

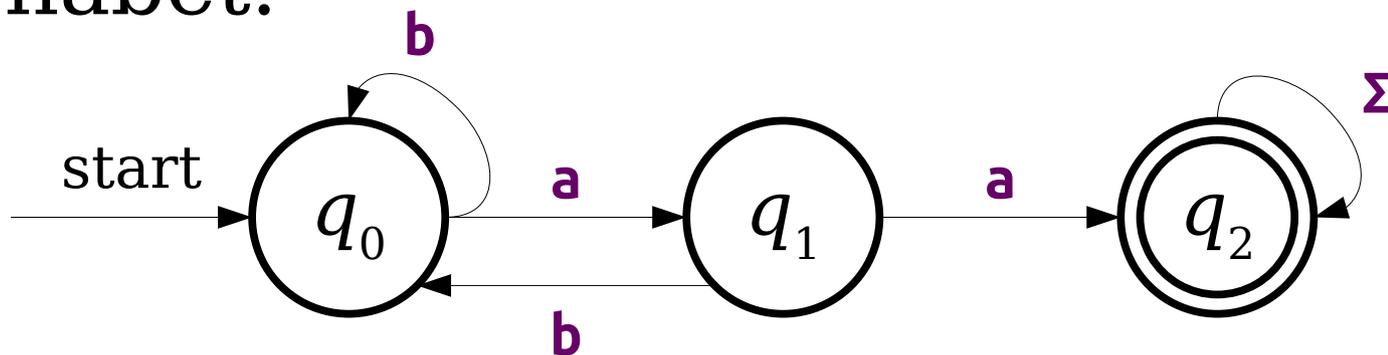
# Finite Automata

## Part Three

Recap from Last Time

# DFA's

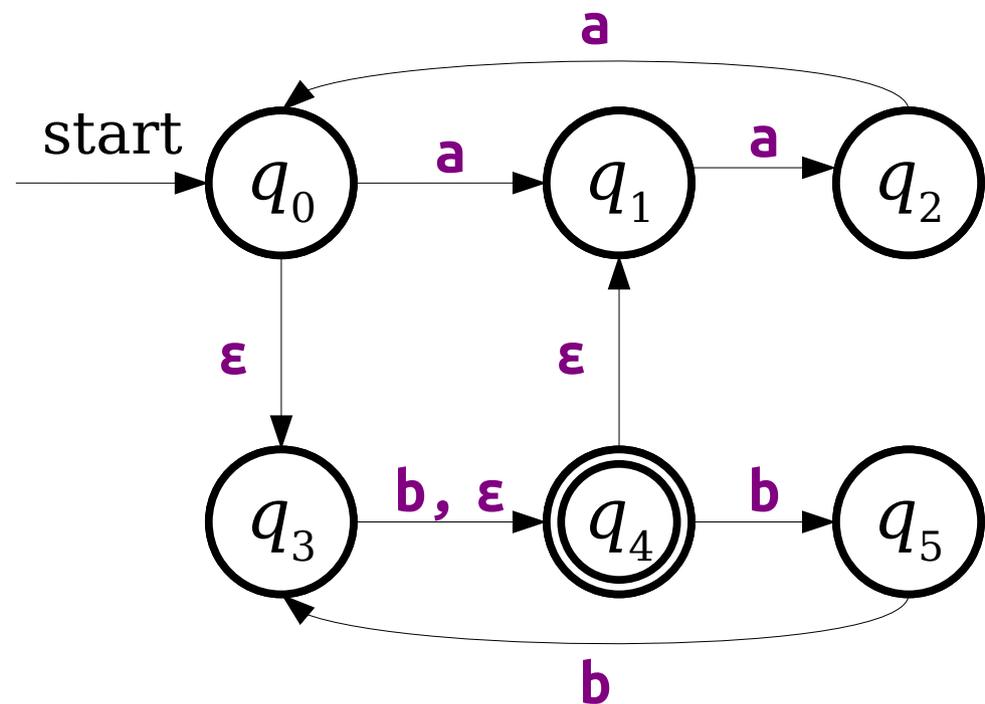
- A **DFA** is a
  - **D**eterministic
  - **F**inite
  - **A**utomaton
- Every state must have exactly one transition defined for each symbol in the alphabet.



If  $D$  is a DFA, the ***language of  $D$*** , denoted  $\mathcal{L}(D)$ , is  $\{ w \in \Sigma^* \mid D \text{ accepts } w \}$ .

# NFAs

- An **NFA** is a
  - **N**ondeterministic
  - **F**inite
  - **A**utomaton
- NFAs have no restrictions on how many transitions are allowed per state.
- They can also use  $\epsilon$ -transitions.
- An NFA accepts a string  $w$  if there is some sequence of choices that leads to an accepting state.



# Massive Parallelism

- An NFA can be thought of as a DFA that can be in many states at once.
- At each point in time, when the NFA needs to follow a transition, it tries all the options at the same time.
- The NFA accepts if *any* of the states that are active at the end are accepting states. It rejects otherwise.

New Stuff!

# Designing NFAs

# Designing NFAs

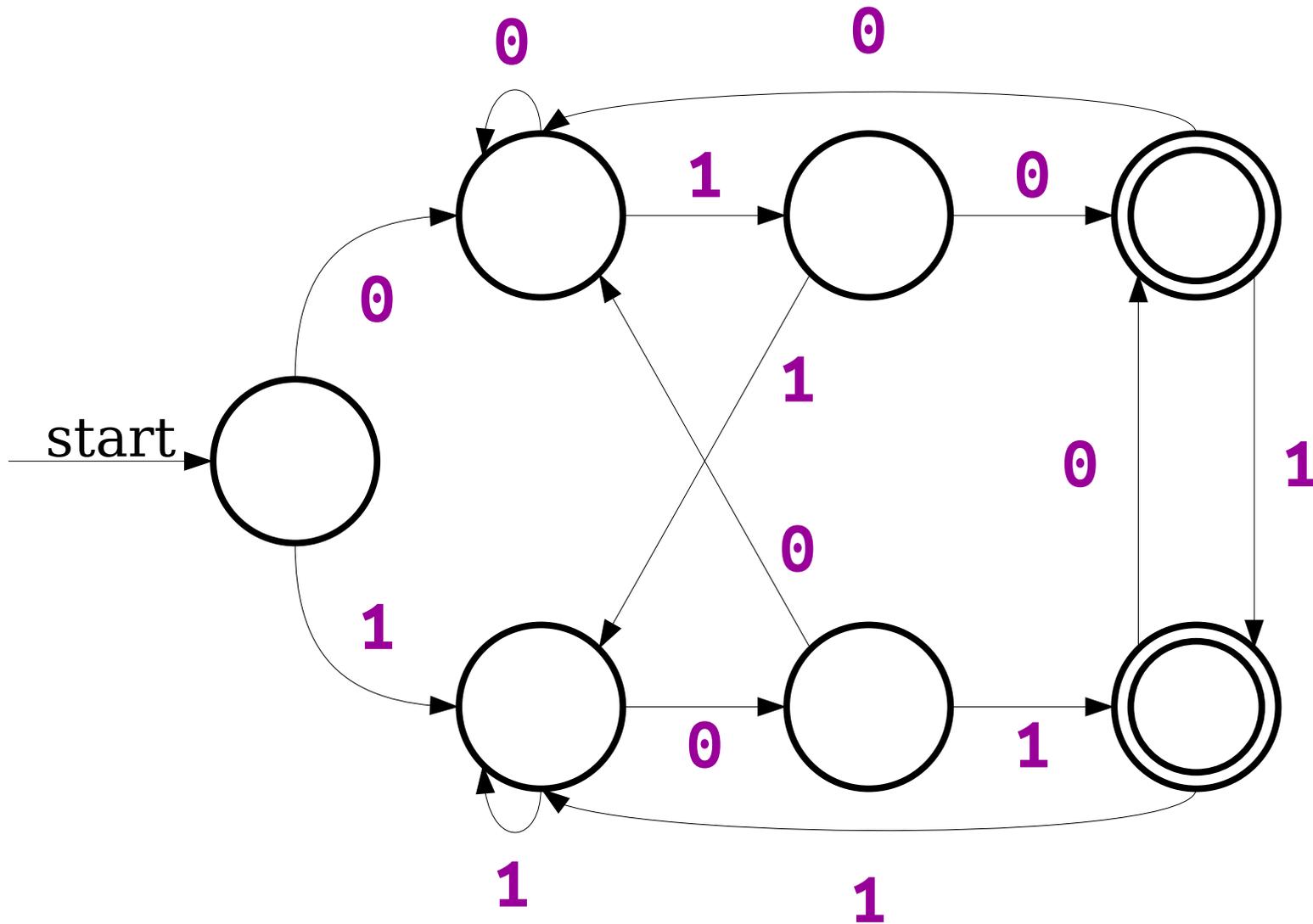
- ***Embrace the nondeterminism!***
- Good model: ***Guess-and-check:***
  - Is there some information that you'd really like to have? Have the machine *nondeterministically guess* that information.
  - Then, have the machine *deterministically check* that the choice was correct.
- The *guess* phase corresponds to trying lots of different options.
- The *check* phase corresponds to filtering out bad guesses or wrong options.

# Guess-and-Check

$$L = \{ w \in \{0, 1\}^* \mid w \text{ ends in } 010 \text{ or } 101 \}$$

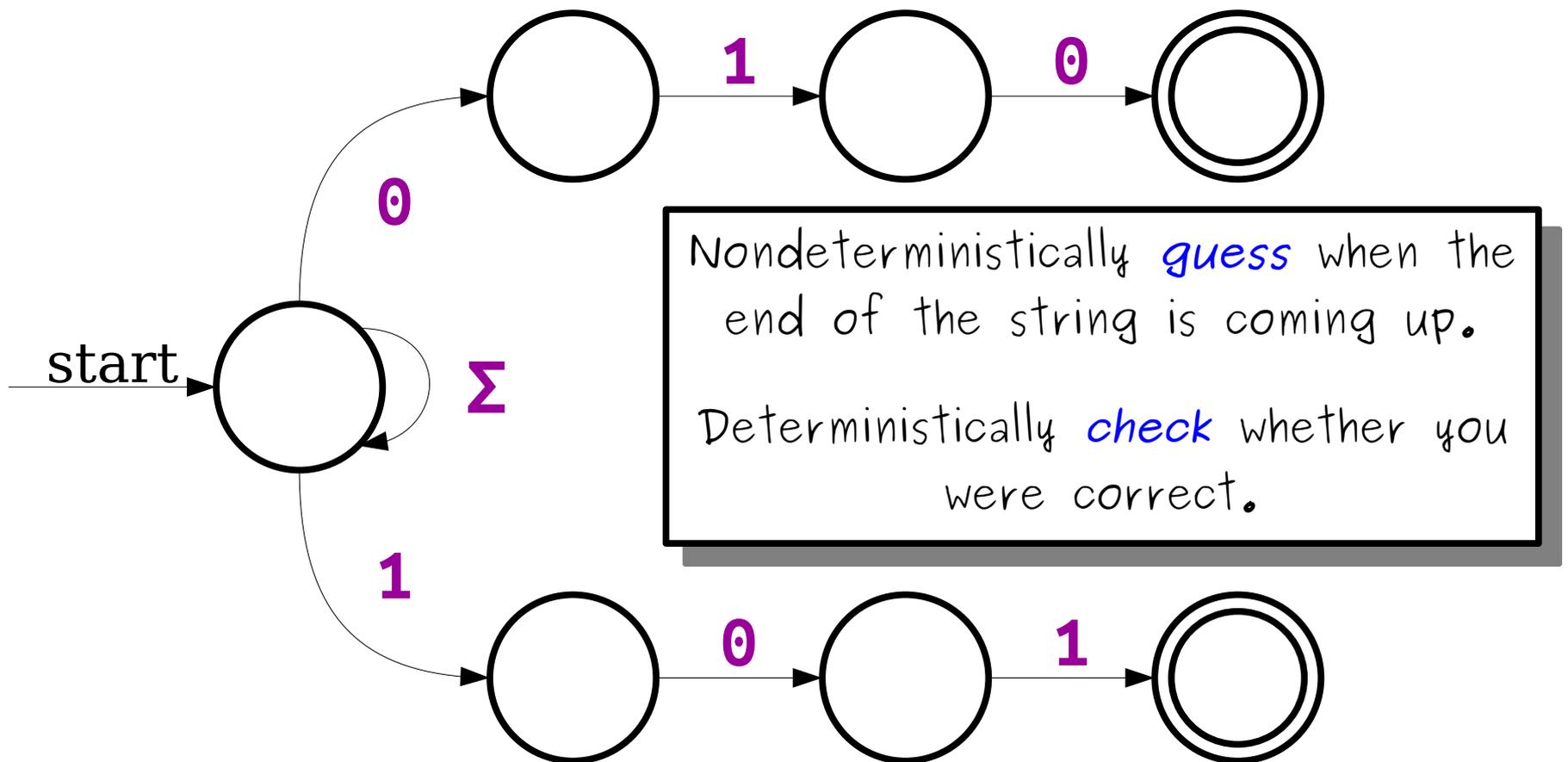
# Guess-and-Check

$$L = \{ w \in \{0, 1\}^* \mid w \text{ ends in } 010 \text{ or } 101 \}$$



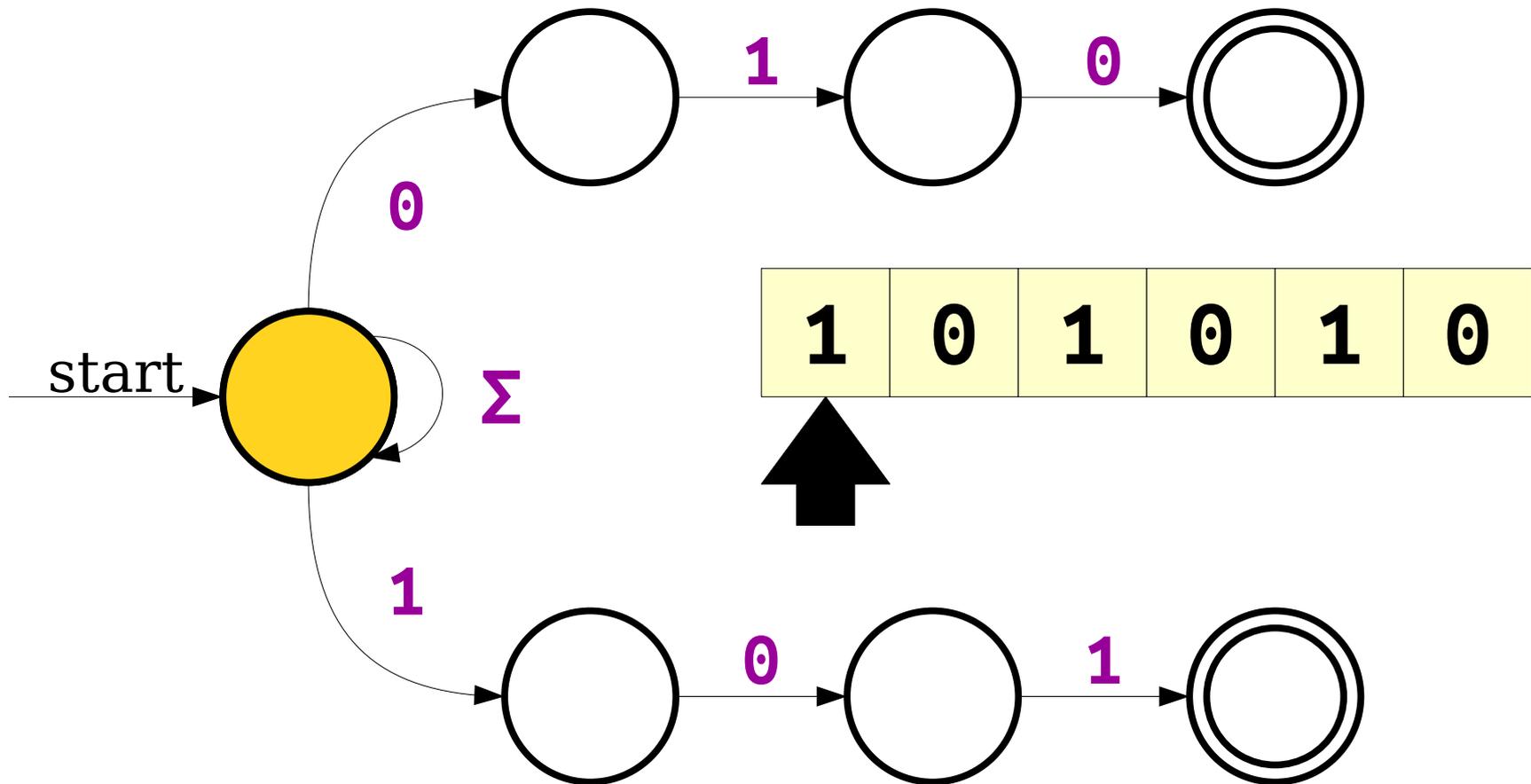
# Guess-and-Check

$$L = \{ w \in \{0, 1\}^* \mid w \text{ ends in } 010 \text{ or } 101 \}$$



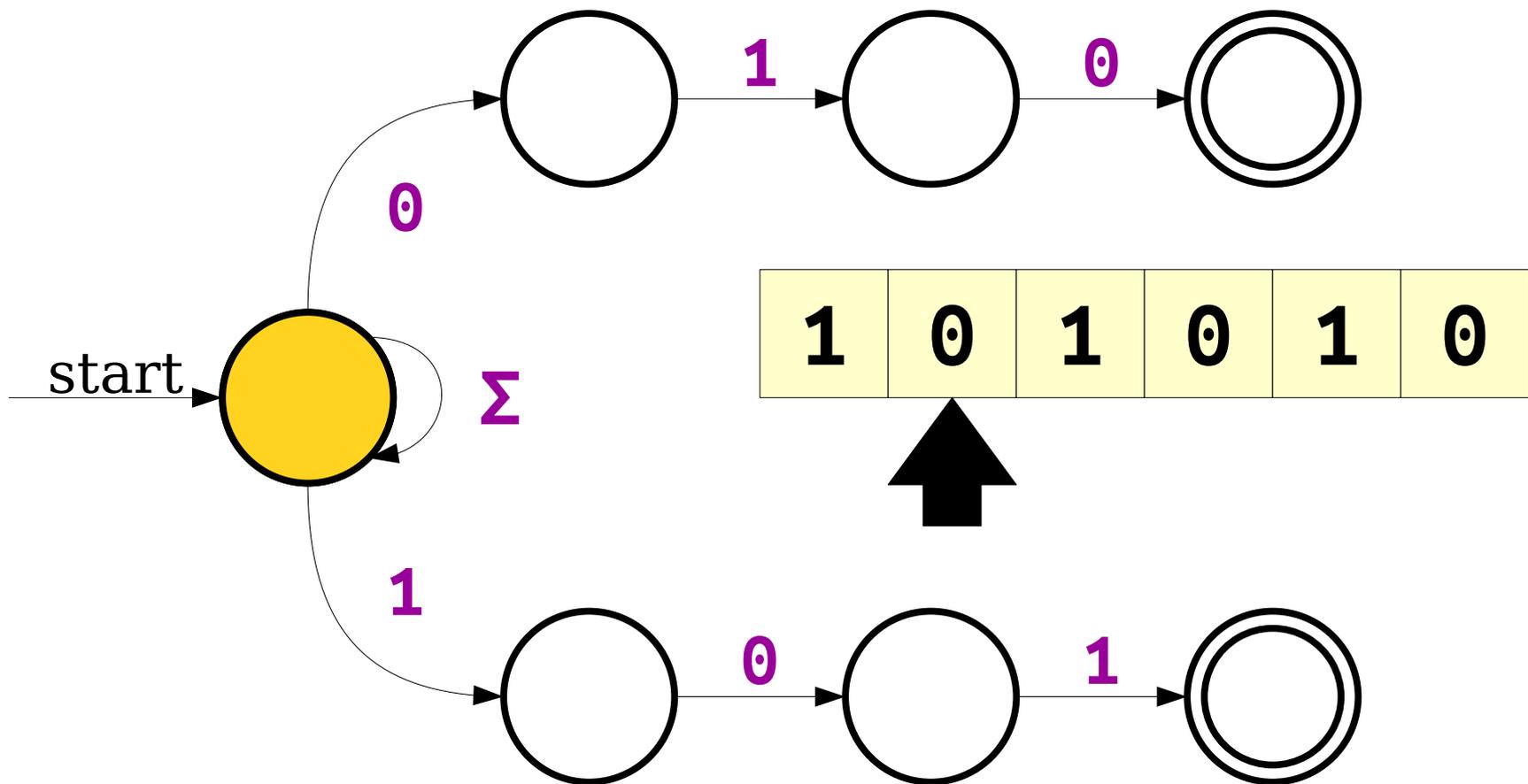
# Guess-and-Check

$$L = \{ w \in \{0, 1\}^* \mid w \text{ ends in } 010 \text{ or } 101 \}$$



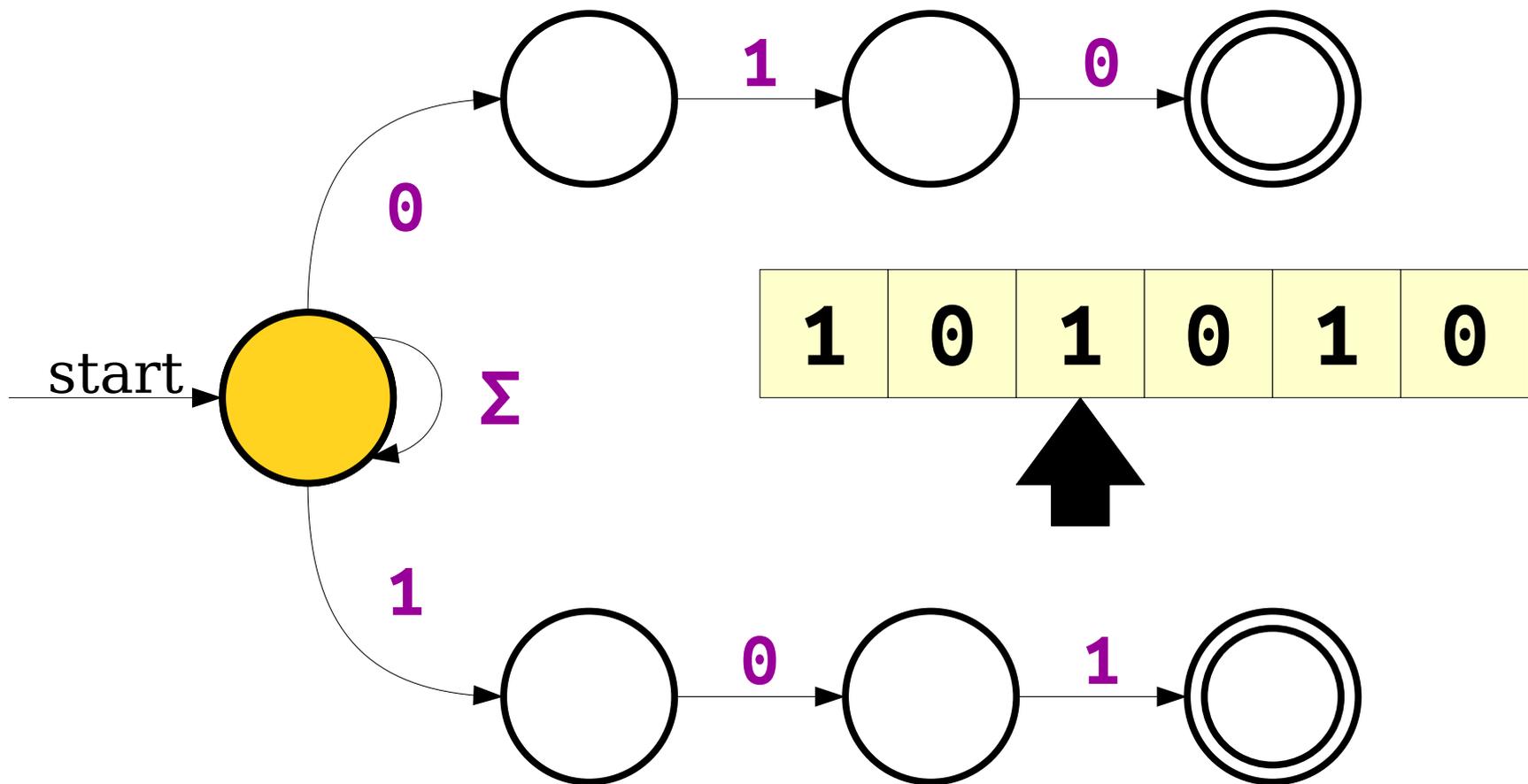
# Guess-and-Check

$$L = \{ w \in \{0, 1\}^* \mid w \text{ ends in } 010 \text{ or } 101 \}$$



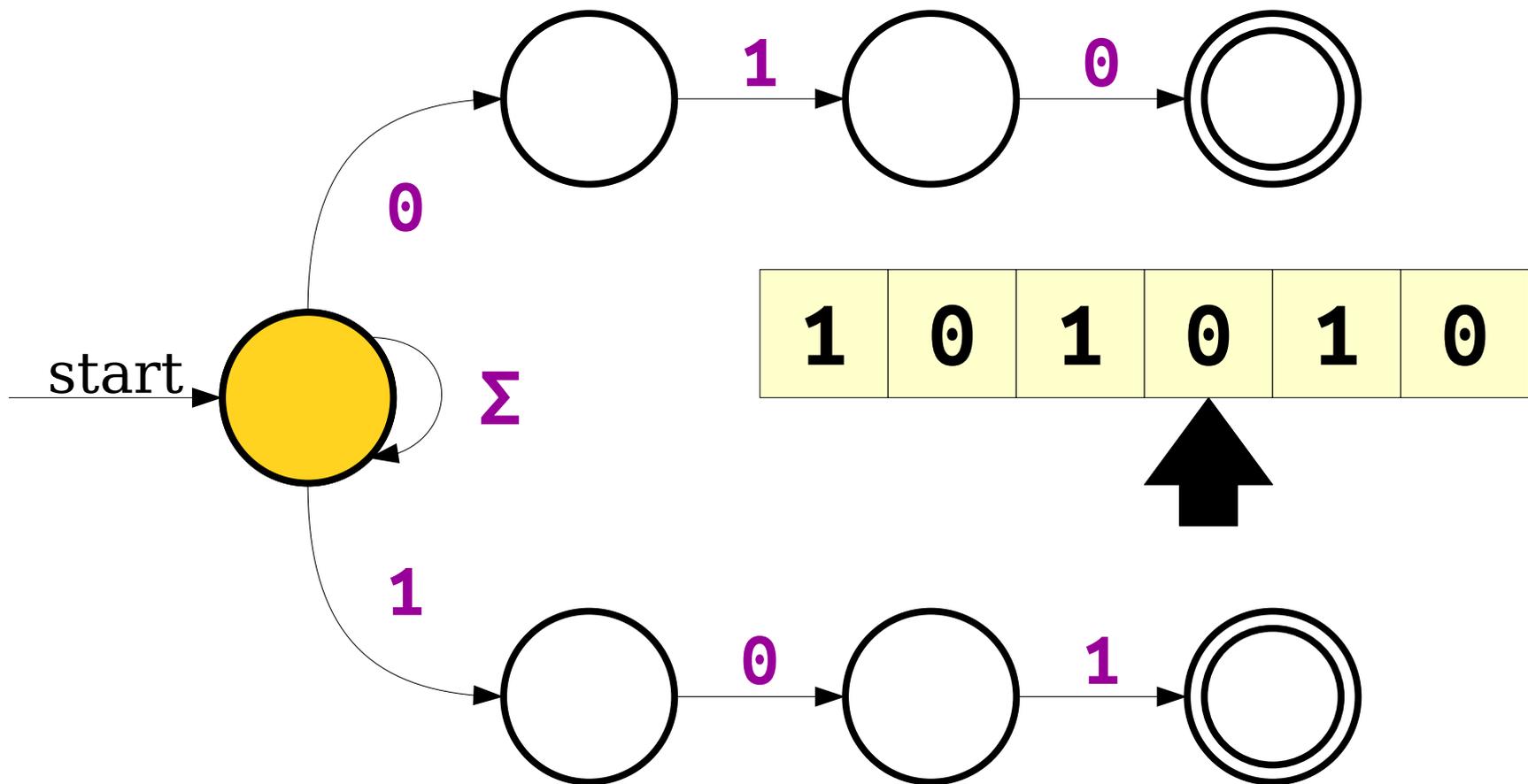
# Guess-and-Check

$$L = \{ w \in \{0, 1\}^* \mid w \text{ ends in } 010 \text{ or } 101 \}$$



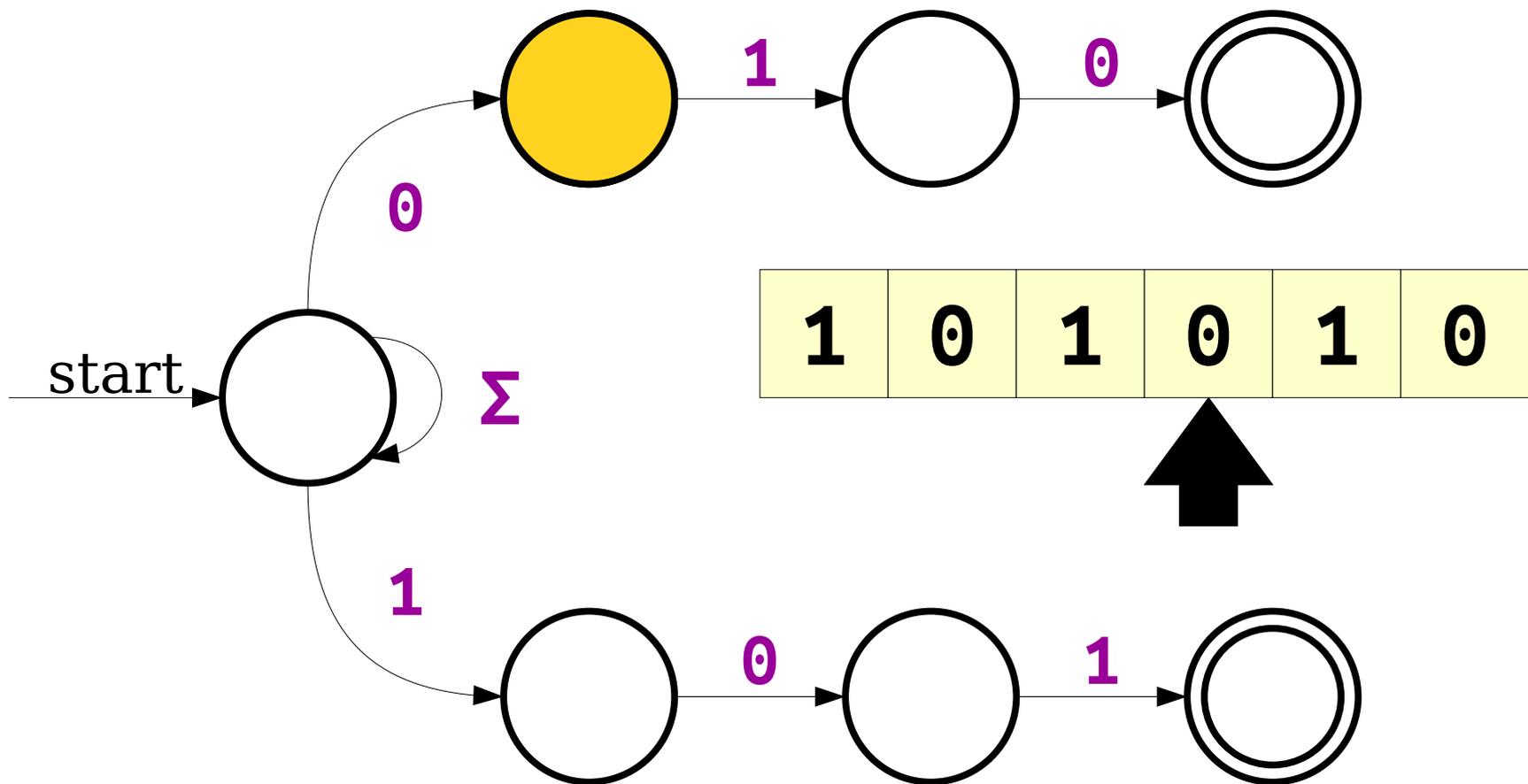
# Guess-and-Check

$$L = \{ w \in \{0, 1\}^* \mid w \text{ ends in } 010 \text{ or } 101 \}$$



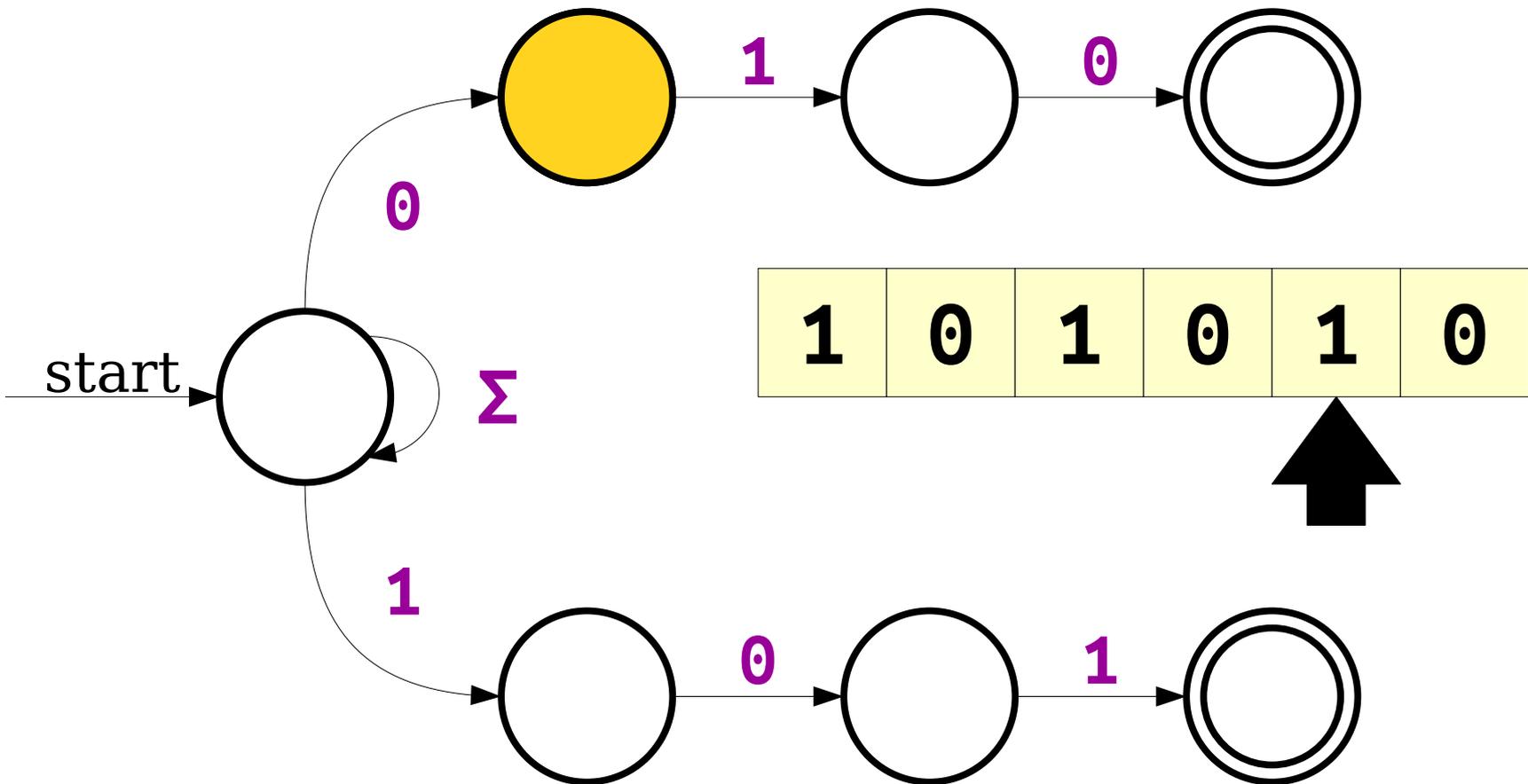
# Guess-and-Check

$$L = \{ w \in \{0, 1\}^* \mid w \text{ ends in } 010 \text{ or } 101 \}$$



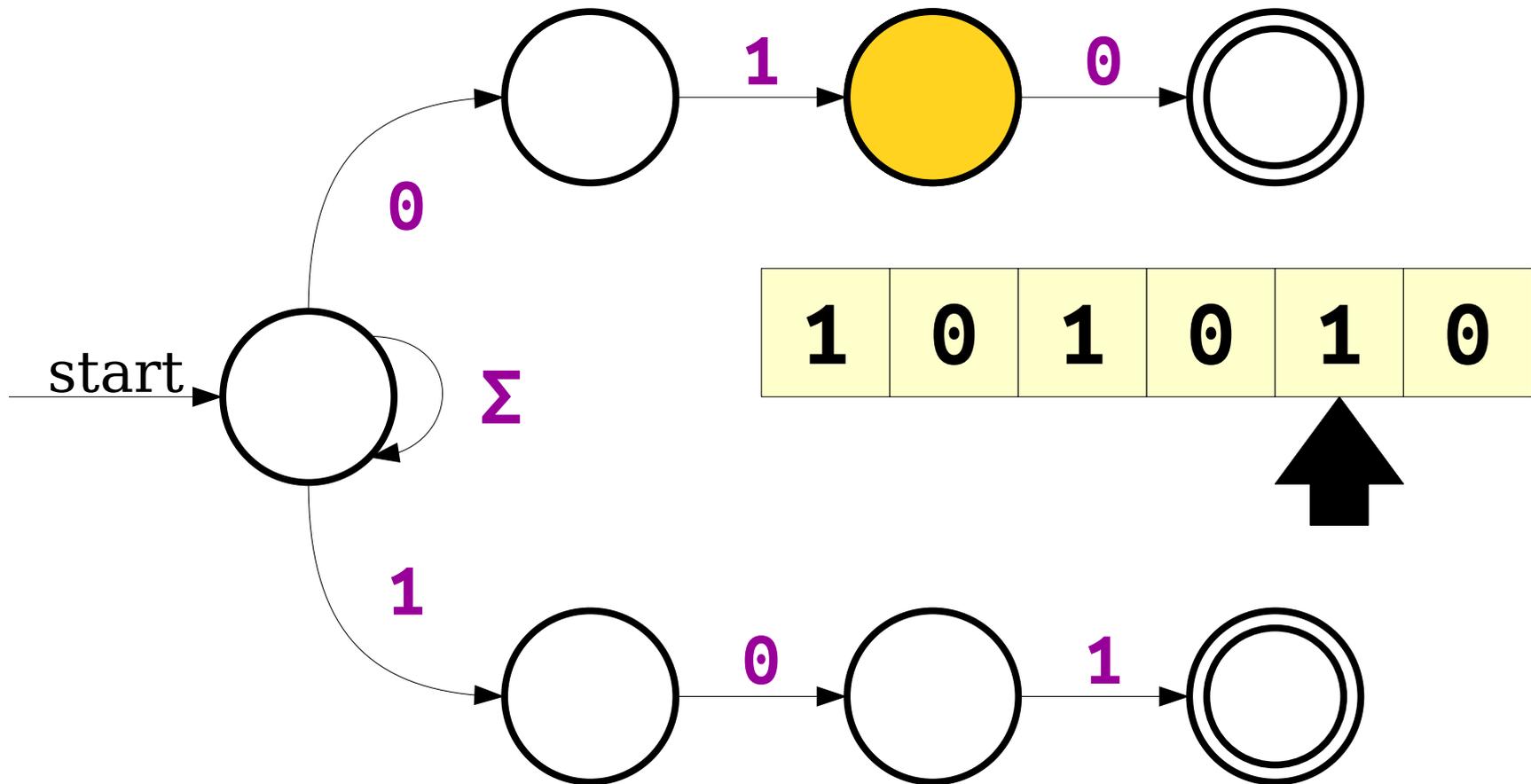
# Guess-and-Check

$$L = \{ w \in \{0, 1\}^* \mid w \text{ ends in } 010 \text{ or } 101 \}$$



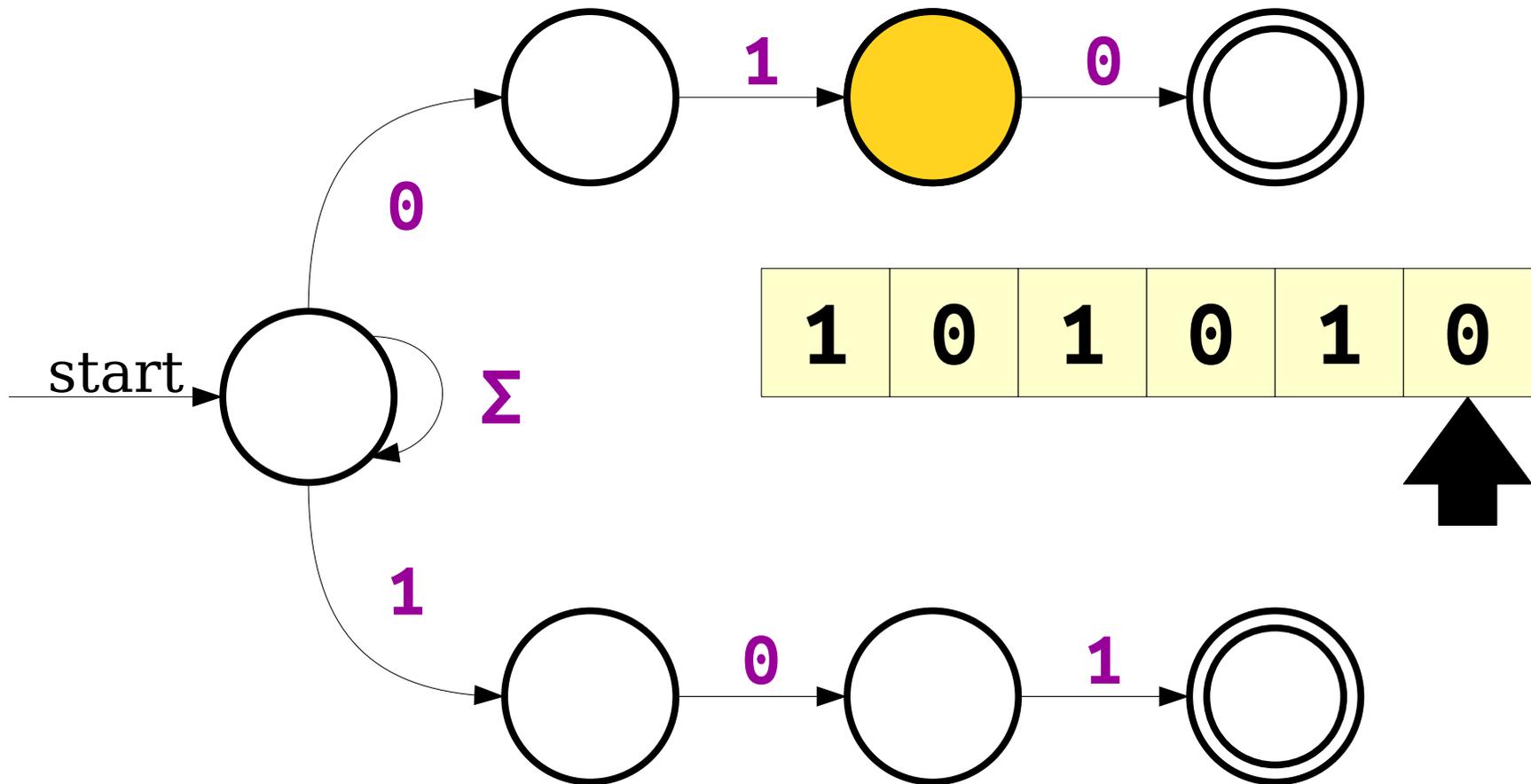
# Guess-and-Check

$$L = \{ w \in \{0, 1\}^* \mid w \text{ ends in } 010 \text{ or } 101 \}$$



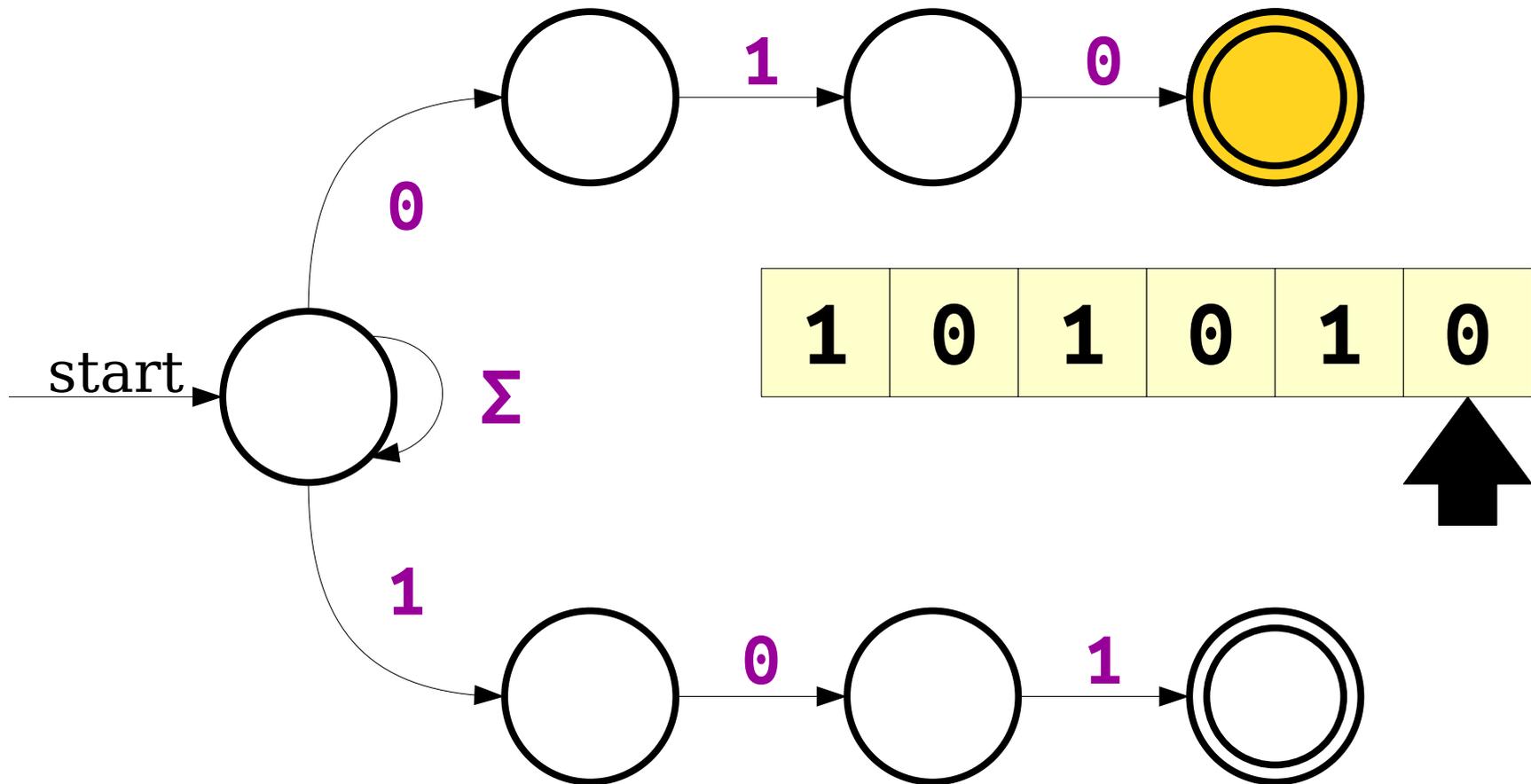
# Guess-and-Check

$$L = \{ w \in \{0, 1\}^* \mid w \text{ ends in } 010 \text{ or } 101 \}$$



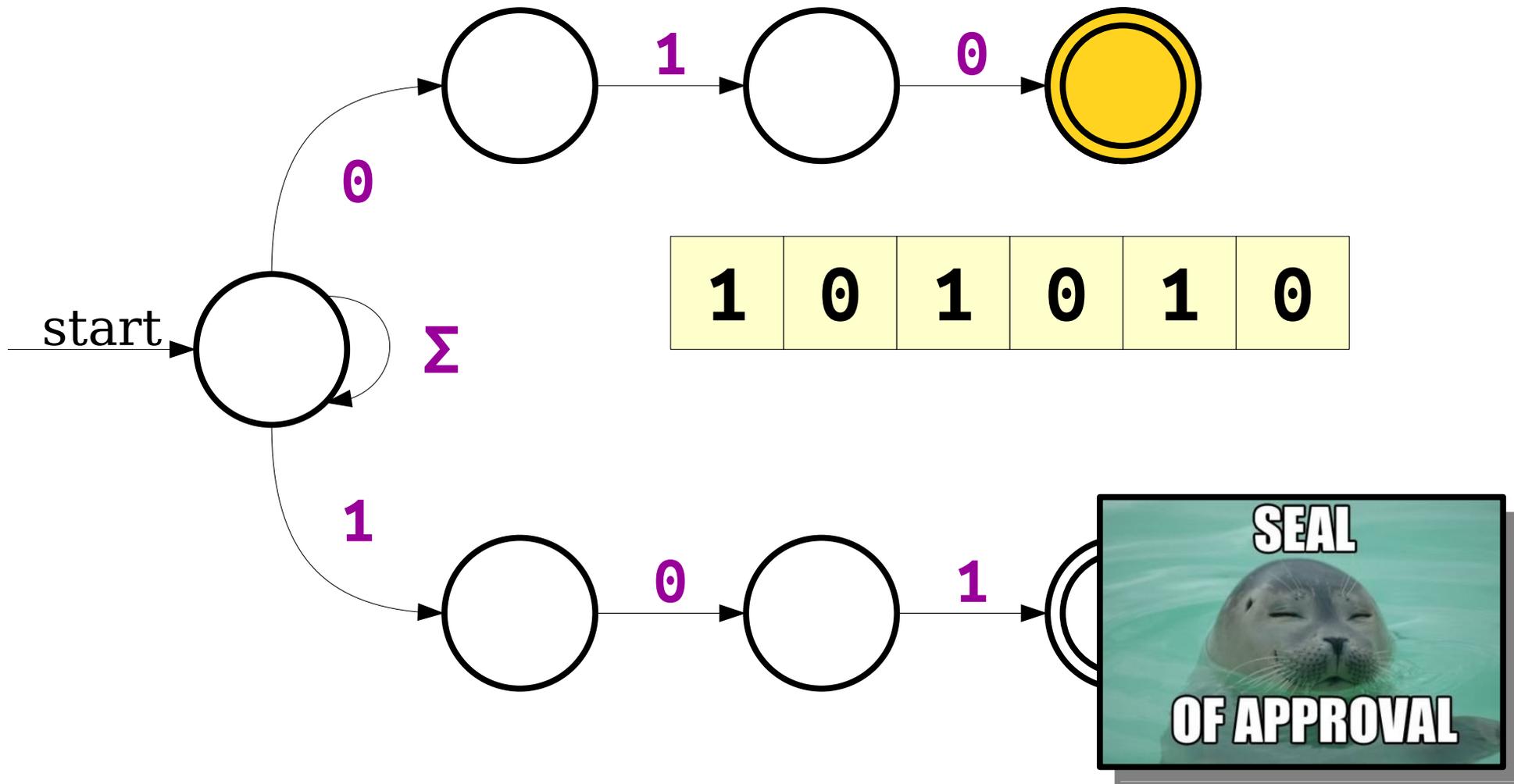
# Guess-and-Check

$$L = \{ w \in \{0, 1\}^* \mid w \text{ ends in } 010 \text{ or } 101 \}$$



# Guess-and-Check

$$L = \{ w \in \{0, 1\}^* \mid w \text{ ends in } 010 \text{ or } 101 \}$$



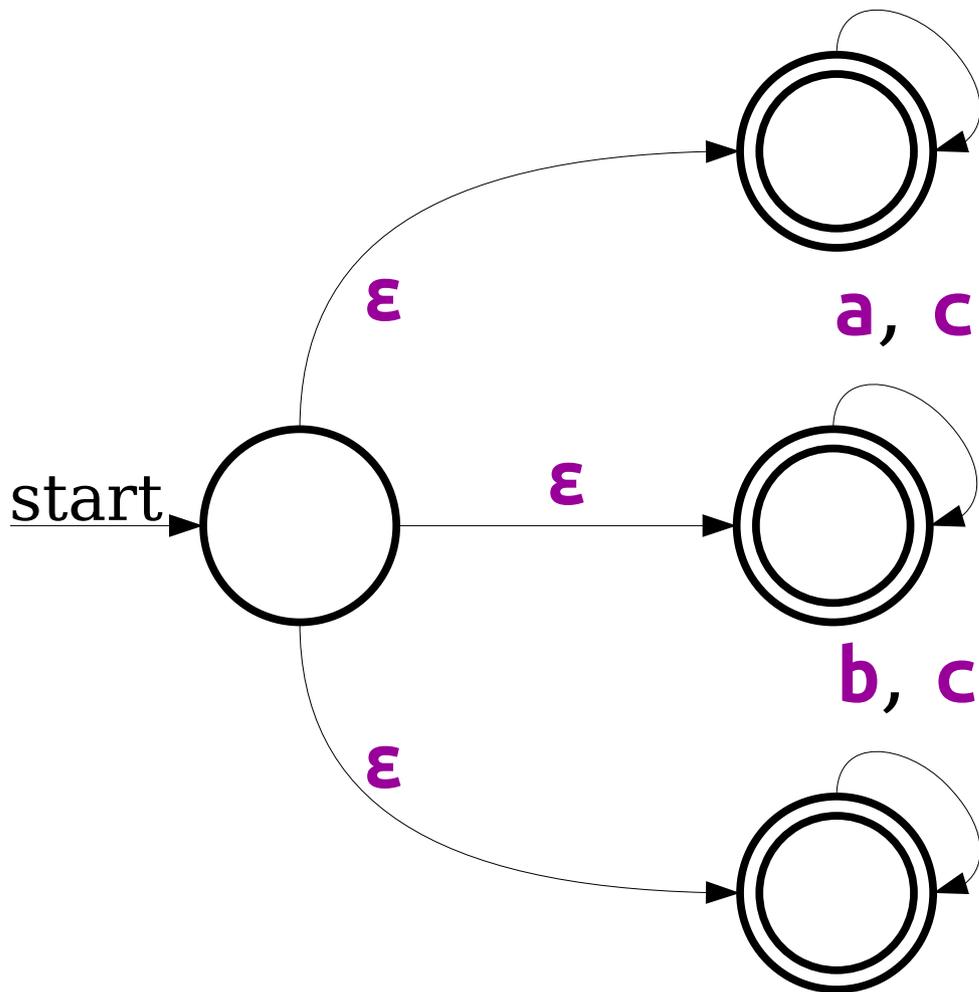
# Guess-and-Check

$L = \{ w \in \{a, b, c\}^* \mid \text{at least one of } a, b, \text{ or } c \text{ is not in } w \}$



# Guess-and-Check

$L = \{ w \in \{a, b, c\}^* \mid \text{at least one of } a, b, \text{ or } c \text{ is not in } w \}$

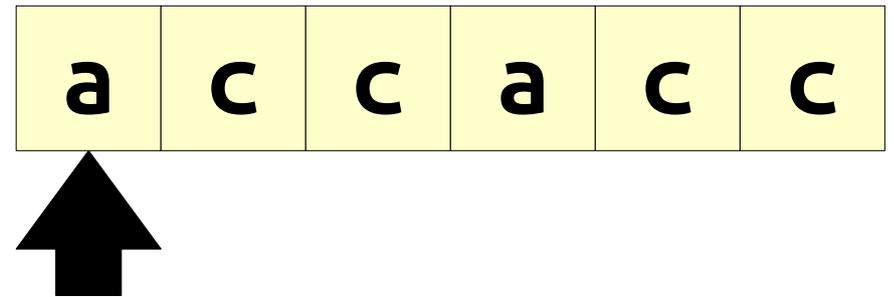
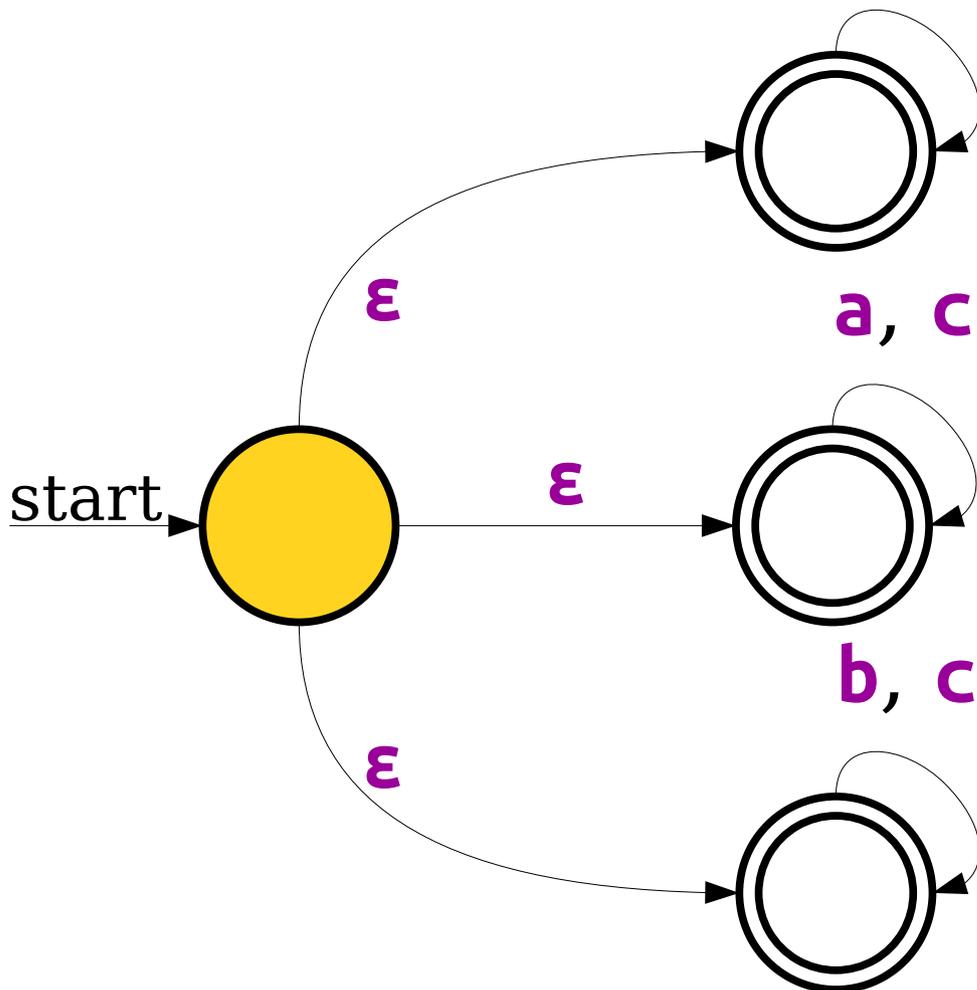


Nondeterministically *guess* which character is missing.

Deterministically *check* whether that character is indeed missing.

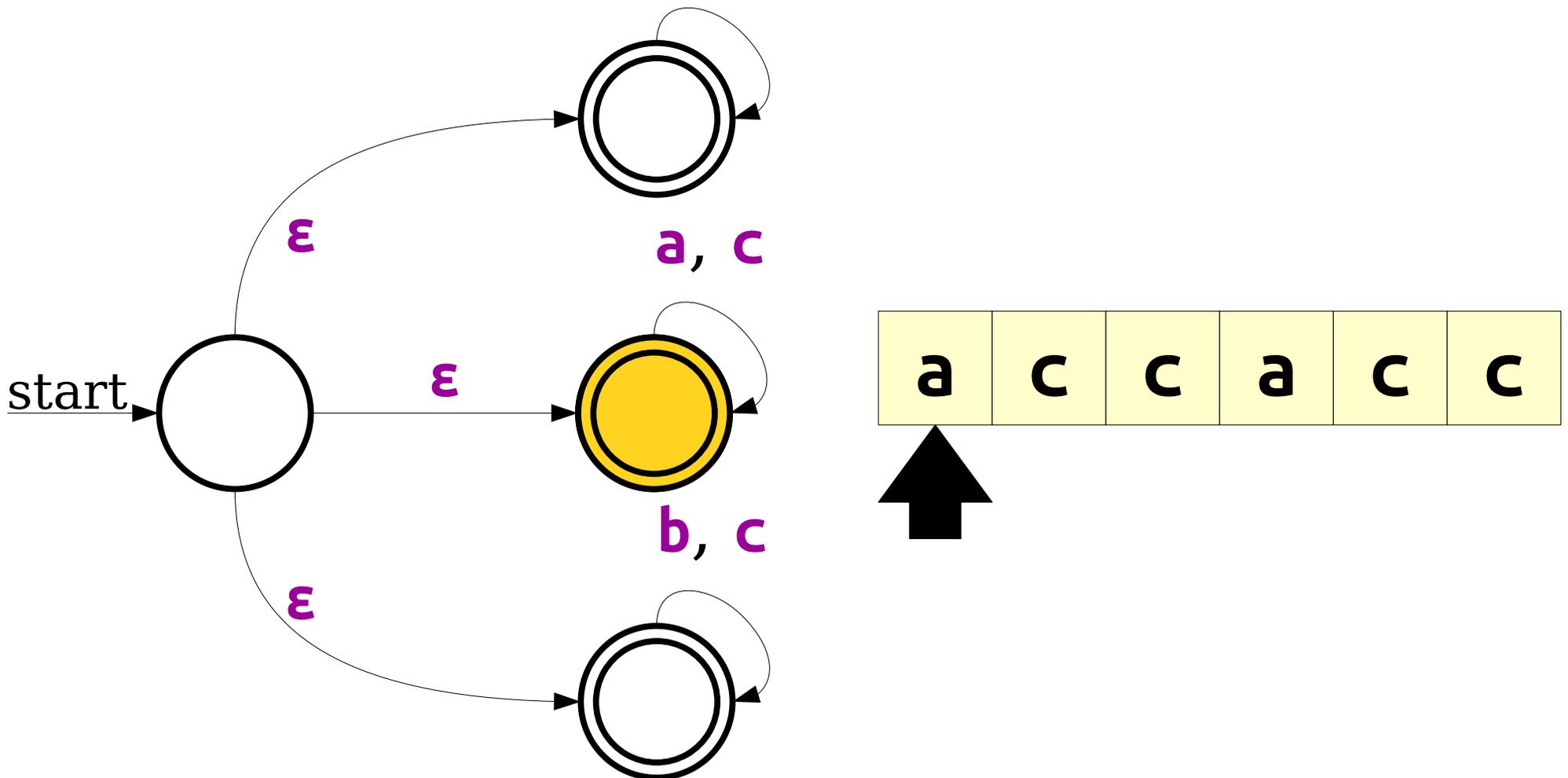
# Guess-and-Check

$L = \{ w \in \{a, b, c\}^* \mid \text{at least one of } a, b, \text{ or } c \text{ is not in } w \}$



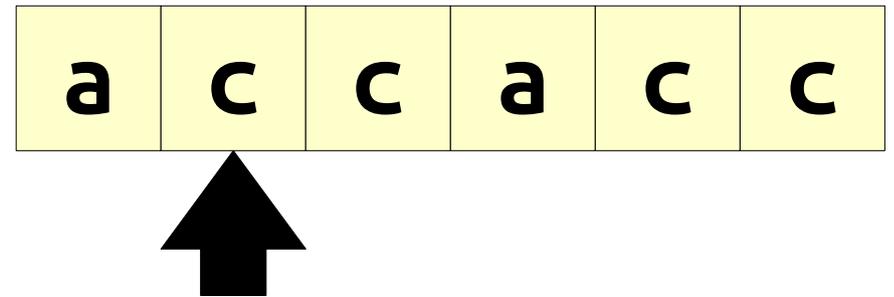
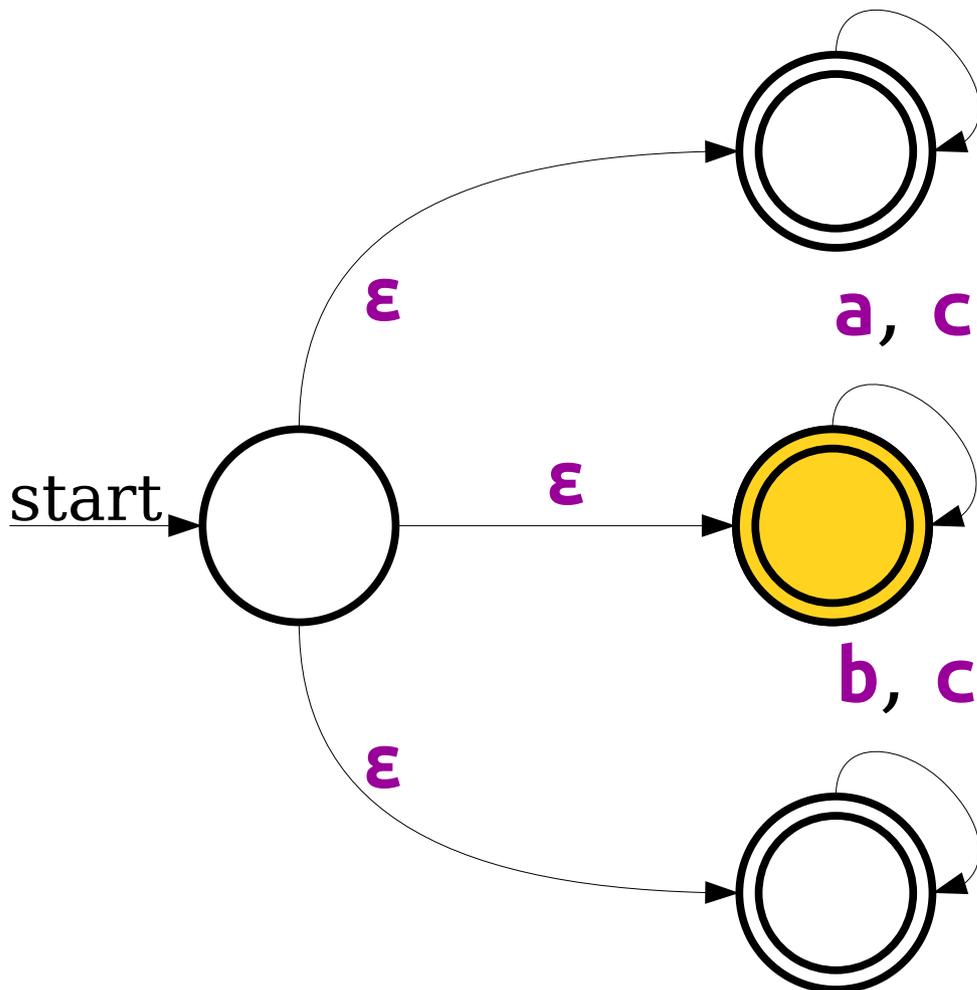
# Guess-and-Check

$L = \{ w \in \{a, b, c\}^* \mid \text{at least one of } a, b, \text{ or } c \text{ is not in } w \}$



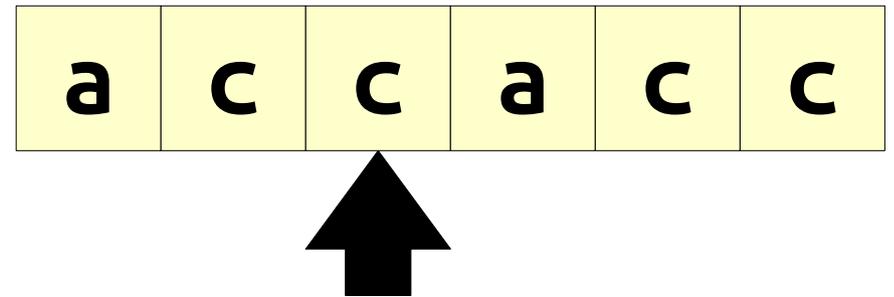
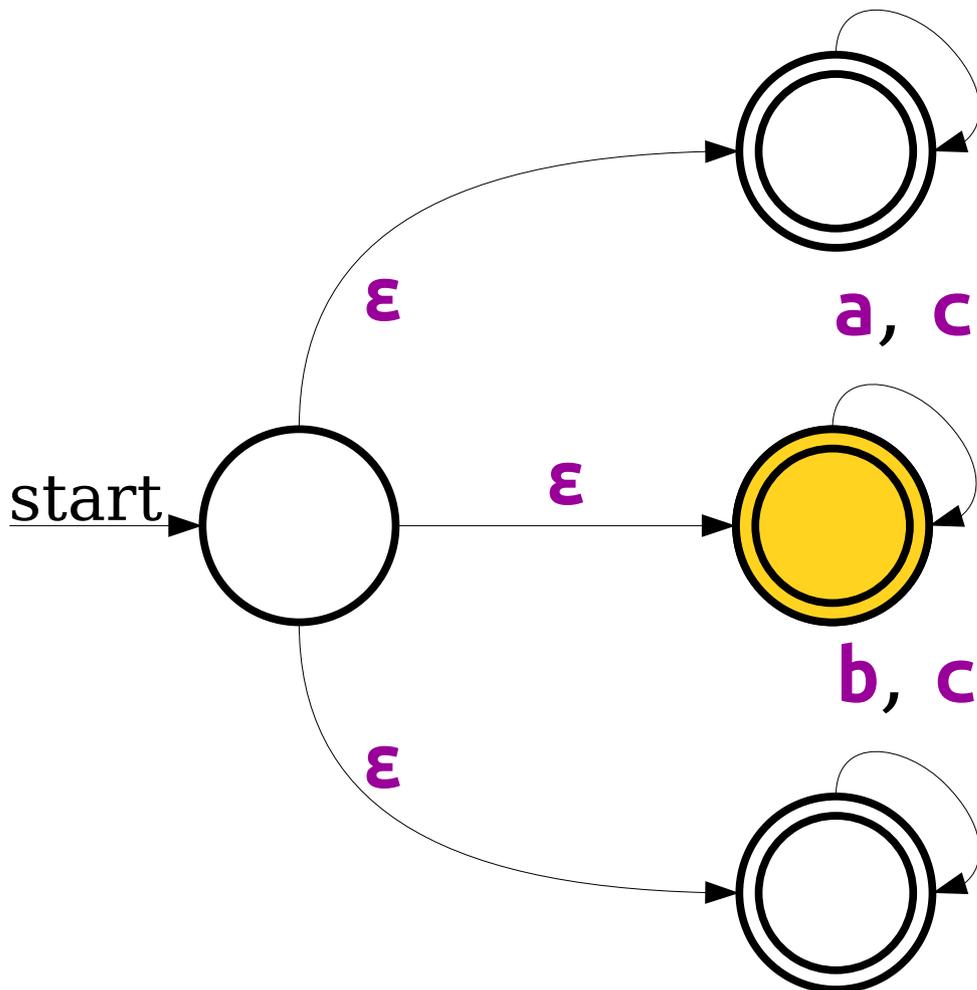
# Guess-and-Check

$L = \{ w \in \{a, b, c\}^* \mid \text{at least one of } a, b, \text{ or } c \text{ is not in } w \}$



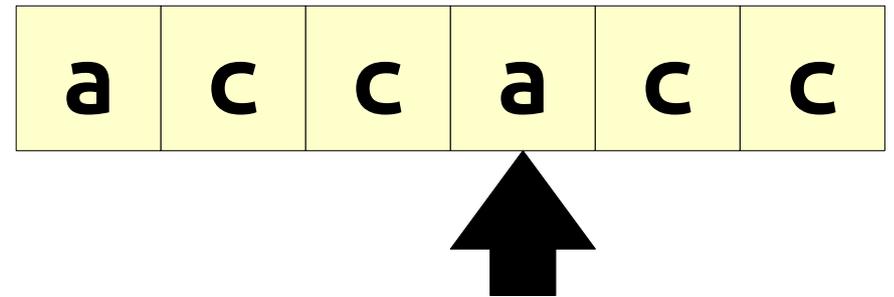
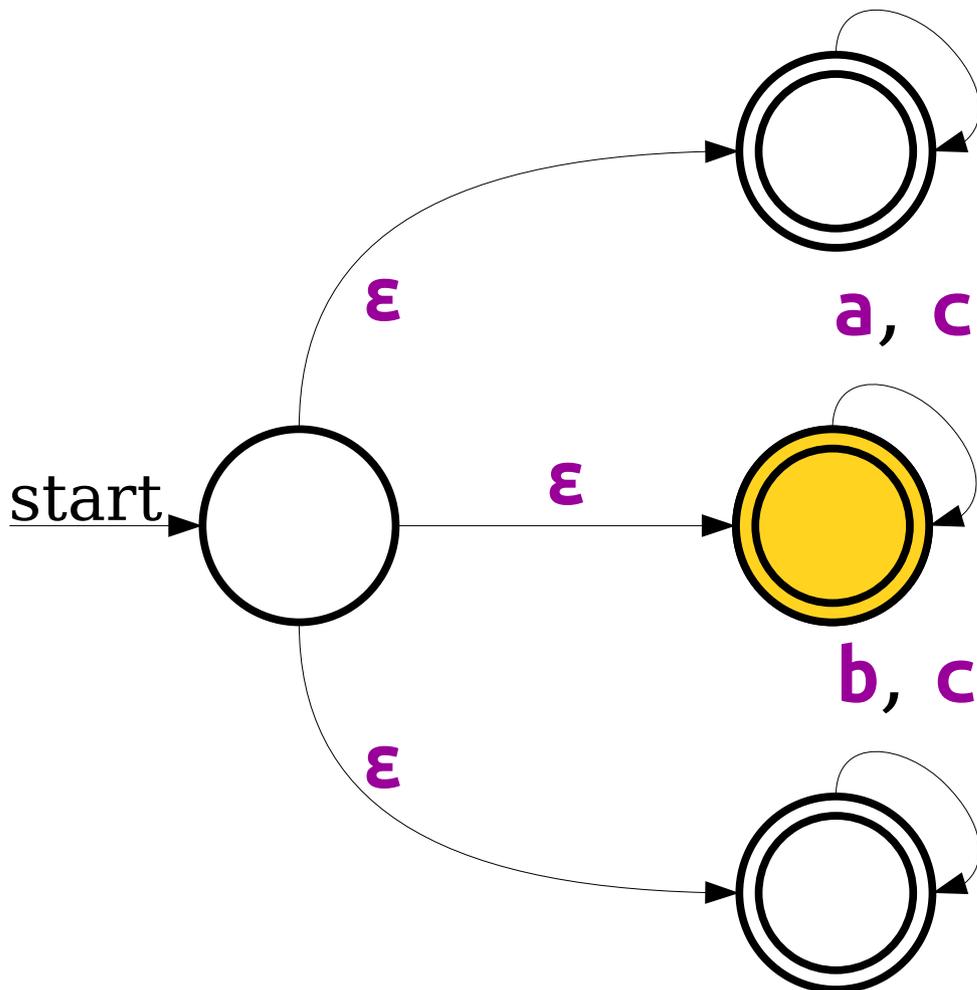
# Guess-and-Check

$L = \{ w \in \{a, b, c\}^* \mid \text{at least one of } a, b, \text{ or } c \text{ is not in } w \}$



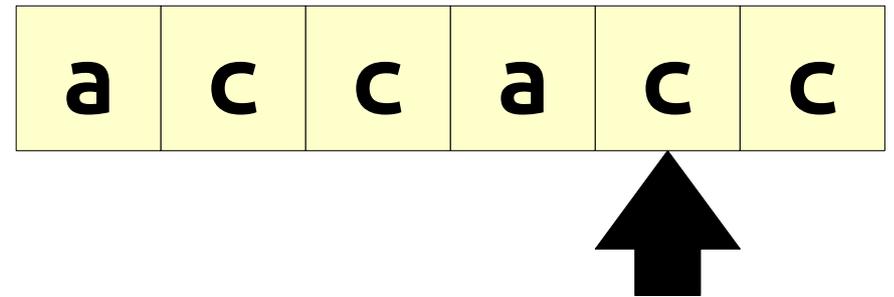
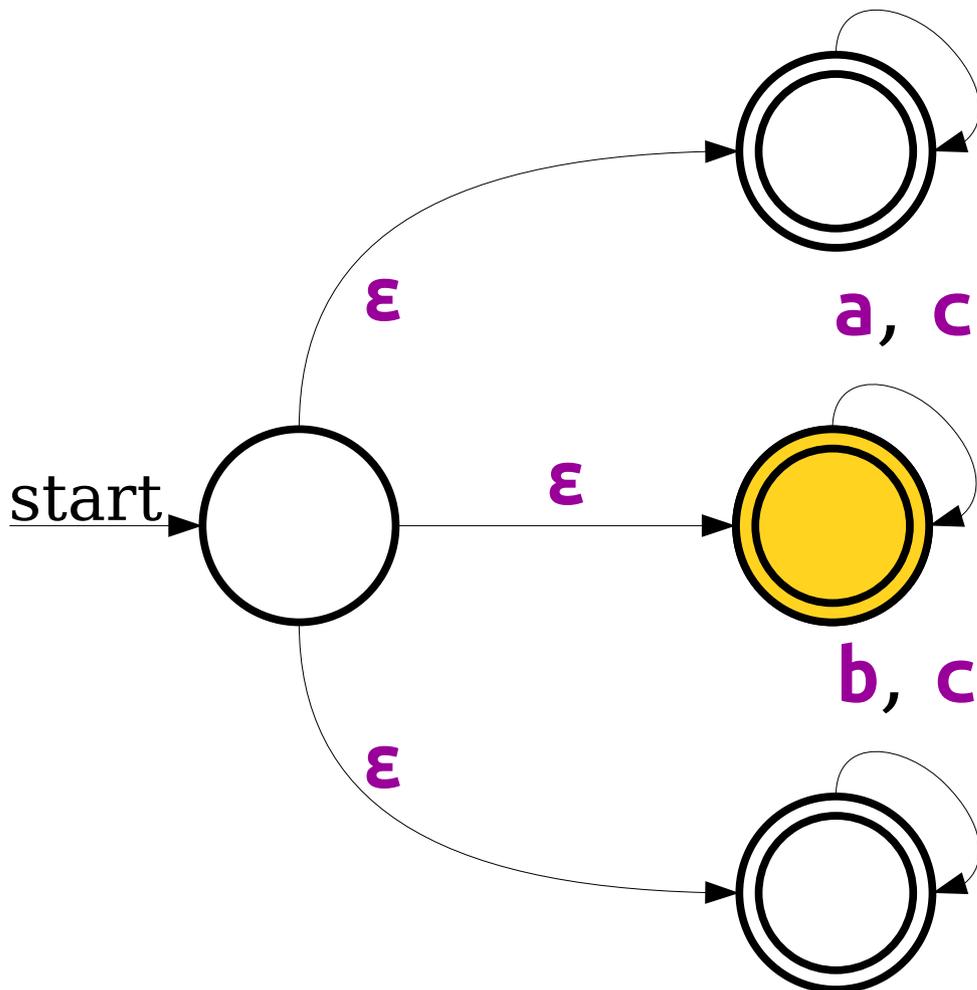
# Guess-and-Check

$L = \{ w \in \{a, b, c\}^* \mid \text{at least one of } a, b, \text{ or } c \text{ is not in } w \}$



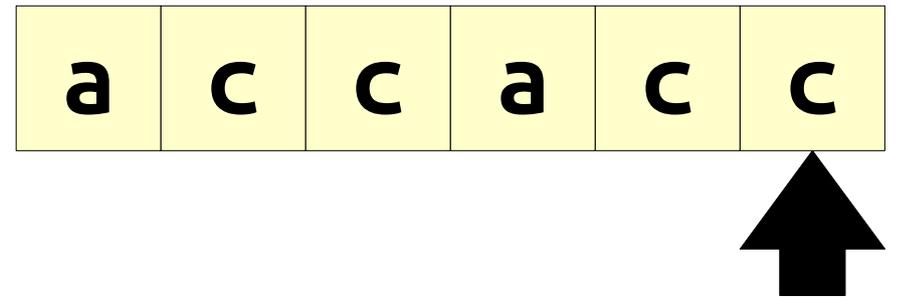
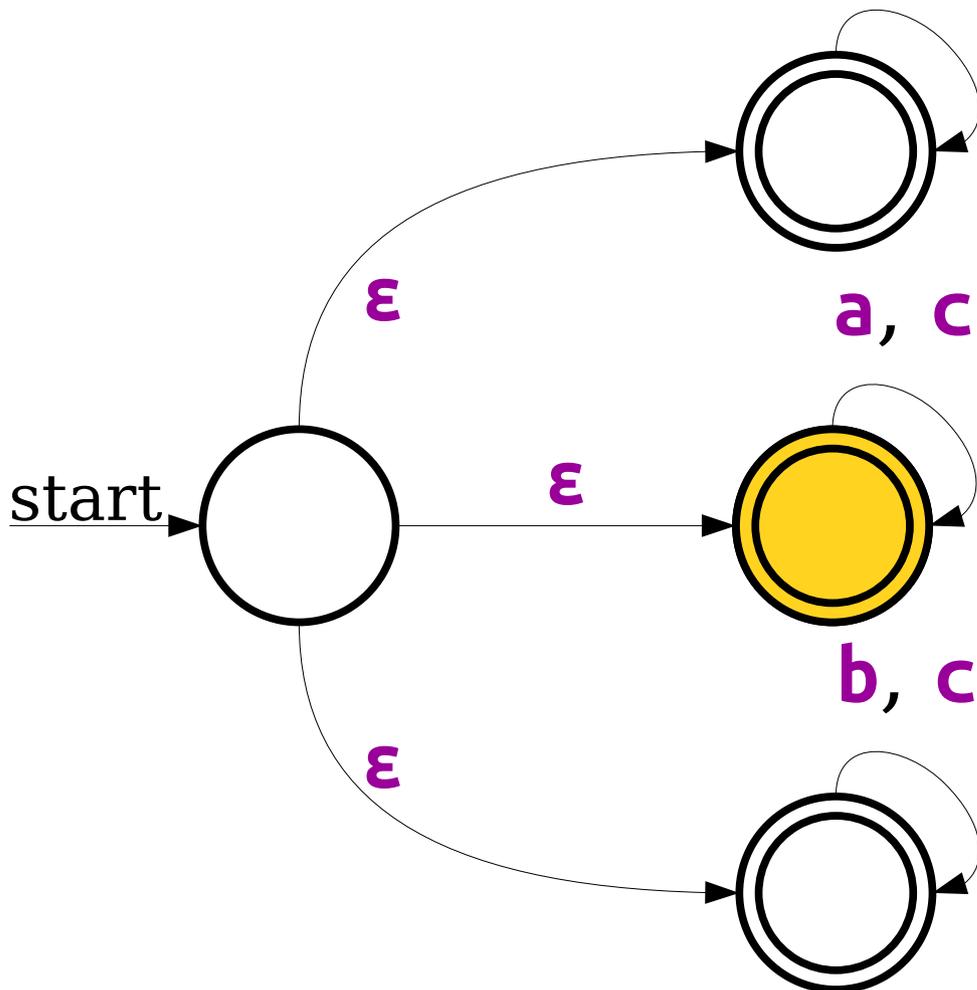
# Guess-and-Check

$L = \{ w \in \{a, b, c\}^* \mid \text{at least one of } a, b, \text{ or } c \text{ is not in } w \}$



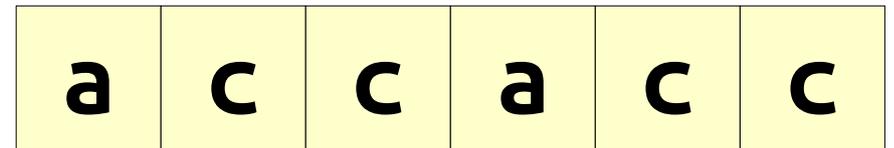
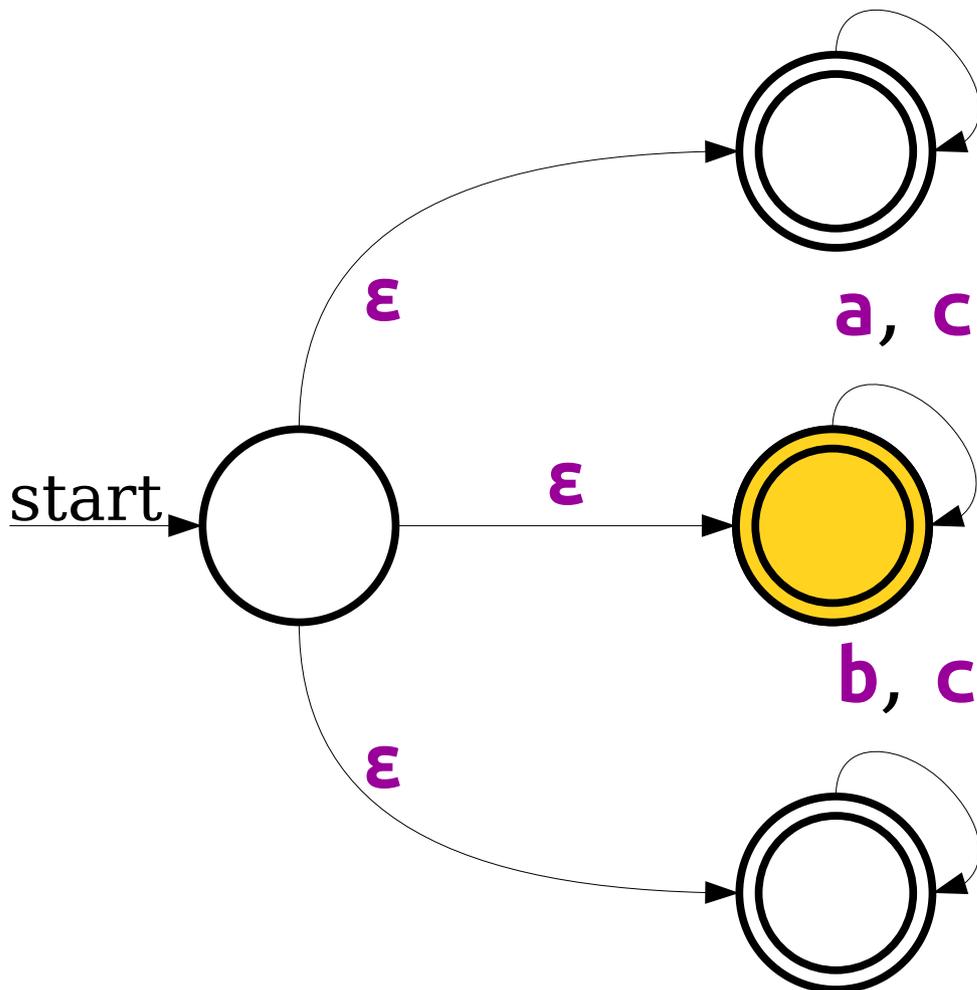
# Guess-and-Check

$L = \{ w \in \{a, b, c\}^* \mid \text{at least one of } a, b, \text{ or } c \text{ is not in } w \}$



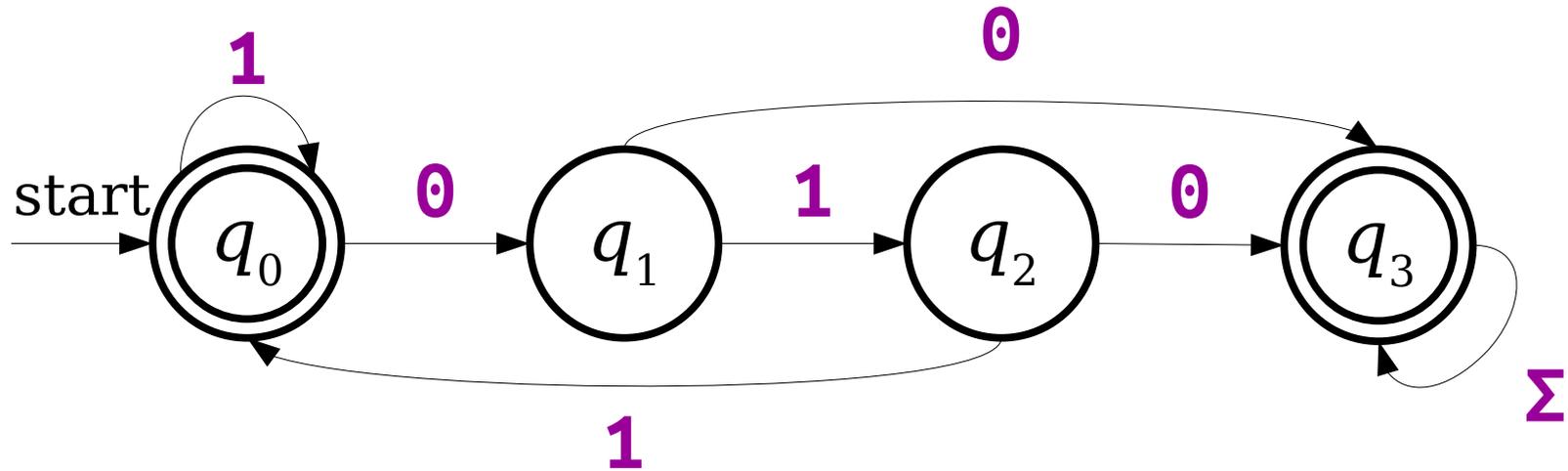
# Guess-and-Check

$L = \{ w \in \{a, b, c\}^* \mid \text{at least one of } a, b, \text{ or } c \text{ is not in } w \}$



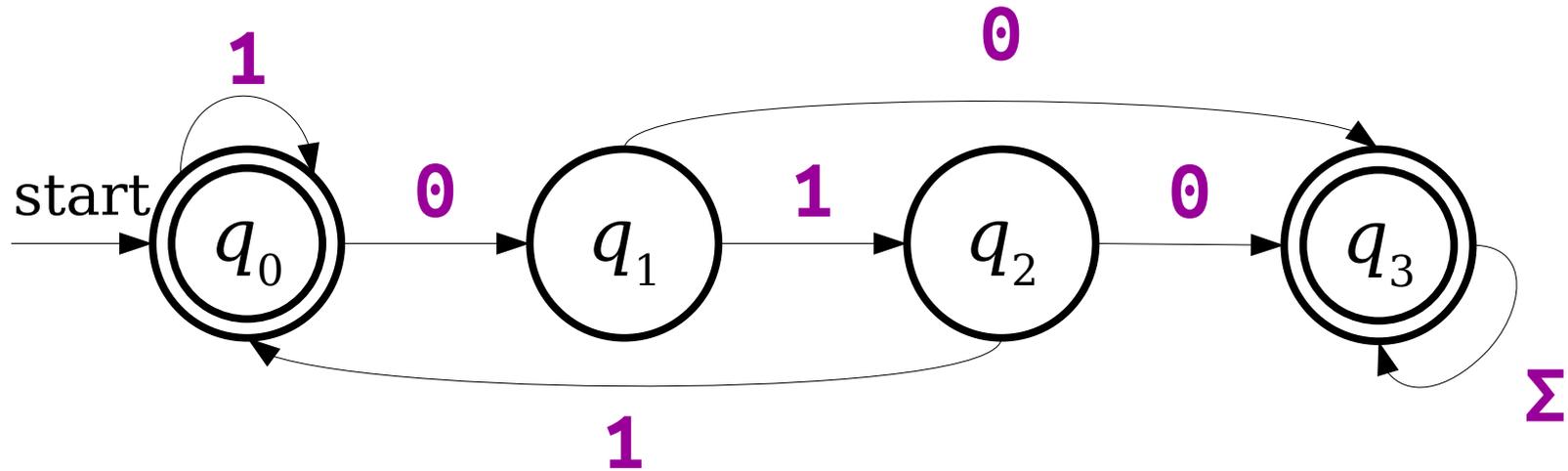
# Implementing DFAs and NFAs

# Tabular DFAs



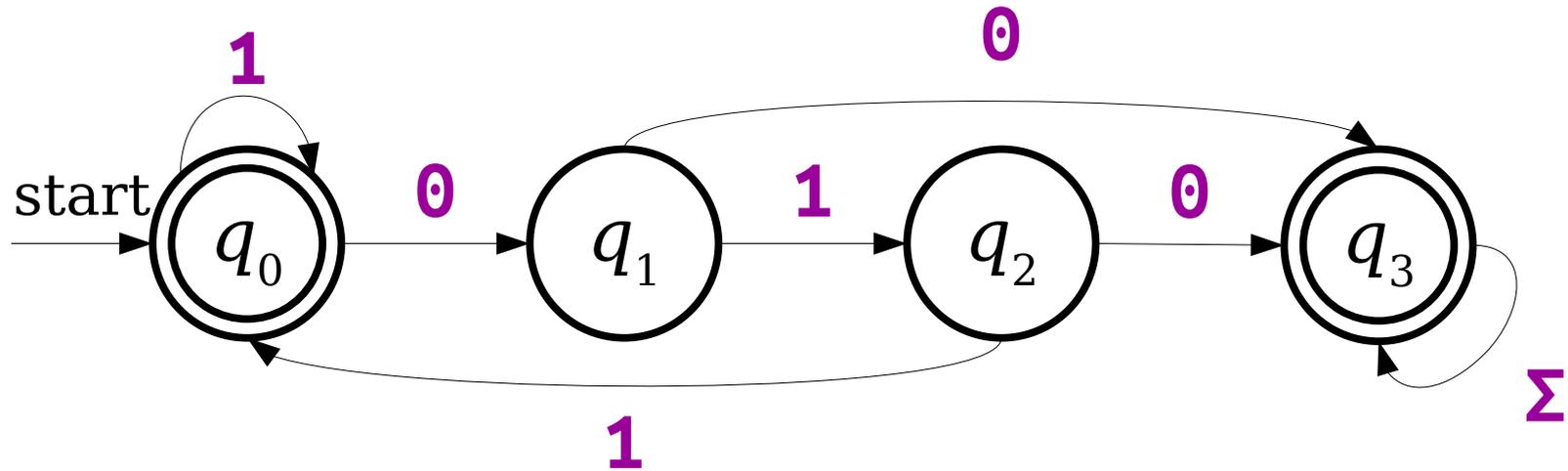
	0	1
$q_0$		
$q_1$		
$q_2$		
$q_3$		

# Tabular DFAs



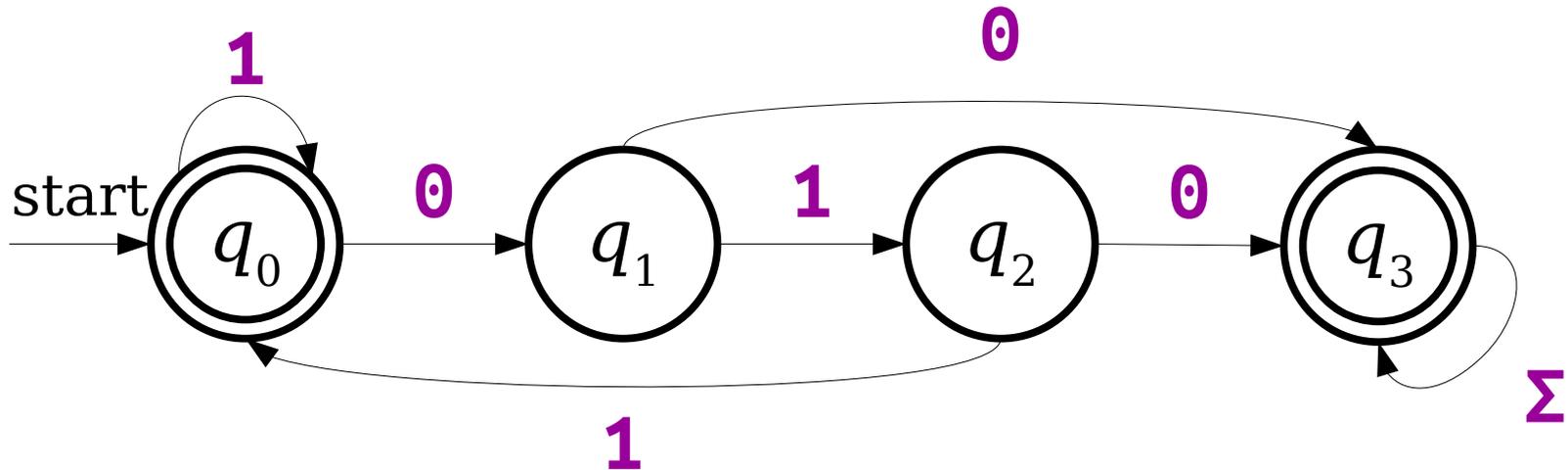
	0	1
$q_0$	$q_1$	$q_0$
$q_1$	$q_3$	$q_2$
$q_2$	$q_3$	$q_0$
$q_3$	$q_3$	$q_3$

# Tabular DFAs



	0	1
* $q_0$	$q_1$	$q_0$
$q_1$	$q_3$	$q_2$
$q_2$	$q_3$	$q_0$
* $q_3$	$q_3$	$q_3$

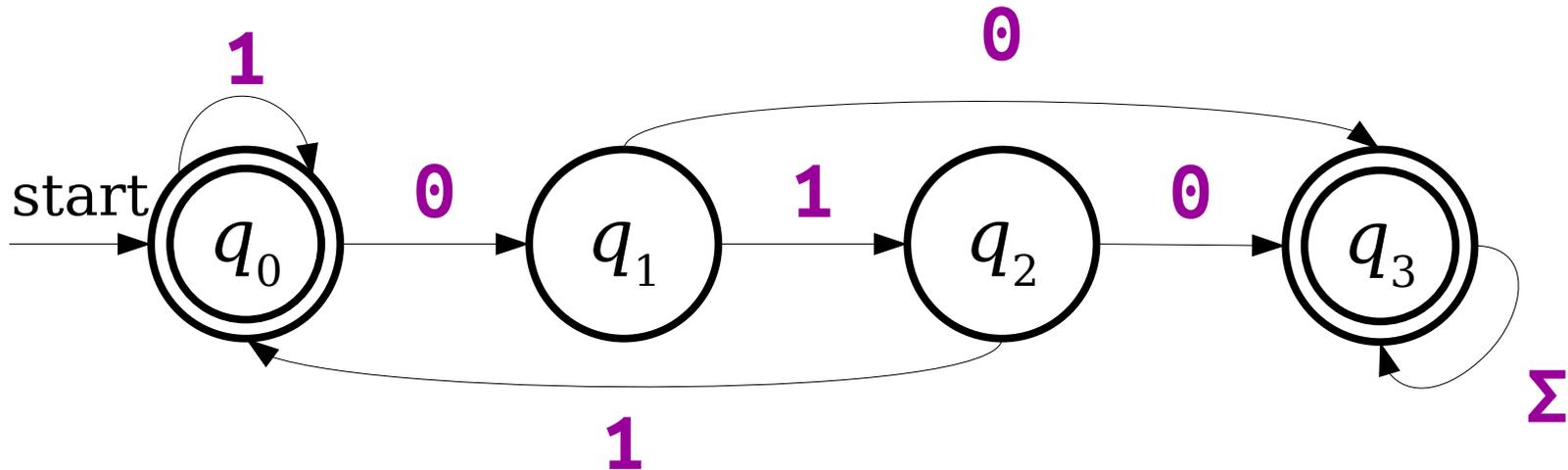
# Tabular DFAs



	0	1
* $q_0$	$q_1$	$q_0$
$q_1$	$q_3$	$q_2$
$q_2$	$q_3$	$q_0$
* $q_3$	$q_3$	$q_3$

These stars indicate accepting states.

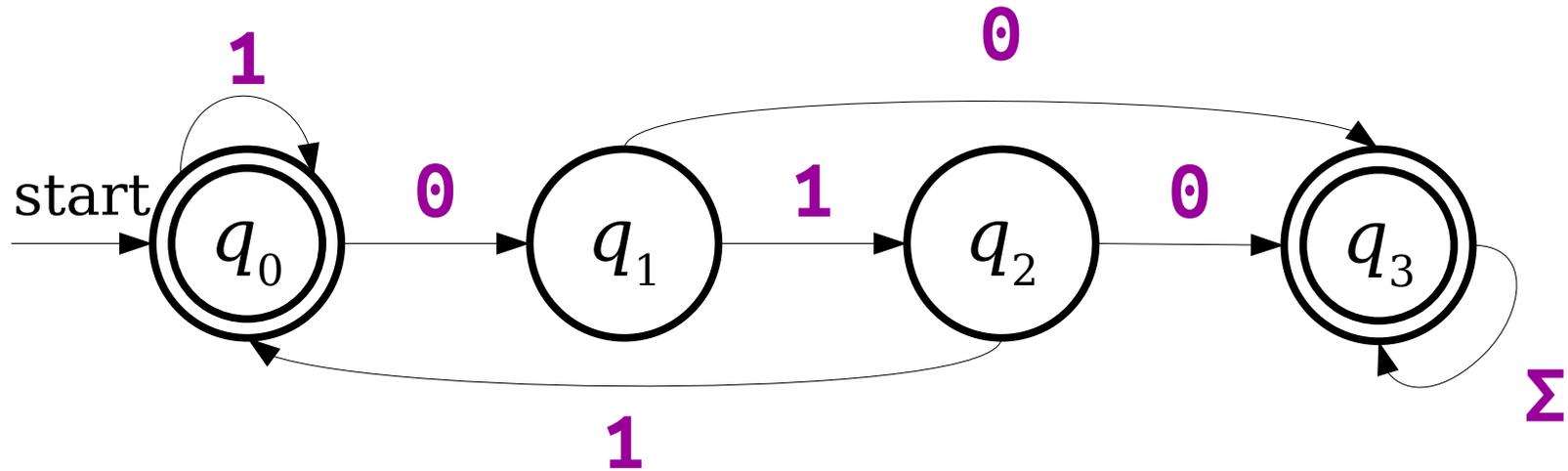
# Tabular DFAs



Since this is the first row, it's the start state.

	0	1
* $q_0$	$q_1$	$q_0$
$q_1$	$q_3$	$q_2$
$q_2$	$q_3$	$q_0$
* $q_3$	$q_3$	$q_3$

# Tabular DFAs



	0	1
* $q_0$	$q_1$	$q_0$
$q_1$	$q_3$	$q_2$
$q_2$	$q_3$	$q_0$
* $q_3$	$q_3$	$q_3$

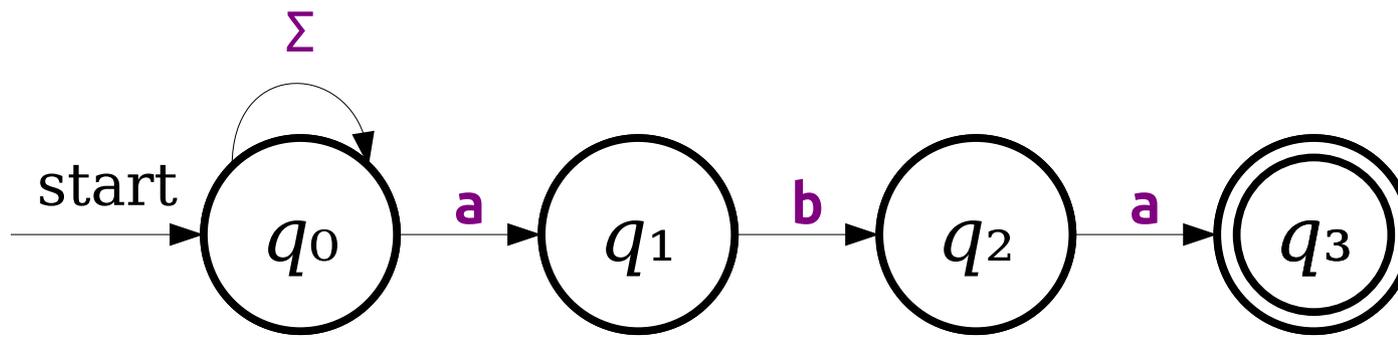
Question to ponder: Why isn't there a column here for  $\Sigma$ ?

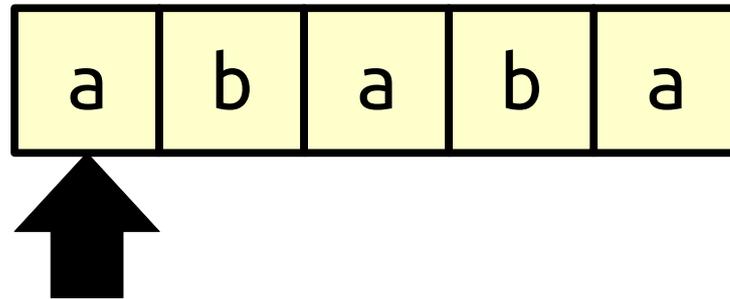
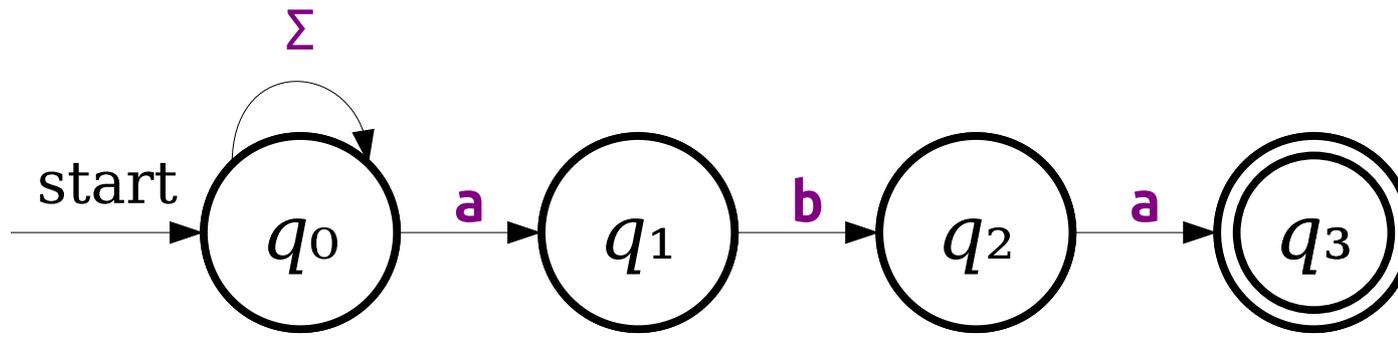
# Code? In a Theory Class?

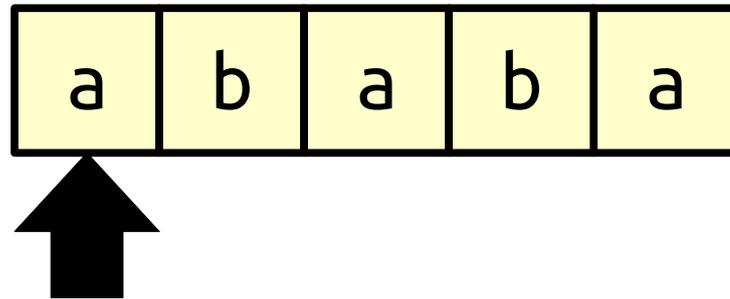
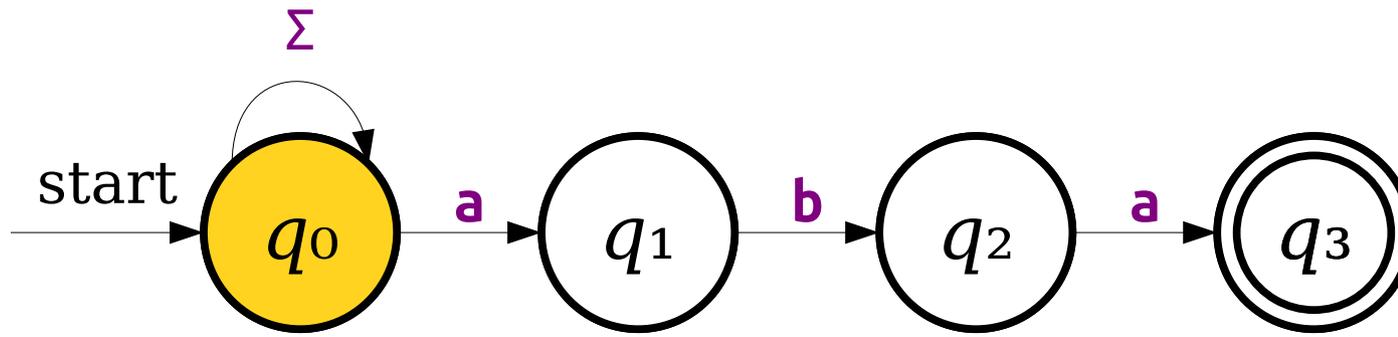
```
int kTransitionTable[kNumStates][kNumSymbols] = {  
    {0, 0, 1, 3, 7, 1, ...},  
    ...  
};  
bool kAcceptTable[kNumStates] = {  
    false,  
    true,  
    true,  
    ...  
};  
bool SimulateDFA(string input) {  
    int state = 0;  
    for (char ch: input) {  
        state = kTransitionTable[state][ch];  
    }  
    return kAcceptTable[state];  
}
```

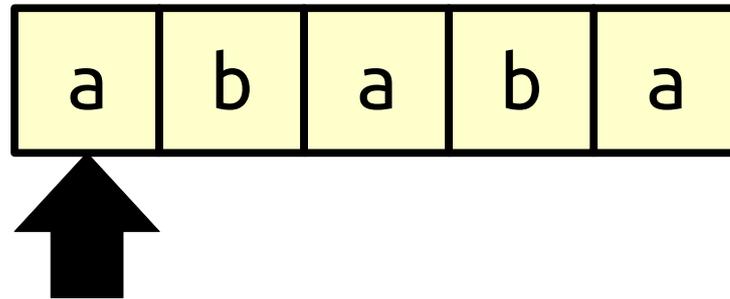
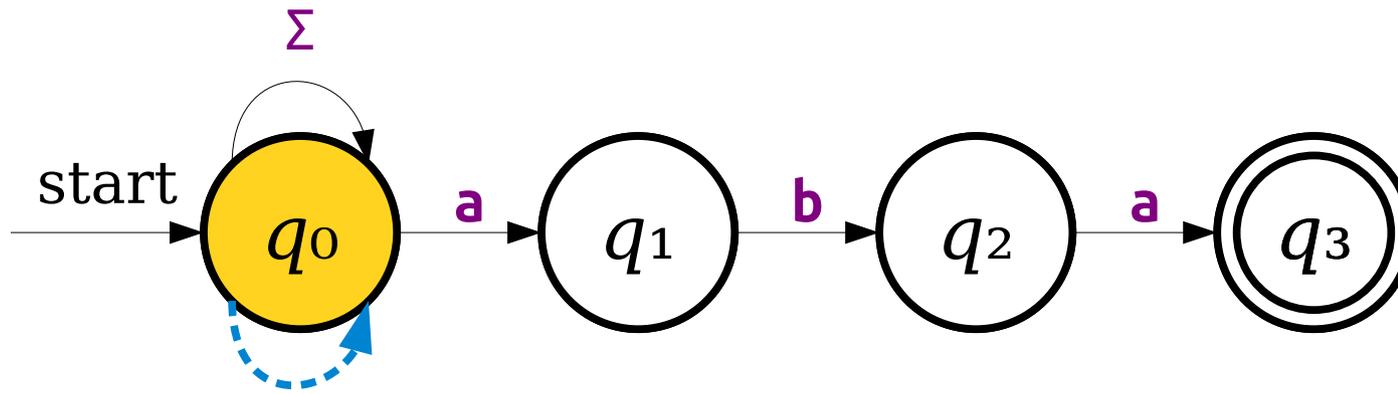
***Thought Experiment:***

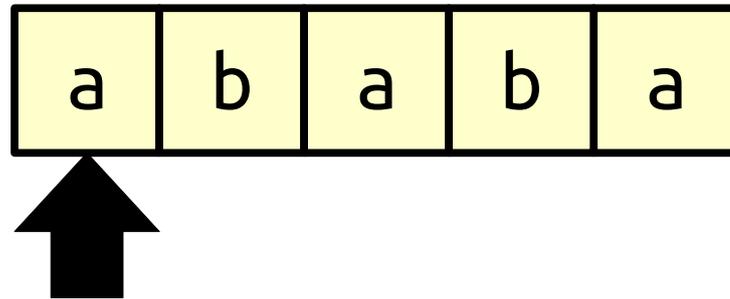
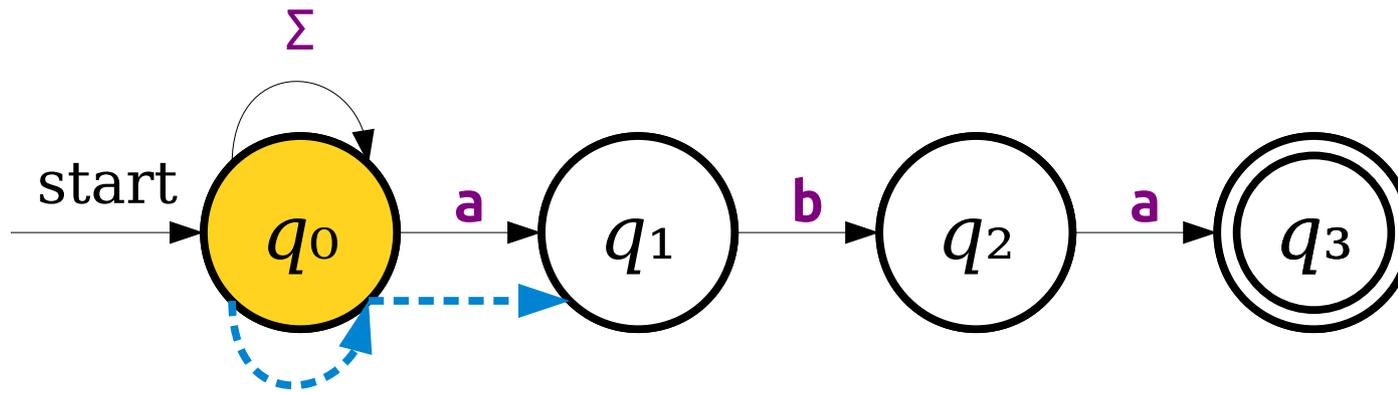
How would you simulate an NFA in software?

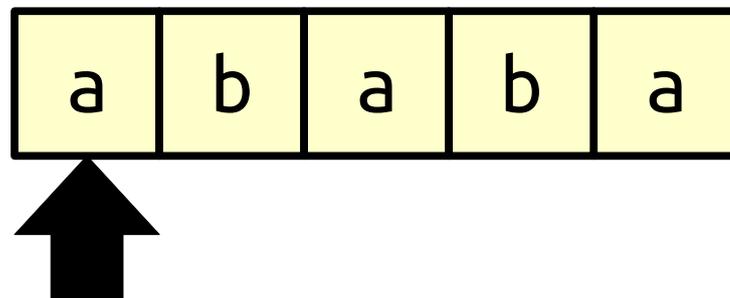
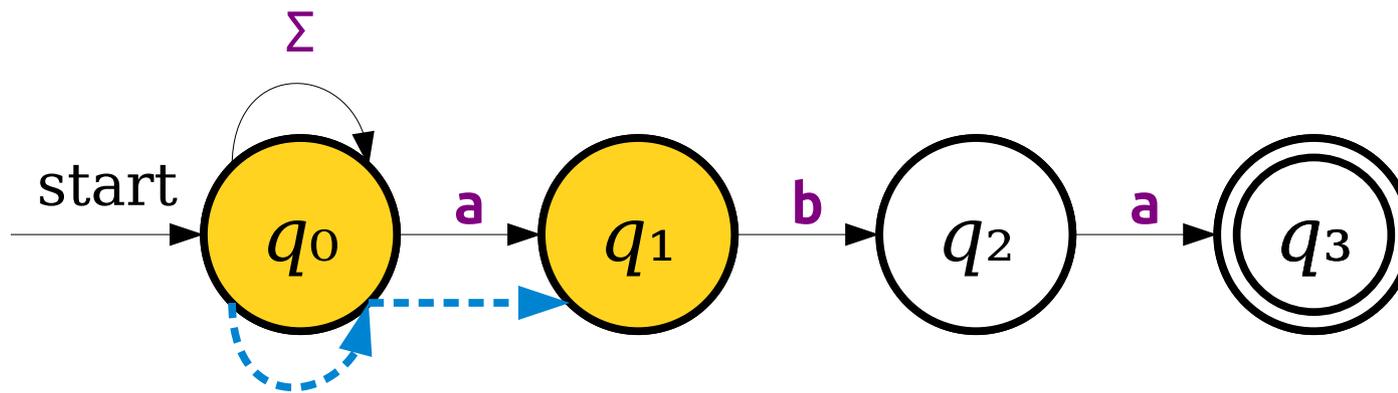


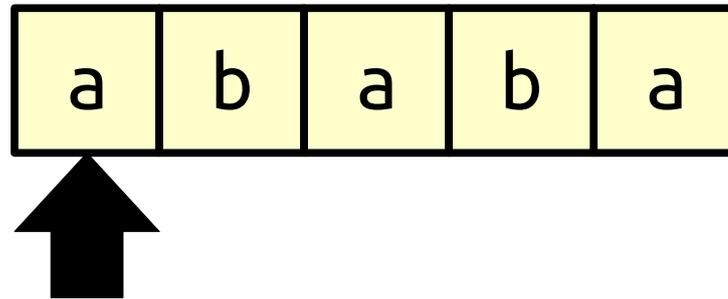
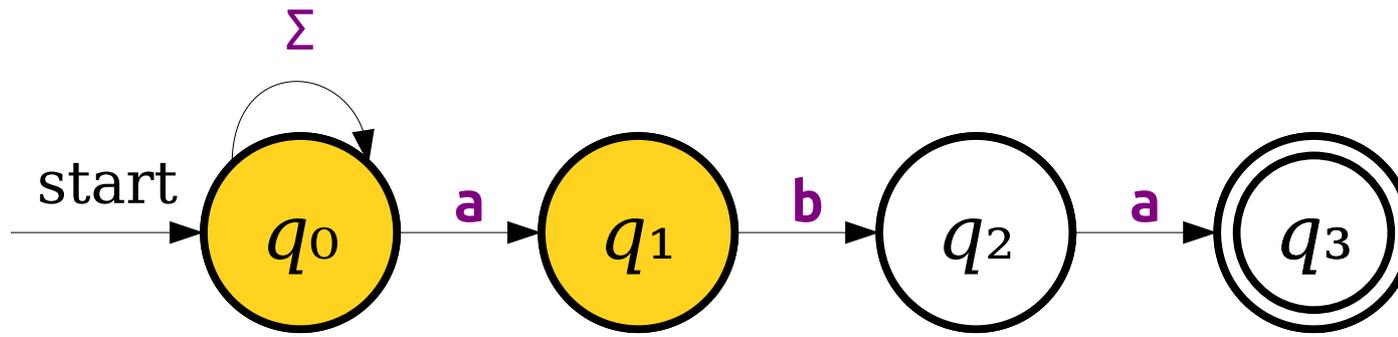


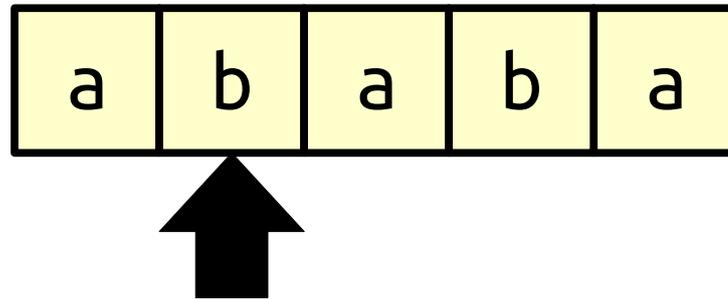
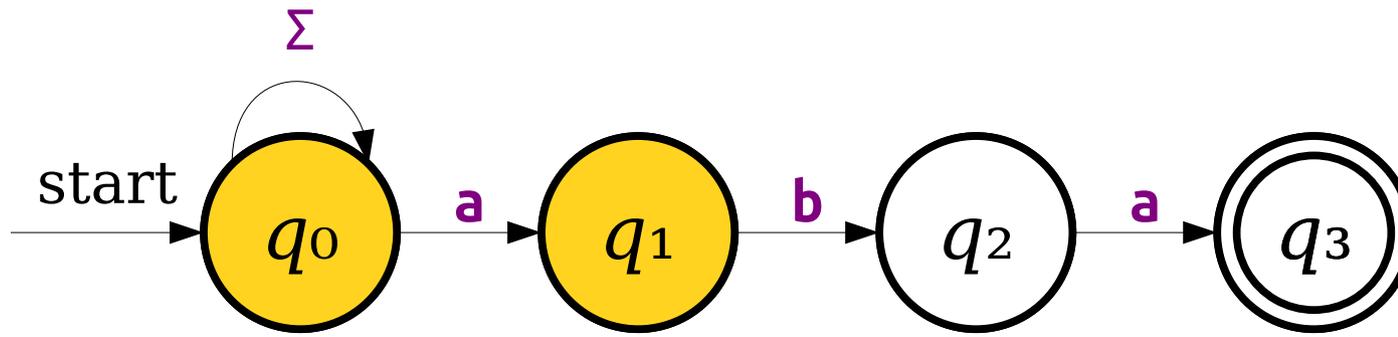


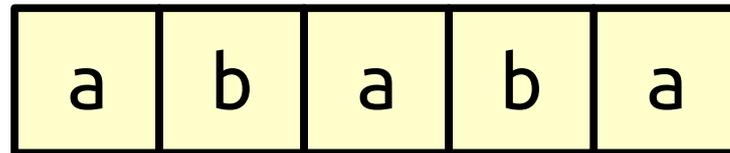
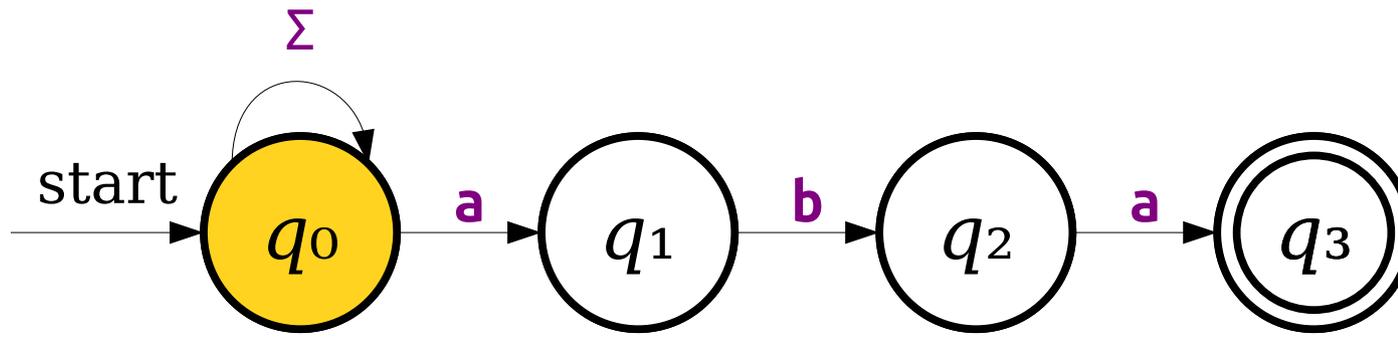


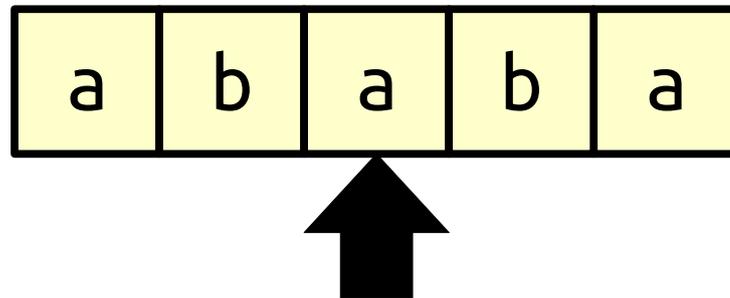
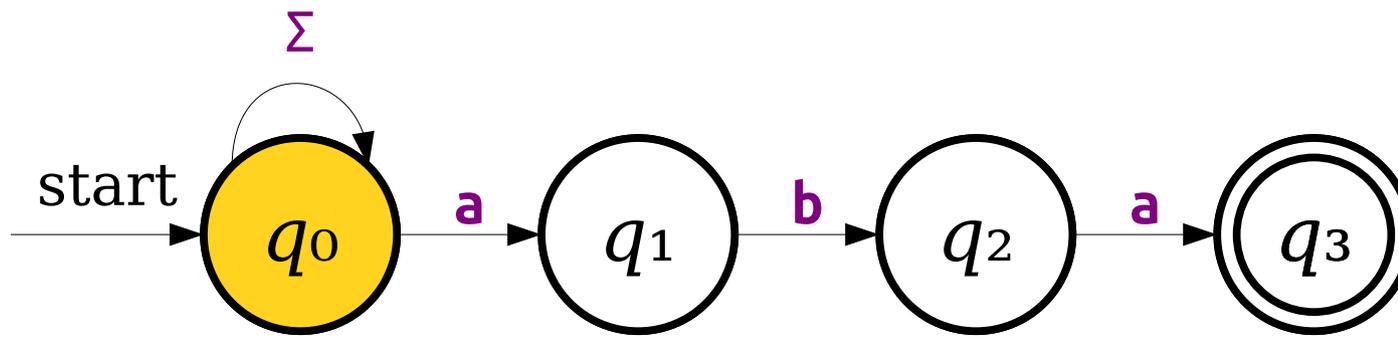


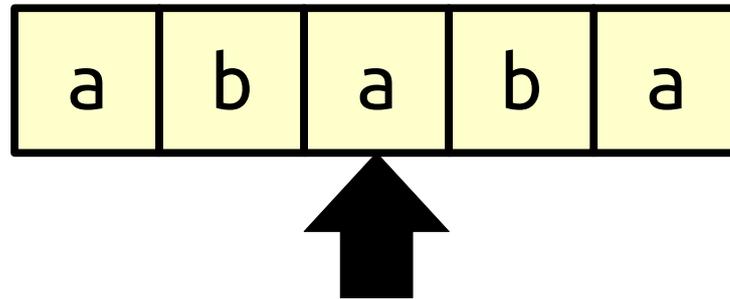
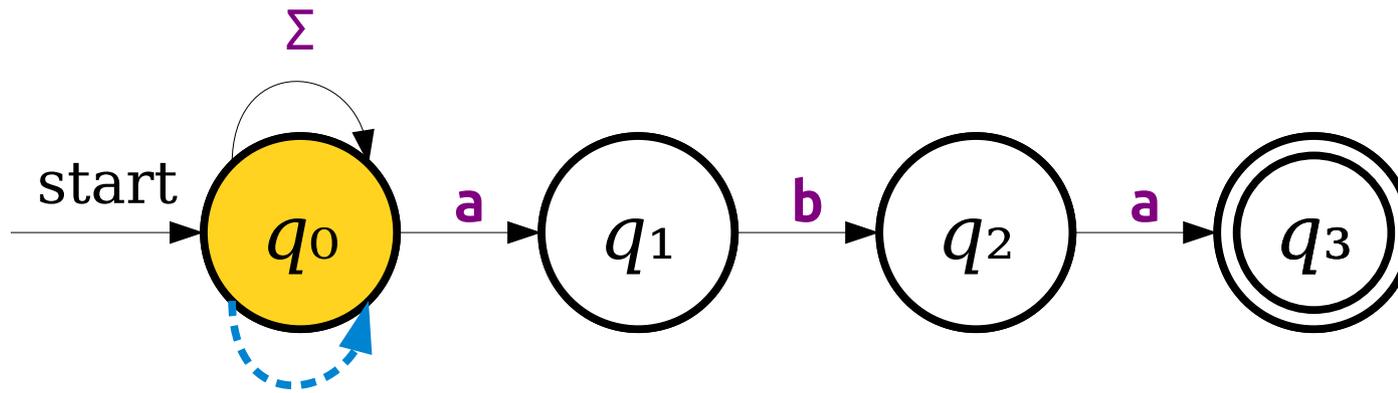


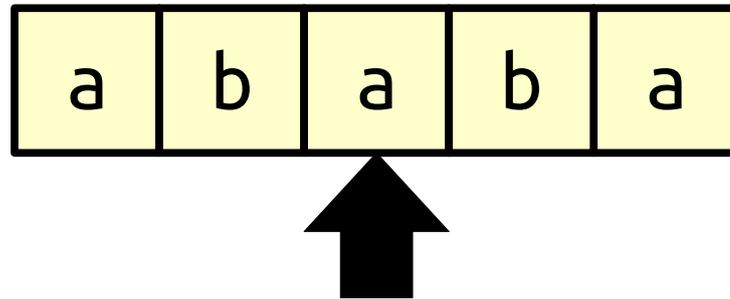
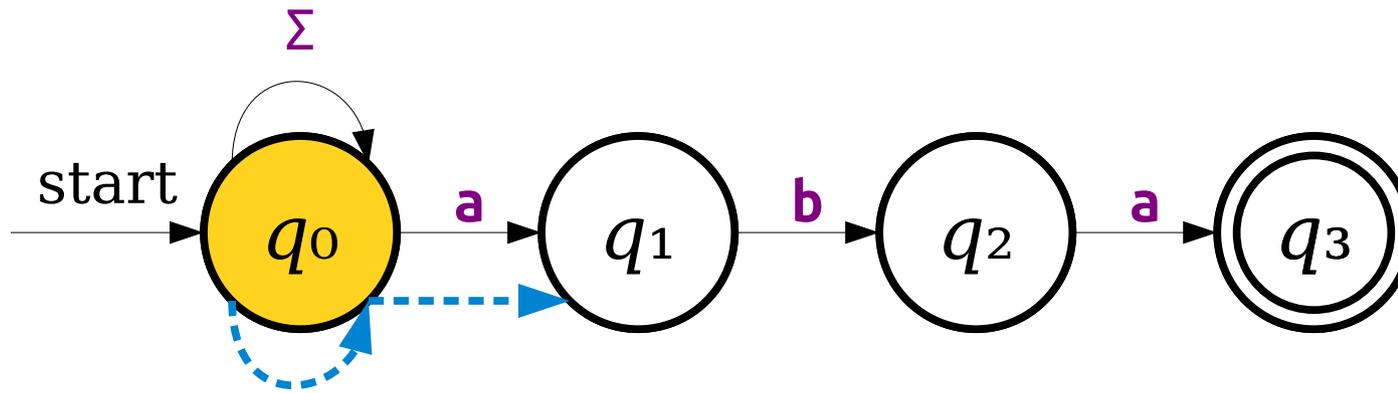


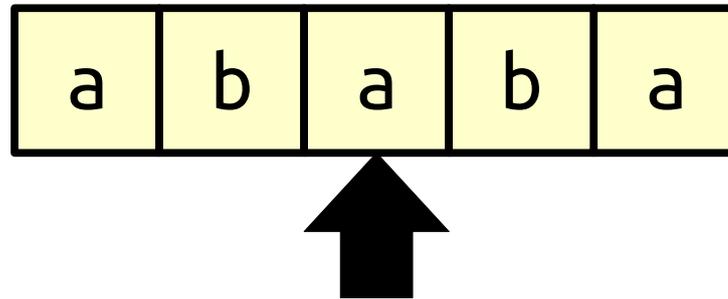
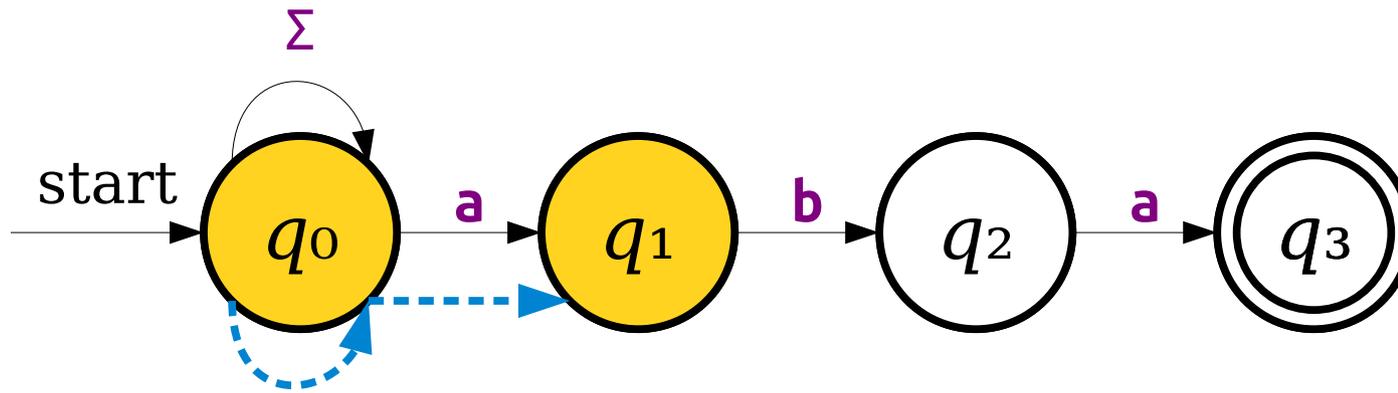


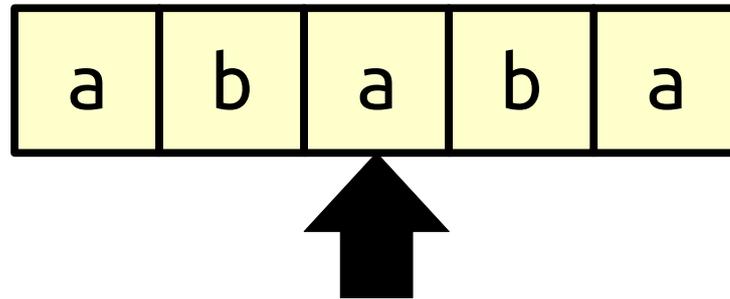
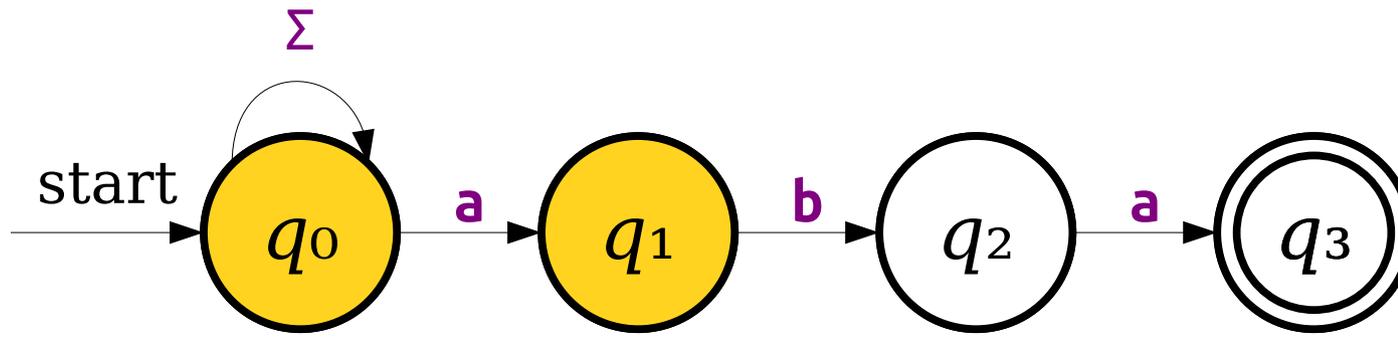


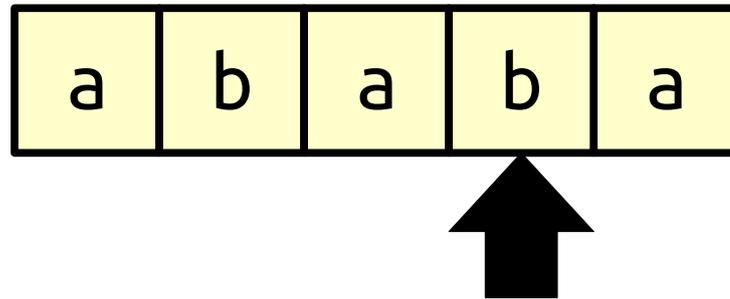
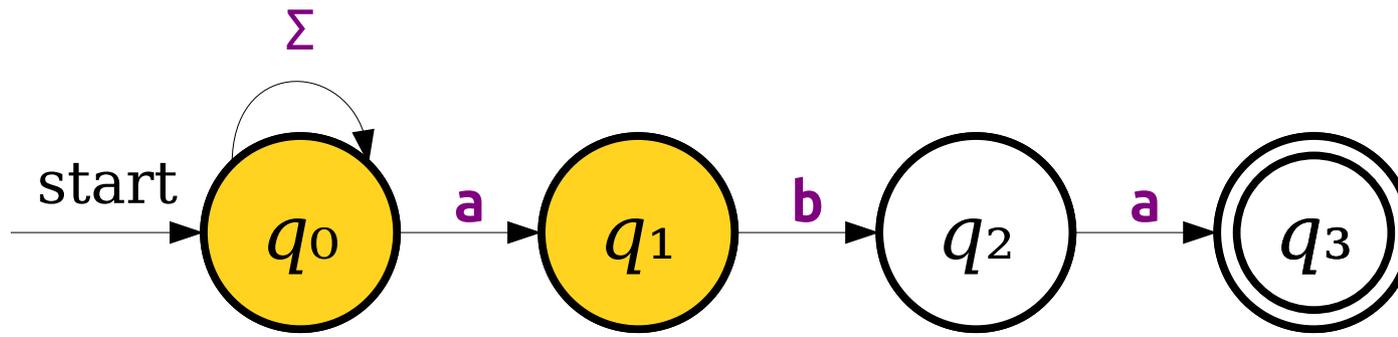


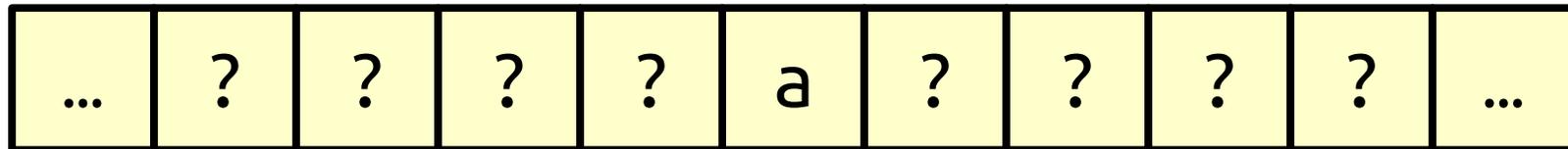
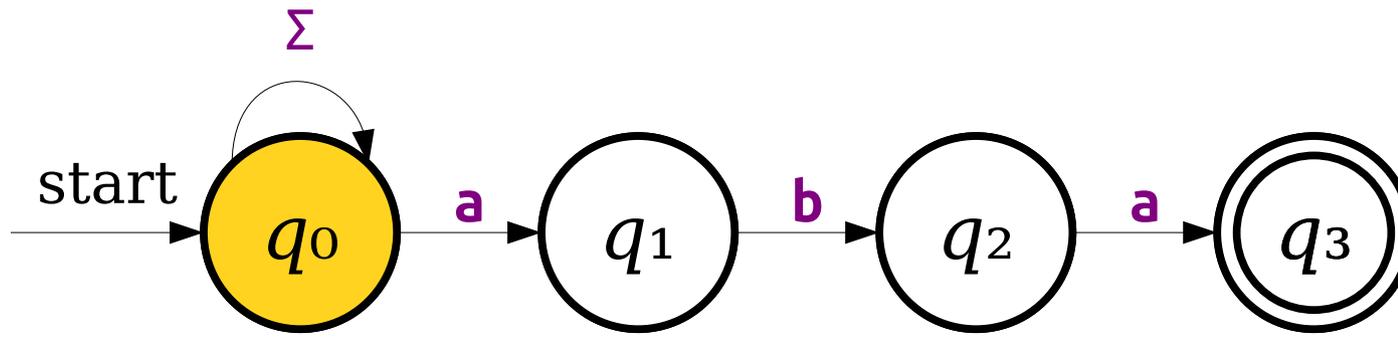


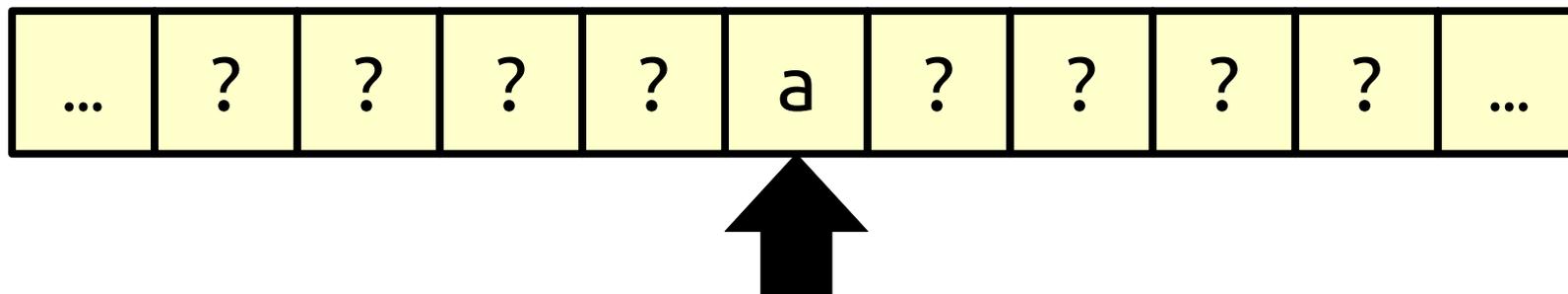
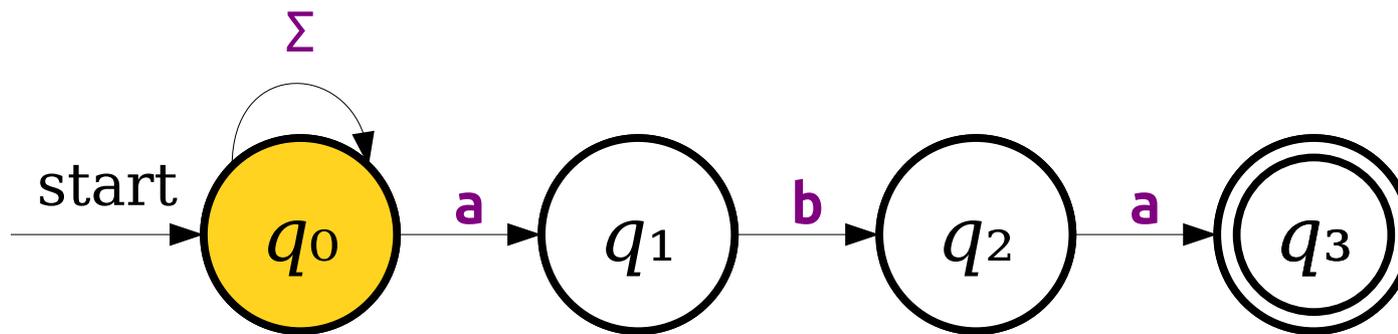


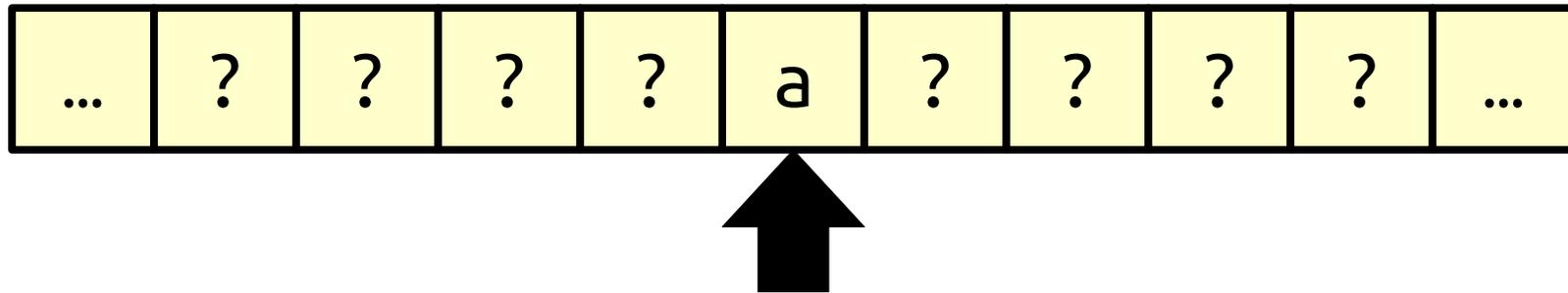
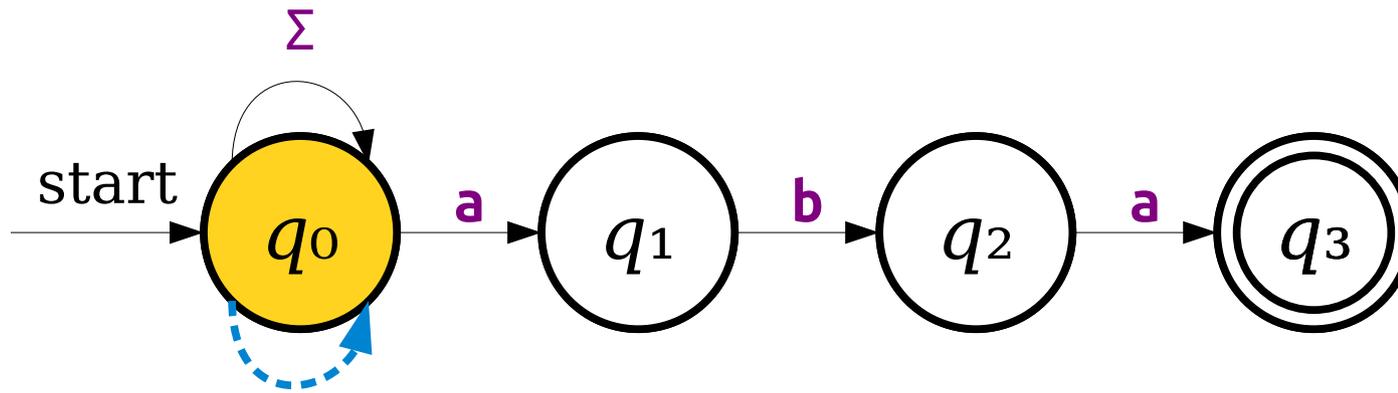


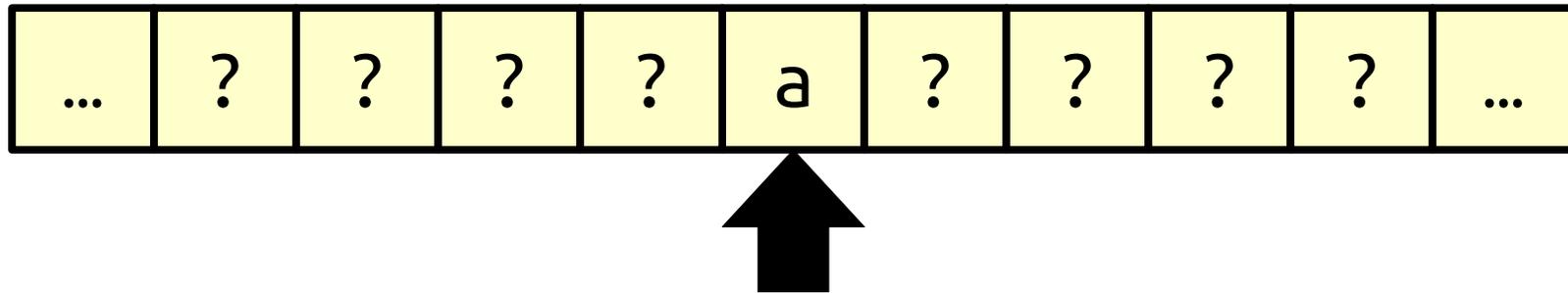
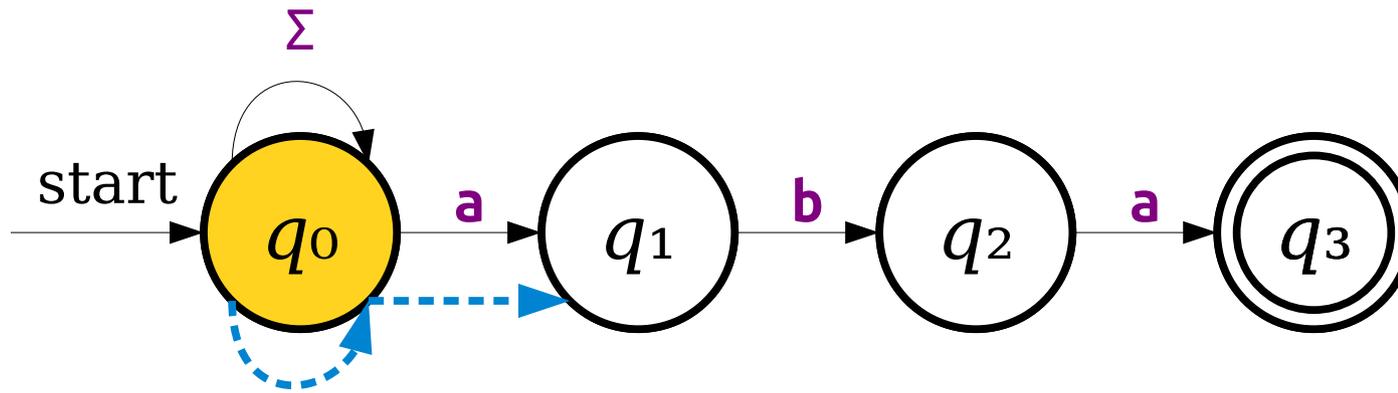


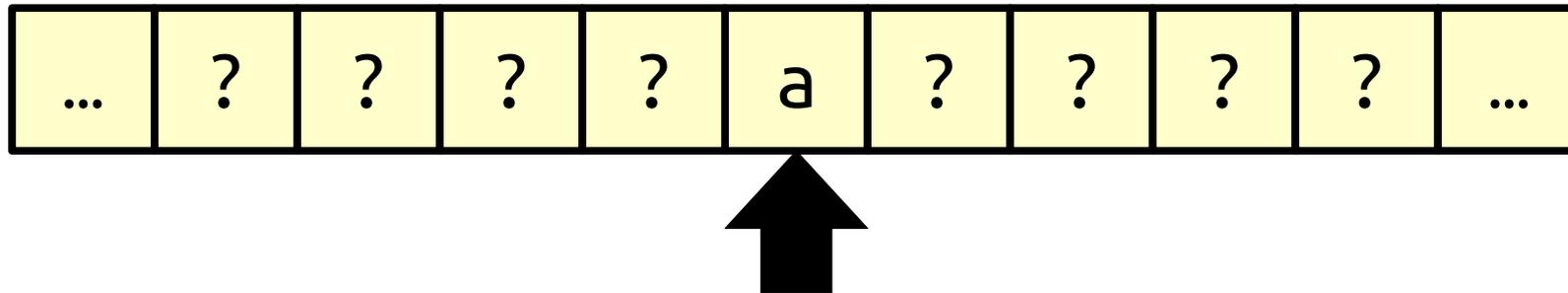
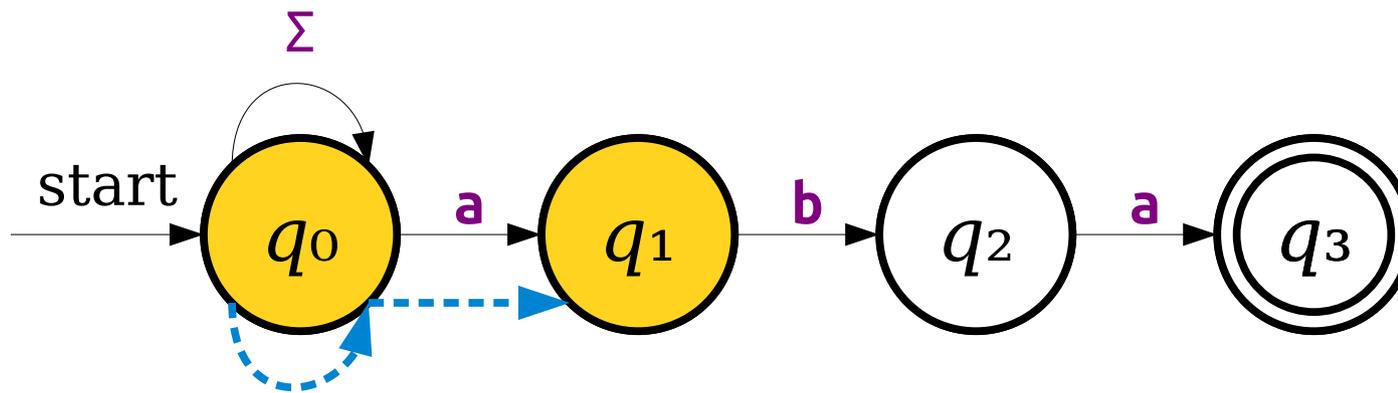


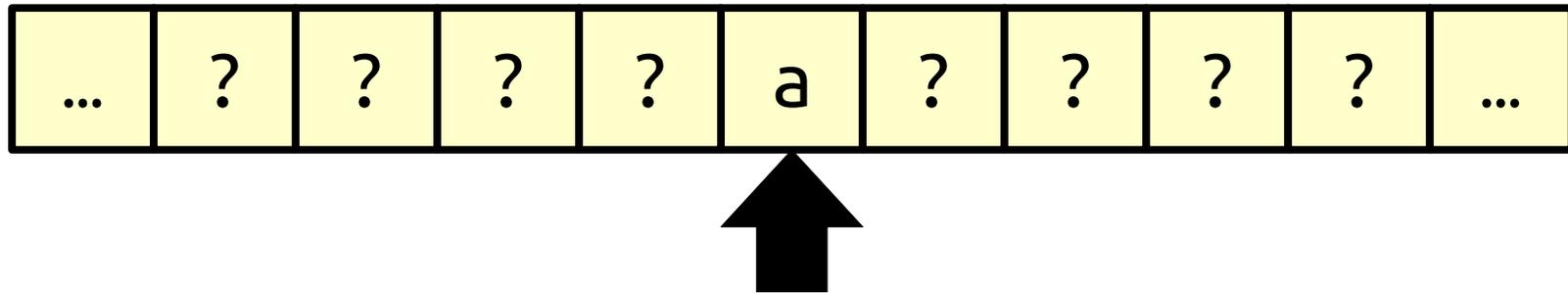
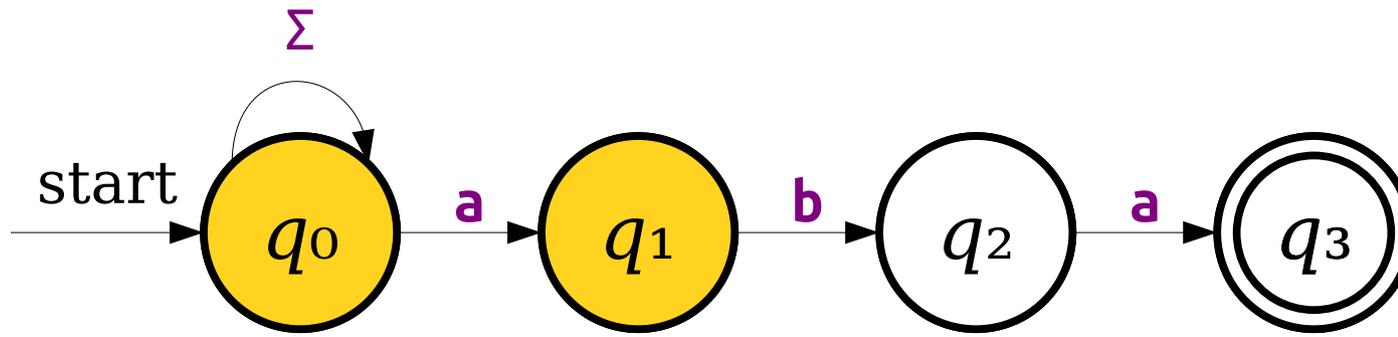


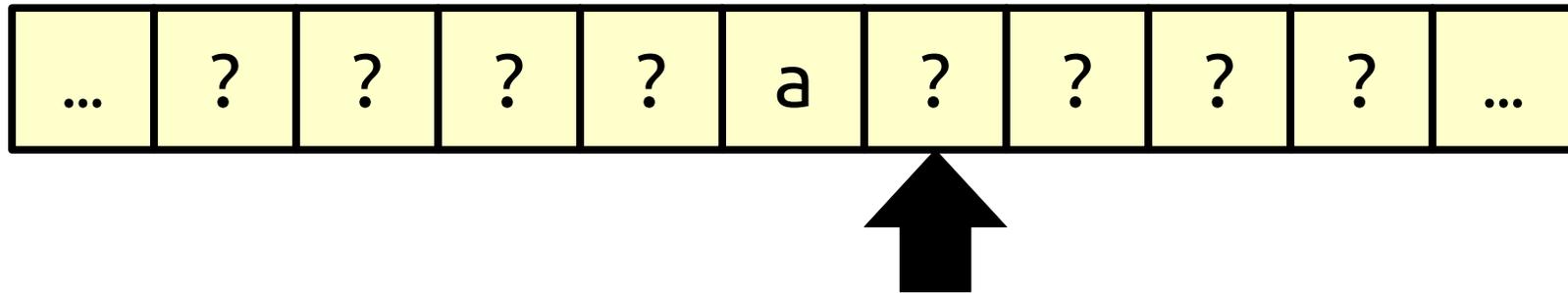
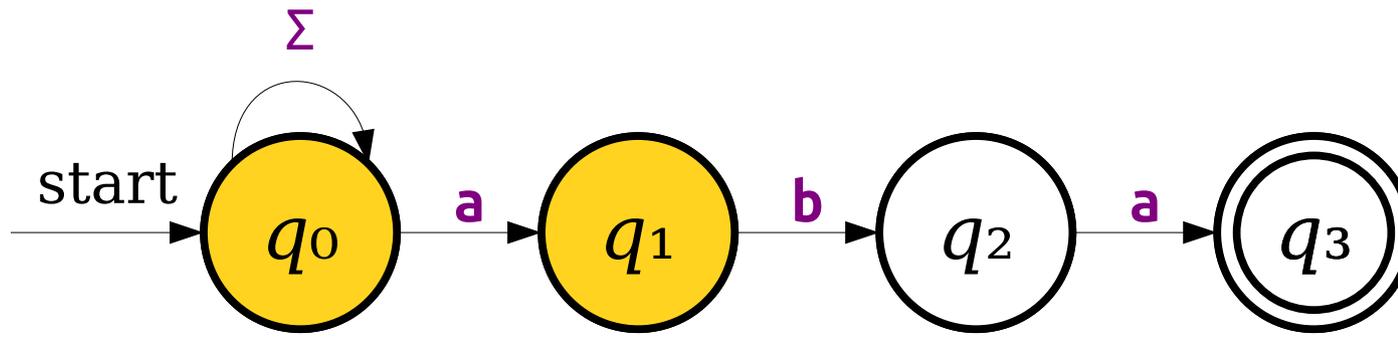


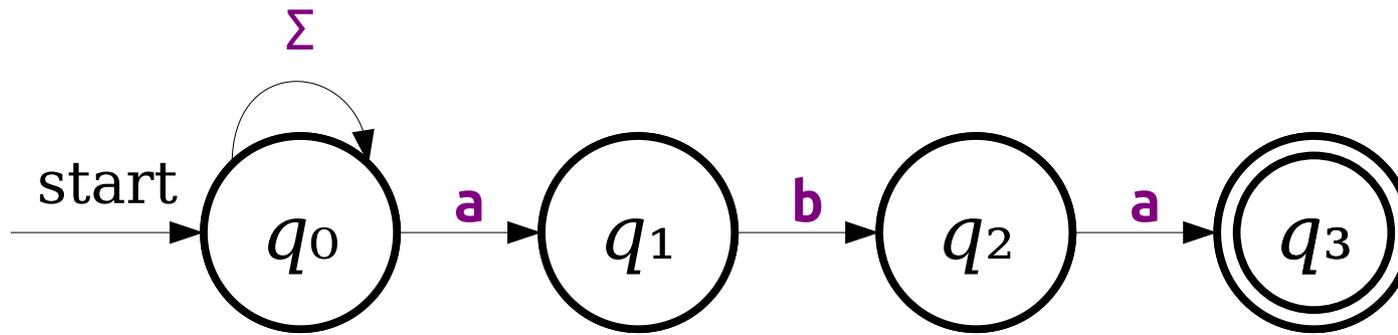




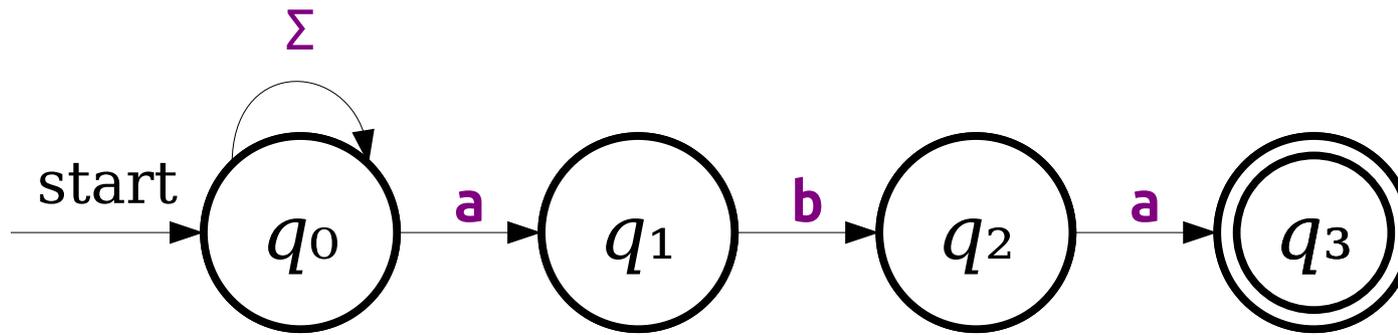




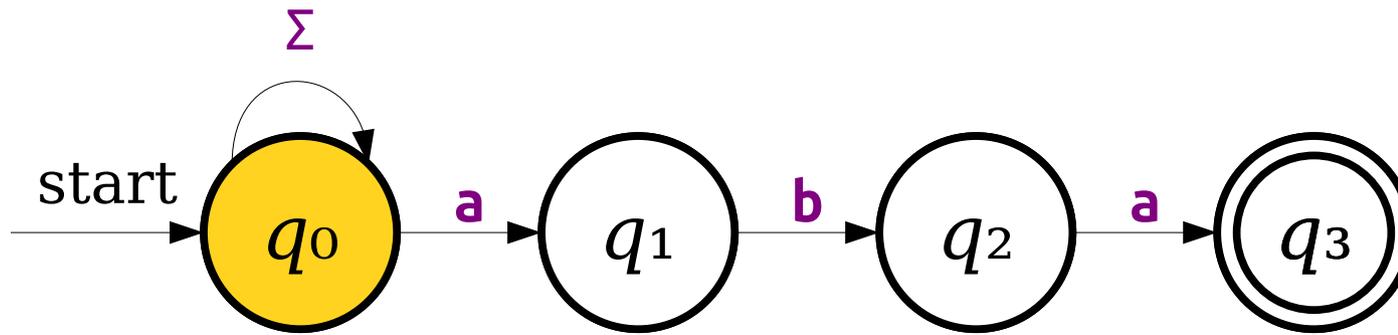




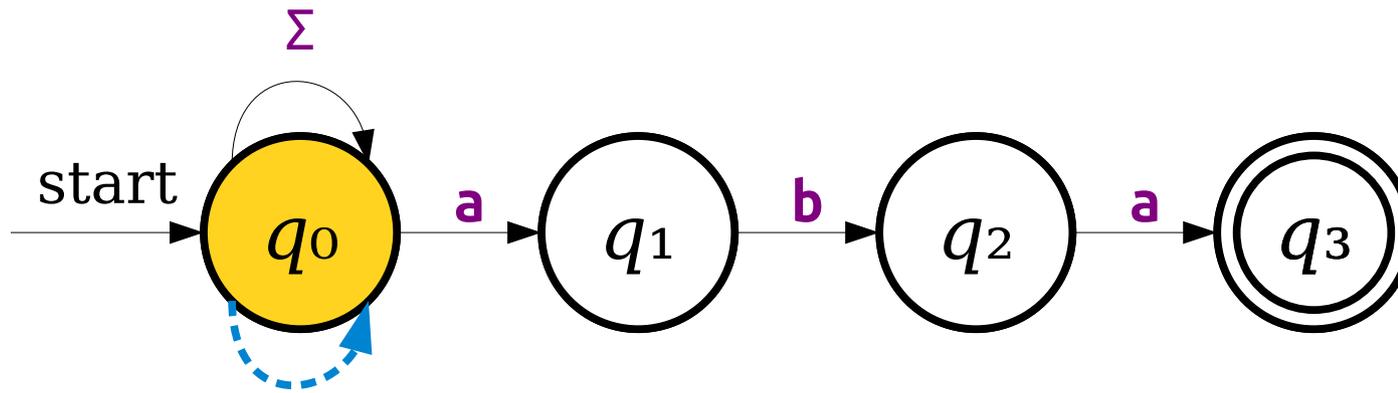
	$a$
$\{q_0\}$	$\{q_0, q_1\}$



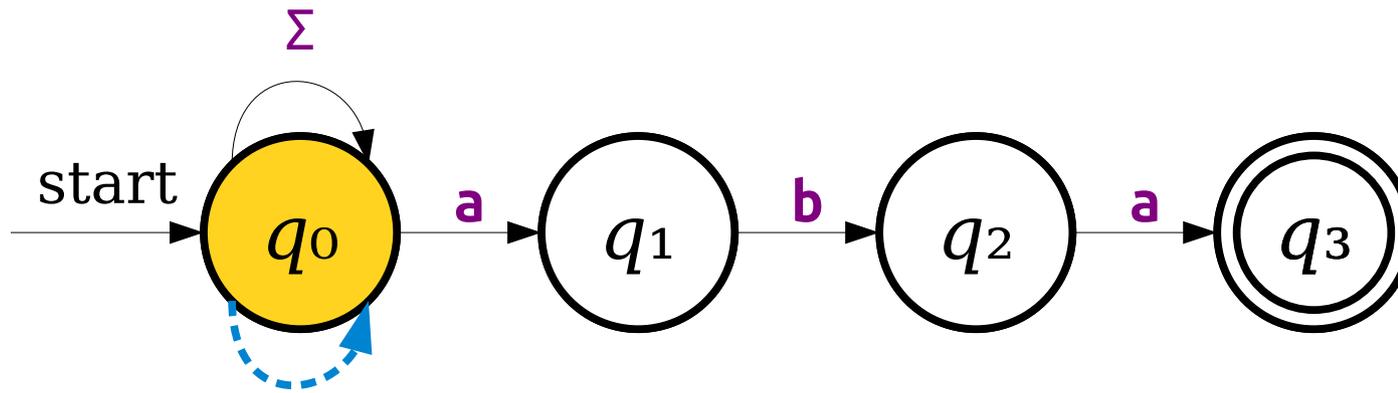
	<i>a</i>	<i>b</i>
$\{q_0\}$	$\{q_0, q_1\}$	



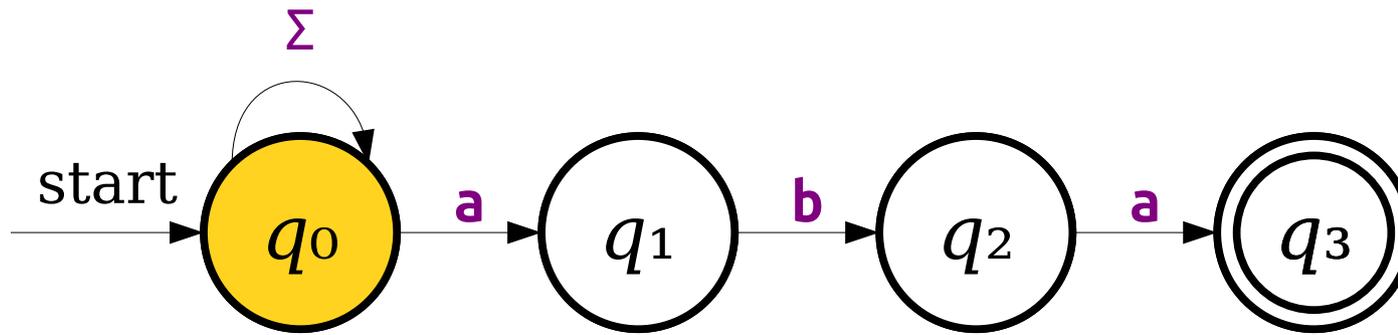
	$a$	$b$
$\{q_0\}$	$\{q_0, q_1\}$	



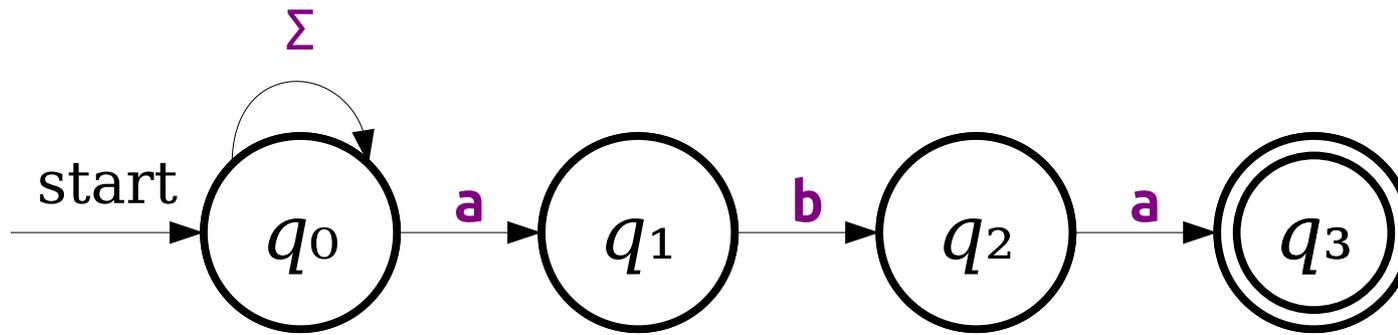
	$a$	$b$
$\{q_0\}$	$\{q_0, q_1\}$	



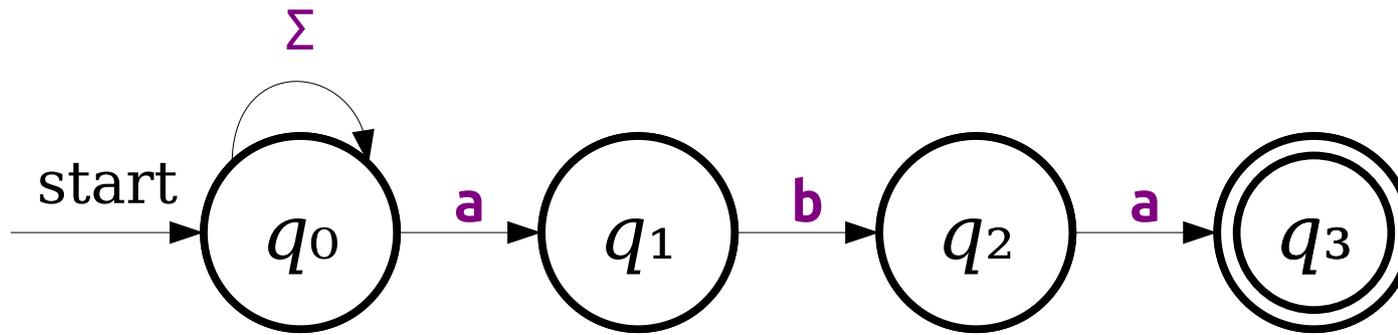
	$a$	$b$
$\{q_0\}$	$\{q_0, q_1\}$	$\{q_0\}$



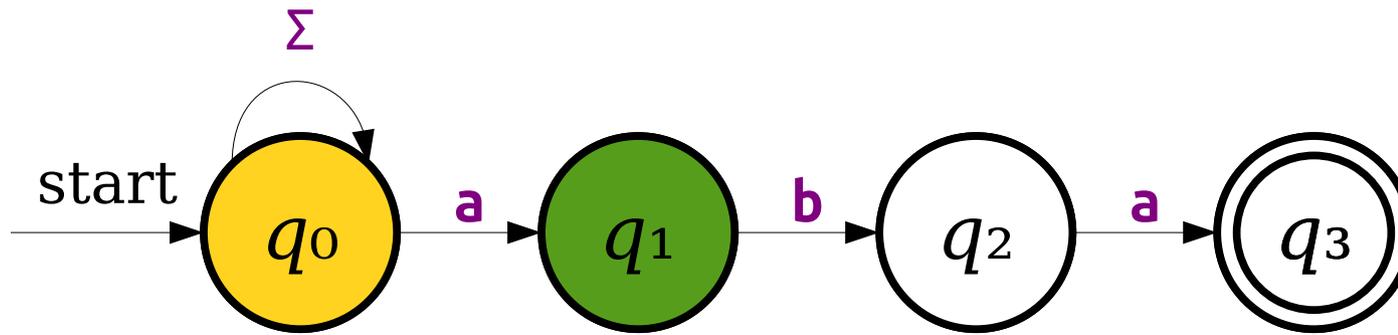
	$a$	$b$
$\{q_0\}$	$\{q_0, q_1\}$	$\{q_0\}$



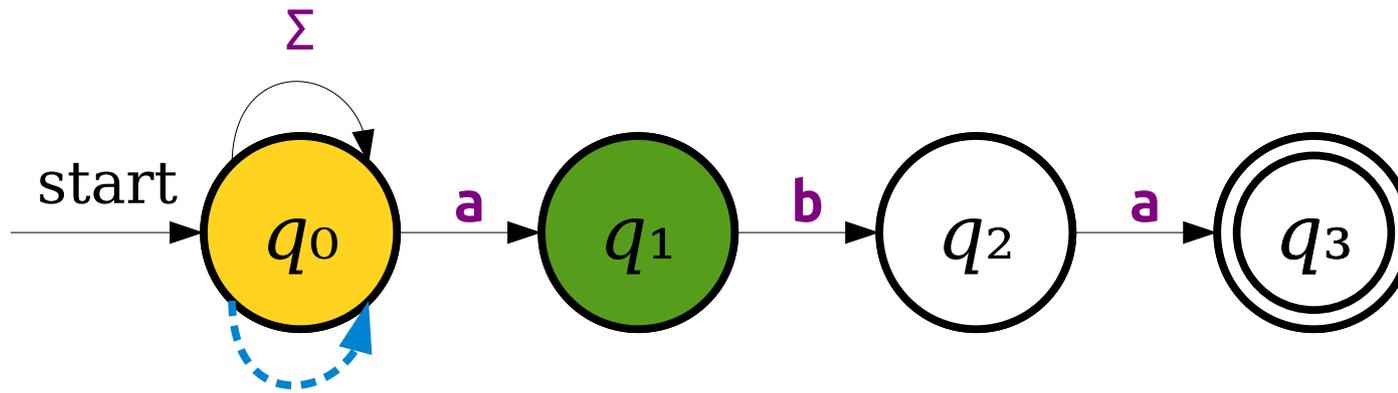
	$a$	$b$
$\{q_0\}$	$\{q_0, q_1\}$	$\{q_0\}$



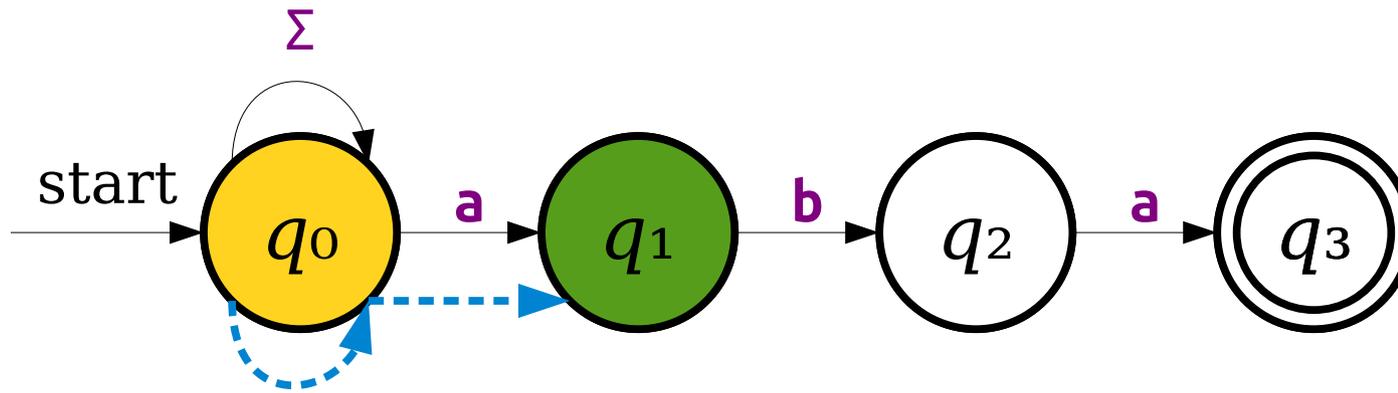
	$a$	$b$
$\{q_0\}$	$\{q_0, q_1\}$	$\{q_0\}$
$\{q_0, q_1\}$		



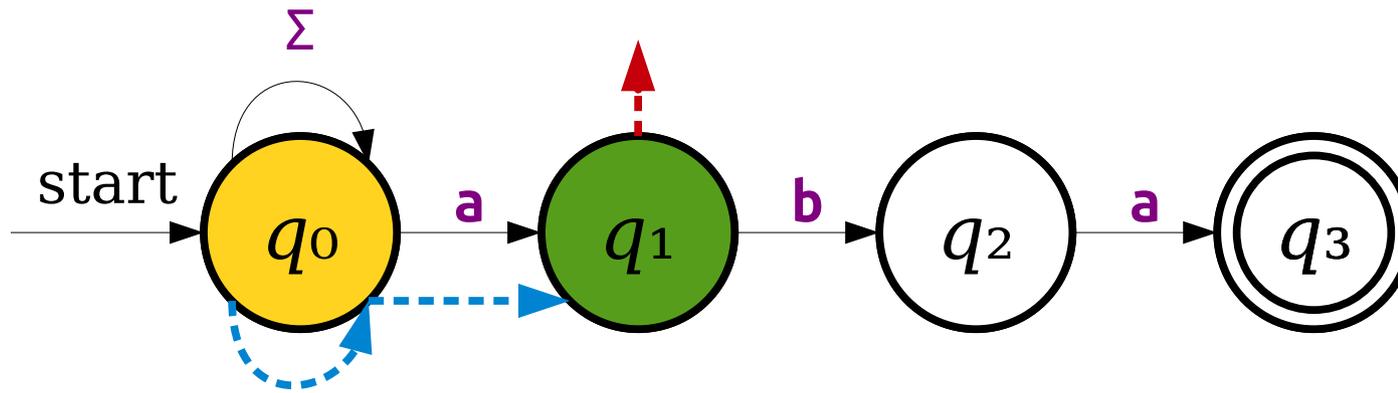
	$a$	$b$
$\{q_0\}$	$\{q_0, q_1\}$	$\{q_0\}$
$\{q_0, q_1\}$		



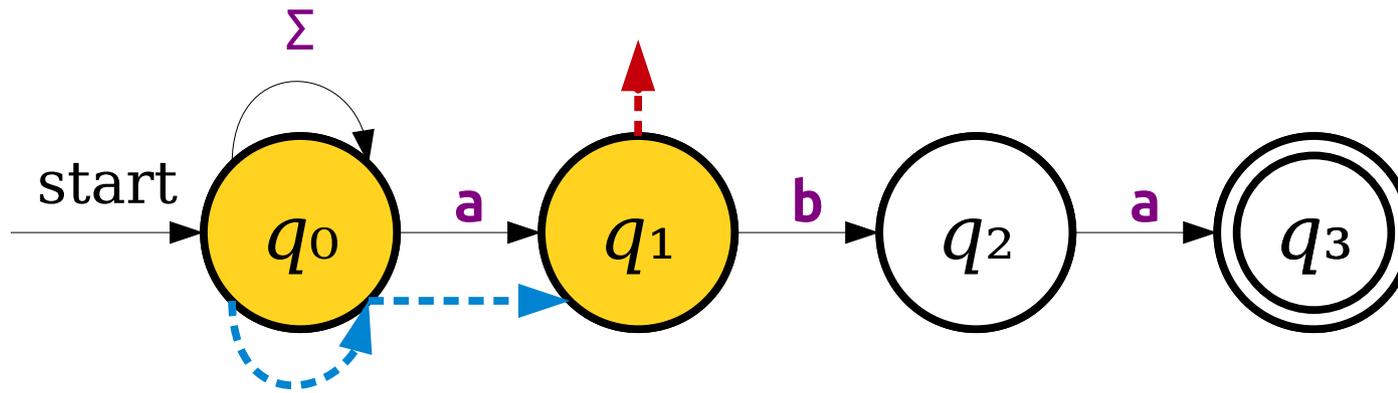
	$a$	$b$
$\{q_0\}$	$\{q_0, q_1\}$	$\{q_0\}$
$\{q_0, q_1\}$		



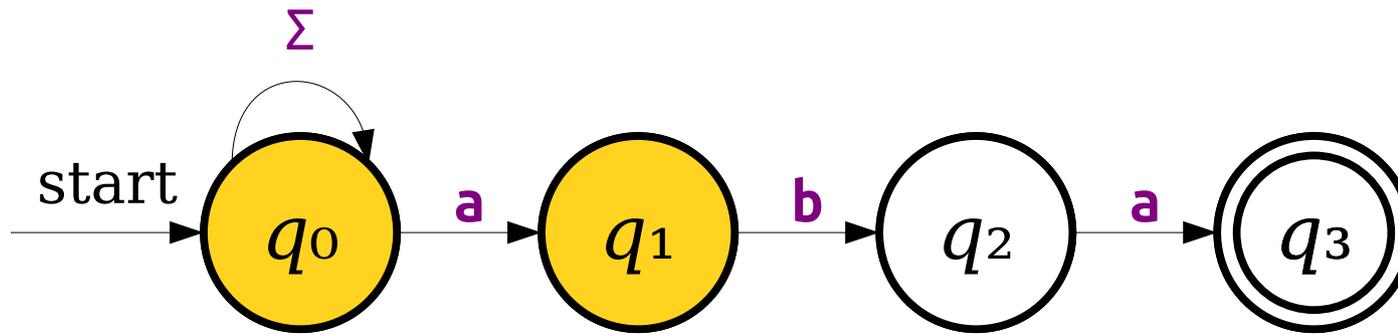
	$a$	$b$
$\{q_0\}$	$\{q_0, q_1\}$	$\{q_0\}$
$\{q_0, q_1\}$		



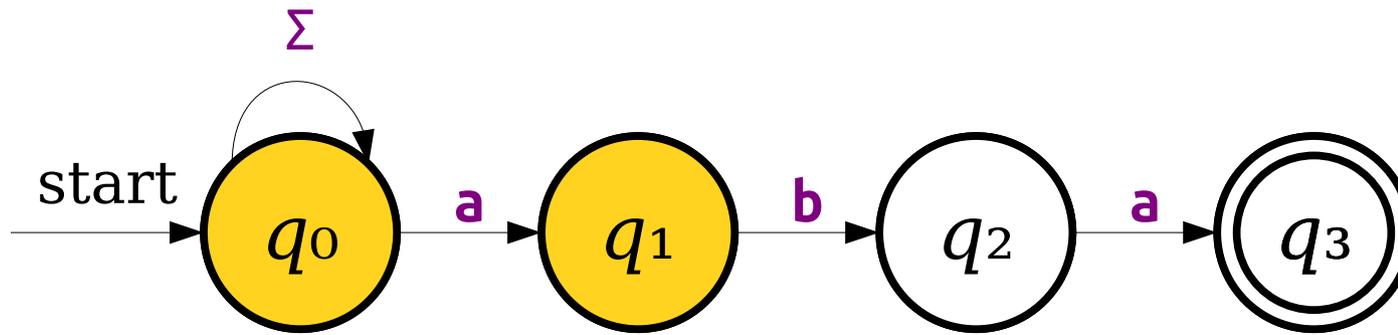
	$a$	$b$
$\{q_0\}$	$\{q_0, q_1\}$	$\{q_0\}$
$\{q_0, q_1\}$		



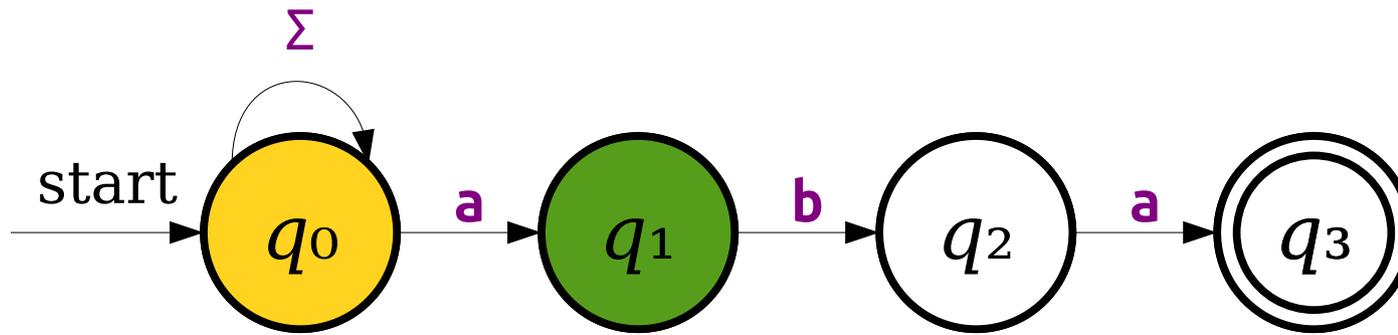
	$a$	$b$
$\{q_0\}$	$\{q_0, q_1\}$	$\{q_0\}$
$\{q_0, q_1\}$		



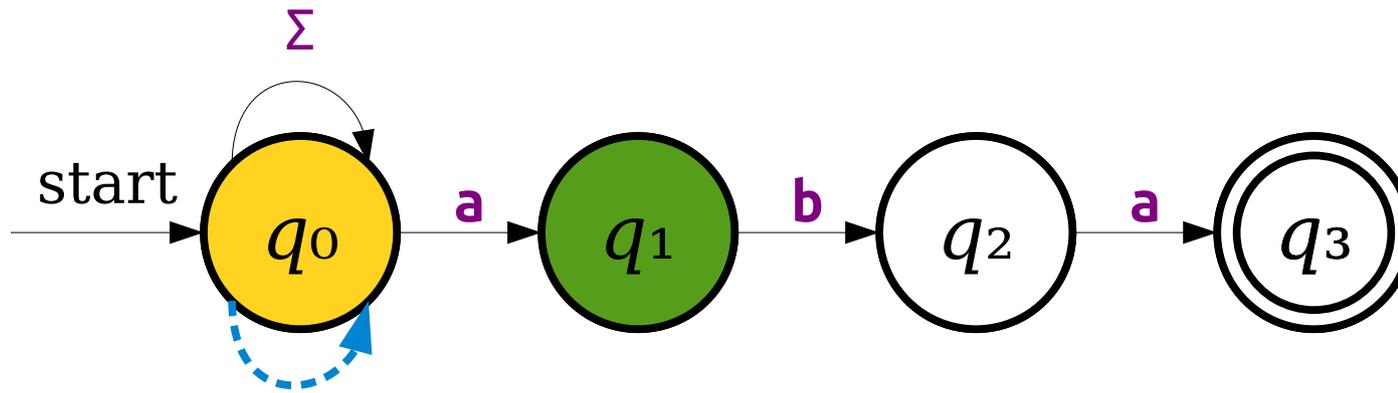
	$a$	$b$
$\{q_0\}$	$\{q_0, q_1\}$	$\{q_0\}$
$\{q_0, q_1\}$		



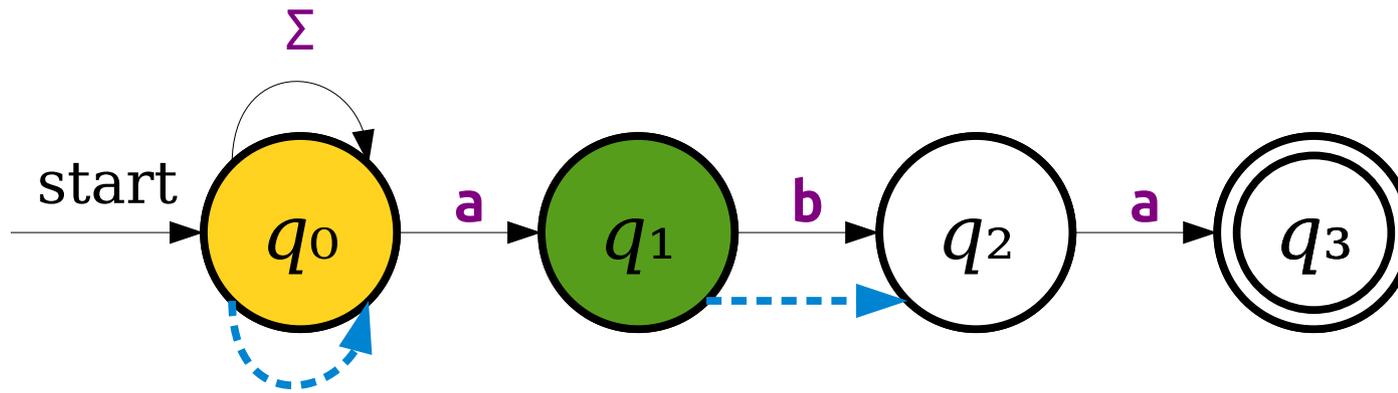
	$a$	$b$
$\{q_0\}$	$\{q_0, q_1\}$	$\{q_0\}$
$\{q_0, q_1\}$	$\{q_0, q_1\}$	



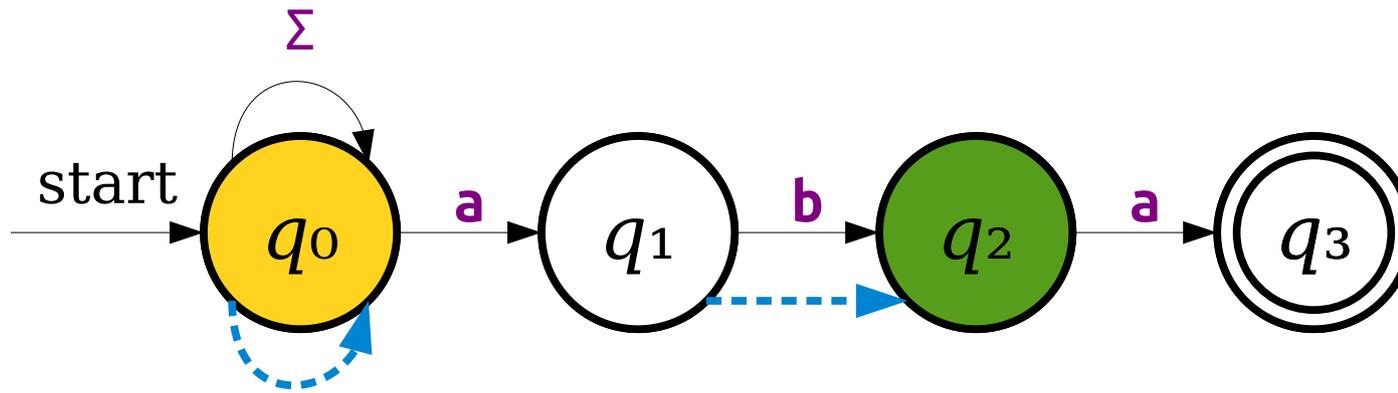
	$a$	$b$
$\{q_0\}$	$\{q_0, q_1\}$	$\{q_0\}$
$\{q_0, q_1\}$	$\{q_0, q_1\}$	



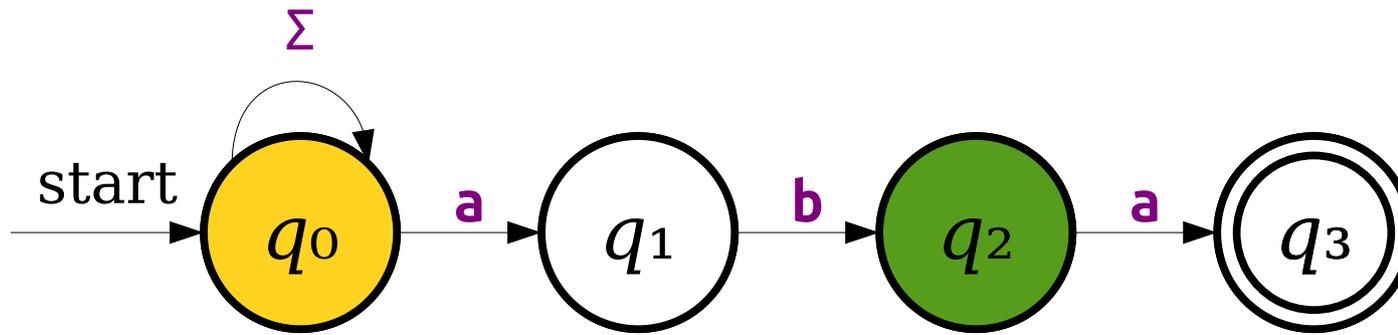
	$a$	$b$
$\{q_0\}$	$\{q_0, q_1\}$	$\{q_0\}$
$\{q_0, q_1\}$	$\{q_0, q_1\}$	



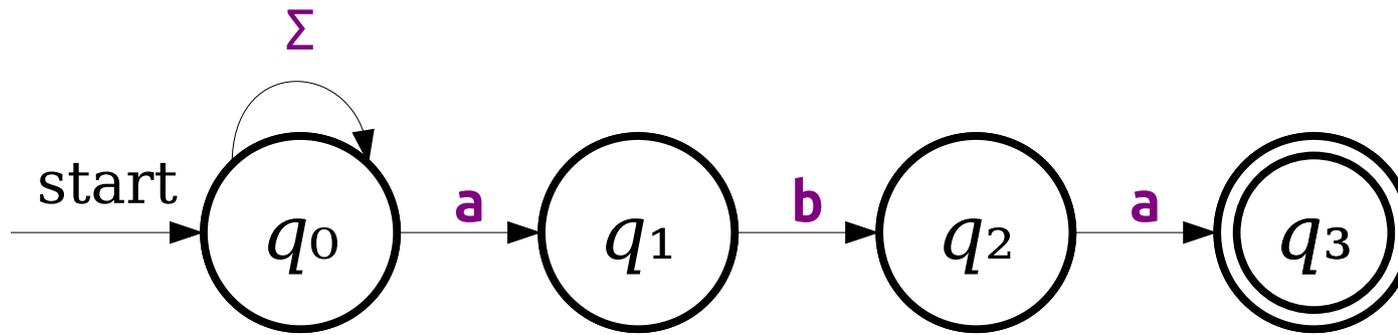
	$a$	$b$
$\{q_0\}$	$\{q_0, q_1\}$	$\{q_0\}$
$\{q_0, q_1\}$	$\{q_0, q_1\}$	



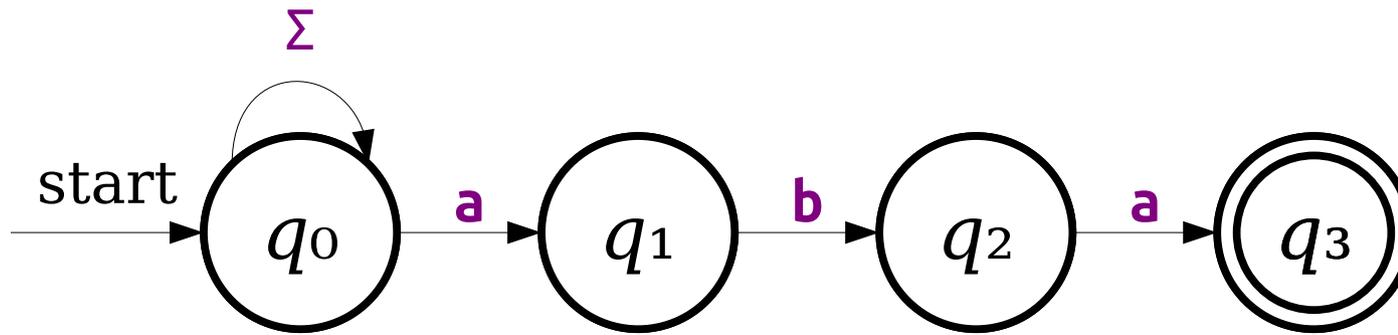
	$a$	$b$
$\{q_0\}$	$\{q_0, q_1\}$	$\{q_0\}$
$\{q_0, q_1\}$	$\{q_0, q_1\}$	



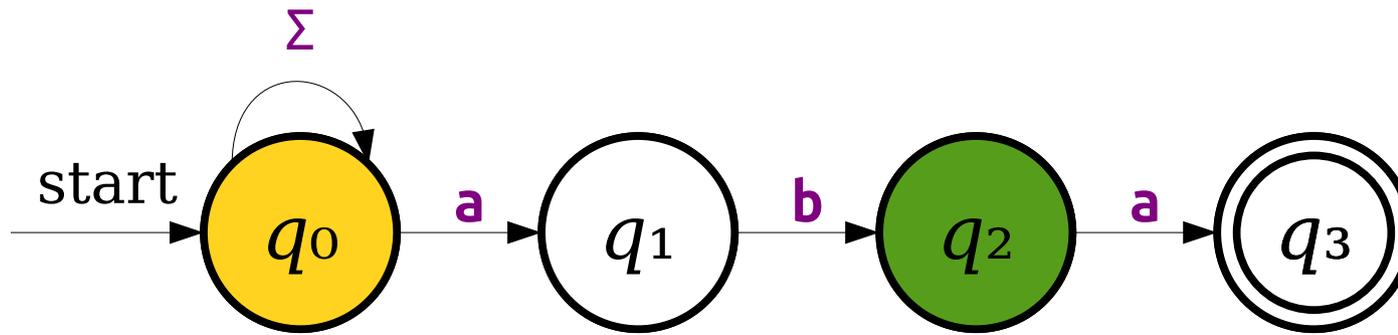
	<i>a</i>	<i>b</i>
$\{q_0\}$	$\{q_0, q_1\}$	$\{q_0\}$
$\{q_0, q_1\}$	$\{q_0, q_1\}$	$\{q_0, q_2\}$



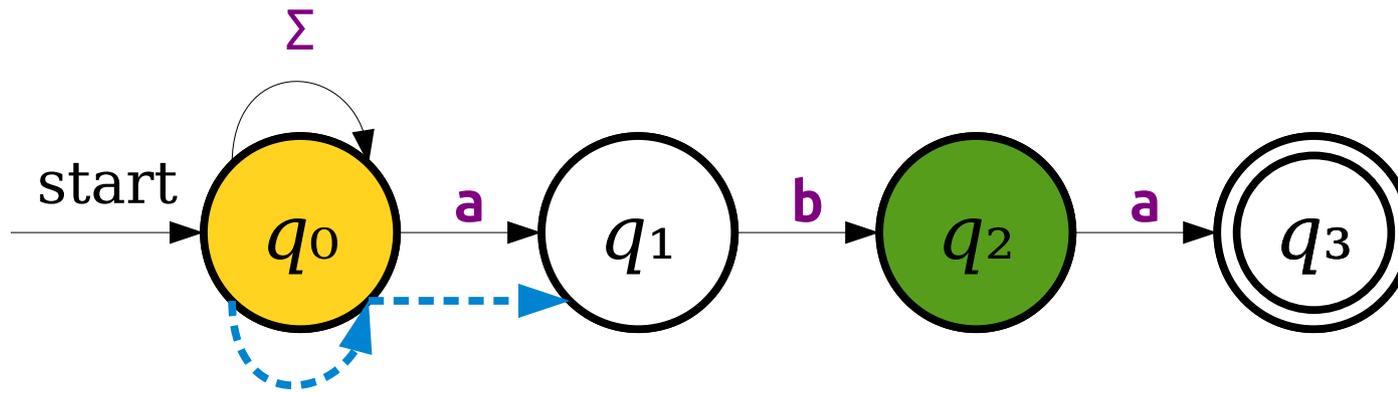
	$a$	$b$
$\{q_0\}$	$\{q_0, q_1\}$	$\{q_0\}$
$\{q_0, q_1\}$	$\{q_0, q_1\}$	$\{q_0, q_2\}$



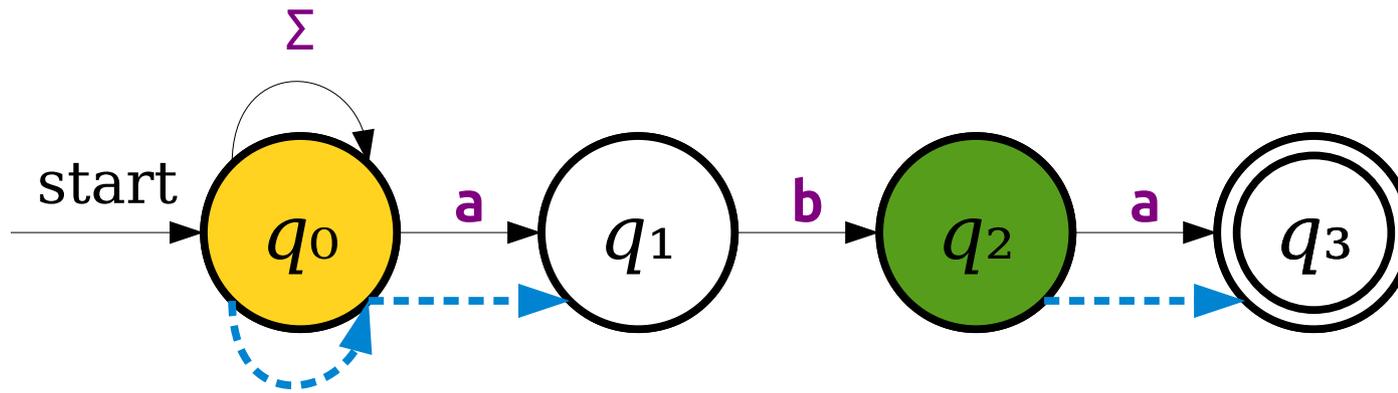
	$a$	$b$
$\{q_0\}$	$\{q_0, q_1\}$	$\{q_0\}$
$\{q_0, q_1\}$	$\{q_0, q_1\}$	$\{q_0, q_2\}$
$\{q_0, q_2\}$		



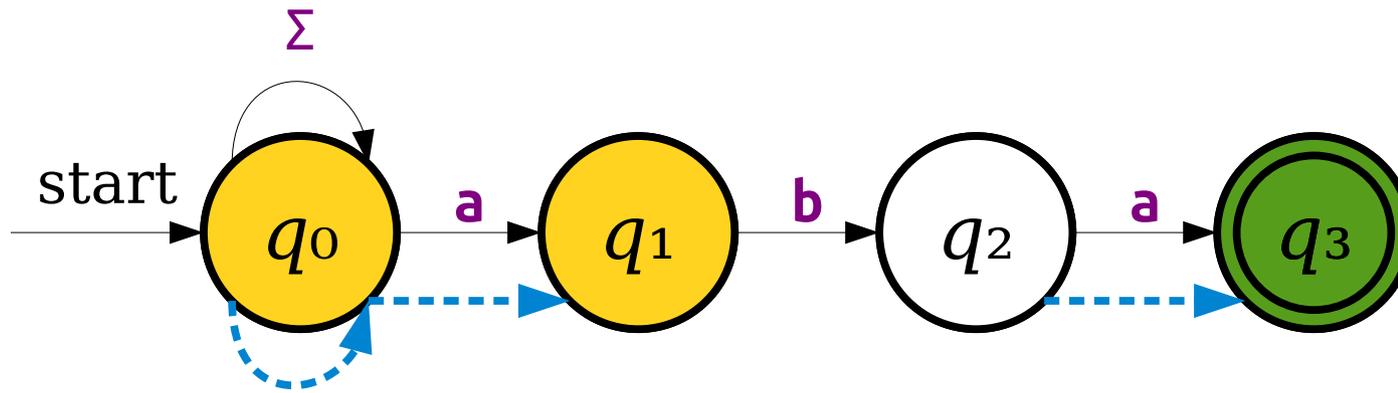
	$a$	$b$
$\{q_0\}$	$\{q_0, q_1\}$	$\{q_0\}$
$\{q_0, q_1\}$	$\{q_0, q_1\}$	$\{q_0, q_2\}$
$\{q_0, q_2\}$		



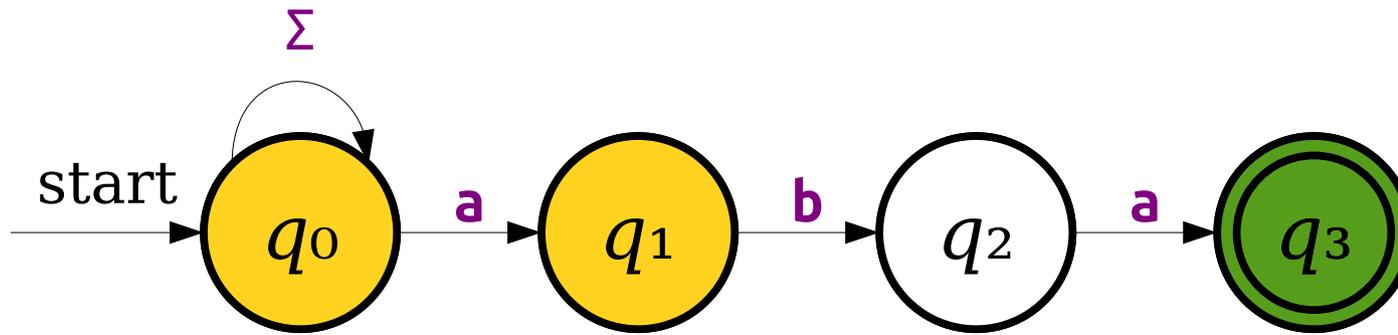
	$a$	$b$
$\{q_0\}$	$\{q_0, q_1\}$	$\{q_0\}$
$\{q_0, q_1\}$	$\{q_0, q_1\}$	$\{q_0, q_2\}$
$\{q_0, q_2\}$		



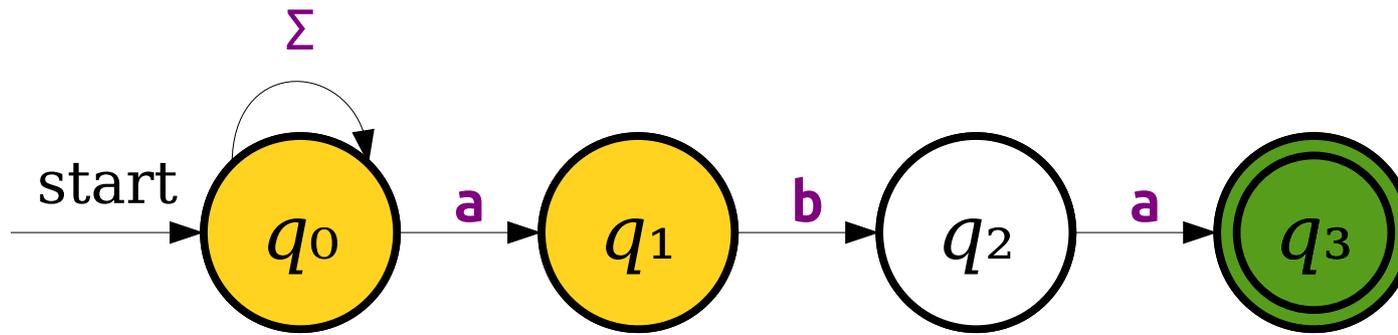
	$a$	$b$
$\{q_0\}$	$\{q_0, q_1\}$	$\{q_0\}$
$\{q_0, q_1\}$	$\{q_0, q_1\}$	$\{q_0, q_2\}$
$\{q_0, q_2\}$		



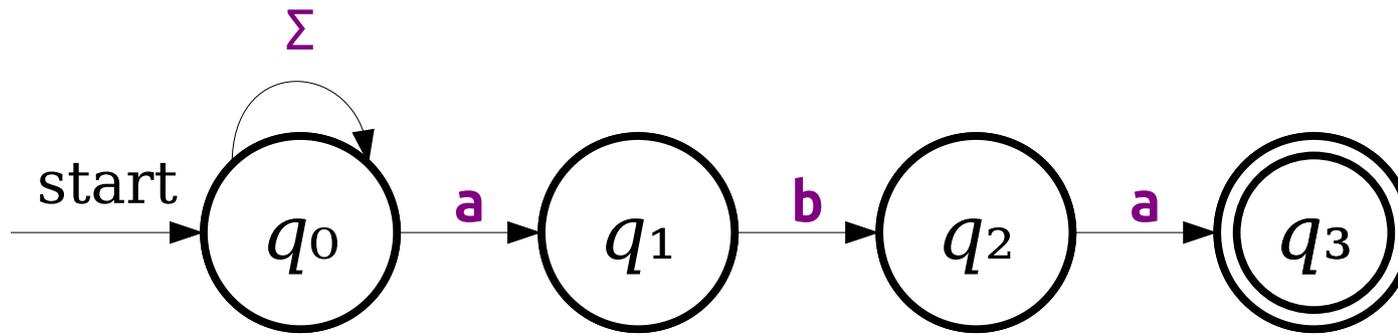
	$a$	$b$
$\{q_0\}$	$\{q_0, q_1\}$	$\{q_0\}$
$\{q_0, q_1\}$	$\{q_0, q_1\}$	$\{q_0, q_2\}$
$\{q_0, q_2\}$		



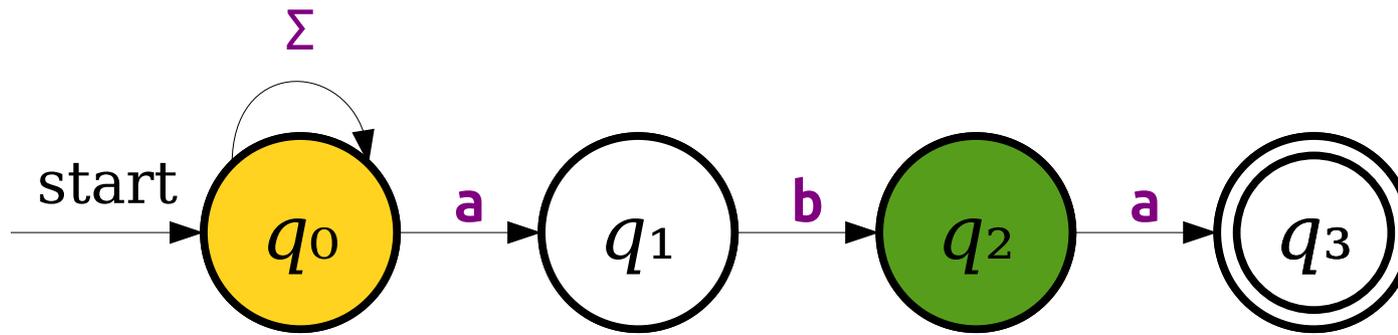
	$a$	$b$
$\{q_0\}$	$\{q_0, q_1\}$	$\{q_0\}$
$\{q_0, q_1\}$	$\{q_0, q_1\}$	$\{q_0, q_2\}$
$\{q_0, q_2\}$		



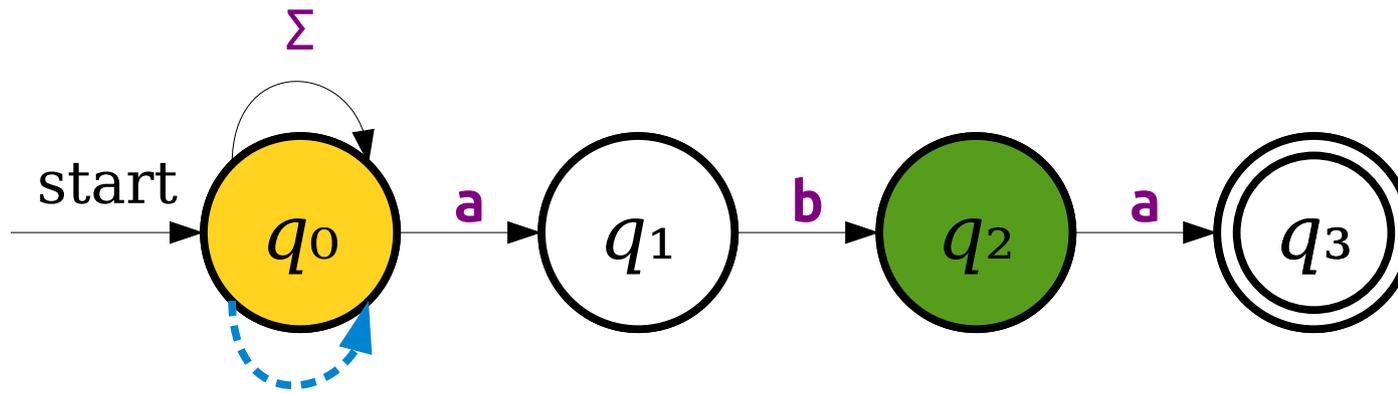
	$a$	$b$
$\{q_0\}$	$\{q_0, q_1\}$	$\{q_0\}$
$\{q_0, q_1\}$	$\{q_0, q_1\}$	$\{q_0, q_2\}$
$\{q_0, q_2\}$	$\{q_0, q_1, q_3\}$	



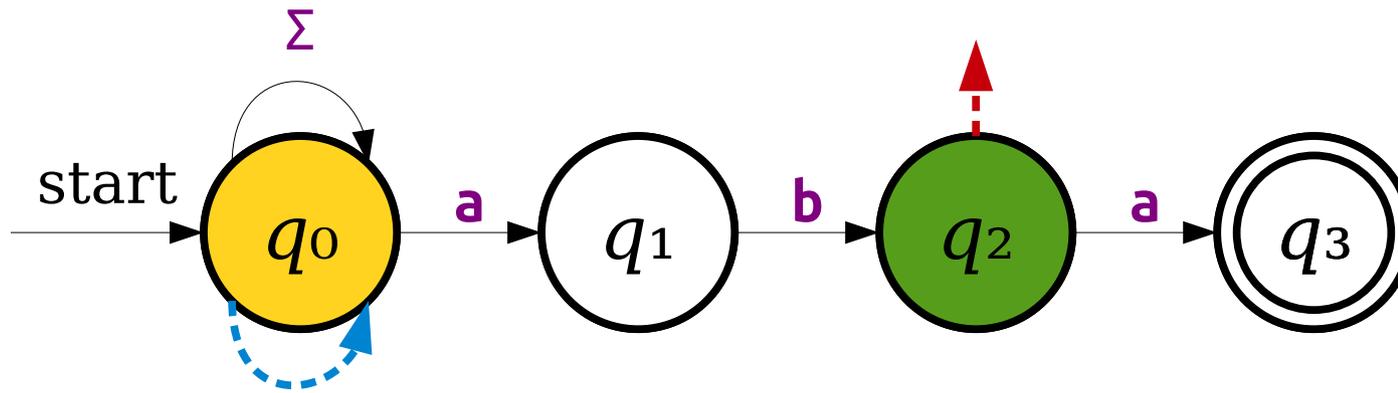
	$a$	$b$
$\{q_0\}$	$\{q_0, q_1\}$	$\{q_0\}$
$\{q_0, q_1\}$	$\{q_0, q_1\}$	$\{q_0, q_2\}$
$\{q_0, q_2\}$	$\{q_0, q_1, q_3\}$	



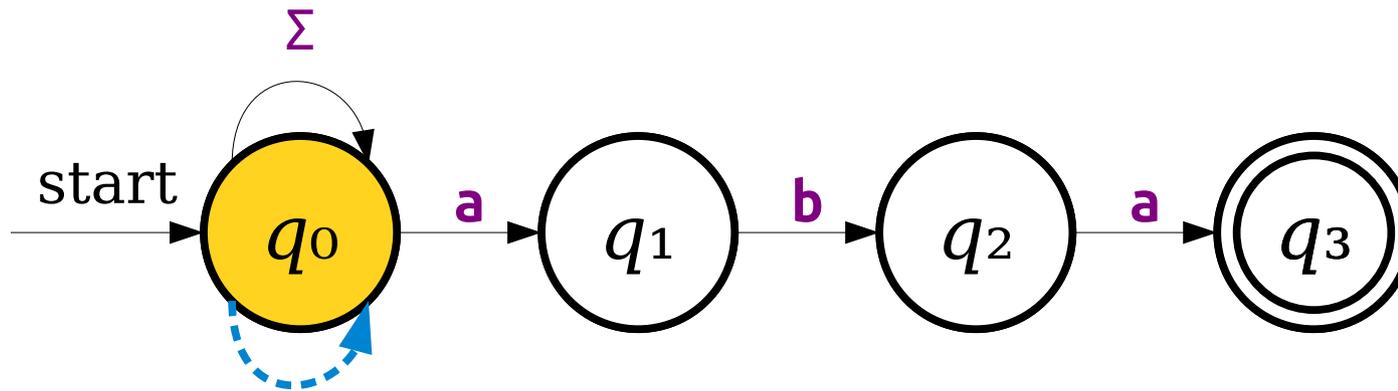
	$a$	$b$
$\{q_0\}$	$\{q_0, q_1\}$	$\{q_0\}$
$\{q_0, q_1\}$	$\{q_0, q_1\}$	$\{q_0, q_2\}$
$\{q_0, q_2\}$	$\{q_0, q_1, q_3\}$	



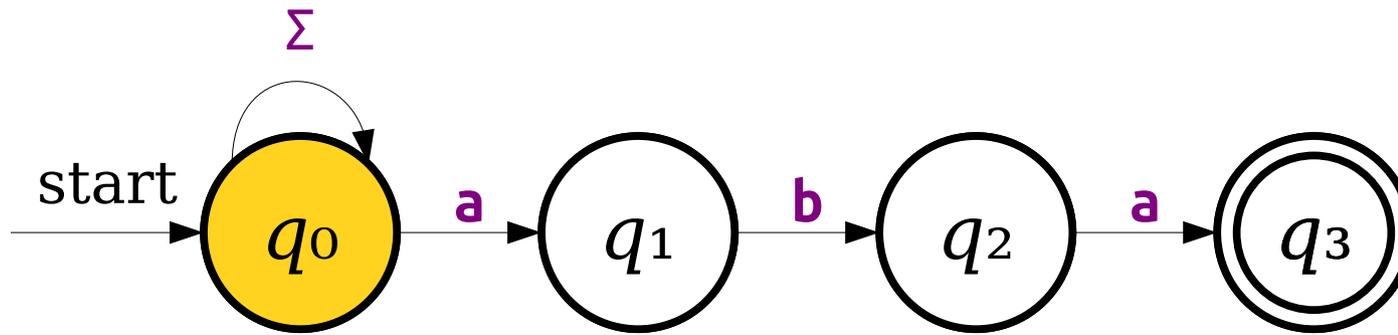
	$a$	$b$
$\{q_0\}$	$\{q_0, q_1\}$	$\{q_0\}$
$\{q_0, q_1\}$	$\{q_0, q_1\}$	$\{q_0, q_2\}$
$\{q_0, q_2\}$	$\{q_0, q_1, q_3\}$	



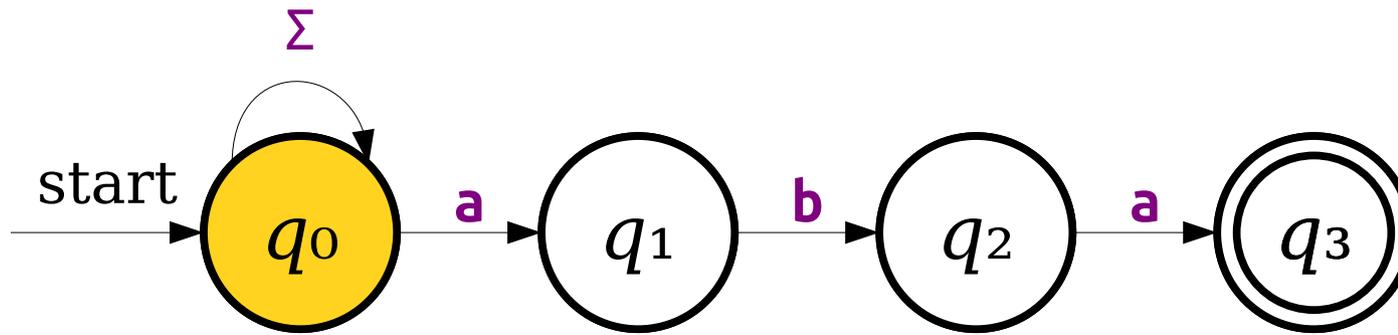
	$a$	$b$
$\{q_0\}$	$\{q_0, q_1\}$	$\{q_0\}$
$\{q_0, q_1\}$	$\{q_0, q_1\}$	$\{q_0, q_2\}$
$\{q_0, q_2\}$	$\{q_0, q_1, q_3\}$	



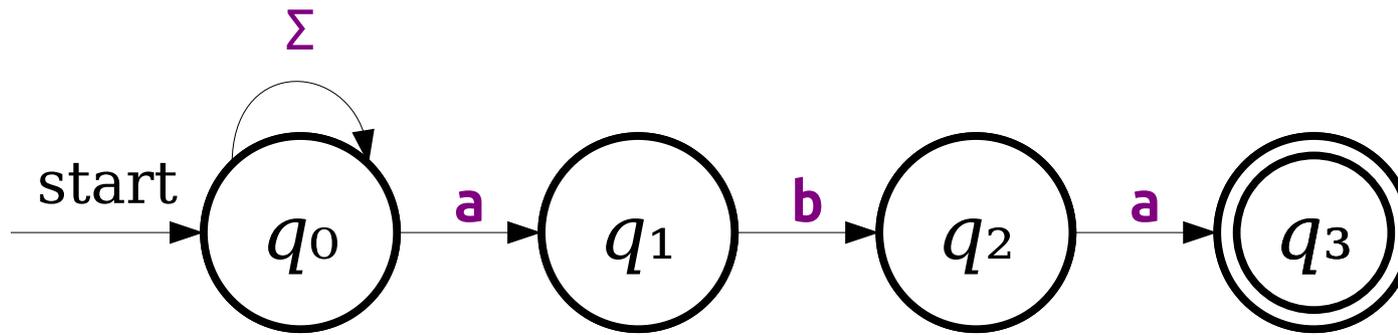
	$a$	$b$
$\{q_0\}$	$\{q_0, q_1\}$	$\{q_0\}$
$\{q_0, q_1\}$	$\{q_0, q_1\}$	$\{q_0, q_2\}$
$\{q_0, q_2\}$	$\{q_0, q_1, q_3\}$	



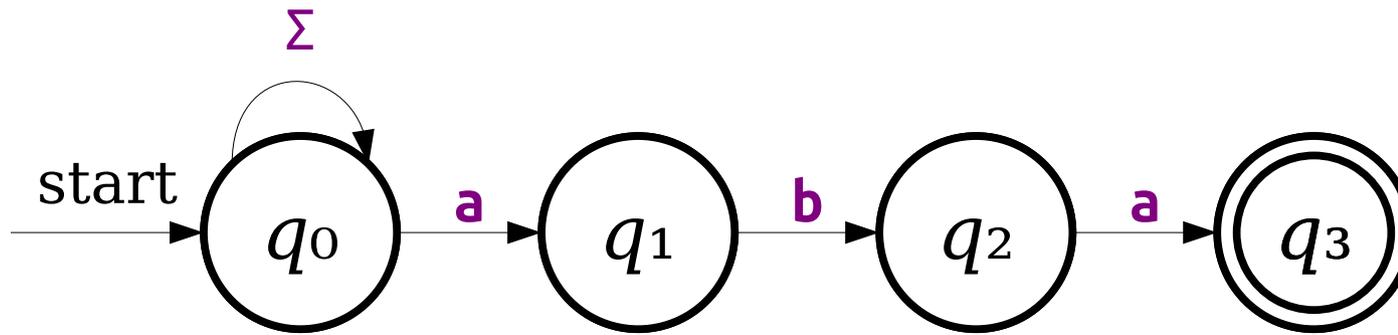
	$a$	$b$
$\{q_0\}$	$\{q_0, q_1\}$	$\{q_0\}$
$\{q_0, q_1\}$	$\{q_0, q_1\}$	$\{q_0, q_2\}$
$\{q_0, q_2\}$	$\{q_0, q_1, q_3\}$	



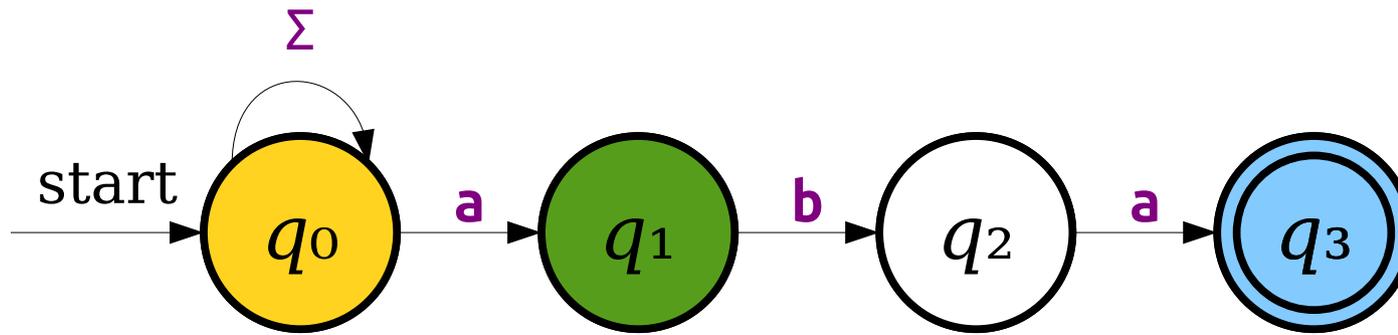
	$a$	$b$
$\{q_0\}$	$\{q_0, q_1\}$	$\{q_0\}$
$\{q_0, q_1\}$	$\{q_0, q_1\}$	$\{q_0, q_2\}$
$\{q_0, q_2\}$	$\{q_0, q_1, q_3\}$	$\{q_0\}$



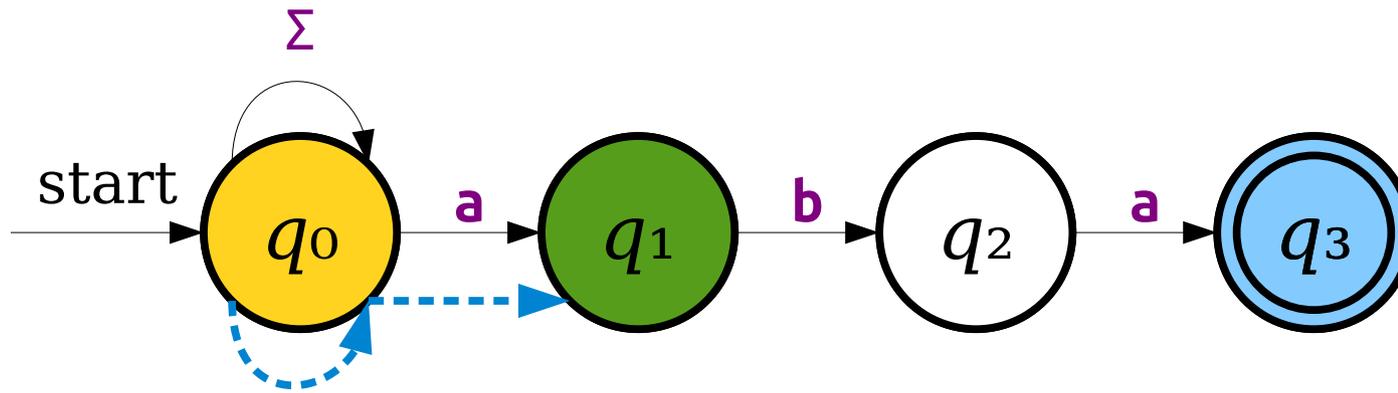
	<i>a</i>	<i>b</i>
$\{q_0\}$	$\{q_0, q_1\}$	$\{q_0\}$
$\{q_0, q_1\}$	$\{q_0, q_1\}$	$\{q_0, q_2\}$
$\{q_0, q_2\}$	$\{q_0, q_1, q_3\}$	$\{q_0\}$



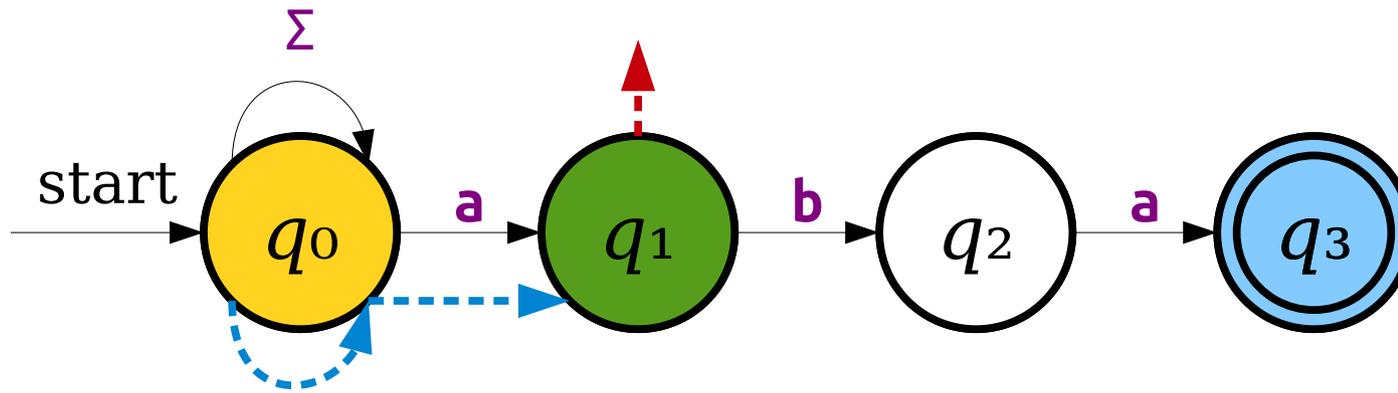
	<i>a</i>	<i>b</i>
$\{q_0\}$	$\{q_0, q_1\}$	$\{q_0\}$
$\{q_0, q_1\}$	$\{q_0, q_1\}$	$\{q_0, q_2\}$
$\{q_0, q_2\}$	$\{q_0, q_1, q_3\}$	$\{q_0\}$
$\{q_0, q_1, q_3\}$		



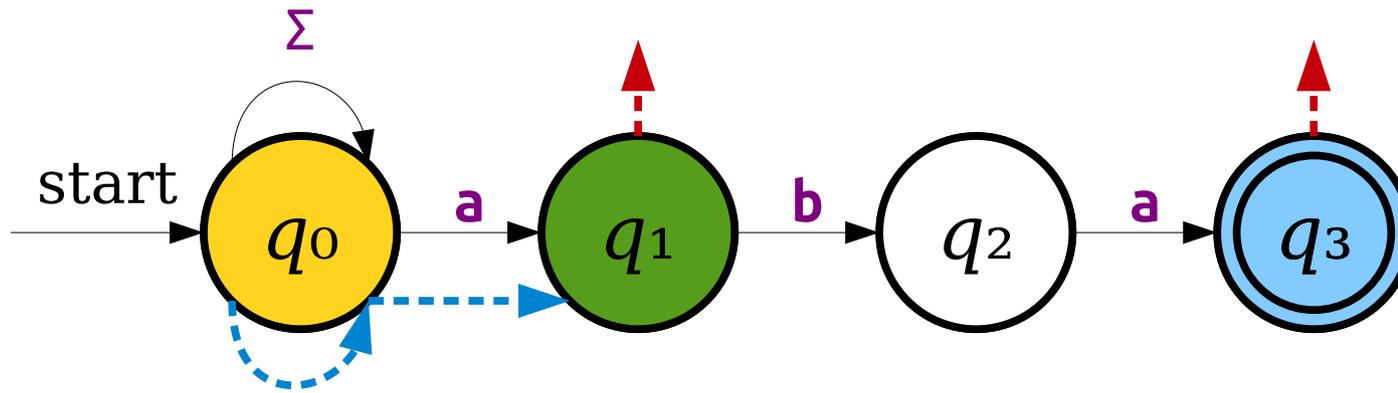
	$a$	$b$
$\{q_0\}$	$\{q_0, q_1\}$	$\{q_0\}$
$\{q_0, q_1\}$	$\{q_0, q_1\}$	$\{q_0, q_2\}$
$\{q_0, q_2\}$	$\{q_0, q_1, q_3\}$	$\{q_0\}$
$\{q_0, q_1, q_3\}$		



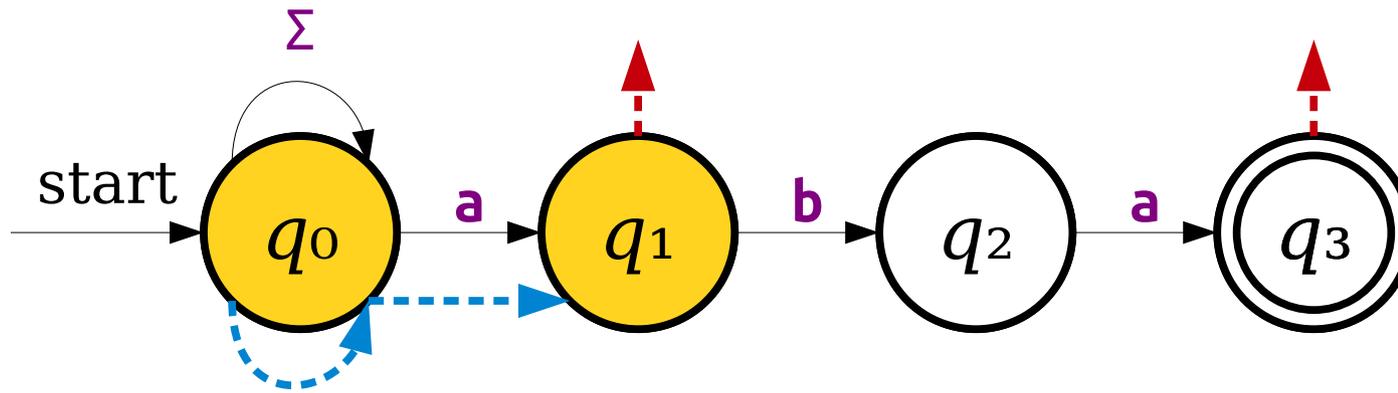
	$a$	$b$
$\{q_0\}$	$\{q_0, q_1\}$	$\{q_0\}$
$\{q_0, q_1\}$	$\{q_0, q_1\}$	$\{q_0, q_2\}$
$\{q_0, q_2\}$	$\{q_0, q_1, q_3\}$	$\{q_0\}$
$\{q_0, q_1, q_3\}$		



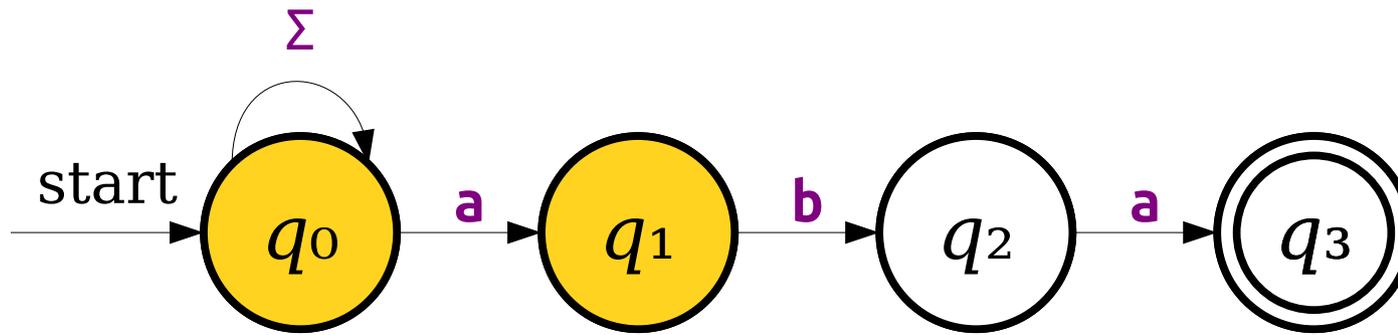
	$a$	$b$
$\{q_0\}$	$\{q_0, q_1\}$	$\{q_0\}$
$\{q_0, q_1\}$	$\{q_0, q_1\}$	$\{q_0, q_2\}$
$\{q_0, q_2\}$	$\{q_0, q_1, q_3\}$	$\{q_0\}$
$\{q_0, q_1, q_3\}$		



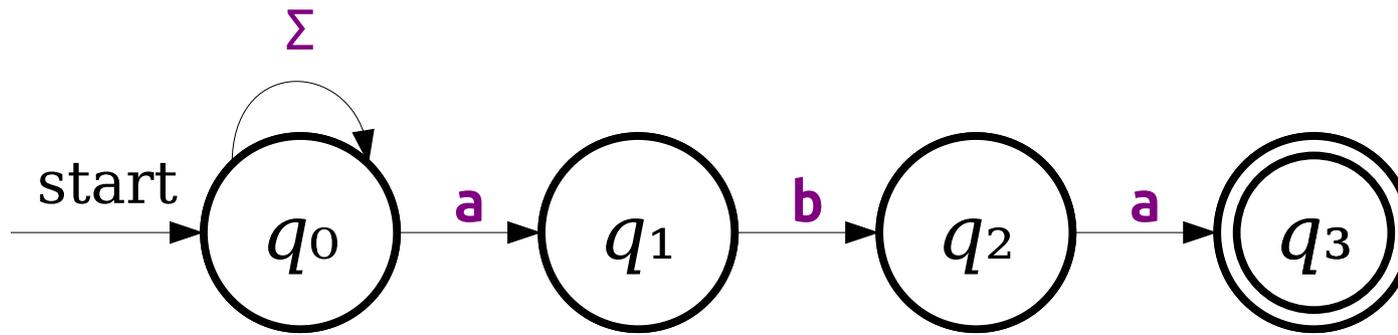
	<i>a</i>	<i>b</i>
$\{q_0\}$	$\{q_0, q_1\}$	$\{q_0\}$
$\{q_0, q_1\}$	$\{q_0, q_1\}$	$\{q_0, q_2\}$
$\{q_0, q_2\}$	$\{q_0, q_1, q_3\}$	$\{q_0\}$
$\{q_0, q_1, q_3\}$		



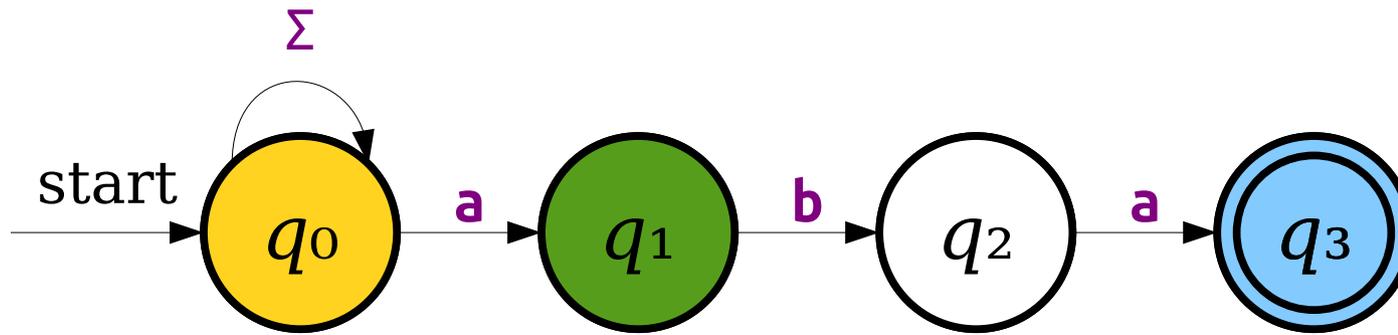
	<i>a</i>	<i>b</i>
$\{q_0\}$	$\{q_0, q_1\}$	$\{q_0\}$
$\{q_0, q_1\}$	$\{q_0, q_1\}$	$\{q_0, q_2\}$
$\{q_0, q_2\}$	$\{q_0, q_1, q_3\}$	$\{q_0\}$
$\{q_0, q_1, q_3\}$		



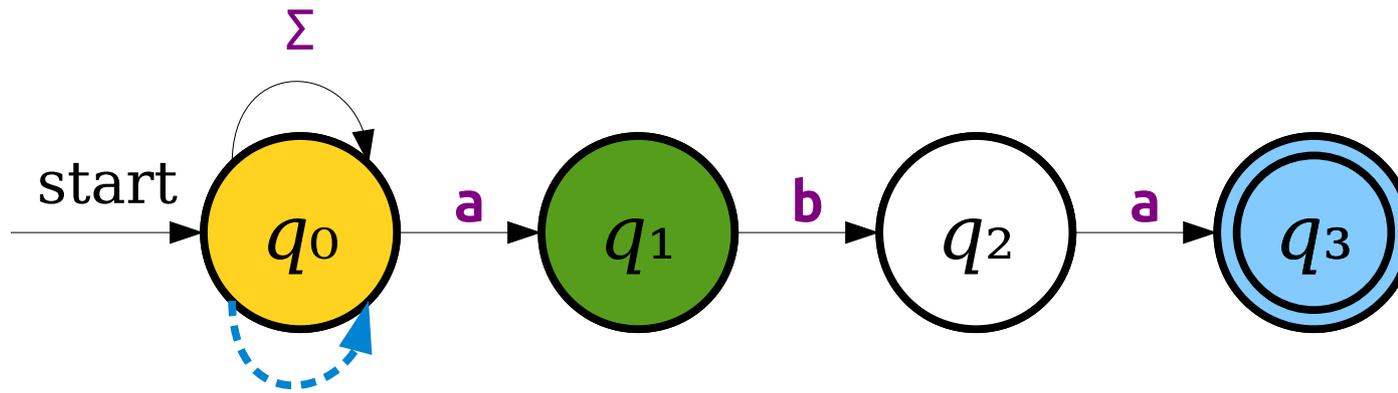
	$a$	$b$
$\{q_0\}$	$\{q_0, q_1\}$	$\{q_0\}$
$\{q_0, q_1\}$	$\{q_0, q_1\}$	$\{q_0, q_2\}$
$\{q_0, q_2\}$	$\{q_0, q_1, q_3\}$	$\{q_0\}$
$\{q_0, q_1, q_3\}$	$\{q_0, q_1\}$	



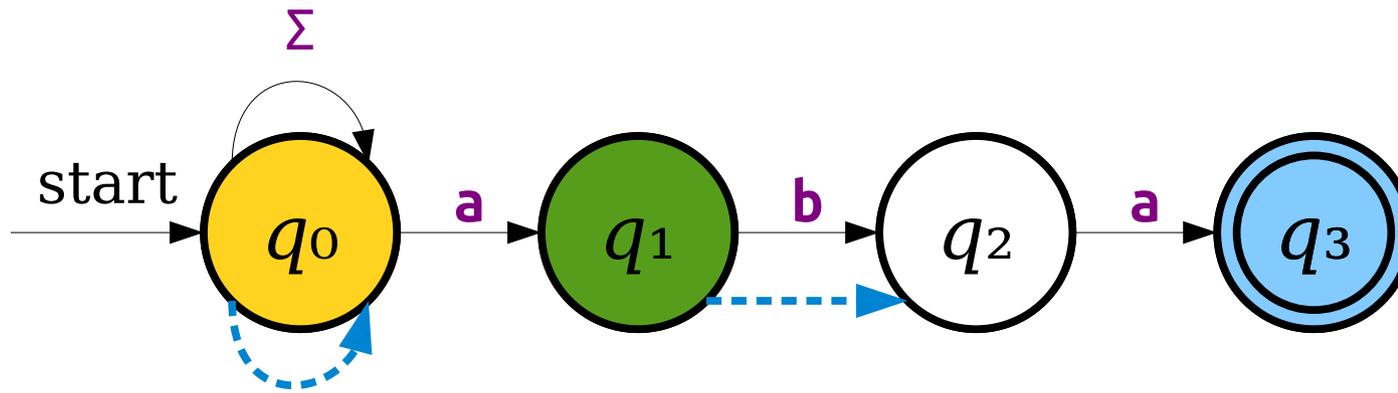
	$a$	$b$
$\{q_0\}$	$\{q_0, q_1\}$	$\{q_0\}$
$\{q_0, q_1\}$	$\{q_0, q_1\}$	$\{q_0, q_2\}$
$\{q_0, q_2\}$	$\{q_0, q_1, q_3\}$	$\{q_0\}$
$\{q_0, q_1, q_3\}$	$\{q_0, q_1\}$	



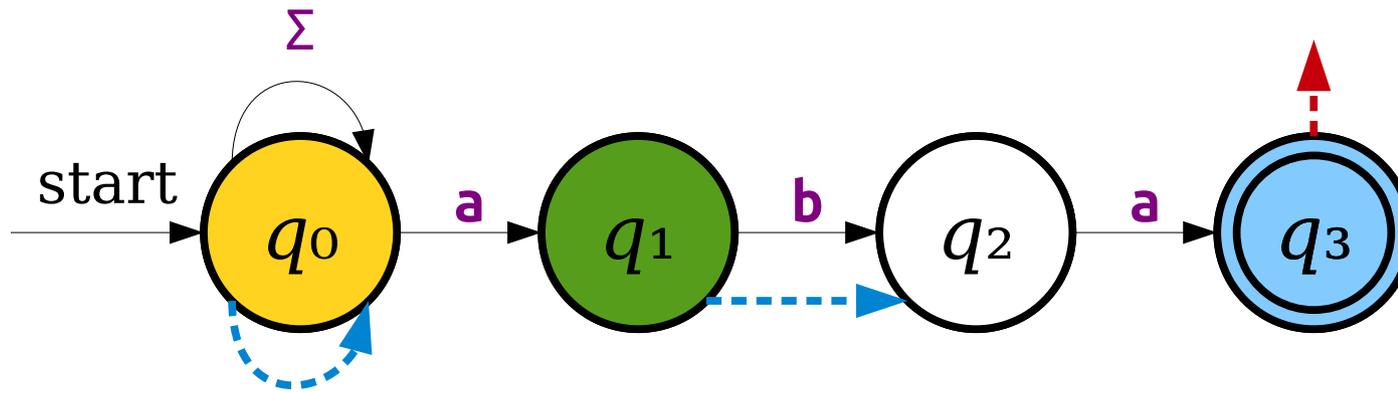
	<i>a</i>	<i>b</i>
$\{q_0\}$	$\{q_0, q_1\}$	$\{q_0\}$
$\{q_0, q_1\}$	$\{q_0, q_1\}$	$\{q_0, q_2\}$
$\{q_0, q_2\}$	$\{q_0, q_1, q_3\}$	$\{q_0\}$
$\{q_0, q_1, q_3\}$	$\{q_0, q_1\}$	



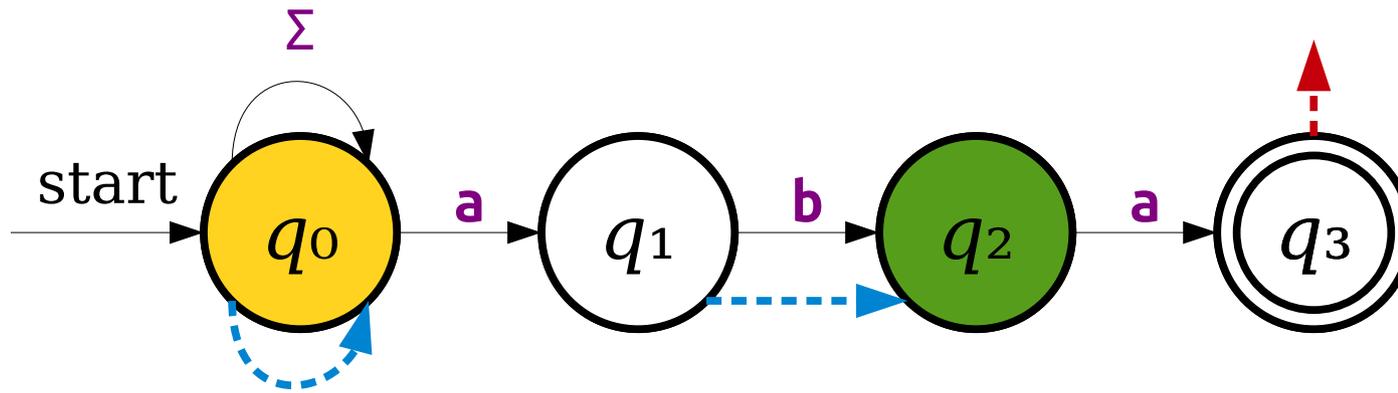
	$a$	$b$
$\{q_0\}$	$\{q_0, q_1\}$	$\{q_0\}$
$\{q_0, q_1\}$	$\{q_0, q_1\}$	$\{q_0, q_2\}$
$\{q_0, q_2\}$	$\{q_0, q_1, q_3\}$	$\{q_0\}$
$\{q_0, q_1, q_3\}$	$\{q_0, q_1\}$	



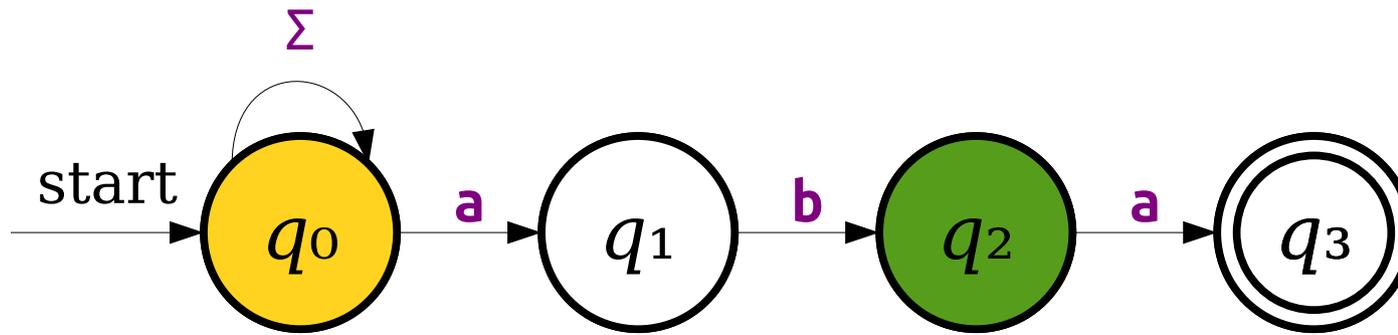
	$a$	$b$
$\{q_0\}$	$\{q_0, q_1\}$	$\{q_0\}$
$\{q_0, q_1\}$	$\{q_0, q_1\}$	$\{q_0, q_2\}$
$\{q_0, q_2\}$	$\{q_0, q_1, q_3\}$	$\{q_0\}$
$\{q_0, q_1, q_3\}$	$\{q_0, q_1\}$	



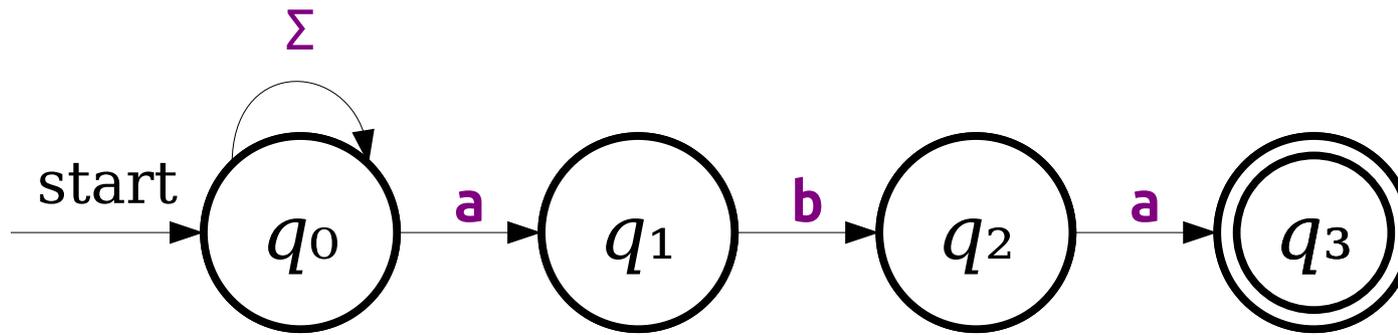
	$a$	$b$
$\{q_0\}$	$\{q_0, q_1\}$	$\{q_0\}$
$\{q_0, q_1\}$	$\{q_0, q_1\}$	$\{q_0, q_2\}$
$\{q_0, q_2\}$	$\{q_0, q_1, q_3\}$	$\{q_0\}$
$\{q_0, q_1, q_3\}$	$\{q_0, q_1\}$	



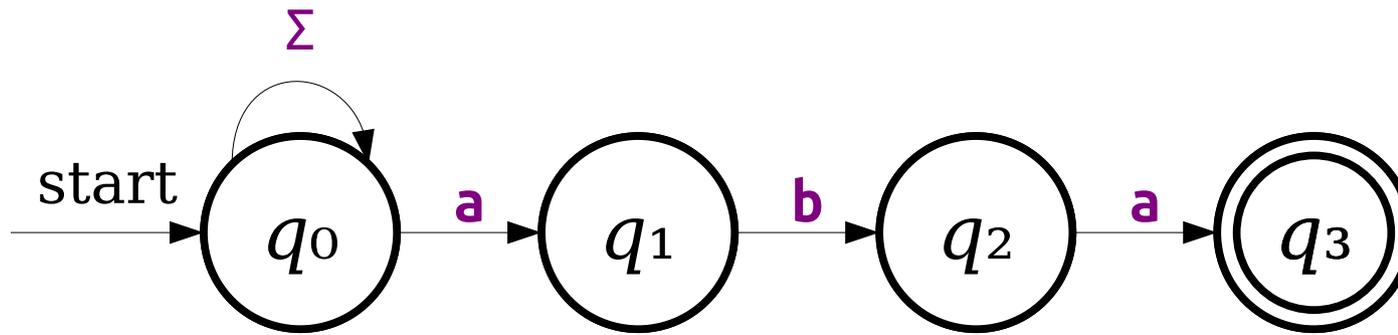
	$a$	$b$
$\{q_0\}$	$\{q_0, q_1\}$	$\{q_0\}$
$\{q_0, q_1\}$	$\{q_0, q_1\}$	$\{q_0, q_2\}$
$\{q_0, q_2\}$	$\{q_0, q_1, q_3\}$	$\{q_0\}$
$\{q_0, q_1, q_3\}$	$\{q_0, q_1\}$	



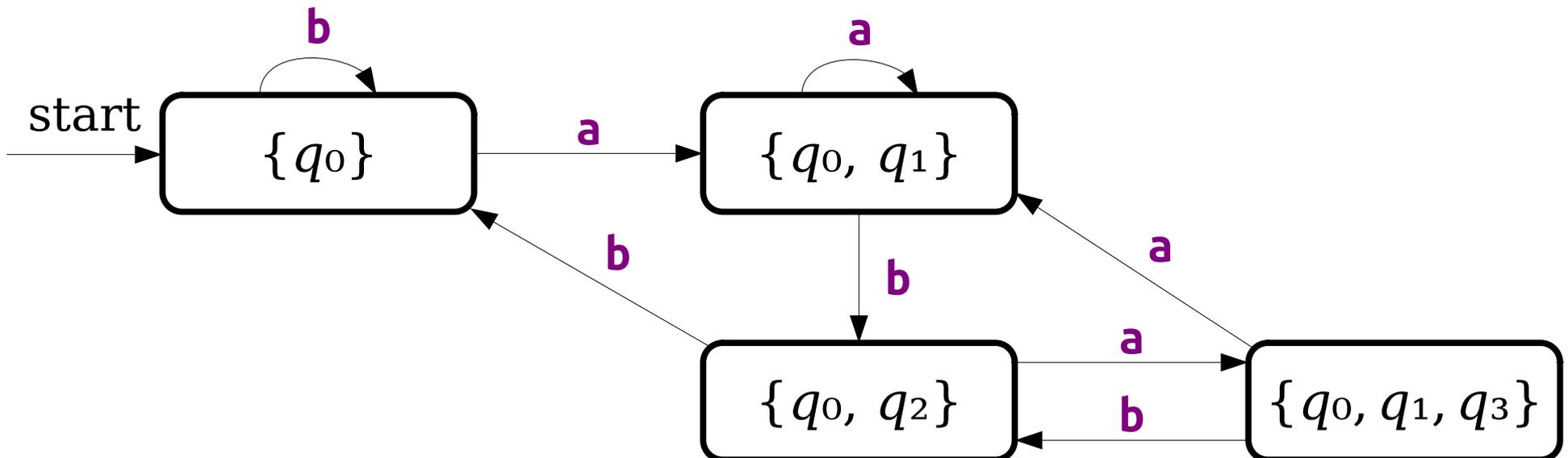
	$a$	$b$
$\{q_0\}$	$\{q_0, q_1\}$	$\{q_0\}$
$\{q_0, q_1\}$	$\{q_0, q_1\}$	$\{q_0, q_2\}$
$\{q_0, q_2\}$	$\{q_0, q_1, q_3\}$	$\{q_0\}$
$\{q_0, q_1, q_3\}$	$\{q_0, q_1\}$	$\{q_0, q_2\}$

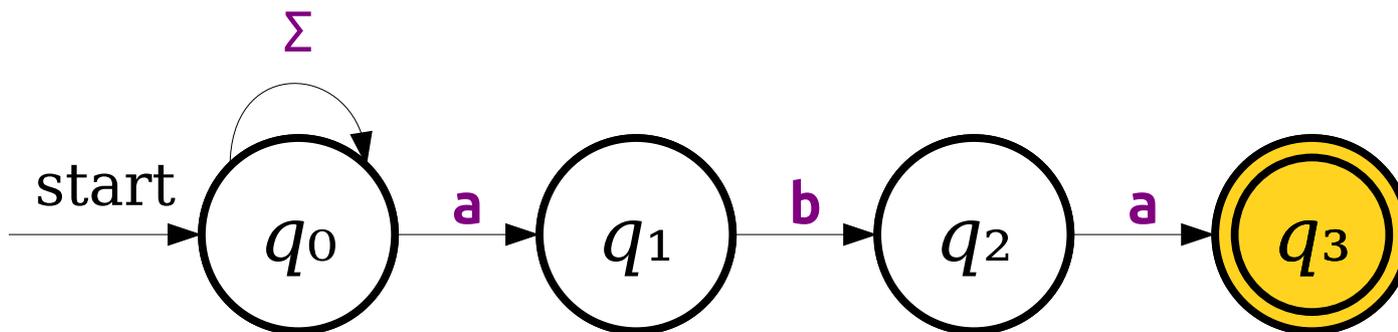


	$a$	$b$
$\{q_0\}$	$\{q_0, q_1\}$	$\{q_0\}$
$\{q_0, q_1\}$	$\{q_0, q_1\}$	$\{q_0, q_2\}$
$\{q_0, q_2\}$	$\{q_0, q_1, q_3\}$	$\{q_0\}$
$\{q_0, q_1, q_3\}$	$\{q_0, q_1\}$	$\{q_0, q_2\}$

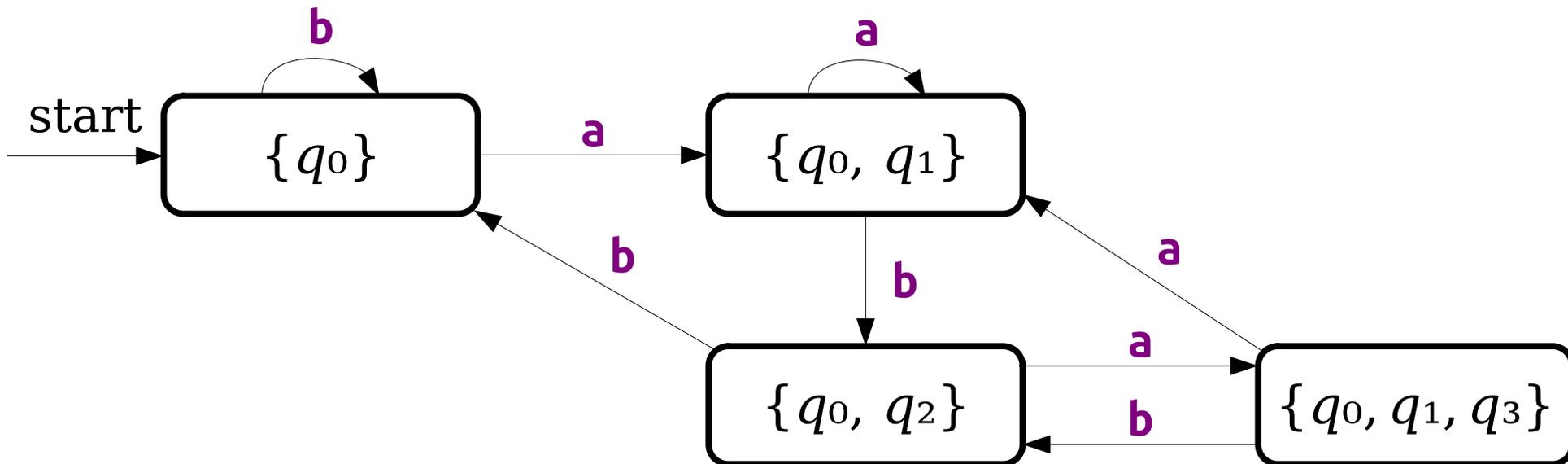


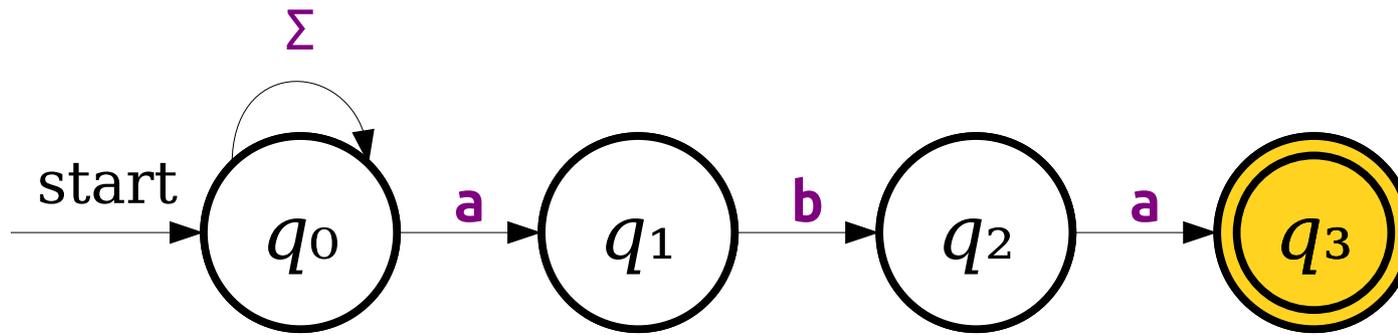
	<i>a</i>	<i>b</i>
$\{q_0\}$	$\{q_0, q_1\}$	$\{q_0\}$
$\{q_0, q_1\}$	$\{q_0, q_1\}$	$\{q_0, q_2\}$
$\{q_0, q_2\}$	$\{q_0, q_1, q_3\}$	$\{q_0\}$
$\{q_0, q_1, q_3\}$	$\{q_0, q_1\}$	$\{q_0, q_2\}$



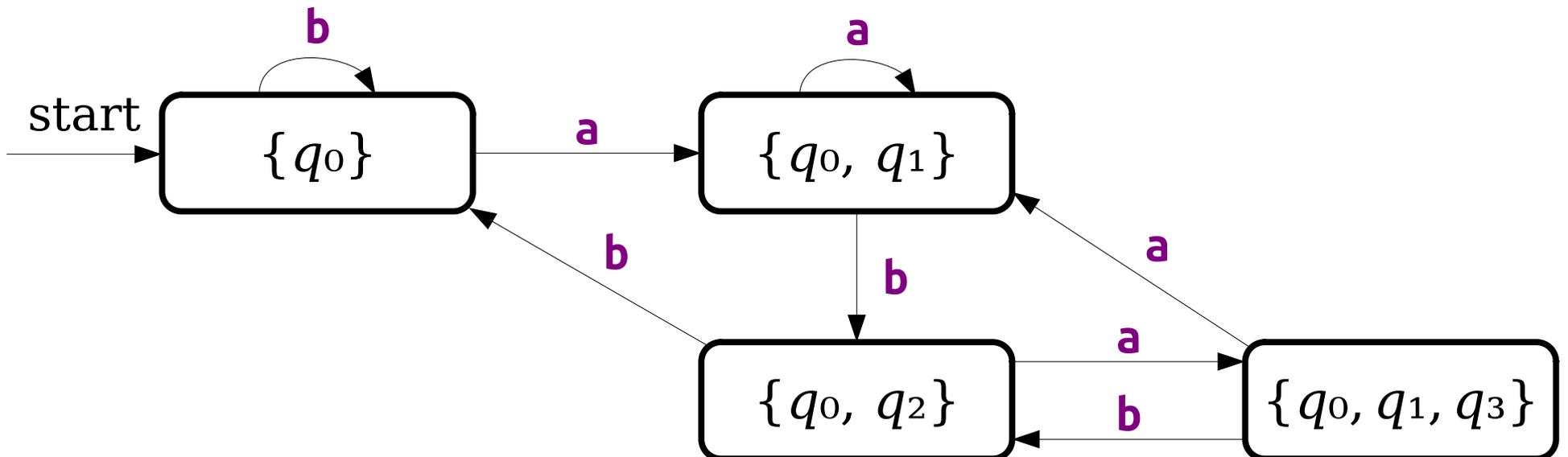


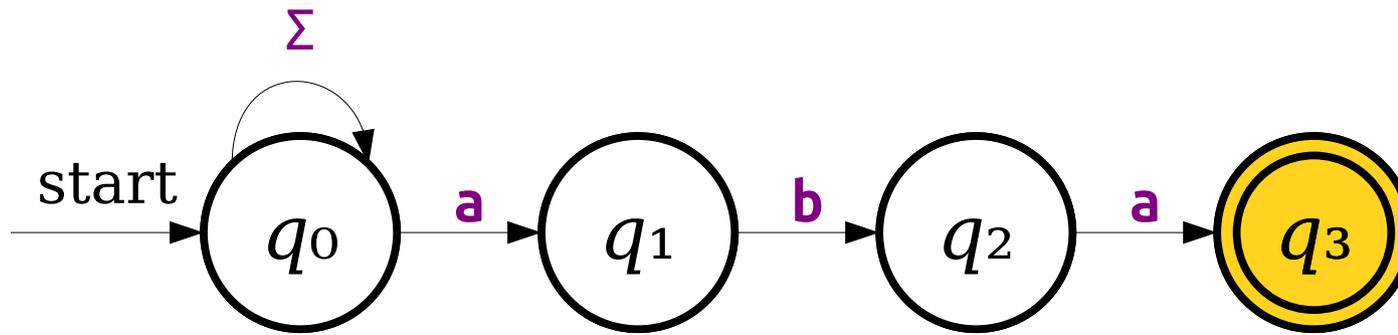
	<i>a</i>	<i>b</i>
$\{q_0\}$	$\{q_0, q_1\}$	$\{q_0\}$
$\{q_0, q_1\}$	$\{q_0, q_1\}$	$\{q_0, q_2\}$
$\{q_0, q_2\}$	$\{q_0, q_1, q_3\}$	$\{q_0\}$
$\{q_0, q_1, q_3\}$	$\{q_0, q_1\}$	$\{q_0, q_2\}$



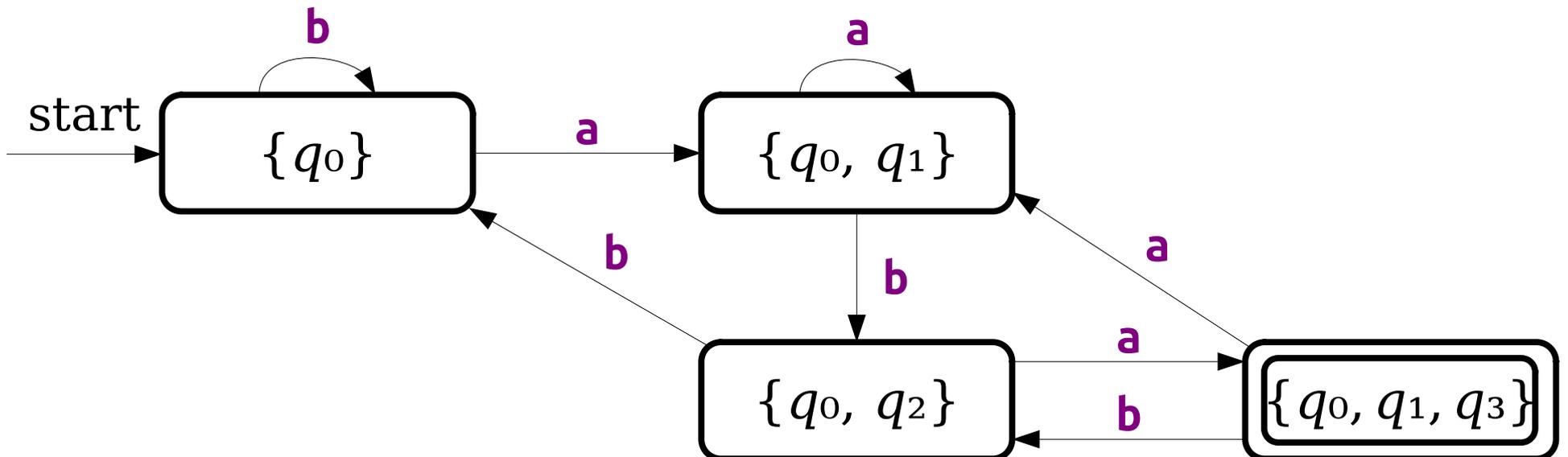


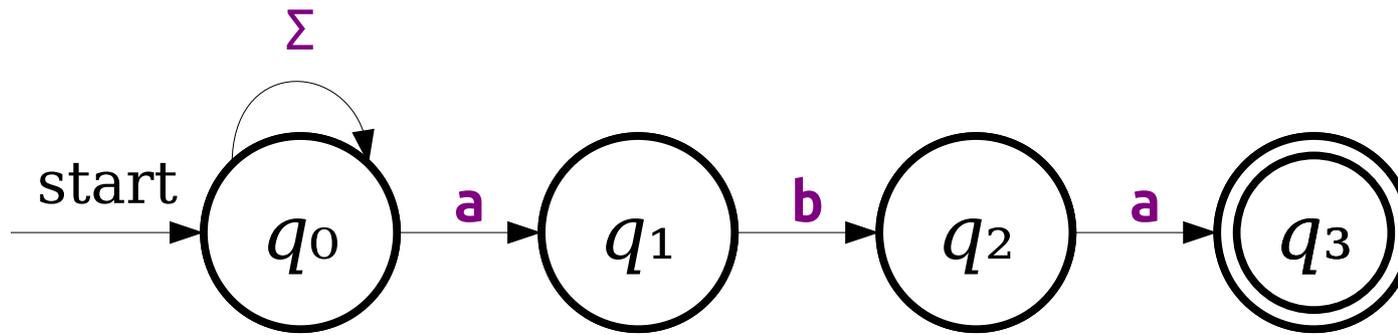
	<i>a</i>	<i>b</i>
$\{q_0\}$	$\{q_0, q_1\}$	$\{q_0\}$
$\{q_0, q_1\}$	$\{q_0, q_1\}$	$\{q_0, q_2\}$
$\{q_0, q_2\}$	$\{q_0, q_1, q_3\}$	$\{q_0\}$
$*\{q_0, q_1, q_3\}$	$\{q_0, q_1\}$	$\{q_0, q_2\}$



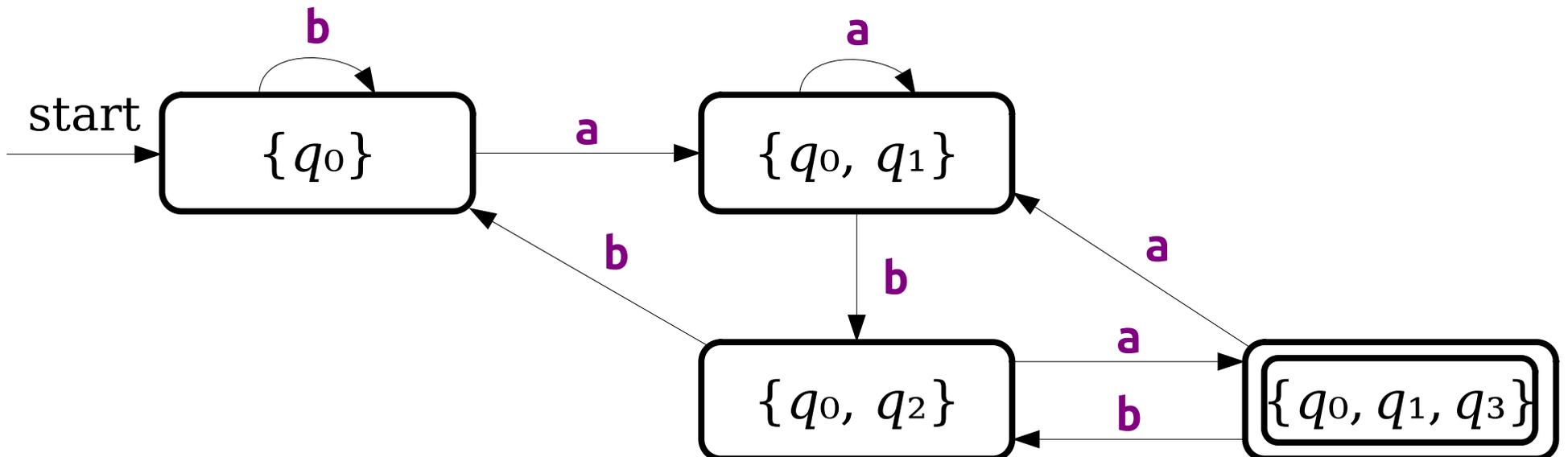


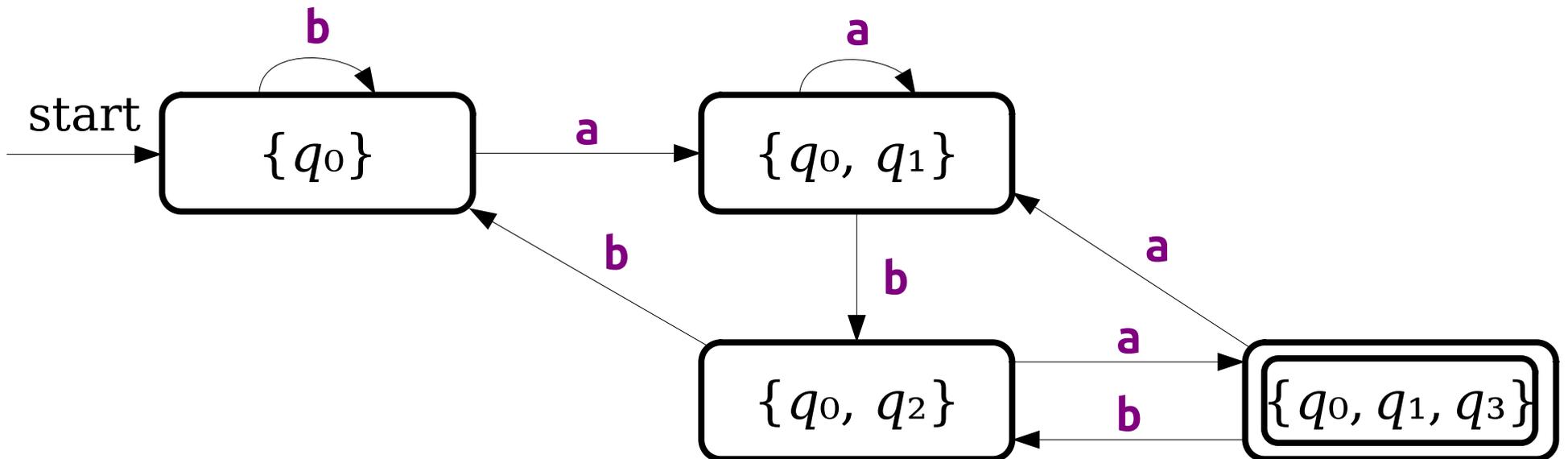
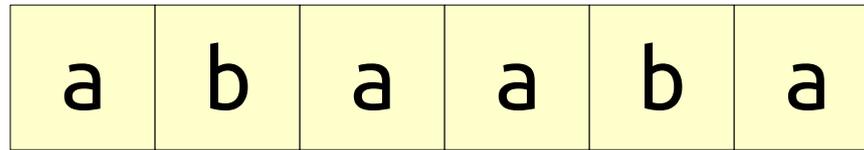
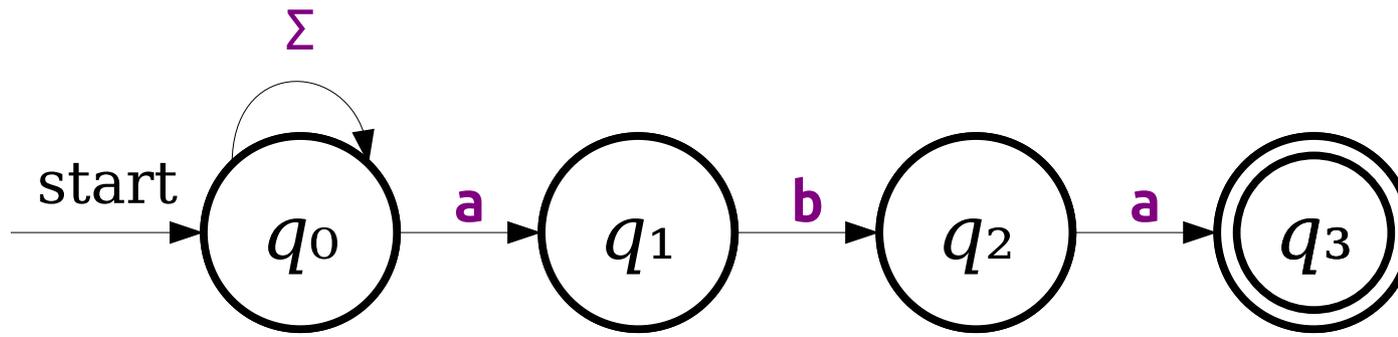
	<i>a</i>	<i>b</i>
$\{q_0\}$	$\{q_0, q_1\}$	$\{q_0\}$
$\{q_0, q_1\}$	$\{q_0, q_1\}$	$\{q_0, q_2\}$
$\{q_0, q_2\}$	$\{q_0, q_1, q_3\}$	$\{q_0\}$
$*\{q_0, q_1, q_3\}$	$\{q_0, q_1\}$	$\{q_0, q_2\}$

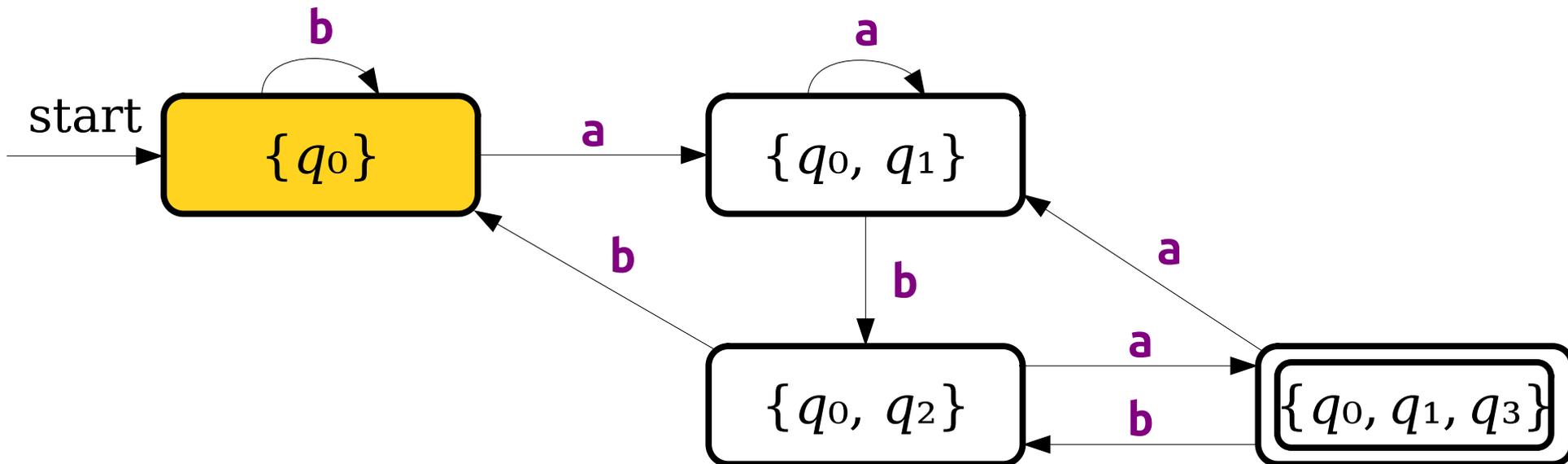
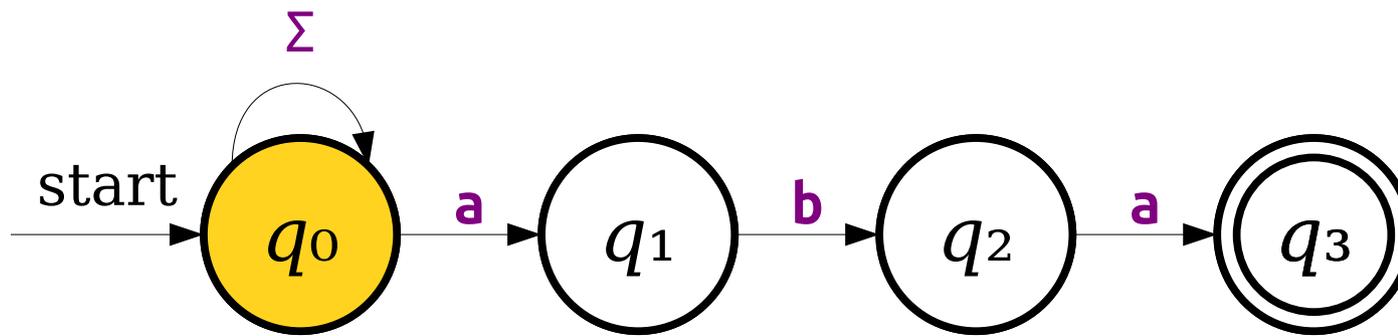


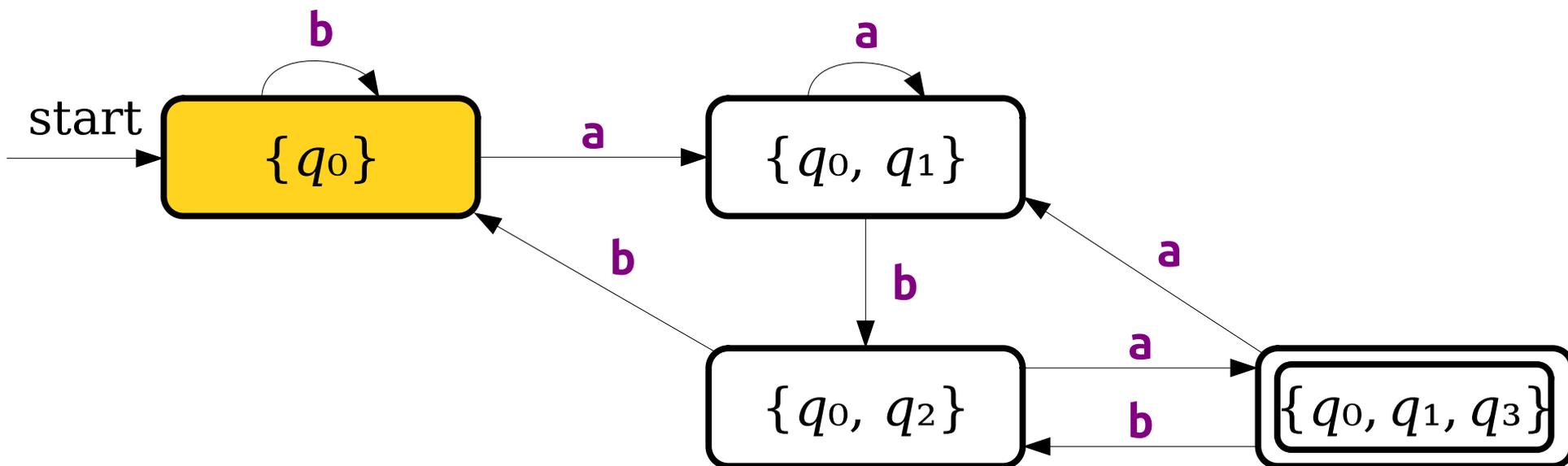
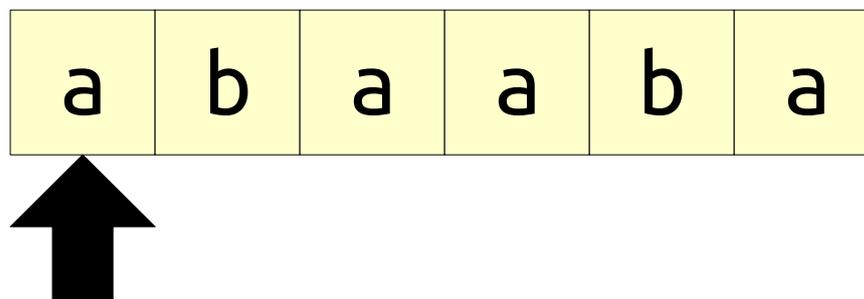
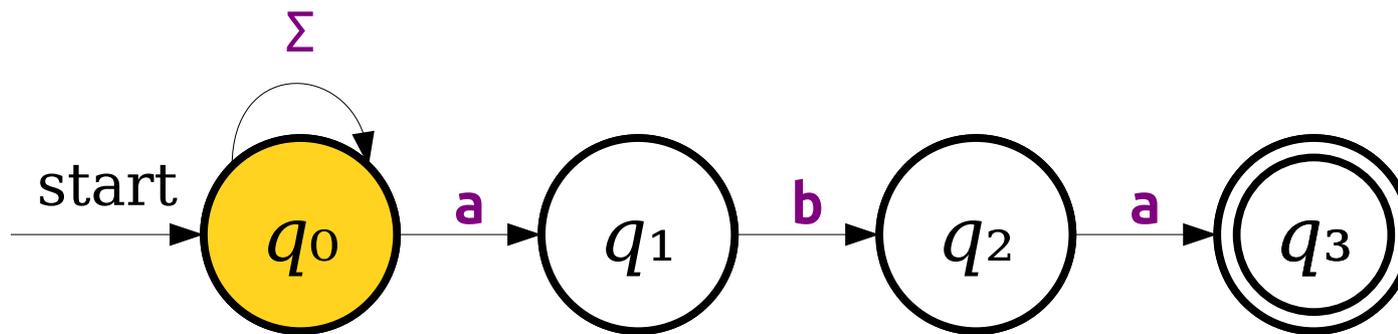


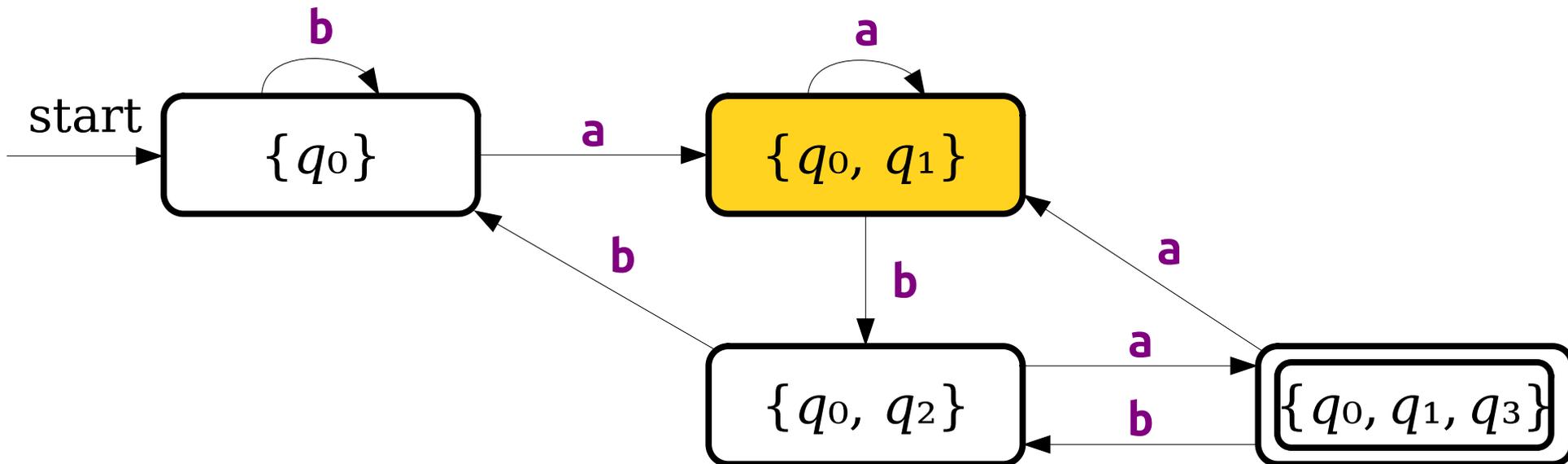
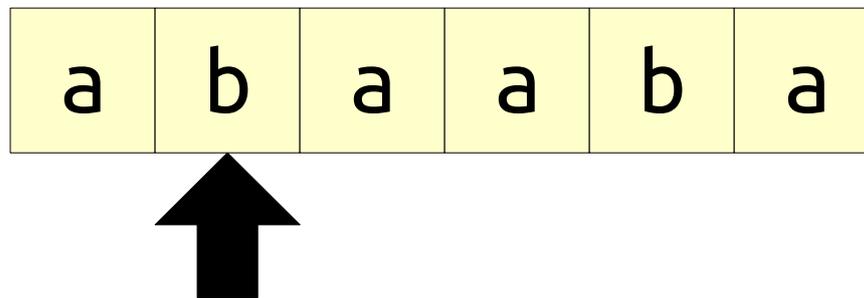
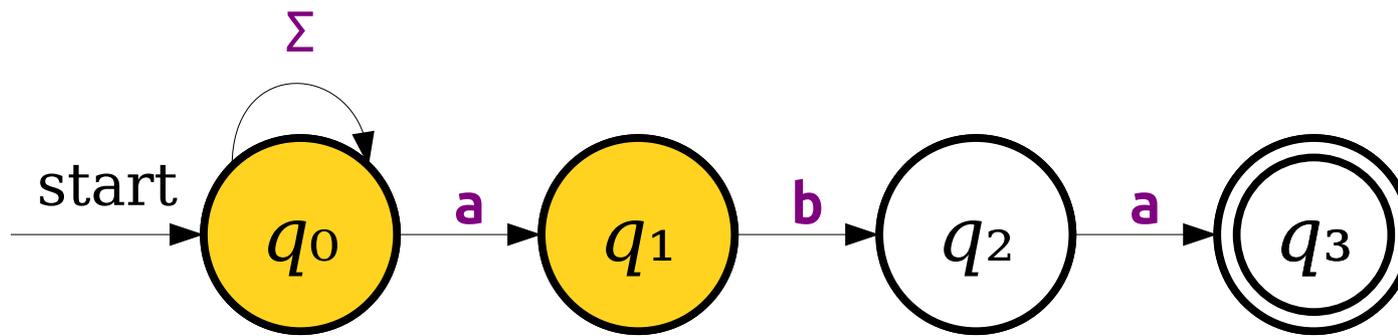
	<i>a</i>	<i>b</i>
$\{q_0\}$	$\{q_0, q_1\}$	$\{q_0\}$
$\{q_0, q_1\}$	$\{q_0, q_1\}$	$\{q_0, q_2\}$
$\{q_0, q_2\}$	$\{q_0, q_1, q_3\}$	$\{q_0\}$
$*\{q_0, q_1, q_3\}$	$\{q_0, q_1\}$	$\{q_0, q_2\}$

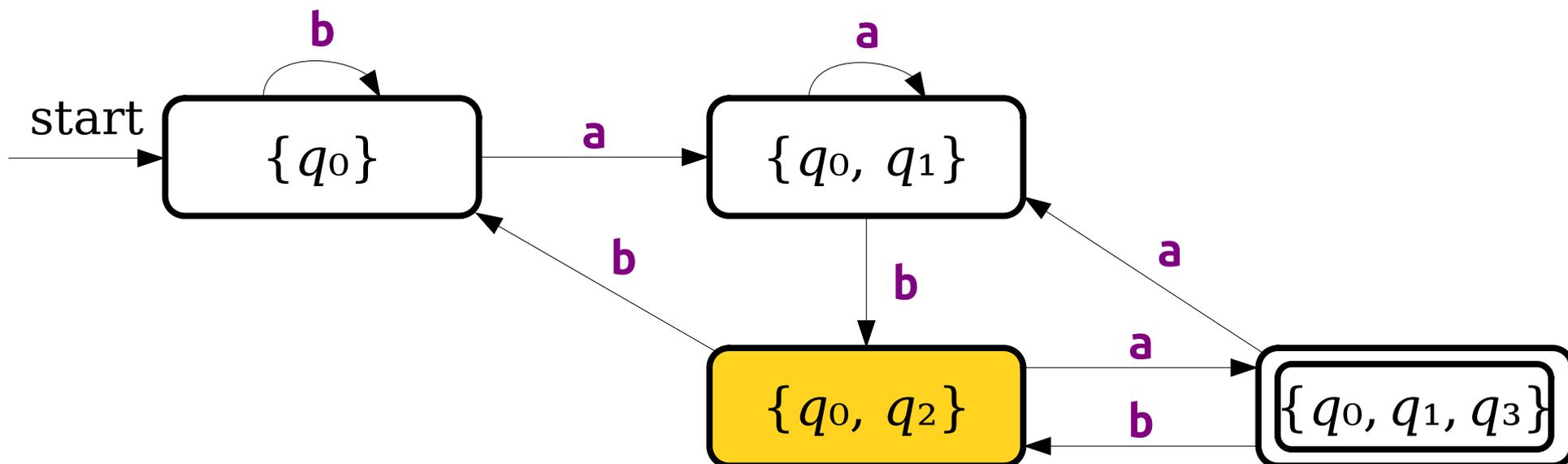
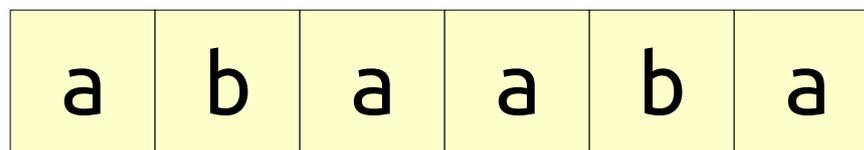
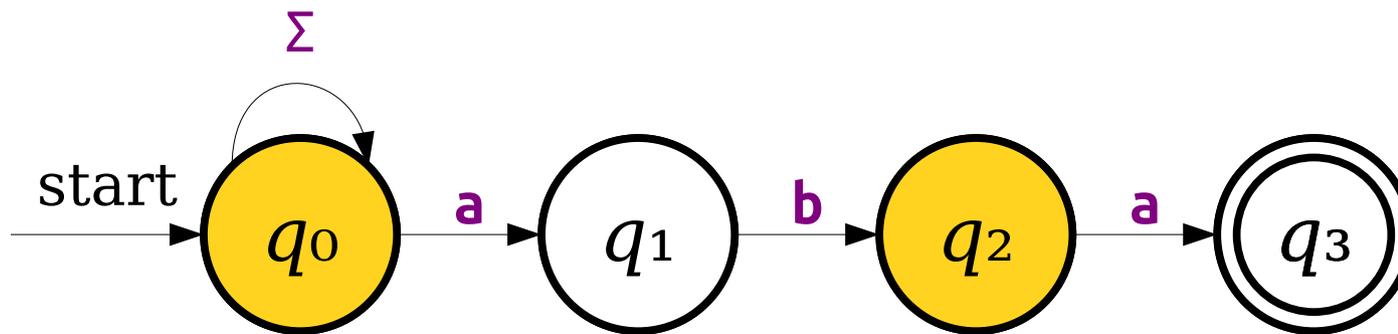


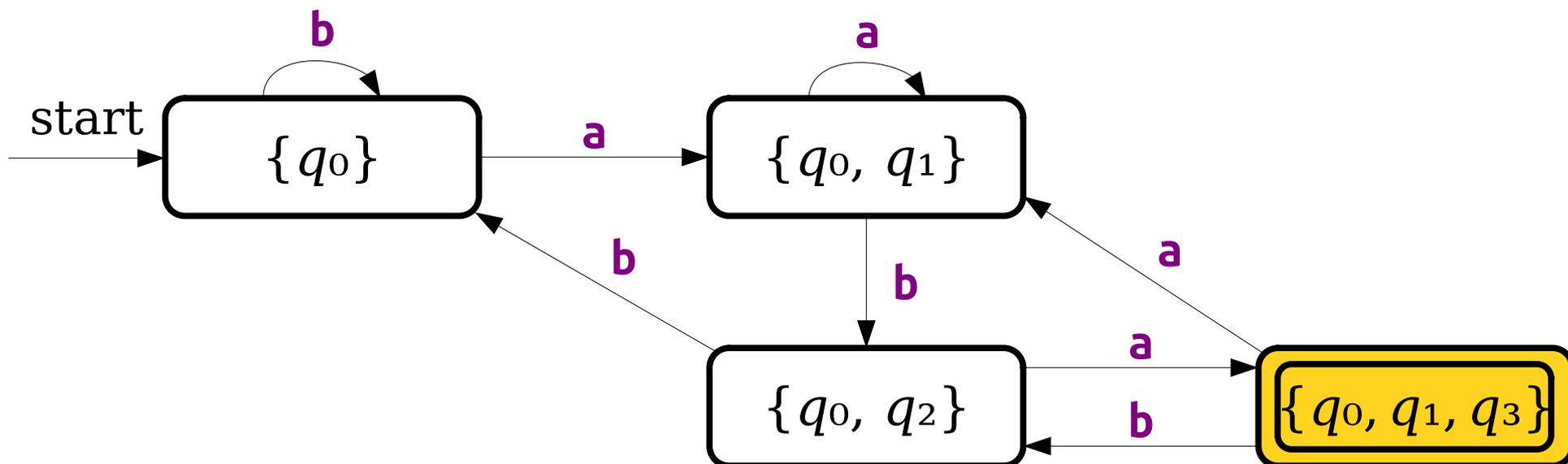
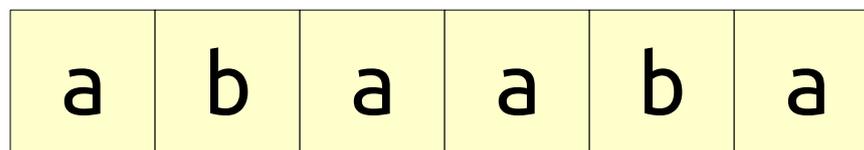
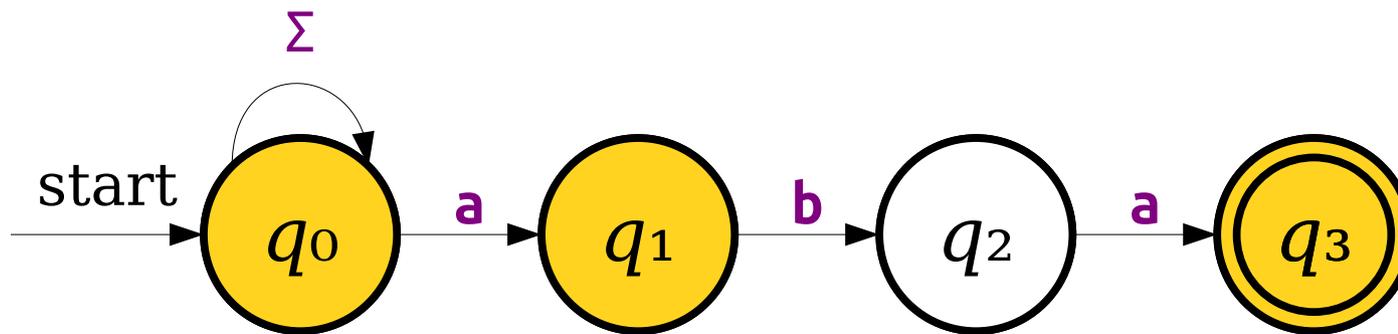


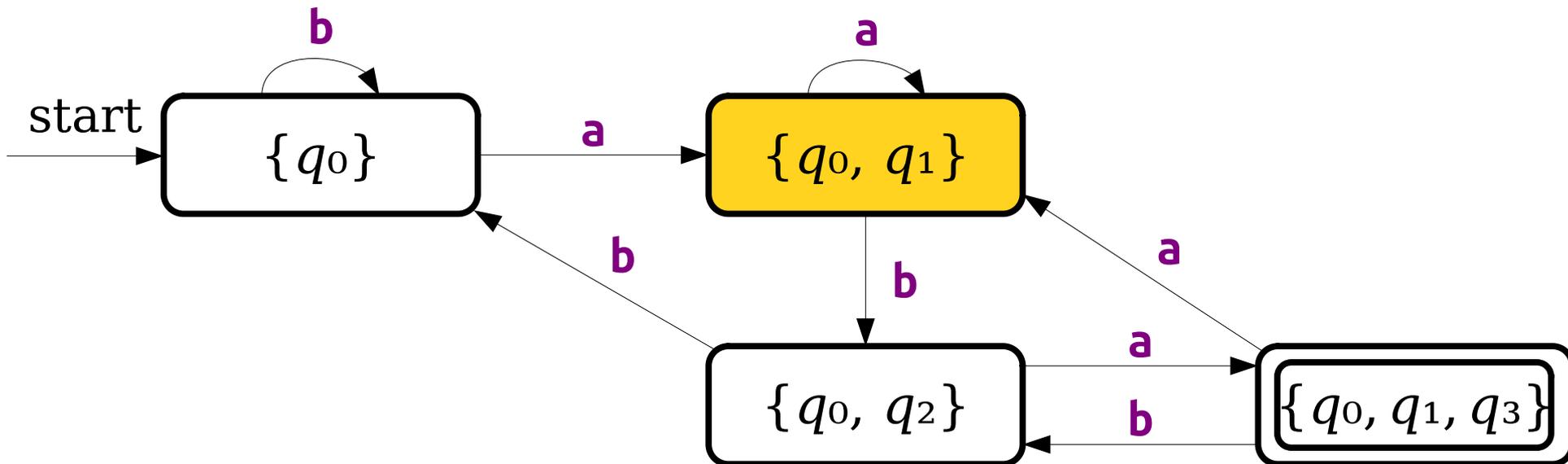
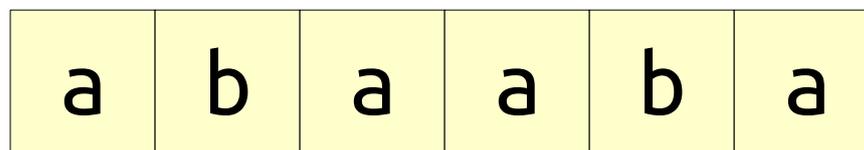
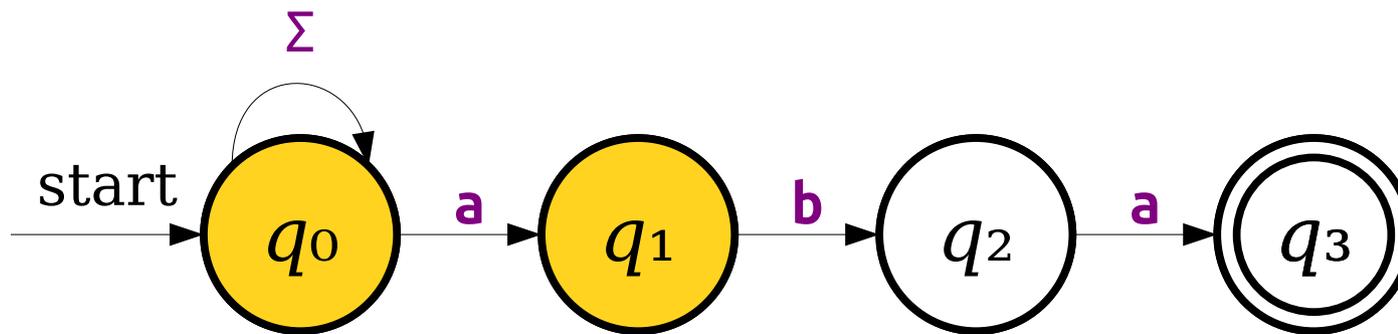


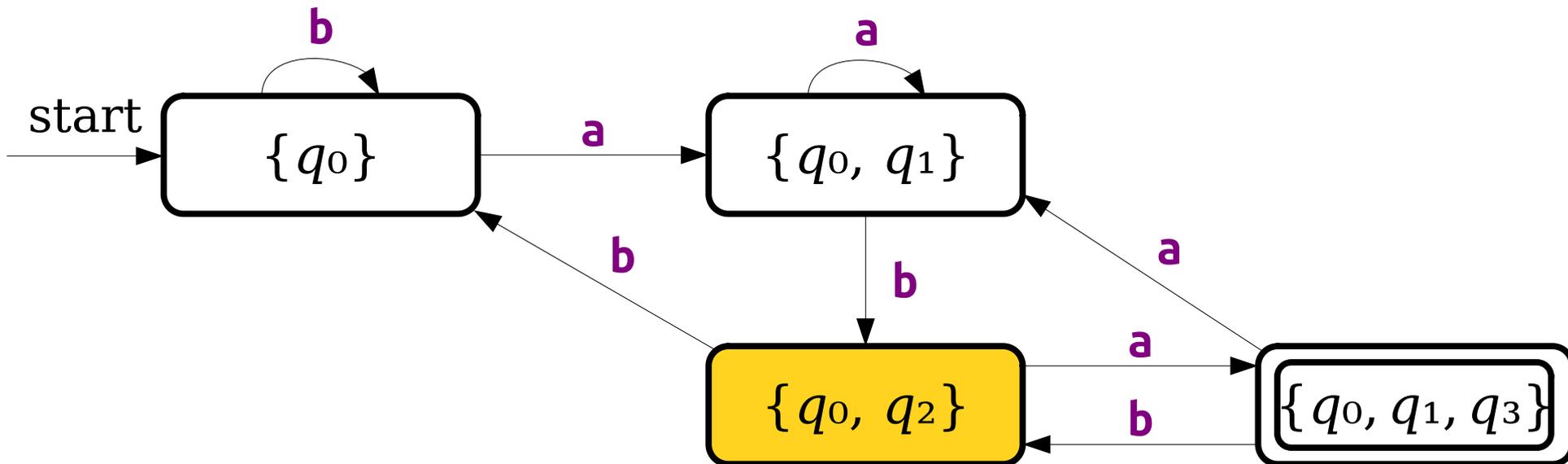
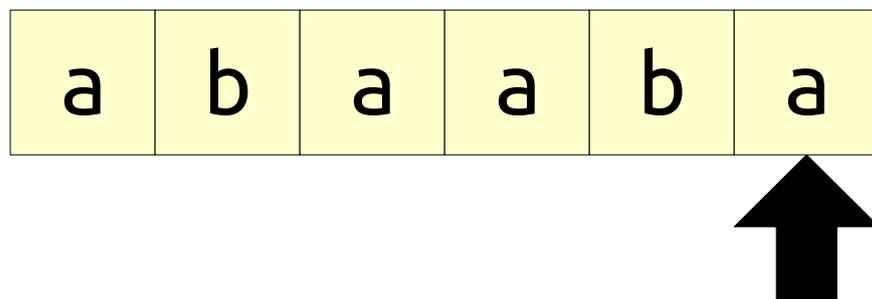
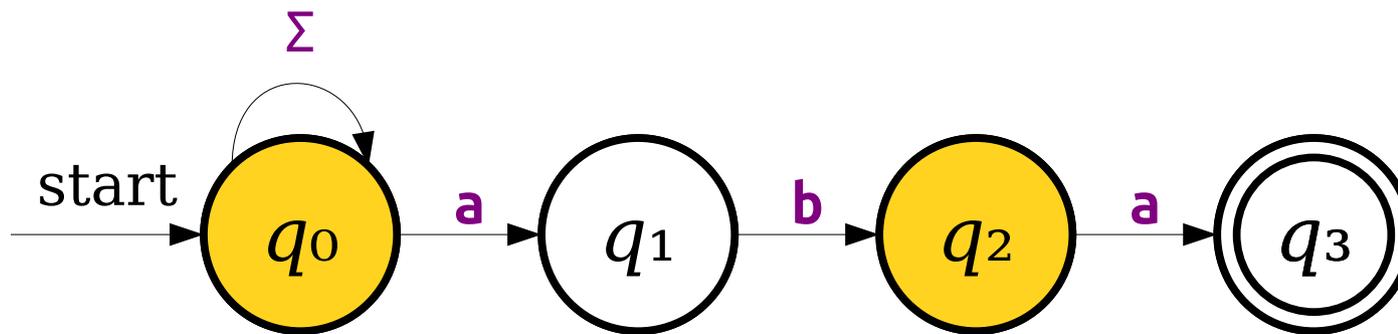


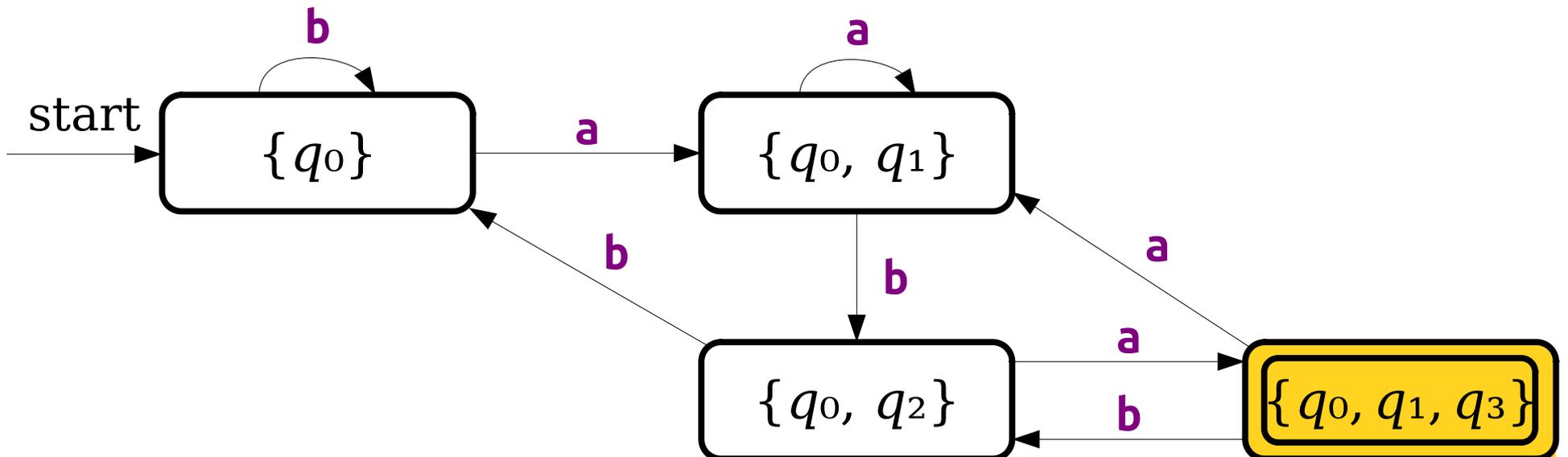
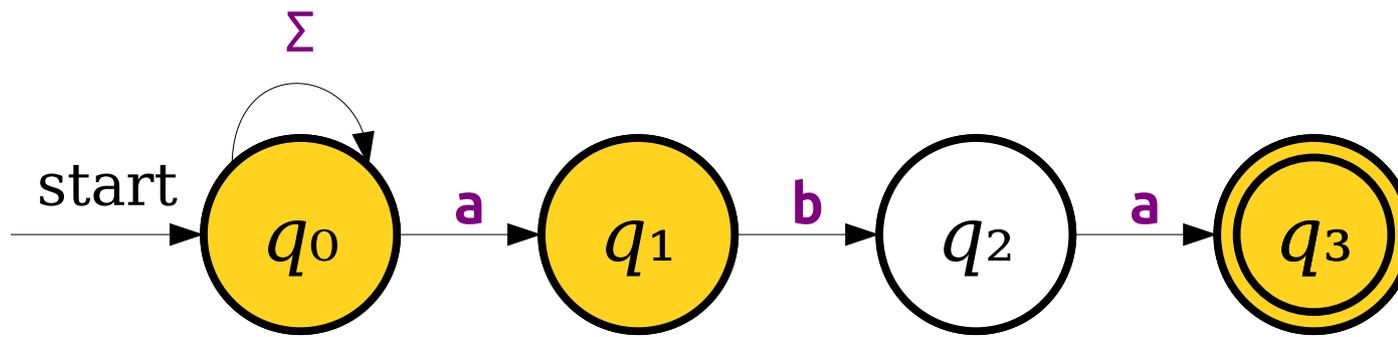












# The Subset Construction

- This procedure for turning an NFA for a language  $L$  into a DFA for a language  $L$  is called the **subset construction**.
  - It's sometimes called the **powerset construction**; it's different names for the same thing!
- Intuitively:
  - Each state in the DFA corresponds to a set of states from the NFA.
  - Each transition in the DFA corresponds to what transitions would be taken in the NFA when using the massive parallel intuition.
  - The accepting states in the DFA correspond to which sets of states would be considered accepting in the NFA when using the massive parallel intuition.
- There's an online **Guide to the Subset Construction** with a more elaborate example involving  $\epsilon$ -transitions and cases where the NFA dies; check that for more details.

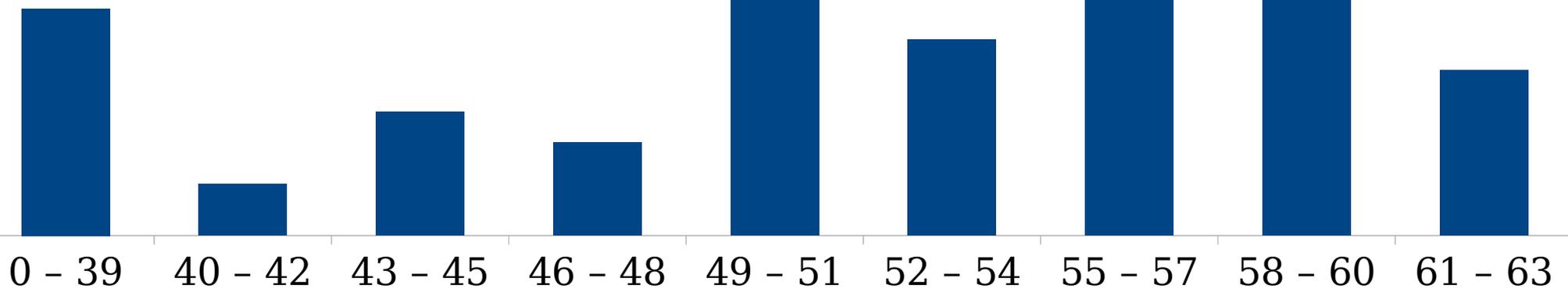
# The Subset Construction

- In converting an NFA to a DFA, the DFA's states correspond to sets of NFA states.
- **Useful fact:**  $|\wp(S)| = 2^{|S|}$  for any finite set  $S$ .
- In the worst-case, the construction can result in a DFA that is *exponentially larger* than the original NFA.
- **Question to ponder:** Can you find a family of languages that have NFAs of size  $n$ , but no DFAs of size less than  $2^n$ ?

**Time-Out for Announcements!**

# Problem Set Four Grades

75<sup>th</sup> Percentile: **58 / 63 (92%)**  
50<sup>th</sup> Percentile: **55 / 63 (87%)**  
25<sup>th</sup> Percentile: **48 / 63 (77%)**



Many of these grades are because folks forgot to list partners - please check to make sure you're getting credit for the work you're doing, and let us know if your partner forgot to add you.

# Problem Set Six

- Problem Set Five was due at 2:30PM today.
- Problem Set Six goes out today. It's due next Friday at 2:30PM.
  - Design DFAs and NFAs for a range of problems!
  - Explore formal language theory!
  - See some clever applications!

# Midterm Revise-and-Resubmit

- The first midterm exam is now open for revise-and-resubmit.
- For each problem you'd like to submit a new answer to, file a regrade request and type in your new answer.
- The deadline to submit is ***Monday at 2:30PM.***

# Extra Practice Problems 2

- Looking for more practice with functions, graphs, and induction? We've just posted Extra Practice Problems 2 to the course website.
- It's a collection of 30 problems on these topics, plus others from PS3 - PS5.
- Solutions are available as well.

Your Questions

## ~~Your Questions~~

We'll do this next time when we have a little more breathing room.

Back to CS103!

# The Regular Languages

# Regular Languages

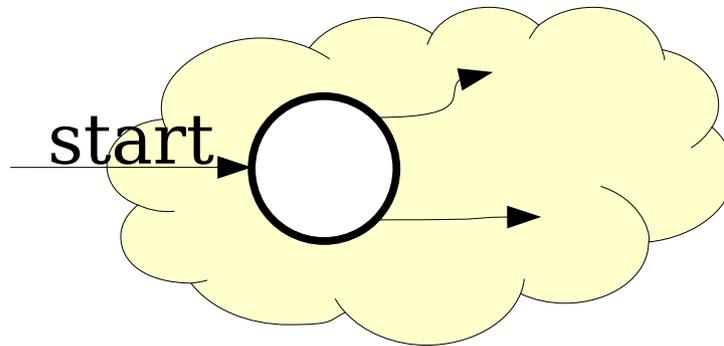
- Let  $L \subseteq \Sigma^*$  be a language.
- We say that  $L$  is a **regular language** if there is a DFA  $D$  where  $\mathcal{L}(D) = L$ .
- Equivalently,  $L$  is a regular language if there is an NFA  $N$  where  $\mathcal{L}(N) = L$ .
- Key questions:
  - What do the regular languages “feel” like?
  - What properties do they have?
  - What languages *aren't* regular?

# Closure Under Union

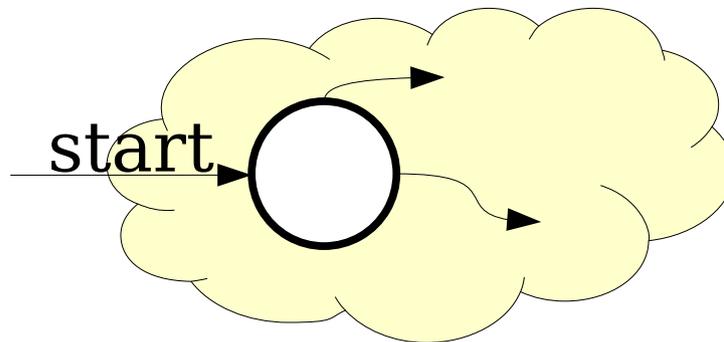
- If  $L_1$  and  $L_2$  are languages over the alphabet  $\Sigma$ , the language  $L_1 \cup L_2$  is the language of all strings in at least one of the two languages.
- If  $L_1$  and  $L_2$  are regular languages, is  $L_1 \cup L_2$ ?

# Closure Under Union

- If  $L_1$  and  $L_2$  are languages over the alphabet  $\Sigma$ , the language  $L_1 \cup L_2$  is the language of all strings in at least one of the two languages.
- If  $L_1$  and  $L_2$  are regular languages, is  $L_1 \cup L_2$ ?



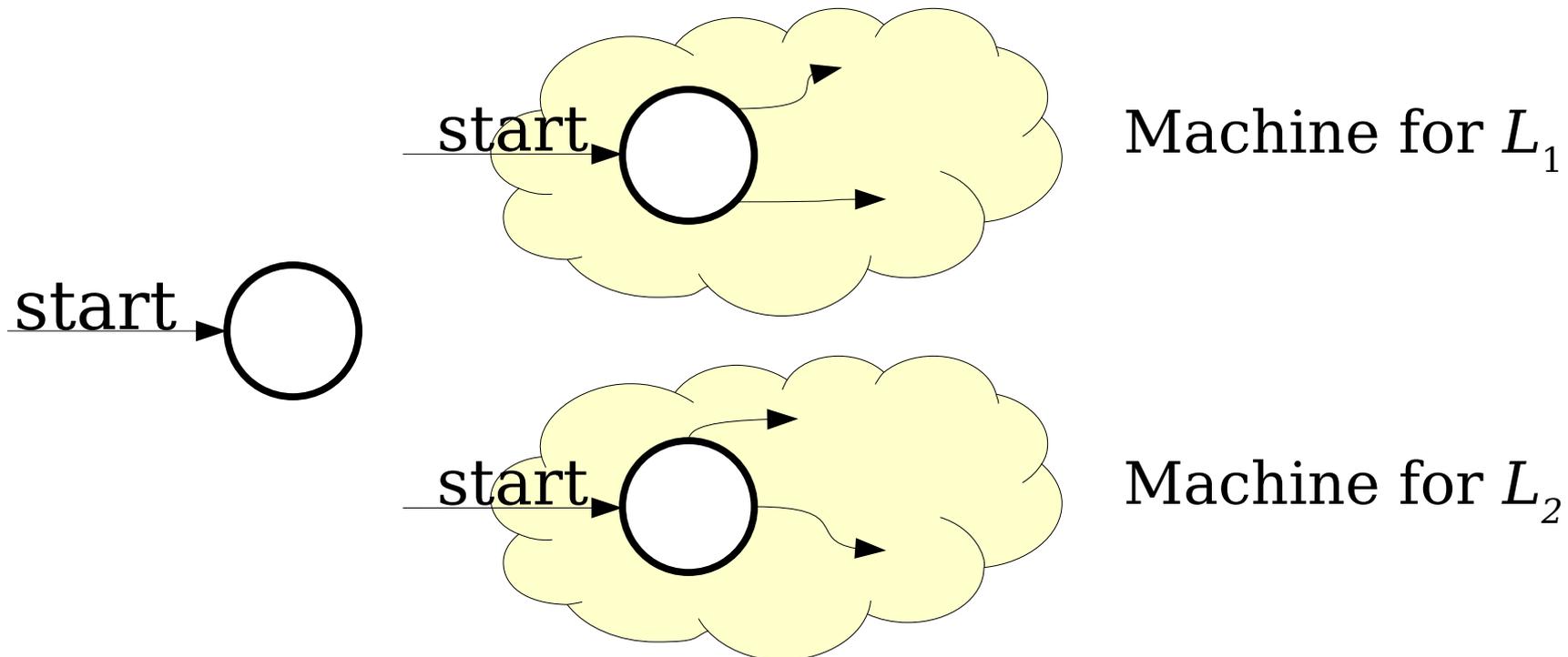
Machine for  $L_1$



Machine for  $L_2$

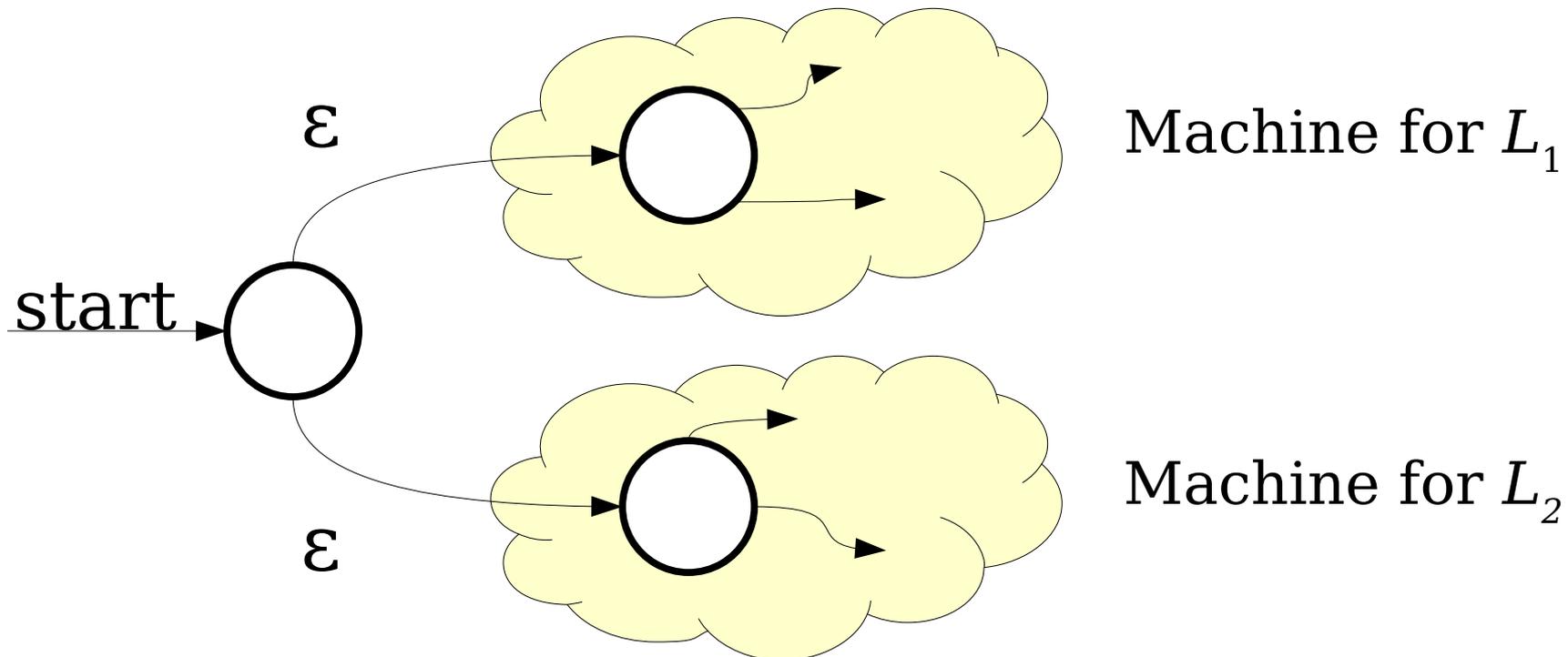
# Closure Under Union

- If  $L_1$  and  $L_2$  are languages over the alphabet  $\Sigma$ , the language  $L_1 \cup L_2$  is the language of all strings in at least one of the two languages.
- If  $L_1$  and  $L_2$  are regular languages, is  $L_1 \cup L_2$ ?



# Closure Under Union

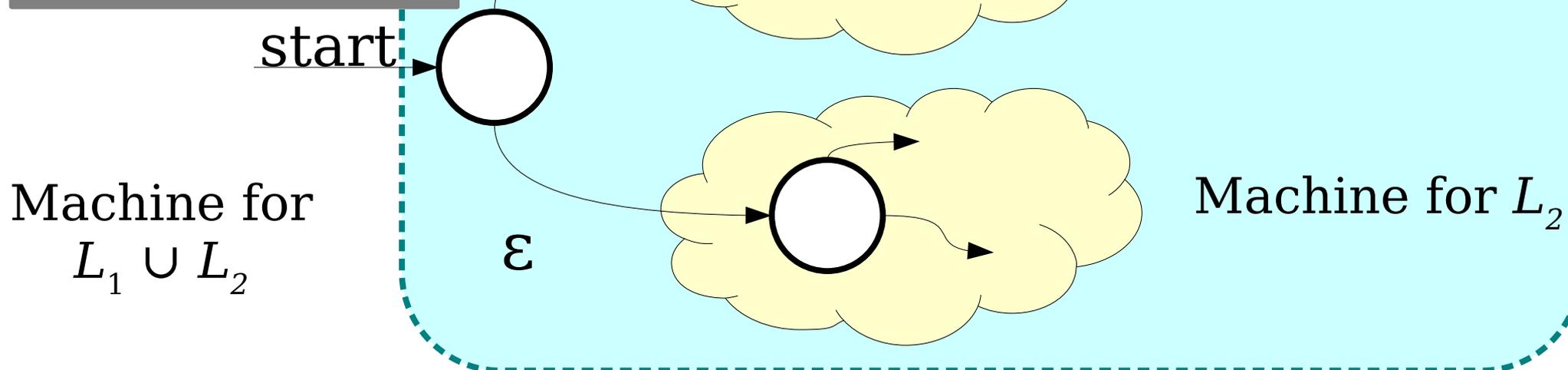
- If  $L_1$  and  $L_2$  are languages over the alphabet  $\Sigma$ , the language  $L_1 \cup L_2$  is the language of all strings in at least one of the two languages.
- If  $L_1$  and  $L_2$  are regular languages, is  $L_1 \cup L_2$ ?



# Closure Under Union

- If  $L_1$  and  $L_2$  are languages over the alphabet  $\Sigma$ , the language  $L_1 \cup L_2$  is the language of all strings in at least one of the two languages.
- If  $L_1$  and  $L_2$  are regular languages, is  $L_1 \cup L_2$ ?

**Question to ponder:** where have you seen this idea before?



Concatenation

# String Concatenation

- If  $w \in \Sigma^*$  and  $x \in \Sigma^*$ , the **concatenation** of  $w$  and  $x$ , denoted  $wx$ , is the string formed by tacking all the characters of  $x$  onto the end of  $w$ .
- Example: if  $w = \text{quo}$  and  $x = \text{kka}$ , the concatenation  $wx = \text{quokka}$ .
- This is analogous to the  $+$  operator for strings in many programming languages.
- Some facts about concatenation:
  - The empty string  $\varepsilon$  is the **identity element** for concatenation:

$$w\varepsilon = \varepsilon w = w$$

- Concatenation is **associative**:

$$wxy = w(xy) = (wx)y$$

# Concatenation

- The **concatenation** of two languages  $L_1$  and  $L_2$  over the alphabet  $\Sigma$  is the language  $L_1L_2 = \{ x \mid \exists w_1 \in L_1. \exists w_2 \in L_2. x = w_1w_2 \}$

# Concatenation Example

- Let  $\Sigma = \{ a, b, \dots, z, A, B, \dots, Z \}$  and consider these languages over  $\Sigma$ :
  - ***Noun*** = { **Puppy, Rainbow, Whale, ...** }
  - ***Verb*** = { **Hugs, Juggles, Loves, ...** }
  - ***The*** = { **The** }
- The language ***TheNounVerbTheNoun*** is
  - { **ThePuppyHugsTheWhale,**  
**TheWhaleLovesTheRainbow,**  
**TheRainbowJugglesTheRainbow, ...** }

# Concatenation

- The **concatenation** of two languages  $L_1$  and  $L_2$  over the alphabet  $\Sigma$  is the language

$$L_1L_2 = \{ x \mid \exists w_1 \in L_1. \exists w_2 \in L_2. x = w_1w_2 \}$$

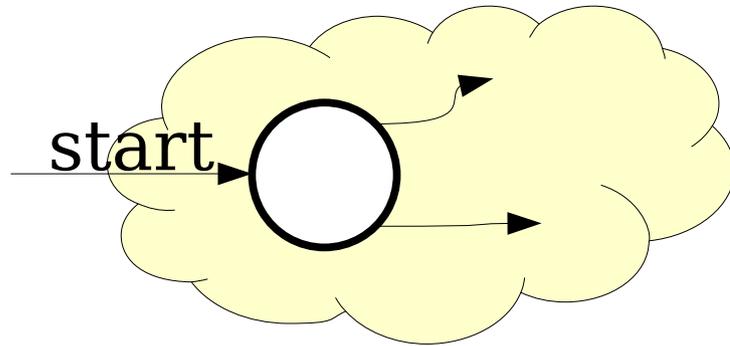
- Two views of  $L_1L_2$ :
  - The set of all strings that can be made by concatenating a string in  $L_1$  with a string in  $L_2$ .
  - The set of strings that can be split into two pieces: a piece from  $L_1$  and a piece from  $L_2$ .

# Closure Under Concatenation

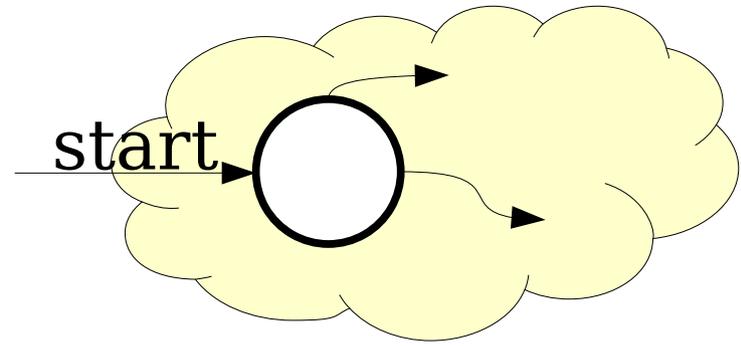
- If  $L_1$  and  $L_2$  are regular languages, is  $L_1L_2$ ?
- Intuition - can we split a string  $w$  into two strings  $xy$  such that  $x \in L_1$  and  $y \in L_2$ ?

# Closure Under Concatenation

- If  $L_1$  and  $L_2$  are regular languages, is  $L_1L_2$ ?
- Intuition - can we split a string  $w$  into two strings  $xy$  such that  $x \in L_1$  and  $y \in L_2$ ?



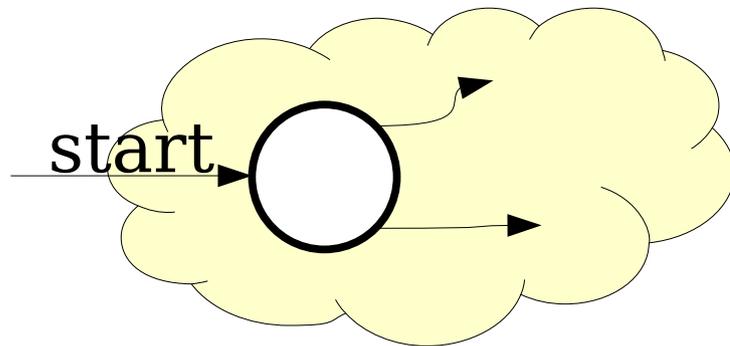
Machine for  $L_1$



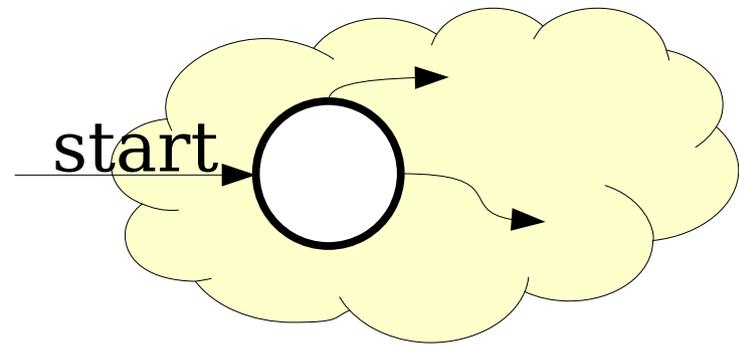
Machine for  $L_2$

# Closure Under Concatenation

- If  $L_1$  and  $L_2$  are regular languages, is  $L_1L_2$ ?
- Intuition - can we split a string  $w$  into two strings  $xy$  such that  $x \in L_1$  and  $y \in L_2$ ?



Machine for  $L_1$

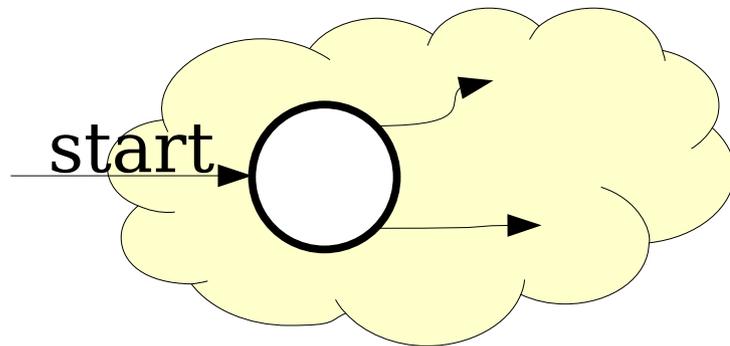


Machine for  $L_2$

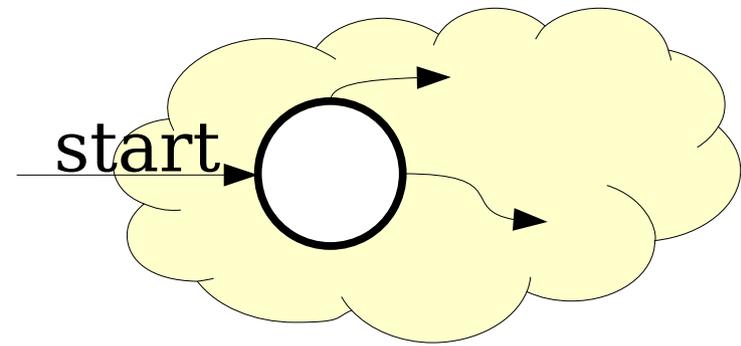
<b>b</b>	<b>o</b>	<b>o</b>	<b>k</b>	<b>k</b>	<b>e</b>	<b>e</b>	<b>p</b>	<b>e</b>	<b>r</b>
----------	----------	----------	----------	----------	----------	----------	----------	----------	----------

# Closure Under Concatenation

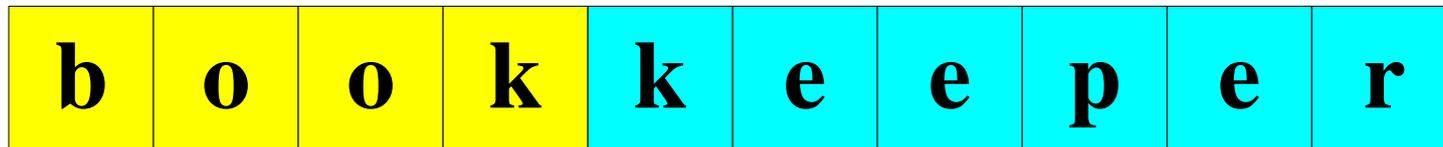
- If  $L_1$  and  $L_2$  are regular languages, is  $L_1L_2$ ?
- Intuition - can we split a string  $w$  into two strings  $xy$  such that  $x \in L_1$  and  $y \in L_2$ ?



Machine for  $L_1$

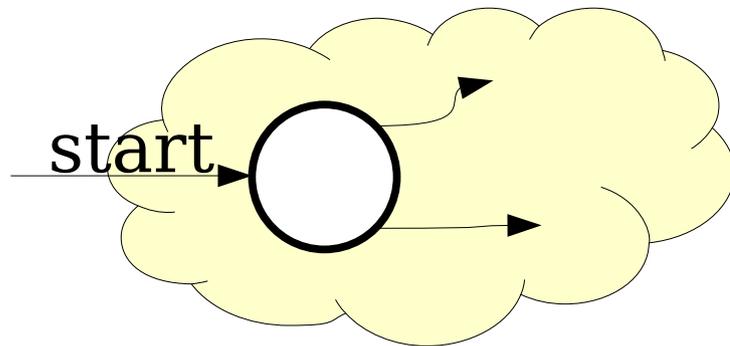


Machine for  $L_2$



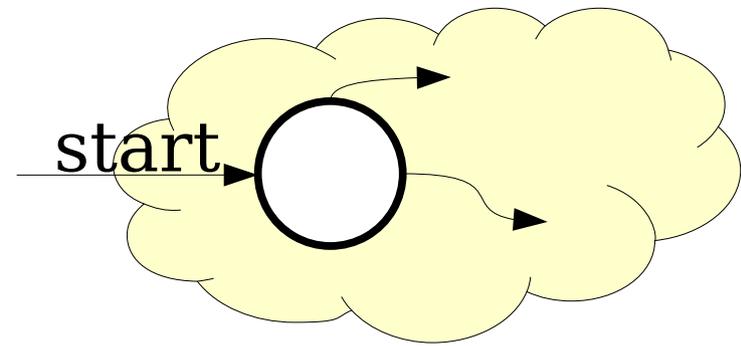
# Closure Under Concatenation

- If  $L_1$  and  $L_2$  are regular languages, is  $L_1L_2$ ?
- Intuition - can we split a string  $w$  into two strings  $xy$  such that  $x \in L_1$  and  $y \in L_2$ ?



Machine for  $L_1$

<b>b</b>	<b>o</b>	<b>o</b>	<b>k</b>
----------	----------	----------	----------



Machine for  $L_2$

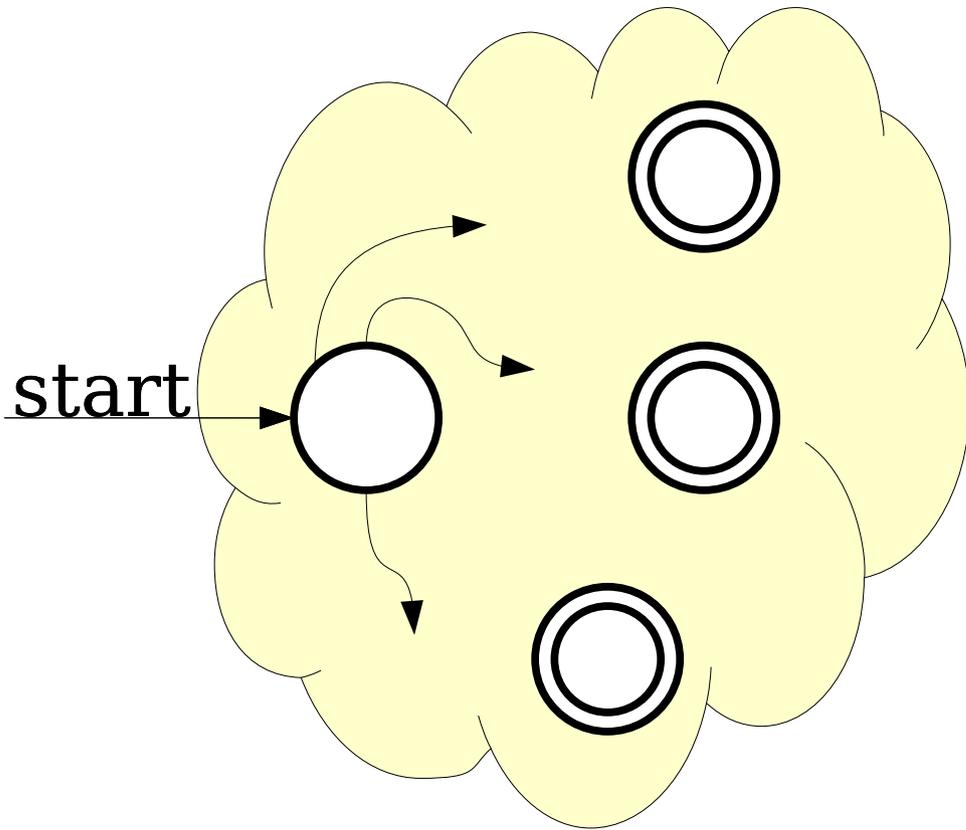
<b>k</b>	<b>e</b>	<b>e</b>	<b>p</b>	<b>e</b>	<b>r</b>
----------	----------	----------	----------	----------	----------

# Closure Under Concatenation

- If  $L_1$  and  $L_2$  are regular languages, is  $L_1L_2$ ?
- Intuition - can we split a string  $w$  into two strings  $xy$  such that  $x \in L_1$  and  $y \in L_2$ ?
- **Idea:**
  - Run a DFA/NFA for  $L_1$  on  $w$ .
  - Whenever it reaches an accepting state, optionally hand the rest of  $w$  to a DFA/NFA for  $L_2$ .
  - If the automaton for  $L_2$  accepts the rest,  $w \in L_1L_2$ .
  - If the automaton for  $L_2$  rejects the remainder, the split was incorrect.

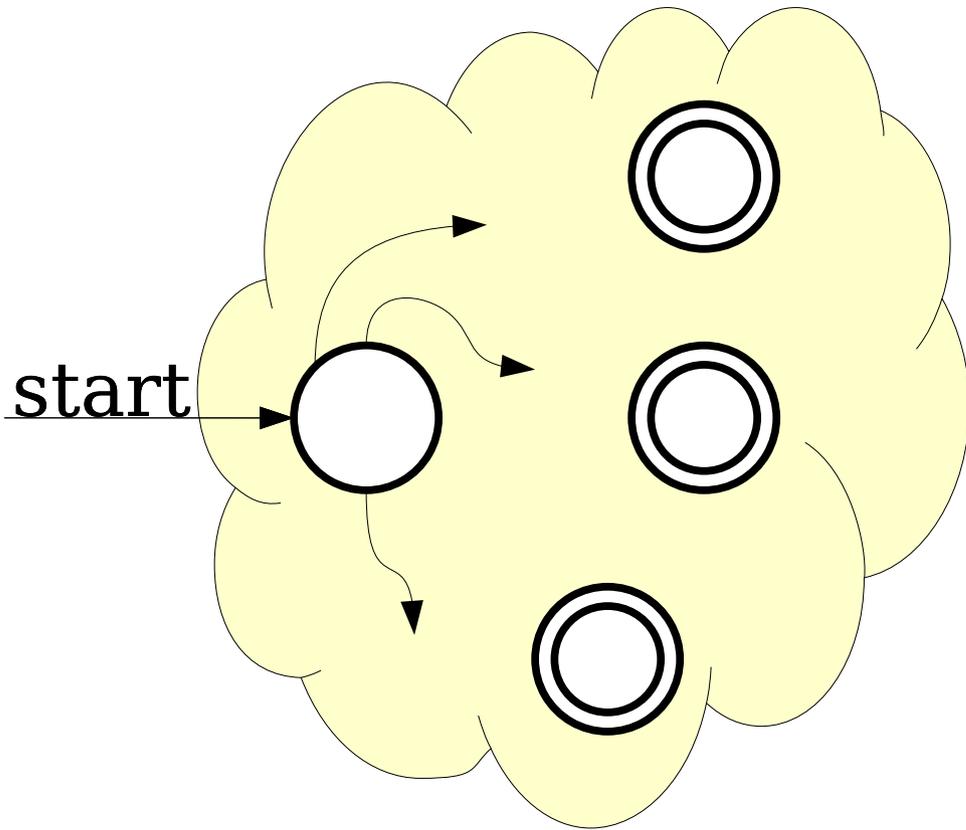
# Closure Under Concatenation

# Closure Under Concatenation

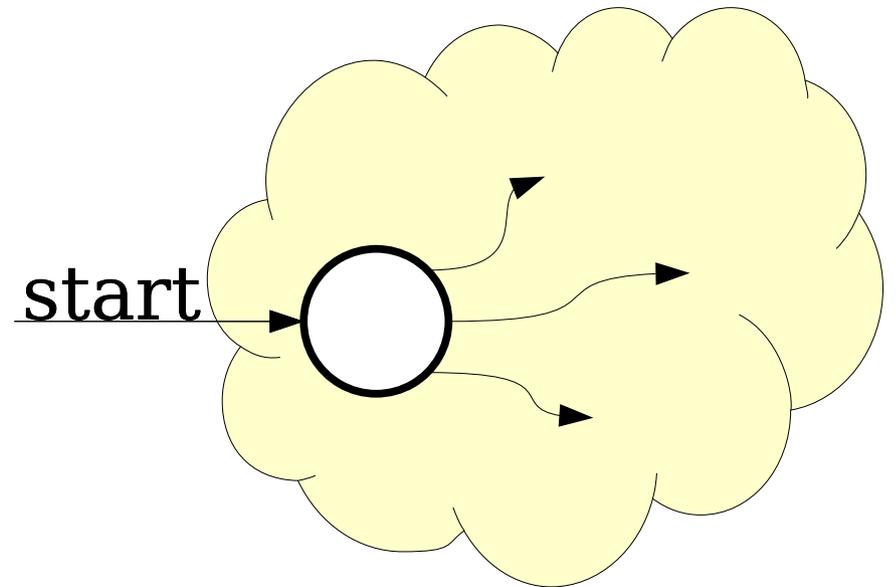


Machine for  
 $L_1$

# Closure Under Concatenation

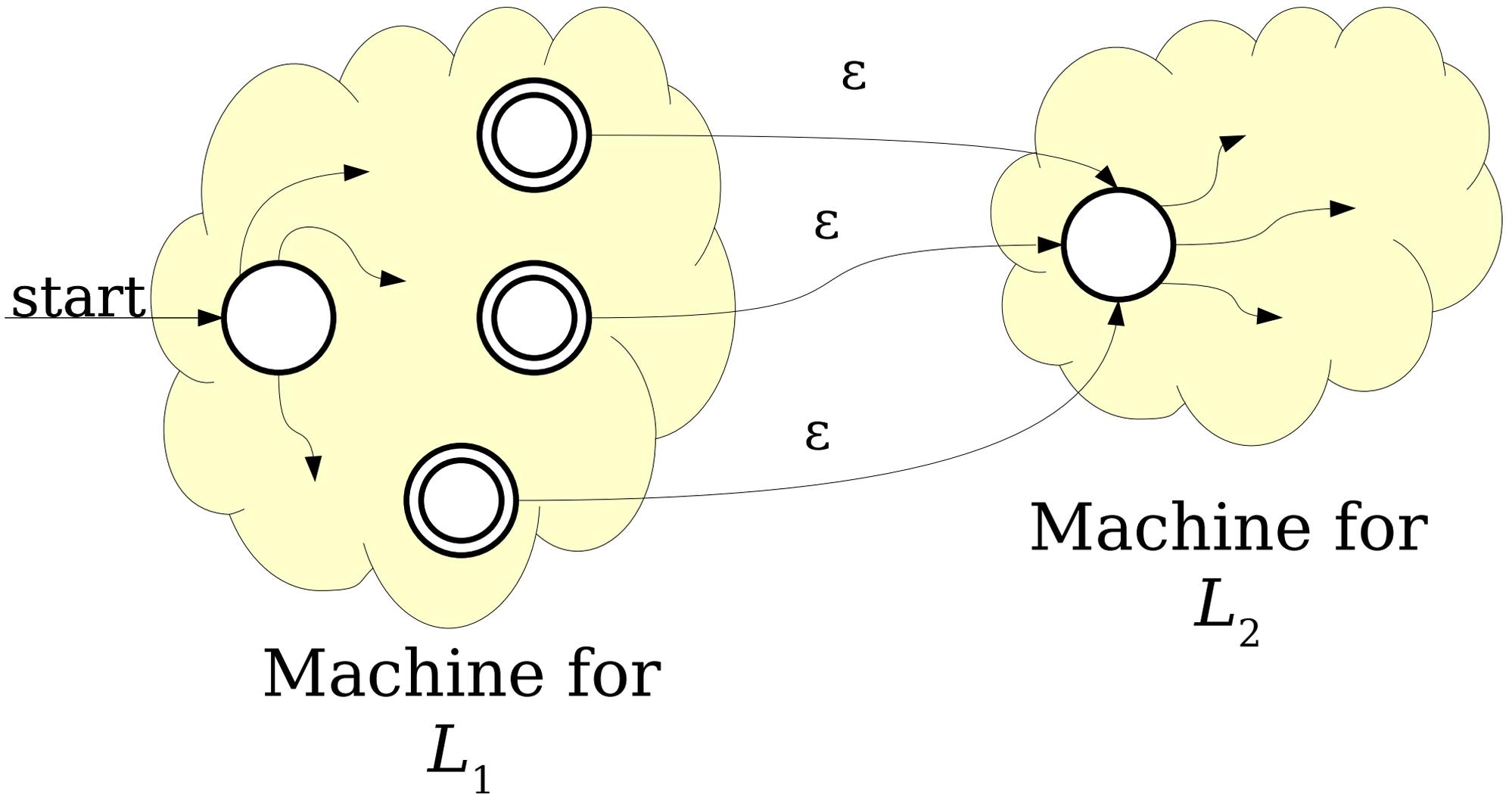


Machine for  
 $L_1$

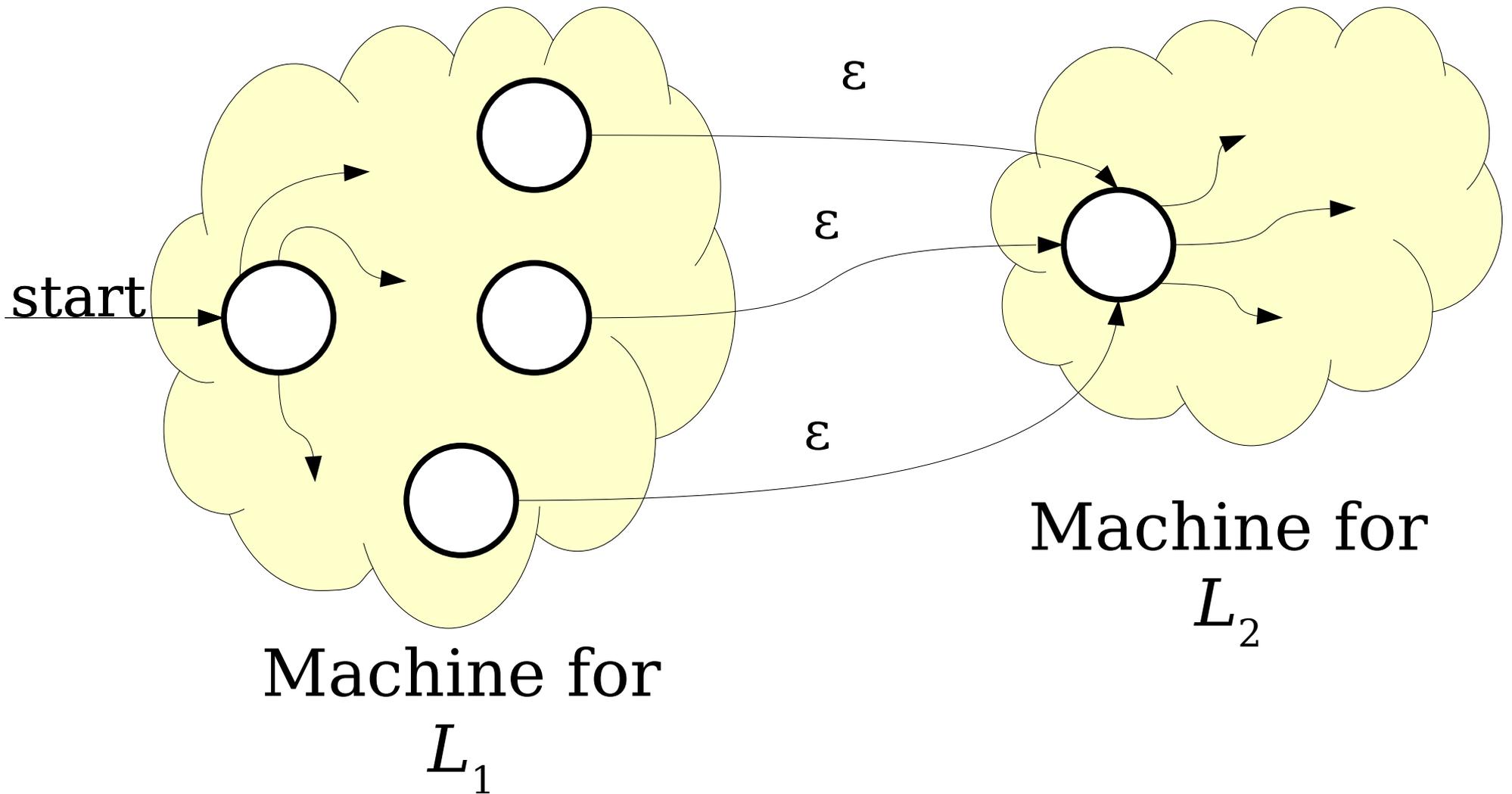


Machine for  
 $L_2$

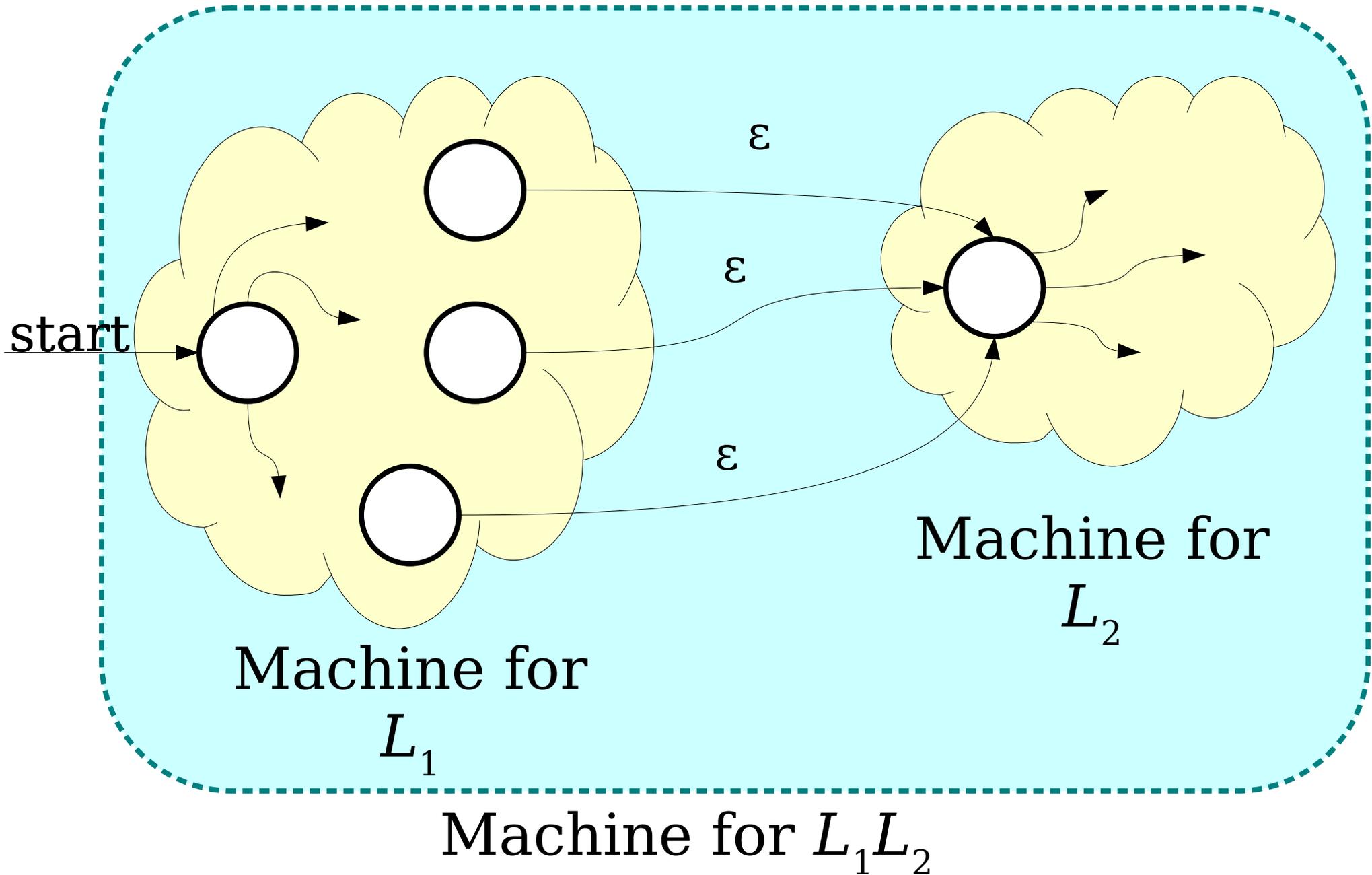
# Closure Under Concatenation



# Closure Under Concatenation



# Closure Under Concatenation



The Kleene Star

# Lots and Lots of Concatenation

- Consider the language  $L = \{ \mathbf{aa}, \mathbf{b} \}$
- $LL$  is the set of strings formed by concatenating pairs of strings in  $L$ .

$\{ \mathbf{aaaa}, \mathbf{aab}, \mathbf{baa}, \mathbf{bb} \}$

- $LLL$  is the set of strings formed by concatenating triples of strings in  $L$ .

$\{ \mathbf{aaaaaa}, \mathbf{aaaab}, \mathbf{aabaa}, \mathbf{aabb}, \mathbf{baaaa}, \mathbf{baab}, \mathbf{bbaa}, \mathbf{bbb} \}$

- $LLLL$  is the set of strings formed by concatenating quadruples of strings in  $L$ .

$\{ \mathbf{aaaaaaaa}, \mathbf{aaaaaab}, \mathbf{aaaabaa}, \mathbf{aaaabb}, \mathbf{aabaaaa}, \mathbf{aabaab}, \mathbf{aabbaa}, \mathbf{aabbb}, \mathbf{baaaaaa}, \mathbf{baaaab}, \mathbf{baabaa}, \mathbf{baabb}, \mathbf{bbaaaa}, \mathbf{bbaab}, \mathbf{bbbaa}, \mathbf{bbbb} \}$

# Language Exponentiation

- We can define what it means to “exponentiate” a language as follows:
- $L^0 = \{\varepsilon\}$ 
  - Intuition: The only string you can form by gluing no strings together is the empty string.
  - Notice that  $\{\varepsilon\} \neq \emptyset$ . Can you explain why?
- $L^{n+1} = LL^n$ 
  - Idea: Concatenating  $(n+1)$  strings together works by concatenating  $n$  strings, then concatenating one more.
- **Question to ponder:** Why define  $L^0 = \{\varepsilon\}$ ?
- **Question to ponder:** What is  $\emptyset^0$ ?

The Kleene Star

# The Kleene Closure

- An important operation on languages is the ***Kleene Closure***, which is defined as

$$L^* = \{ w \in \Sigma^* \mid \exists n \in \mathbb{N}. w \in L^n \}$$

- Mathematically:

$$w \in L^* \quad \leftrightarrow \quad \exists n \in \mathbb{N}. w \in L^n$$

- Intuitively,  $L^*$  is the language all possible ways of concatenating zero or more strings in  $L$  together, possibly with repetition.
- ***Question to ponder:*** What is  $\emptyset^*$ ?

# The Kleene Closure

If  $L = \{ \mathbf{a}, \mathbf{bb} \}$ , then  $L^* = \{$

$\epsilon,$

$\mathbf{a}, \mathbf{bb},$

$\mathbf{aa}, \mathbf{abb}, \mathbf{bba}, \mathbf{bbbb},$

$\mathbf{aaa}, \mathbf{aabb}, \mathbf{abba}, \mathbf{abbbb}, \mathbf{bbaa}, \mathbf{bbabb}, \mathbf{bbbba}, \mathbf{bbbbbb},$

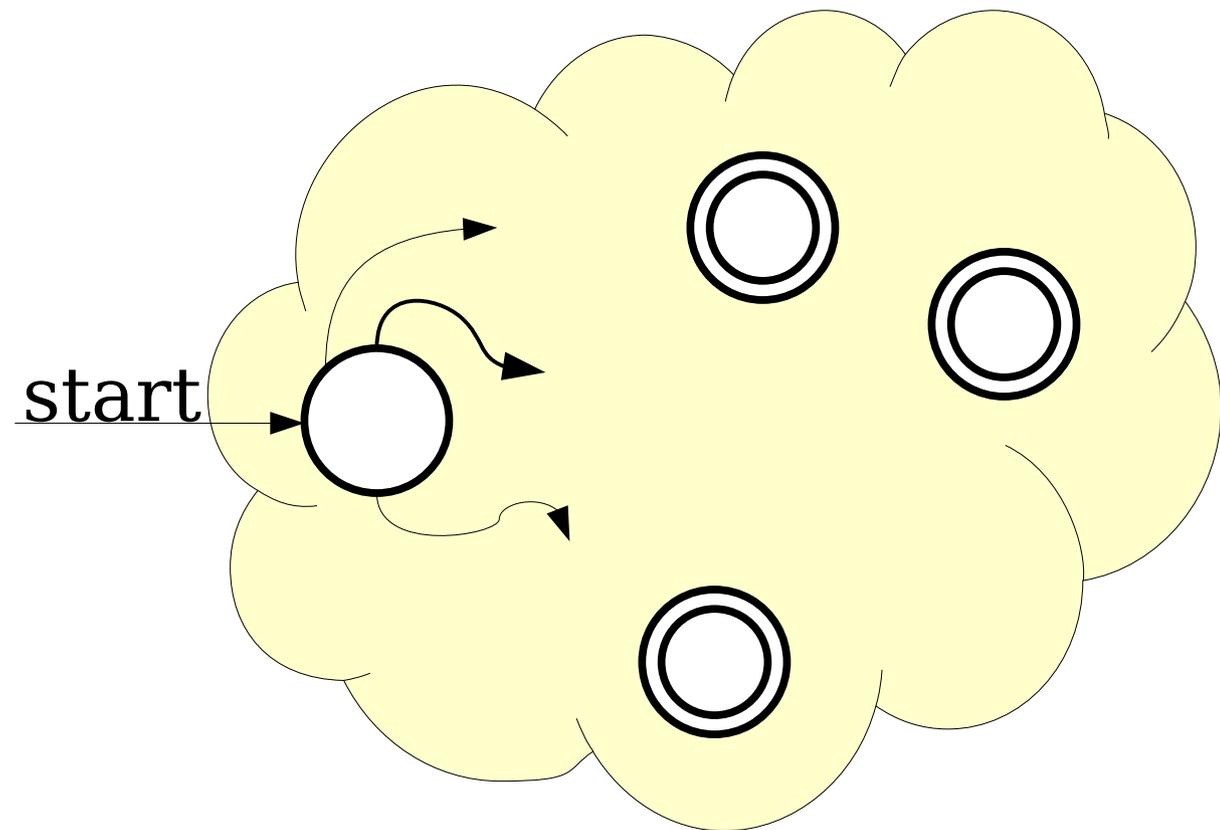
$\dots$

$\}$

Think of  $L^*$  as the set of strings you can make if you have a collection of stamps – one for each string in  $L$  – and you form every possible string that can be made from those stamps.

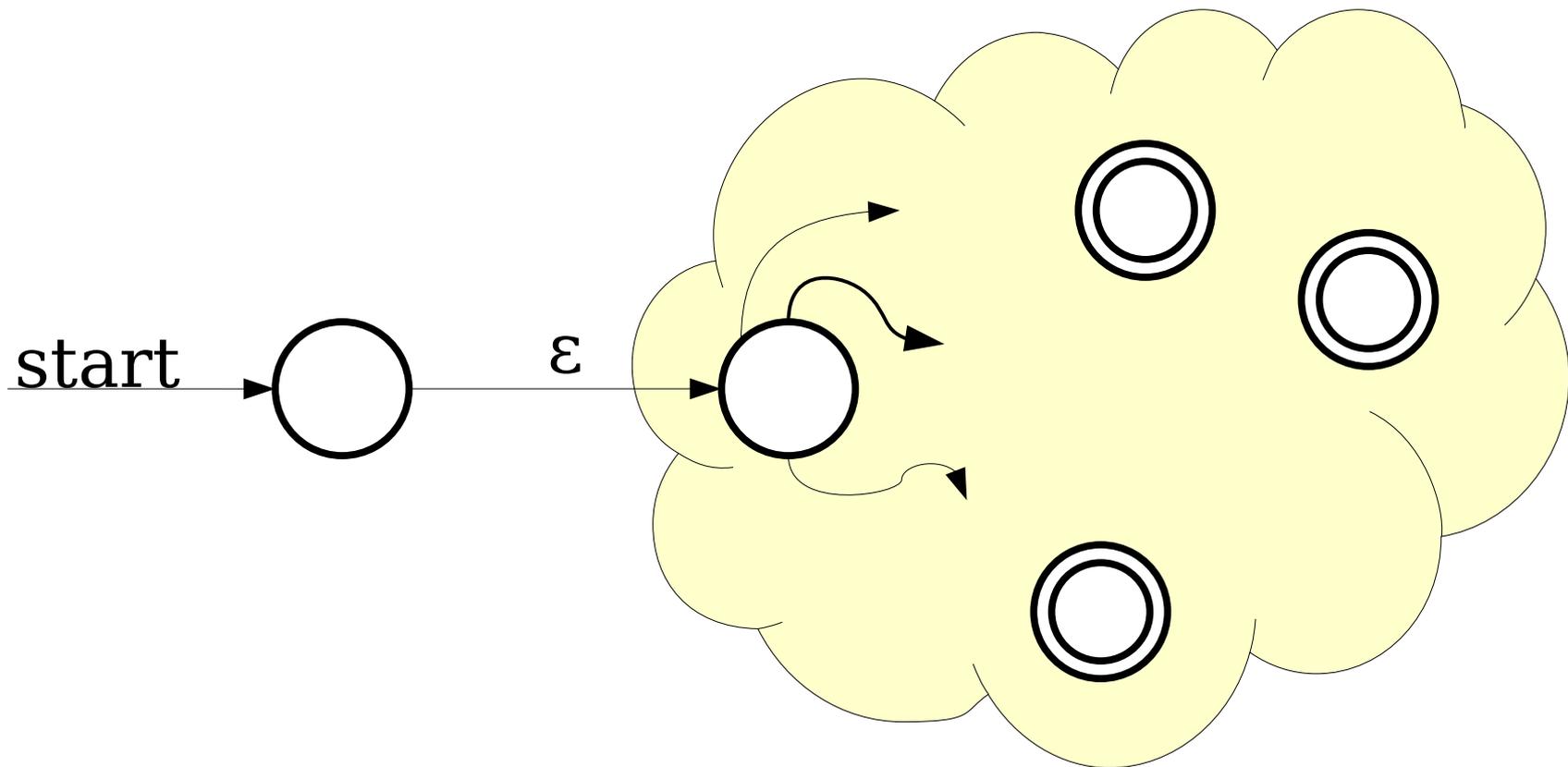
***Idea:*** Can we convert an NFA for a language  $L$  to an NFA for language  $L^*$ ?

# The Kleene Star



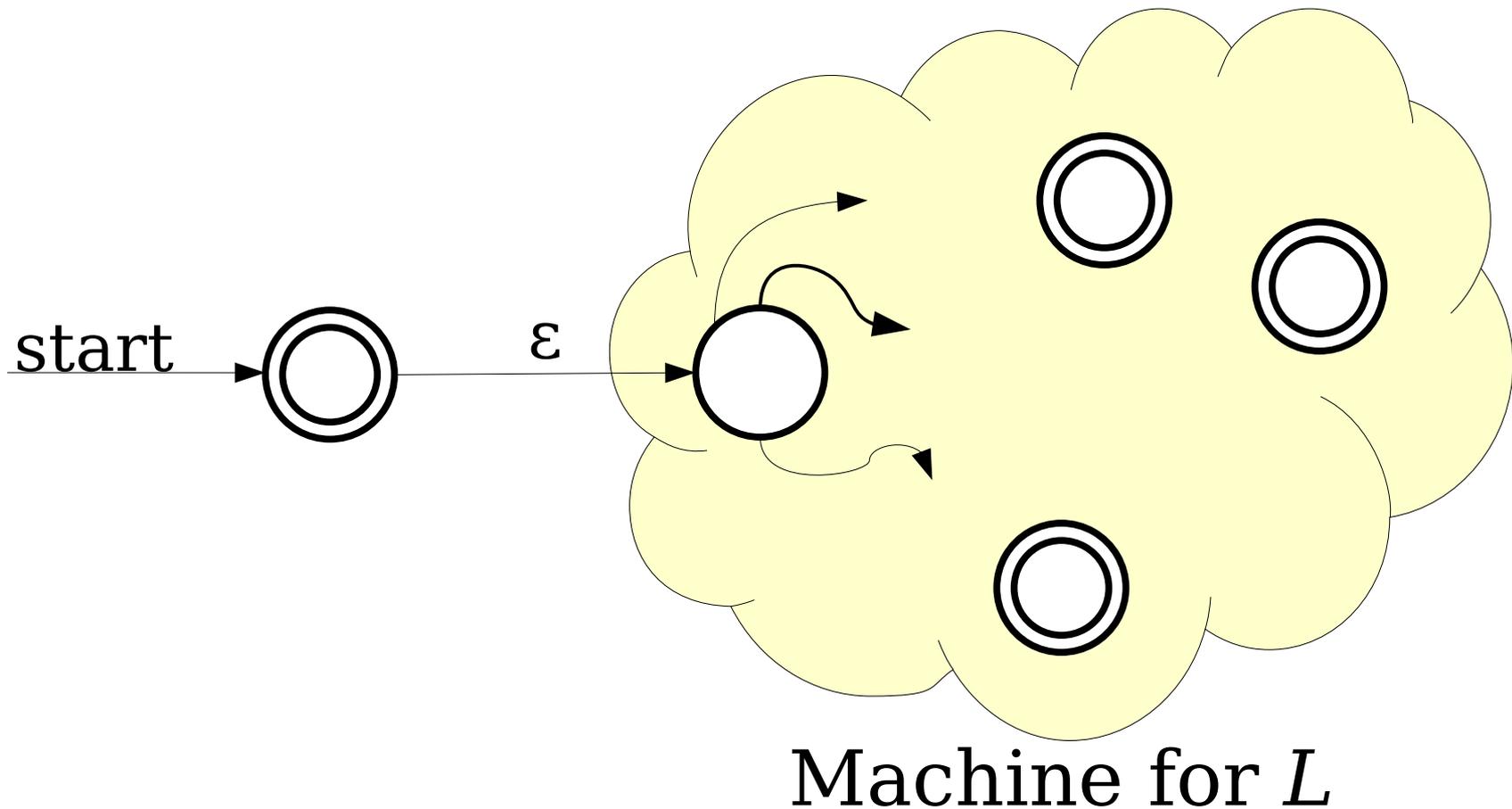
Machine for  $L$

# The Kleene Star

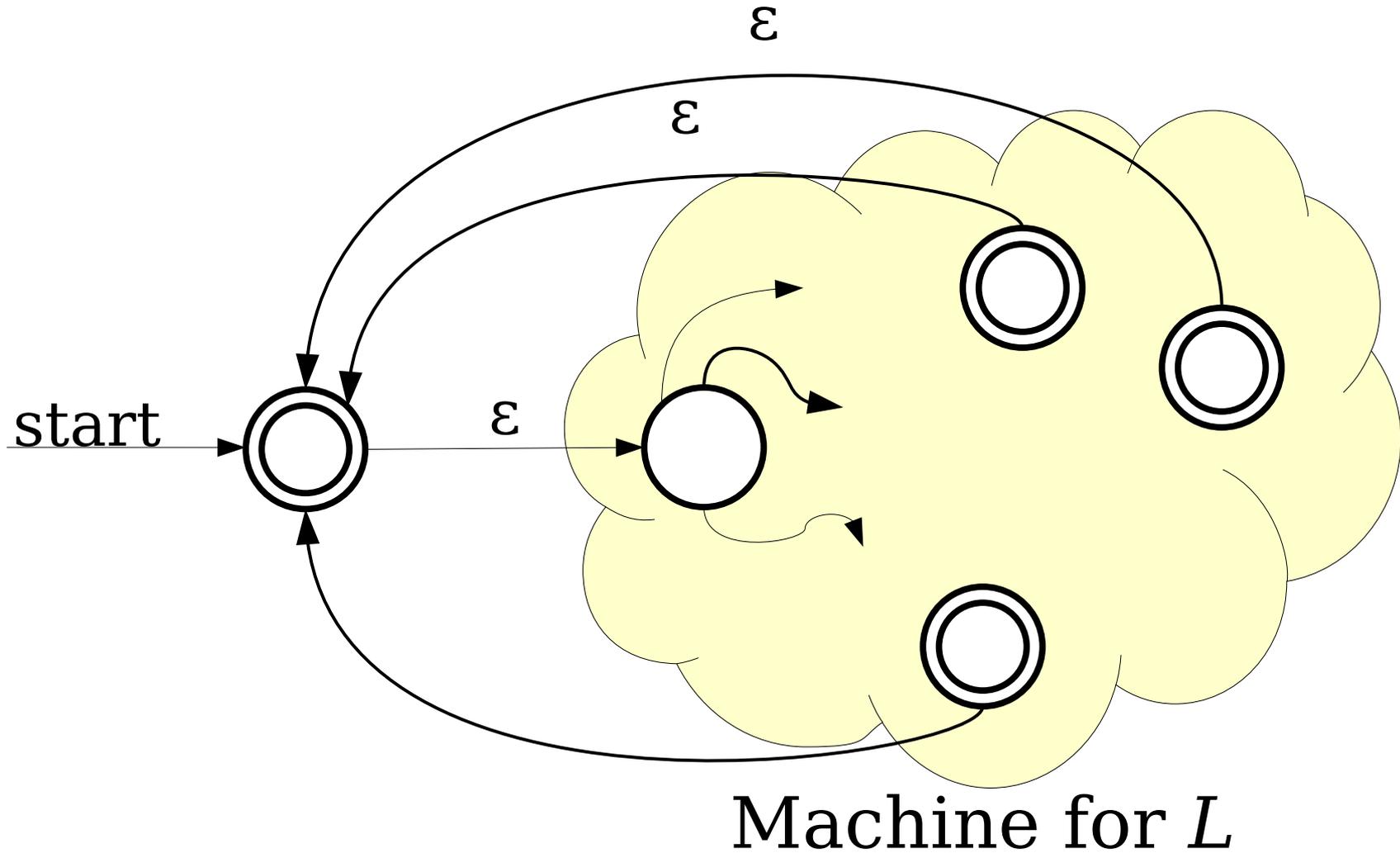


Machine for  $L$

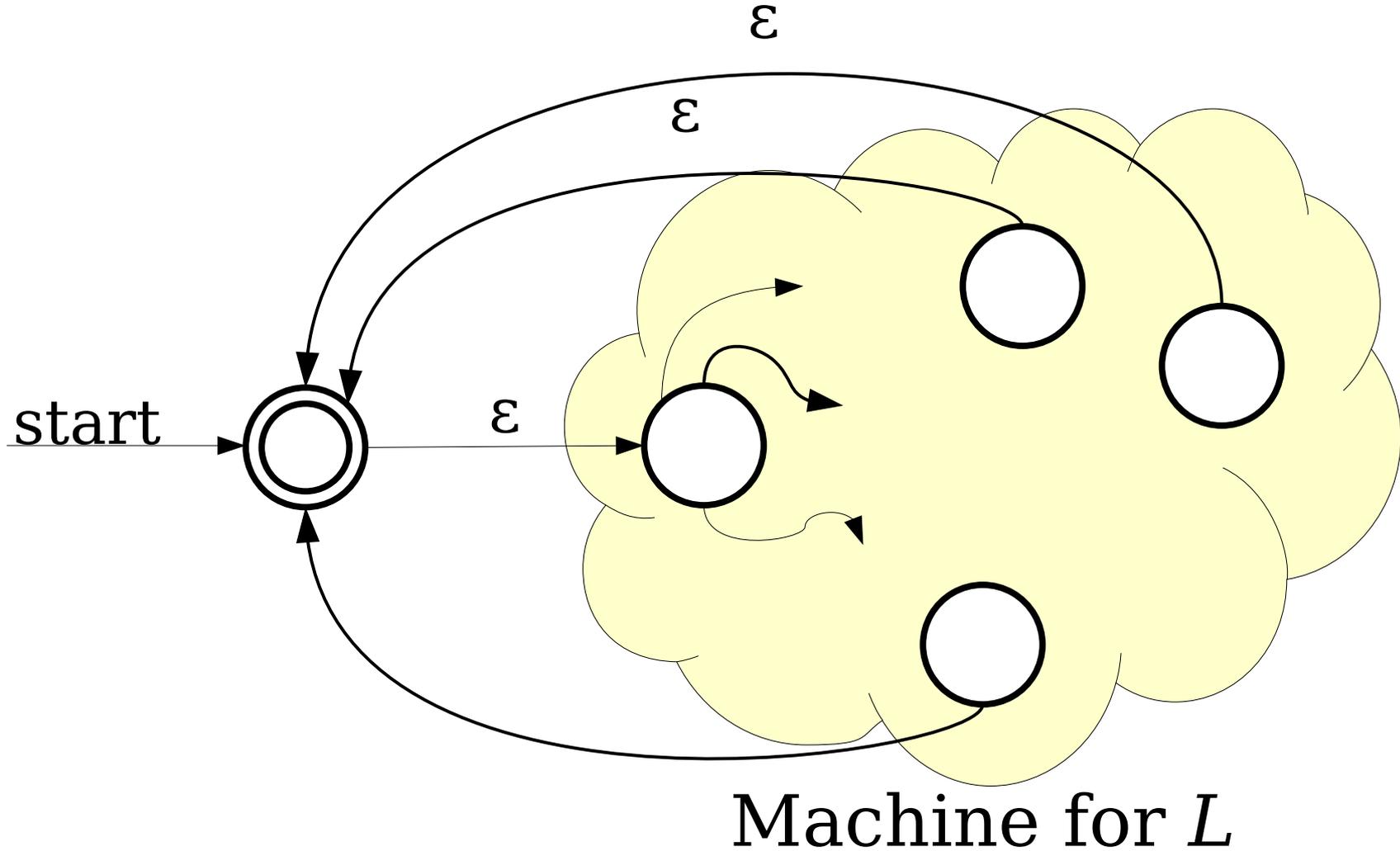
# The Kleene Star



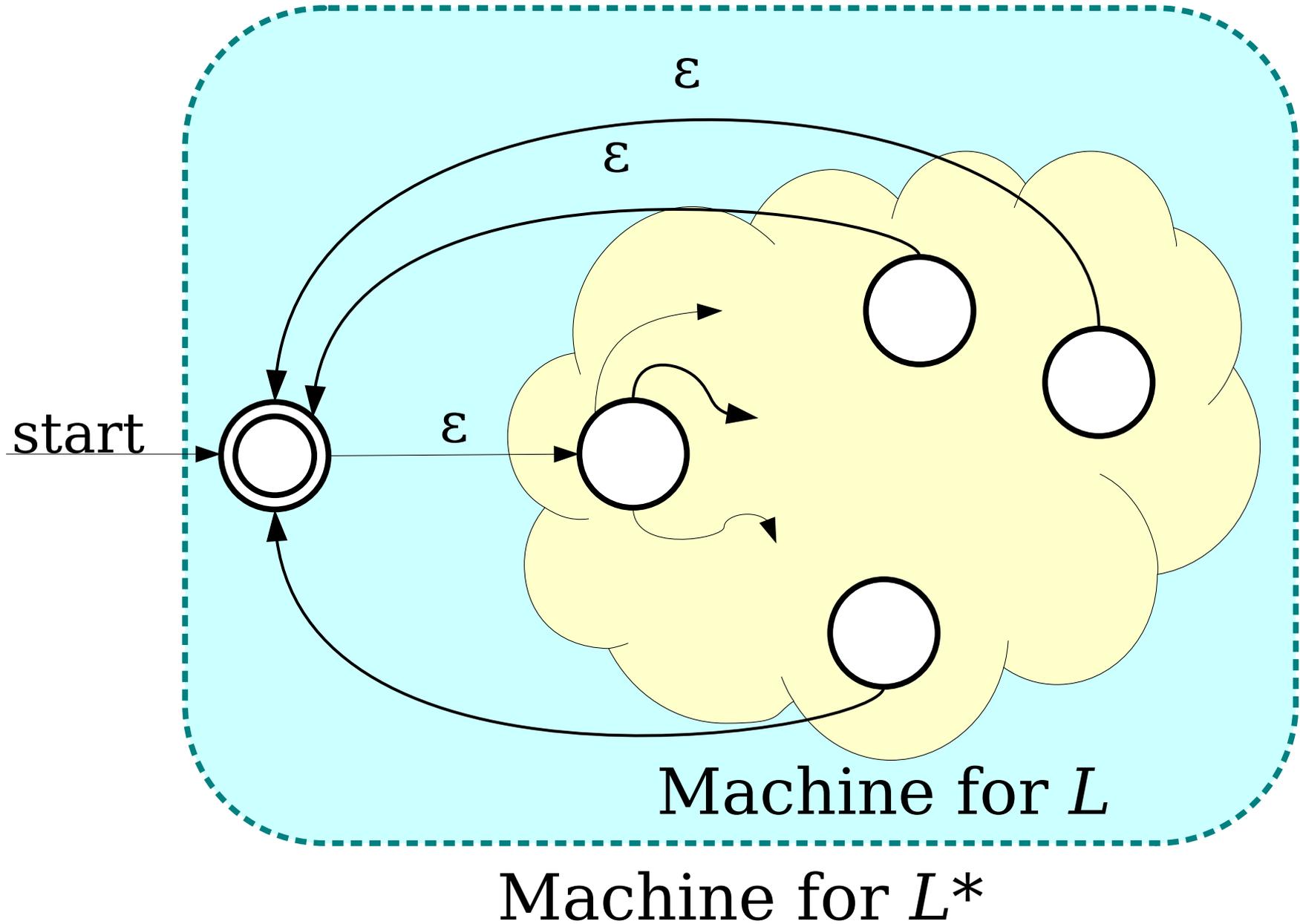
# The Kleene Star



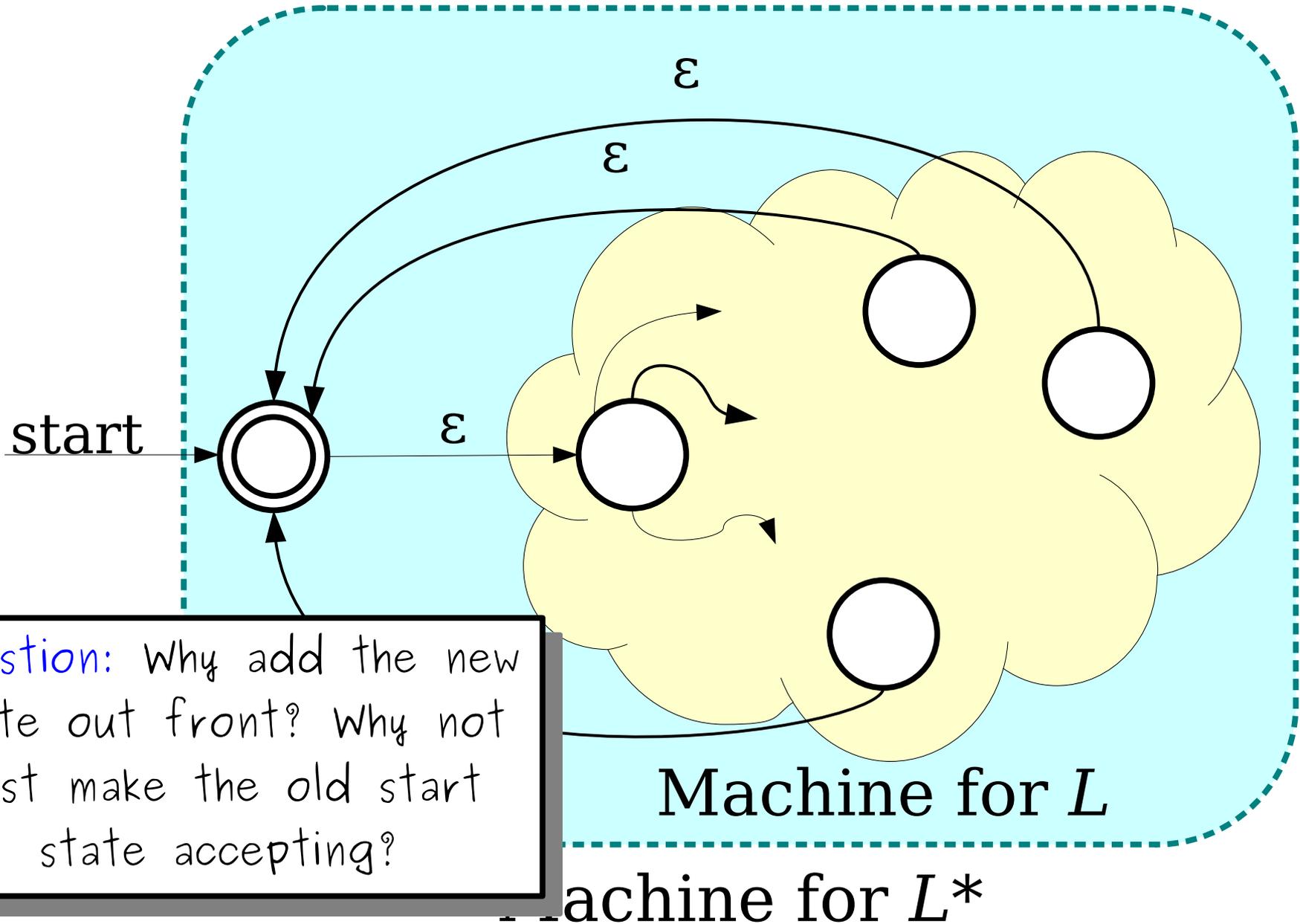
# The Kleene Star



# The Kleene Star



# The Kleene Star



Question: Why add the new state out front? Why not just make the old start state accepting?

# Closure Properties

- ***Theorem:*** If  $L_1$  and  $L_2$  are regular languages over an alphabet  $\Sigma$ , then so are the following languages:
  - $L_1 \cup L_2$
  - $L_1L_2$
  - $L_1^*$
- These properties are called ***closure properties of the regular languages.***

# In the Appendix

- The appendix to this slide deck contains
  - three examples of applying closure properties to concrete languages and how it relates to machines, and
  - two more closure properties beyond what we saw today.
- Take a look – there's some really cool stuff here!

# Next Time

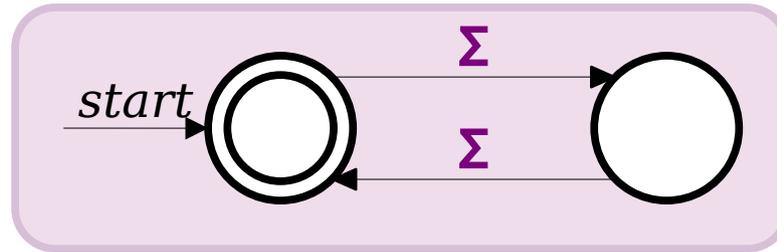
- ***Regular Expressions***
  - Building languages from the ground up!
- ***Thompson's Algorithm***
  - A UNIX Programmer in Theoryland.
- ***Kleene's Theorem***
  - From machines to programs!

## ***Appendix 1:*** Closure Properties Applied

---

$$L_1 = \{ w \in \{a, b\}^* \mid w \text{ has even length} \}$$
$$L_2 = \{ w \in \{a, b\}^* \mid w \text{ has length exactly three} \}$$

Construct an NFA for  $L_1 \cup L_2$ .

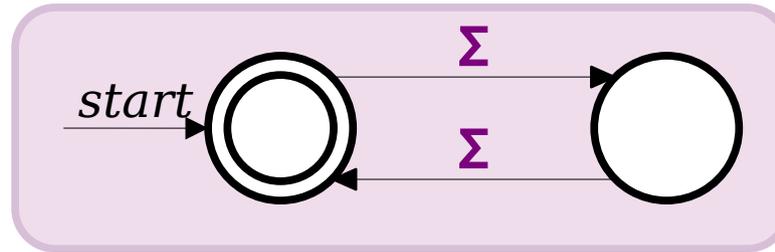


DFA for  $L_1$

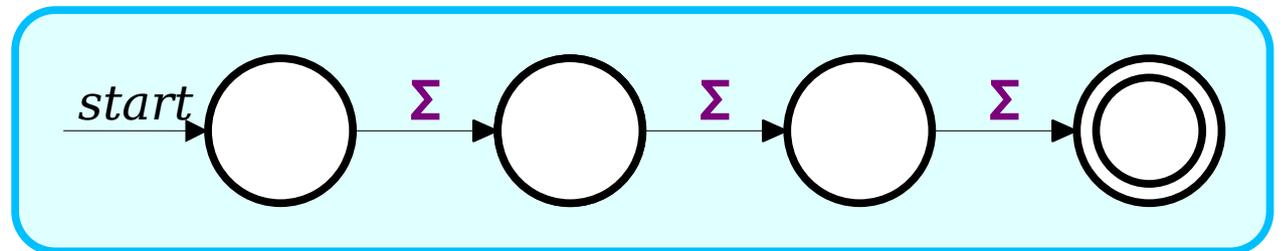
---

$$L_1 = \{ w \in \{a, b\}^* \mid w \text{ has even length} \}$$
$$L_2 = \{ w \in \{a, b\}^* \mid w \text{ has length exactly three} \}$$

Construct an NFA for  $L_1 \cup L_2$ .



DFA for  $L_1$

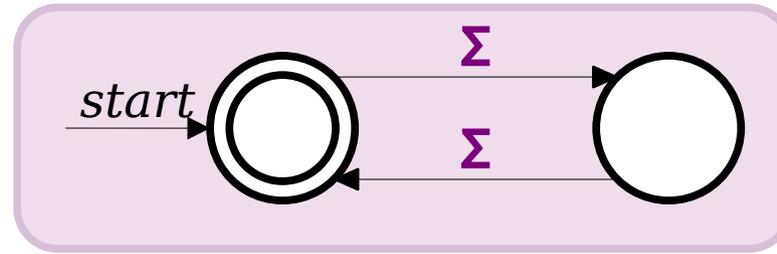
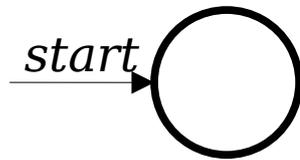


NFA for  $L_2$

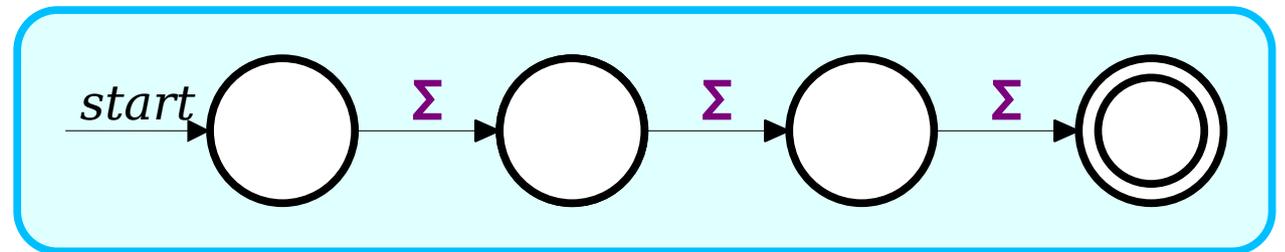
$$L_1 = \{ w \in \{a, b\}^* \mid w \text{ has even length} \}$$

$$L_2 = \{ w \in \{a, b\}^* \mid w \text{ has length exactly three} \}$$

Construct an NFA for  $L_1 \cup L_2$ .



DFA for  $L_1$

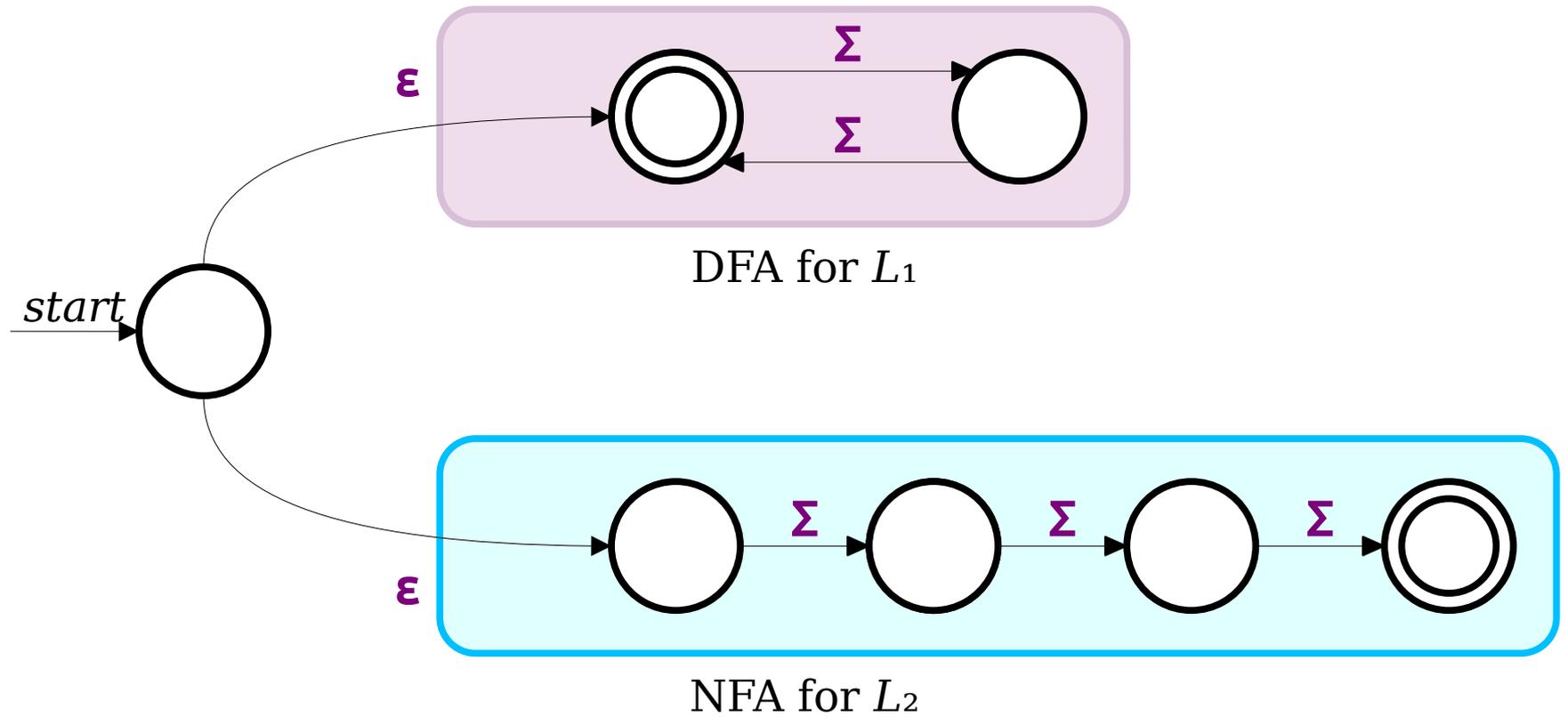


NFA for  $L_2$

$$L_1 = \{ w \in \{a, b\}^* \mid w \text{ has even length} \}$$

$$L_2 = \{ w \in \{a, b\}^* \mid w \text{ has length exactly three} \}$$

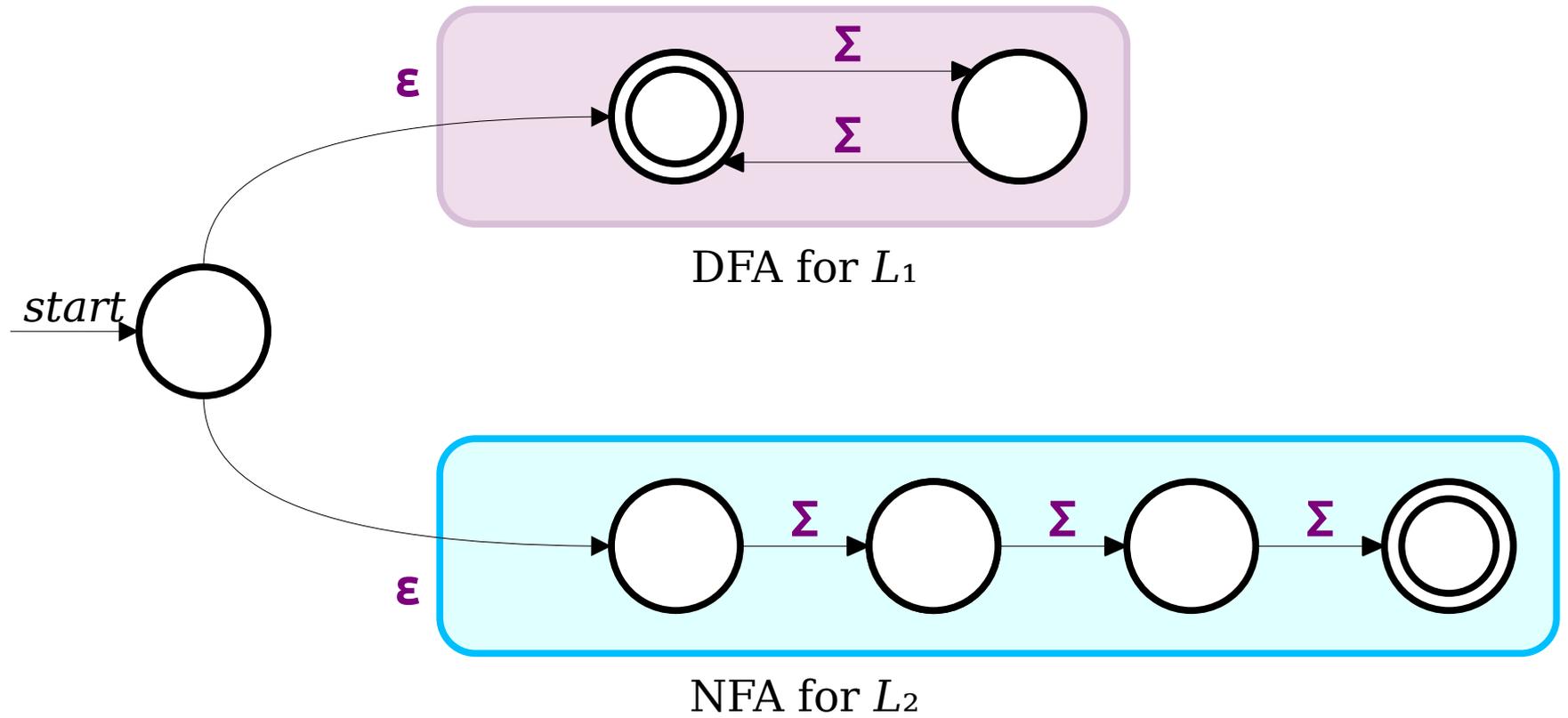
Construct an NFA for  $L_1 \cup L_2$ .



$$L_1 = \{ w \in \{a, b\}^* \mid w \text{ has even length} \}$$

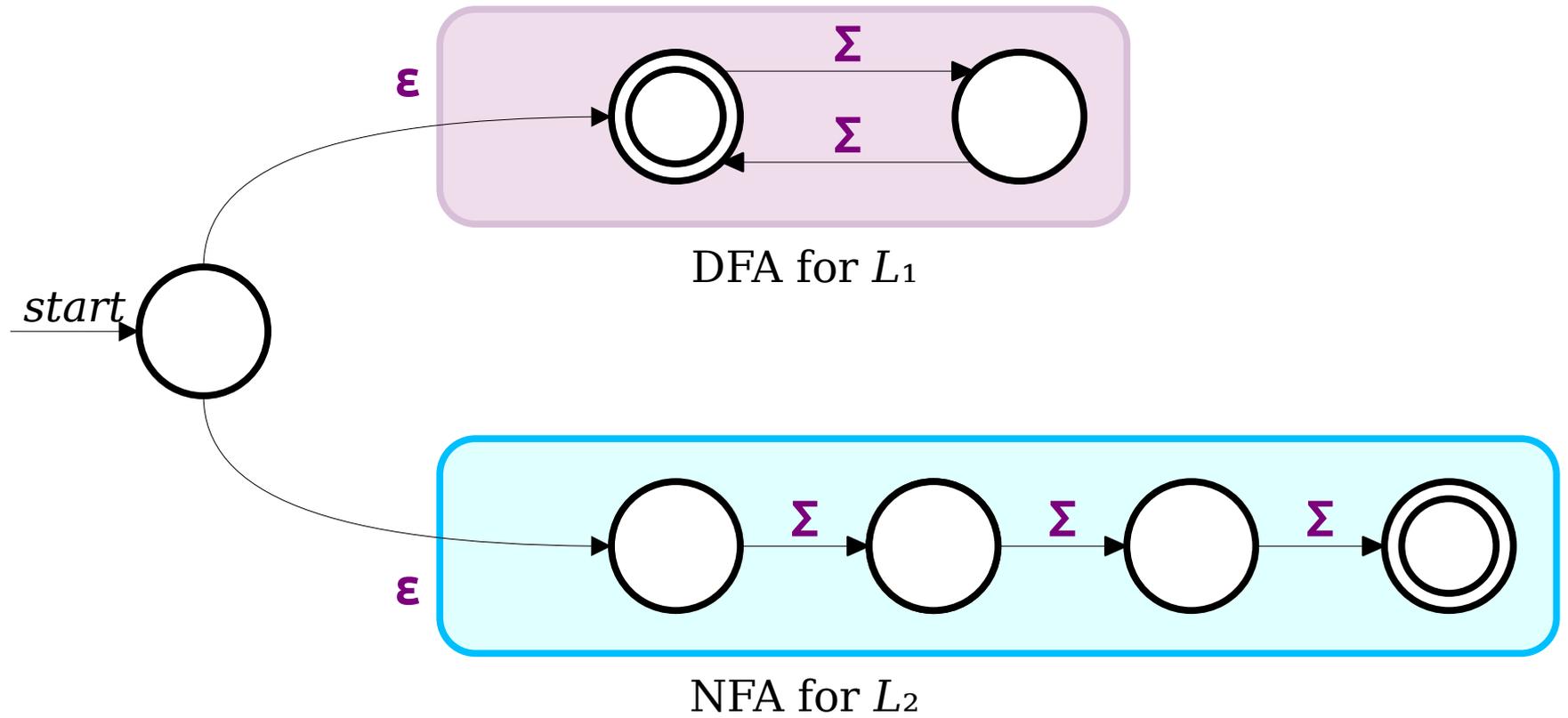
$$L_2 = \{ w \in \{a, b\}^* \mid w \text{ has length exactly three} \}$$

Construct an NFA for  $L_1 \cup L_2$ .

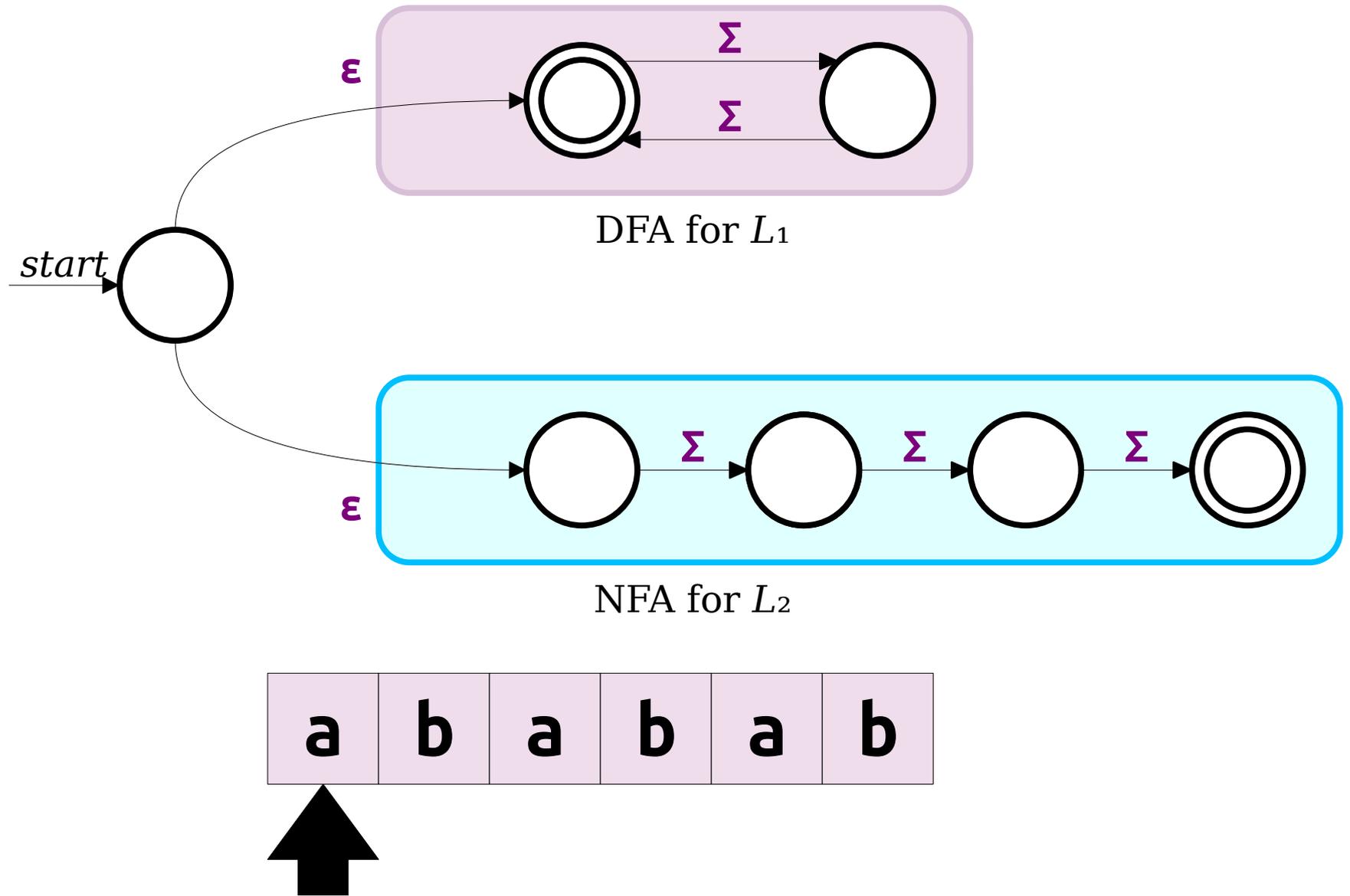


$L_1 = \{ w \in \{a, b\}^* \mid w \text{ has even length} \}$   
 $L_2 = \{ w \in \{a, b\}^* \mid w \text{ has length exactly three} \}$

Construct an NFA for  $L_1 \cup L_2$ .



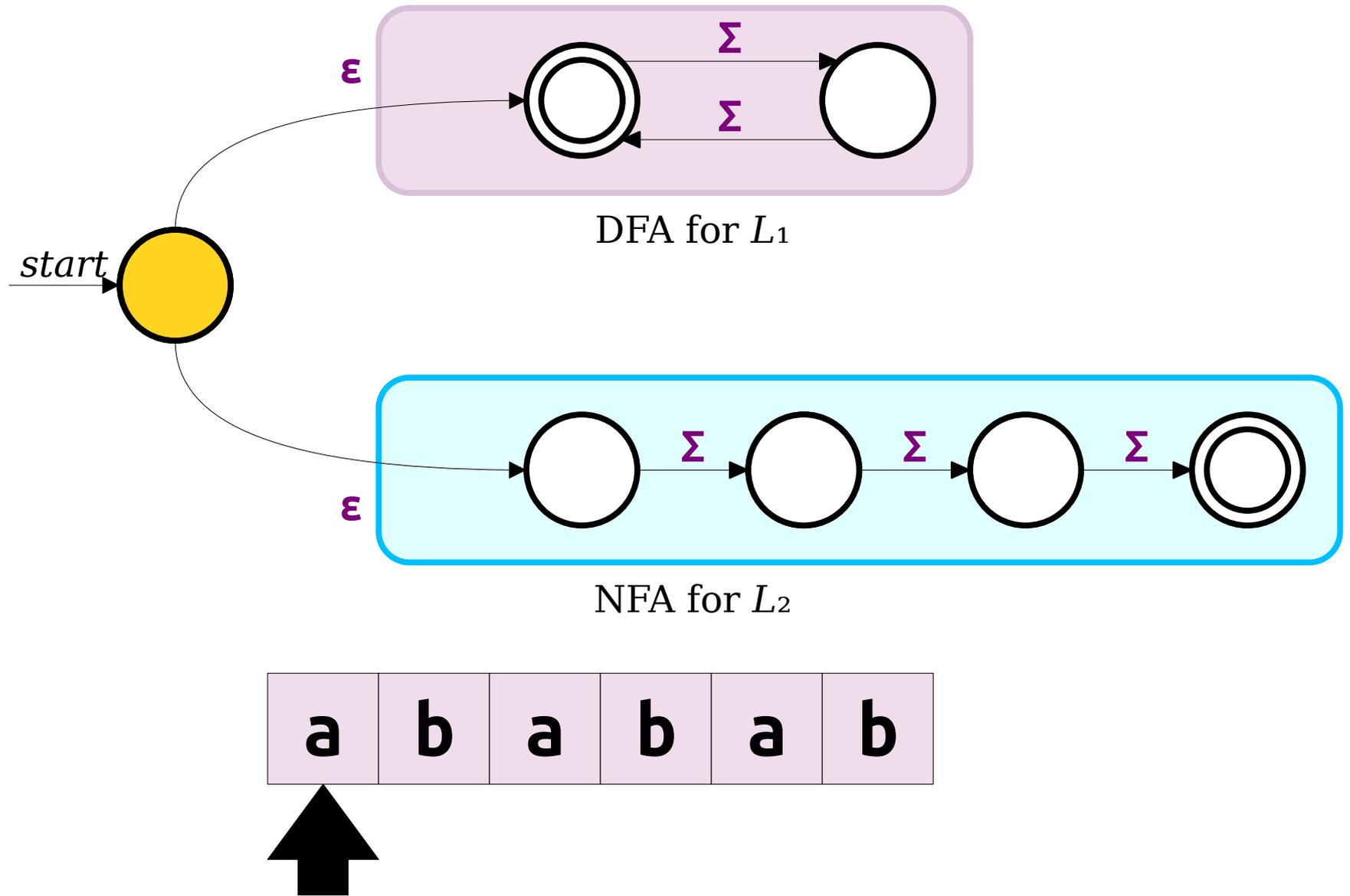
$L_1 = \{ w \in \{a, b\}^* \mid w \text{ has even length} \}$   
 $L_2 = \{ w \in \{a, b\}^* \mid w \text{ has length exactly three} \}$   
 Construct an NFA for  $L_1 \cup L_2$ .



$$L_1 = \{ w \in \{a, b\}^* \mid w \text{ has even length} \}$$

$$L_2 = \{ w \in \{a, b\}^* \mid w \text{ has length exactly three} \}$$

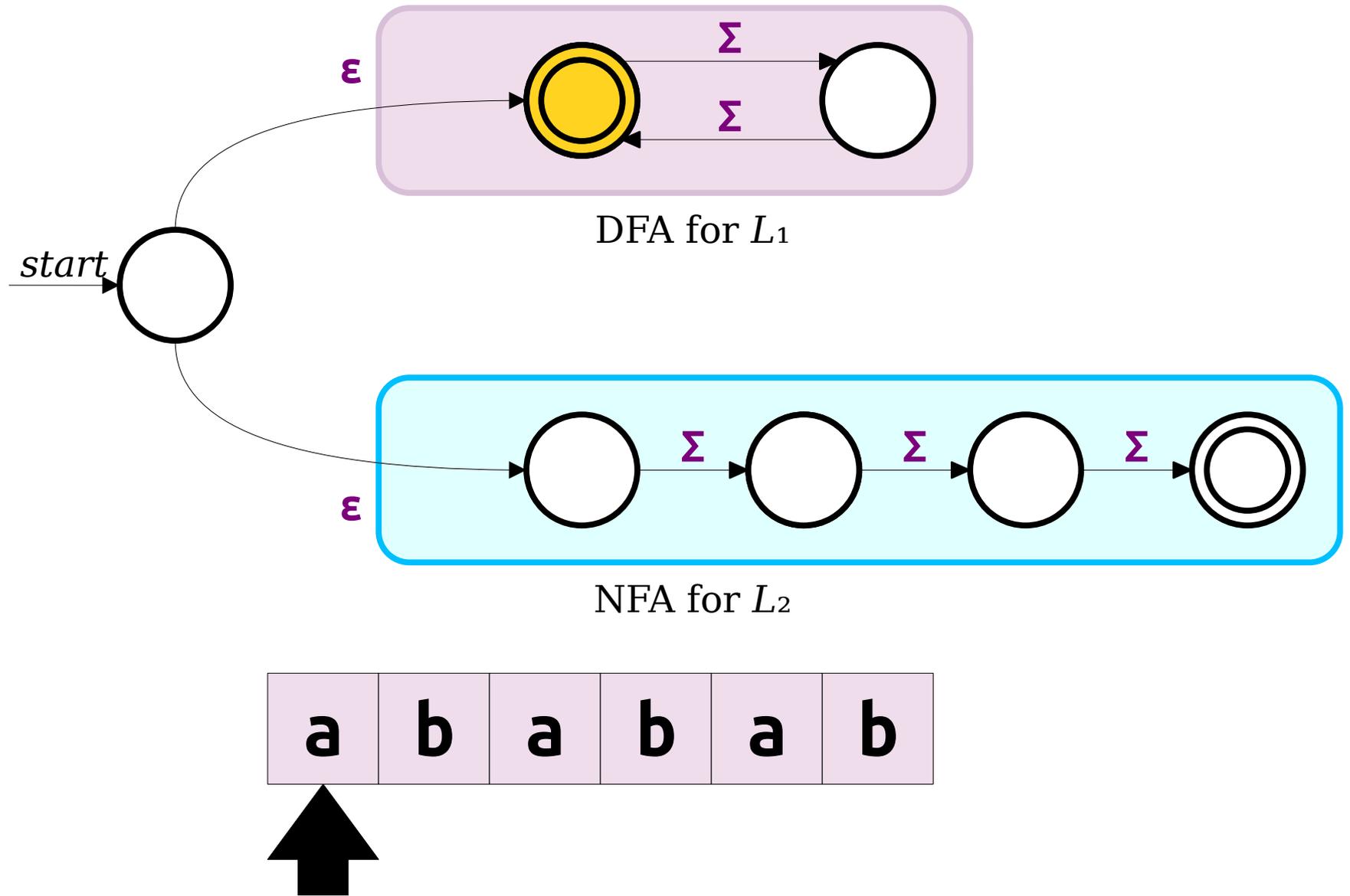
Construct an NFA for  $L_1 \cup L_2$ .



$$L_1 = \{ w \in \{a, b\}^* \mid w \text{ has even length} \}$$

$$L_2 = \{ w \in \{a, b\}^* \mid w \text{ has length exactly three} \}$$

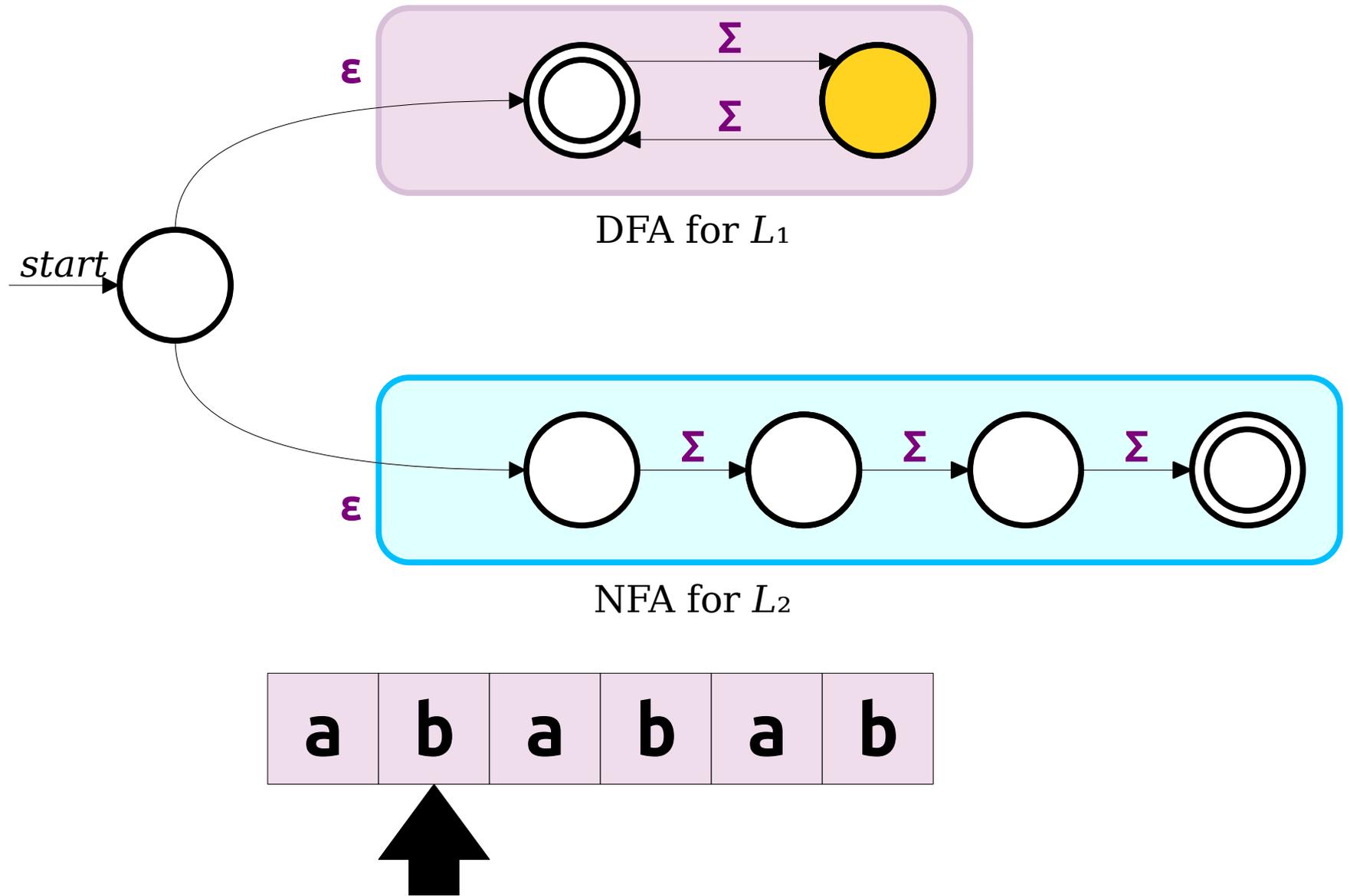
Construct an NFA for  $L_1 \cup L_2$ .



$$L_1 = \{ w \in \{a, b\}^* \mid w \text{ has even length} \}$$

$$L_2 = \{ w \in \{a, b\}^* \mid w \text{ has length exactly three} \}$$

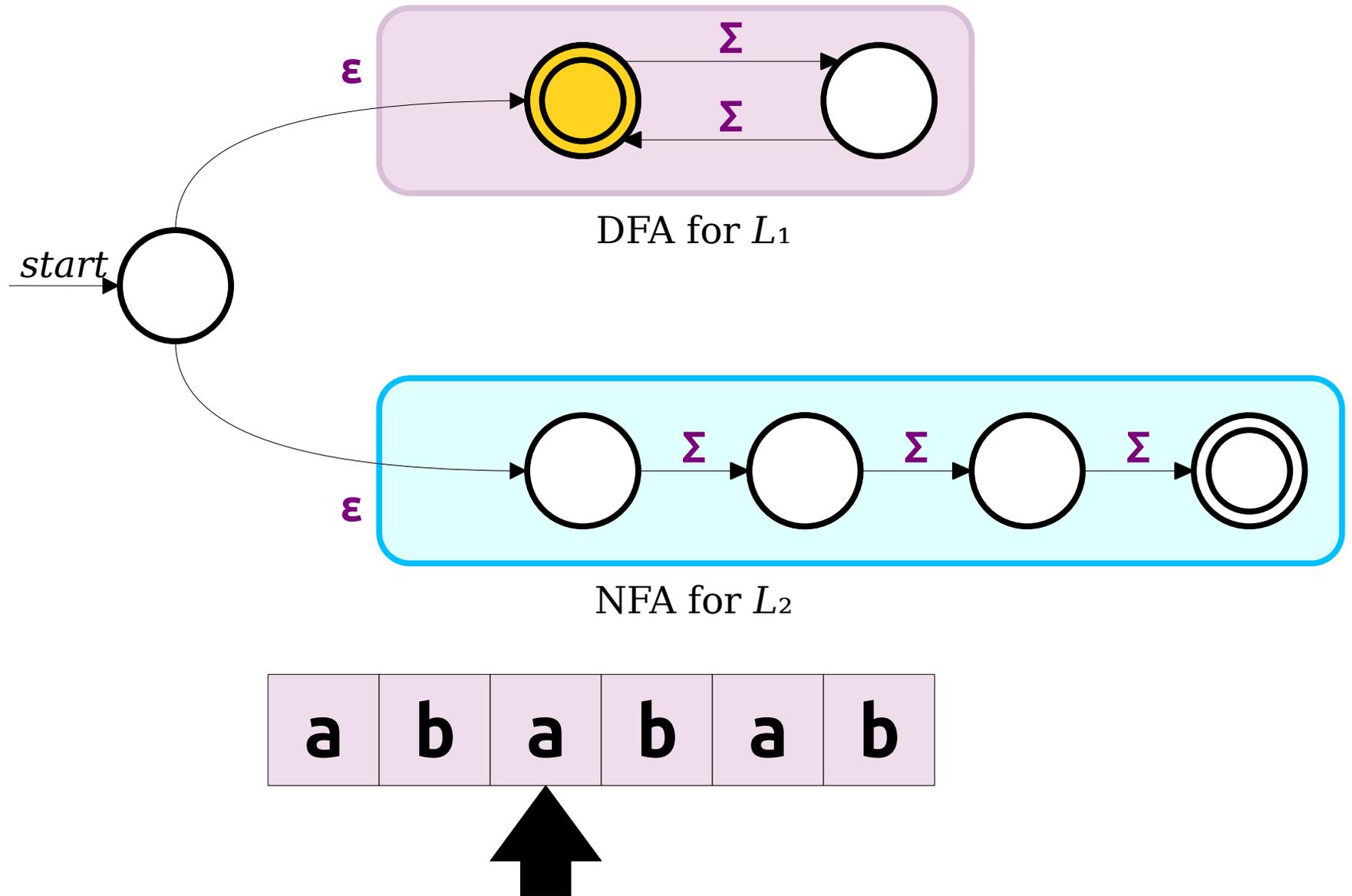
Construct an NFA for  $L_1 \cup L_2$ .



$$L_1 = \{ w \in \{a, b\}^* \mid w \text{ has even length} \}$$

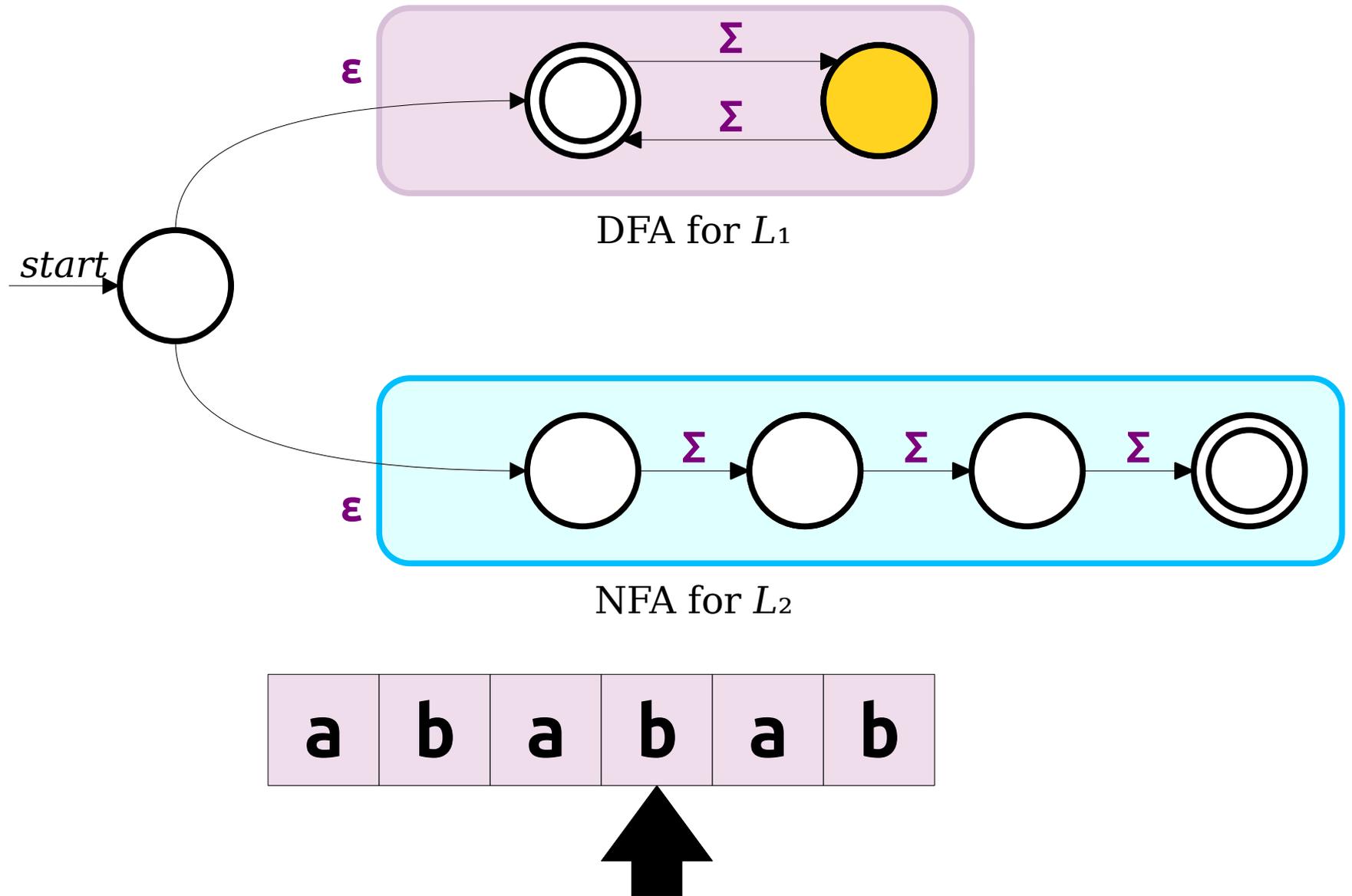
$$L_2 = \{ w \in \{a, b\}^* \mid w \text{ has length exactly three} \}$$

Construct an NFA for  $L_1 \cup L_2$ .



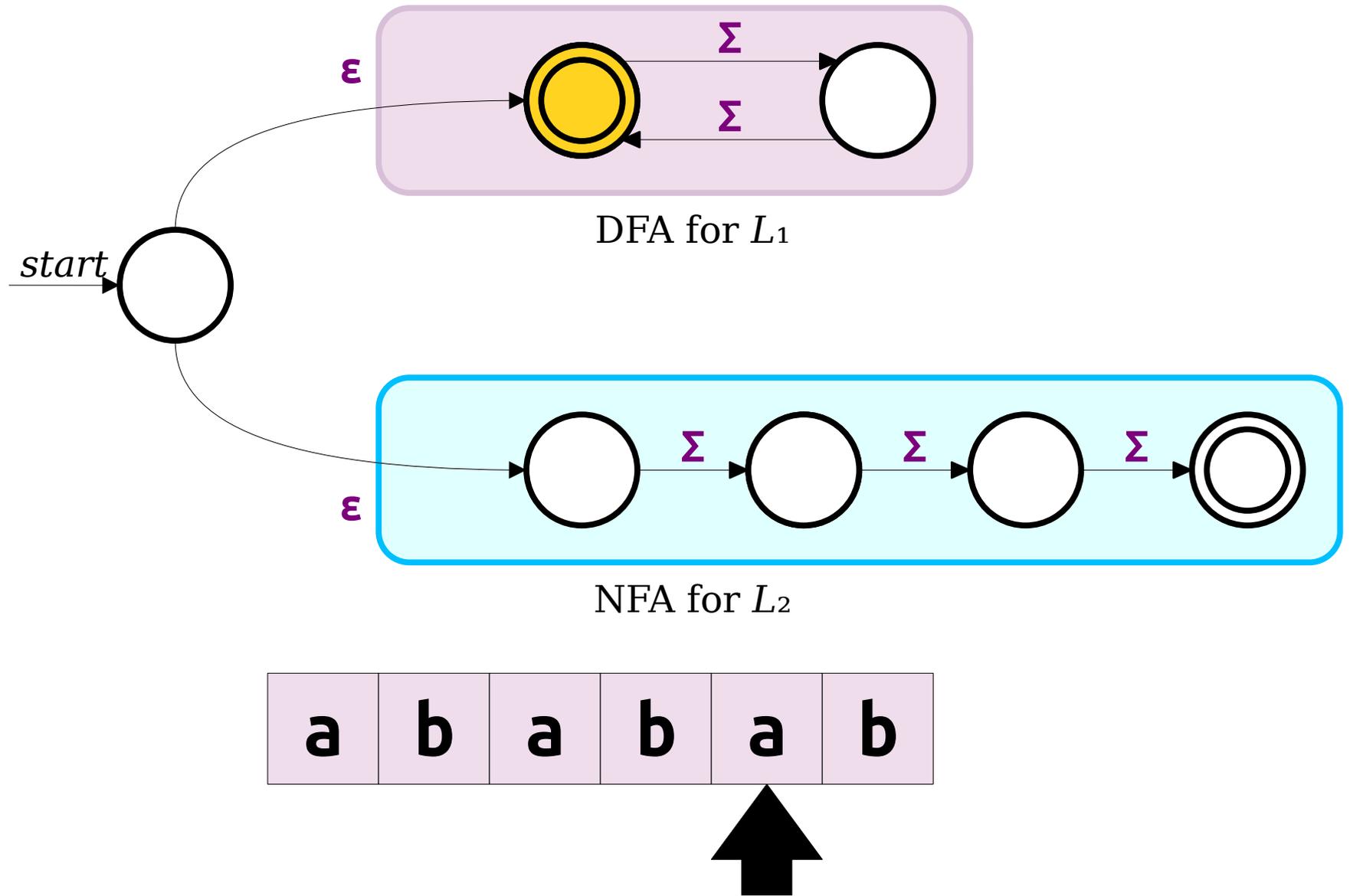
$L_1 = \{ w \in \{a, b\}^* \mid w \text{ has even length} \}$   
 $L_2 = \{ w \in \{a, b\}^* \mid w \text{ has length exactly three} \}$

Construct an NFA for  $L_1 \cup L_2$ .



$L_1 = \{ w \in \{a, b\}^* \mid w \text{ has even length} \}$   
 $L_2 = \{ w \in \{a, b\}^* \mid w \text{ has length exactly three} \}$

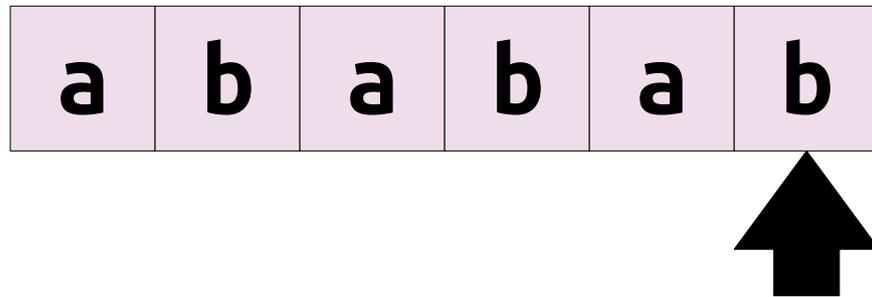
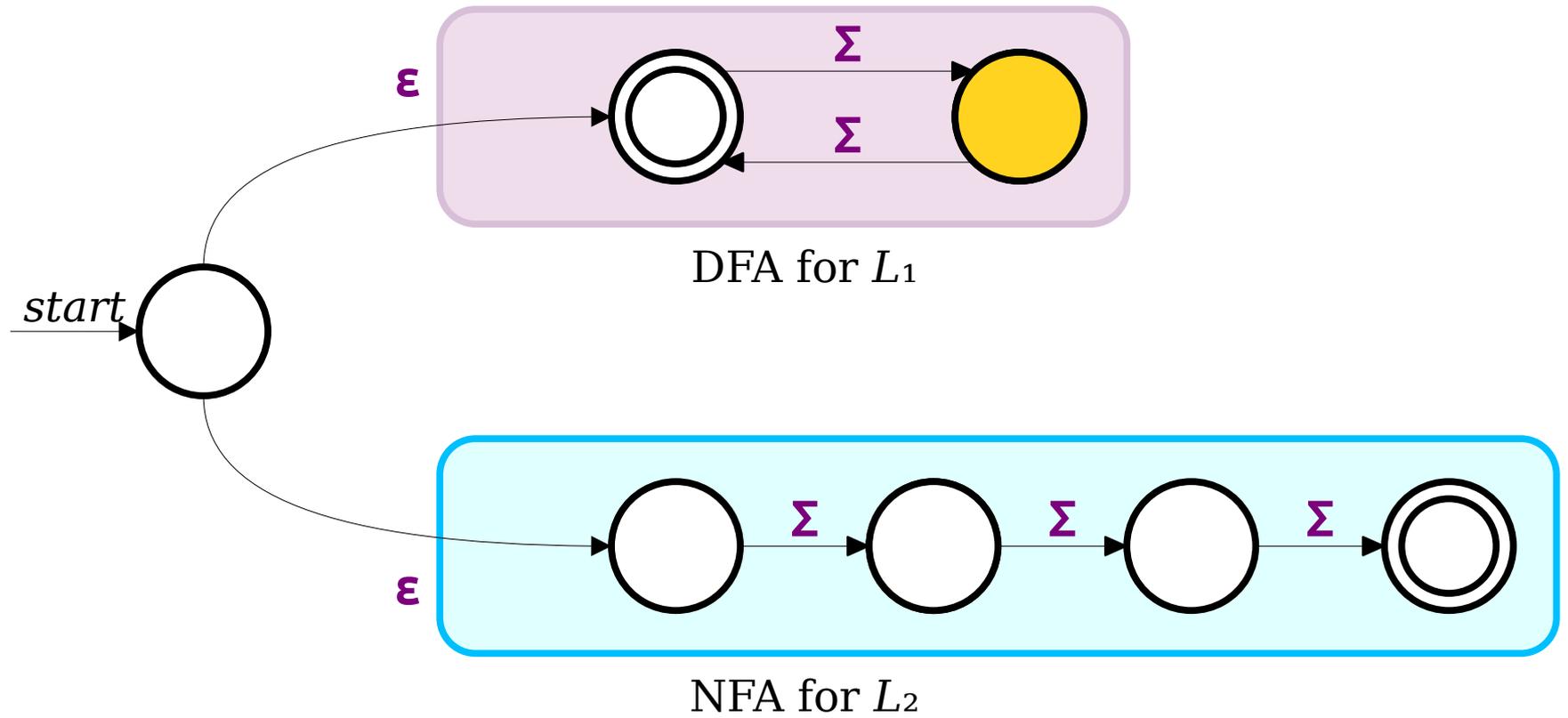
Construct an NFA for  $L_1 \cup L_2$ .



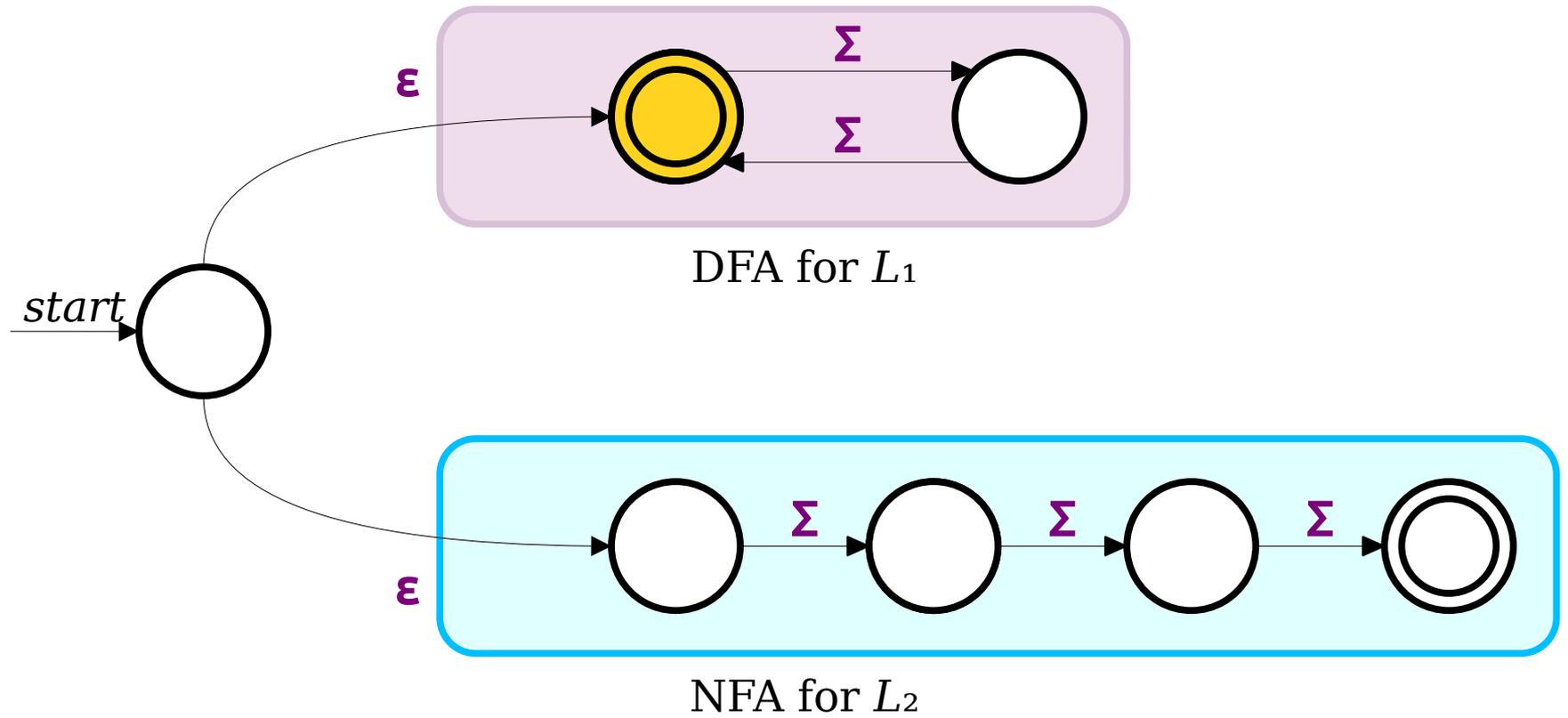
$$L_1 = \{ w \in \{a, b\}^* \mid w \text{ has even length} \}$$

$$L_2 = \{ w \in \{a, b\}^* \mid w \text{ has length exactly three} \}$$

Construct an NFA for  $L_1 \cup L_2$ .



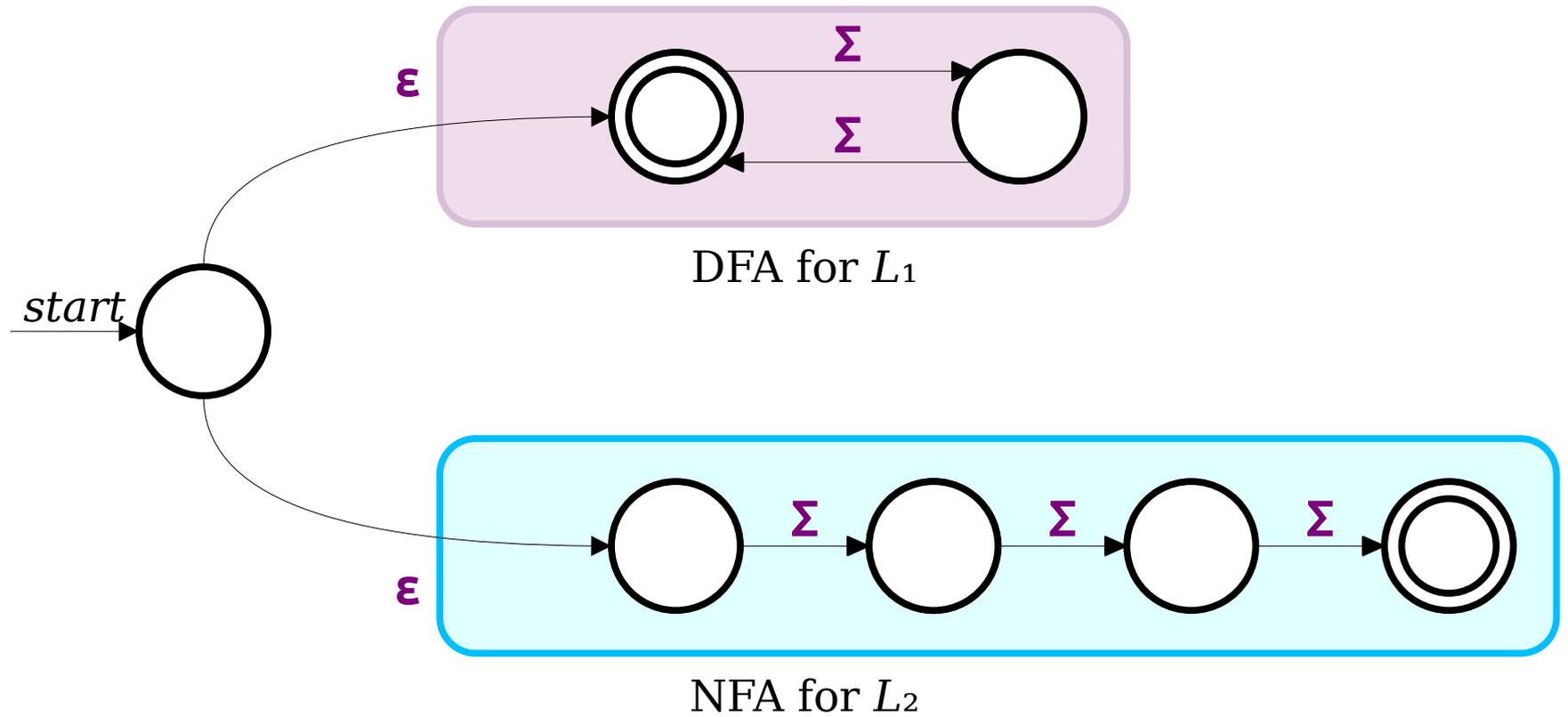
$L_1 = \{ w \in \{a, b\}^* \mid w \text{ has even length} \}$   
 $L_2 = \{ w \in \{a, b\}^* \mid w \text{ has length exactly three} \}$   
 Construct an NFA for  $L_1 \cup L_2$ .



$$L_1 = \{ w \in \{a, b\}^* \mid w \text{ has even length} \}$$

$$L_2 = \{ w \in \{a, b\}^* \mid w \text{ has length exactly three} \}$$

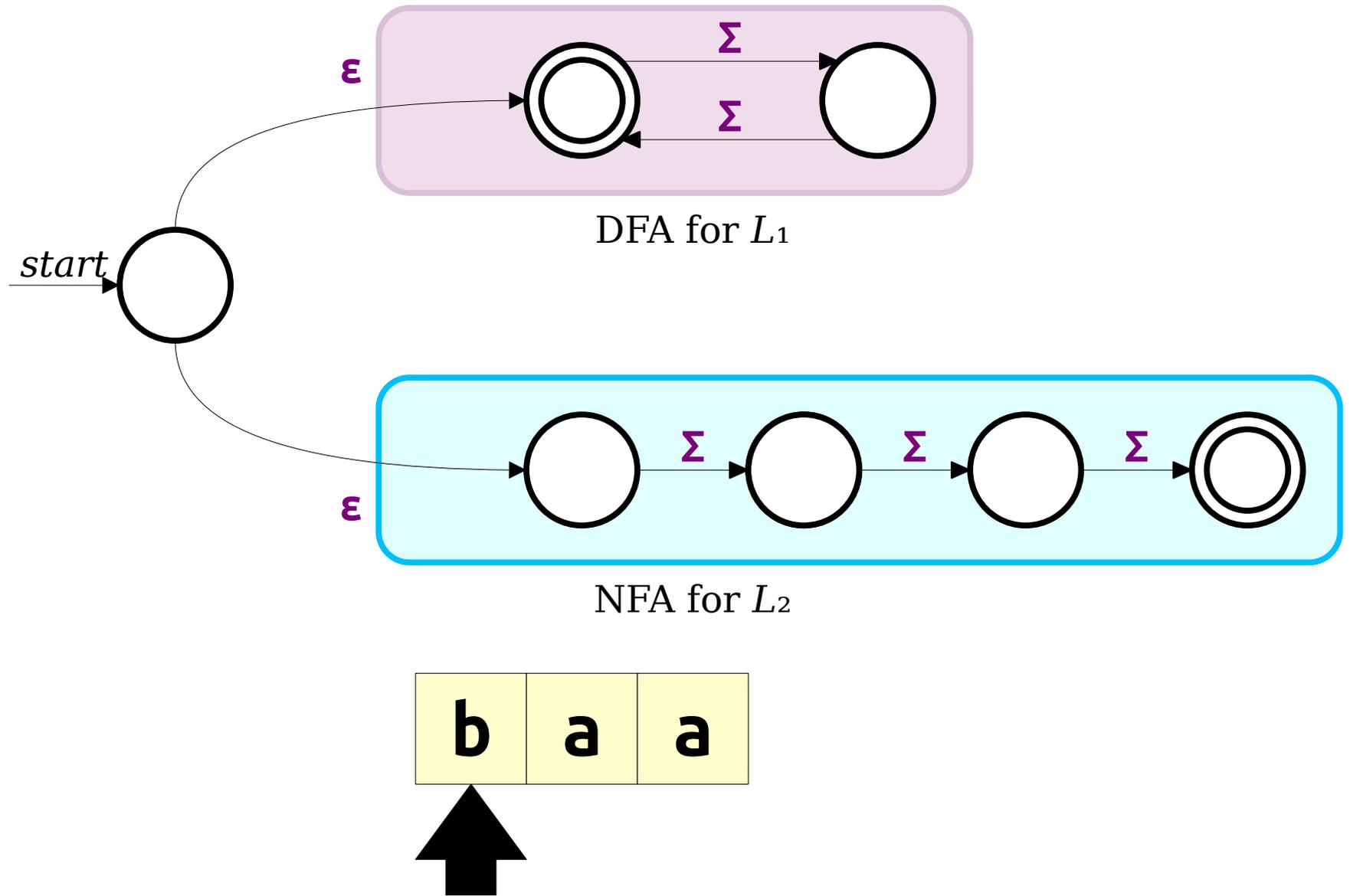
Construct an NFA for  $L_1 \cup L_2$ .



$$L_1 = \{ w \in \{a, b\}^* \mid w \text{ has even length} \}$$

$$L_2 = \{ w \in \{a, b\}^* \mid w \text{ has length exactly three} \}$$

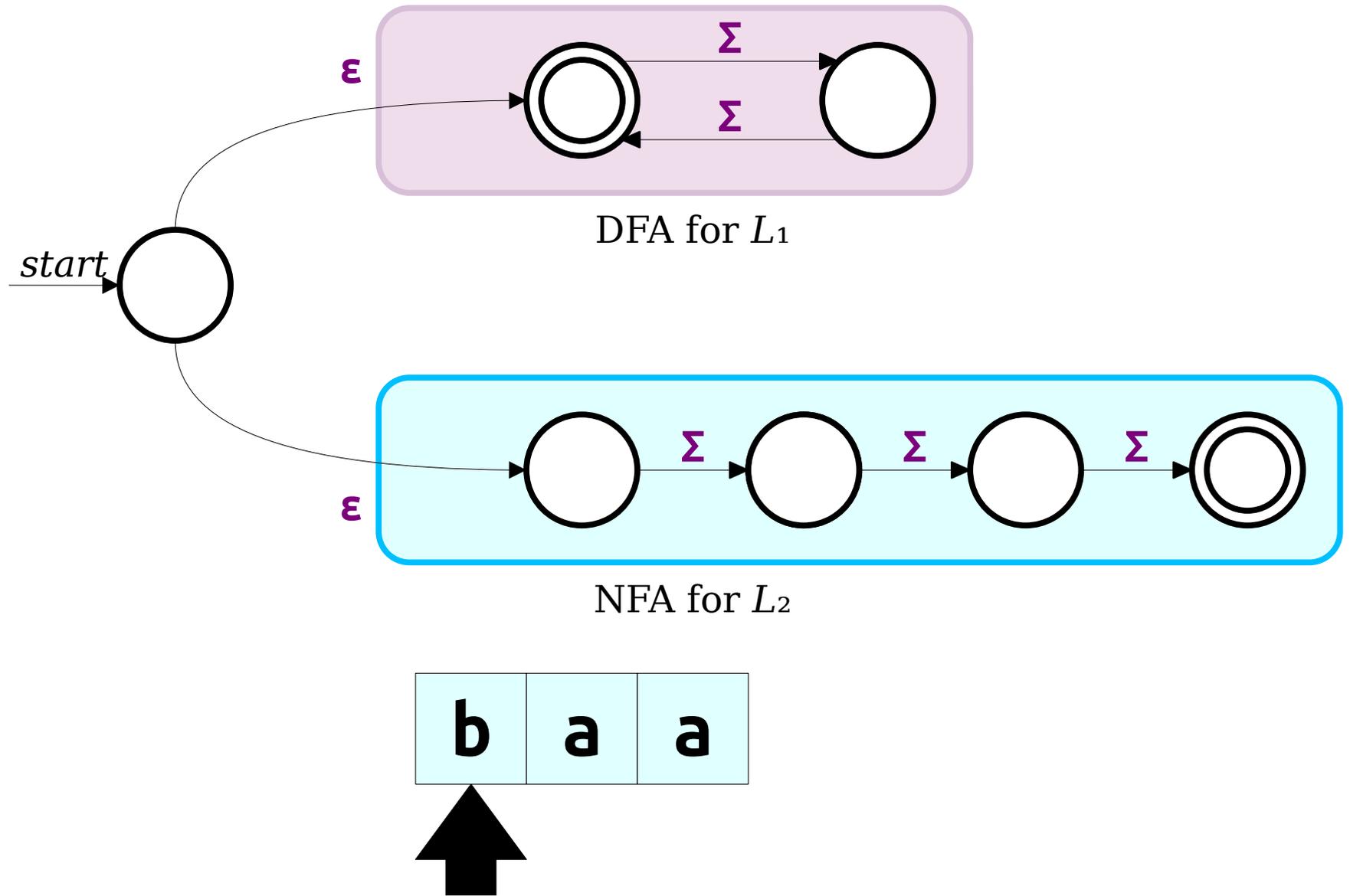
Construct an NFA for  $L_1 \cup L_2$ .



$$L_1 = \{ w \in \{a, b\}^* \mid w \text{ has even length} \}$$

$$L_2 = \{ w \in \{a, b\}^* \mid w \text{ has length exactly three} \}$$

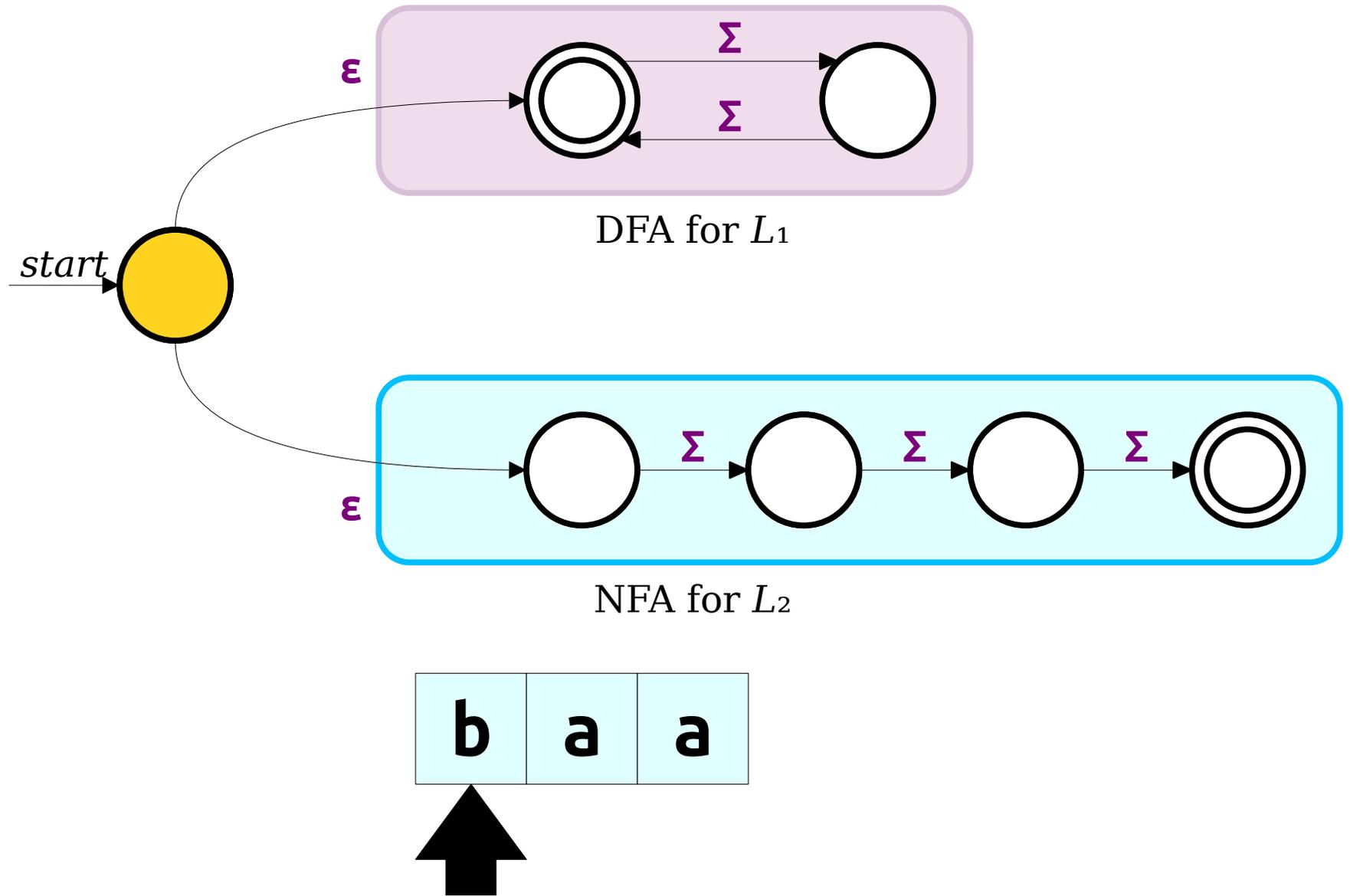
Construct an NFA for  $L_1 \cup L_2$ .



$$L_1 = \{ w \in \{a, b\}^* \mid w \text{ has even length} \}$$

$$L_2 = \{ w \in \{a, b\}^* \mid w \text{ has length exactly three} \}$$

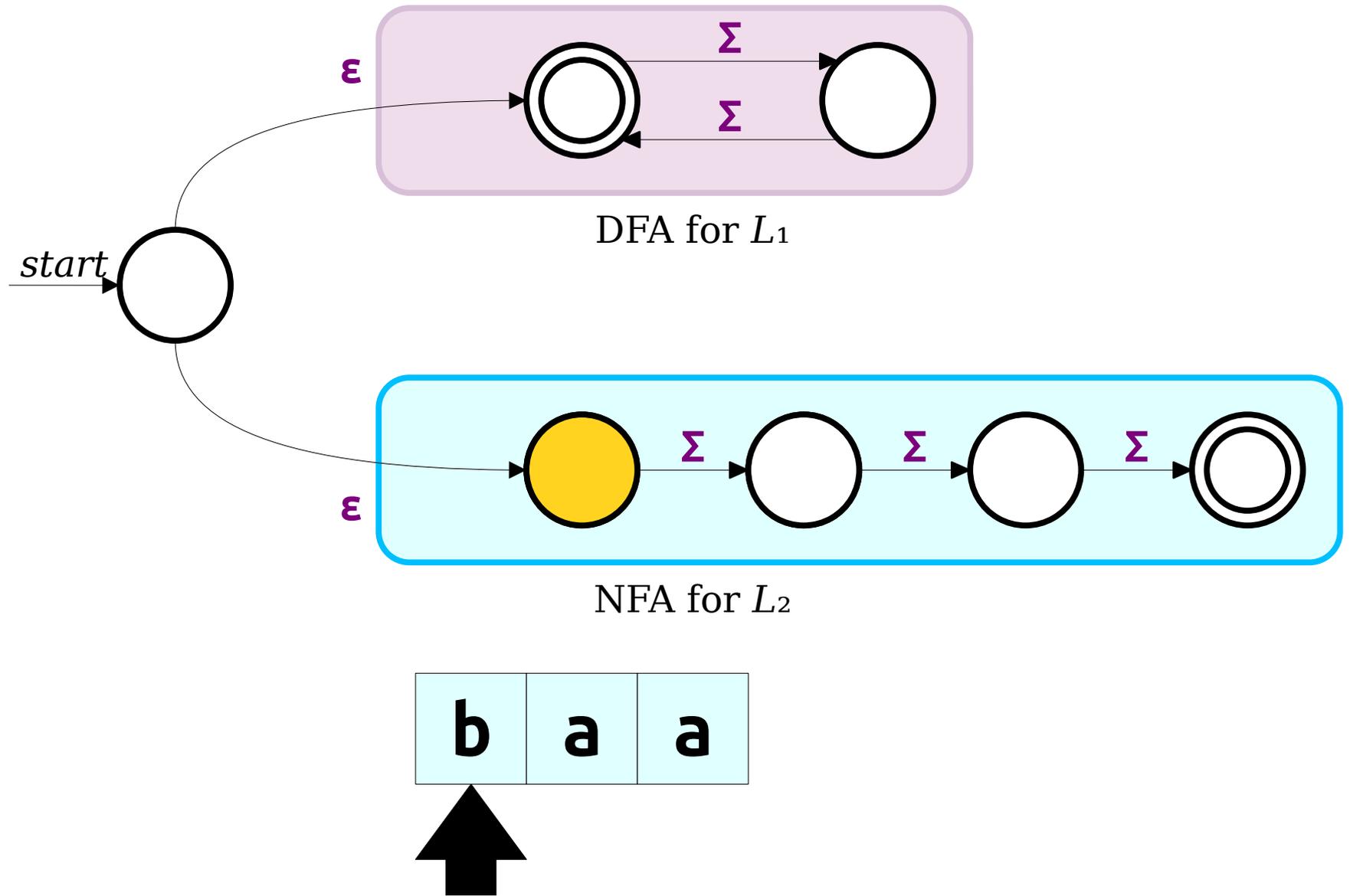
Construct an NFA for  $L_1 \cup L_2$ .



$$L_1 = \{ w \in \{a, b\}^* \mid w \text{ has even length} \}$$

$$L_2 = \{ w \in \{a, b\}^* \mid w \text{ has length exactly three} \}$$

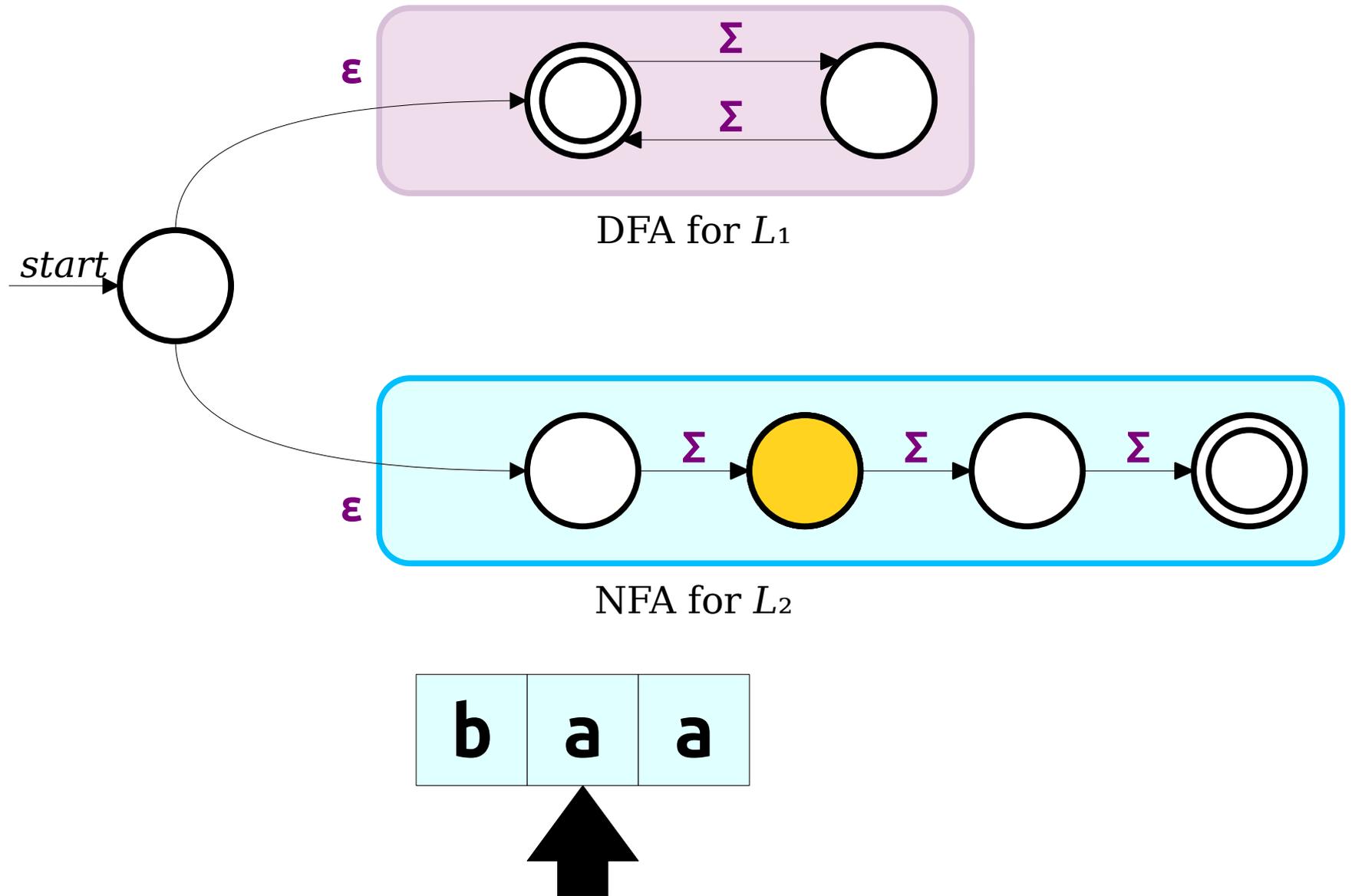
Construct an NFA for  $L_1 \cup L_2$ .



$$L_1 = \{ w \in \{a, b\}^* \mid w \text{ has even length} \}$$

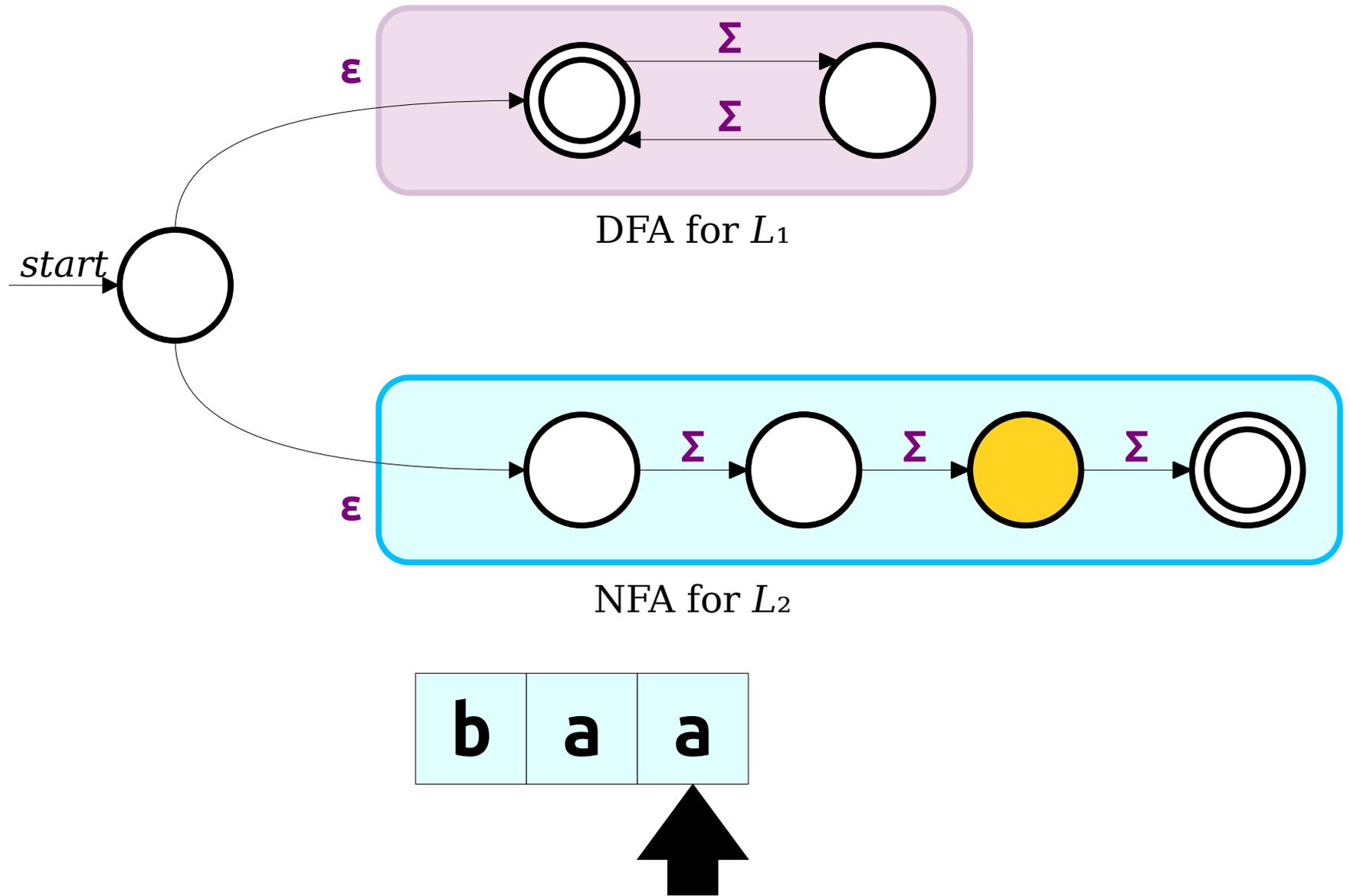
$$L_2 = \{ w \in \{a, b\}^* \mid w \text{ has length exactly three} \}$$

Construct an NFA for  $L_1 \cup L_2$ .



$L_1 = \{ w \in \{a, b\}^* \mid w \text{ has even length} \}$   
 $L_2 = \{ w \in \{a, b\}^* \mid w \text{ has length exactly three} \}$

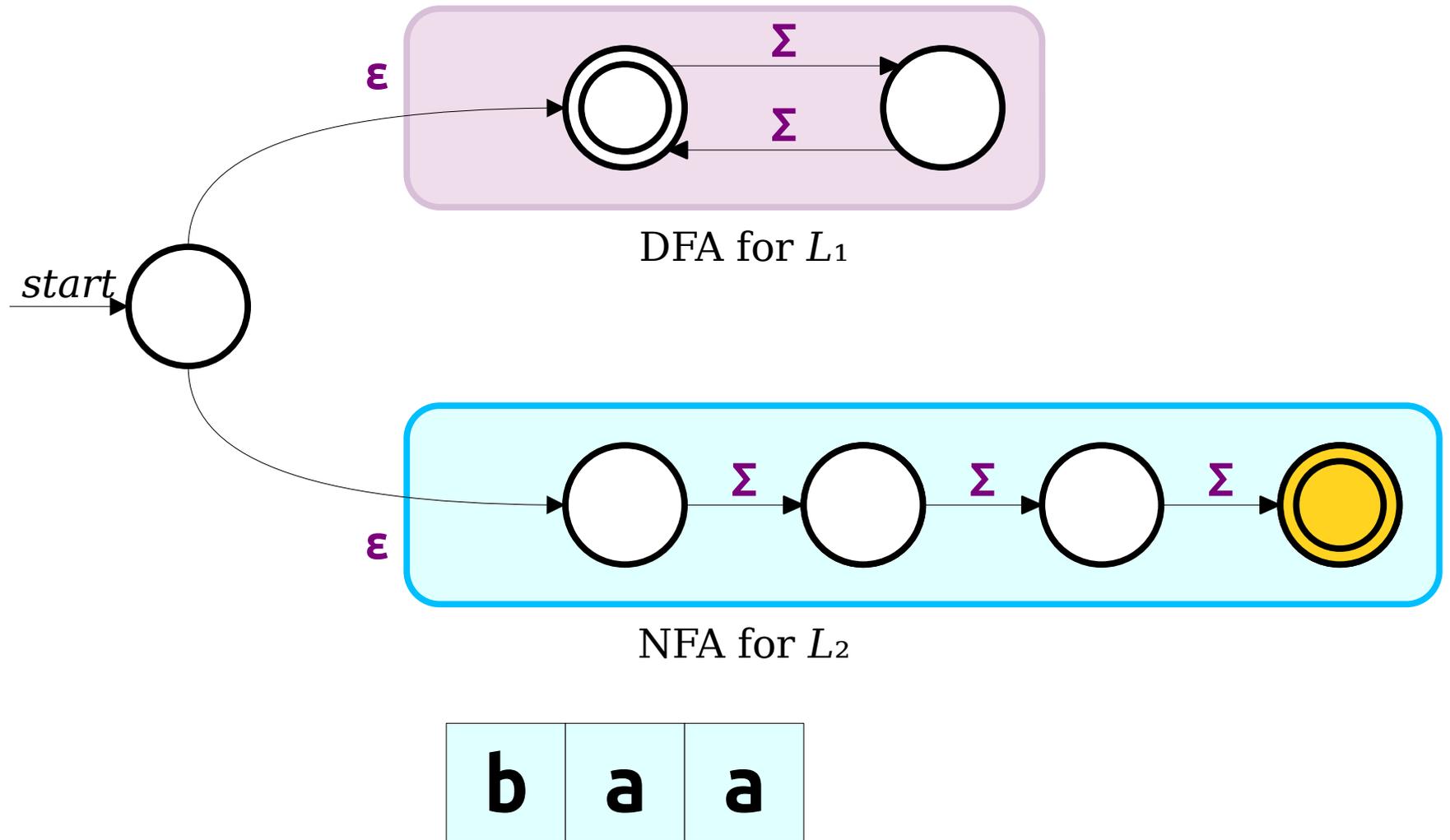
Construct an NFA for  $L_1 \cup L_2$ .



$$L_1 = \{ w \in \{a, b\}^* \mid w \text{ has even length} \}$$

$$L_2 = \{ w \in \{a, b\}^* \mid w \text{ has length exactly three} \}$$

Construct an NFA for  $L_1 \cup L_2$ .

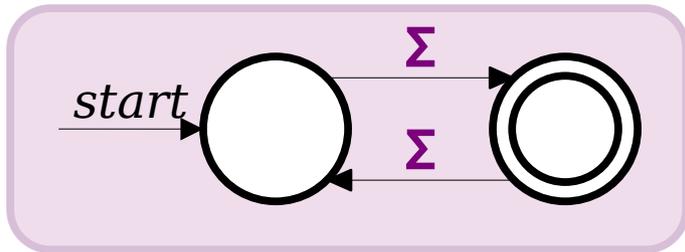


$L_1 = \{ w \in \{a, b\}^* \mid w \text{ has even length} \}$   
 $L_2 = \{ w \in \{a, b\}^* \mid w \text{ has length exactly three} \}$   
 Construct an NFA for  $L_1 \cup L_2$ .

---

$L_1 = \{ w \in \{\mathbf{a}, \mathbf{b}\}^* \mid w \text{ has } \textit{odd} \text{ length} \}$   
 $L_2 = \{ w \in \{\mathbf{a}, \mathbf{b}\}^* \mid w \text{ has length exactly three} \}$

Construct an NFA for  $L_1L_2$ .

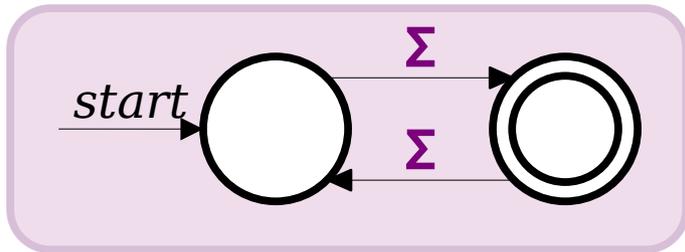


DFA for  $L_1$

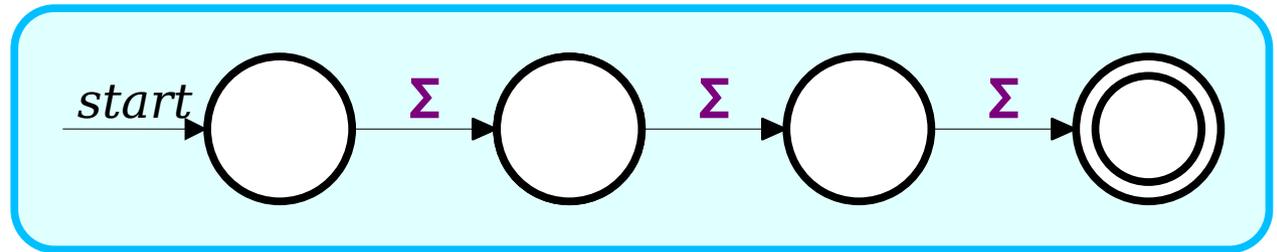
---

$$L_1 = \{ w \in \{a, b\}^* \mid w \text{ has odd length} \}$$
$$L_2 = \{ w \in \{a, b\}^* \mid w \text{ has length exactly three} \}$$

Construct an NFA for  $L_1L_2$ .



DFA for  $L_1$

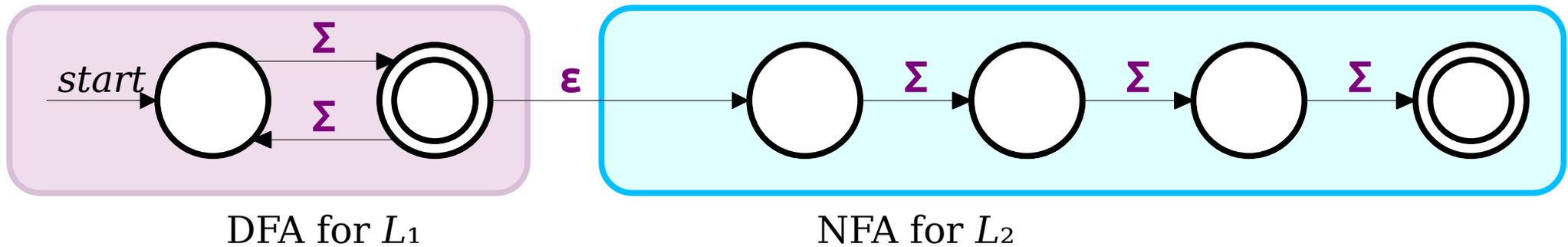


NFA for  $L_2$

$$L_1 = \{ w \in \{ \mathbf{a}, \mathbf{b} \}^* \mid w \text{ has } \textit{odd} \text{ length} \}$$

$$L_2 = \{ w \in \{ \mathbf{a}, \mathbf{b} \}^* \mid w \text{ has length exactly three} \}$$

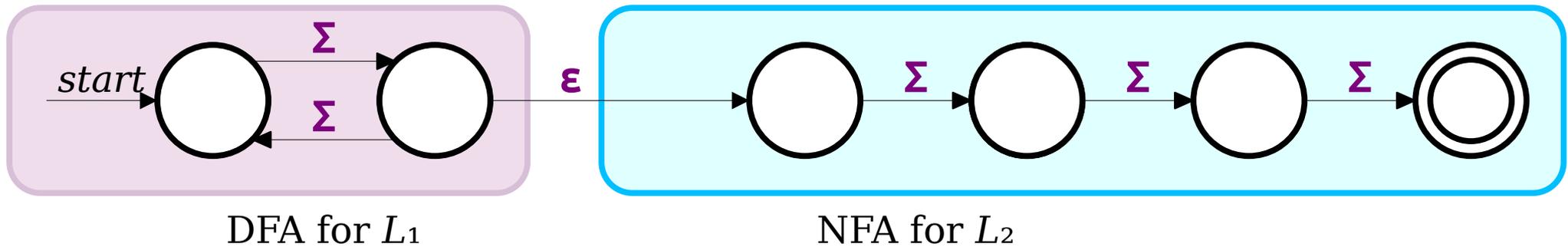
Construct an NFA for  $L_1L_2$ .



$$L_1 = \{ w \in \{a, b\}^* \mid w \text{ has odd length} \}$$

$$L_2 = \{ w \in \{a, b\}^* \mid w \text{ has length exactly three} \}$$

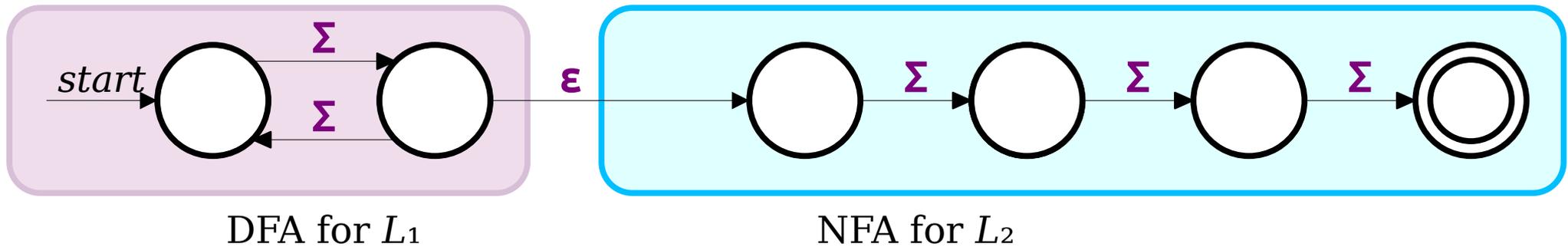
Construct an NFA for  $L_1L_2$ .



$$L_1 = \{ w \in \{a, b\}^* \mid w \text{ has odd length} \}$$

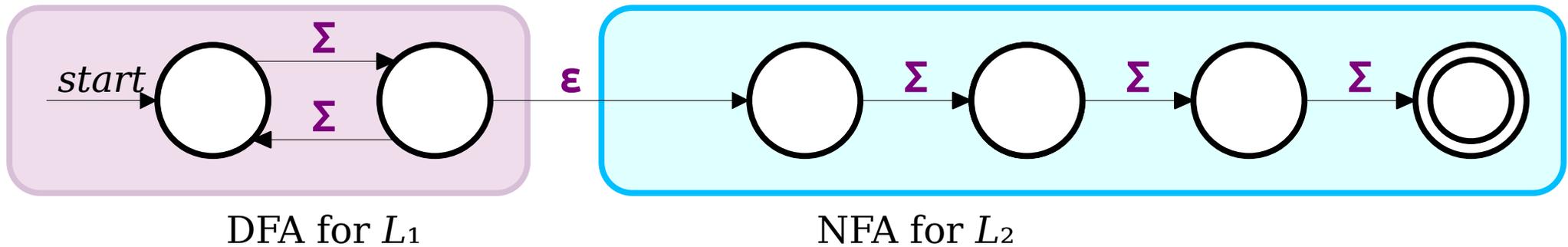
$$L_2 = \{ w \in \{a, b\}^* \mid w \text{ has length exactly three} \}$$

Construct an NFA for  $L_1L_2$ .




---

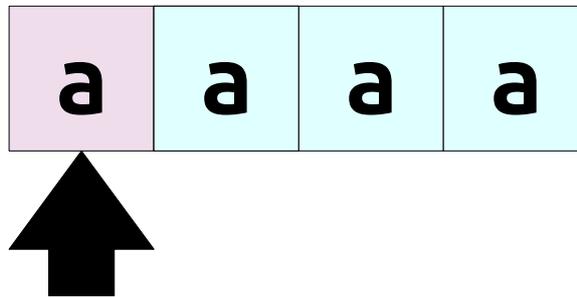
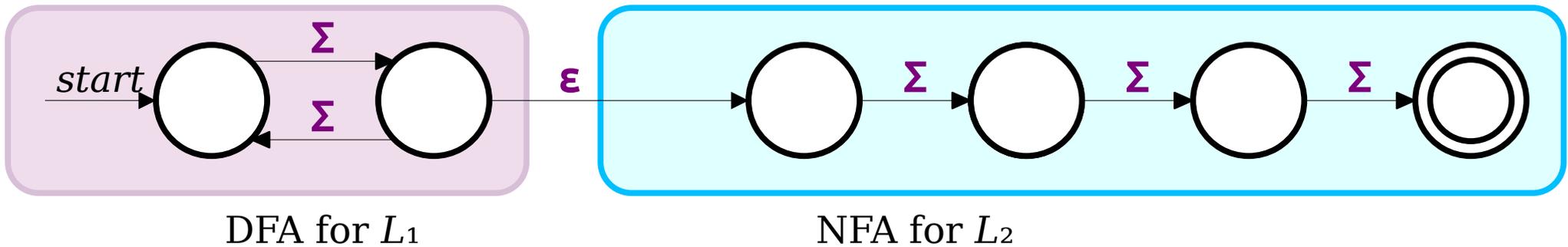
$L_1 = \{ w \in \{\mathbf{a}, \mathbf{b}\}^* \mid w \text{ has } \textit{odd} \text{ length} \}$   
 $L_2 = \{ w \in \{\mathbf{a}, \mathbf{b}\}^* \mid w \text{ has length exactly three} \}$   
 Construct an NFA for  $L_1L_2$ .



$$L_1 = \{ w \in \{a, b\}^* \mid w \text{ has odd length} \}$$

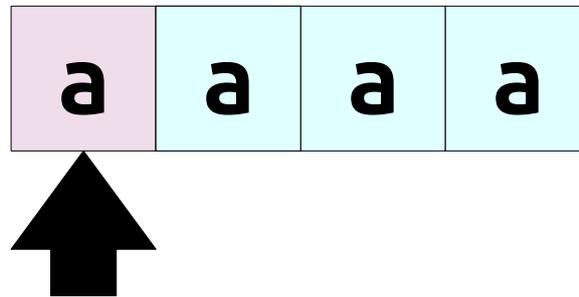
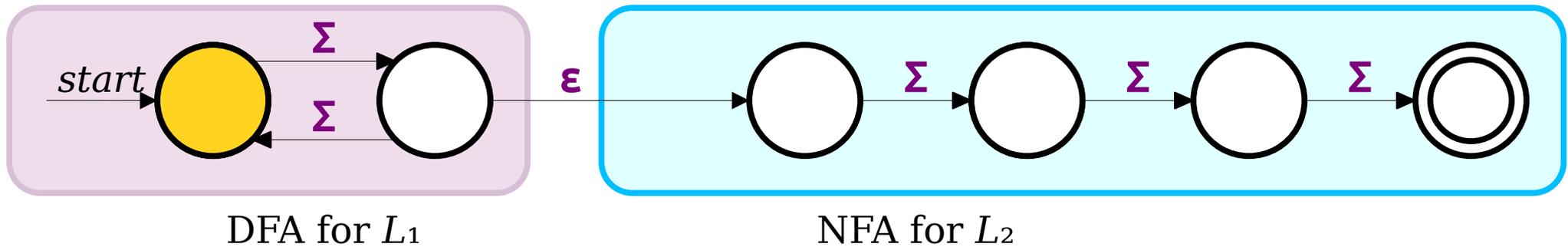
$$L_2 = \{ w \in \{a, b\}^* \mid w \text{ has length exactly three} \}$$

Construct an NFA for  $L_1L_2$ .




---

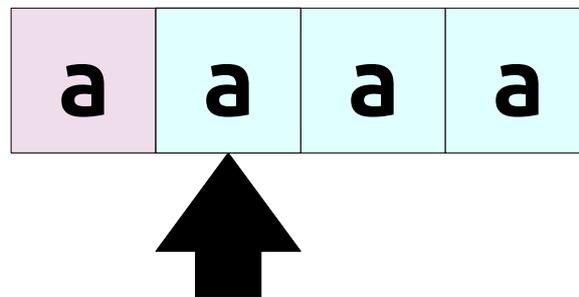
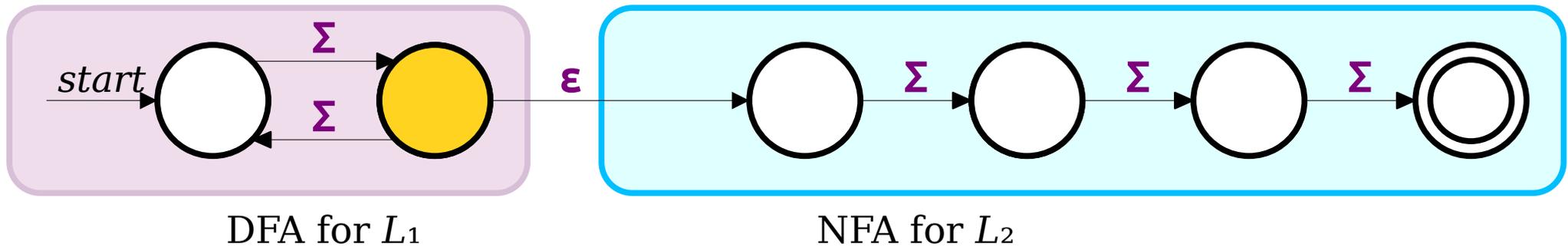
$L_1 = \{ w \in \{a, b\}^* \mid w \text{ has odd length} \}$   
 $L_2 = \{ w \in \{a, b\}^* \mid w \text{ has length exactly three} \}$   
 Construct an NFA for  $L_1L_2$ .



$$L_1 = \{ w \in \{a, b\}^* \mid w \text{ has odd length} \}$$

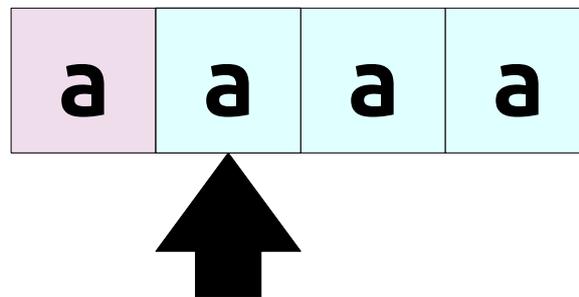
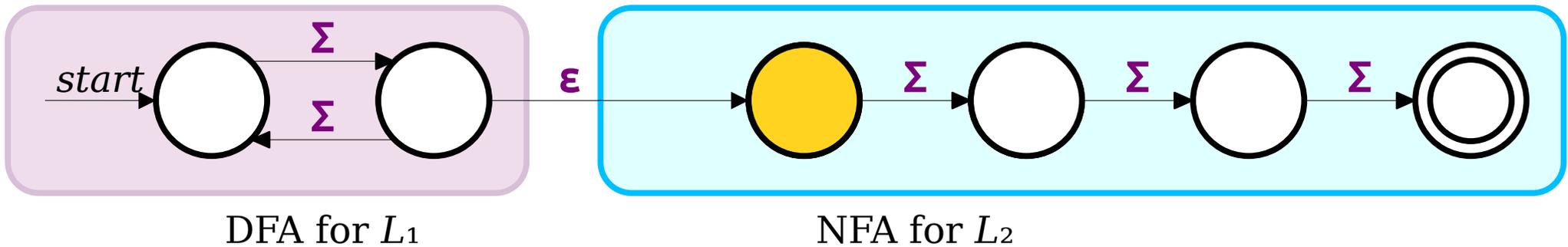
$$L_2 = \{ w \in \{a, b\}^* \mid w \text{ has length exactly three} \}$$

Construct an NFA for  $L_1L_2$ .



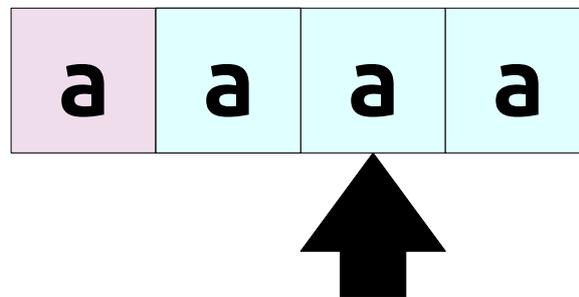
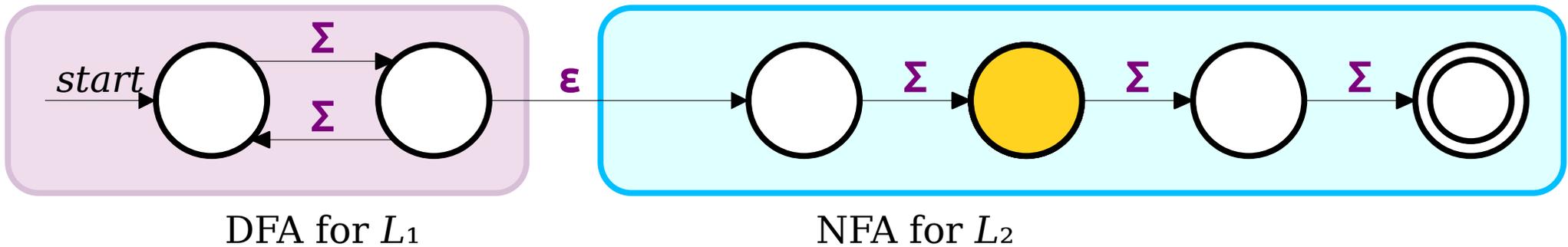
$L_1 = \{ w \in \{a, b\}^* \mid w \text{ has odd length} \}$   
 $L_2 = \{ w \in \{a, b\}^* \mid w \text{ has length exactly three} \}$

Construct an NFA for  $L_1L_2$ .



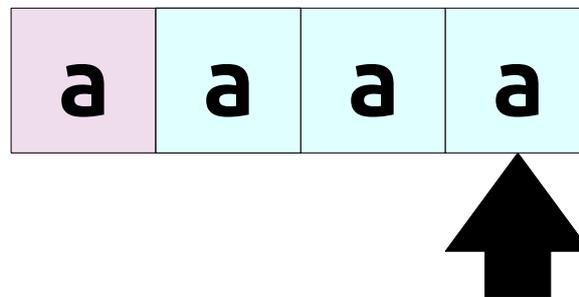
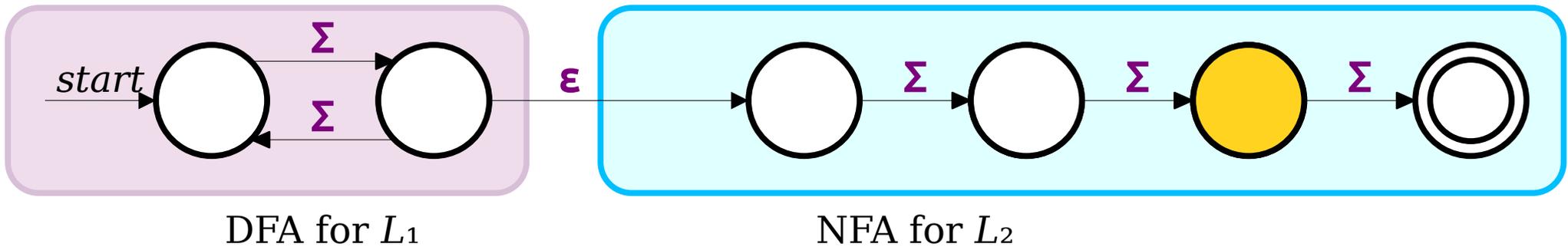

---

$L_1 = \{ w \in \{a, b\}^* \mid w \text{ has odd length} \}$   
 $L_2 = \{ w \in \{a, b\}^* \mid w \text{ has length exactly three} \}$   
 Construct an NFA for  $L_1L_2$ .



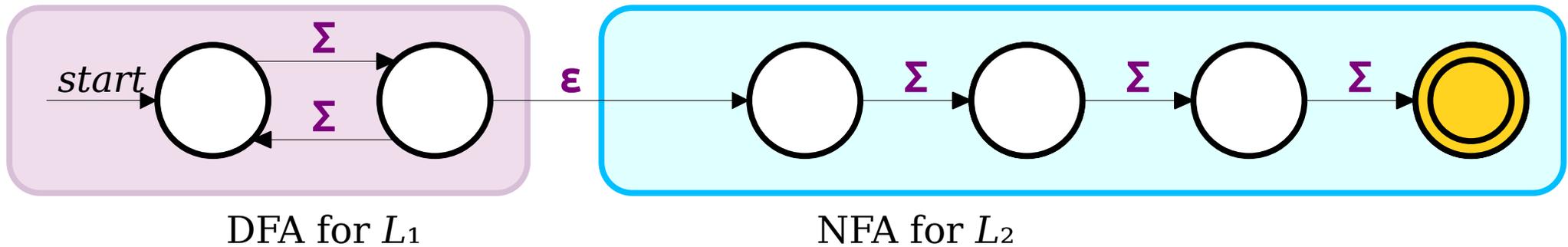

---

$L_1 = \{ w \in \{a, b\}^* \mid w \text{ has odd length} \}$   
 $L_2 = \{ w \in \{a, b\}^* \mid w \text{ has length exactly three} \}$   
 Construct an NFA for  $L_1L_2$ .



$L_1 = \{ w \in \{a, b\}^* \mid w \text{ has odd length} \}$   
 $L_2 = \{ w \in \{a, b\}^* \mid w \text{ has length exactly three} \}$

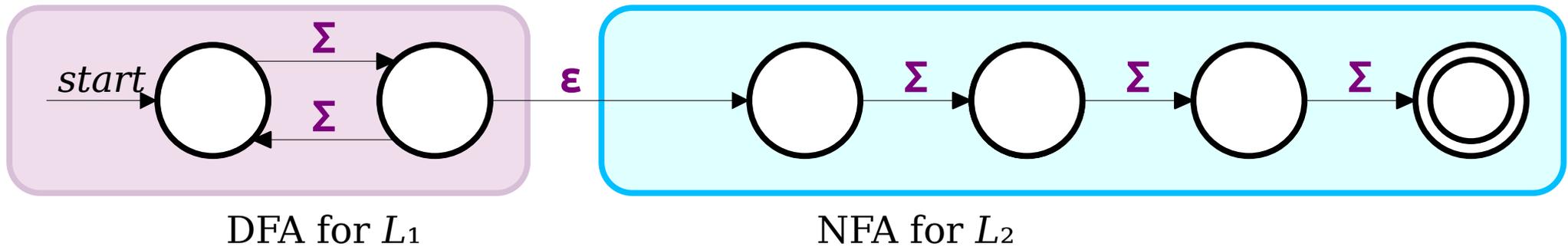
Construct an NFA for  $L_1L_2$ .



$$L_1 = \{ w \in \{a, b\}^* \mid w \text{ has odd length} \}$$

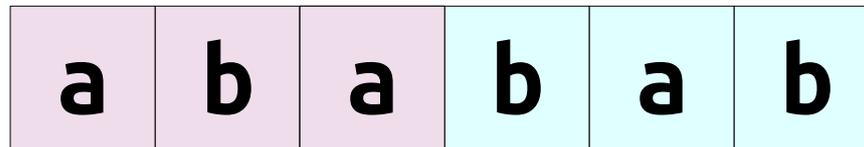
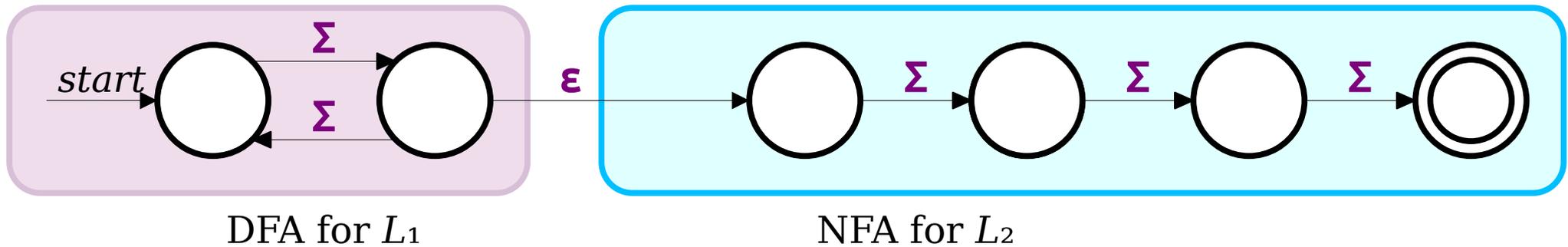
$$L_2 = \{ w \in \{a, b\}^* \mid w \text{ has length exactly three} \}$$

Construct an NFA for  $L_1L_2$ .



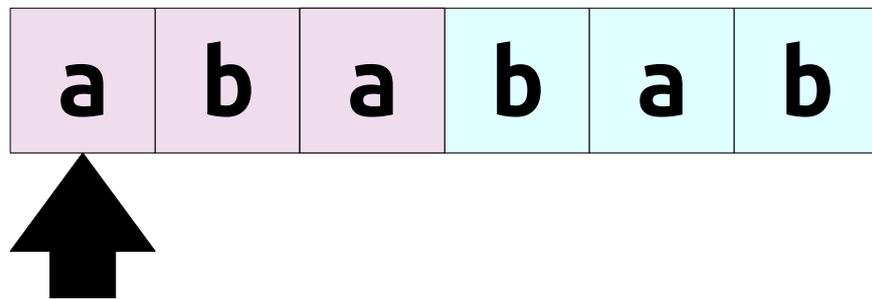
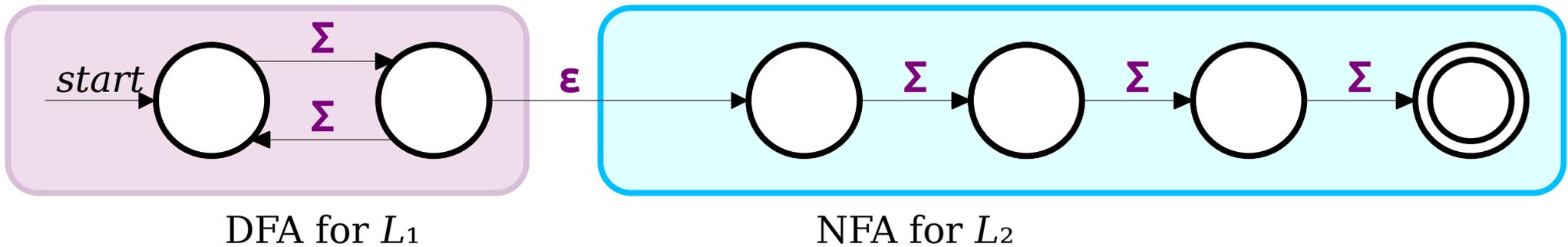
$L_1 = \{ w \in \{a, b\}^* \mid w \text{ has odd length} \}$   
 $L_2 = \{ w \in \{a, b\}^* \mid w \text{ has length exactly three} \}$

Construct an NFA for  $L_1L_2$ .



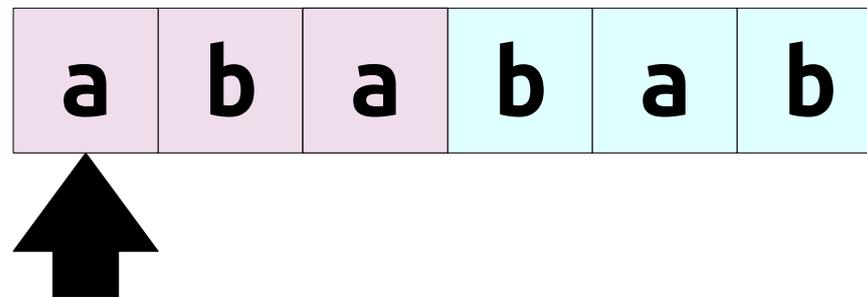
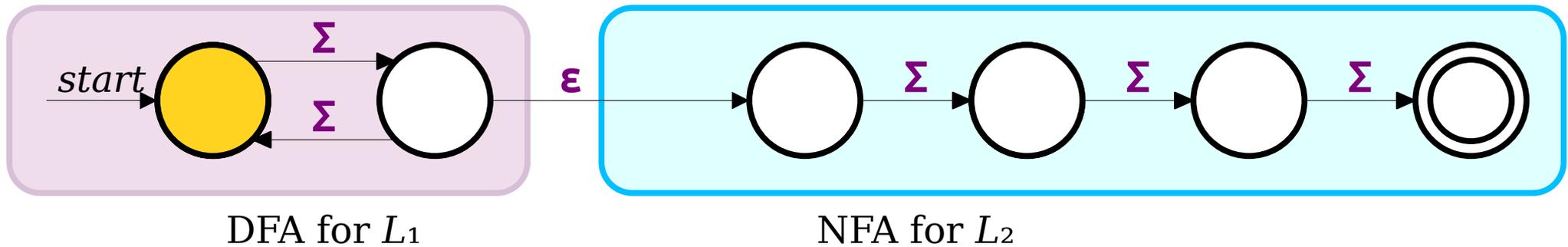
$L_1 = \{ w \in \{a, b\}^* \mid w \text{ has odd length} \}$   
 $L_2 = \{ w \in \{a, b\}^* \mid w \text{ has length exactly three} \}$

Construct an NFA for  $L_1L_2$ .



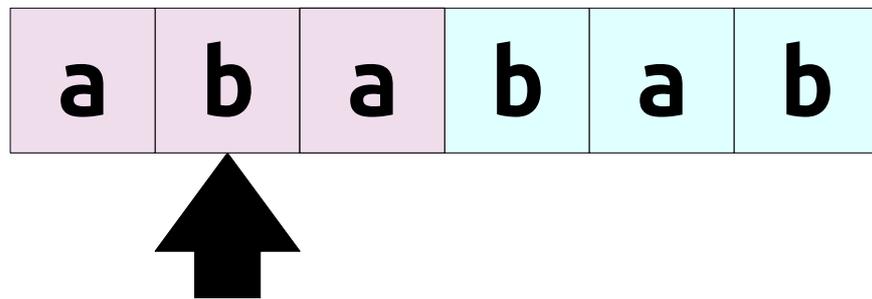
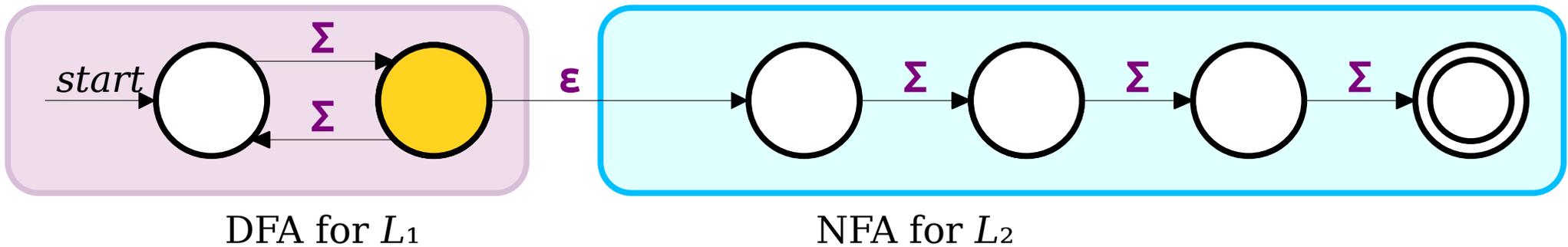

---

$L_1 = \{ w \in \{a, b\}^* \mid w \text{ has odd length} \}$   
 $L_2 = \{ w \in \{a, b\}^* \mid w \text{ has length exactly three} \}$   
 Construct an NFA for  $L_1L_2$ .



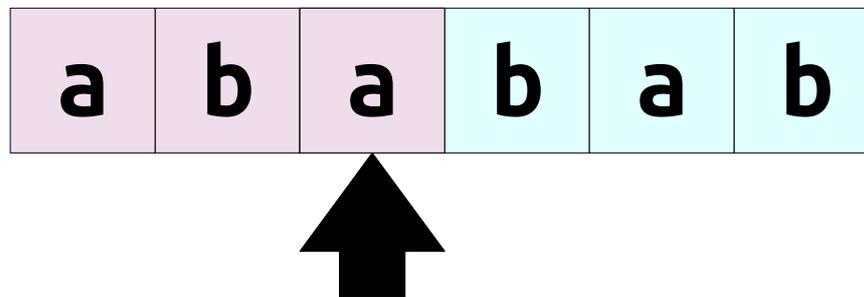
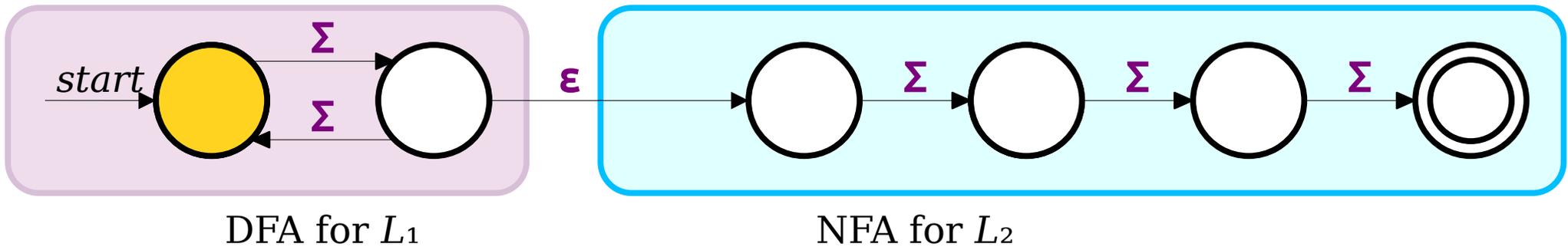
$L_1 = \{ w \in \{a, b\}^* \mid w \text{ has odd length} \}$   
 $L_2 = \{ w \in \{a, b\}^* \mid w \text{ has length exactly three} \}$

Construct an NFA for  $L_1L_2$ .



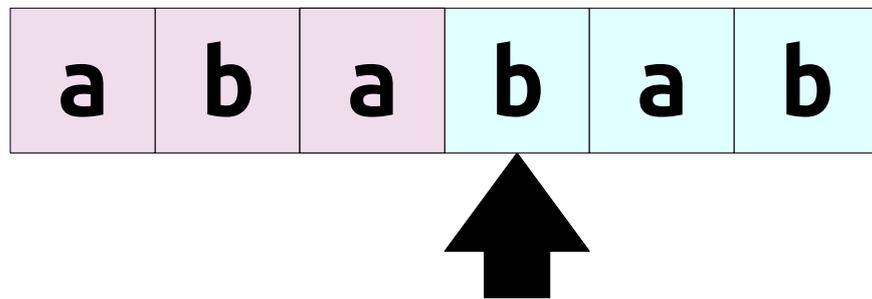
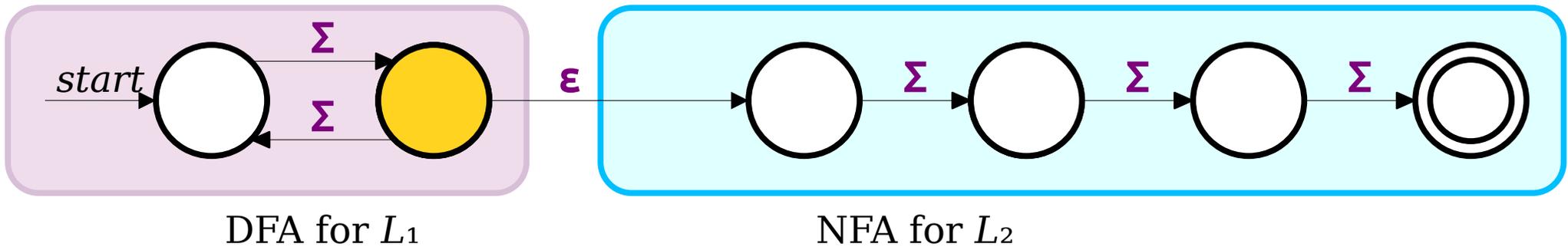
$L_1 = \{ w \in \{a, b\}^* \mid w \text{ has odd length} \}$   
 $L_2 = \{ w \in \{a, b\}^* \mid w \text{ has length exactly three} \}$

Construct an NFA for  $L_1L_2$ .



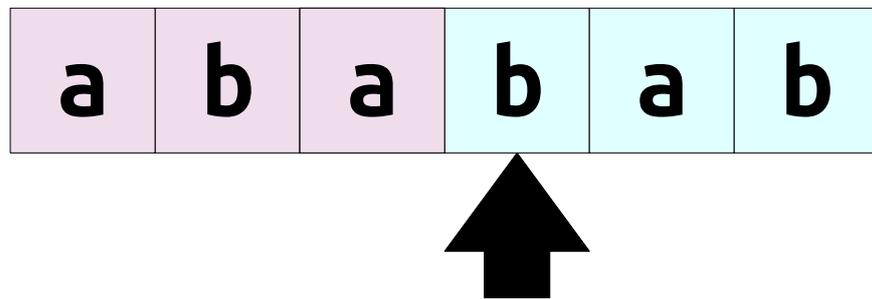
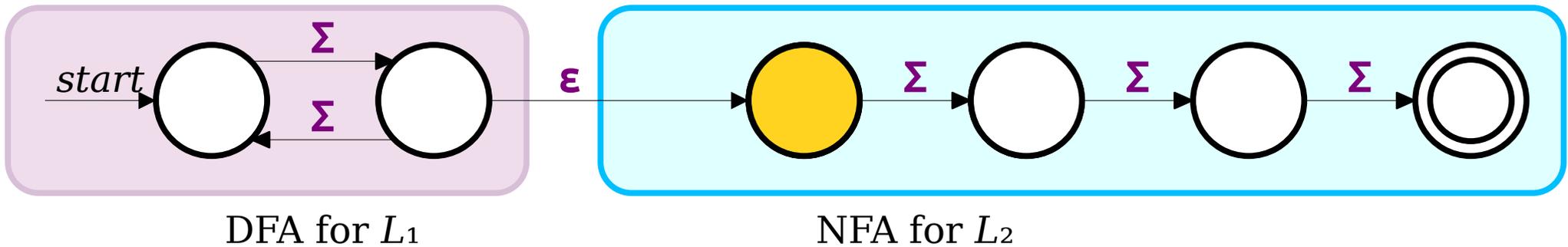
$L_1 = \{ w \in \{a, b\}^* \mid w \text{ has odd length} \}$   
 $L_2 = \{ w \in \{a, b\}^* \mid w \text{ has length exactly three} \}$

Construct an NFA for  $L_1L_2$ .



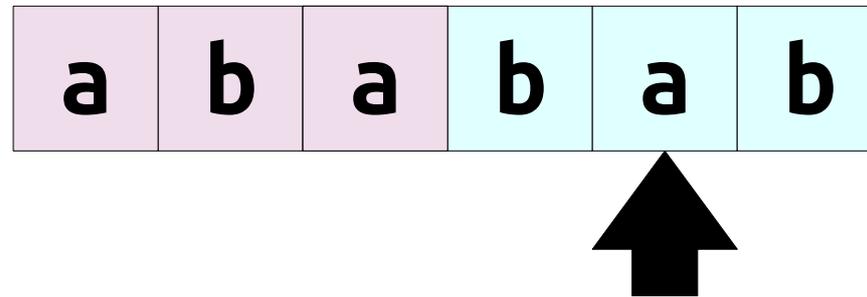
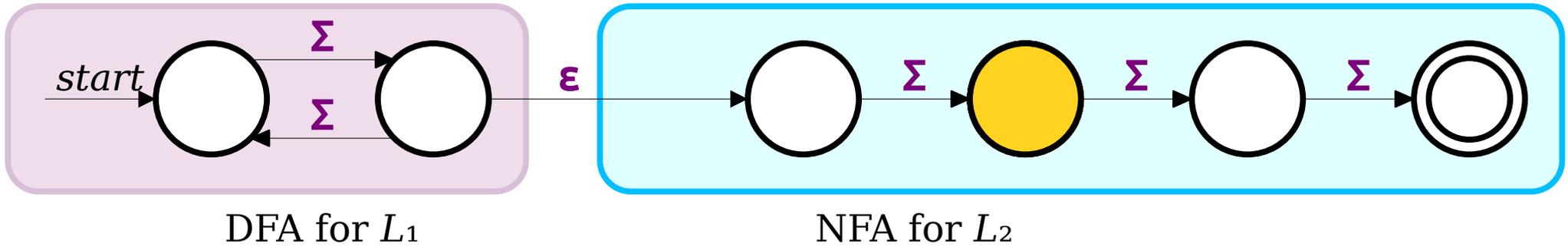
$L_1 = \{ w \in \{a, b\}^* \mid w \text{ has odd length} \}$   
 $L_2 = \{ w \in \{a, b\}^* \mid w \text{ has length exactly three} \}$

Construct an NFA for  $L_1L_2$ .



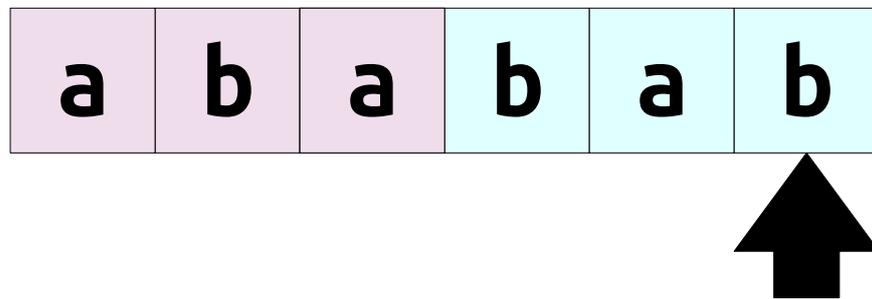
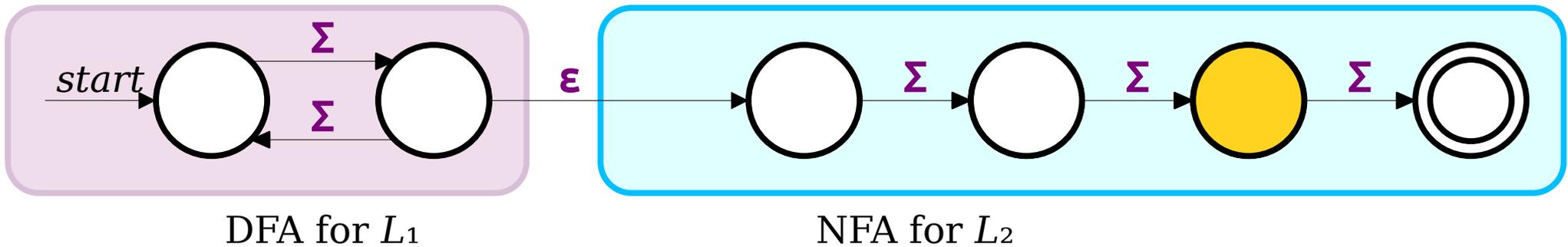
$L_1 = \{ w \in \{a, b\}^* \mid w \text{ has odd length} \}$   
 $L_2 = \{ w \in \{a, b\}^* \mid w \text{ has length exactly three} \}$

Construct an NFA for  $L_1L_2$ .



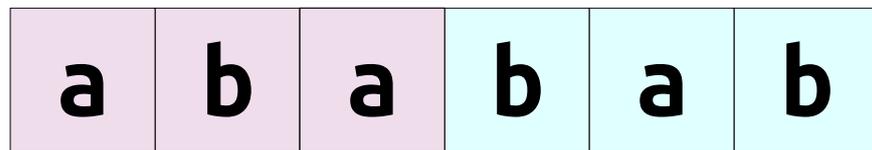
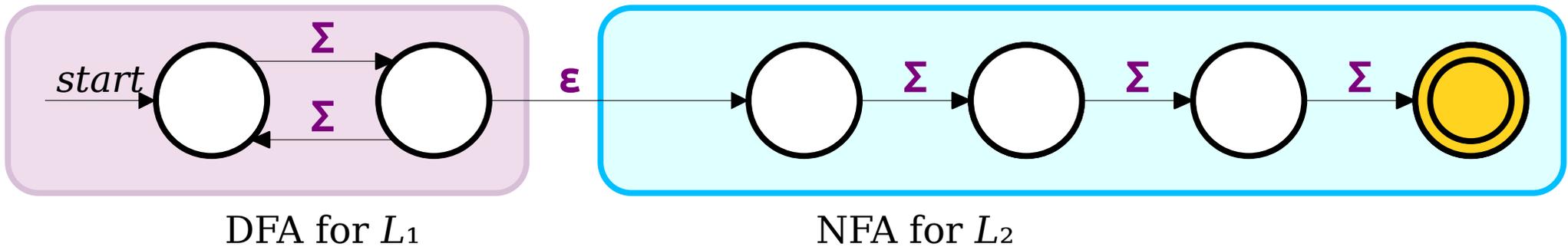
$L_1 = \{ w \in \{a, b\}^* \mid w \text{ has odd length} \}$   
 $L_2 = \{ w \in \{a, b\}^* \mid w \text{ has length exactly three} \}$

Construct an NFA for  $L_1L_2$ .



$L_1 = \{ w \in \{a, b\}^* \mid w \text{ has odd length} \}$   
 $L_2 = \{ w \in \{a, b\}^* \mid w \text{ has length exactly three} \}$

Construct an NFA for  $L_1L_2$ .



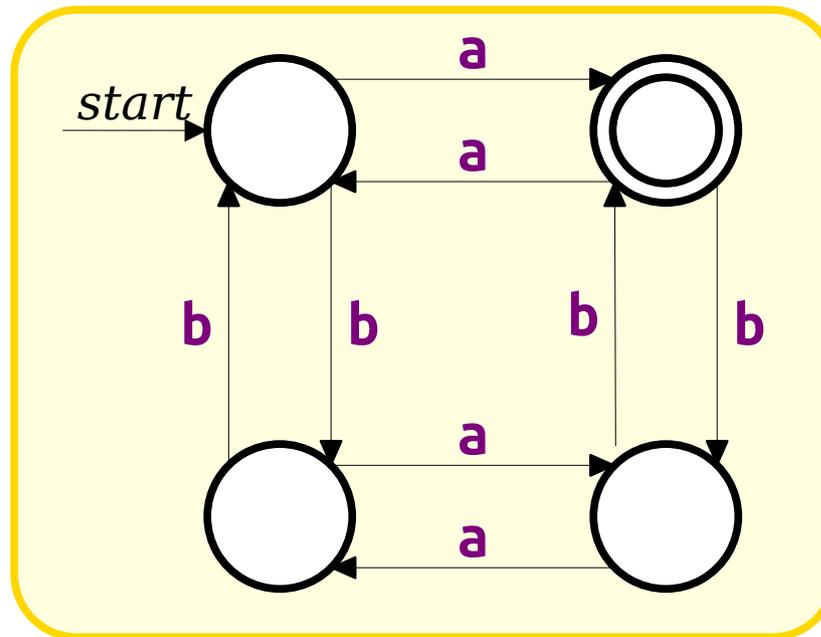
$L_1 = \{ w \in \{a, b\}^* \mid w \text{ has odd length} \}$   
 $L_2 = \{ w \in \{a, b\}^* \mid w \text{ has length exactly three} \}$

Construct an NFA for  $L_1L_2$ .

---

$L = \{ w \in \{\mathbf{a}, \mathbf{b}\}^* \mid w \text{ has an odd number of } \mathbf{a}'\text{s and an even number of } \mathbf{b}'\text{s} \}$

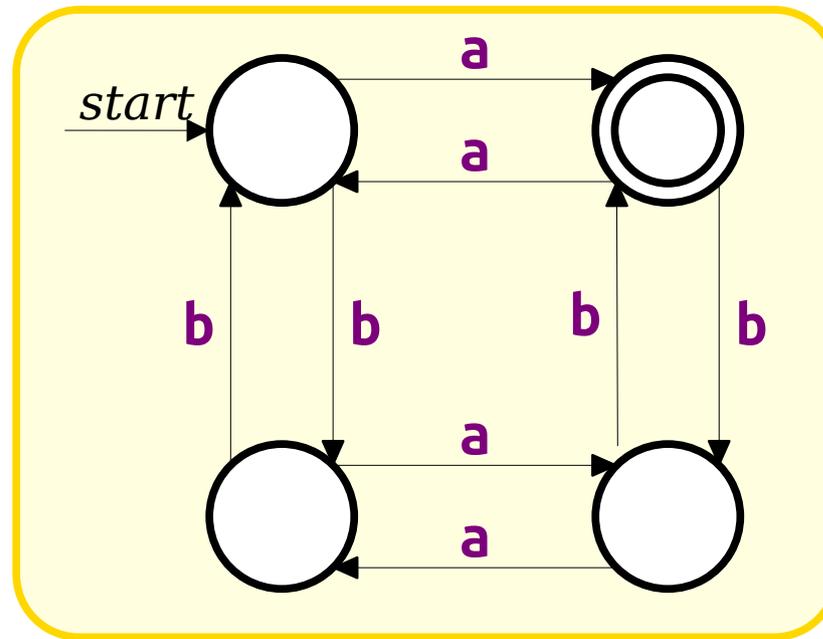
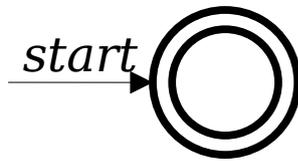
Construct an NFA for  $L^*$ .



DFA for  $L$

$L = \{ w \in \{a, b\}^* \mid w \text{ has an odd number of } a\text{'s and an even number of } b\text{'s} \}$

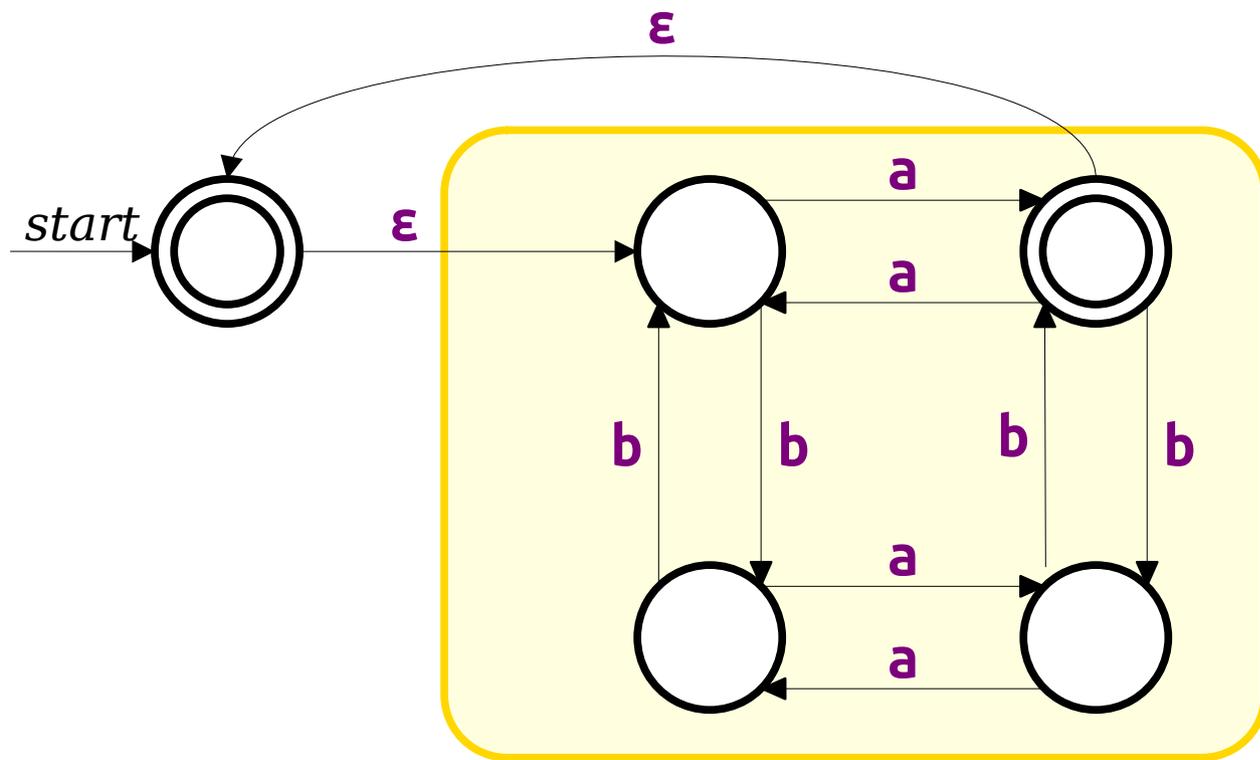
Construct an NFA for  $L^*$ .



DFA for  $L$

$L = \{ w \in \{a, b\}^* \mid w \text{ has an odd number of } a\text{'s and an even number of } b\text{'s} \}$

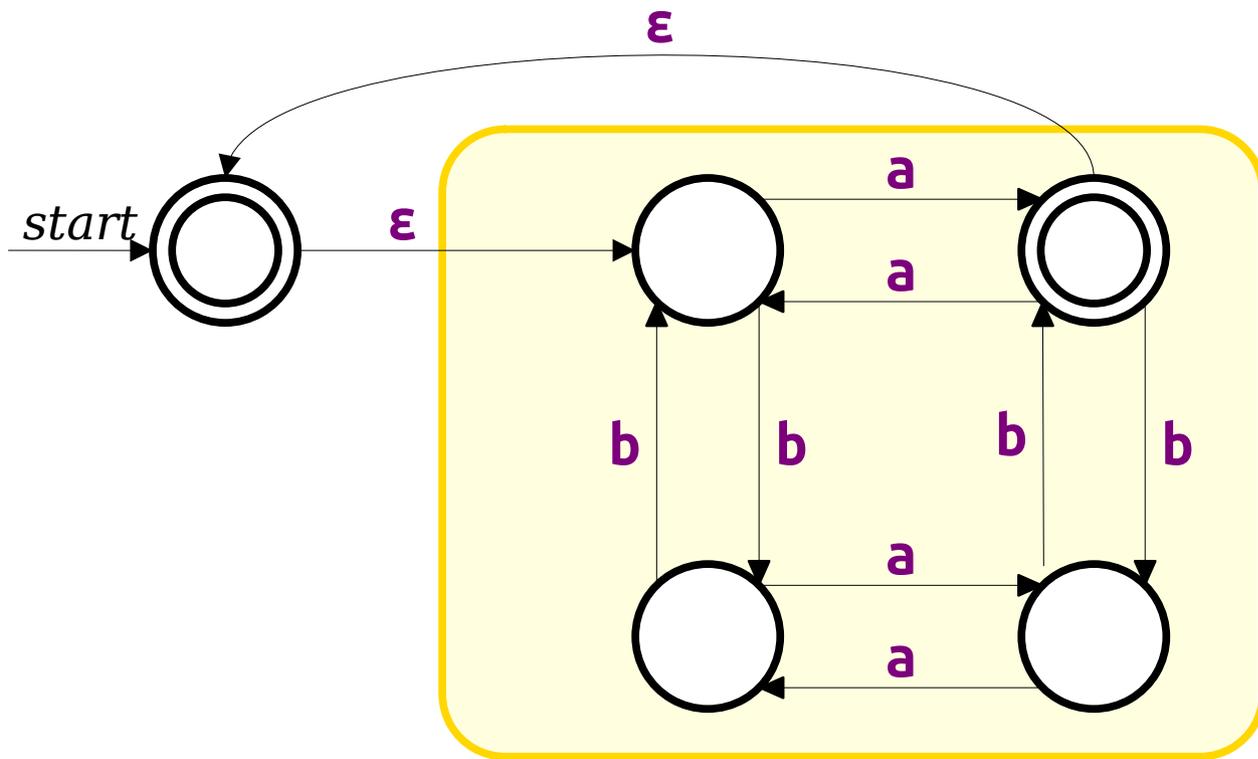
Construct an NFA for  $L^*$ .



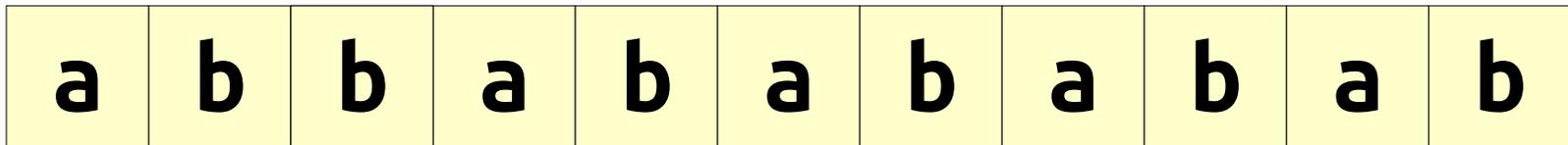
DFA for  $L$

$L = \{ w \in \{a, b\}^* \mid w \text{ has an odd number of } a\text{'s and an even number of } b\text{'s} \}$

Construct an NFA for  $L^*$ .

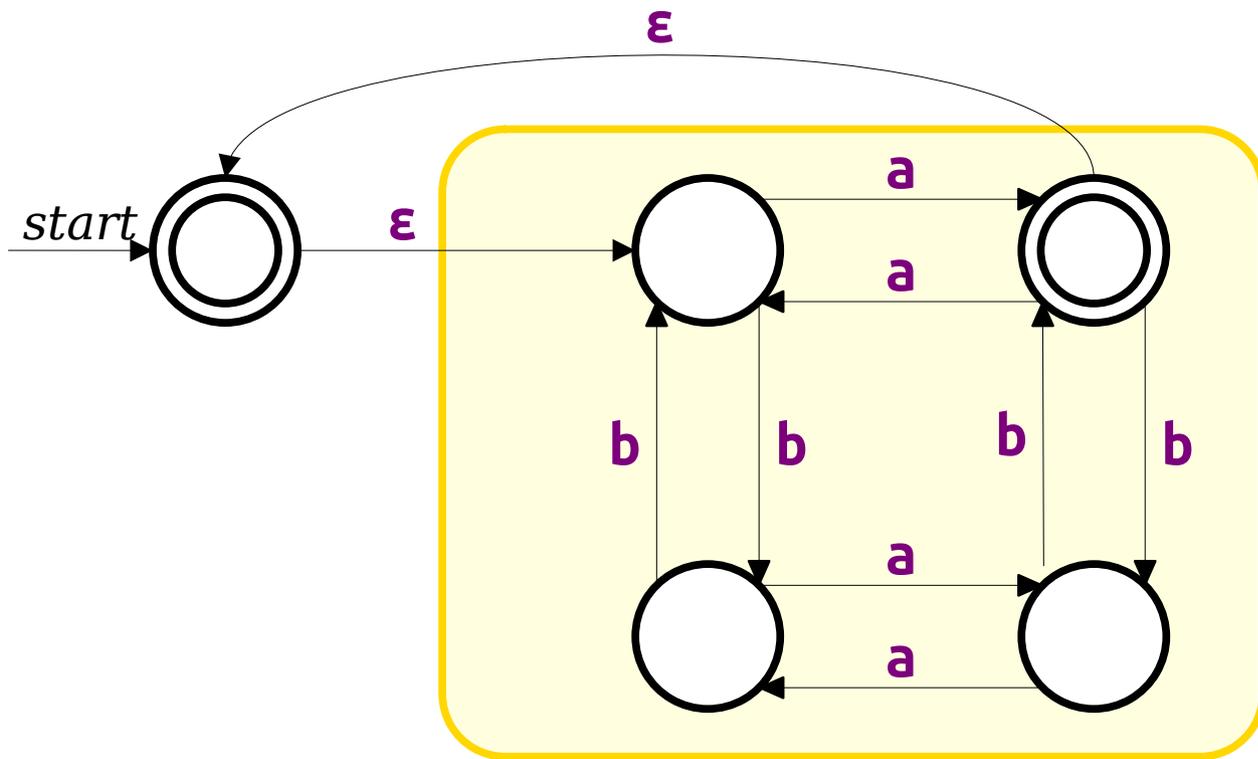


DFA for  $L$

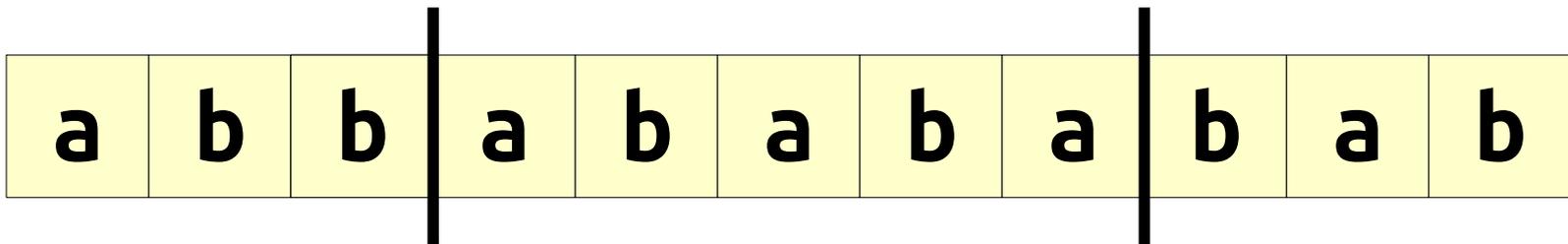


$L = \{ w \in \{a, b\}^* \mid w \text{ has an odd number of } a\text{'s and an even number of } b\text{'s} \}$

Construct an NFA for  $L^*$ .

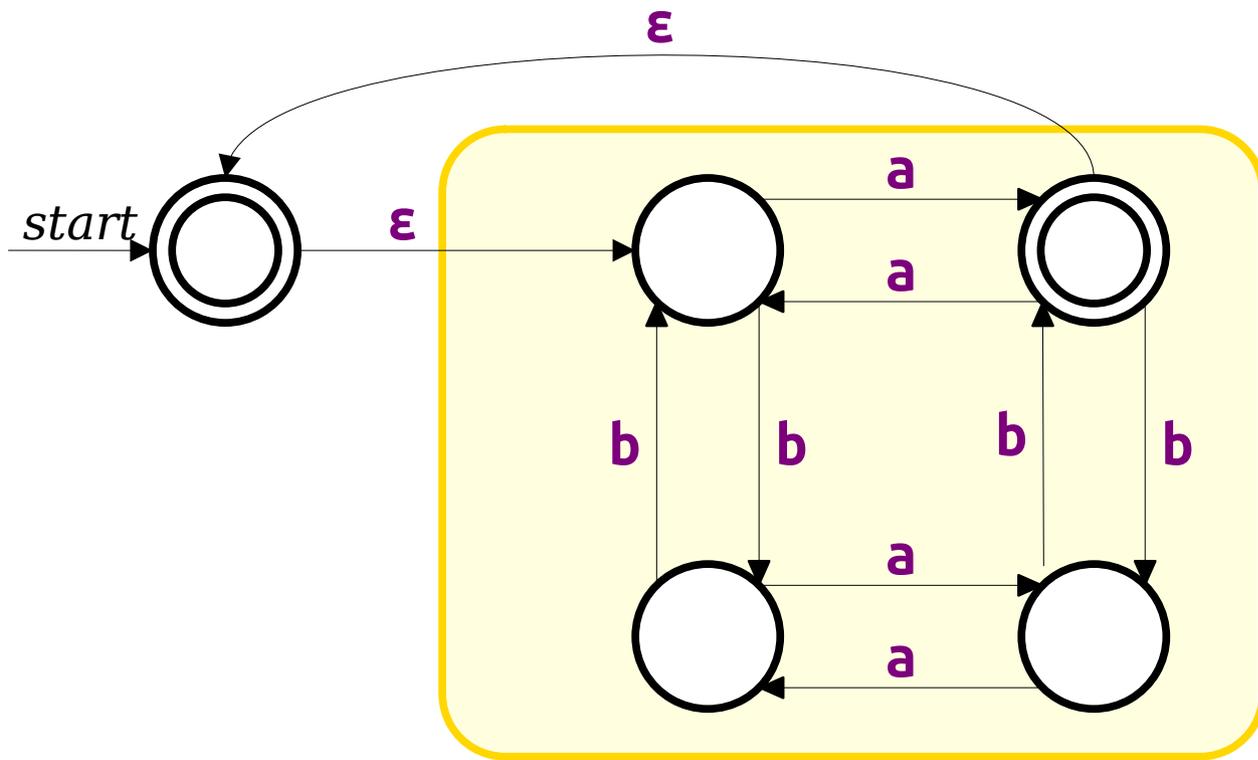


DFA for  $L$

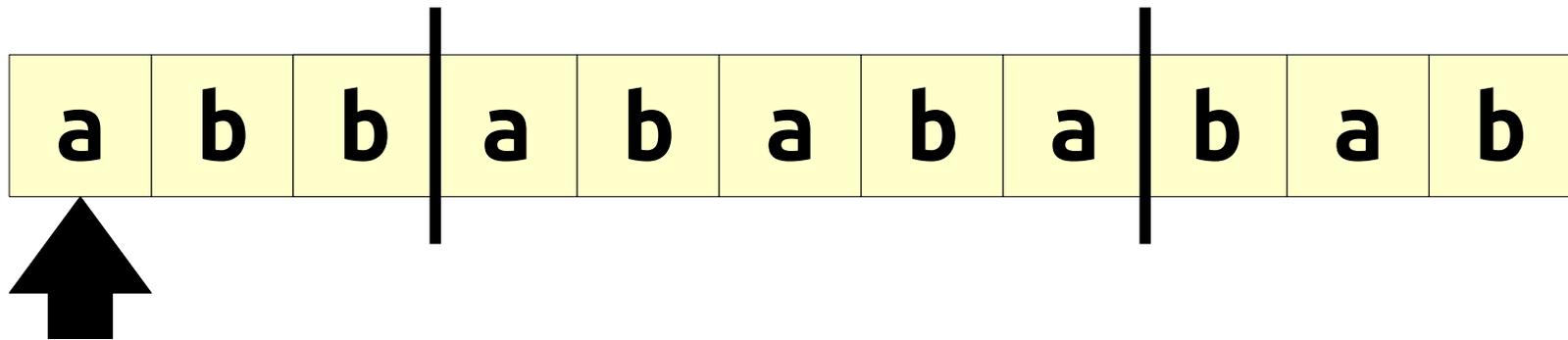


$L = \{ w \in \{a, b\}^* \mid w \text{ has an odd number of } a\text{'s and an even number of } b\text{'s} \}$

Construct an NFA for  $L^*$ .

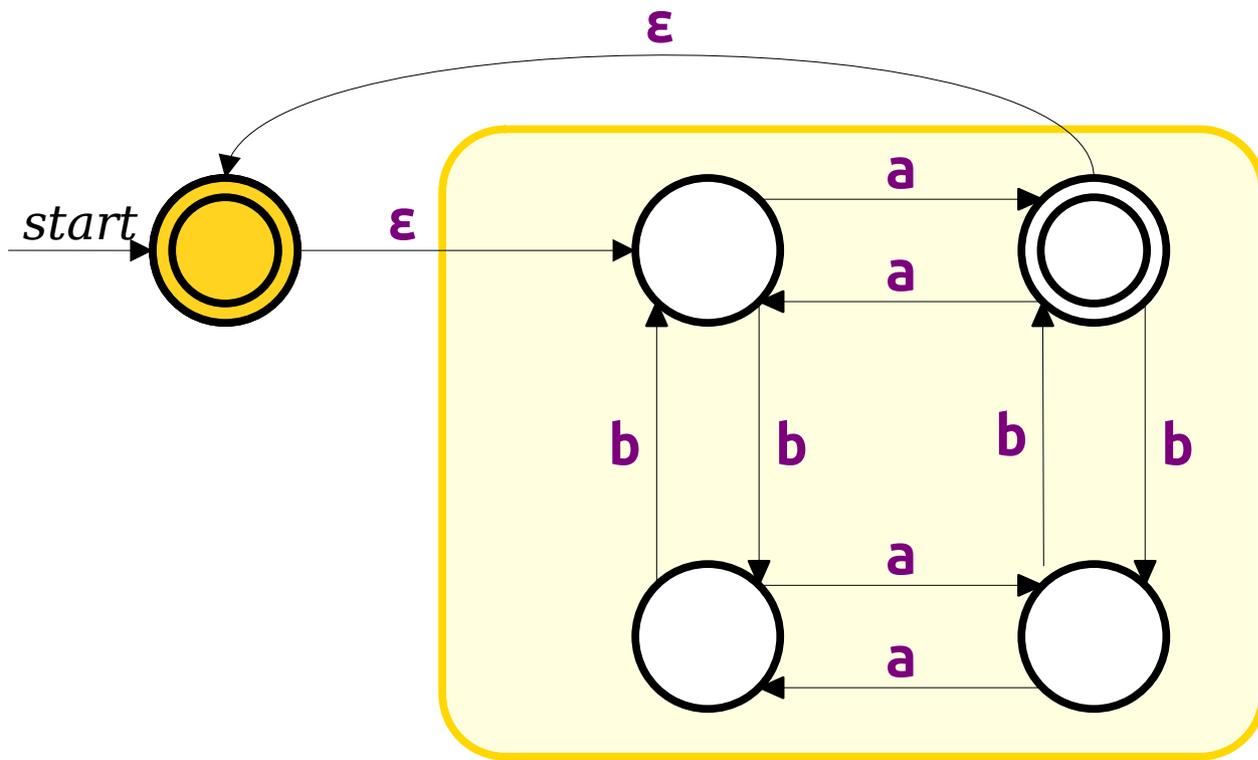


DFA for  $L$

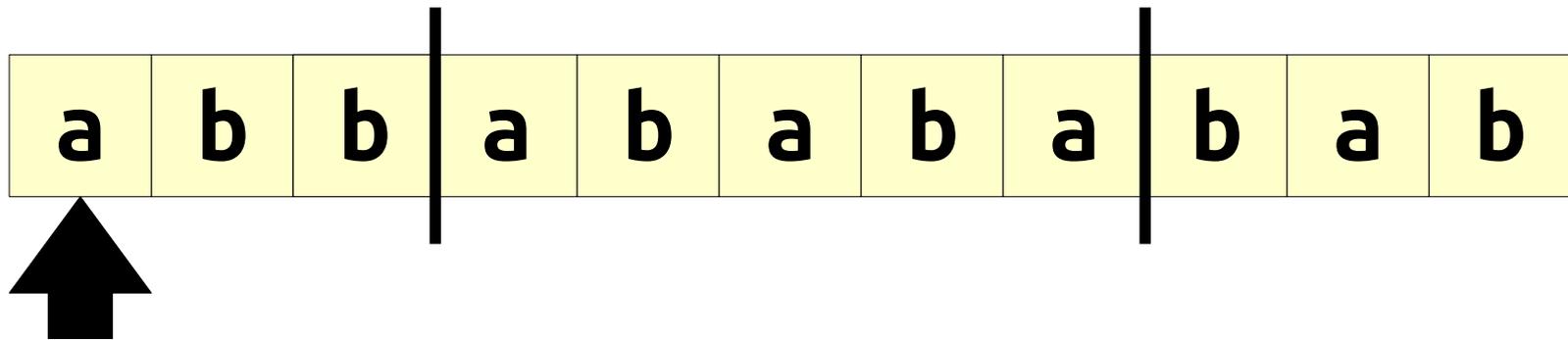


$L = \{ w \in \{a, b\}^* \mid w \text{ has an odd number of } a\text{'s and an even number of } b\text{'s} \}$

Construct an NFA for  $L^*$ .

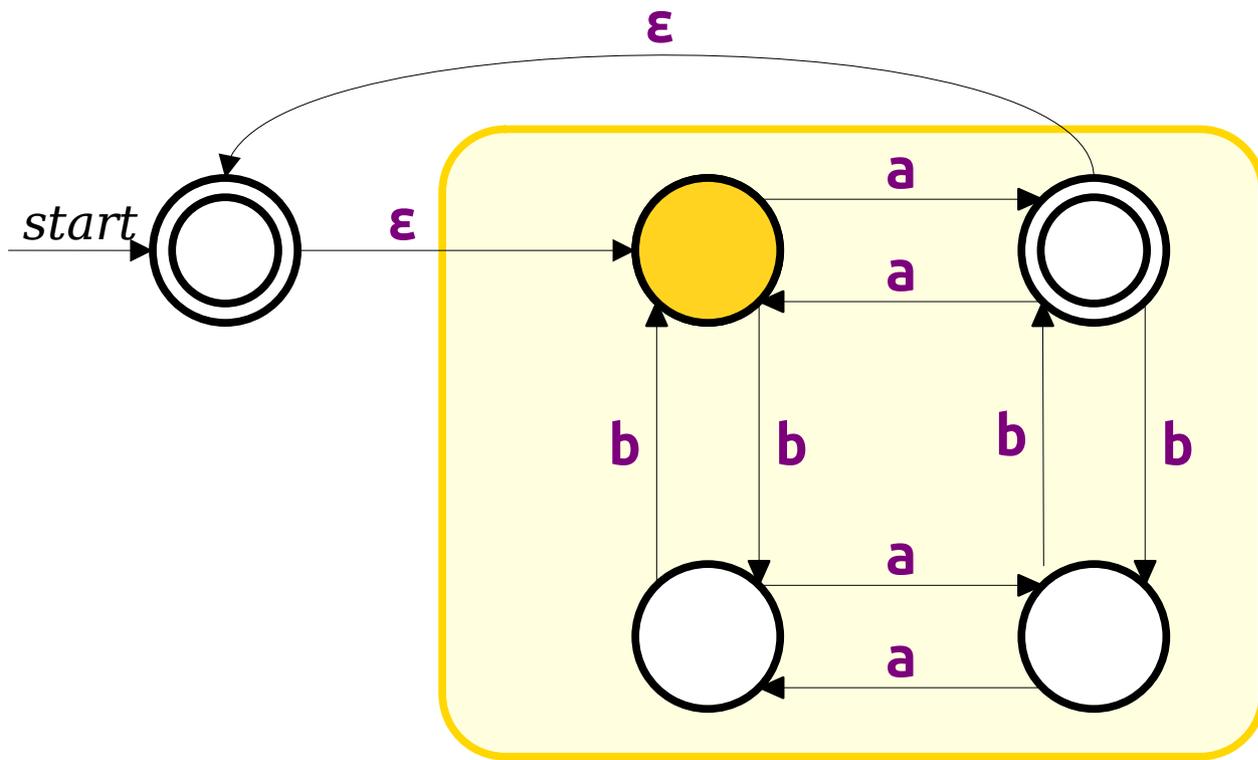


DFA for  $L$

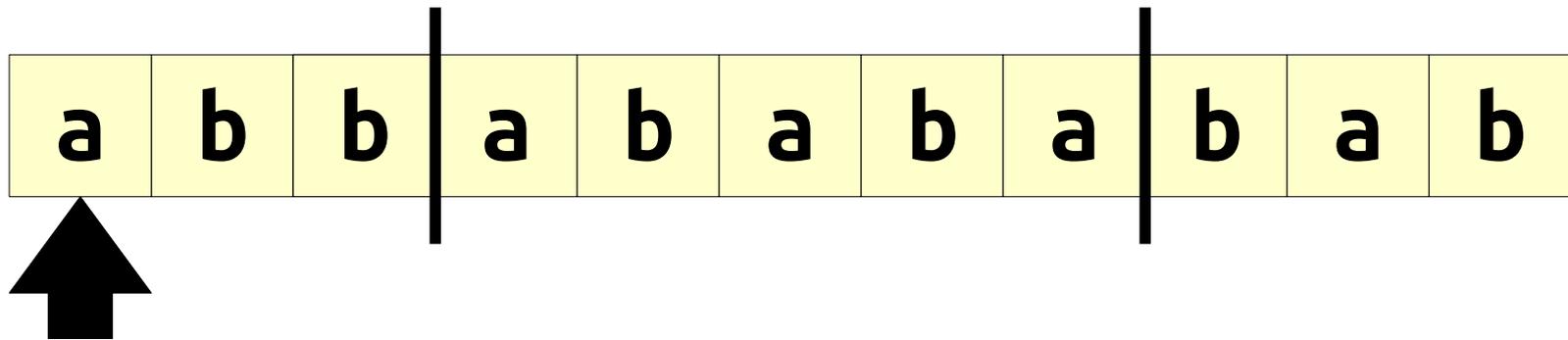


$L = \{ w \in \{a, b\}^* \mid w \text{ has an odd number of } a\text{'s and an even number of } b\text{'s} \}$

Construct an NFA for  $L^*$ .

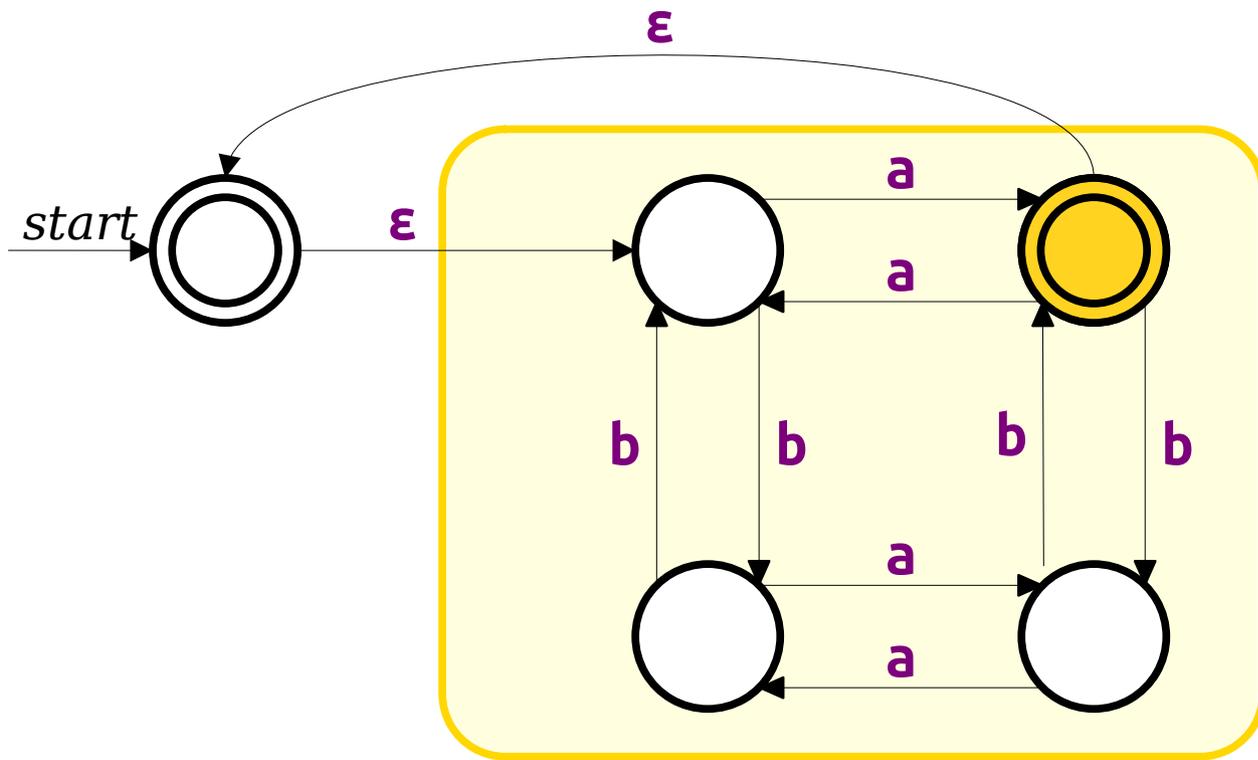


DFA for  $L$

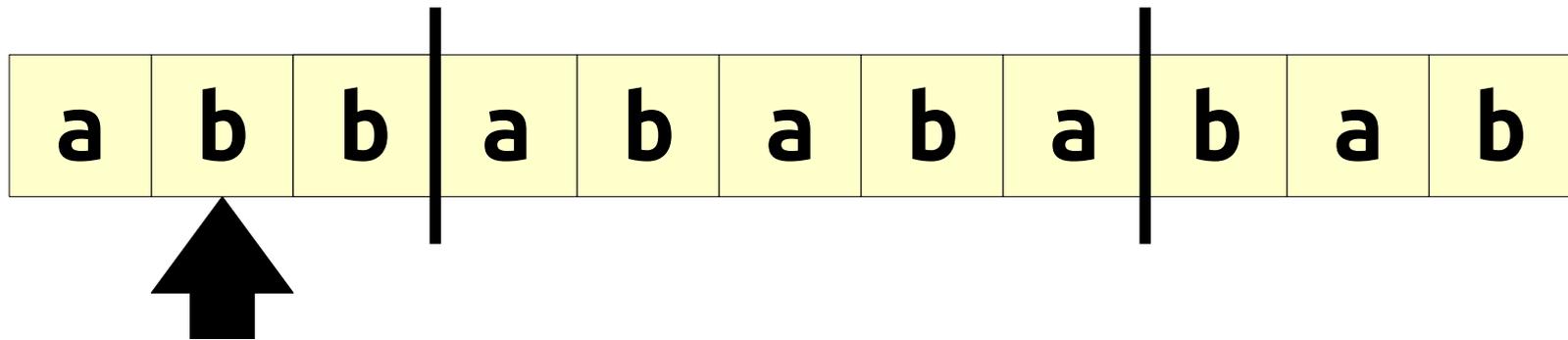


$L = \{ w \in \{a, b\}^* \mid w \text{ has an odd number of } a\text{'s and an even number of } b\text{'s} \}$

Construct an NFA for  $L^*$ .

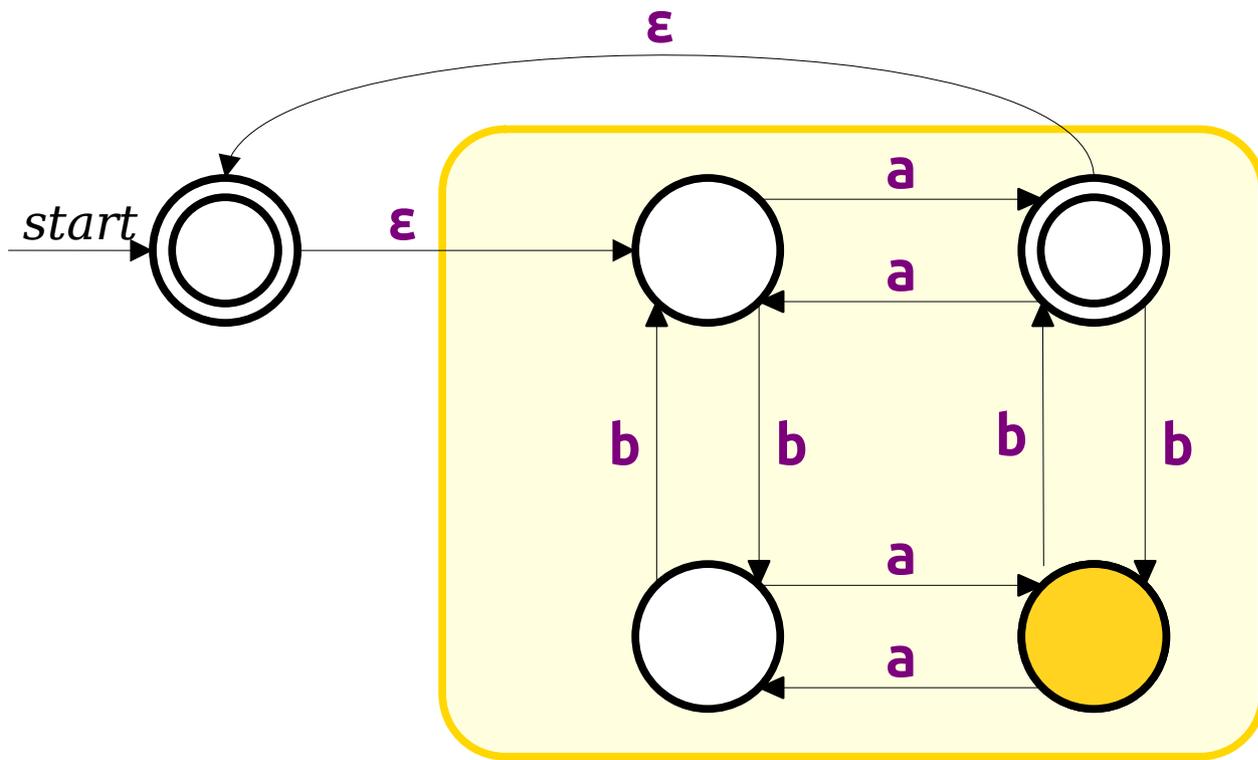


DFA for  $L$

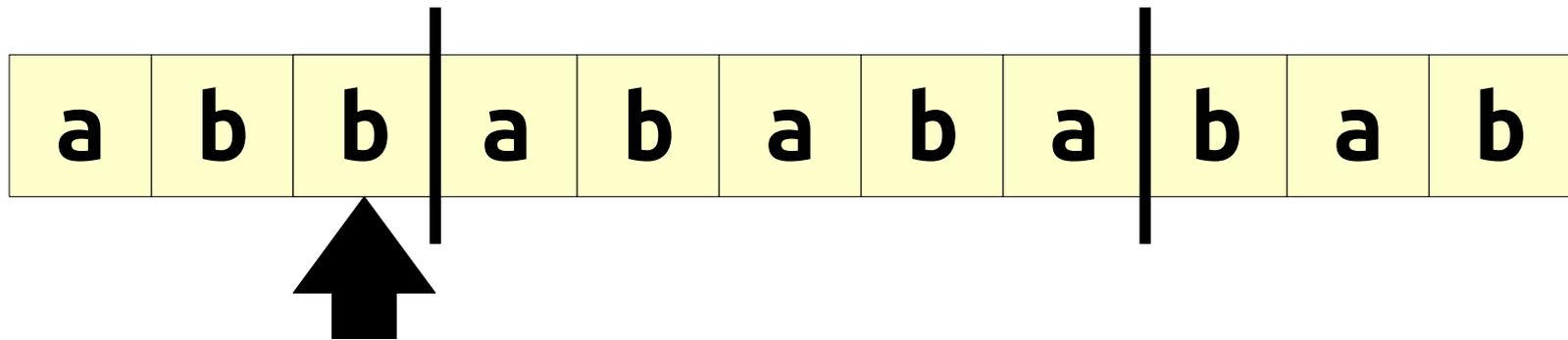


$L = \{ w \in \{a, b\}^* \mid w \text{ has an odd number of } a\text{'s and an even number of } b\text{'s} \}$

Construct an NFA for  $L^*$ .

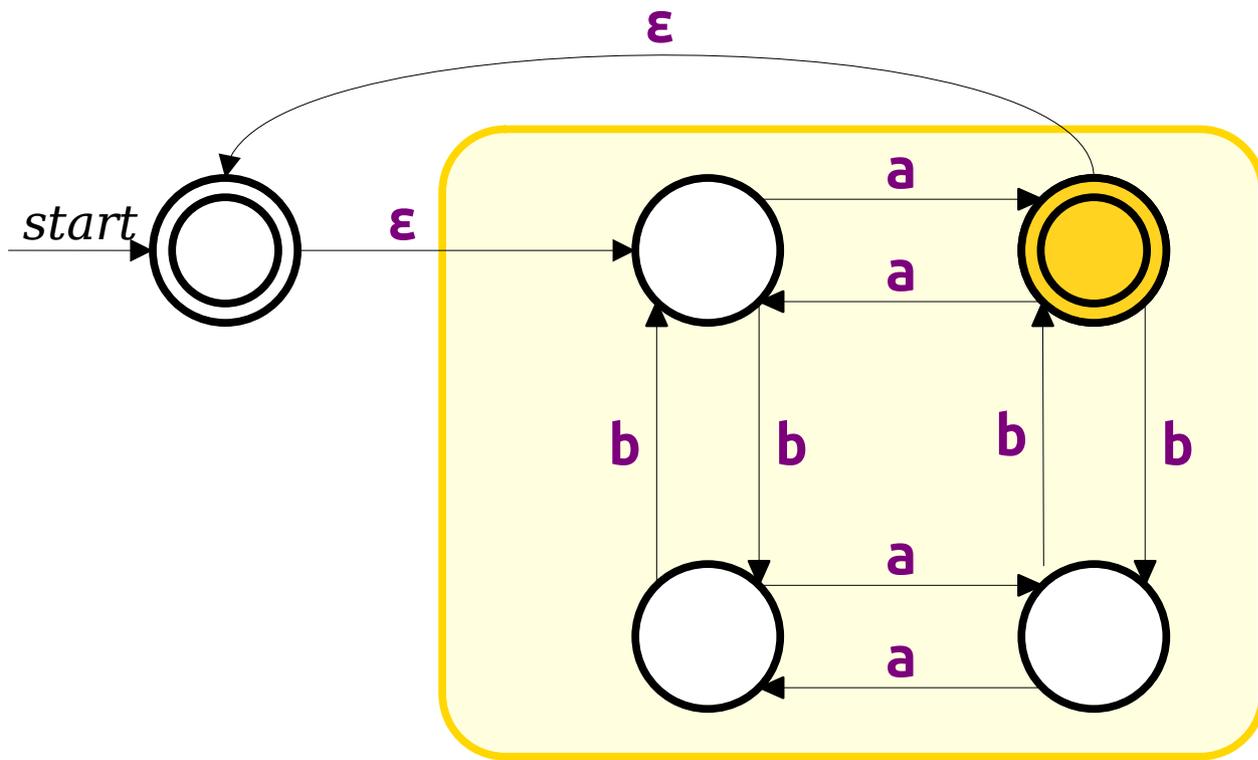


DFA for  $L$

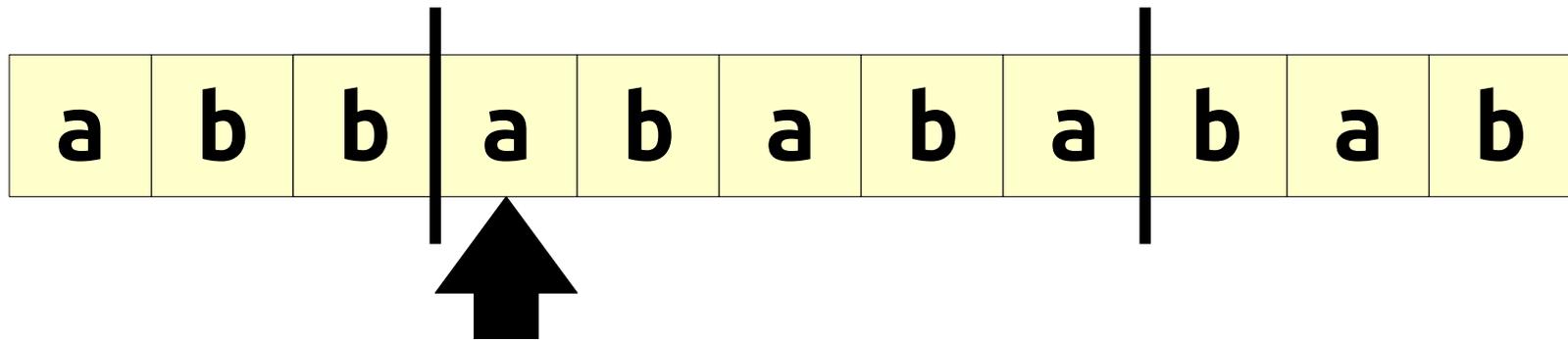


$L = \{ w \in \{a, b\}^* \mid w \text{ has an odd number of } a\text{'s and an even number of } b\text{'s} \}$

Construct an NFA for  $L^*$ .

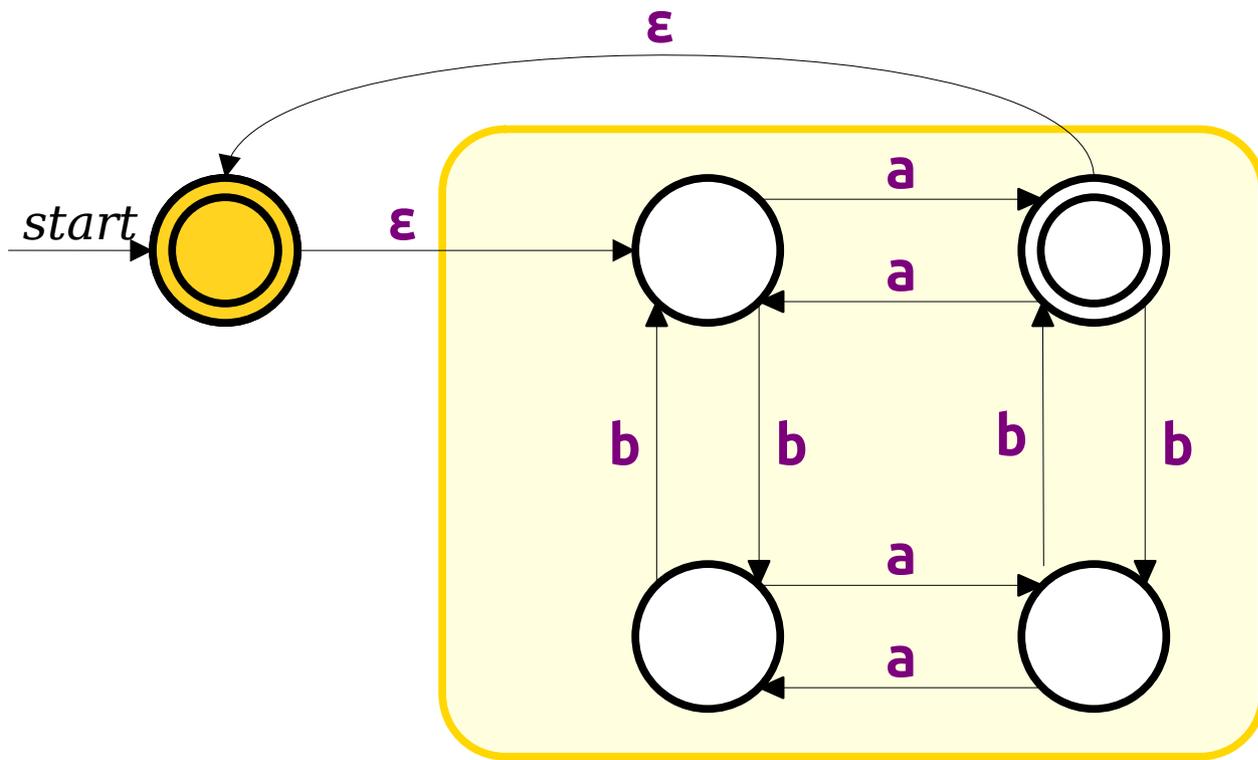


DFA for  $L$

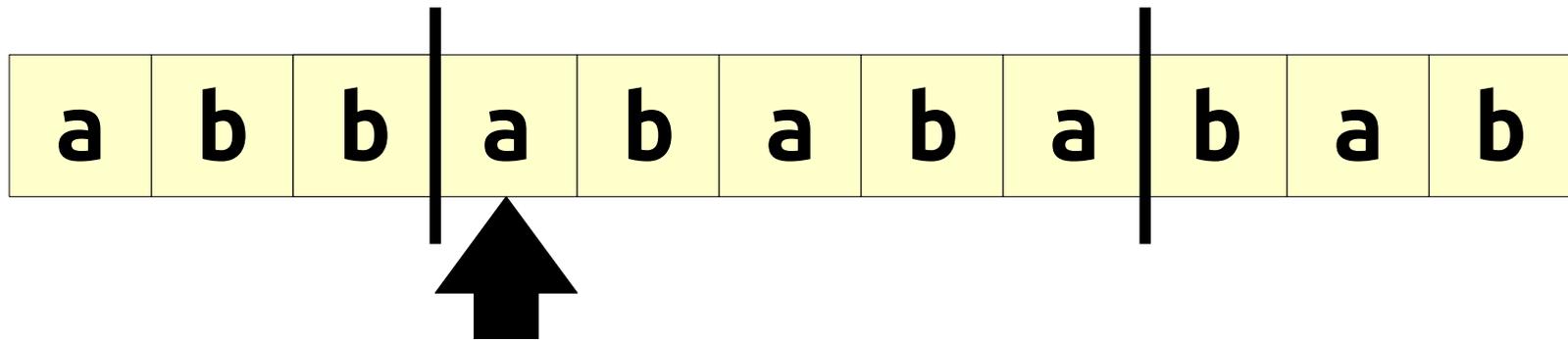


$L = \{ w \in \{a, b\}^* \mid w \text{ has an odd number of } a\text{'s and an even number of } b\text{'s} \}$

Construct an NFA for  $L^*$ .

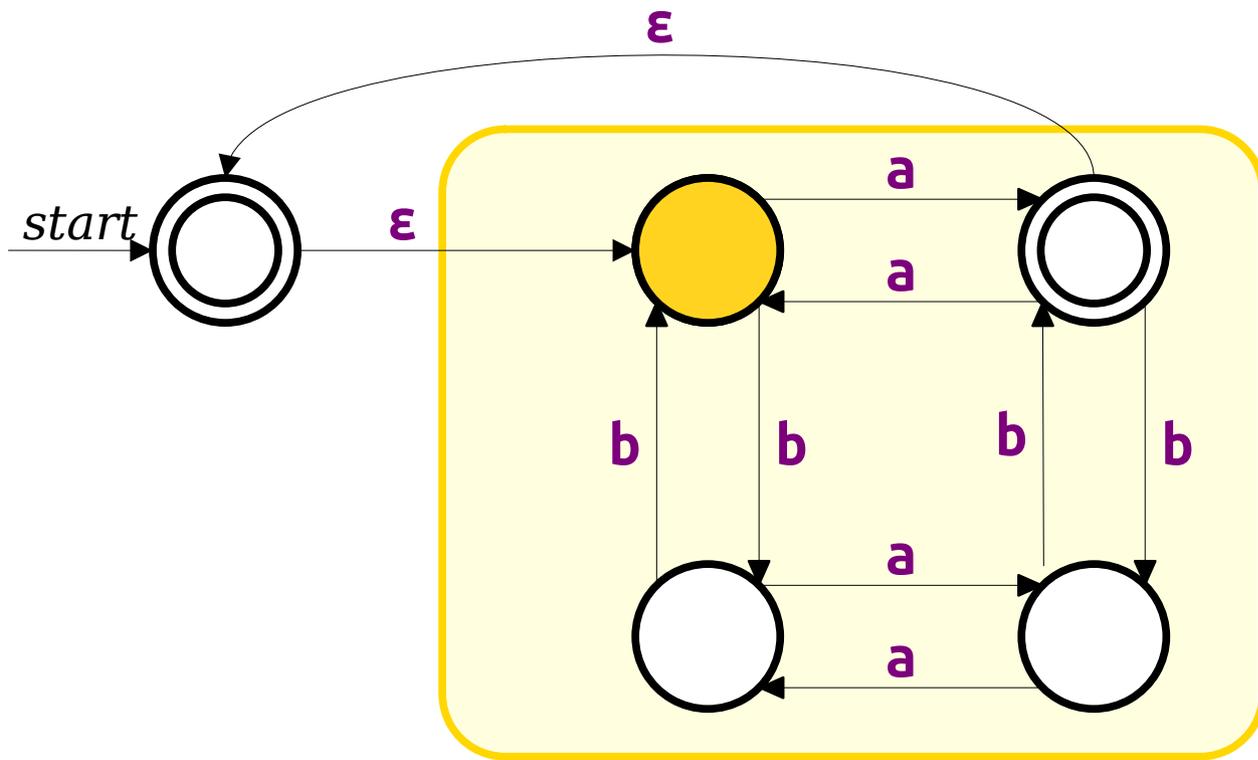


DFA for  $L$

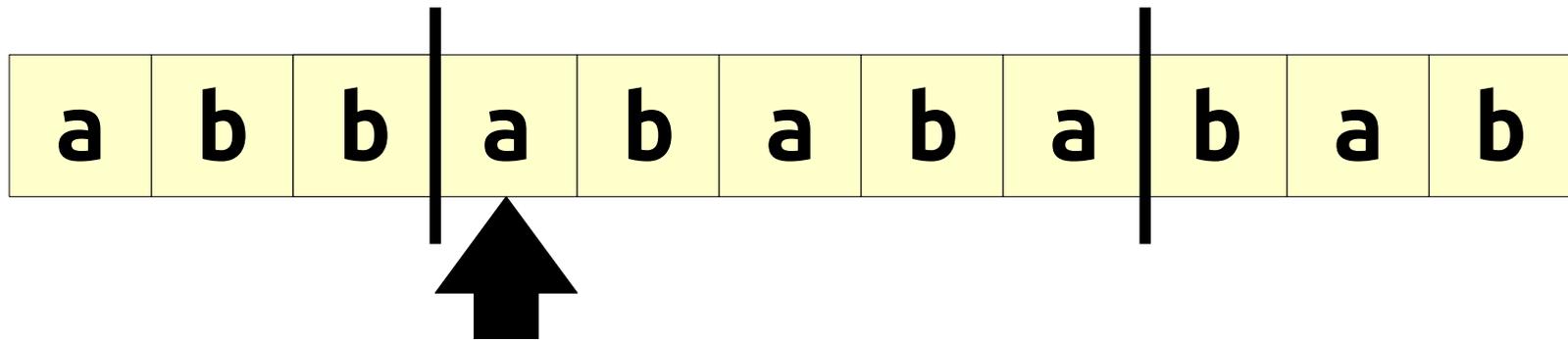


$L = \{ w \in \{a, b\}^* \mid w \text{ has an odd number of } a\text{'s and an even number of } b\text{'s} \}$

Construct an NFA for  $L^*$ .

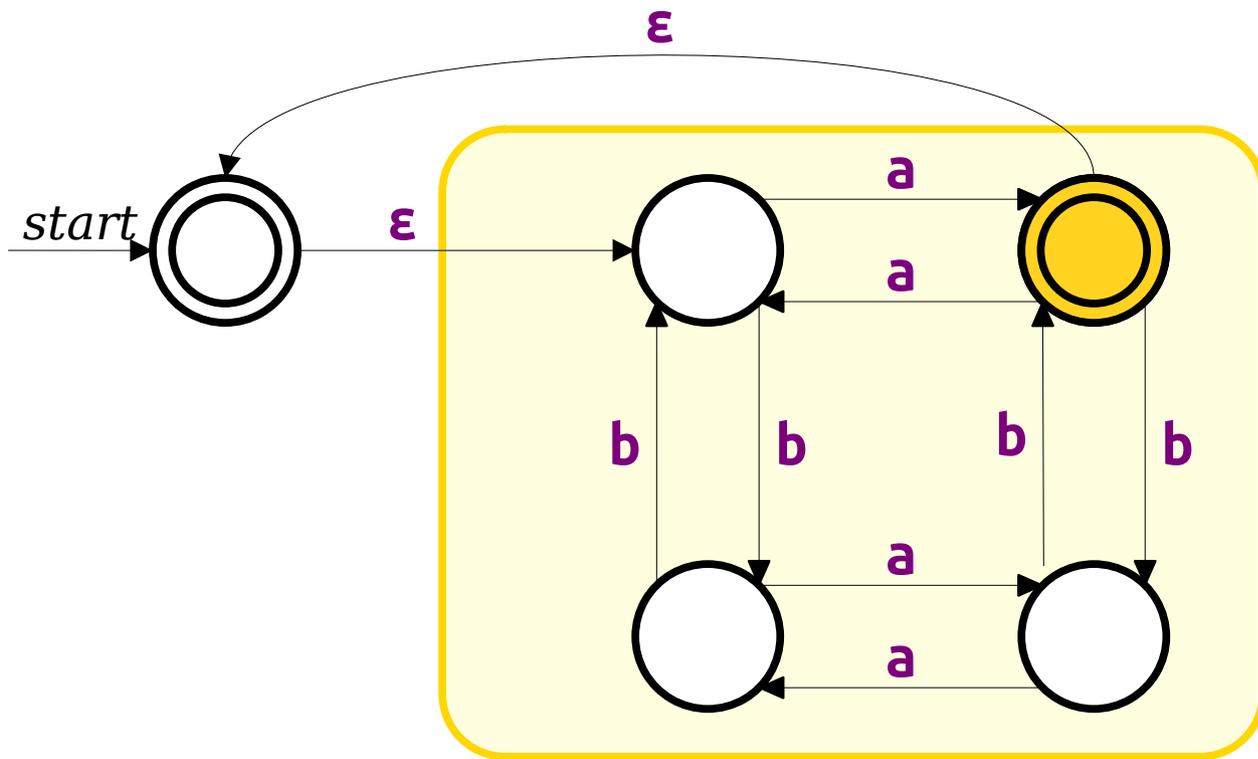


DFA for  $L$

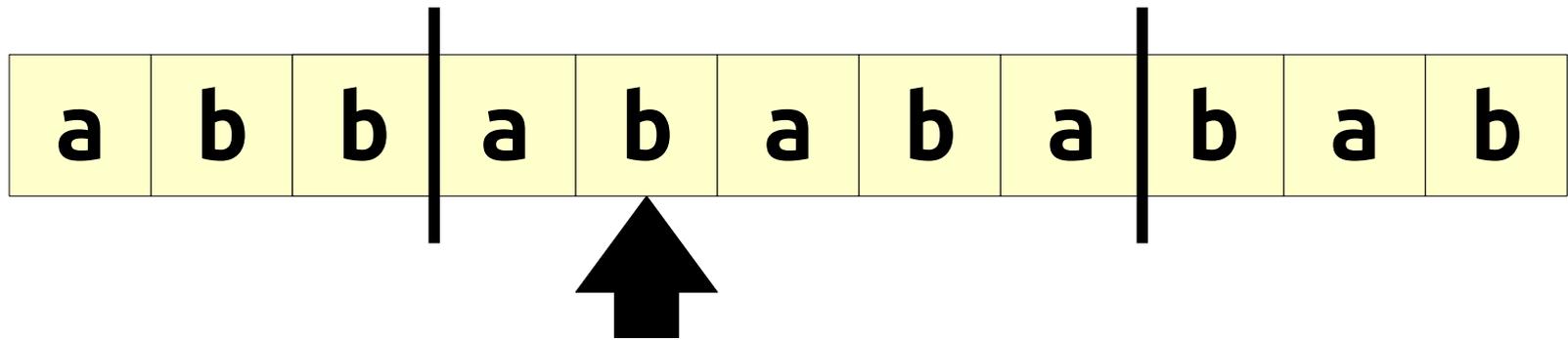


$L = \{ w \in \{a, b\}^* \mid w \text{ has an odd number of } a\text{'s and an even number of } b\text{'s} \}$

Construct an NFA for  $L^*$ .

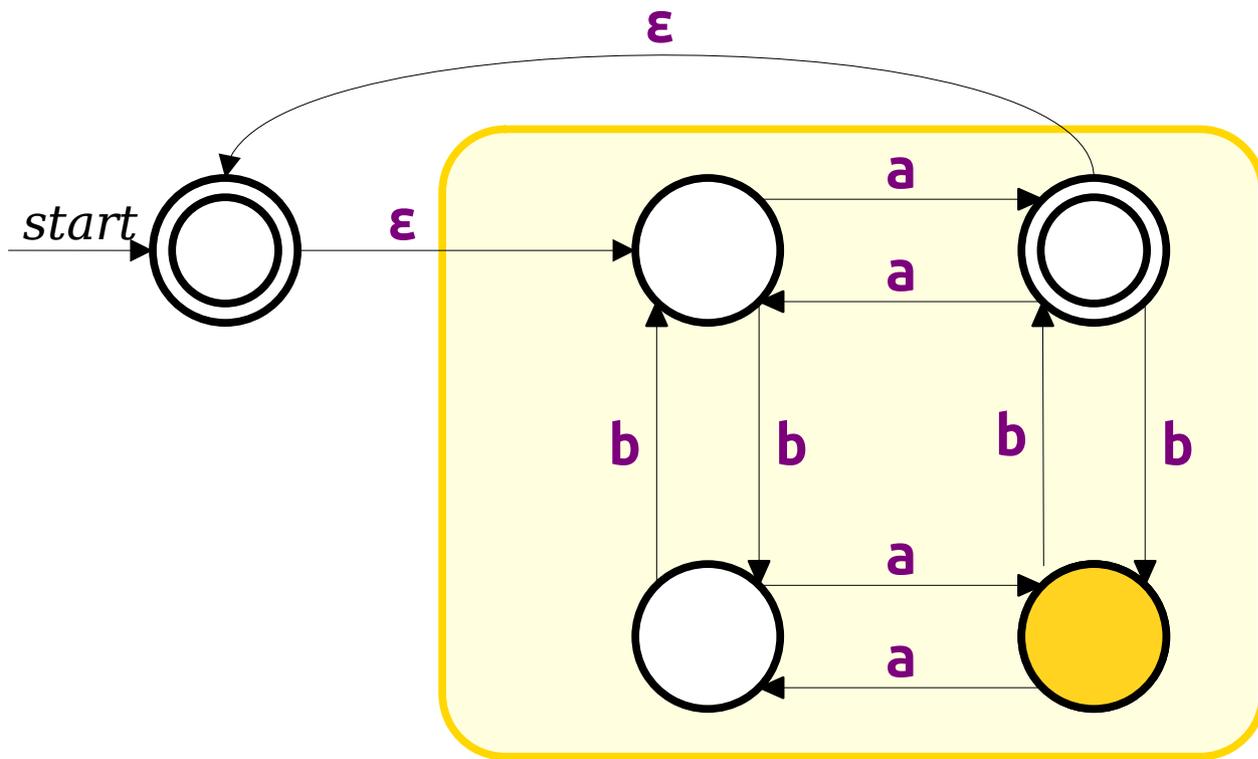


DFA for  $L$

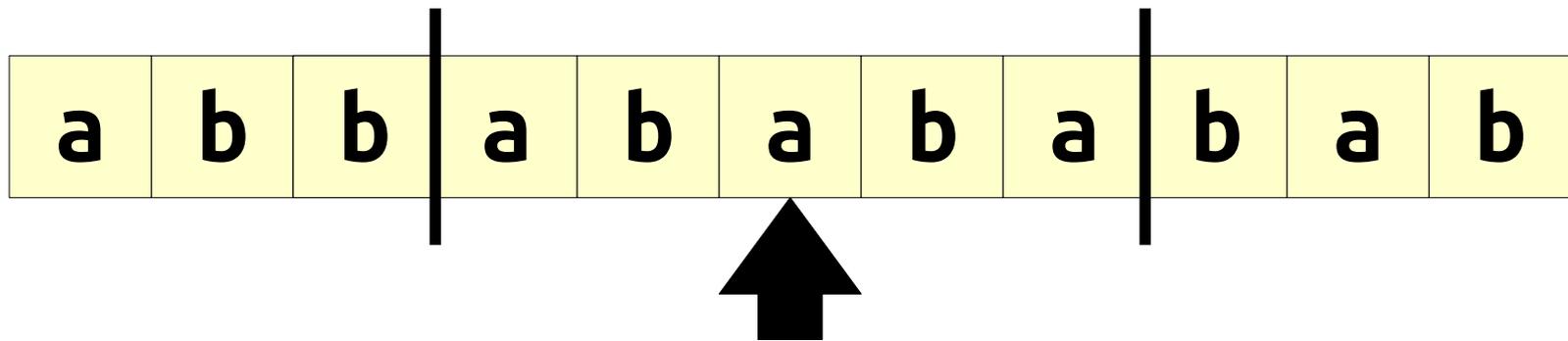


$L = \{ w \in \{a, b\}^* \mid w \text{ has an odd number of } a\text{'s and an even number of } b\text{'s} \}$

Construct an NFA for  $L^*$ .

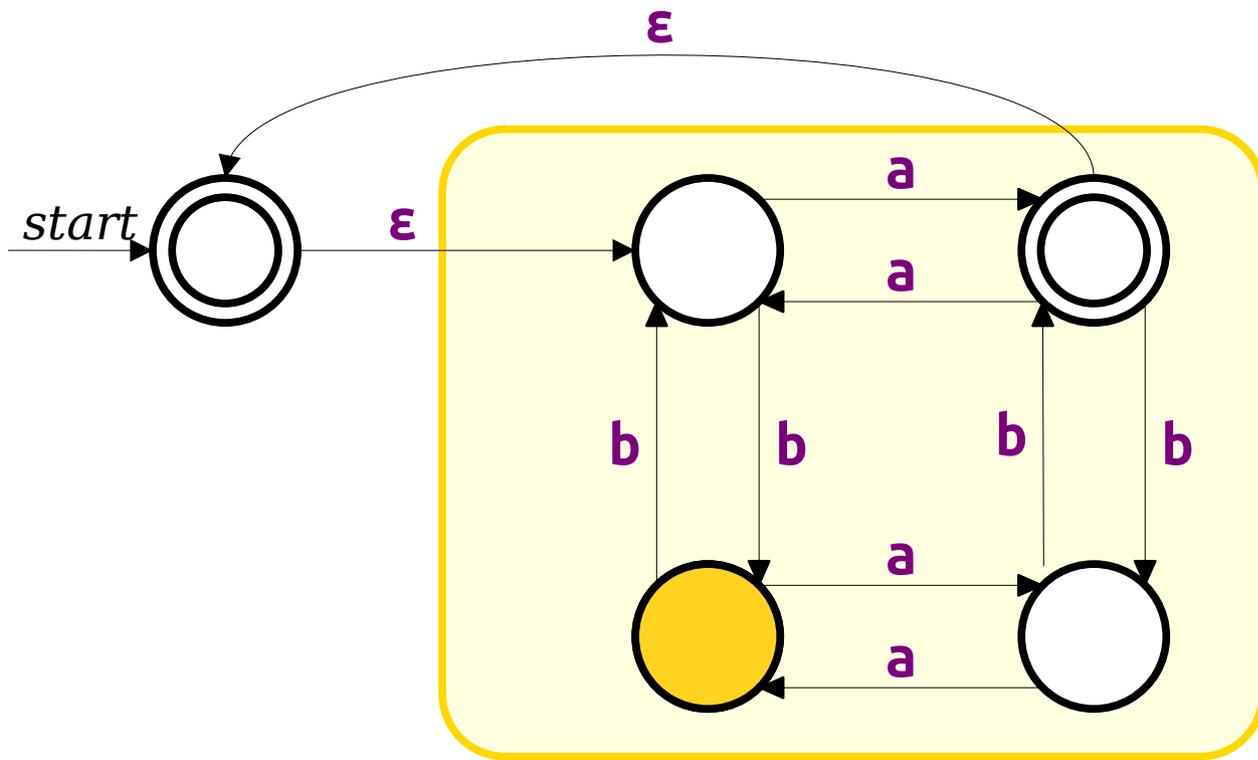


DFA for  $L$

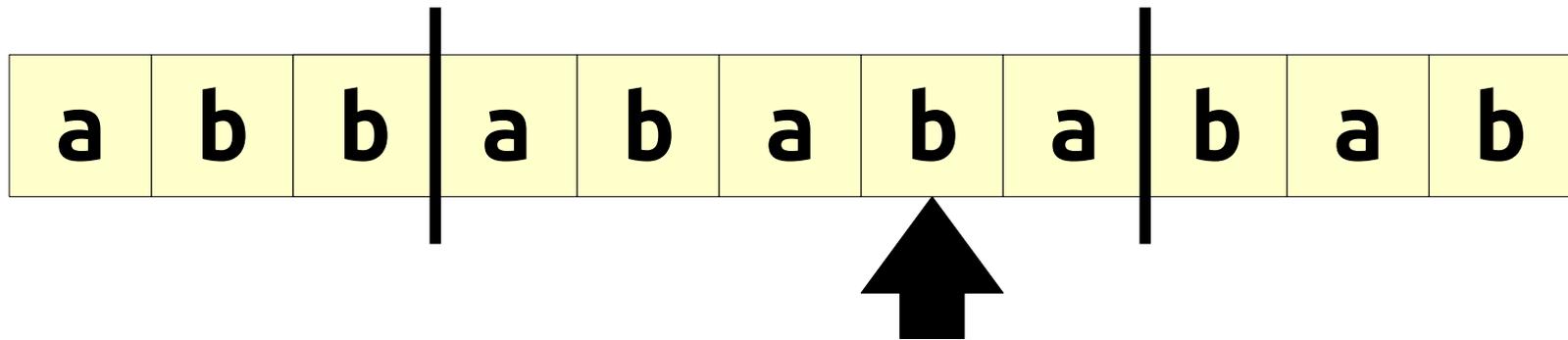


$L = \{ w \in \{a, b\}^* \mid w \text{ has an odd number of } a\text{'s and an even number of } b\text{'s} \}$

Construct an NFA for  $L^*$ .

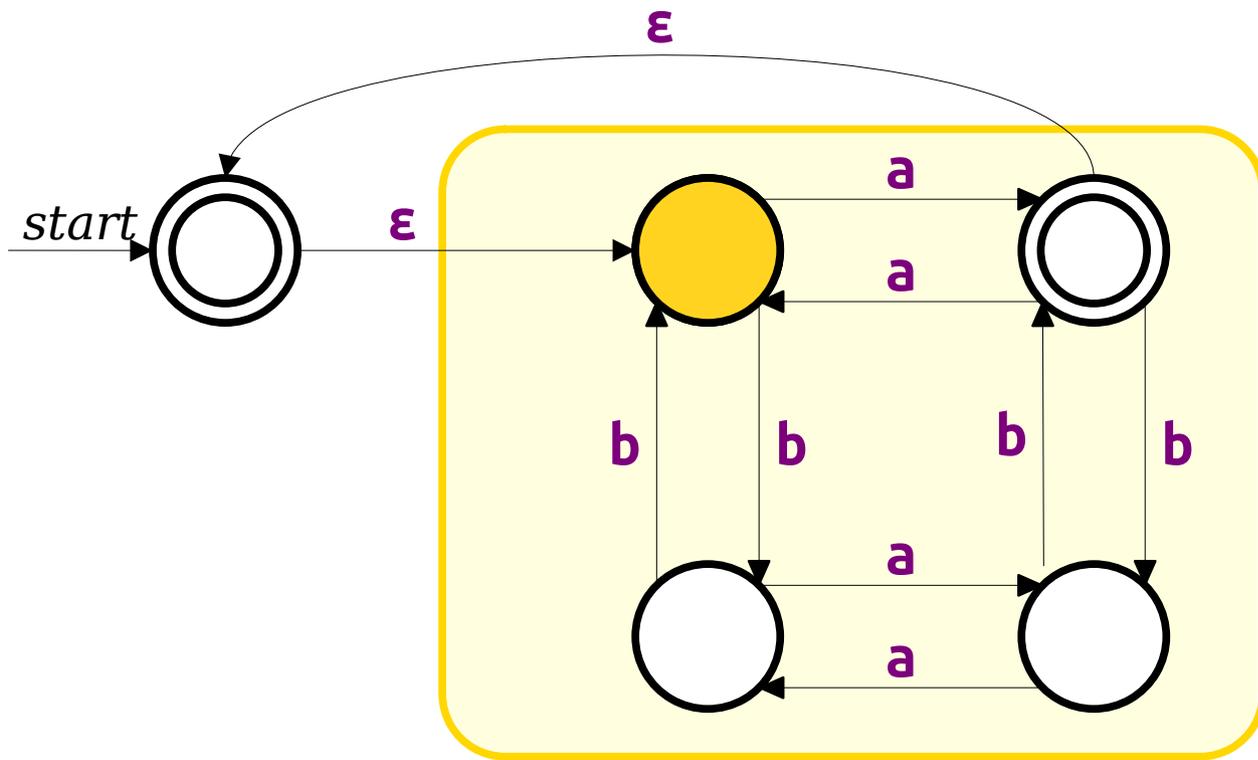


DFA for  $L$

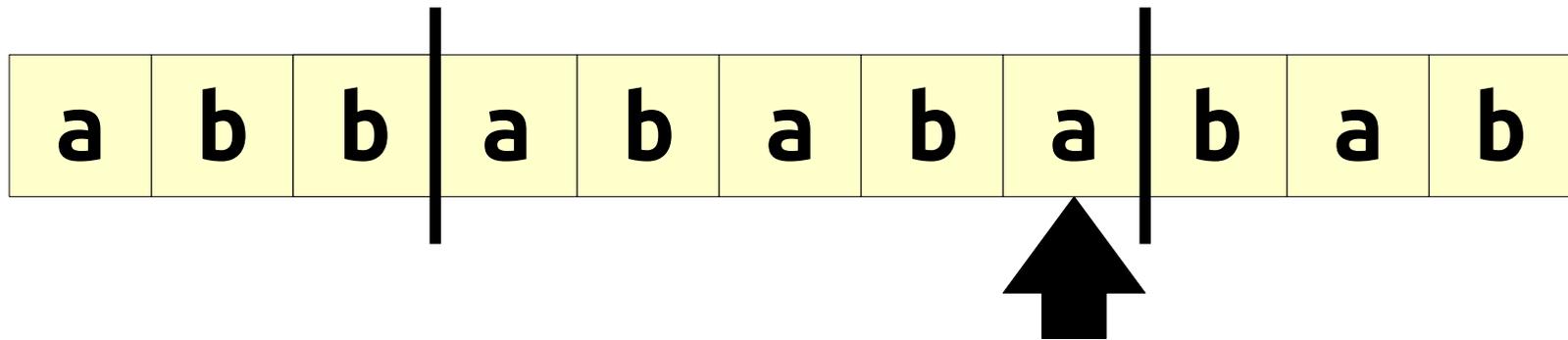


$L = \{ w \in \{a, b\}^* \mid w \text{ has an odd number of } a\text{'s and an even number of } b\text{'s} \}$

Construct an NFA for  $L^*$ .

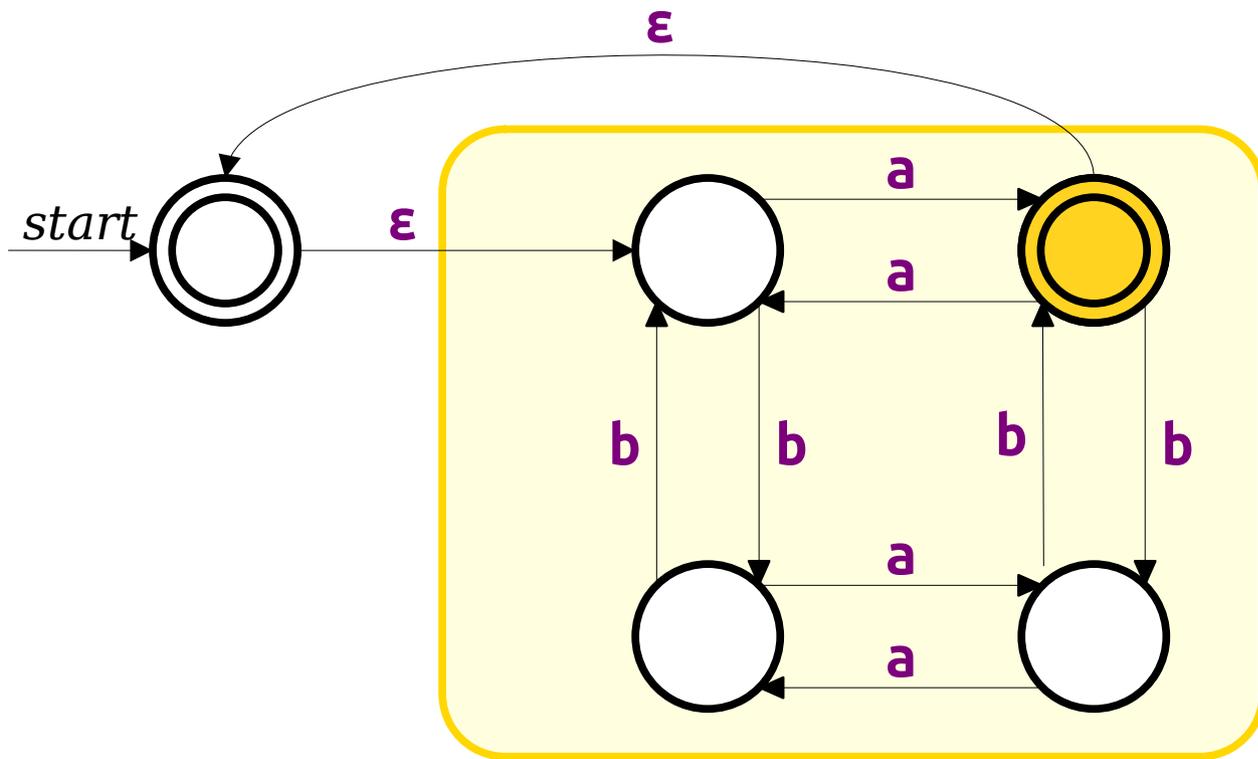


DFA for  $L$

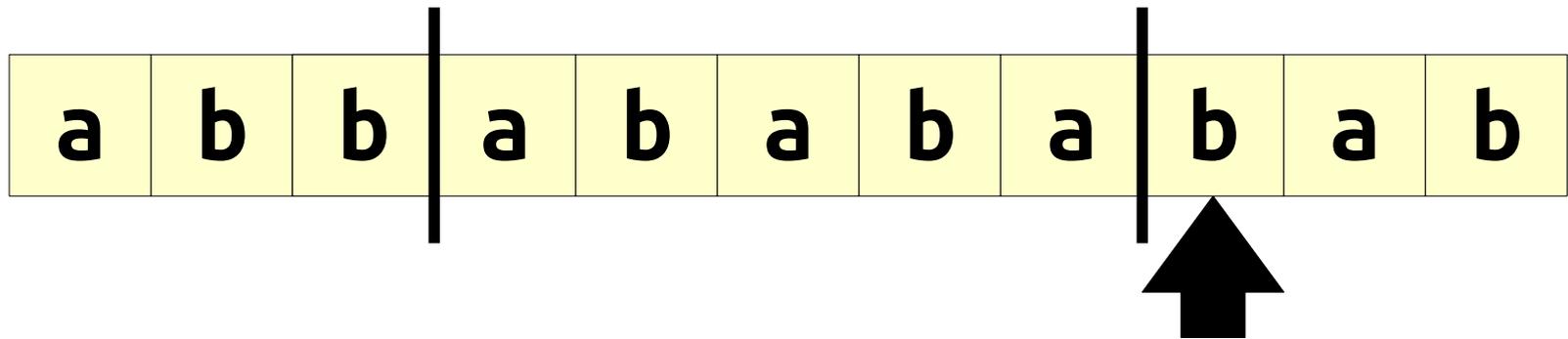


$L = \{ w \in \{a, b\}^* \mid w \text{ has an odd number of } a\text{'s and an even number of } b\text{'s} \}$

Construct an NFA for  $L^*$ .

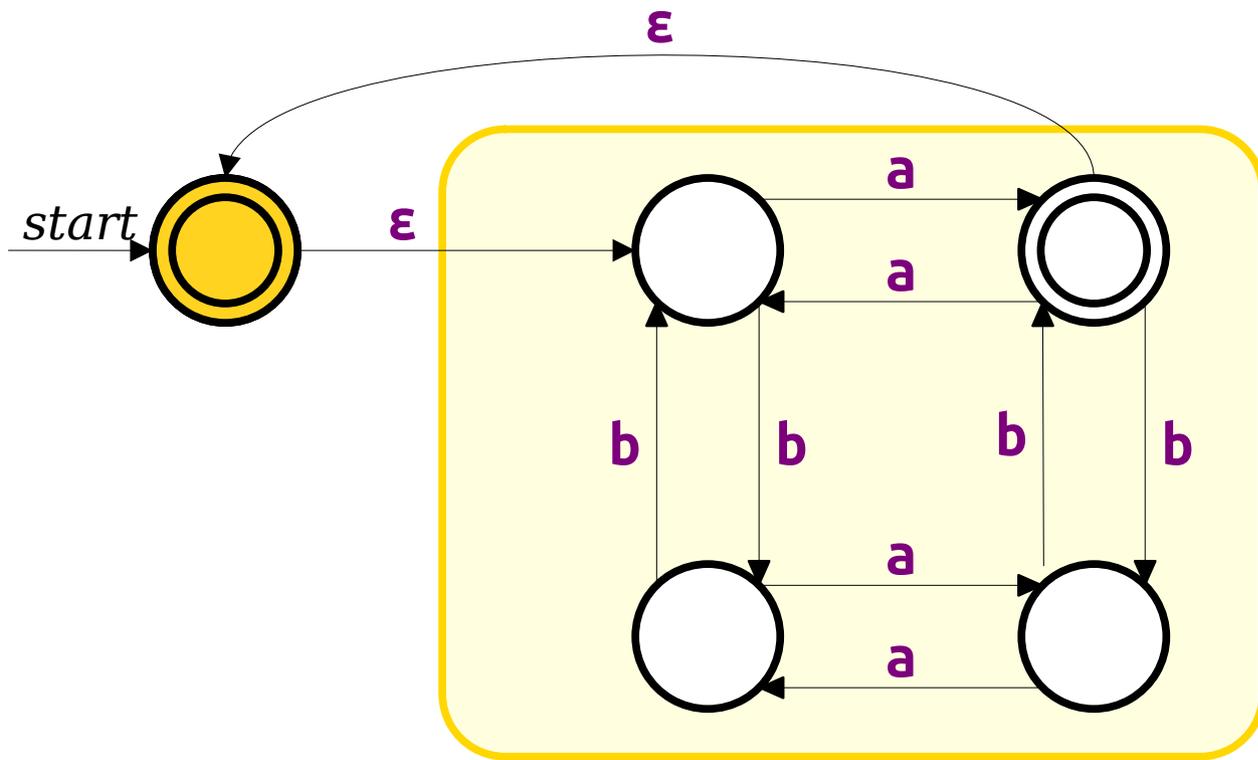


DFA for  $L$

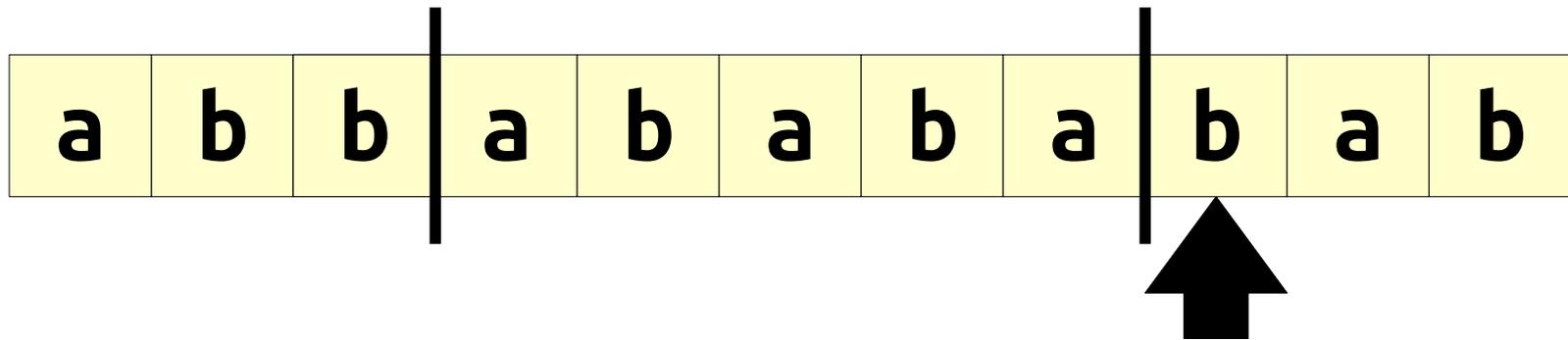


$L = \{ w \in \{a, b\}^* \mid w \text{ has an odd number of } a\text{'s and an even number of } b\text{'s} \}$

Construct an NFA for  $L^*$ .

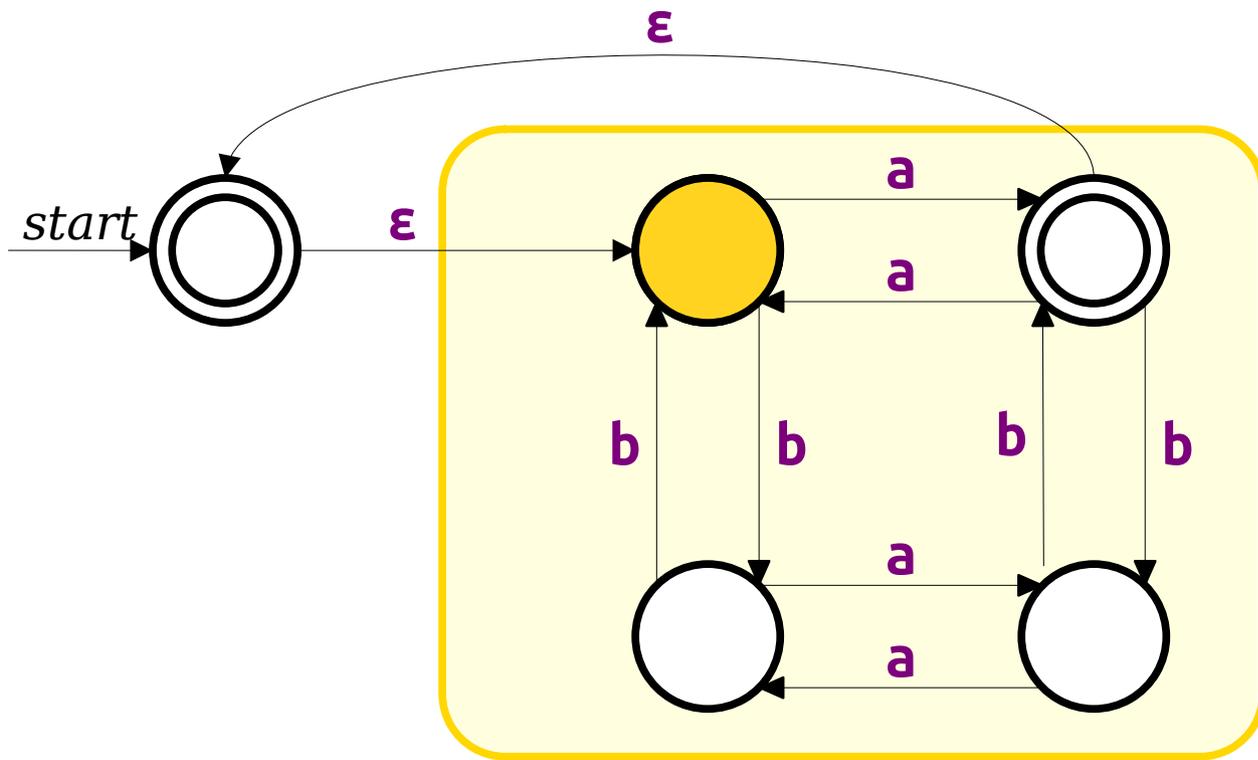


DFA for  $L$

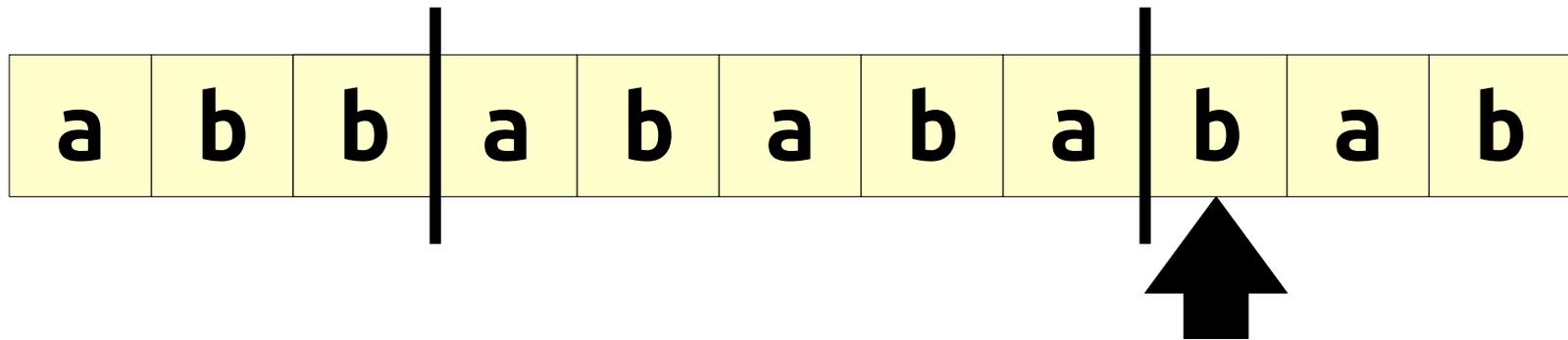


$L = \{ w \in \{a, b\}^* \mid w \text{ has an odd number of } a\text{'s and an even number of } b\text{'s} \}$

Construct an NFA for  $L^*$ .

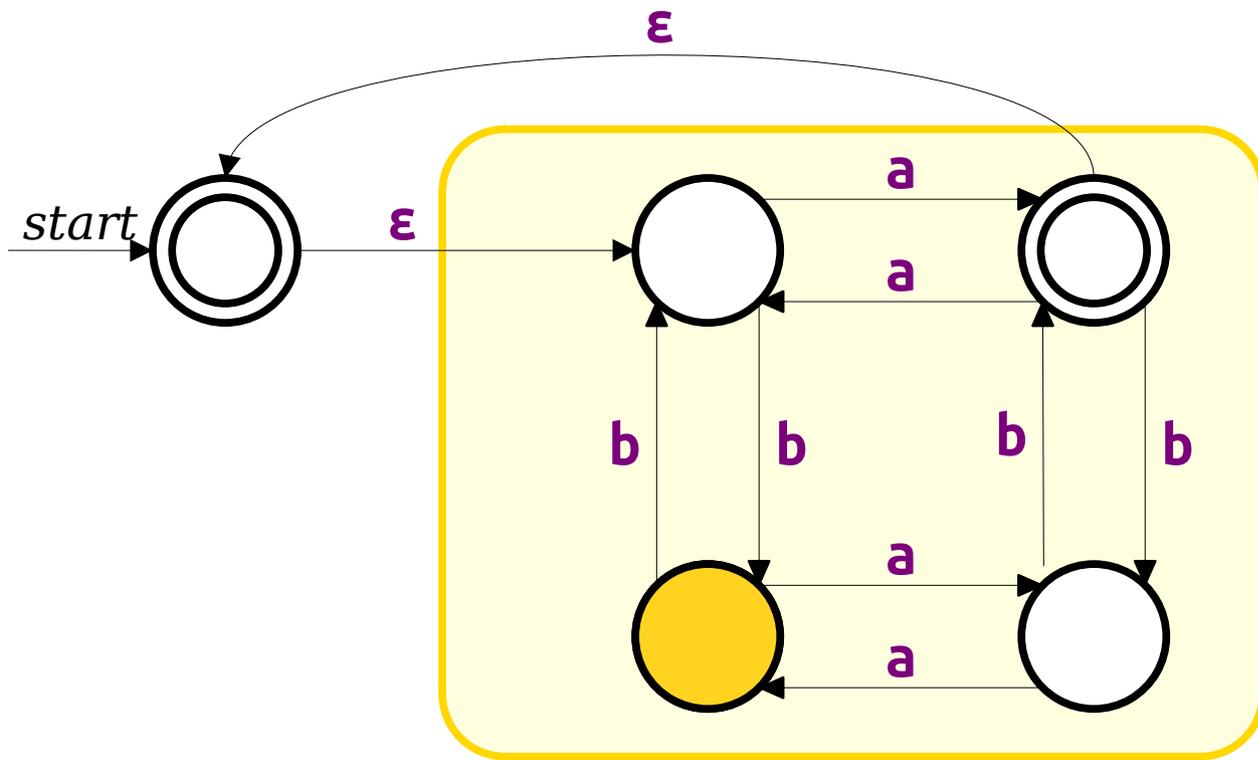


DFA for  $L$

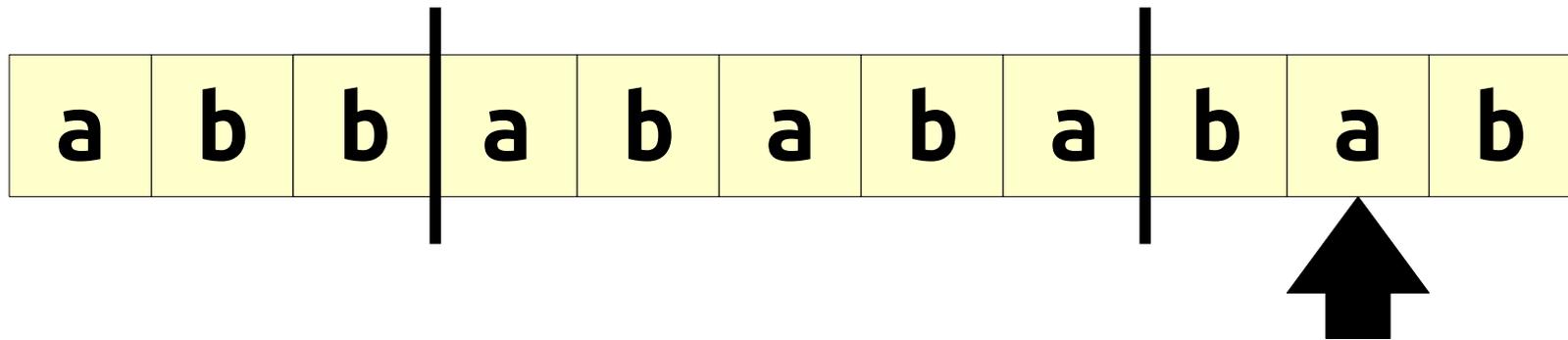


$L = \{ w \in \{a, b\}^* \mid w \text{ has an odd number of } a\text{'s and an even number of } b\text{'s} \}$

Construct an NFA for  $L^*$ .

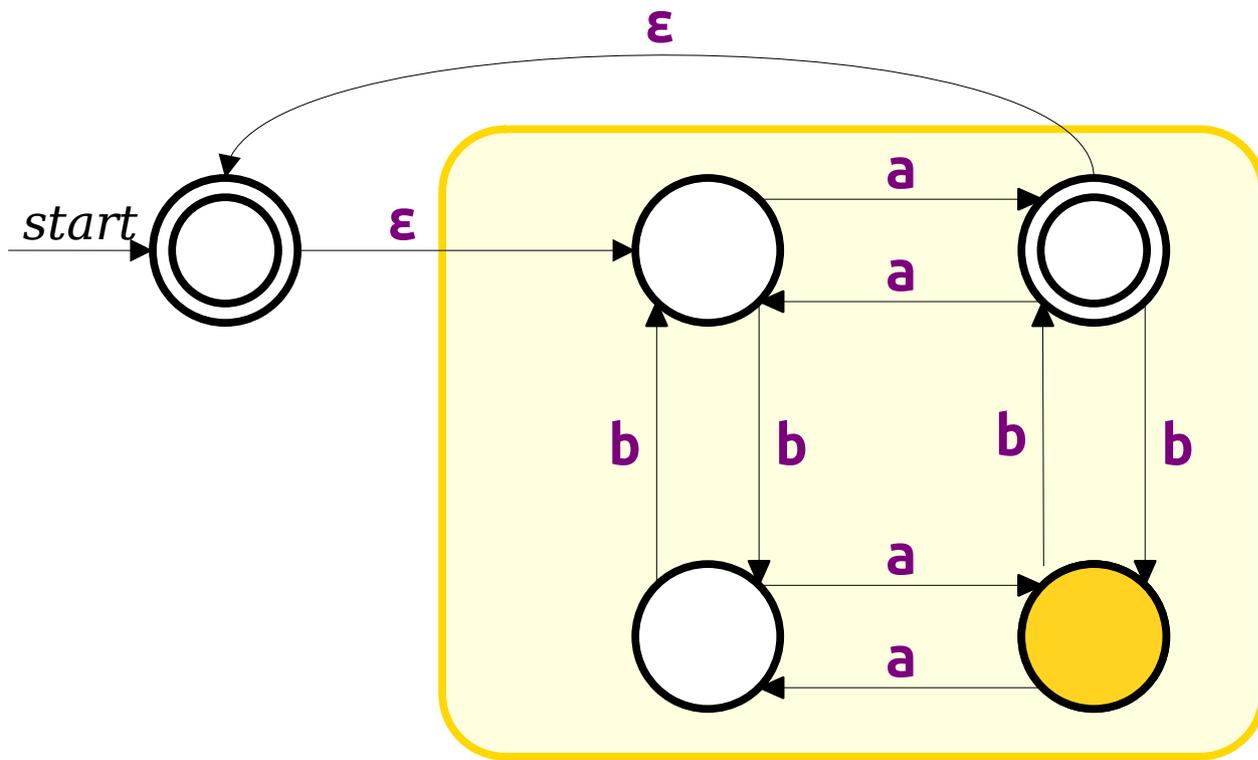


DFA for  $L$

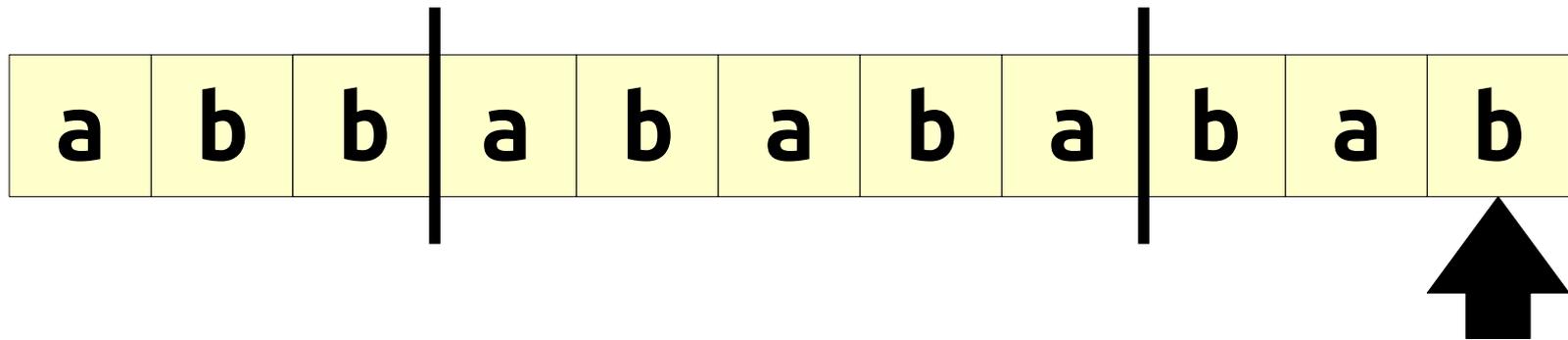


$L = \{ w \in \{a, b\}^* \mid w \text{ has an odd number of } a\text{'s and an even number of } b\text{'s} \}$

Construct an NFA for  $L^*$ .

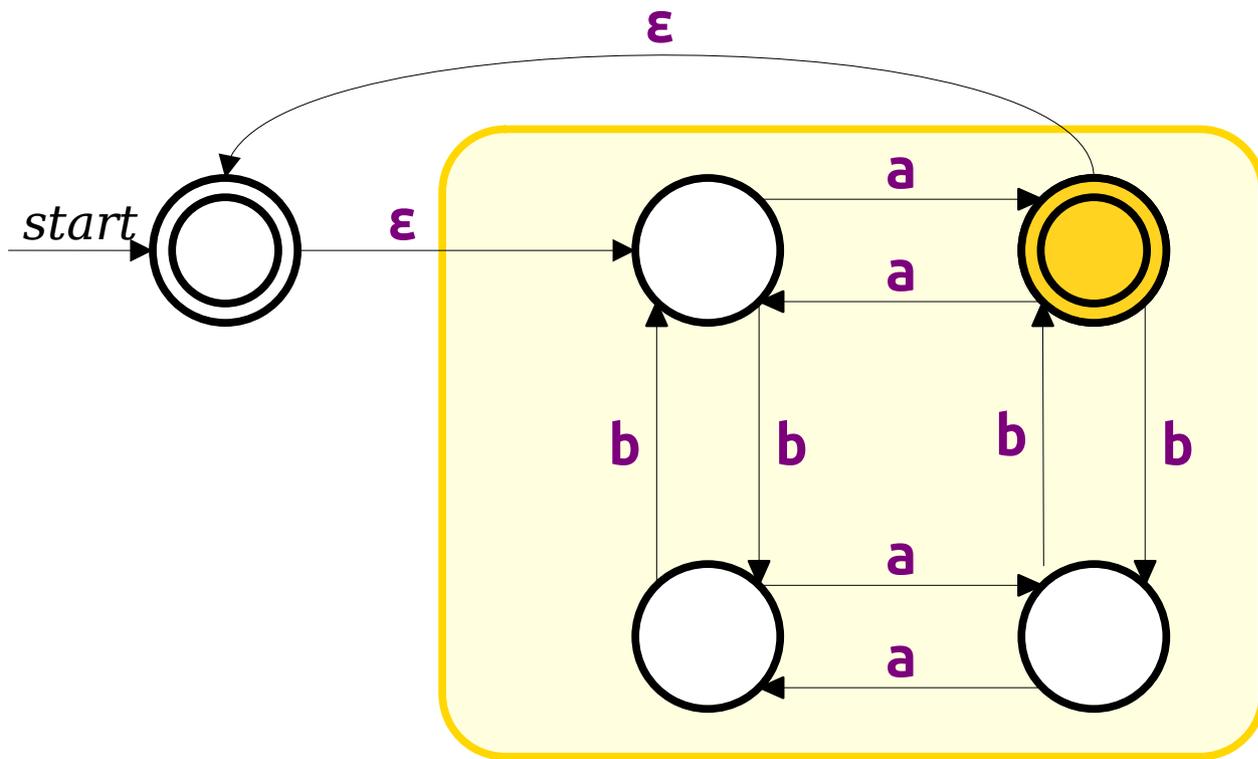


DFA for  $L$

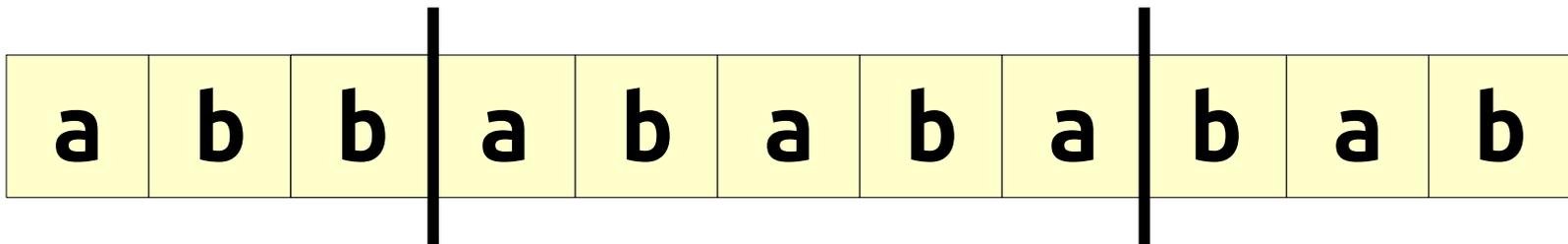


$L = \{ w \in \{a, b\}^* \mid w \text{ has an odd number of } a\text{'s and an even number of } b\text{'s} \}$

Construct an NFA for  $L^*$ .



DFA for  $L$



$L = \{ w \in \{a, b\}^* \mid w \text{ has an odd number of } a\text{'s and an even number of } b\text{'s} \}$

Construct an NFA for  $L^*$ .

## ***Appendix 2:*** More Closure Properties

# The Complement of a Language

- Given a language  $L \subseteq \Sigma^*$ , the **complement** of that language (denoted  $\bar{L}$ ) is the language of all strings in  $\Sigma^*$  that aren't in  $L$ .
- Formally:

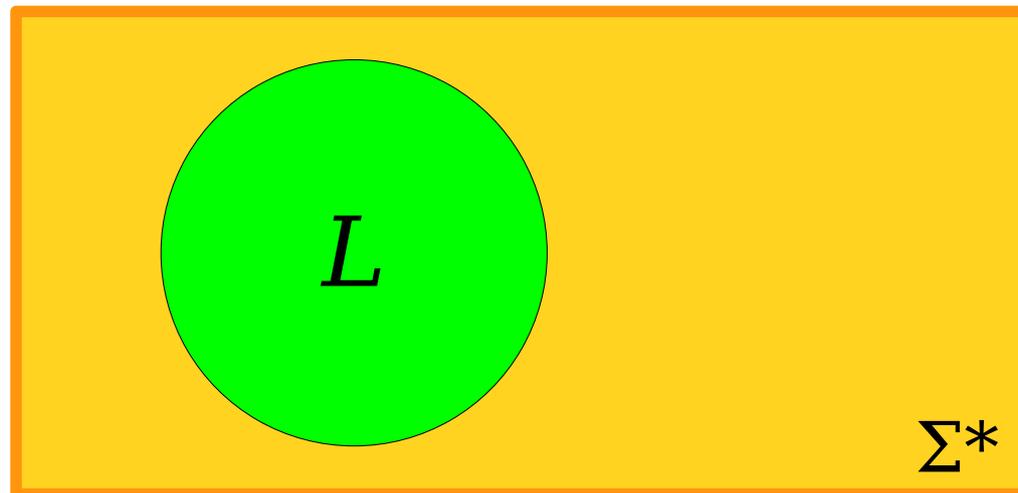
$$\bar{L} = \Sigma^* - L$$



# The Complement of a Language

- Given a language  $L \subseteq \Sigma^*$ , the **complement** of that language (denoted  $\bar{L}$ ) is the language of all strings in  $\Sigma^*$  that aren't in  $L$ .
- Formally:

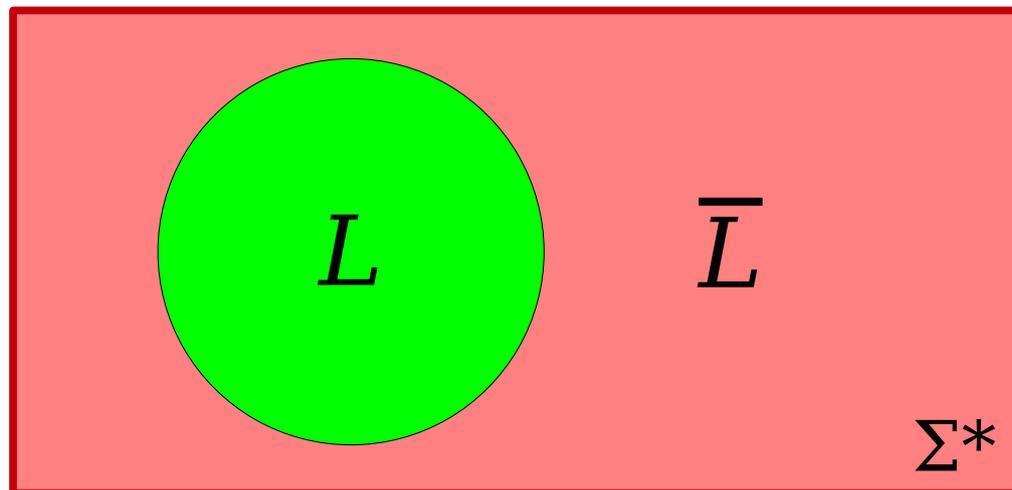
$$\bar{L} = \Sigma^* - L$$



# The Complement of a Language

- Given a language  $L \subseteq \Sigma^*$ , the **complement** of that language (denoted  $\bar{L}$ ) is the language of all strings in  $\Sigma^*$  that aren't in  $L$ .
- Formally:

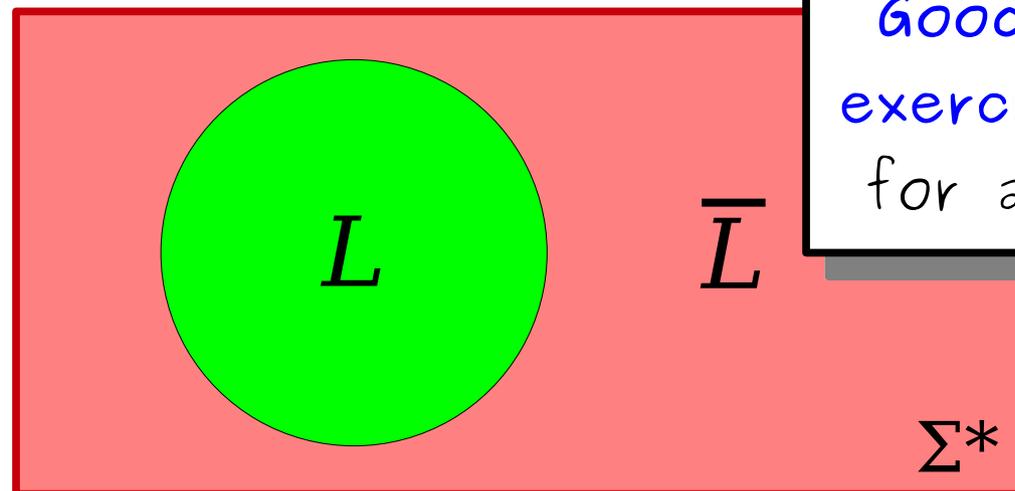
$$\bar{L} = \Sigma^* - L$$



# The Complement of a Language

- Given a language  $L \subseteq \Sigma^*$ , the **complement** of that language (denoted  $\bar{L}$ ) is the language of all strings in  $\Sigma^*$  that aren't in  $L$ .
- Formally:

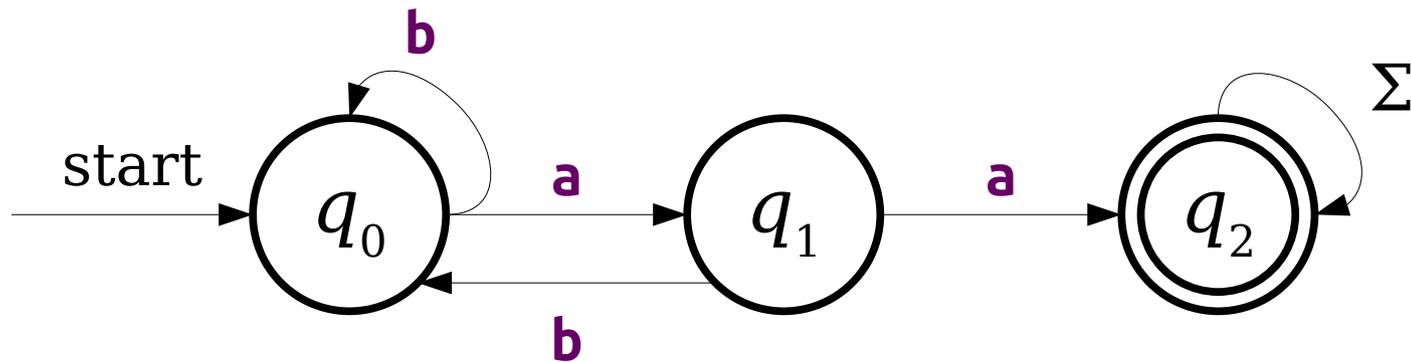
$$\bar{L} = \Sigma^* - L$$



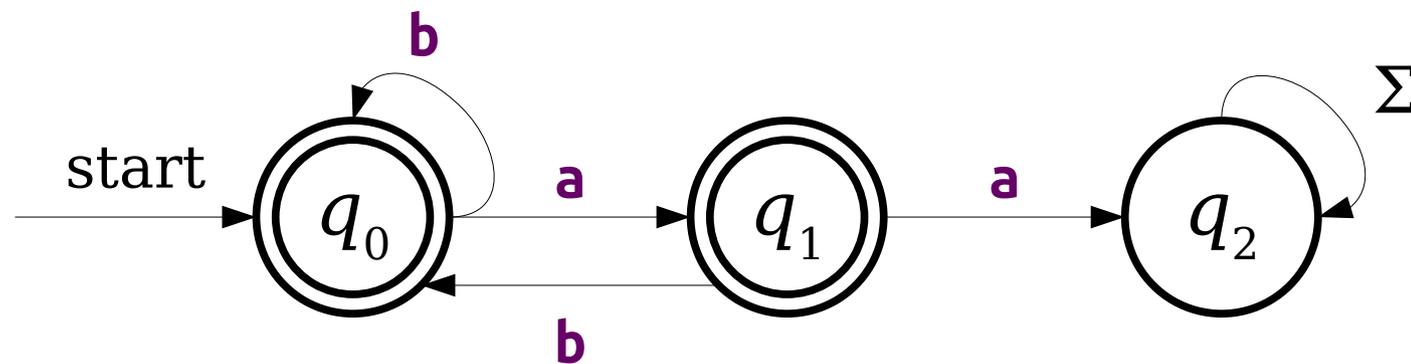
Good proofwriting  
exercise: prove  $\bar{\bar{L}} = L$   
for any language  $L$ .

# Complementing Regular Languages

$$L = \{ w \in \{a, b\}^* \mid w \text{ contains } \mathbf{aa} \text{ as a substring} \}$$

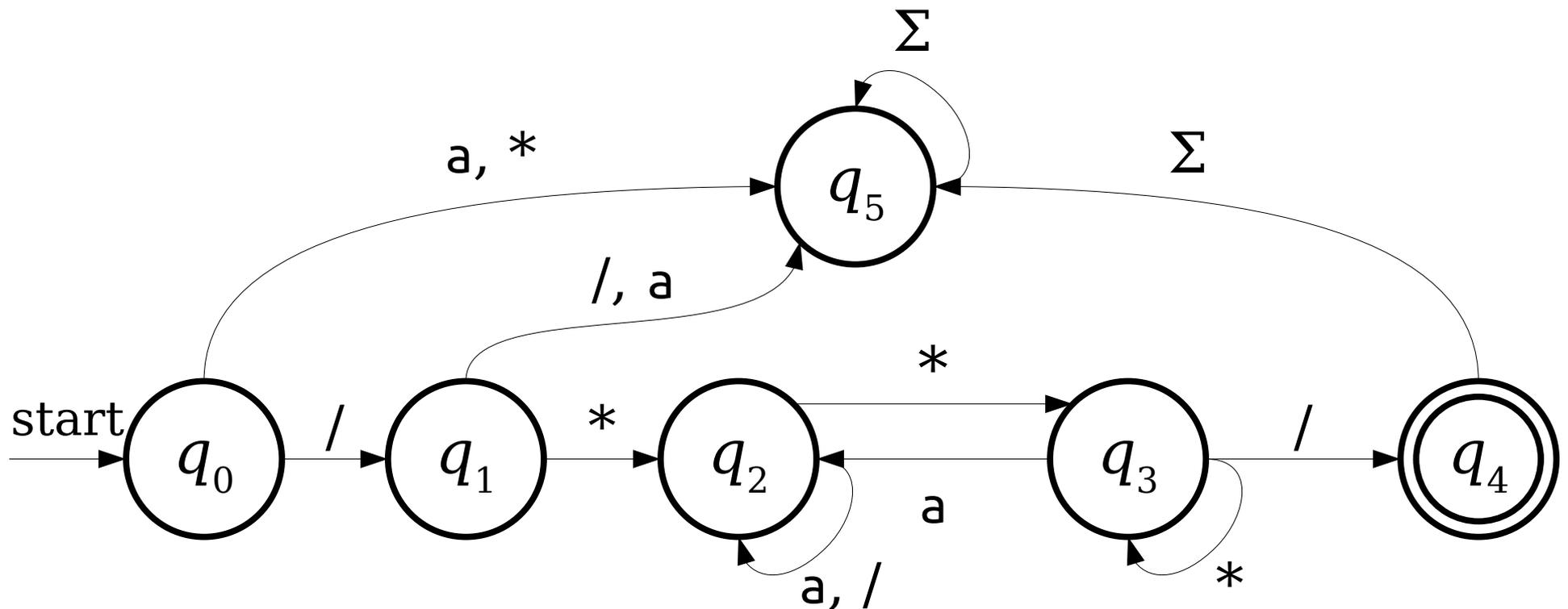


$$\bar{L} = \{ w \in \{a, b\}^* \mid w \text{ **does not** contain } \mathbf{aa} \text{ as a substring} \}$$



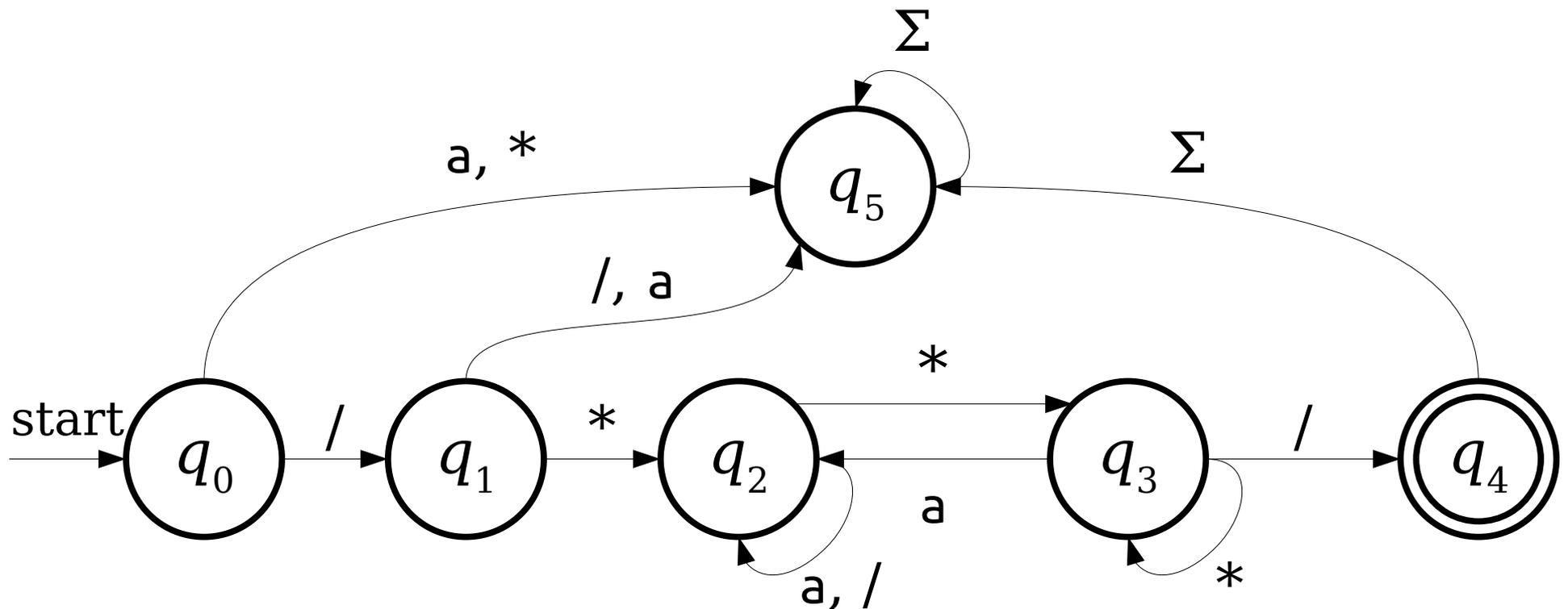
# Complementing Regular Languages

$L = \{ w \in \{a, *, /\}^* \mid w \text{ represents a C-style comment} \}$



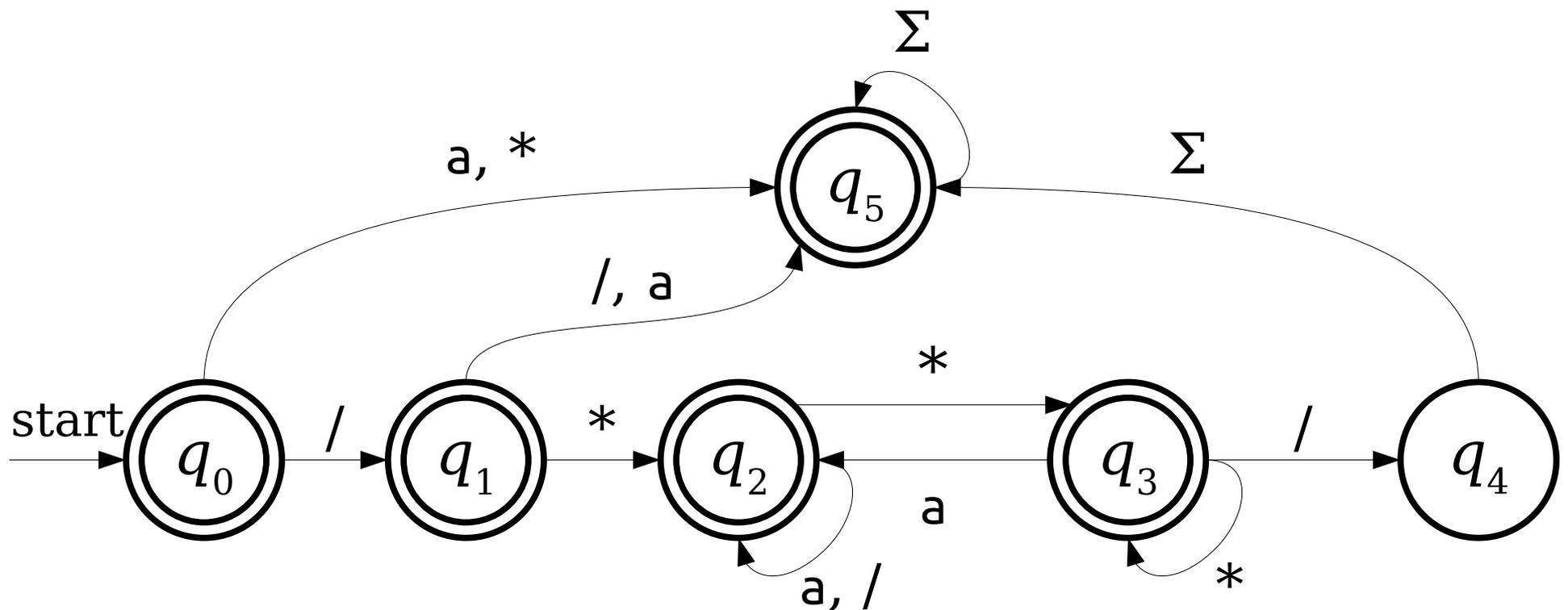
# Complementing Regular Languages

$\bar{L} = \{ w \in \{a, *, /\}^* \mid w \text{ *doesn't* represent a C-style comment} \}$



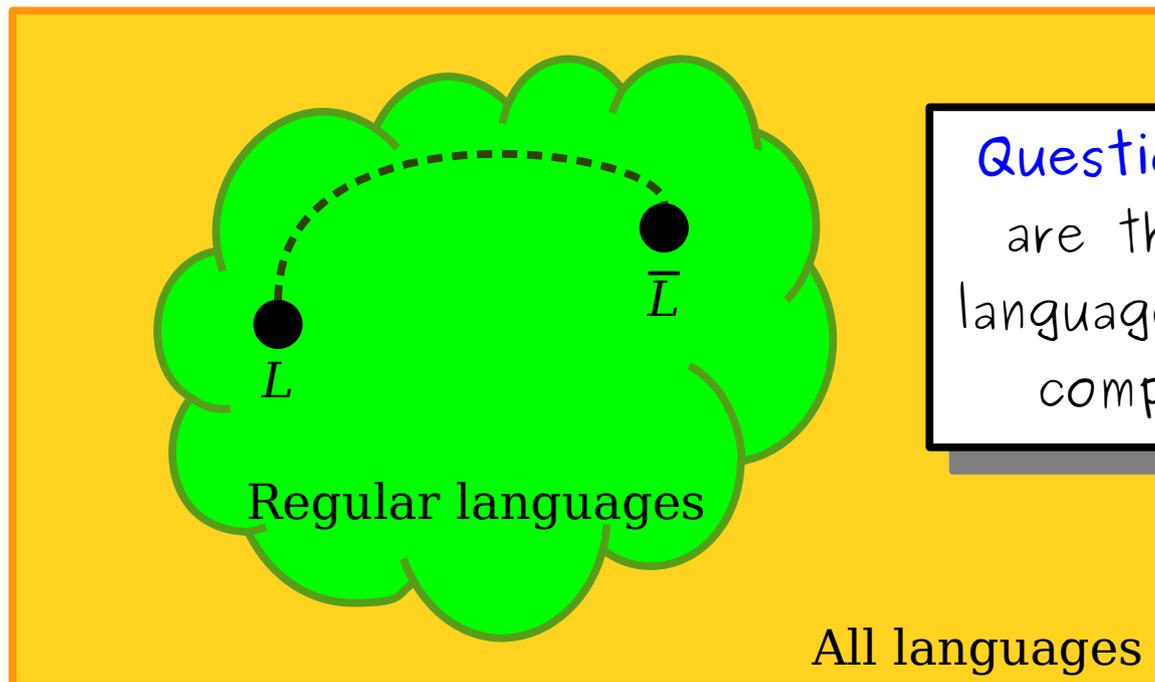
# Complementing Regular Languages

$\bar{L} = \{ w \in \{a, *, /\}^* \mid w \text{ *doesn't* represent a C-style comment} \}$



# Closure Properties

- **Theorem:** If  $L$  is a regular language, then  $\bar{L}$  is also a regular language.
- As a result, we say that the regular languages are **closed under complementation**.



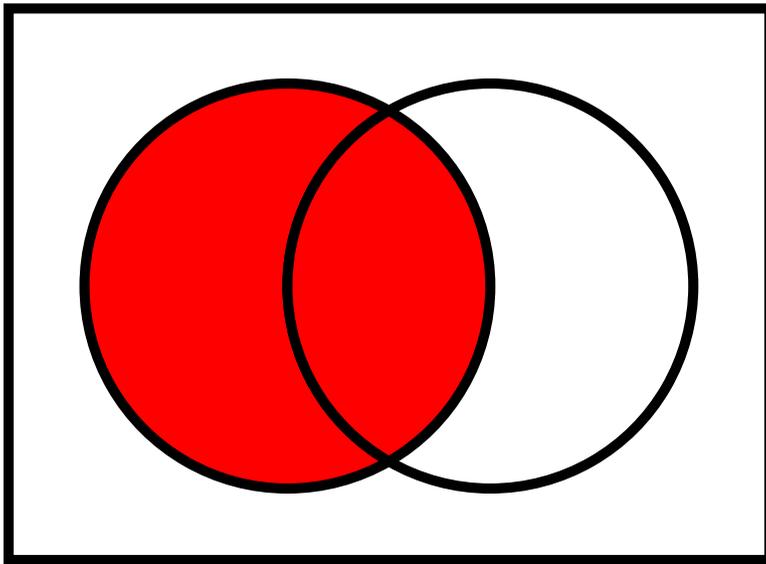
Question to ponder:  
are the *nonregular*  
languages closed under  
complementation?

# Closure Under Intersection

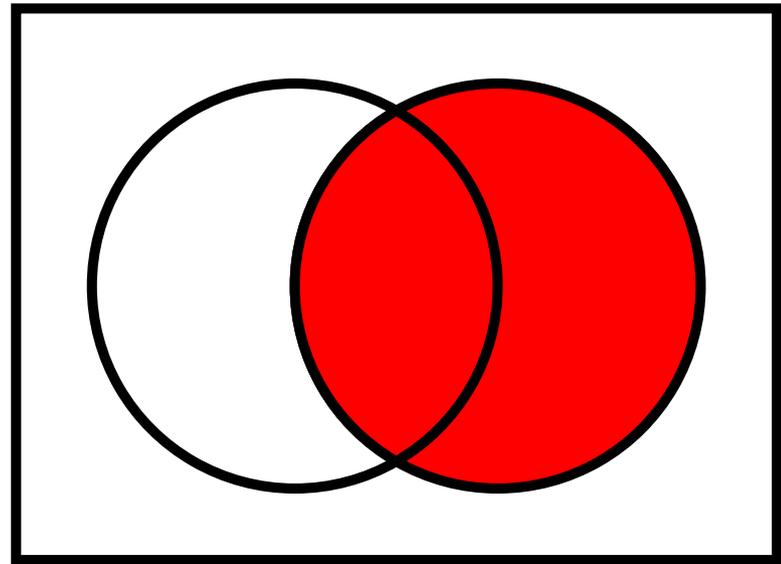
- If  $L_1$  and  $L_2$  are languages over  $\Sigma$ , then  $L_1 \cap L_2$  is the language of strings in both  $L_1$  and  $L_2$ .
- Question: If  $L_1$  and  $L_2$  are regular, is  $L_1 \cap L_2$  regular as well?

# Closure Under Intersection

- If  $L_1$  and  $L_2$  are languages over  $\Sigma$ , then  $L_1 \cap L_2$  is the language of strings in both  $L_1$  and  $L_2$ .
- Question: If  $L_1$  and  $L_2$  are regular, is  $L_1 \cap L_2$  regular as well?



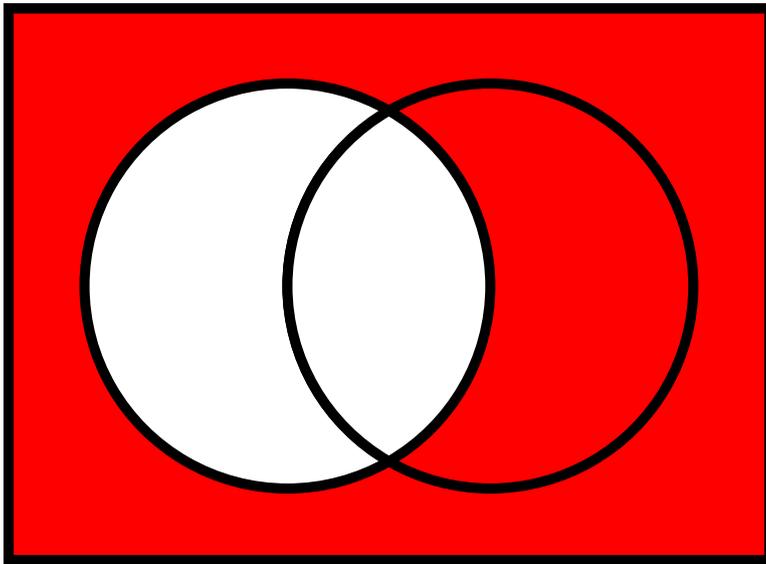
$L_1$



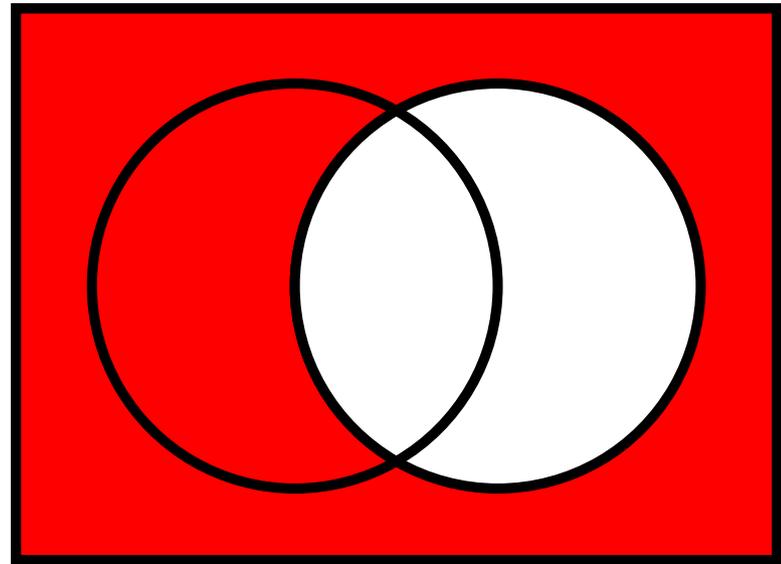
$L_2$

# Closure Under Intersection

- If  $L_1$  and  $L_2$  are languages over  $\Sigma$ , then  $L_1 \cap L_2$  is the language of strings in both  $L_1$  and  $L_2$ .
- Question: If  $L_1$  and  $L_2$  are regular, is  $L_1 \cap L_2$  regular as well?



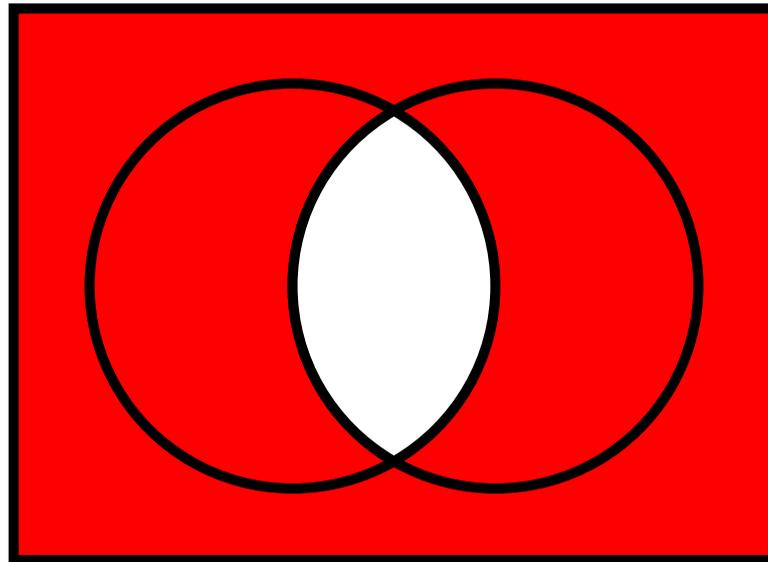
$\bar{L}_1$



$\bar{L}_2$

# Closure Under Intersection

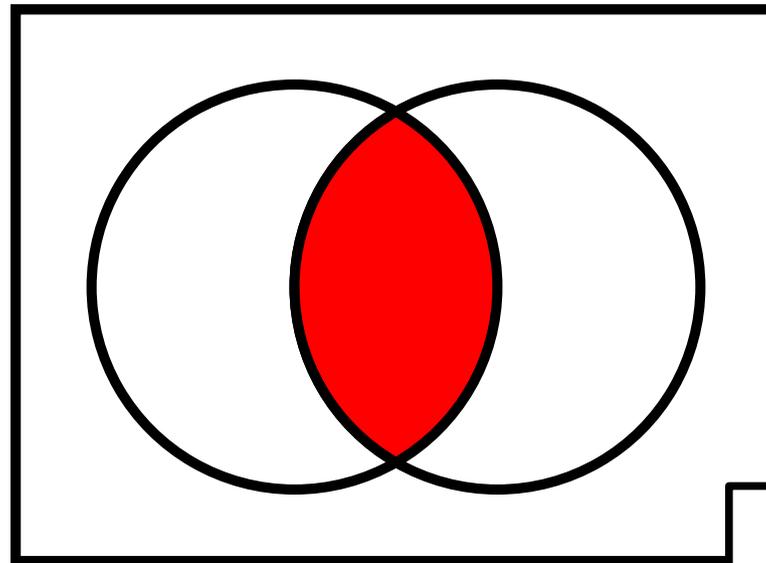
- If  $L_1$  and  $L_2$  are languages over  $\Sigma$ , then  $L_1 \cap L_2$  is the language of strings in both  $L_1$  and  $L_2$ .
- Question: If  $L_1$  and  $L_2$  are regular, is  $L_1 \cap L_2$  regular as well?



$$\overline{L_1} \cup \overline{L_2}$$

# Closure Under Intersection

- If  $L_1$  and  $L_2$  are languages over  $\Sigma$ , then  $L_1 \cap L_2$  is the language of strings in both  $L_1$  and  $L_2$ .
- Question: If  $L_1$  and  $L_2$  are regular, is  $L_1 \cap L_2$  regular as well?



$$\overline{L_1} \cup \overline{L_2}$$

Hey, it's De Morgan's laws!