

# Fundamentele Informatica 1 (I&E)

najaar 2014

<http://www.liacs.leidenuniv.nl/~vlietrvan1/fi1ie/>

**Rudy van Vliet**

rvvliet(at)liacs(dot)nl

college 5, 10 november 2014

3.2 Nondeterministic Finite Automata

3.3 The Nondeterminism in an NFA Can Be Eliminated

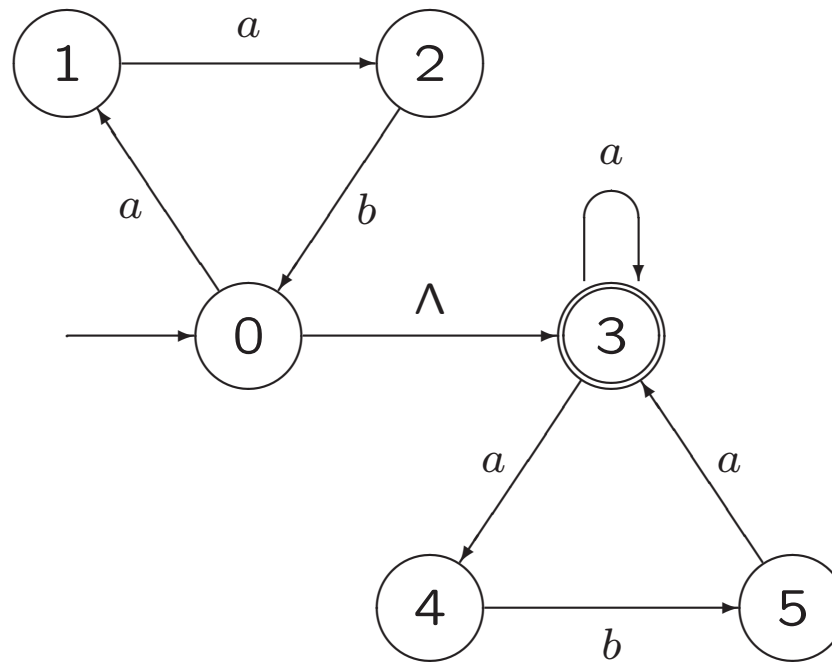
3.4 Kleene's Theorem, Part 1

## 3.2 Nondeterministic Finite Automata

reg. languages	FA	reg. grammar	reg. expression
cf. languages	PDA	cf. grammar	
re. languages	TM	unrestr. grammar	



**Example 3.9.** Accepting the Language  $\{aab\}^*\{a, aba\}^*$



*Slide from lecture 2:*

**Definition 2.11.** A Finite Automaton

A *finite automaton* (FA) is a 5-tuple  $(Q, \Sigma, q_0, A, \delta)$ , where

$Q$  is a finite set of *states*

$\Sigma$  is a finite *input alphabet*

$q_0 \in Q$  is the *initial* state

$A \subseteq Q$  is the set of *accepting* states

$\delta : Q \times \Sigma \rightarrow Q$  is the *transition function*

For any state  $q$  of  $Q$  and any symbol  $\sigma \in \Sigma$ , we interpret  $\delta(q, \sigma)$  as the state to which the FA moves, if it is in state  $q$  and receives the input  $\sigma$ .

**Definition 3.12.** A **Nondeterministic** Finite Automaton

A *nondeterministic finite automaton* (NFA) is a 5-tuple  $(Q, \Sigma, q_0, A, \delta)$ , where

$Q$  is a finite set of *states*

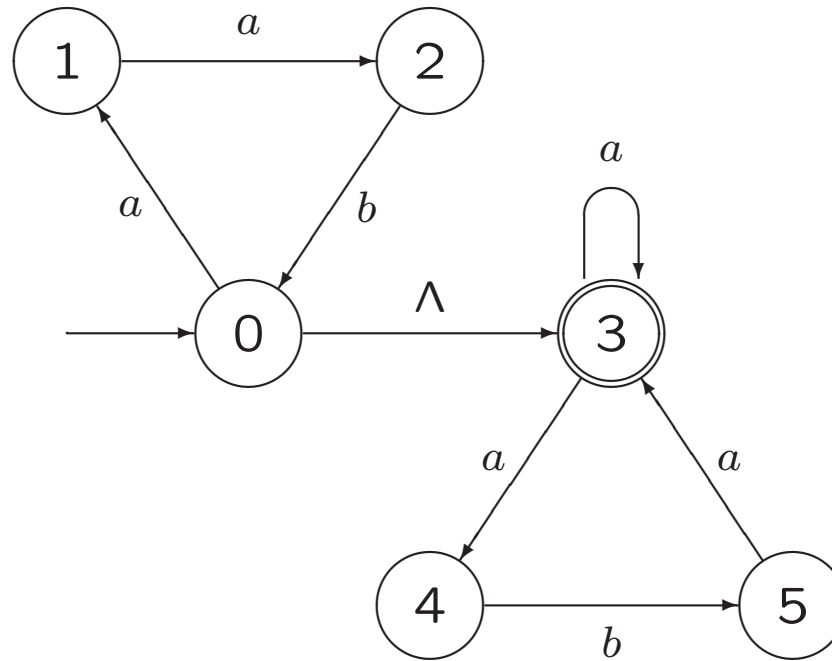
$\Sigma$  is a finite *input alphabet*

$q_0 \in Q$  is the *initial* state

$A \subseteq Q$  is the set of *accepting* states

$\delta : \dots$  is the *transition* function

**Example 3.9.** Accepting the Language  $\{aab\}^*\{a, aba\}^*$



$$\delta(2, b) = \dots$$

$$\delta(3, a) = \dots$$

$$\delta(5, b) = \dots$$

$$\delta(0, \Lambda) = \dots$$

**Definition 3.12.** A **Nondeterministic** Finite Automaton

A *nondeterministic finite automaton* (NFA) is a 5-tuple  $(Q, \Sigma, q_0, A, \delta)$ , where

$Q$  is a finite set of *states*

$\Sigma$  is a finite *input alphabet*

$q_0 \in Q$  is the *initial* state

$A \subseteq Q$  is the set of *accepting* states

$\delta : Q \times (\Sigma \cup \{\Lambda\}) \rightarrow 2^Q$  is the *transition* function

For every element  $q$  of  $Q$  and every element  $\sigma$  of  $\Sigma \cup \{\Lambda\}$ , we interpret  $\delta(q, \sigma)$  as the **set of states** to which **the NFA can move**, if it is in state  $q$  and receives the input  $\sigma$ , **or, if  $\sigma = \Lambda$ , the set of states other than  $q$  to which the NFA can move from state  $q$  without receiving any input symbol.**

The  $\Lambda$ -Closure of a Set of Strings  
(instead of formal **Definition 3.13**):

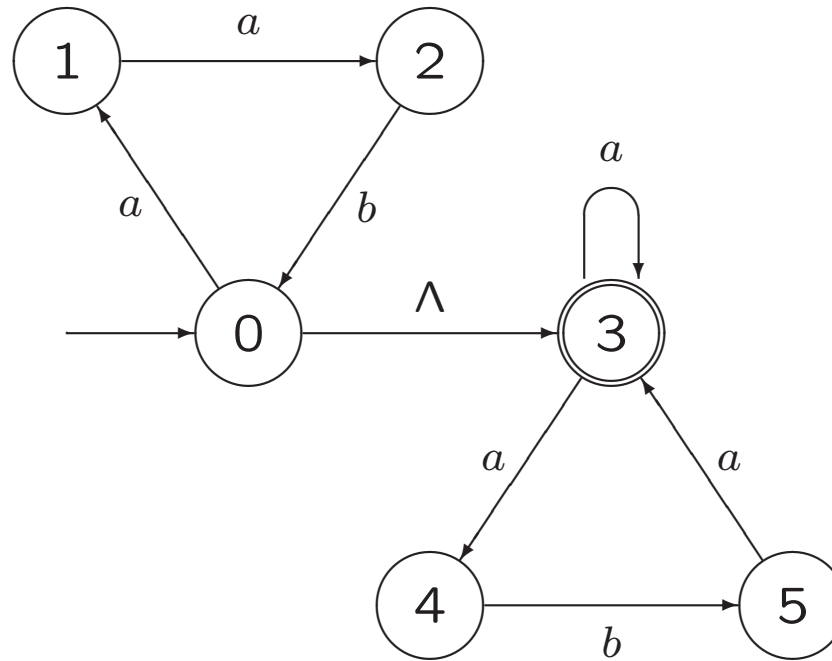
Suppose  $M = (Q, \Sigma, q_0, A, \delta)$  is an NFA,  
and  $S \subseteq Q$  is a set of states.

The  $\Lambda$ -closure of  $S$  is the set  $\Lambda(S)$  that consists of all states that  
can be reached from states in  $S$  via 0 or more  $\Lambda$ -transitions.

**N.B.:** by definition,  $\Lambda(S)$  includes all states from  $S$ .



**Example 3.9.** Accepting the Language  $\{aab\}^*\{a, aba\}^*$



$$\Lambda(\{0\}) = \dots$$

$$\Lambda(\{0, 3\}) = \dots$$

$$\Lambda(\{0, 3, 4\}) = \dots$$

The *extended transition function*  $\delta^*$   
(instead of formal **Definition 3.14**):

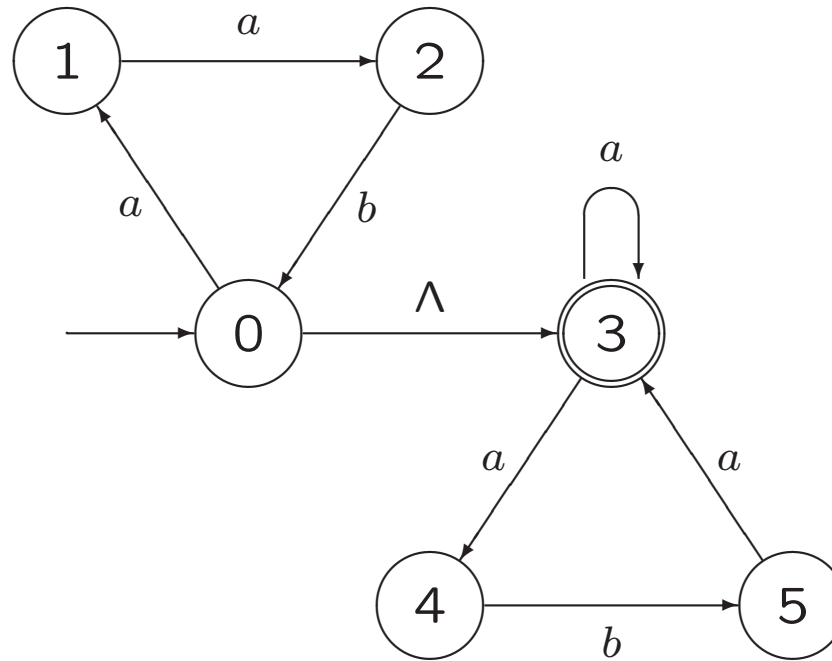
Let  $M = (Q, \Sigma, q_0, A, \delta)$  be an NFA.

$\delta^*(q, x)$  is the set of states that the NFA can get to  
by starting at  $q$  and using the symbols in the string  $x$ .

A string  $x \in \Sigma^*$  is accepted by  $M$  if  $\delta^*(q_0, x) \cap A \neq \emptyset$ .

The language  $L(M)$  accepted by  $M$  is the set of all strings accepted by  $M$ .

**Example 3.9.** Accepting the Language  $\{aab\}^*\{a, aba\}^*$



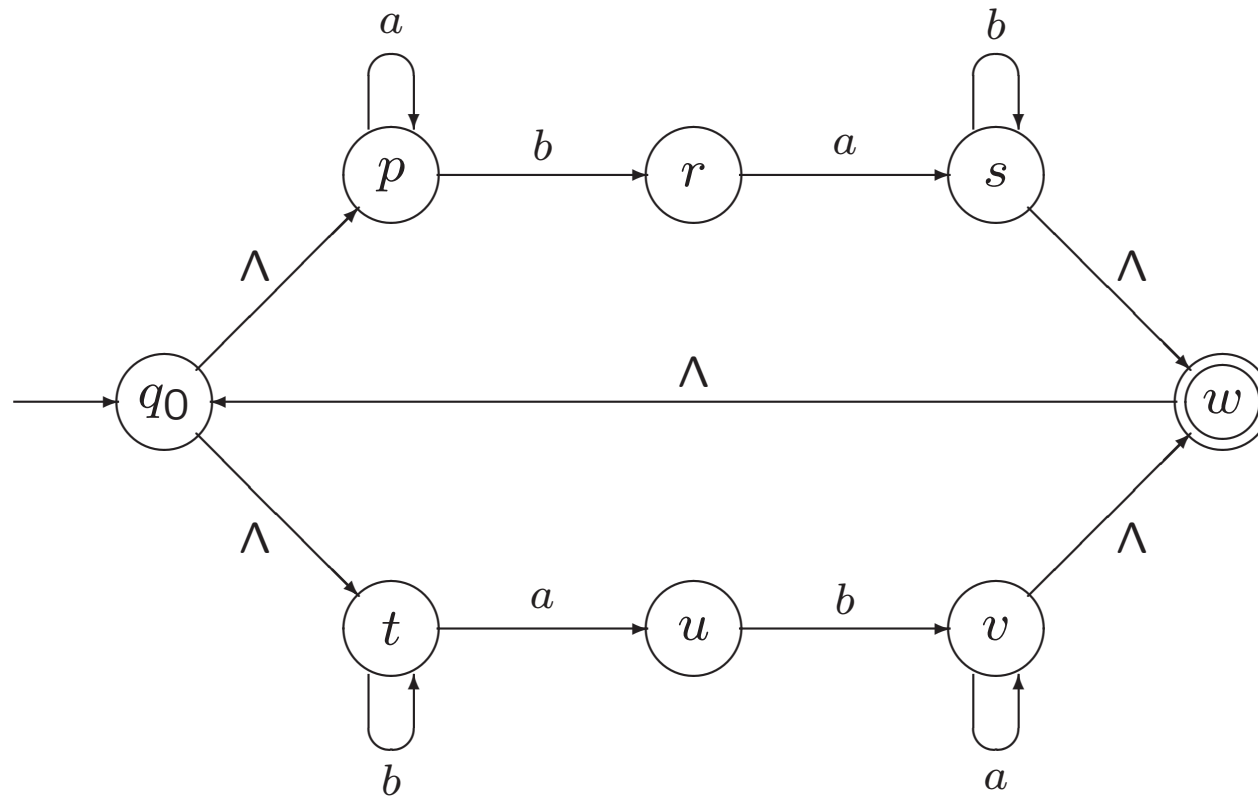
$$\delta^*(0, \Lambda) = \dots$$

$$\delta^*(0, a) = \dots$$

$$\delta^*(0, aa) = \dots$$

$$\delta^*(0, aab) = \dots$$

**Example 3.15.** Applying the Definitions of  $\Lambda(S)$  and  $\delta^*$



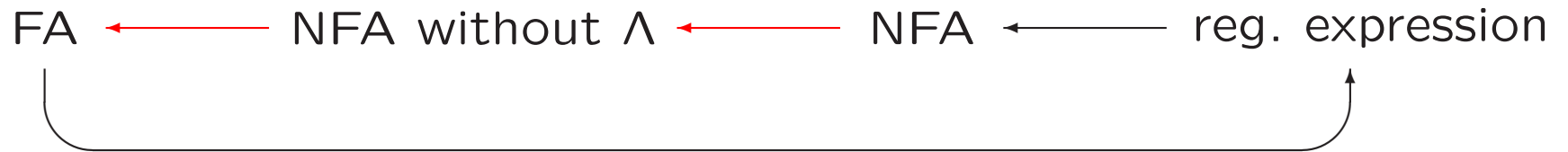
$$\delta^*(q_0, \Lambda) = \dots$$

$$\delta^*(q_0, a) = \dots$$

$$\delta^*(q_0, ab) = \dots$$

$$\delta^*(q_0, aba) = \dots$$

### 3.3 The Nondeterminism in an NFA Can Be Eliminated



### **Theorem.**

For every language  $L \subseteq \Sigma^*$  accepted by an FA  $M = (Q, \Sigma, q_0, A, \delta)$ , there is an NFA  $M_1 = (Q_1, \Sigma, q_1, A_1, \delta_1)$  that also accepts  $L$ .

In other words: NFAs are at least as 'strong' as FAs.

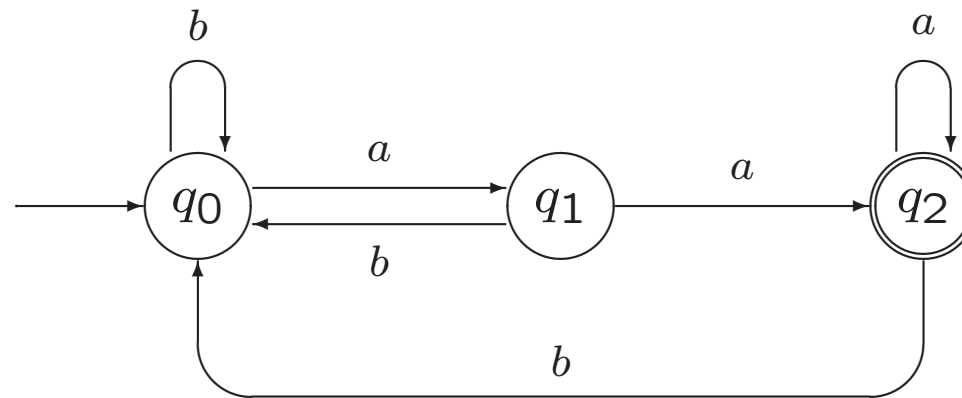
**Proof...**

A slide from lecture 3:

**Example 2.1.**

A finite automaton for accepting

$$L_1 = \{x \in \{a, b\}^* \mid x \text{ ends with } aa\}$$



### **Theorem 3.17.**

For every language  $L \subseteq \Sigma^*$  accepted by an NFA  $M = (Q, \Sigma, q_0, A, \delta)$ , there is an NFA  $M_1$  with no  $\Lambda$ -transitions that also accepts  $L$ .

In other words:  $\Lambda$ -transitions in an NFA can be avoided

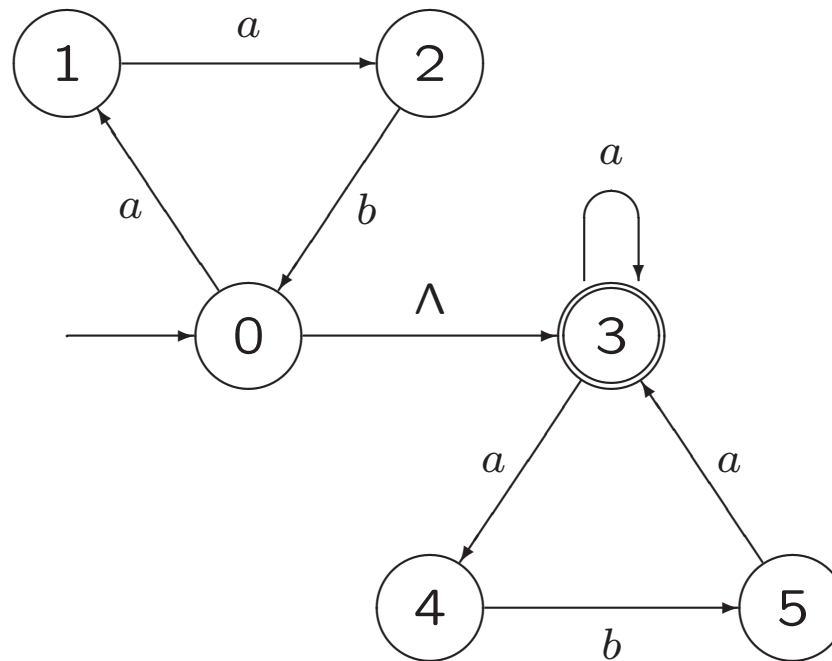
The formal proof of this result does not have to be known for the exam. However, the construction of  $M_1$  has to be known.



**Example 3.19.** Eliminating  $\Lambda$ -transitions from an NFA

$q$	$\Lambda(q)$	$\delta^*(q, a)$	$\delta^*(q, b)$
1			
2			
3			
4			
5			

**Example 3.9.** Accepting the Language  $\{aab\}^*\{a, aba\}^*$



### **Theorem 3.18.**

For every language  $L \subseteq \Sigma^*$  accepted by an NFA  $M = (Q, \Sigma, q_0, A, \delta)$ , there is an FA  $M_1 = (Q_1, \Sigma, q_1, A_1, \delta_1)$  that also accepts  $L$ .

In other words: FAs are at least as 'strong' as NFAs.

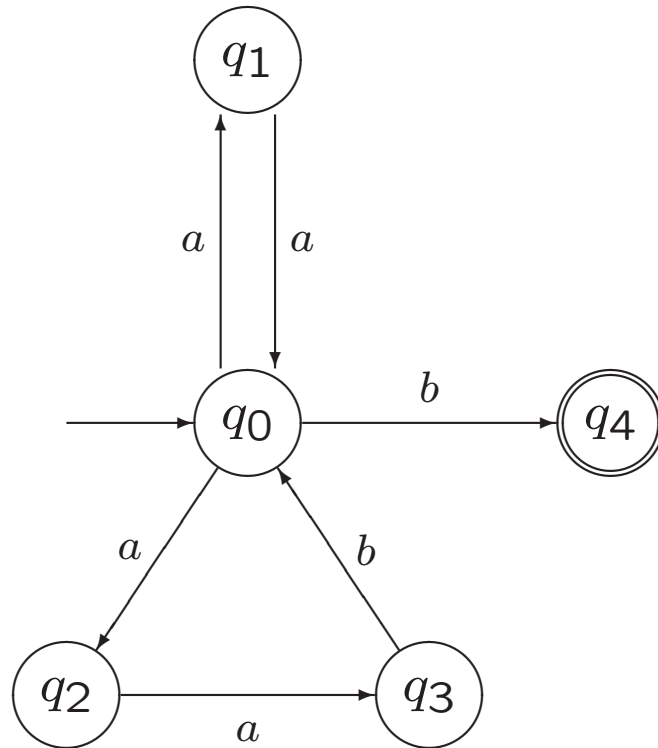
The formal proof of this result does not have to be known for the exam. However, the construction of  $M_1$  has to be known.

### **Corollary.**

FAs are equally 'strong' as NFAs.

### Example 3.21.

Using the Subset Construction to Eliminate Nondeterminism



$q$	$\delta(q, a)$	$\delta(q, b)$
0		
1		
2		
3		
4		

**Example 3.23.**

Converting an NFA with  $\Lambda$ -Transitions to an FA

Study this example yourself

## 3.4 Kleene's Theorem, Part 1

### Theorem 3.25. Kleene's Theorem, Part 1.

For every alphabet  $\Sigma$ , every regular language over  $\Sigma$  can be accepted by a finite automaton.

The formal proof of this result does not have to be known for the exam. However, the construction of an NFA from a regular expression has to be known.



A slide from lecture 4:

**Definition 3.1.** Regular Languages over an Alphabet  $\Sigma$ .

If  $\Sigma$  is an alphabet,  
the set  $\mathcal{R}$  of regular languages over  $\Sigma$  is defined as follows.

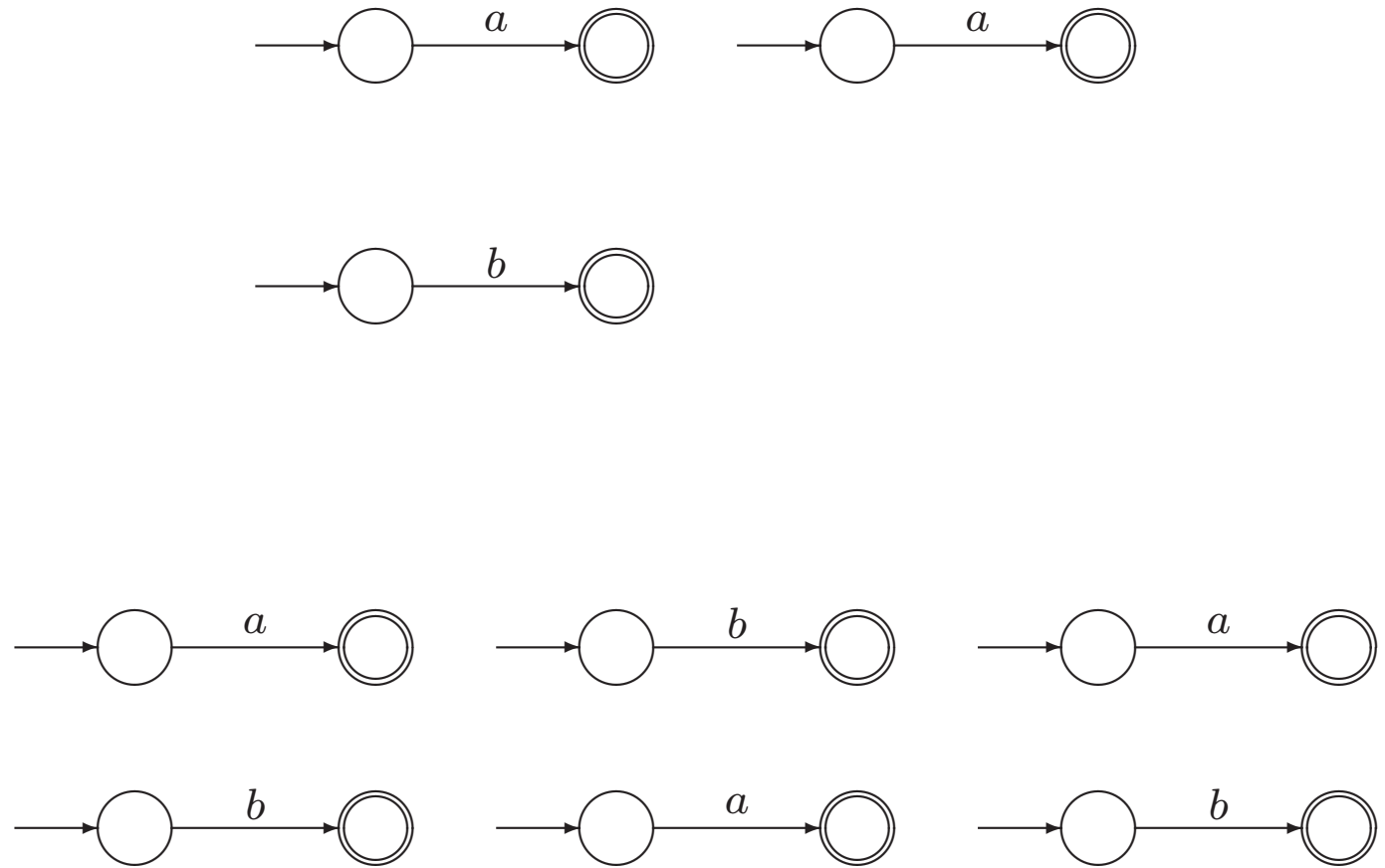
1. The language  $\emptyset$  is an element of  $\mathcal{R}$ ,  
and for every  $\sigma \in \Sigma$ , the language  $\{\sigma\}$  is in  $\mathcal{R}$ .
2. For any two languages  $L_1$  and  $L_2$  in  $\mathcal{R}$ ,  
the three languages  
 $L_1 \cup L_2$ ,  $L_1L_2$ , and  $L_1^*$   
are elements of  $\mathcal{R}$ .

(and nothing more)

$((aa + b)^*(aba)^*bab)^*$

**Example 3.28.** An NFA Corresponding to  $((aa + b)^*(aba)^*bab)^*$

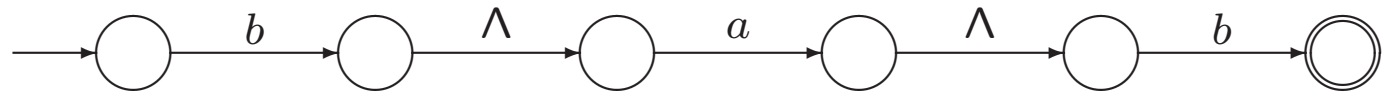
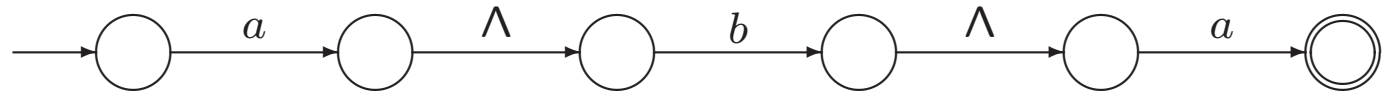
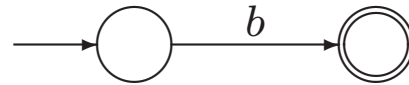
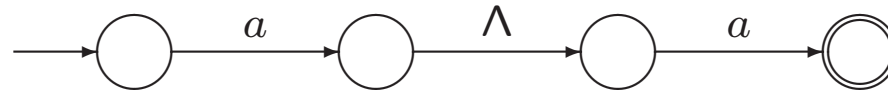
Step 1





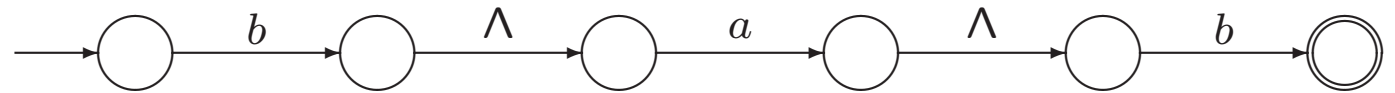
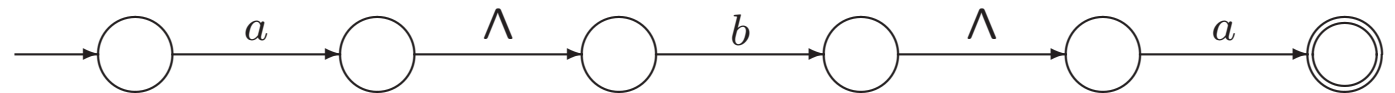
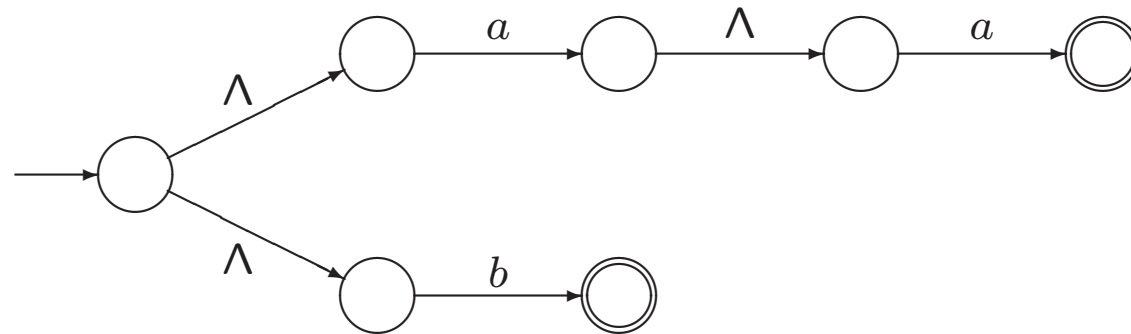
**Example 3.28.** An NFA Corresponding to  $((aa + b)^*(aba)^*bab)^*$

Step 2



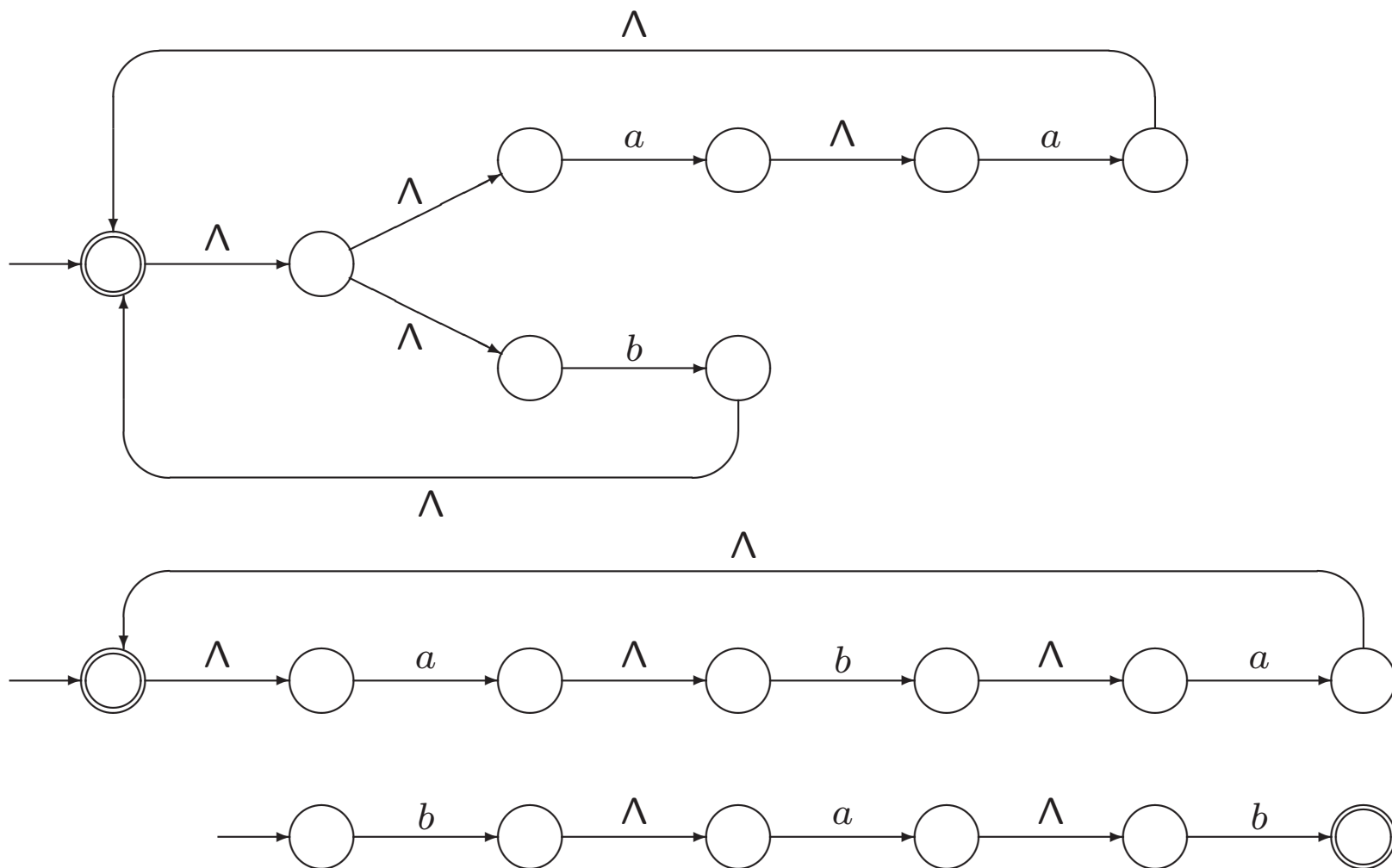
**Example 3.28.** An NFA Corresponding to  $((aa + b)^*(aba)^*bab)^*$

Step 3



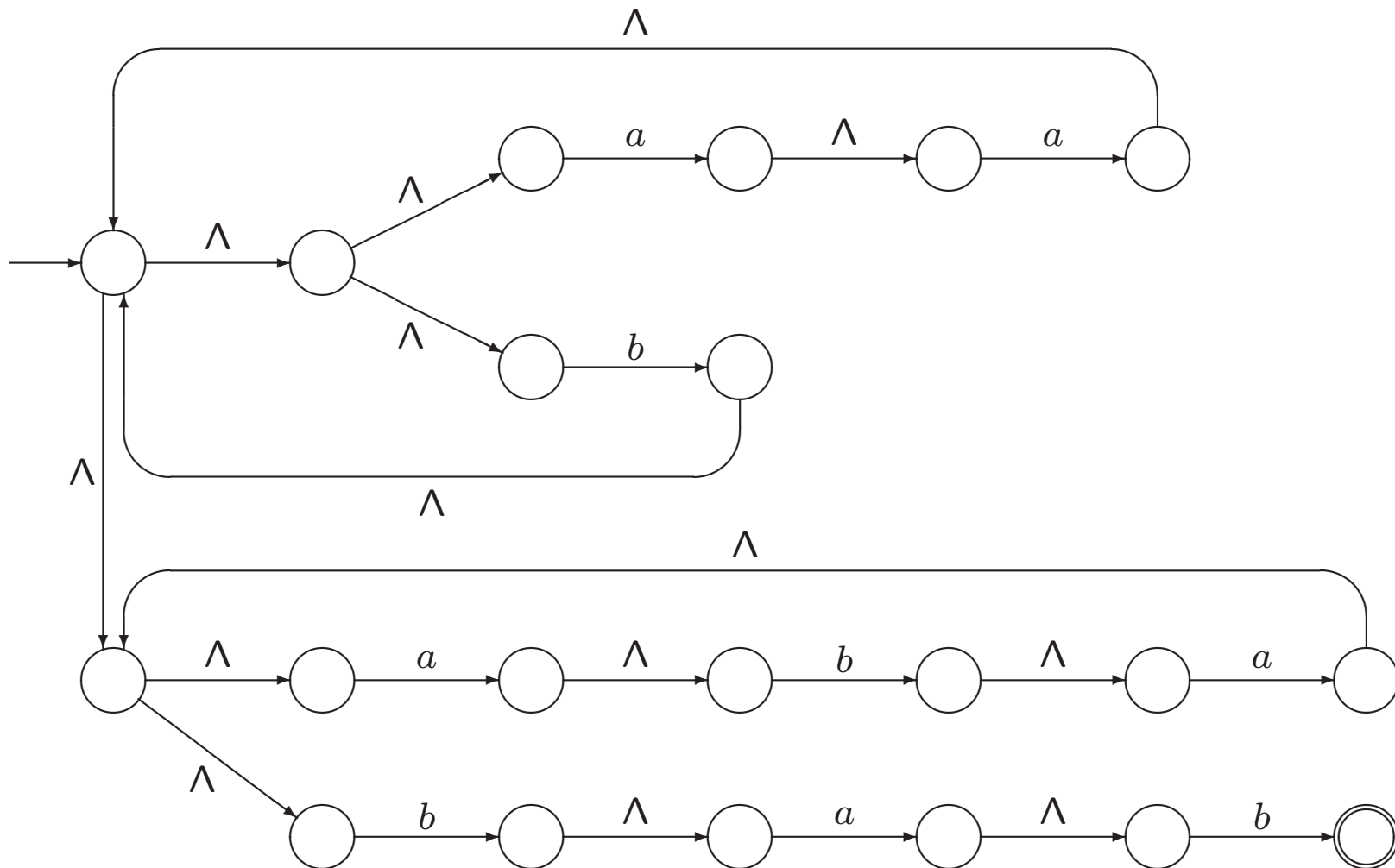
**Example 3.28.** An NFA Corresponding to  $((aa + b)^*(aba)^*bab)^*$

Step 4



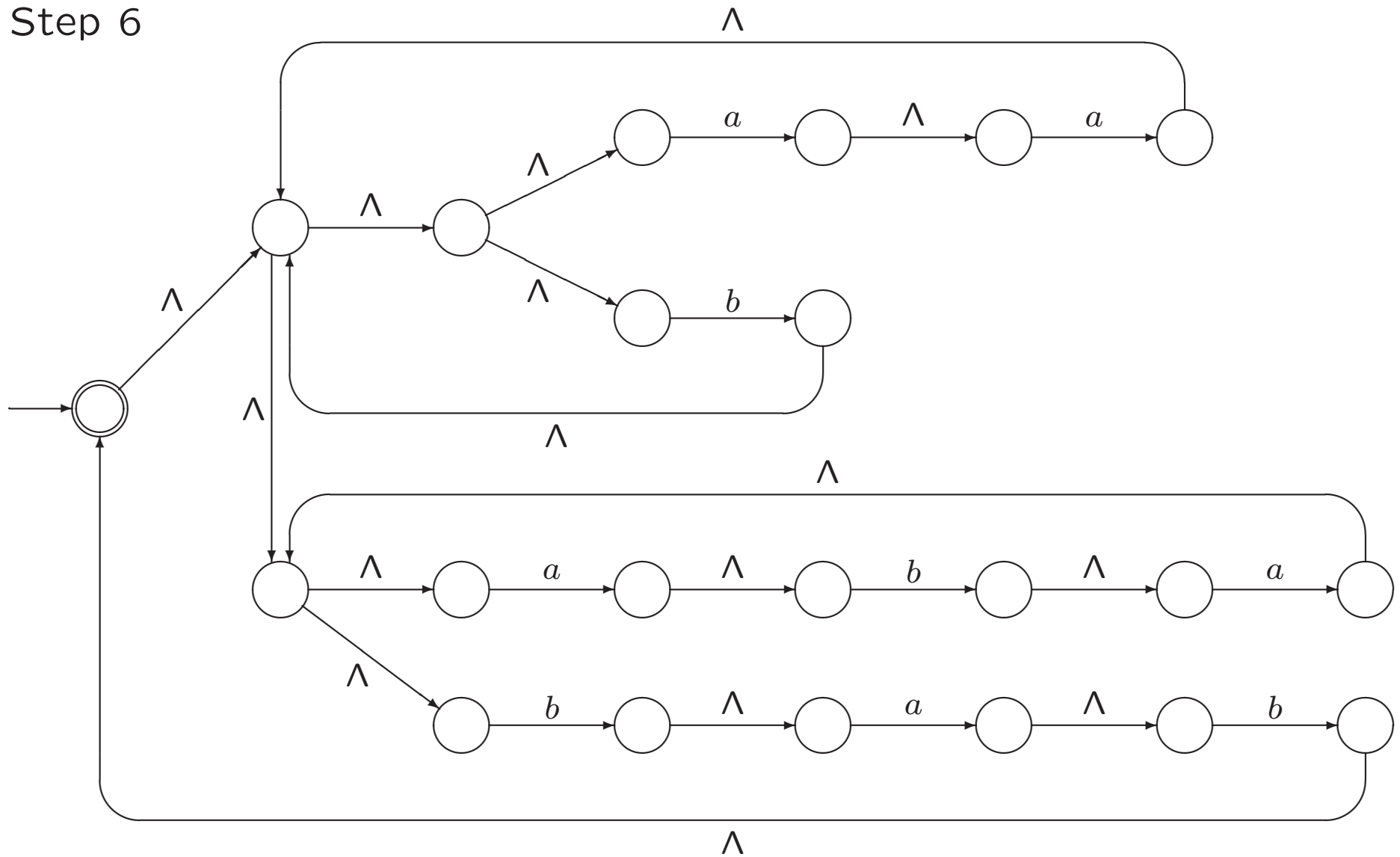
**Example 3.28.** An NFA Corresponding to  $((aa + b)^*(aba)^*bab)^*$

Step 5



**Example 3.28.** An NFA Corresponding to  $((aa + b)^*(aba)^*bab)^*$

Step 6



## Main result from Section 3.5.

### Theorem 3.30. Kleene's Theorem, Part 2.

For every finite automaton  $M = (Q, \Sigma, q_0, A, \delta)$ , the language  $L(M)$  is regular.

The proof of this result and the rest of Section 3.5 do not have to be known for the exam.

