Fundamentele Informatica 1 (I&E)

najaar 2015

`http://www.liacs.leidenuniv.nl/~vlietrvan1/fi1ie/`

Rudy van Vliet
kamer 124 Snellius, tel. 071-527 5777
rvvliet(at)liacs(dot)nl

college 5, 10 november 2015

3.2 Nondeterministic Finite Automata

3.3 The Nondeterminism in an NFA Can Be Eliminated

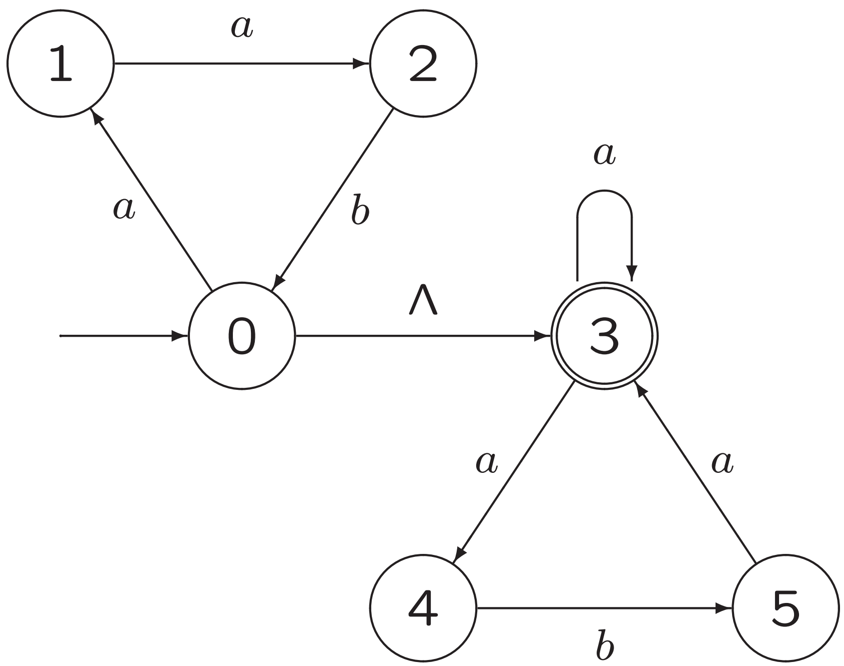
3.4 Kleene's Theorem, Part 1

3.2 Nondeterministic Finite Automata

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

$$\begin{tikzcd} \text{FA} \arrow[rrr, bend right] & \text{NFA without } \Lambda \arrow[l] & \text{NFA} \arrow[l] & \text{reg. expression} \arrow[l] \end{tikzcd}$$

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*
Σ is a finite *input alphabet*
$q_0 \in Q$ is the *initial* state
$A \subseteq Q$ is the set of *accepting* states
$\delta : Q \times \Sigma \to Q$ is the *transition* function

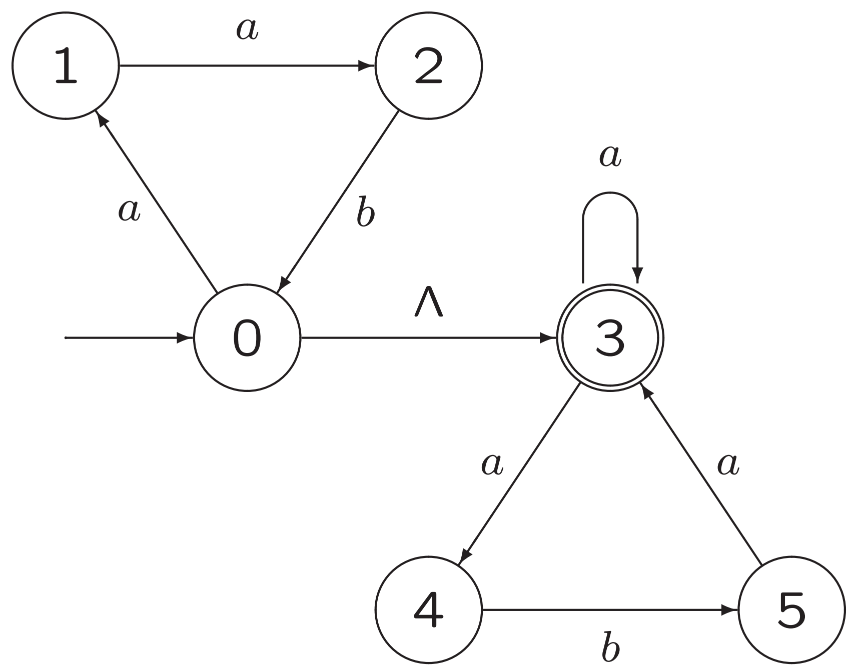
For any state q of Q and any symbol $\sigma \in \Sigma$,
we interpret $\delta(q, \sigma)$ as the state to which the FA moves,
if it is in state q and receives the input σ.

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*
Σ is a finite *input alphabet*
$q_0 \in Q$ is the *initial* state
$A \subseteq Q$ is the set of *accepting* states
$\delta :$ . . . is the *transition* function

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



$\delta(2, b) = \ldots$
$\delta(3, a) = \ldots$
$\delta(5, b) = \ldots$
$\delta(0, \Lambda) = \ldots$

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*
Σ is a finite *input alphabet*
$q_0 \in Q$ is the *initial* state
$A \subseteq Q$ is the set of *accepting* states
$\delta : Q \times (\Sigma \cup \{\Lambda\}) \to 2^Q$ is the *transition* function

For every element q of Q and every element σ 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 σ, 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.

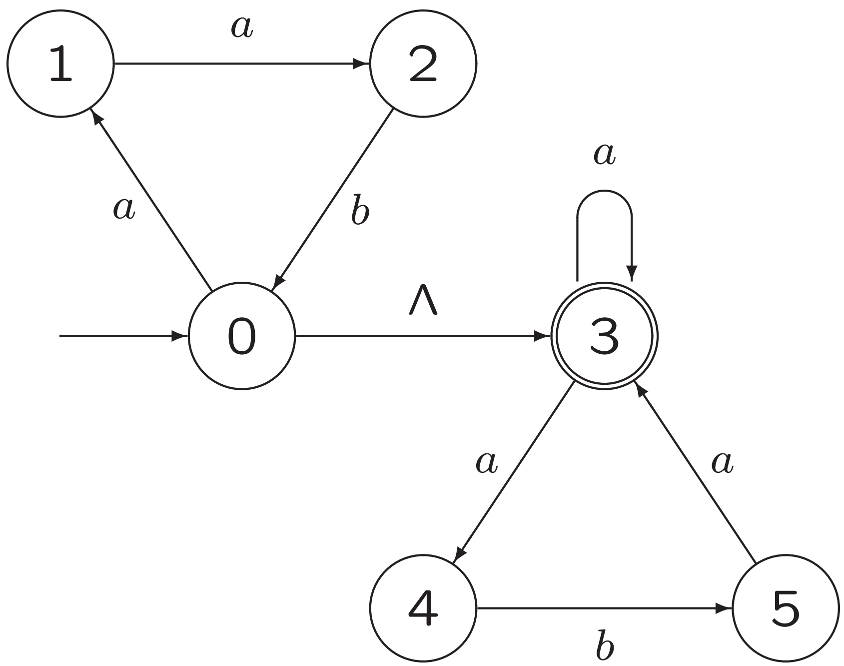
The Λ-*Closure of a Set of States*
(instead of formal **Definition 3.13**):

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

The Λ-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 Λ-transitions.

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

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



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

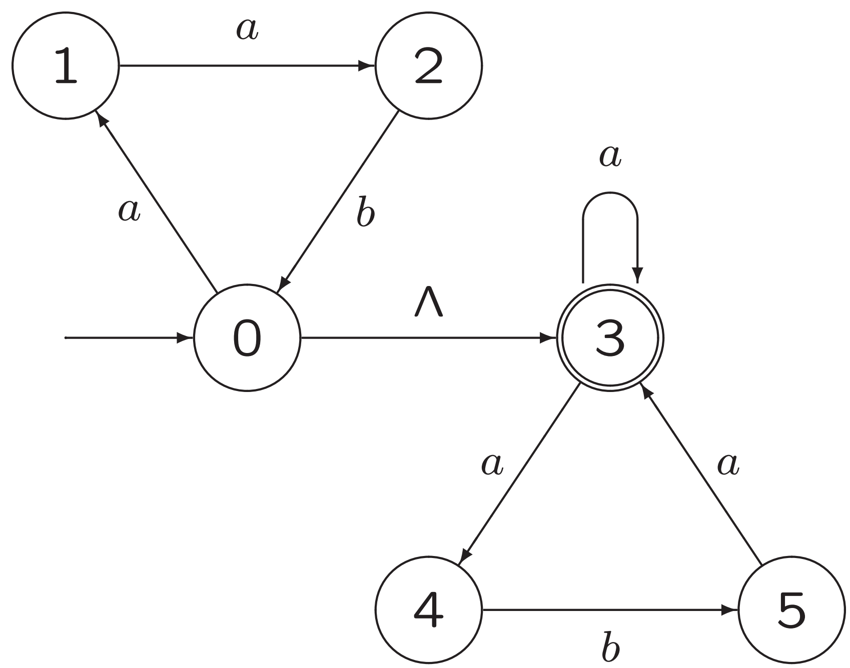
The *extended transition function* δ^*
(instead of formal **Definition 3.14**):

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.

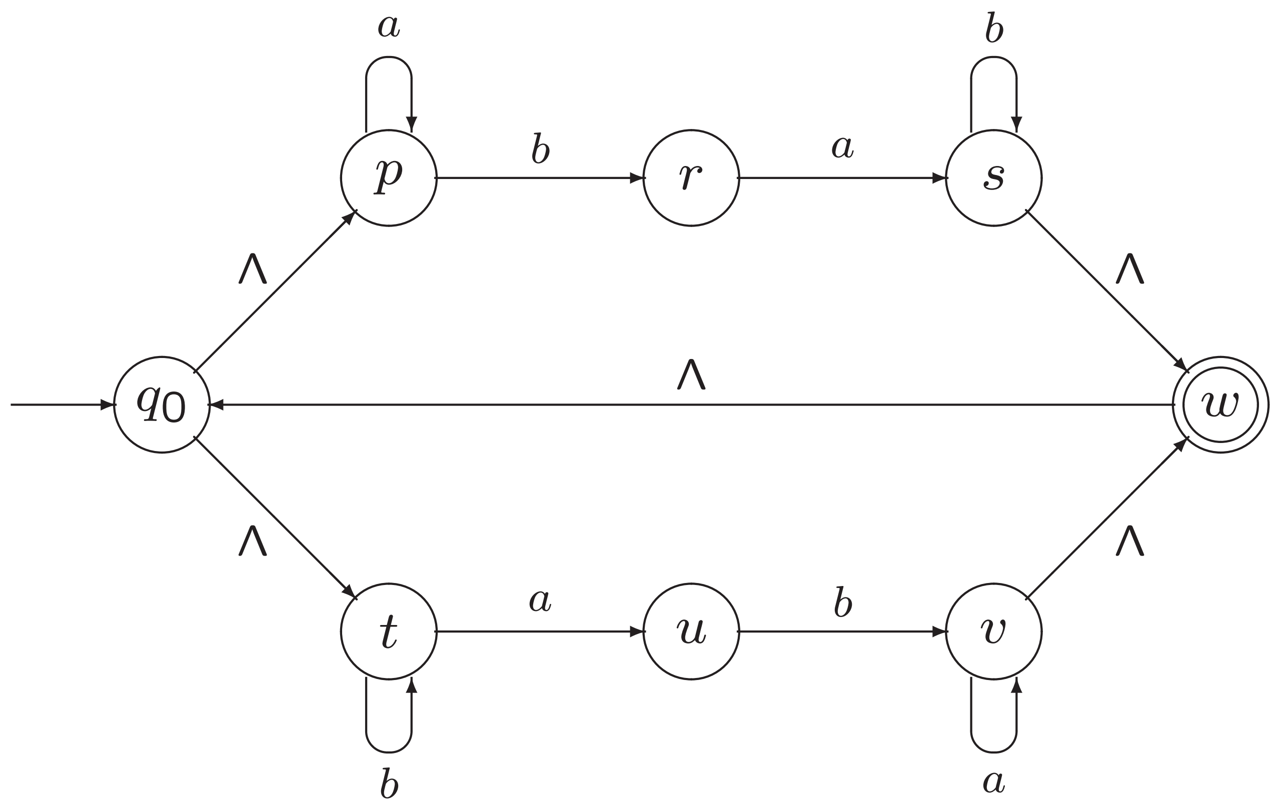
A string $x \in \Sigma^*$ is accepted by M if $\delta^*(q_0, x) \cap A \neq \emptyset$.
The language $L(M)$ accepted by M is the set of all strings accepted by M.

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



$\delta^*(0, \Lambda) = \ldots$
$\delta^*(0, a) = \ldots$
$\delta^*(0, aa) = \ldots$
$\delta^*(0, aab) = \ldots$

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



$\delta^*(q_0, \Lambda) = \ldots$
$\delta^*(q_0, a) = \ldots$
$\delta^*(q_0, ab) = \ldots$
$\delta^*(q_0, aba) = \ldots$

3.3 The Nondeterminism in an NFA Can Be Eliminated

$$\begin{tikzcd} \text{FA} \arrow[rrr, bend right] & \text{NFA without } \Lambda \arrow[l] & \text{NFA} \arrow[l] & \text{reg. expression} \arrow[l] \end{tikzcd}$$

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.

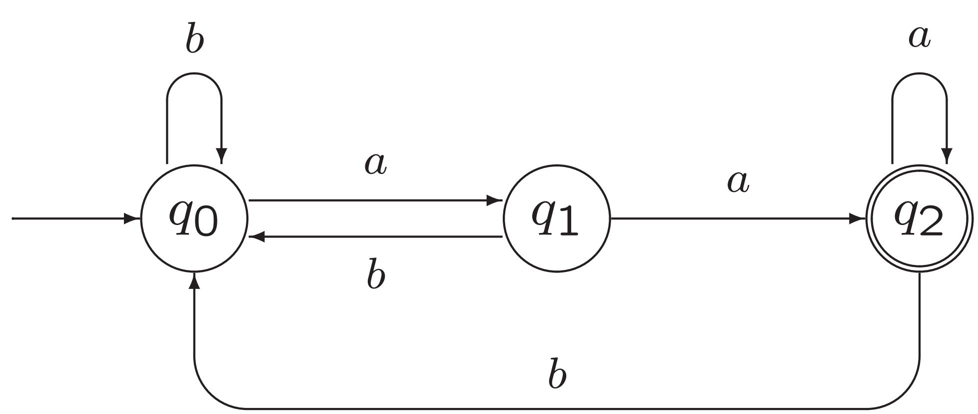
Proof...

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 Λ-transitions that also accepts L.

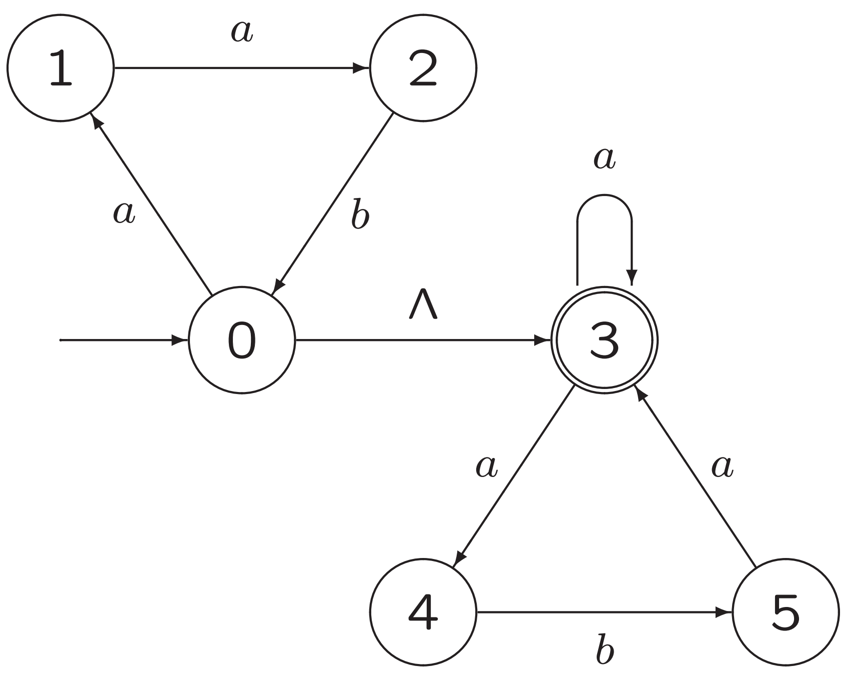
In other words: Λ-transitions in an NFA can be avoided

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 Λ-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.

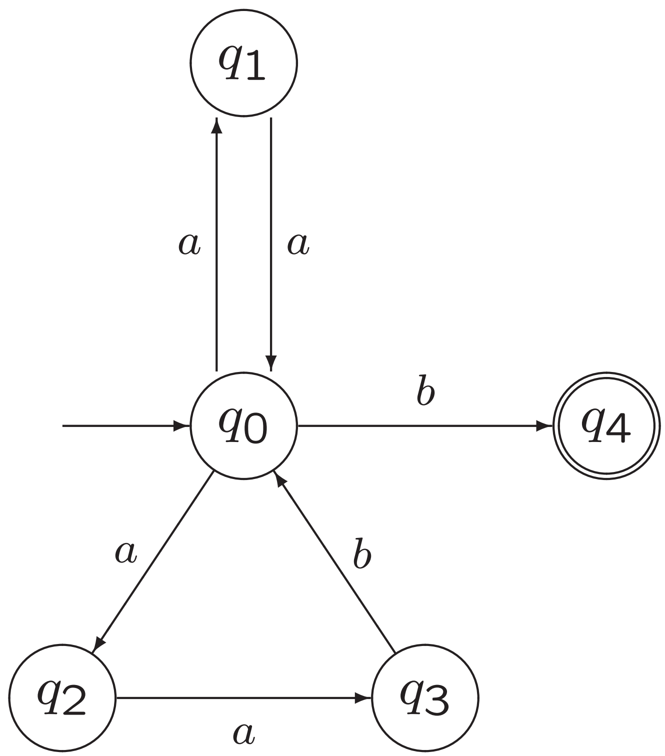
Corollary.
FAs are equally 'strong' as NFAs.

Example 3.19. Eliminating Λ-transitions from an NFA

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

Using the Subset Construction to Eliminate Nondeterminism

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 Λ-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 Σ, every regular language over Σ 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.

$$\begin{tikzcd} \text{FA} \arrow[rrr, bend right] & \text{NFA without } \Lambda \arrow[l] & \text{NFA} \arrow[l] & \text{reg. expression} \arrow[l] \end{tikzcd}$$

A slide from lecture 4:

Definition 3.1. Regular Languages over an Alphabet Σ.

If Σ is an alphabet,
the set $\mathcal{R}$ of regular languages over Σ 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 $\mathcal{R}$.

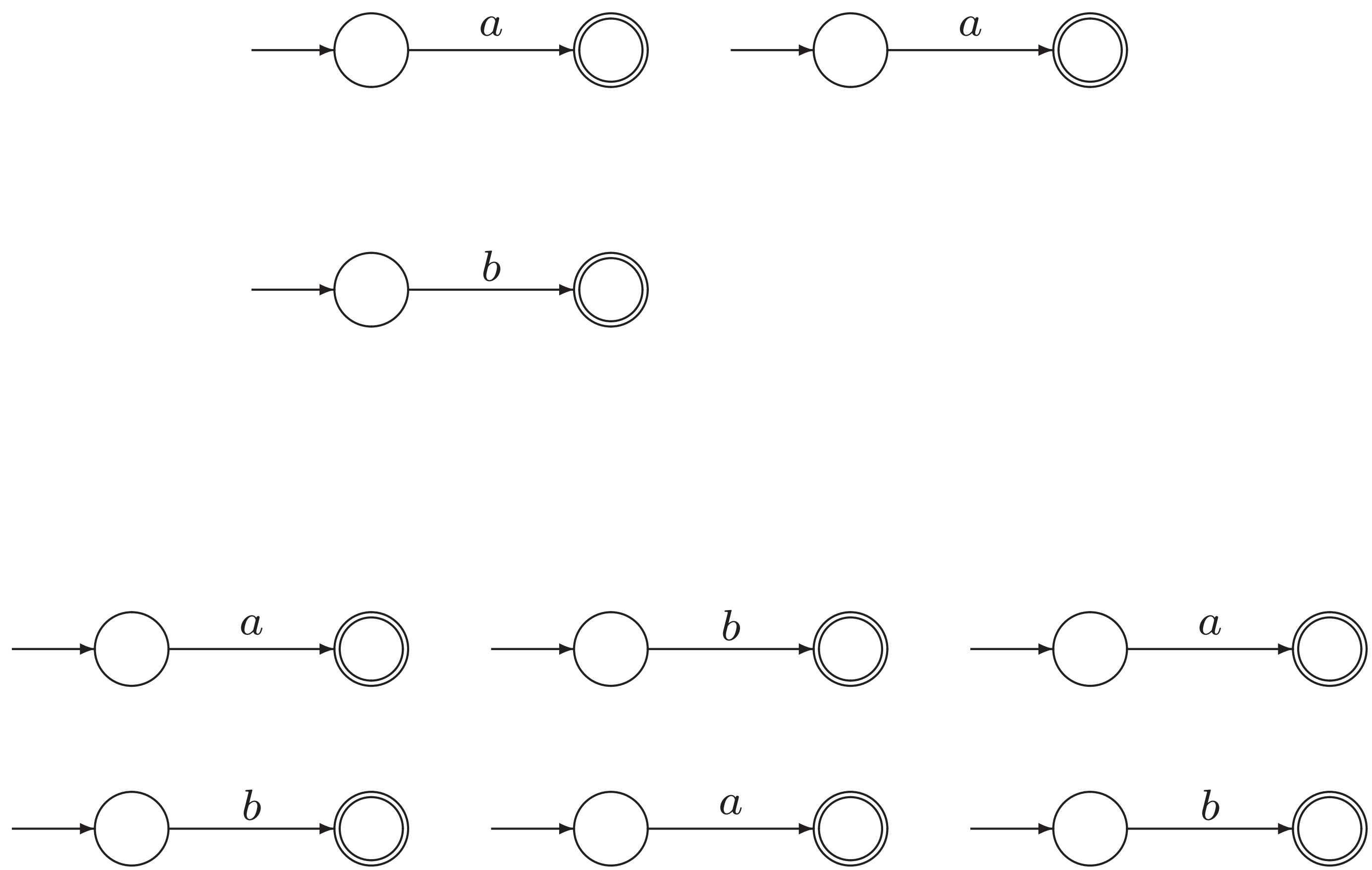
2. For any two languages L_1 and L_2 in $\mathcal{R}$,
   the three languages
   $L_1 \cup L_2$, $L_1 L_2$, and L_1^*
   are elements of $\mathcal{R}$.

(and nothing more)

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

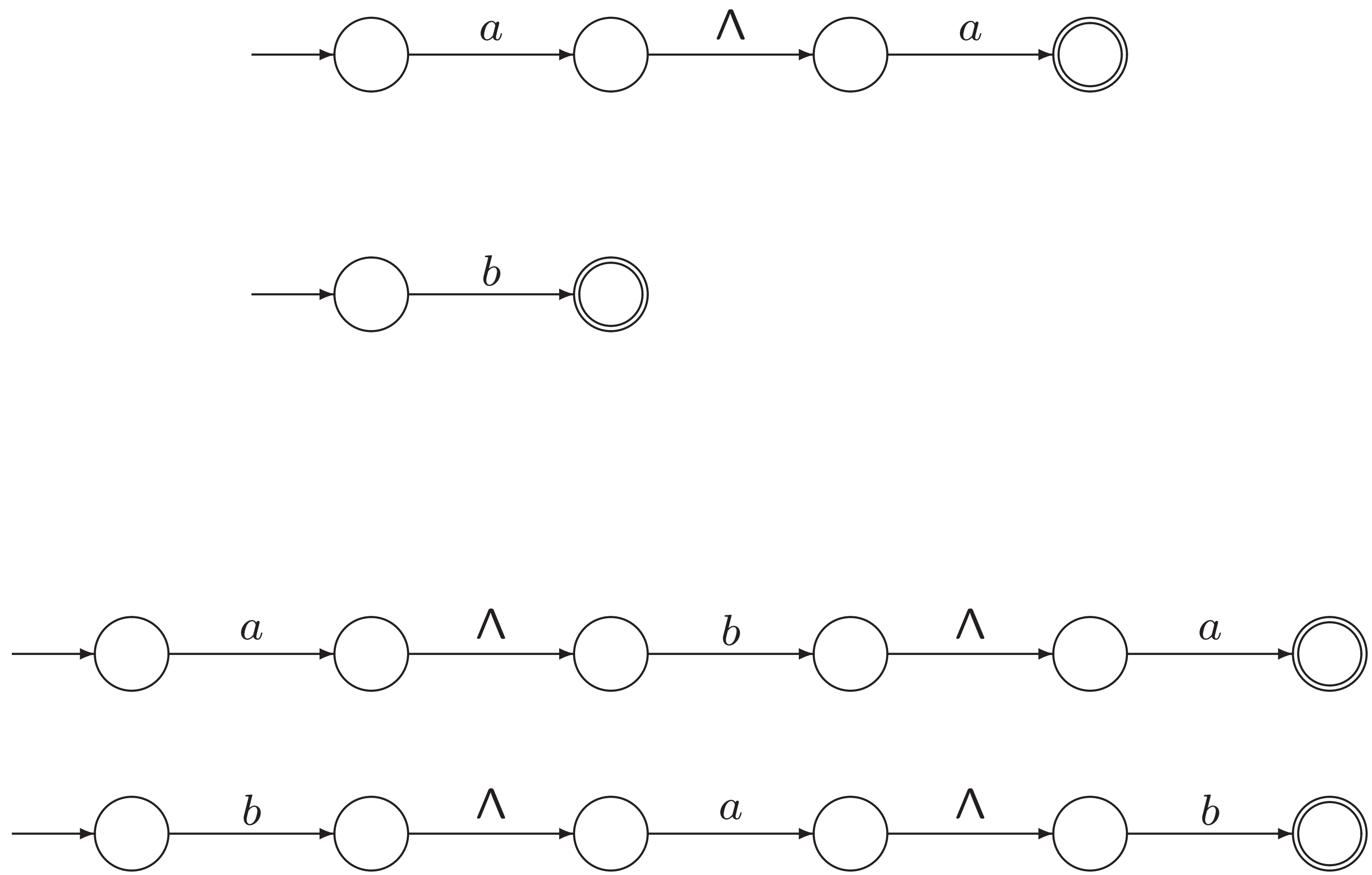
Example 3.28. An NFA Corresponding to $((aa+b)^*(aba)^*bab)^*$

Step 1



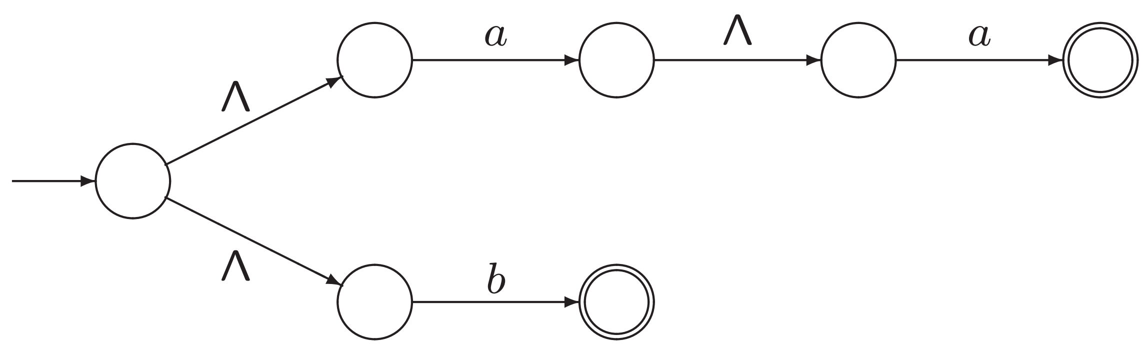
Example 3.28. An NFA Corresponding to $((aa+b)^*(aba)^*bab)^*$

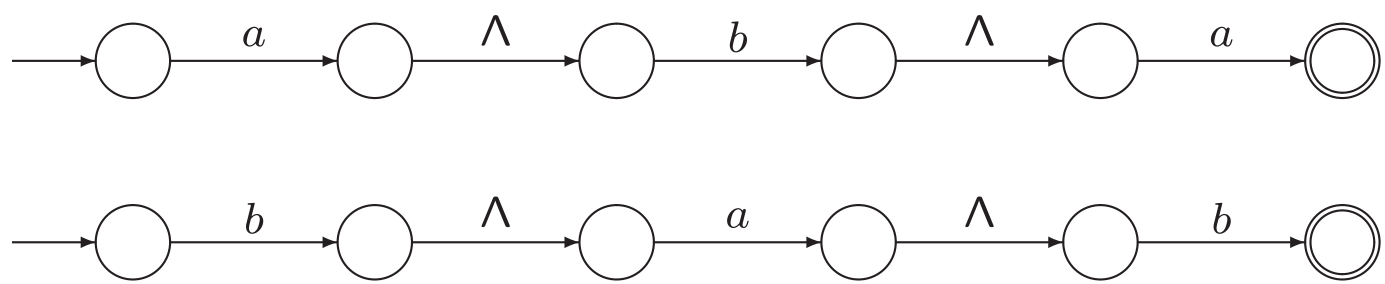
Step 2



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

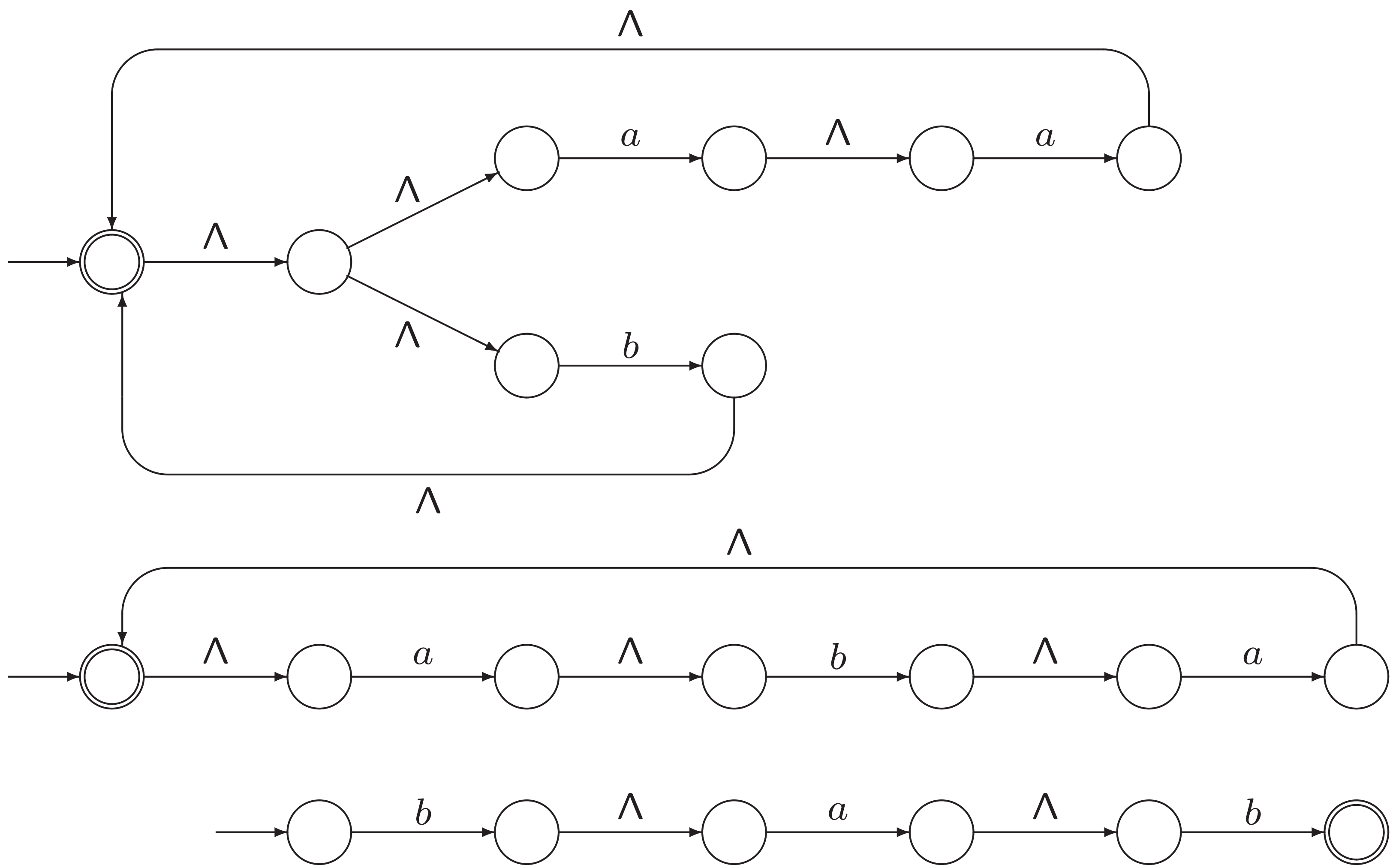
Step 3





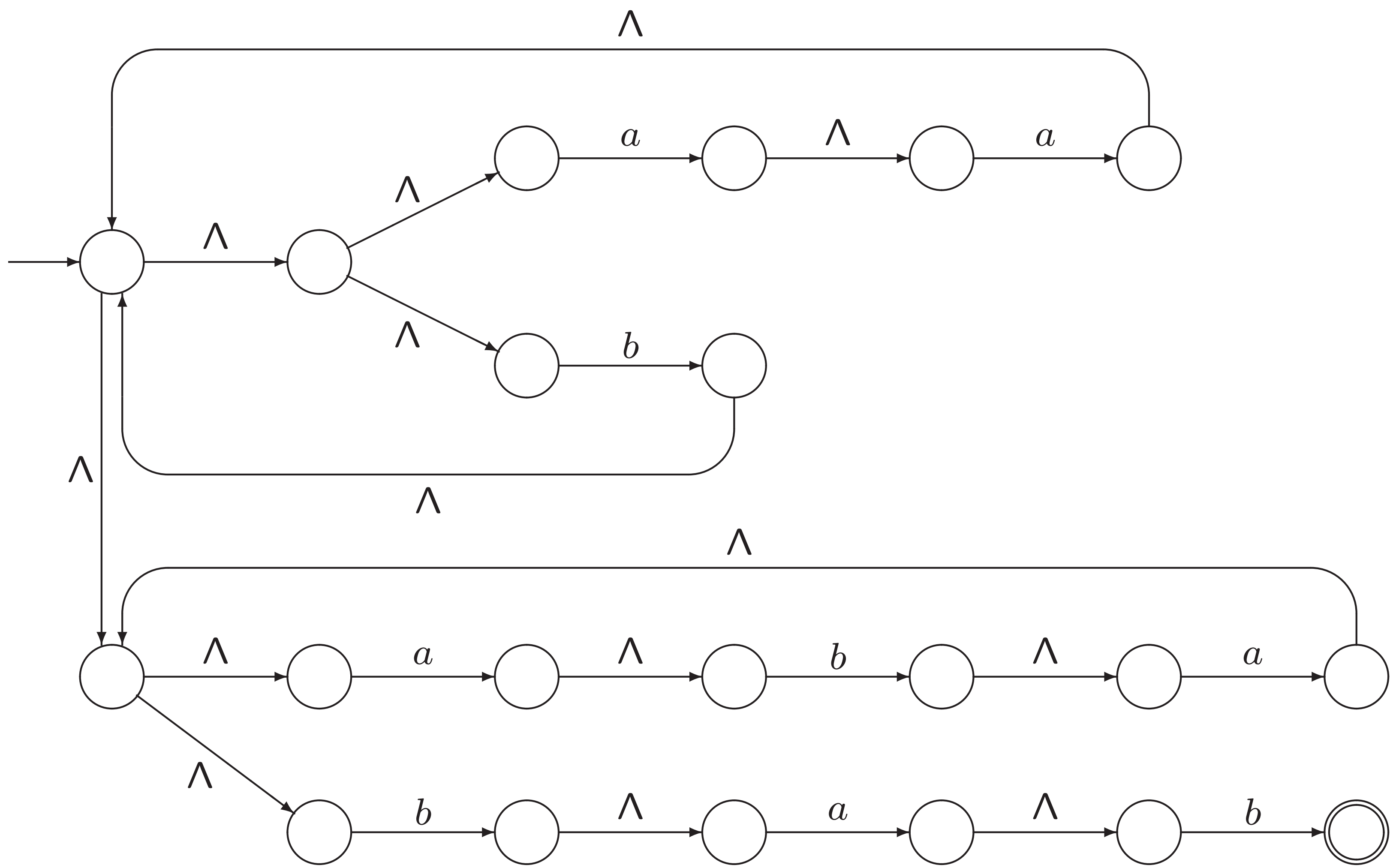
Example 3.28. An NFA Corresponding to $((aa+b)^*(aba)^*bab)^*$

Step 4



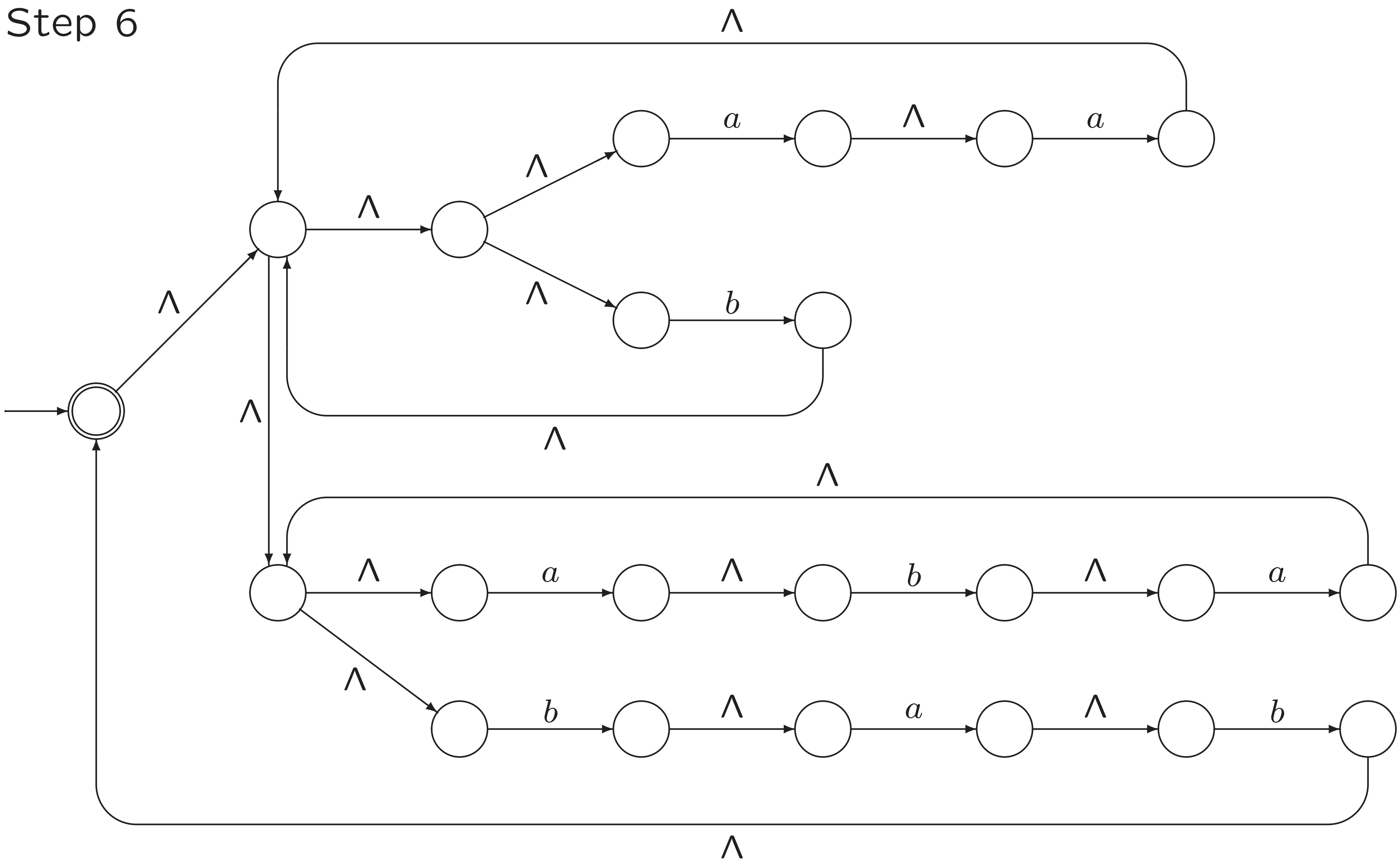
Example 3.28. An NFA Corresponding to $((aa+b)^*(aba)^*bab)^*$

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.

$$\begin{tikzcd} \text{FA} \arrow[rrr, bend right] & \text{NFA without } \Lambda \arrow[l] & \text{NFA} \arrow[l] & \text{reg. expression} \arrow[l] \end{tikzcd}$$