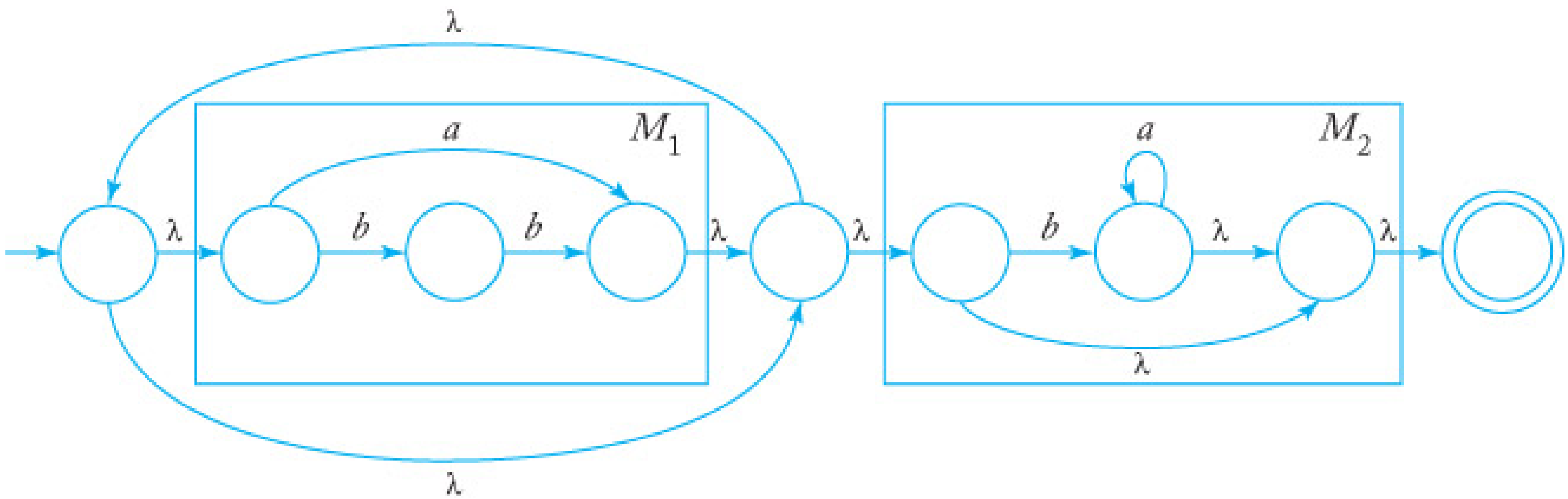
(a+bb) and (ba* + λ)



(a)

(b)

Putting them together
(a + bb)*(ba* + λ)



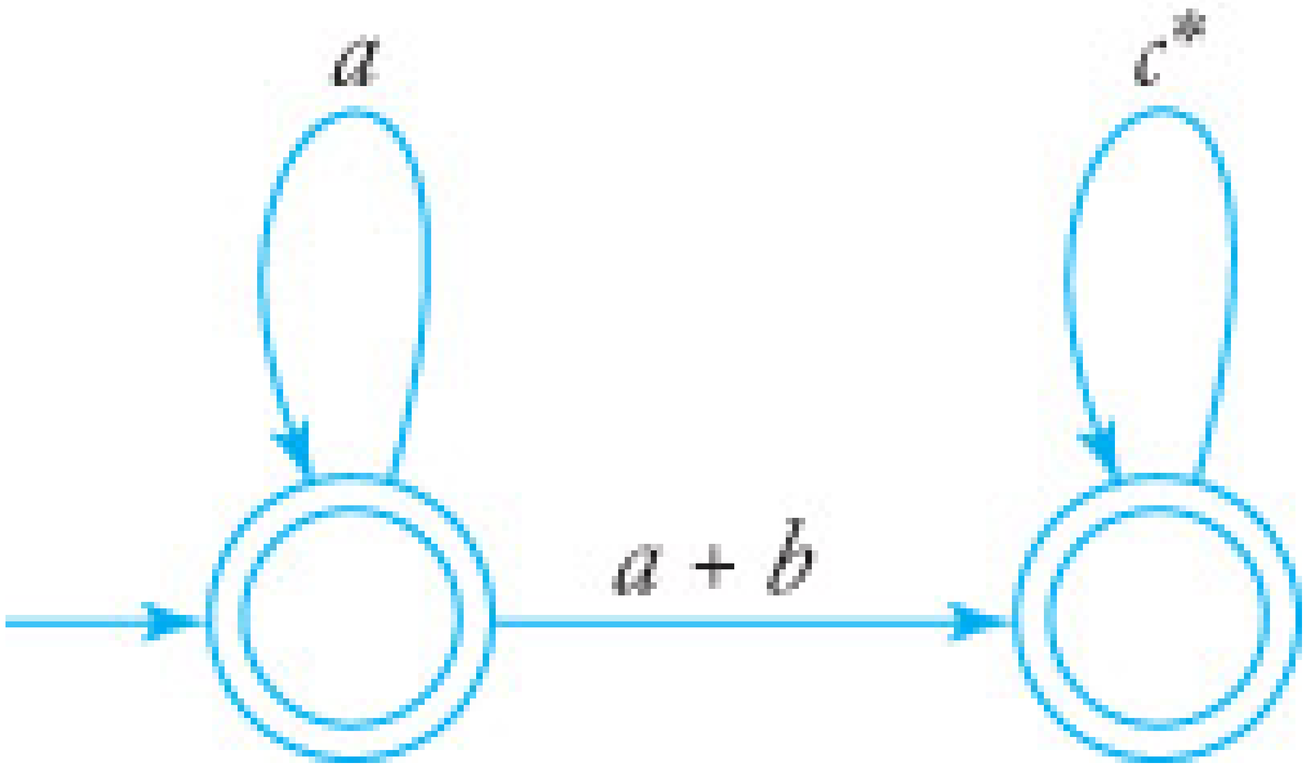
RE to TG In-class exercises

- Convert RE to TG:
  - $(a+b)^*$
  - $(a+b)^*a(a+b)^*$
  - $(a+b)^*aaa$
  - $a(a+b)^*b$
  - Even-Even: even number of a's and even number of b's

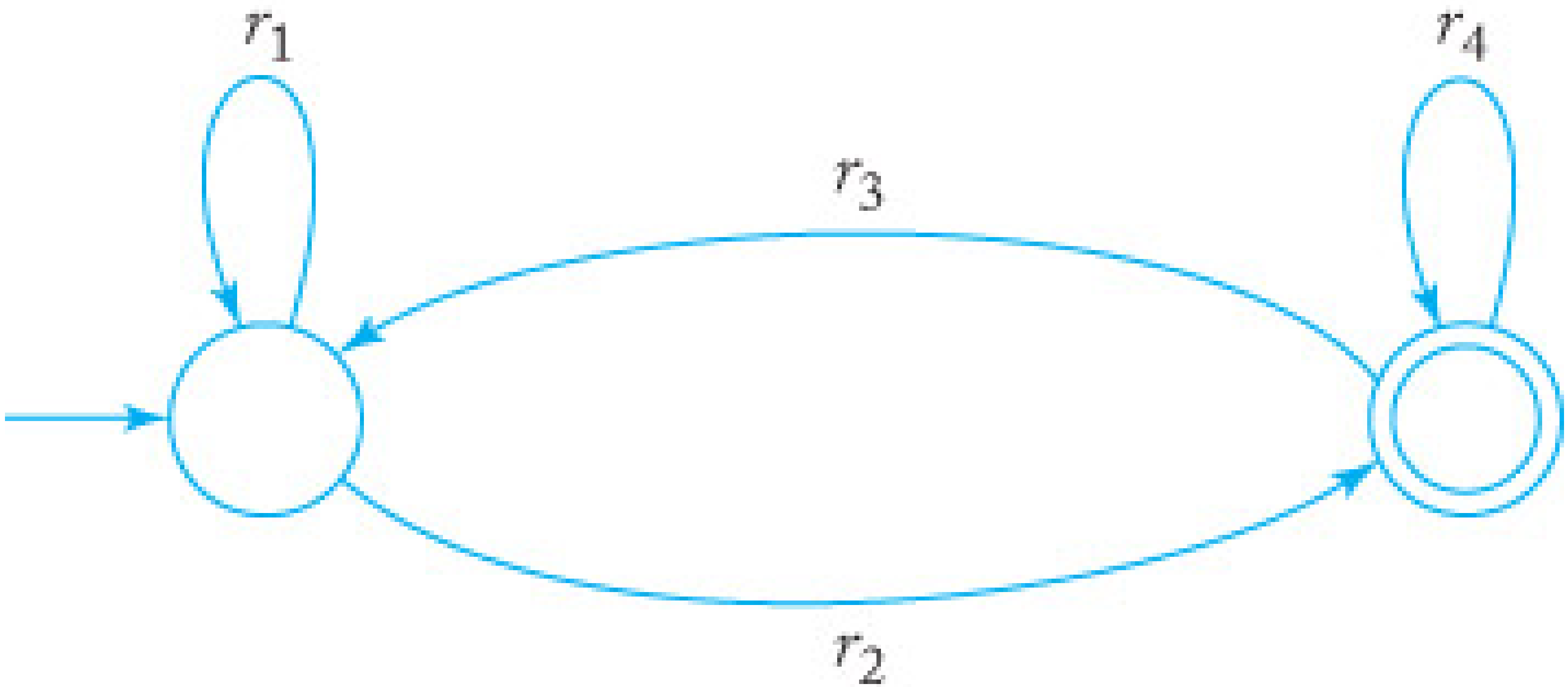
TG $\Rightarrow$ RE

- Theorem 3.2
  - Let L be a regular language. Then there exists a RE r such that L = L(r).
  - Proof: If L is regular, there exists an nfa.
    - Convert nfa to a two-state complete GTG
    - Apply the procedure nfa-to-rex $\Rightarrow$ RE
  - GTG – generalized transition graphs

An Example of GTG ⇒ RE
L(a* +a*(a+b)c*)



Two-state complete GTG to $r_1^* r_2(r_4 + r_3 r_1^* r_2)^*$



Procedure: nfa-to-rex

- Page 83-84
- Step 1 to step 6
- This is a reading assignment

Grammars -- 1

- **Grammar**
  - a scheme for specifying the sentences allowed in the language, indicating the syntactic rules for combining words into well-formed phrases and clauses
  - Defined by Feighenbaum et al.
  - *Natural language understanding* has long been a goal of AI researchers
  - Example: google cross-language info retrieval

Grammars -- 2

- MIT linguist *Noam Chomskey* did the seminal work in the systematic and mathematical study of language syntax →computational linguistics
- **Formal language** – a set of strings composed of a vocabulary of symbols according to rules of grammar

3.3 Regular Grammar

- One way of describing regular language: right- and left-linear grammar
- Definition 3.3
  - A grammar $G = (V, T, S, P)$ is said to be right-linear if all productions are of the form
    - $A \to xB$,
    - $A \to x$,
    - where $A, B \in V$, and $x \in T^*$.
- A regular grammar is one that is either right-linear or left-linear

Example 3.13
right- and left-linear grammars

- $G_1 = (\{S\}, \{a, b\}, S, P_1)$, with P_1 given as
  - $S \to abS \mid a$ is right-linear
  - A derivation with G_1: $S \Rightarrow abS \Rightarrow ababS \Rightarrow ababa$
  - $L(G_1)$ is $L((ab)^*a)$
- $G_2 = (\{S, S_1, S_2\}, \{a, b\}, S, P_2)$, with P_2 as
  - $S \to S_1ab$,
  - $S_1 \to S_1ab \mid S_2$,
  - $S_2 \to a$ is left-linear
  - $L(G_2) = ?$
- Both G_1 and G_2 are regular grammars.

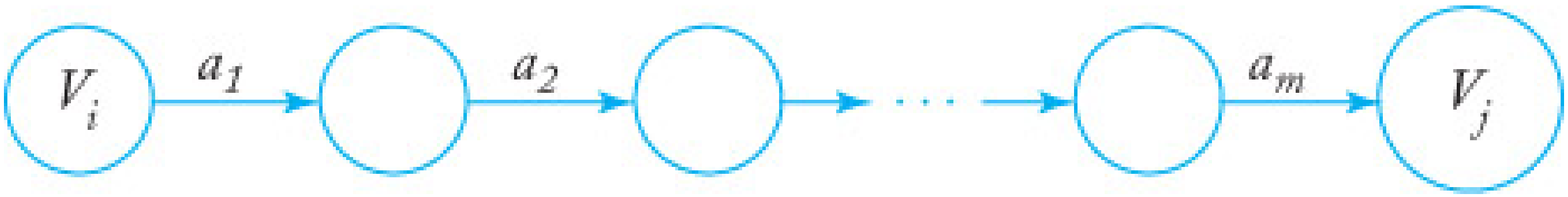
Example 3.14
a non-Regular Grammar

- $S \rightarrow A$
- $A \rightarrow aB \mid \lambda$
- $B \rightarrow Ab$

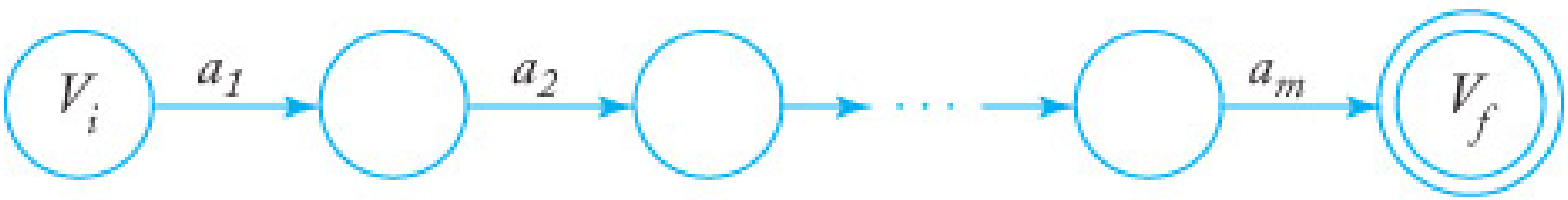
Right-Linear Grammars Generate RL - 1

- Theorem 3.3 RG $\Rightarrow$ FA
  - Let G = (V, T, S, P) be a right-linear grammar. Then L(G) is a regular language.
  - Proof. Assume that $V = \{V_0, V_1, \ldots\}$, $S = V_0$
    - For each production, we convert it into a corresponding transition
    - We only have two possible forms of production in a right-linear grammar
    - Do you know which two?

Right-Linear Grammars Generate RL - 2



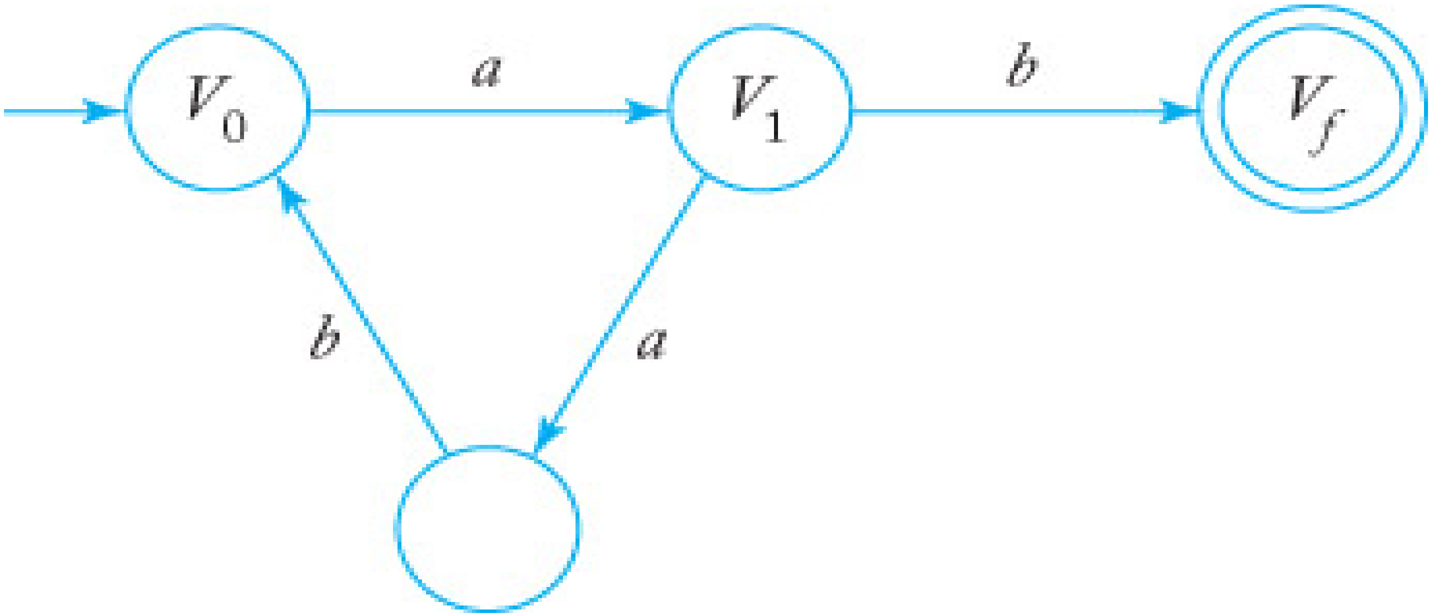
Represents $V_i \rightarrow a_1 a_2 \ldots a_m V_j$



Represents $V_i \rightarrow a_1 a_2 \ldots a_m$

Example 3.15
Construct a FA that accepts the language generated by a RG

- $V_0 \rightarrow a\ V_1$
- $V_1 \rightarrow ab\ V_0 | b$



Right-Linear Grammars for Regular Language

- Theorem 3.4 FA $\Rightarrow$ RG
  - If L is a regular language on the alphabet Σ, then there exists a right-linear grammar G = (V, T, S, P) such that L = L(G).
  - Proof – constructive
    - Let M = $(Q, \Sigma, \delta, q_0, F)$ be a dfa that accepts L
    - Construct the right-linear grammar G with
      - $V = \{q_0, q_1, \ldots, q_n\}$
      - $S = q_0$
      - Construct a production for each transition

Equivalence of RL and RG

- Putting Theorems 3.4 and 3.5 together, we arrive at the equivalence of RL and RG
- Theorem 3.6:
  - A language L is regular if and only if there exists a regular grammar G such that L = L(G)

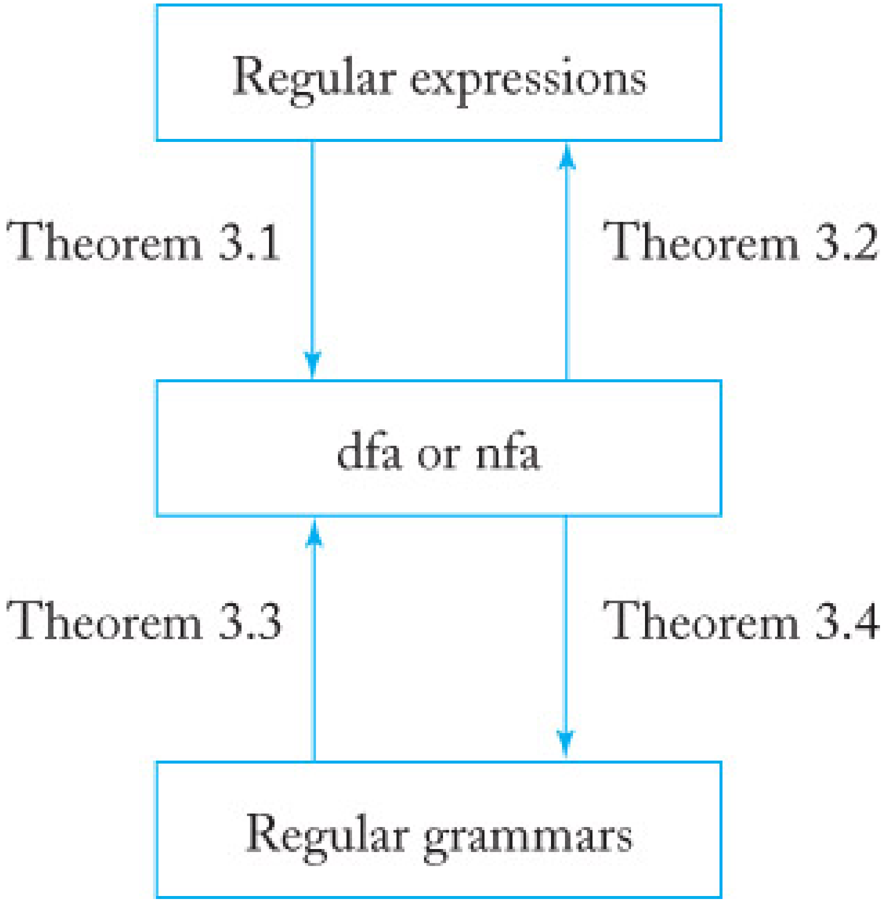
In-class Exercises

- Represent the following RL in 4 representations of RL (RE, TG, FA, RG):
  - Even number of a’s followed by odd number of b’s.

Summary-1

- We now have all 4 representations of RL:
  - FA, TG, RE, RG
- In some instance, one of them may be most suitable
- They are equally powerful in representing RL
- When you need all 4 representations for a given RL, you can start to work on one of the representation that is easiest to you first then ….

Summary-2


Regular expressions
Theorem 3.1
Theorem 3.2
dfa or nfa
Theorem 3.3
Theorem 3.4
Regular grammars